Université Mohammed Premier
École Nationale des Sciences Appliquées d'Oujda

Cours d' *Informatique 2 : MATLAB*

Version 3.0 (Janvier 2023)

MATLAB POUR L'INGÉNIEUR

STPI-1

ENSAO, 2022 – 2023

Partie 2

Prof. **Kamal GHOUMID**

Université Mohammed Premier
École Nationale des Sciences Appliquées d'Oujda

Cours d' ***Informatique 2 : MATLAB***

Version 3.0 (Janvier 2023)

MATLAB POUR L'INGÉNIEUR

STPI1

Chapitre 5

Programmation MATLAB

Partie 2

Prof. **Kamal GHOUMID**

Vocabulaires (1/3)

- La programmation consiste à combiner des opérations mathématiques, logiques, …. qui agissent sur les données afin de réaliser des tâches spécifiques à l'aide de l'ordinateur (machine électronique capable d'exécuter des opérations arithmétiques et logiques).

- La programmation est un ensemble des activités qui permettent la saisie et l'écriture des programmes informatiques : une façon d'écrire des instructions qui seront ensuite traduites en opérations et fonctions pour l'ordinateur.

- Un logiciel est un ensemble de programmes dédiés à la réalisation de certaines tâches par un ou plusieurs utilisateurs.

- Dans la conception d'un programme, les données essentielles qui vont être traitées "**données d'entrée**" sont développées par la méthode employée "**algorithme**" pour aboutir au résultat "**données de sortie**".

- Les données d'entrée et de sortie peuvent être de natures différentes.

Vocabulaires (2/3)

- Les procédures de la programmation sont basées sur l'algorithmique, de façon à ce qu'on retrouve en général les mêmes fonctionnalités et principes de base.

- Chaque ligne d'un programme effectue soit une opération simple, soit exécute une fonction qui est elle-même une suite d'opérations simples.

- Un **script** (ou **m-fille**) est un fichier d'extension '**.m**' qui regroupe une suite d'instructions. Il peut être exécuté en écrivant son nom dans l'espace de commande ou en cliquant sur '**Run**' ▶.

- Un **script** peut contenir un nombre quelconque de commandes, ainsi que des appels à des fonctions déjà existantes ou écrites par l'utilisateur (voir plus loin 'Macros').

- Un logiciel en informatique peut être vu comme un livre en littérature, écrit en une langue particulière et peut être traduit dans différentes langues (éq. langages de programmation).

Vocabulaires (3/3)

- Un Framework (canevas, socle d'applications, cadre de travail) : sert généralement à simplifier le travail des développeurs informatiques, en leur offrant une architecture prête à l'emploi, qui leur permet de :
  - * ne pas repartir de zéro ;
  - * la réutilisation des codes ;
  - * la standardisation de la programmation ;
  - * la formalisation d'une architecture adaptée aux besoins ;
  - * …

- Matlab offre différents Frameworks (voir toolboxes) qui proposent des fonctionnalités et des opérations très avancées permettant ainsi de réaliser facilement beaucoup de tâches complexes.

- Un Framework est un ensemble d'outils (boite à outils) constituant les fondations d'un logiciel informatique, destiné autant à faciliter le travail (en termes de rapidité, flexibilité, productivité, …).

Caractères et chaînes de caractères (1/4)

- Les chaines de caractères servent à stocker les informations non numériques.

- Les caractères et les chaînes de caractères sont des vecteurs lignes, encadrés par deux quottes ' ……. ' dont la déclaration est identique à un tableau normal.

- 'a' , 'nana' , c = 'c contient une chaine de caractères' .

- Pour MATLAB, les chaînes de caractères et les listes de caractères sont des objets de même nature:

- **Example** :

  Les listes de caractères **['E' 'N' 'S' 'A' 'O']** et **['EN' 'S' 'AO']** sont identiques à la chaîne de caractères **['ENSAO']** :

  >> **['E' 'N' 'S' 'A' 'O']**

  ans =

  ENSAO

Caractères et chaînes de caractères (2/4)

```
>> Ch1 = 'STPI1,ENSAO'
Ch1 =
        STPI1,ENSAO

>> Ch1'
ans =
        S
        T
        P
        I
        1
        ,
        E
        N
        S
        A
        O

>> [n,m] = size(Ch1)
n =
         1

m =
        11

>> length(Ch1)
ans =
   11
```

```
>> abs(Ch1)
ans =
   83   84   80   73   49   44   69   78   83   65   79

>> abs('STPI1,ENSAO')
ans =
   83   84   80   73   49   44   69   78   83   65   79

>> double(Ch1)
ans =
   83   84   80   73   49   44   69   78   83   65   79

>> char([83,84,80,73,49,44,69,78,83,65,79])
ans =
        STPI1,ENSAO

>> bin2dec('1011')
ans =
         11

>> dec2bin(17)
ans =
10001
```

```
>> dec2hex(247)
ans =
          F7

>> X = [137,3455,1097,13354,7863]
X(:) =
         137
        3455
        1097
       13354
        7863

>> dec2hex(X)
ans =
      0089
      0D7F
      0449
      342A
      1EB7

>> Ch1 = 'STPI1,ENSAO'
Ch1 =
          STPI1,ENSAO

>> Ch2 = ['1er année ' Ch1]
Ch2 =
          1er année STPI1,ENSAO
```

```
>> TestEgalite1 = strcmp('ENSAO','ENSAO')
TestEgalite1 =
    1

>> TestEgalite2 = strcmp('ENSAO','ENSAo')
TestEgalite2 =
    0

>> TestEgalite3 = strcmp('ENSAO','ensao')
TestEgalite3 =
    0

>> TestEgalite4 = strcmpi('ENSAO','ensao')
TestEgalite4 =
    1

>> Ch3 = 'ALI';
>> Ch4 = 'ABDELALI';
>> findstr(Ch3, Ch4) %Compare les 2 chaines
ans =
    6

>> Ch4(ans:length(Ch4)) %Vérification
ans =
    ALI
```

```
>> Maj = upper('école nationale des sciences appliquées d''oujda')
Maj =
        ÉCOLE NATIONALE DES SCIENCES APPLIQUÉES D'OUJDA

>> Min = lower('MATLAB')
Min =
        matlab

>> findstr(Ch1,'S')
ans =
        1     9

>> findstr(Ch1,'N')
ans =
        8

>> x = pi/4;
>> Ch5 = 'cos(x)'
Ch5 =
        cos(x)

>> eval(Ch5)
ans =
        0.7071

>> P = poly2str([3,-5,0,7],'x')
P =
        3 x^3 - 5 x^2 + 7
```

Formats d'affichage et de lecture (1/11)

Affichage simple ‘disp’:

- La commande **disp** permet d'afficher un tableau de valeurs numériques ou de caractères, elle affiche un message à l’écran.
- L’instruction **disp** permet l’affichage de variables de toutes natures (scalaires, matrices, textes, …)
- La commande **disp** peut être utilisée pour afficher un résultat.
- Dans son utilisation, l’instruction **disp** s’accommode d’afficher le résultat sans écrire le nom de la variable.
- Pour utiliser les valeurs numériques avec l’instruction **disp**, on a recours à la commande **num2str** pour les convertir en une chaîne de caractères.

Affichage simple 'disp':

- Exemples:

```
>> a=2017;
>> disp(a)
        2017

>> disp([5 ; -41.87 ; sqrt(2)])
        5.0000
       -41.8700
        1.4159

>> d=16;
>> disp([13   sqrt(d)   log10(a+84)])
    13   4   2
```

```
>> b='Bonjour';
>> disp(b)
        Bonjour

>> c=[7   22   -4];
>> disp(c)
        7  22  -4

>> E=[1  -9; 4  2];
>> disp(E)
        1   -9
        4    2
```

Affichage simple '**disp**':

- Exemples:

```
>> disp(['Première'  blanks(1)  'année '  blanks(1) ' ENSAO'])
        Première année ENSAO

>> disp(['En première année vous êtes '  num2str(180)  ' étudiants'])
        En première année vous êtes 180 étudiants

>> disp(['Le cours de Matlab est programmé à '  num2str(8)  ' heures '  num2str(15)])
        Le cours de Matlab est programmé à 8 heures 15
```

* " **blanks(n)** " : affiche *n* espaces.
* " **num2str** " : convertit un nombre en une chaîne de caractères.

Lecture ‘input’:

- La commande **iuput** permet de demander à l'utilisateur de fournir des données ou d’entrer les valeurs de variables à utiliser (saisie de valeurs considérées ou de caractères depuis le clavier).

- L’instruction **input** est utilisée pour questionner l’utilisateur du programme, puis attendre une réponse dactylographiée. La réponse tapée est ensuite renvoyée comme résultat de la commande et doit généralement être assignée à une variable.

- La commande **input** effectue la saisie de valeurs numériques ou bien de textes.

- Pour la saisie de textes, **on doit mettre ‘s’**.

- Pour la saisie de valeurs numériques, **il ne faut pas mettre ‘s’**.

Lecture 'input':

- Exemples :

```
>> A = input('Entrer un chiffre :  ');
    Entrer un chiffre :  78756

>> disp('A')
        A

>> disp(['A'])
        A

>> disp([num2str(A)])
        78756

>> disp(['Le chiffre entré est :  ',num2str(A)])
        Le chiffre entré est :  78756
```

```
>> B = input('Tu es dans quel établissement :  ','s');
Tu es dans quel établissement :  ENSAO

>> disp('B')
      B

>> disp(B)
      ENSAO
```

Lecture 'input':

- Exemples:

```
>> % CoordonnéesPersonnelles.m
disp('=============== Bonjour ===============');
nom=input('Quel est ton nom de famille :  ','s');
prenom=input('Quel est ton prénom :  ','s');
age=input('Quel âge as-tu :  ');
disp(['Ton nom de famille est : ' nom ','  blanks(2) 'ton prénom est : ' prenom]);
disp(['Tu as ' num2str(age) ' ans']);
```

```
---------- Après exécution ----------
=============== Bonjour ===============
Quel est ton nom de famille :  TOTO
Quel est ton prénom :  titi
Quel âge as-tu :  18
Ton nom de famille est : TOTO,  ton prénom est : titi
Tu as 18 ans
```

Impression à l'écran : fprintf

- **fprintf** est une commande d'écriture dans la fenêtre d'exécution. En général, elle a la structure suivante :

  fprintf(*format*,*var1*,*var2*,…)

  Où *format* est une chaine de caractère décrivant le format d'écriture des variables *var1*, *var2*, … qu'on les souhaite afficher.

- Les principaux types de formats d'écritures sont:

  %d : pour un entier.

  %i : pour un entier.

  %f : pour un réel.

  %e : pour exponentiel

  %s : pour une chaine de caractère.

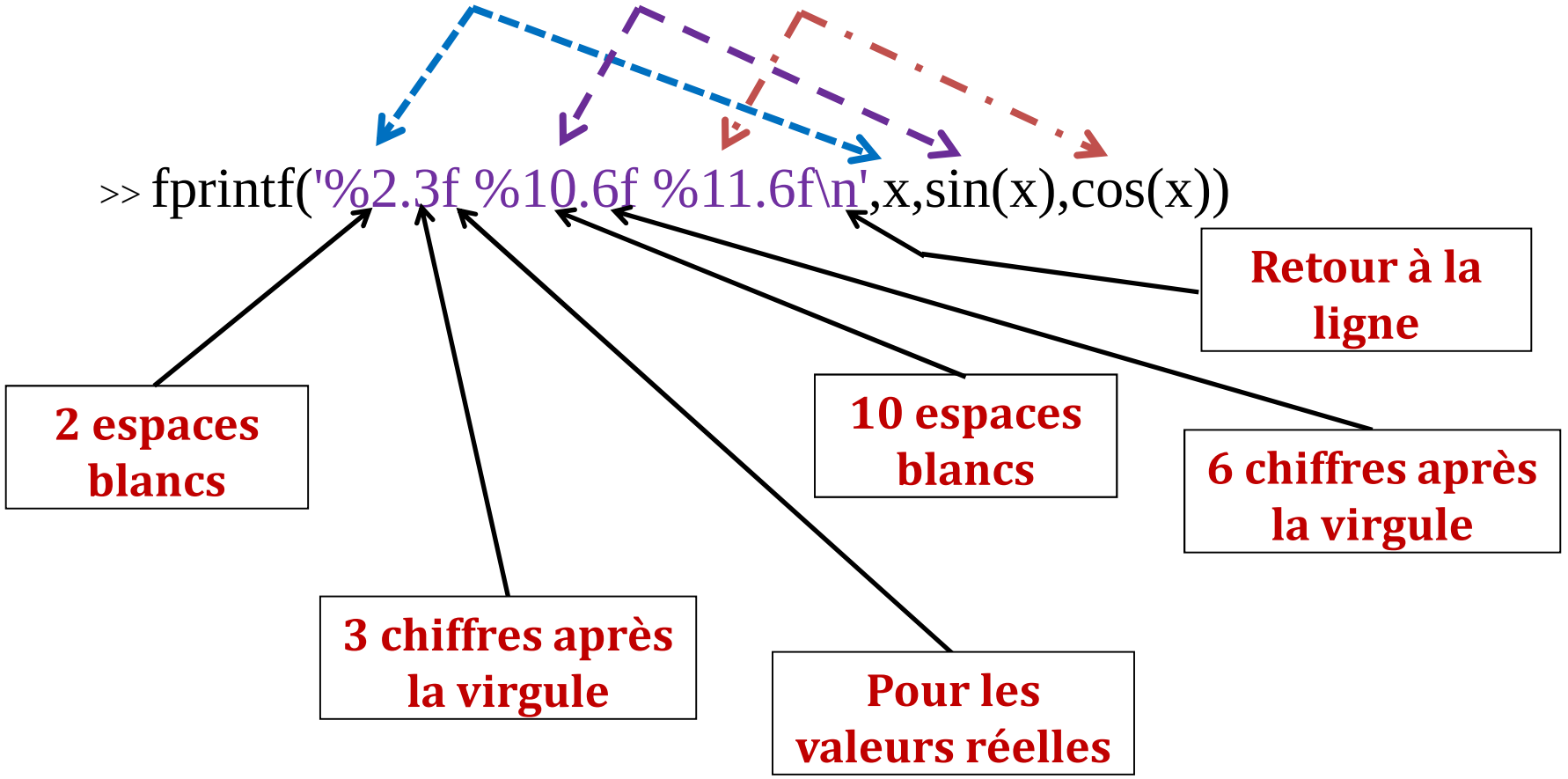
Impression à l'écran : fprintf

- Exemples:

```
>> A = [4.42   13.78   5.16   -7.63];
>> fprintf('%d\n', round(A)); %round renvoie la partie entière la plus proche, \n pour revenir à la ligne
        4
        14
        5
        -8

>> n = 34.8752293472;
>> disp([num2str(n)]);
    34.8752

>> fprintf('%0.4f ',n);
     34.8752

>> fprintf('%0.10f ',n);
     34.8752293472
```

Impression à l'écran : fprintf

```
>> x = pi/3;
>> fprintf('%2.3f  %10.6f  %11.6f\n',x,sin(x),cos(x));
    1.047      0.866025       0.500000
```



Exemples : fprintf

```
>> A = 12;
>> B = 4;
>> C = 2.145;
>> fprintf('%3d x %1d + %1.3f = %2.3f ',A,B,C,A*B+C);
   12 x 4 + 2.145 = 50.145

%Une autre façon pour écrire les choses
>> fprintf('%3d %1s %1d %1s %1.3f %1s %2.3f ',A,'x',B,'+',C,'=',A*B+C);
   12 x 4 + 2.145 = 50.145

>> fprintf('%3s %1.3f ','A - B - C = ',A-B-C);
    A - B - C =  5.855

>> fprintf('%3d %s %1d %1s %1.3f %1s %1.3f ',A,'-',B,'-',C,'=',A-B-C);
    12 - 4 - 2.145 = 5.855
```

Instruction : sprintf

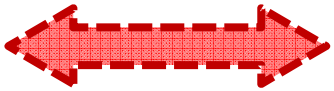
- **sprintf** est une commande d'écriture qui permet l'impression de variables selon un modèle donné. En général, elle a la structure suivante :

  sprintf(*format*,*variables*)

- **Example**

```
sprintf('%s','Cours Matlab pour l''ingénieur')
ans =
Cours Matlab pour l'ingénieur

>> x = pi/4;
>> y = cos(x);
>> sprintf('sin(%2.6f) = %1.2f',x,y)
ans =
sin(0.785398) = 0.71
```

```
>> y = sprintf('STPI1\n');
>> disp(y)
STPI1
```



```
>> x = fprintf('STPI1\n');
STPI1
>> disp(x)
     6
```

L'instruction sprintf créee une variable chaîne.
L'instruction fprintf écrit dans un fichier ou dans la fenêtre de commande.

Principales syntaxes de programmation algorithmiques utilisables sous Matlab:

* **Structures conditionnelles** :

  - La structure **if**

  - La structure **switch**

* **Structures itératives** :

  - les boucles de type **for**

  - les boucles de type **while**

Instruction conditionnelle if … end

- La structure conditionnelle **if** est basée sur l'évaluation d'une condition parmi un choix multiple, et le traitement est exécuté une fois la condition logique est satisfaite.

- La structure conditionnelle **if** est indispensable, elle est présente dans tous les langages de programmation, mais elle utilise des syntaxes qui diffèrent un peu selon chaque langage.

- En langage Matlab, elle est déclarée de la manière suivante:

```
if   condition logique
                Instructions
elseif   condition logique
                Instructions
elseif   condition logique
                Instructions
        ...
else
                instructions
end
```

Instruction conditionnelle if … end

- **if ... end** est obligatoire, par contre **else** et **elseif** sont facultatifs , elles permettent d'effectuer des tests supplémentaires.

*** Exemple** :

HeureJour.m

```
heure = input('Quelle heure est-il?  :    ');
if  (heure >= 0)  &&  (heure <= 12)
        disp('On est le matin !');
elseif  (heure > 12)  &&  (heure <= 24)
        disp('On est l''après-midi !');
else
        disp('Ce n''est pas possible...');
end
```

Instruction conditionnelle if

- Exemple :

Comparaison.m

```
a = input(' Saisir un nombre  a = ' );
b = input(' Saisir un nombre  b = ' );
if   a < b
        disp(['Le nombre  ', num2str(a) , '  est inférieur à    ', num2str(b)]);
elseif  a > b
        disp(['Le nombre  ', num2str(a) , '  est supérieur à  ', num2str(b)]);
else
        disp(['Le nombre  ', num2str(a) , '  est égale à  ', num2str(b)]);
end
```

Instruction conditionnelle if

- Exemple:

MoyenneBac.m

```
Moy = input('Combien vous avez obtenu comme moyenne l''année dernière :   ');
if  (Moy >= 0)  &&  (Moy < 10)
                    disp('------- Ajourné(e) --------')
    elseif  (Moy >= 10)  &&  (Moy < 12)
                    disp('-------- Mention : Passable ---------')
    elseif  (Moy >= 12)  &&  (Moy < 14)
                    disp('-------- Mention : Assez-Bien ---------')
    elseif  (Moy >= 14)  &&  (Moy < 16)
                    disp('-------- Mention : Bien ---------')
    elseif  (Moy >= 16)  &&  (Moy < 20)
                    disp('-------- Mention : Très-Bien ---------')
    else
                    disp('Erreur, la moyenne entrée n''est pas valide')
end
```

Instruction conditionnelle switch

- **switch** est une structure conditionnelle, c'est-à-dire qu'elle comporte différents blocs d'instructions qui seront exécutés de manière conditionnelle (choix multiple).

- Pas de conditions logiques, le critère de choix est la valeur d'une expression (ou d'une variable). Chaque **case** (cas), permet la sélection du bloc à exécuter.

- La deuxième structure permet le choix entre différents cas. Le seul test effectué dans cette structure est donc un test d'égalité.

```
switch
        case
        case
        otherwise
end
```

Instruction conditionnelle switch

- Exemple:

Divisionpar3.m

```
n = input('Entrer un nombre entier  n :  ');
switch   mod(n,3) % reste de la division de n par 3, on peut utiliser l'instruction rem(n,3)
   case 0,
        disp(['Le numéro ',num2str(n),' est un multiple de 3']);
   case 1,
        disp(['Le reste de la division de ',num2str(n),' par 3 est égale à 1']);
   case 2,
        disp(['Le reste de la division de ',num2str(n),' par 3 est égale à 2']);
   otherwise
        disp(['Le nombre ', num2str(n) , ' n'est pas un entier, ce n'est pas ce qui est demandé']);
end
```

Instruction conditionnelle switch

- Exemple:

VenteBanane.m

```
disp('*=*=*=*Grandes Surfaces, Magasin Carrefour au rayon des légumes *=*=*=*');
disp('          ---------- Le prix d''un kilo de banane est 10 dh ----------          ');
x = input('Vous voulez acheter combien de kilos? :  ');
switch  x
    case {1,2}
            disp('Le prix d'un kilo est 10 dh');
            fprintf('Le prix total à payer en casse est :  %1.2f  %1s',x*10, 'dh');
    case {3,4,5}
            disp('Le prix par kilo devient 9.5 dh');
            fprintf('Le prix total à payer en casse est :  %1.2f  %1s',x*9.5, 'dh');
    case {6,7,8,9,10}
            disp('Le prix par kilo devient 9 dh');
            fprintf('Le prix total à payer en casse est :  %1.2f  %1s',x*9, 'dh');
    otherwise
            disp('À partir de 11 kilos, le prix devient 8.5 dh/kilo');
            fprintf('Le prix total à payer en casse est :  %1.2f  %1s',x*8.5, 'dh');
end
```

Instruction conditionnelle switch

OperationsArithmetiques.m

- Exemple:

```
disp('############---------- Opérations arithmétiques sur deux nombres ----------
############');
fprintf('\n');
a = input('Veillez saisir un nombre réel  a =  ');
b = input('Veillez saisir un nombre réel  b =  ');
c = input('Quelle opération voulez-vous effectuer  " + " , " - " , " * " , " / " , " ^ " ?   ','s');
switch  c
   case  '+'
      Sm = a + b;
      disp([num2str(a) ' + ' num2str(b) ' = ' num2str(Sm)]); %Affichage avec l'instruction disp
   case  '-'
      Ss = a - b;
      disp([num2str(a) ' - ' num2str(b) ' = ' num2str(Ss)]);
   case  '*'
      M = a * b;
      disp([num2str(a) ' * ' num2str(b) ' = ' num2str(M)]);
   case  '/'
      D = a / b;
      fprintf('%2.4f %2s %2.4f %2s %2.4f',a,'/',b,'=',D); %Affichage avec l'instruction fprintf
   case  '^'
      P = a^b;
      fprintf('%2.4f %2s %2.4f %2s %2.4f',a,'^',b,'=',P);
   otherwise
      disp('L''opération demandée est non reconnue');
end
```

Structure itératives switch

JoursSemaine.m

- Exemple

```
disp('############---------- Numéro des jours de la semaine ----------############');
fprintf('\n');
NumJour = input('Saisir un numéro du jour de la semaine (compris entre 1 et 7)  n  =  ');
switch  NumJour
   case  1
      disp(['Le jour numéro ',num2str(NumJour),' de la semaine est le lundi']);
      disp(['Encore ',num2str(5-NumJour),' jours avant le week end']);
   case  2
      disp(['Le jour numéro ',num2str(NumJour),' de la semaine est le mardi']);
      disp(['Encore ',num2str(5-NumJour),' jours avant le week end']);
   case  3
      disp(['Le jour numéro ',num2str(NumJour),' de la semaine est le mercredi']);
      disp(['Encore ',num2str(5-NumJour),' jours avant le week end']);
   case  4
      disp(['Le jour numéro ',num2str(NumJour),' de la semaine est le jeudi']);
      disp(['Encore ',num2str(5-NumJour),' jours avant le week end']);
   case  5
      disp(['Le jour numéro ',num2str(NumJour),' de la semaine est le vendredi']);
      disp(['En plus il reste juste un jour pour le week-end']);
   case  6
      disp(['Le jour numéro ',num2str(NumJour),' de la semaine est le samedi']);
      disp(['C''est le week end en plus']);
   case  7
      disp(['Le jour numéro ',num2str(NumJour),' de la semaine est le dimanche']);
      disp(['C''est le week end en plus']);
   otherwise
      disp('  "Erreur", il faut donner un nombre entier compris entre 1 et 7');
end
```

Exemple récapitulatif

CreneauxSalleTD.m

```
disp('          ');
disp('=============================================================');
disp('  ---- Exemples récapitulatif sur les utilisations de "disp", "input", "if ... end", "switch ... end"  ----- ');
disp('=============================================================');
disp('          ');
disp('En 1er année vous êtes 120 étudiants.');
disp('Ce petit code vous aide à connaître vos créneaux et salles de TD d''Info2 :');
disp('          ');
EtudiantCP1 = input('Vous êtes étudiants en STPI1 (Oui ou Non?) :   ','s');
switch  EtudiantCP1
   case  'Non'
      disp('Ces affectations de groupes et de salles ne concernent que les étudiants CP1, Excusez-nous
SVP.');
   case  'Oui'
         NumCP1 = input('Quel est votre numéro CP1-ENSAO? :  ');
      if  NumCP1 >= 1 && NumCP1 <= 30
            disp(['L''étudiant dont le numéro CP1-ENSAO est ' num2str(NumCP1) ' appartient au groupe A, il
a le TD d''Info 2 le Lundi après-midi, Salle AE1.']);
      elseif  NumCP1 >= 31 && NumCP1 <= 60
            disp(['L''étudiant dont le numéro CP1-ENSAO est ' num2str(NumCP1) ' appartient au groupe B, il
a le TD d''Info 2 le Mardi matin, Salle AE2.']);
```

Exemple récapitulatif

CreneauxSalleTD.m

----- La suite du programme ------

```
        elseif  NumCP1 >= 61 && NumCP1 <= 90
          disp(['L''étudiant dont le numéro CP1-ENSAO est ' num2str(NumCP1) ' appartient au groupe C, il a le TD d''Info 2 le Mercredi après-midi, Salle AE3.']);
        elseif  NumCP1 >= 91 && NumCP1 <= 120
          disp(['L''étudiant dont le numéro CP1-ENSAO est ' num2str(NumCP1) ' appartient au groupe D, il a le TD d''Info 2 le Jeudi matin, Salle AE4.']);
        else
          disp('Le numéro CP1-ENSAO doit être compris entre 1 et 120.');
      end
   otherwise
      disp('Votre réponse n''est pas valide, Vous devriez répondre par«   Oui'' ou ''Non'' seulement.');
end
```

Structure itératives for

- Une boucle **for** permet l'exécution d'un certain nombre de fois un même bloc d'instructions (itère le même traitement plusieurs fois selon le compteur initialisé).

- La boucle **for** est une structure itérative à éviter autant que possible sous Matlab, car elle est très coûteuse en temps de calcul par rapport au calcul matriciel.

- La boucle **for** permet d'exécuter une séquence d'instructions répétitive dans une boucle pour les valeurs d'un indice incrémenté à chaque itération. L'ensemble des valeurs pour lesquelles le bloc est effectué est un ensemble fini, déclaré en début de structure.

- La syntaxe de la boucle **for** est :

```
for Compteur = vecteur des valeurs (bloc d'ensembles des valeurs)
        Instructions
end
```

Structure itératives for

- Exemples

```
for  s = 0 : sqrt(2) : 10;
   disp(s)
end
```

```
0
1.4142
2.8284
4.2426
5.6569
7.0711
8.4853
9.8995
```

```
for  i = 1 : 4
    for  j = 1 : 6
        A(i,j) = i+j;
    end
end
```

```
disp(A);
    2    3    4    5    6    7
    3    4    5    6    7    8
    4    5    6    7    8    9
    5    6    7    8    9   10
```

Structure itératives for

- Exemple

TableauSinCos.m

```
disp(['x' blanks(10) 'sin(x)' blanks(10) 'cos(x)' blanks(10) 'sin(x)^2+cos^2(x)']);
for  x = 1 : 1 : 5
   disp('[num2str(x)  blanks(10)  num2str(sin(x))  blanks(10)  num2str(cos(x))  blanks(17)  ...
   num2str(sin(x)^2 + cos(x)^2)]');
end
```

---------- Après exécution ----------

x	sin(x)	cos(x)	sin(x)^2+cos^2(x)
1	0.84147	0.5403	1
2	0.9093	-0.41615	1
3	0.14112	-0.98999	1
4	-0.7568	-0.65364	1
5	-0.95892	0.28366	1

Structure itératives for

DevTaylorSin.m

```
fprintf([' x' blanks(18) 'sin(x)' blanks(13) 'DévLimité sin(x)\n']);
for x = 0:pi/48:pi/12
   DLsin = x - x^3/6 + x^5/120;
   fprintf('%2.3f  %15.6f  %17.6f\n',x,sin(x),DLsin);
end
```

---------- Après exécution ----------

x	sin(x)	DévLimité sin(x)
0.000	0.000000	0.000000
0.065	0.065403	0.065403
0.131	0.130526	0.130526
0.196	0.195090	0.195090
0.262	0.258819	0.258819

Structure itératives for

```
disp(['T en ° Celsius'  blanks(13)  'T en Kelvin']);
for T = 0:5:25
        K = T + 273.5;
        fprintf('%12i %34.2f\n',T,K);
end
```

ConversionCel2Kel

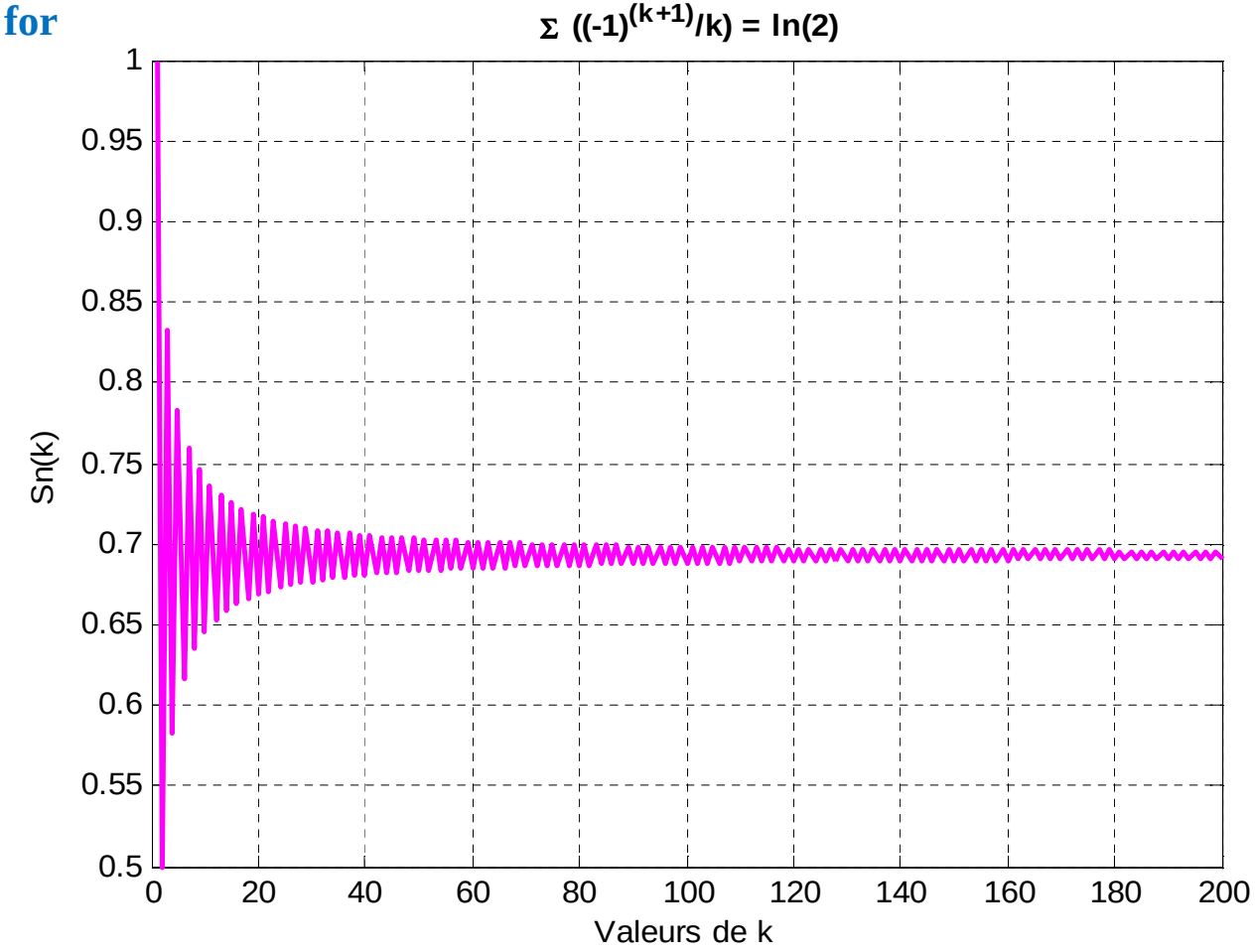
---------- Après exécution ----------

T en ° Celsius	T en Kelvin
0	273.50
5	278.50
10	283.50
15	288.50
20	293.50
25	298.50

Structure itératives for

LimiteSerie1.m

```
Sn = 0;
for  k = 1 : 200
        Sn = Sn + (-1)^(k+1)/k;
        x(k) = k;
        y(k) = Sn;
end
plot(x,y,'m','linewidth',2);
grid on;
xlabel('Valeurs de k');
ylabel('Sn(k)');
title('\bf \Sigma ((-1)^{(k+1)}/k) = ln(2)');
```



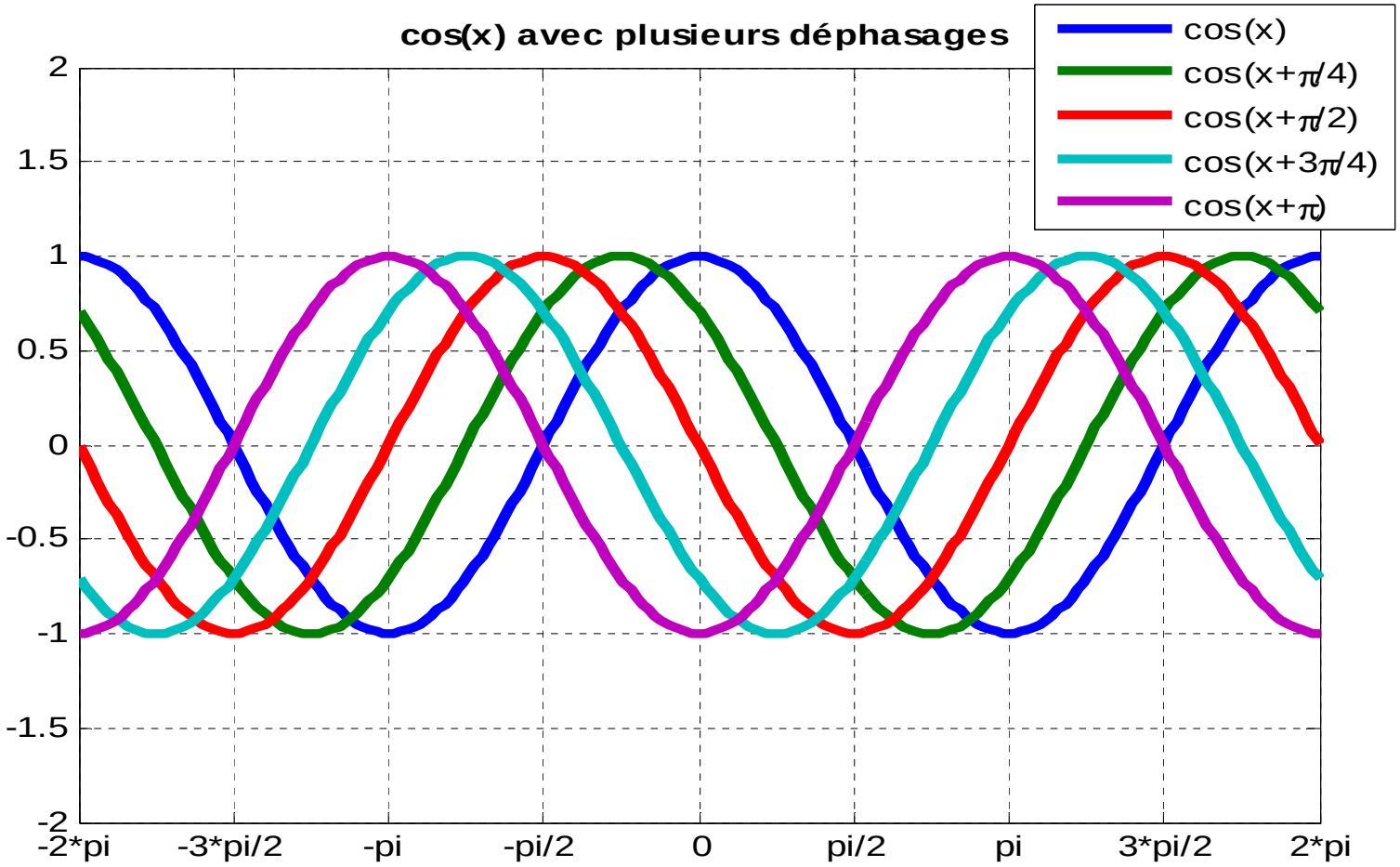
$$\sum_{k=1}^{+\infty} \frac{(-1)^{k+1}}{k} = \ln(2)$$

Structure itératives for

AlluresCosDephases.m

```
x = -2*pi : pi/50 : 2*pi;
m = 1;
tic;
for  n = [0,pi/4,pi/2,3*pi/4,pi]
     y(m,:) = cos(x + n);
     m = m + 1;
end
Temps=toc;
plot(x,y,'linewidth',3);
axis([-2*pi 2*pi -2 2]);
set(gca,'XTick',-2*pi:pi/2:2*pi);
set(gca,'XTickLabel',{'-2*pi','-3*pi/2','-pi','-pi/2','0','pi/2','pi','3*pi/2','2*pi'});
title('\bf cos(x) avec plusieurs déphasages');
legend('cos(x)','cos(x+\pi/4)','cos(x+\pi/2)','cos(x+3\pi/4)','cos(x+\pi)');
grid on
fprintf('Cette boucle a pris un temps égale à %1.4f seconde',Temps);
```

Structure itératives for



>> Cette boucle a pris un temps égale à 0.0023 seconde

Structure itératives for

TransposeMatrice.m

```
clear all;
close all;
clc;
A = input('Entrer les coefficients de la matrice A  =  ') ;
[n,m] = size(A) ;
for  i = 1 : n
   for   j = 1 : m
      B(j,i) = A(i,j) ;
   end
end
disp('La matrice  A est :  ');
disp(A);
disp('La matrice B transposée de A est :  ‘);
disp(B);
```

Structure itératives for

---------- Après exécution du code ----------

Entrer les coefficients de la matrice A = [1,3,-2;2,9,3;2,8,-1]

La matrice A est :

1 3 -2

2 9 3

2 8 -1

TransposeMatrice.m

La matrice B transposée de A est :

1 2 2

3 9 8

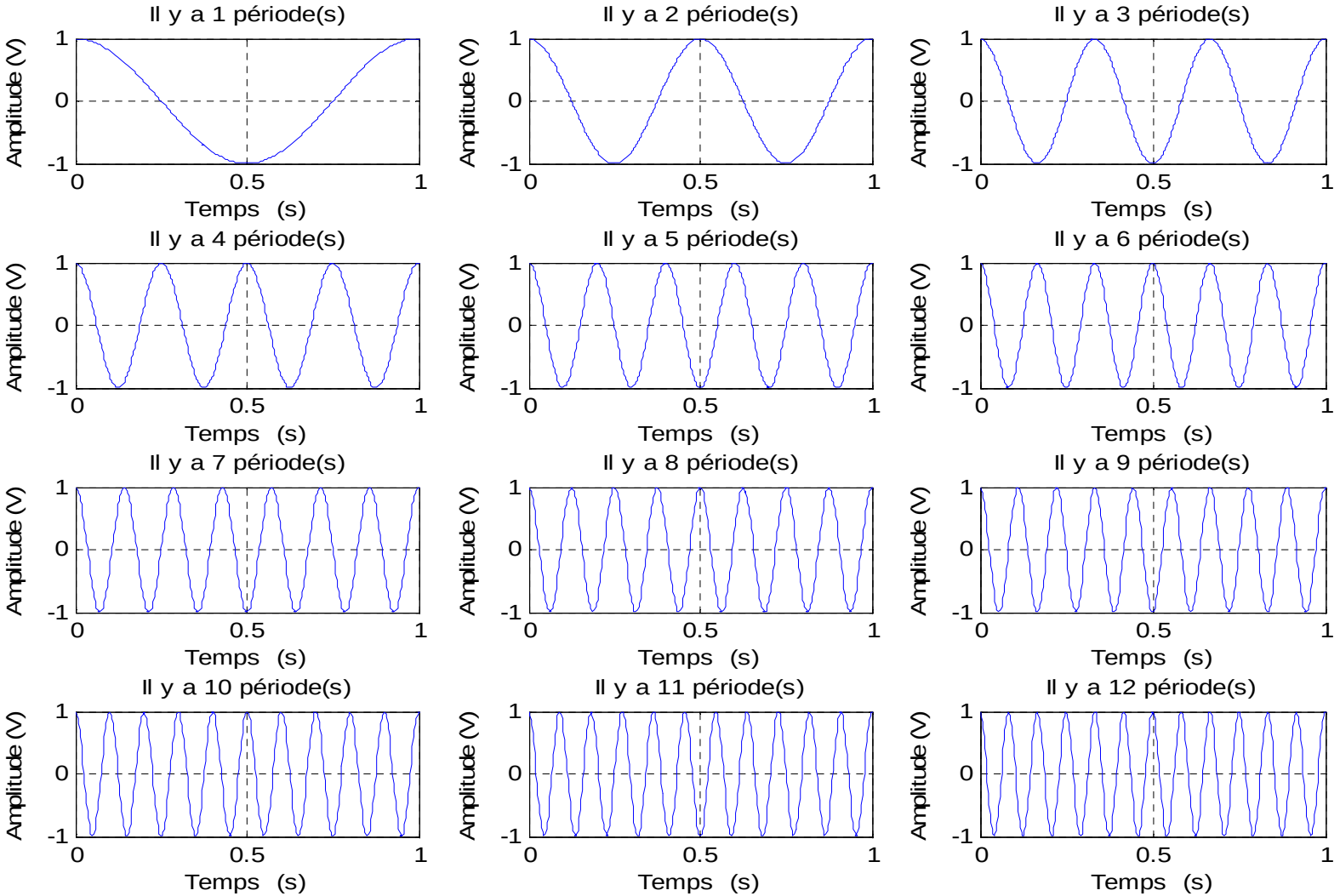
-2 3 -1

Structure itératives for

PlusieursCosPause.m

```
clear all
close all
clc
disp('#########---------- Visualisation de plusieurs courbes avec pause ----------
########');
f = 1; %Fréquence égale à 1 Hz
t = linspace(0,1,400);
for n = 1:12
    S_n = cos(2*pi*f*n*t); %Signal à tracer
    subplot(3,4,n);
    plot(t,S_n);
    grid on
    title(['Il y a ',num2str(n),' période(s)']);
    pause(2) %Une pause de 2 secondes
end
```

Structure itératives for



Structure itératives for (avec la boucle if)

```
clear all
close all
clc
disp('############---------- Recherche des diviseurs d''un nombre ----------#############');
fprintf('\n');
nbre = input('Veillez saisir un nombre entier positif  =   ');
x = nbre - round(nbre);

if  nbre < 0
      disp('Le nombre saisi n''est pas un entier positif');
elseif  nbre == 0
      disp('Le nombre saisi est nul, il n'' a pas de diviseurs');
elseif  nbre - round(nbre) ~= 0
      disp('Le nombre saisi n''est pas un entier positif');
else
      disp(['Les diviseurs du nombre saisi ',num2str(nbre),' sont : ']);
   for  i = 1:nbre
      if  mod(nbre,i) == 0
         disp(i);
      end
   end
end
```

RecherchePlusieursDiviseurs.m

Structure itératives for (avec la boucle if)

Exemples d'exécutions du code

```
############---------- Recherche des diviseurs d'un nombre ----------############
Veillez saisir un nombre entier positif  =  -42
Le nombre saisi n'est pas un entier positif


############---------- Recherche des diviseurs d'un nombre ----------############
Veillez saisir un nombre entier positif  =  0
Le nombre saisi est nul, il n'pas pas de diviseurs


############---------- Recherche des diviseurs d'un nombre ----------############
Veillez saisir un nombre entier positif  =  86.3
Le nombre saisi n'est pas un entier positif


############---------- Recherche des diviseurs d'un nombre ----------############
Veillez saisir un nombre entier positif  =  32
Les diviseurs du nombre saisi 32 sont :
   1
   2
   4
   8
  16
  32
```

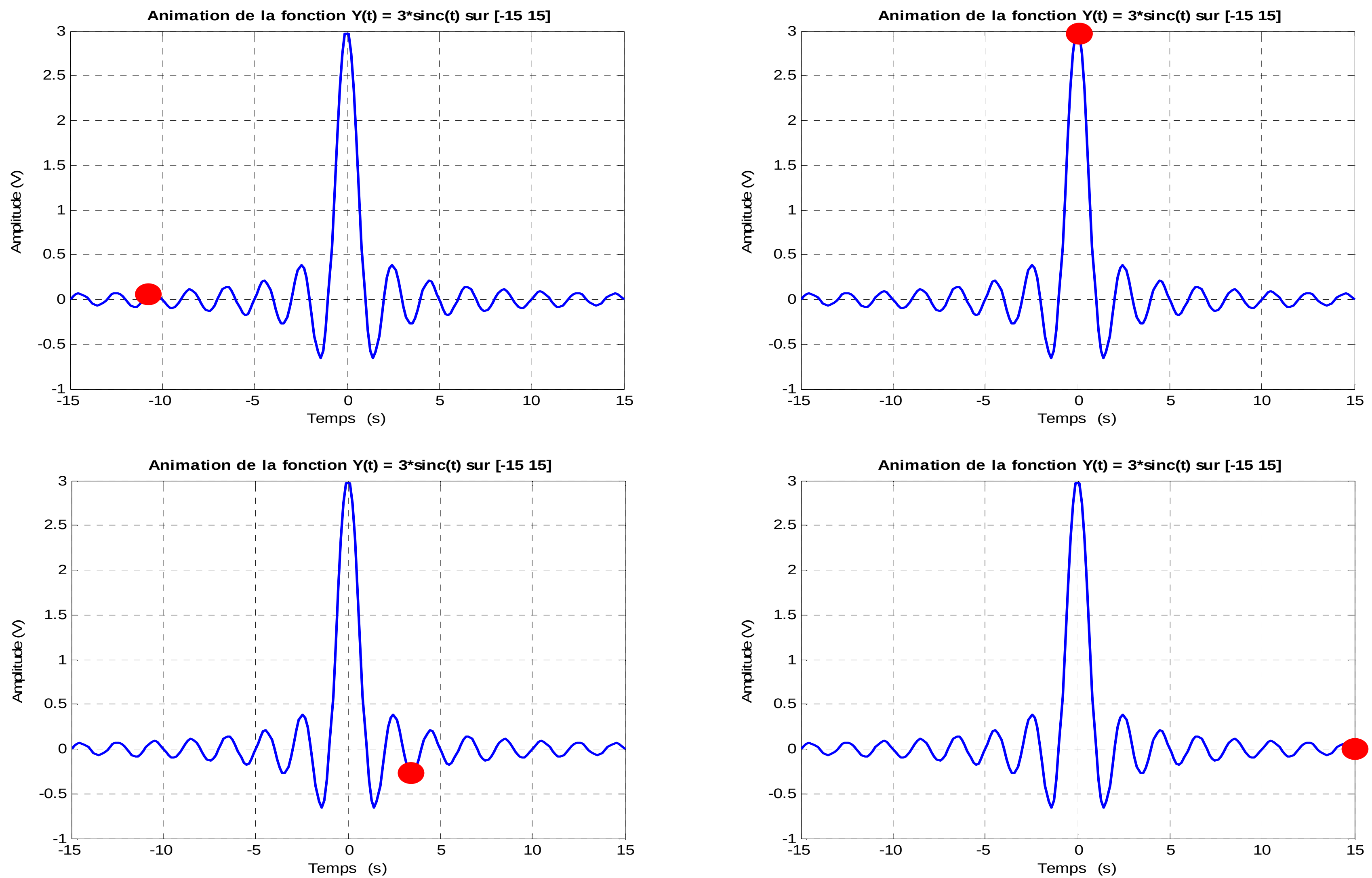
Structure itératives for (avec animation)

```
clear all
close all
clc
disp('##########--------- Visualisation de l''animation d''une allure de courbe --------##########');
VitesseAnimation = input('Saisir la vitesse de l''animation (de préférence < 1) =  ');
t = linspace(-15,15,200);
Y = 3*sinc(t);
plot(t,Y,'LineWidth',2);
grid on
xlabel('Temps  (s)');
ylabel('Amplitude (V)');
title('\bf Animation de la fonction Y(t) = 3*sinc(t) sur [-15 15]');
hold on
h = plot(t(1),Y(1),'ro','MarkerFaceColor','r','LineWidth',5);

for  n = 2:length(t)
    set(h,'XData',t(n));
    set(h,'YData',Y(n));
    drawnow;
    pause(VitesseAnimation);
end
```

AnimationCourbeSinc.m

AnimationCourbeSinc.m

Structure itératives for (avec animation)



Structure itératives while

- La boucle **while** est similaire à la boucle for, elle permet à un groupe d'instructions d'être exécuté un certain nombre de fois, mais contrôlé par un prédicat qui veille à ce que la condition de maintien de la boucle ne soit pas violée.

- La boucle **while** continue à exécuter un bloc d'instructions tant que la condition logique est vraie. Ce test est aussi appelé condition d'arrêt.

- La variable de test doit être actualisée pendant l'exécution de la boucle afin que celle-ci s'arrête

- La syntaxe de la boucle while est :

```
while   Condition
        Instructions
end
```

Structure itératives while

- Exemple

```
i = 0;
a = 0;
while (i < 7)
    a = a + sum(i^2)
    i = i + 1;
end
```

```
a =
     0

a =
     1

a =
     5

a =
    14

a =
    30

a =
    55

a =
    91
```

Structure itératives while

- Exemple

CalculAffichageSomme1.m

```
disp(['Valeur de n', blanks(12) ,'Valeur de S']);
n = 1;
S = 0;
while  (S < 5*1e-2)
    S = S + sind(n/pi)/n;
    n = n + 1;
    fprintf('%11d %31.4f\n',n-1,S);
end
```

---------- Après exécution du code ----------

Valeur de n	Valeur de S
1	0.0056
2	0.0111
3	0.0167
4	0.0222
5	0.0278
6	0.0333
7	0.0389
8	0.0444
9	0.0500
10	0.0555

Structure itératives while

- Exemple

ChoixAleatoireCondition.m

```
disp('###----- Choix de nombre aléatoire -----###');
NbreAleatoire = input('Saisir un numéro entre 1 et 20  =  ');
x = 1;
I = 0;
while  x ~= NbreAleatoire
            pause(0.5);
            x = randi(NbreAleatoire,1);
             disp(x);
            i = i + 1;
end
```

Structure itératives while

- Exemple

ChoixAleatoireCondition.m

---------- Après exécution du code ----------

```
###---- Choix de nombre aléatoire ----###
Saisir un numéro entre 1 et 20  =  17
     9
    14
     3
     8
    16
    14
    17
```

```
###---- Choix de nombre aléatoire ----###
Saisir un numéro entre 1 et 20  =  8
     6
    13
    19
     8
```

Structures itératives while + if

- **Exemple** : Écriture d'un m-File qui trouve le plus petit entier pair divisible par 13 et par 16 dont la racine carrée est supérieure à 120.

RechercheNombreConditions.m

```
clear all;
clc;
i=0;
s=0;
while   s <= 120
        i=i+1;
    if  rem(i,2) == 0  &&  rem(i,13) == 0  &&  rem(i,16) == 0
        s = sqrt(i);
    end
end
fprintf('Le numéro requis est : %i\n',i);
fprintf('\n');
disp('       ---------- Pour la vérification ----------');
disp(['Le reste de la division de ',num2str(i),' par 13 est :  ',num2str(mod(i,13))]);
disp(['Le reste de la division de ',num2str(i),' par 16 est :  ',num2str(mod(i,16))]);
disp(['La racine carrée sqrt(',num2str(i),') = ',num2str(sqrt(i)),' est bien supérieur à 120']);
```

Structures itératives while + if

- **Exemple** : Écriture d'un m-File qui trouve le plus petit entier pair divisible par 13 et par 16 dont la racine carrée est supérieure à 120.

RechercheNombreConditions.m

---------- Après exécution du code ----------

```
##############################################################
Le numéro requis est : 14560


---------- Pour la vérification ----------
Le reste de la division de 14560 par 13 est :  0
Le reste de la division de 14560 par 16 est :  0
La racine carrée sqrt(14560) = 120.6648 est bien supérieur à 120
##############################################################
```

Exemple : Jeu de devinette

JeuDichotomie.m

```
%%%------- Jouer à la devinette en faisant de la dichotomie ------%%%
clear all;
close all;
clc;
disp('#############-----------**********----------#############');
disp('     Jeu de la devinette en faisant de la dichotomie      ');
disp(' ');
n = input('Dans ce jeu, le nombre à deviner est compris entre la valeur 0 et la valeur =  ');
disp(['Le but du jeu est de trouver le nombre caché choisi par le logiciel dans l''intervalle
[0,',num2str(n),']']);
disp('C''est parti, tu peux commencer à jouer');
disp('                                  ');
k = randi(n); %randi(n) renvoie un nombre entier aléatoirement tiré entre 1 et n
a = 2*k-3;
while  a~=k
   a=input('Entrez un nombre : ');
 if  a>k
   disp('Le nombre est plus grand');
 elseif  a<k
   disp('Le nombre est plus petit');
 end
end
disp('         Bravo, tu as gagné');
disp('                                  ');
disp(['   Le nombre k qui a été choisi aléatoirement est égale à ',num2str(a)]);
disp('#############---------------------------- Fin ----------------------------#############');
```

Exemple : Jeu de devinette

* **Exemple d'exécution**

```
############-----------**********----------############
     Jeu de la devinette en faisant de la dichotomie

Dans ce jeu, le nombre a deviné est compris entre la valeur 0 et la valeur =  20
Le but du jeu est de trouver le nombre caché choisi par le logiciel dans l'intervalle [0,20]
C'est parti, tu peux commencer à jouer

Entrez un nombre : 3
Le nombre est plus petit
Entrez un nombre : 17
Le nombre est plus grand
Entrez un nombre : 13

         Bravo, tu as gagné

  Le nombre k qui a été choisi aléatoirement est égale à 13
############---------------------------- Fin ----------------------------############
```

Exemple : Circuit RC, Tension aux bornes du condensateur

CircuitRC_Tension.m

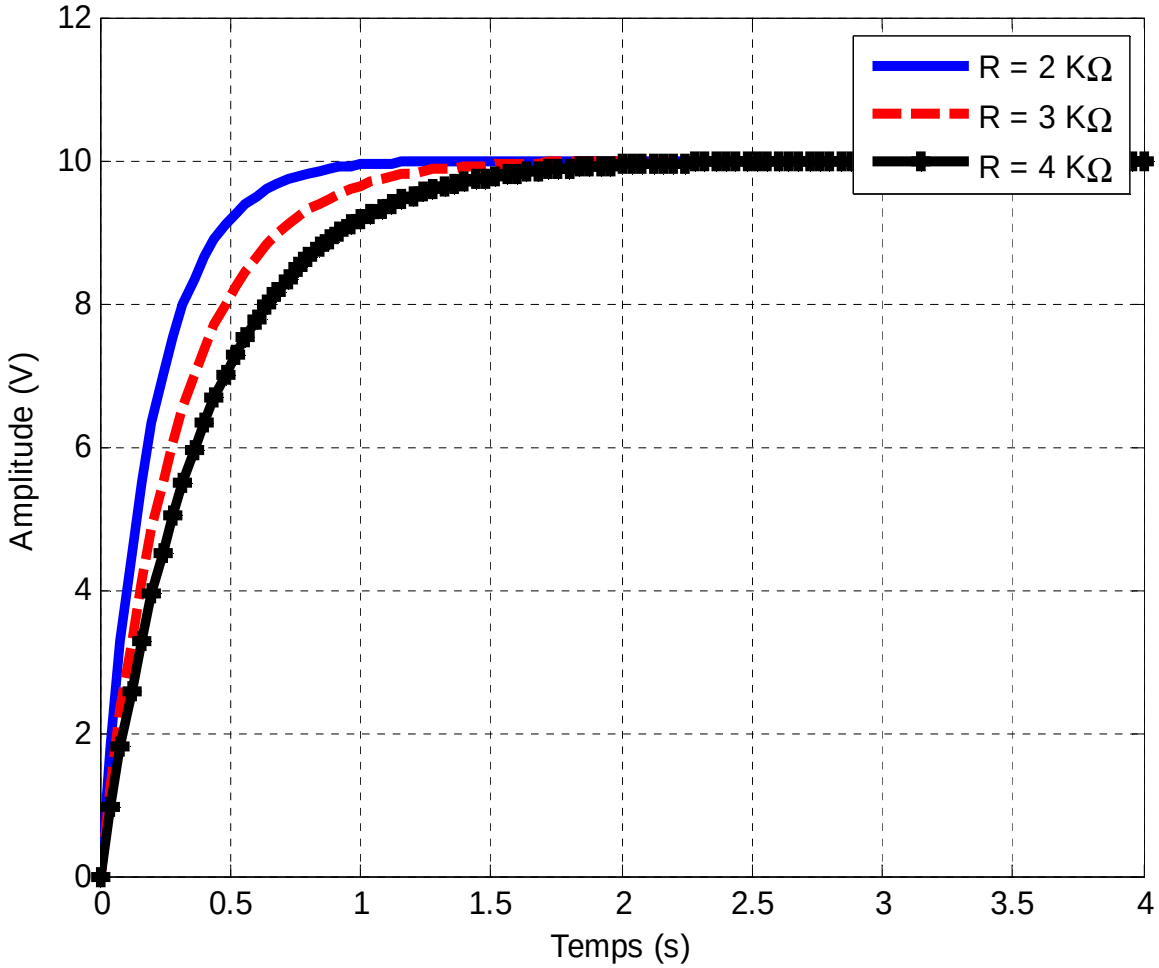
```
disp('       ======== Circuit RC, Tension aux bornes du condensateur ===========       ');
disp('Les allures des courbes sont illustrées sur la figure 1 : ');
C = 100e-6;
R1 = 100e3;
tau1 = R1*C;
R2 = 200e3;
tau2 = R2*C;
R3 = 500e3;
tau3 = R3*C;
Ve = 10;
Temps= 0:tau3/10:10*tau3;
m = 1 ;
for  tau = [tau1,tau2,tau3]
      V_s(m,:) = Ve.*(1-exp(-Temps./tau)).*heaviside(Temps);
      m = m + 1;
end
figure(1);
plot(Temps,V_s,'LineWidth',3)
axis([0,10*tau3,0,1.4*Ve]);
xlabel('Temps (s)');
ylabel('Amplitude (V)');
title('\bf Tension aux bornes du condensateur C pour différentes valeurs de la résistance R');
legend('R = 100 K\Omega','R = 200 K\Omega','R = 500 K\Omega');
grid on
disp('Le circuit étudié est représenté sur la figure 2 : ');
figure(2);
A = imread('Circuit_RC_Serie.jpg');  %Appel et visualisation de la figure du circuit enregistrée en format .jpg
image(A);
```

Exemple

Circuit RC, Tension aux bornes du condensateur

CircuitRC_Tension.m

Tensions aux bornes du condensateur C pour différentes valeurs de la résistance R



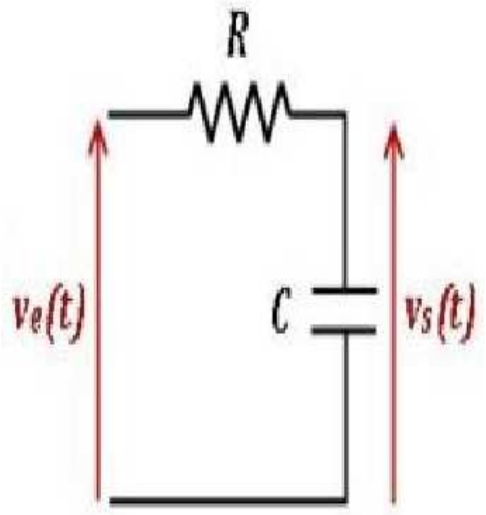
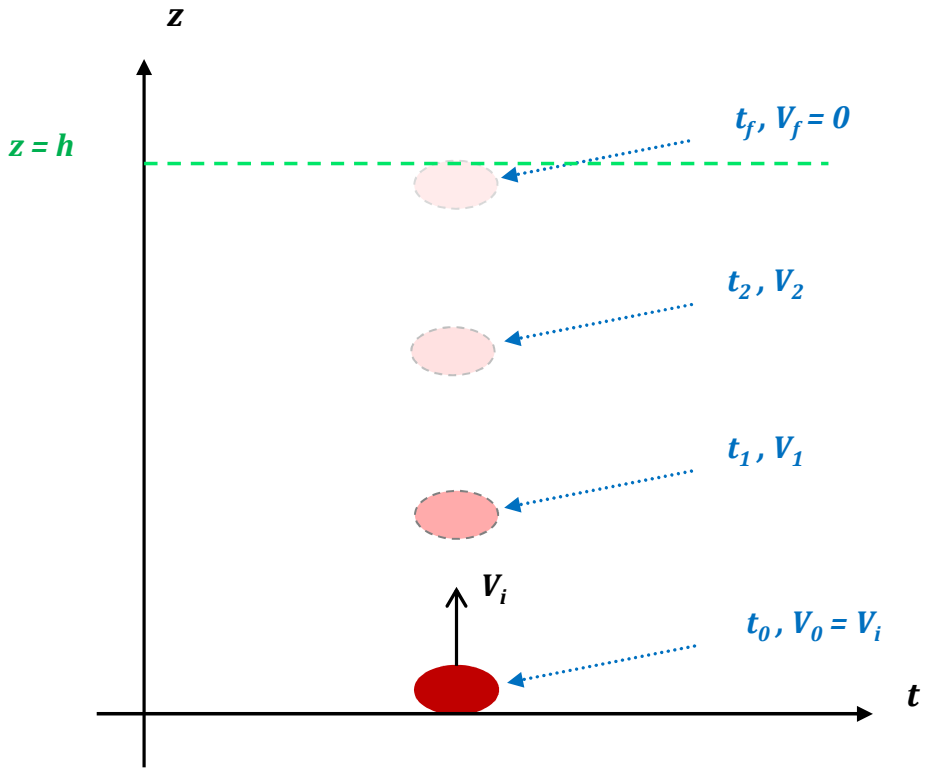


Schéma du circuit RC étudié

Exemple de mécanique : Programmation

Représentation des énergies : conservation de l'énergie mécanique

* On considère un objet de masse m = 500 g lancé verticalement avec une vitesse initiale V_i (frottements négligés).



$$E_c = \frac{1}{2} m v^2 \qquad E_p = m * g * h \qquad E_m = E_c + E_p$$

Exemple de mécanique : Conservation de l'énergie mécanique

ConservationEnergieMecanique.m

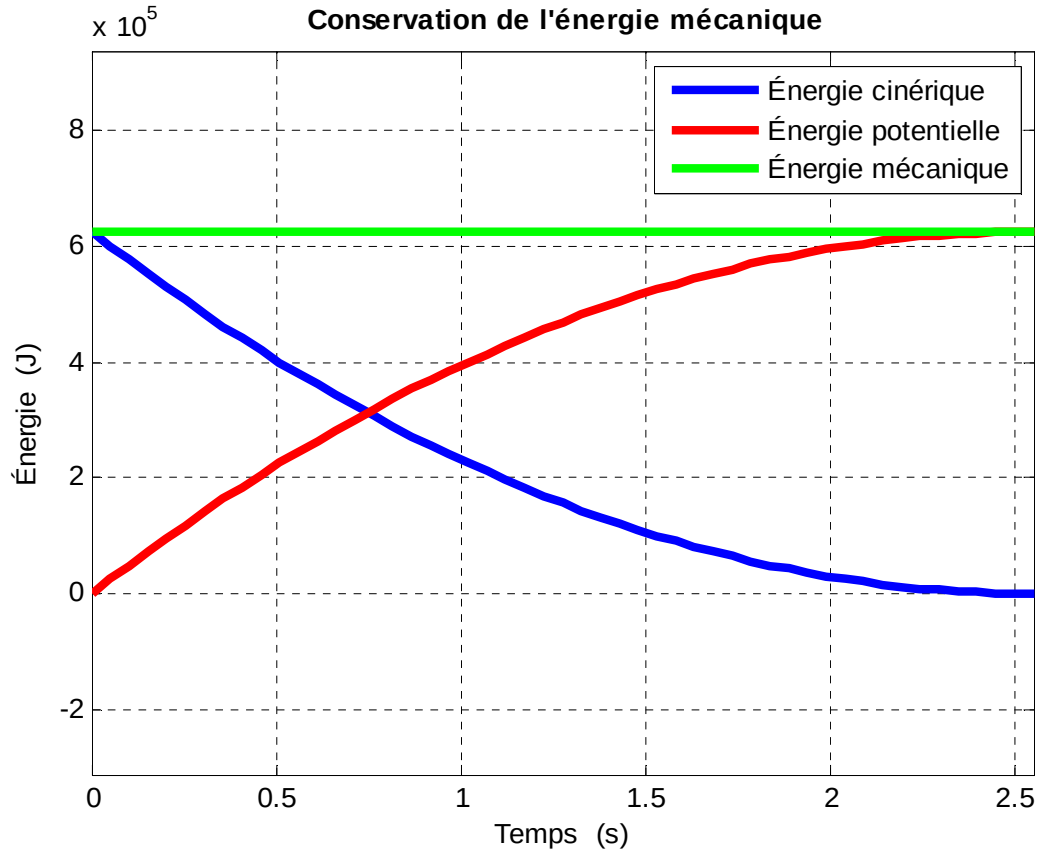
```
disp('#######-------  Conservation de l''énergie mécanique  -------#######');
Vi = input('La vitesse initiale en m/s de l''objet est = ');
g = 9.81; %accélération de la pesanteur en m/s^2
m = 500; %la masse en g
h = Vi^2/(2*g); %hauteur maximale
V = 0 : Vi/50 : Vi; %intervalle des variations de la vitesse V (m/s)
h = Vi^2/(2*g); %hauteur max
tf = h./Vi; %temps max
t = 0 : tf/50 : tf;
Ec = 0.5*m*(Vi-V).^2; %énergie cinétique
Ep = m*g*h - 0.5*m*(Vi-V).^2;%m*(0.5*V.^2); %énergie potentielle
plot(t,Ec,'linewidth',3); % Les courbes
hold on
plot(t,Ep,'r','linewidth',3);
hold on
plot(t,Ec+Ep,'g','linewidth',3);
grid on
axis([0,tf,-max(Ec)/2,max(Ec)+max(Ec)/2]);
xlabel('Temps  (s)');
ylabel('Énergie  (J)');
title('\bf Conservation de l''énergie mécanique');
legend('Énergie cinérique','Énergie potentielle','Énergie mécanique');
disp('      ===> La courbe illustre bien la conservation de l''énergie mécanique');
fprintf('      ===> Le point le plus haut a une vitesse V = 0 m/s et une hauteur h = %1.2f m',Vi^2/(2*g));
disp('   ');
disp('Un tableau récapitulatif des coordonnées de quelques points du mouvement :');
disp(['Vitesse (m/s)',blanks(6),'Hauteur (m)',blanks(6),'Énergie cinétique (J)',blanks(6),'Énergie potentielle
(J)',blanks(6),'Énergie mécanique (J)']);
for V2 = 0 : Vi/5 : Vi
   fprintf('%10.2f %24.2f %27.2f %32.2f %42.2f\n',V2,(Vi-V2)^2/(2*g),0.5*m*V2^2,m*g*(h-
V2^2/(2*g)),0.5*m*V2^2+m*g*(h-V2^2/(2*g)));
end
```

Exemple de mécanique : Conservation de l'énergie mécanique

La vitesse initiale en m/s de l'objet est = 50

===> La courbe illustre bien la conservation de l'énergie mécanique

===> Le point le plus haut a une vitesse V = 0 m/s et une hauteur h = 127.42 m



Exemple de mécanique : Programation

- Représentation des énergies : conservation de l'énergie mécanique

Un tableau récapitulatif des coordonnées de quelques points du mouvement :

Vitesse (m/s)	Hauteur (m)	Énergie cinétique (J)	Énergie potentielle (J)	Énergie mécanique (J)
0.00	127.42	0.00	625000.00	625000.00
10.00	81.55	25000.00	600000.00	625000.00
20.00	45.87	100000.00	525000.00	625000.00
30.00	20.39	225000.00	400000.00	625000.00
40.00	5.10	400000.00	225000.00	625000.00
50.00	0.00	625000.00	0.00	625000.00

Opérateurs relationnels (de comparaison)

== : égale

~= : différent

> : strictement supérieur

< : strictement inférieur

>= : supérieur ou égale

<= : inférieur ou égale

Opérateurs logiques

&& : "**et**" – "**AND**"

|| : "**ou**" – "**XOR**"

~ : "**Non**" – "**NAND**"

Université Mohammed Premier
École Nationale des Sciences Appliquées d'Oujda

Cours d' *Informatique 2 : MATLAB*

Version 3.0 (Janvier 2023)

MATLAB POUR L'INGÉNIEUR

STPI-1

ENSAO, 2022 – 2023

Partie 2

Prof. **Kamal GHOUMID**

Université Mohammed Premier
École Nationale des Sciences Appliquées d'Oujda

Cours d' ***Informatique 2 : MATLAB***

Version 3.0 (Janvier 2023)

MATLAB POUR L'INGÉNIEUR

STPI1

Chapitre 6

Fonctions Programmées

Partie 2

Prof. **Kamal GHOUMID**

Fonctions et Scripts

- Matlab est un langage de programmation puissant ainsi qu'un environnement informatique interactif.

- Dans Matlab, il existe deux types de fichiers de programme :

  * Les **scriptes** : sous formes d'un ensemble d'instructions Matlab, ils opèrent sur les données dans l'espace de travail, ils n'acceptent pas d'argument en entrée et ils ne retournent pas d'argument en sortie.

  * Les **fonctions** : elles prennent des argument en entrée (variables internes et locales à la fonction) pour retourner d'autres en sortie.

- Les fonctions sont des enchaînements de commandes Matlab, elles sont regroupées sous un nom de fonction permettant de commander leur exécution.

- Les macros peuvent contenir un groupe de commandes destiné à être exécuté plusieurs fois au cours du calcul avec éventuellement des valeurs de paramètres différents.

Types de fonctions Matlab

- Matlab offre plusieurs types de fonctions :

  * **Fonctions principales** définies au sein d'un fichier et qui peuvent contenir des **fonctions locales**. Il est possible de les appeler depuis des fichiers extérieurs. Les fonctions principales peuvent être des **fonctions privées** vues que un groupe limité d'autres fonctions.

  * **Fonctions locales** qui ne sont visibles que des fonctions principales et des autres fonctions locales du mêmes fichiers.

  * **Fonctions anonymes** (anonymous functions) qui sont des fonctions simplifiées définies à l'intérieur d'une seule instruction. Elles consistent en une seule expression Matlab et un nombre quelconque d'arguments en entrée et en sortie.

  * **Fonctions imbriquées** qui sont définies dans le corps d'une autre fonction.

Fonction ou Macros : *function*

- Une macro est un script :
  - * qui reçoit des arguments en entrée.
  - * qui renvoie des résultats.

- Pour des programmes longs et compliqués, il est souhaitable de les découper en « plusieurs fonctions (**function**)", représentantes, significatives et liées à des étapes (ou parties) du code entier, afin d'améliorer la lisibilité et la compréhension de l'algorithme.

- Les fonctions sont écrites et enregistrées dans un fichier avec une extension '**.m**' portant le nom de la fonction.

- Les macros peuvent être appelées une fois définies, comme toutes les autres fonctions qui existent dans la boite à outils de Matlab. Elles acceptent des arguments en entrée et retournent des arguments en sortie.

- La fonction peut aussi être chargée de réaliser un calcul avec un certain algorithme.

Fonction ou Macros : *function*

- La syntaxe de définition d'une fonction externe est :

function ***arguments de sortie = Nom de la fonction(arguments d'entrée)***

$$function \quad [y_1,...,y_m] = nom(x_1,....,x_n)$$

Relations sorties entrées

Avec:

'**nom**' est le nom de la fonction (on sauvegarde '**nom.m**').

'$x_1,....,x_n$' les arguments d'entrée.

'$y_1,...,y_m$' les arguments de sortie.

Fonction ou Macros : *function*

- Pour écrire fonction macro dans Matlab, on doit d'abord **commencer par 'function'**, donner les **noms des paramètres en sortie générés** par la fonction, puis le **nom de la fonction**, et enfin les **noms des paramètres en entrée**.

- Le nom du fichier contenant la fonction macro doit porter systématiquement et d'une manière obligatoire le nom de cette dernière.

- Les fonctions peuvent être appelées à partir de :
  - l'éditeur de commandes
  - une autre fonction
  - un script

- Les fichiers doivent être sauvegardés dans le même dossier.

Fonction ou Macros : *function*

- **Exemple** :

fact.m

```
function   y = fact(n)
        y = prod(1:n);  %calcul de n!
```

Une fois sauvegarder sous le nom '**fact.m**', Dans l'espace de commande on peut taper par exemple :

```
>> fact(3)
ans =
     6

>> fact(5)
ans =
   120
```

```
>> fact(6)
ans =
   720

>> fact(10)
ans =
   3628800
```

Fonction ou Macros : *function*

- **Exemple**

```
function  y = fact(n)
          y = prod(1:n);  %calcul de n!
```

fact.m

0n cherche à calculer maintenant :

$$C_n^k = \frac{n!}{k!\,(n-k)!}$$

---------- Exemple de calcul de l'expression de combinaison ----------

```
>> n1 = 10;
>> k1 = 6;
>> C1 = fact(n1)/(fact(k1)*fact(n1-k1))
C1 =
    210
```

```
>> n2 = 30;
>> k2 = 27;
>> C2 = fact(n2)/(fact(k2)*fact(n2-k2))
C2 =
    4.0600e+003
```

Fonction ou Macros : *function*

- Exemple:

  * Calcul de modulo ***Modulo.m***

Modulo.m

```
function [r,q] = Modulo(a,m)
        q = floor(a./m); %partie entière inférieure
        r = a – m*q;
```

Dans **l'éditeur de commande**, on peut taper par exemple :

```
>> [a1,b1] = Modulo(26,13)

a1 =
     0

b1 =
     2
```

```
>> [a2,b2] = Modulo(47,13)

a2 =
     8

b2 =
     3
```

```
>> [a2,b2] = Modulo([4,-4,10,27,-134,242],9)

a2 =
     4     5     1     0     1     8

b2 =
     0    -1     1     3   -15    26
```

Fonction ou Macros : *function*

- Exemple:

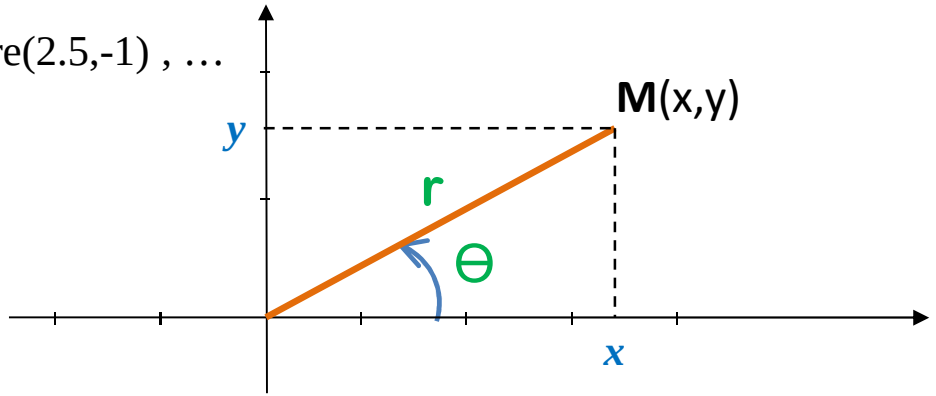
  * Coordonnées polaires: ***Polaire.m***

Polaire.m

```
function  [r,theta] = Polaire(x,y)
          r = sqrt(x^2+y^2);
          theta = atan(y/x);
```

Dans l'éditeur de commande, on peut taper par exemple :

[r,t] = Polaire(3,4) , [a,b] = Polaire(2.5,-1) , …



Fonction ou Macros : *function*

- Exemple:

  * **Coordonnées polaires** :

Polaire.m

Dans l'éditeur de commande, on peut taper par exemple :

```
>> [r,theta] = Polaire(3,4)

r =

     5


theta =

    0.9273
```

```
>> [module,angle] = Polaire(2,7)

module =

    7.2801


angle =

    1.2925
```

Fonction ou Macros : *function*

- **Exemple** : Calculs des racines

```
function [r1,r2] = RacineEqDegre2(a,b,c);
    Delta = b^2 - 4*a*c;
    if  Delta > 0
        r1 = (-b - sqrt(Delta))/(2*a);
        r2 = (-b + sqrt(Delta))/(2*a);
    elseif  Delta < 0
        r1 = (-b - i * sqrt(-Delta))/(2*a);
        r2 = (-b + i * sqrt(-Delta))/(2*a);
    else
        r1 = -b/(2*a);
        r2 = -b/(2*a);
end
```

RacineEqDegre2.m

Fonction ou Macros : *function*

RacineEqDegre2.m

- **Exemple** : Calculs des racines

---------- Exemple de calcul des deux racines ----------

```
>> [r1,r2] = RacineEqDegre2(1,5,-14);

 r1 =
      -7

r2 =
       1
```

```
>> [r1,r2] = RacineEqDegre2(5,-1,3);

    r1 =
       0.1000 - 0.7681i

    r2 =
       0.1000 + 0.7681i
```

```
>> [a,b] = RacineEqDegre2(1,-6,9)

a =
    3

b =
    3
```

```
>> [Racine1,Racine2] = RacineEqDegre2(2,-46,224)

    Racine1 =
          7

    Racine2 =
         16
```

Fonction ou Macros : *function*

Cal2Joule.m

- **Exemple** : 1er script : ***Cal2Joule.m***

```
function  Cal = Cal2Joule(j)
Cal = 4.18*j;  %1 Calorie = 4.18 joules
```

2ème script : **ConversionEnergie.m**

```
disp(['Calorie',blanks(10),'Joule']);
for  j = 1 : 3 : 31
     fprintf('%4d %17.2f\n',j,Cal2Joule(j));
end
%Après exécution du 2ème script dans l'espace de commande
```

Calorie	Joule
1	4.18
4	16.72
7	29.26
10	41.80
13	54.34
16	66.88
19	79.42
22	91.96
25	104.50
28	117.04
31	129.58

Fonction ou Macros : *function*

SurfVolCylindre.m

- Exemple : Calcul de la surface et du volume d'un cylindre

```
function [S,V] = SurfVolCylindre(r,h)
S = pi*2*r*h;
V = pi*r^2*h;
```

---------- Exemple d'appel de la fonction ----------

```
>> [S1,V1] = SurfVolCylindre(2,5)

S1 =

   62.8319


V1 =

   62.8319
```

```
>> [S1,V1] = SurfVolCylindre(3,10)

S1 =

  188.4956


V1 =

  282.7433
```

Fonction ou Macros : *function*

- **Exemple** : Calculs statistiques

CalculStat.m

```
function [Moyenne,Ecarttype] = CalculStat(x)
    n = length(x);
    Moyenne = sum(x)/n;
    Ecarttype = sqrt(sum((x - Moyenne).^2/n));
```

---------- Exemple d'appel de la fonction ----------

```
>> x1 = [15,12,9,13,17,10,15,14];
>> [Moy,Ecart] = CalculStat(x1)

Moy =

   13.1250

Ecart =

    2.5218
```

```
>> x2 = 5:27;
>> [Moy,Ecart] = CalculStat(x2)

Moy =

    16

Ecart =

    6.6332
```

Fonction ou Macros : *function*

ConvDevise.m

```
function  Devise = ConvDevise(Dirham,Taux)
    Devise = Dirham / Taux;
```

ExemplesConversionsDevises.m

---------- Exemple d'appel de la fonction dans un autre script ----------

```
disp('##############---------- Exos -3- de l'examen ----------##############');
disp('                    ---------- Conversions de devises ----------');
fprinft('\n');
disp(['Dirham',blanks(15),'Euro',blanks(17),'USD',blanks(18),'CAD']);
for  d = 100 : 50 : 300
  fprintf('%7d %20.2f %20.2f %18.2f\n',d,ConvDevise(d,11.5),ConvDevise(d,9.17),ConvDevise(d,7.13));
end
```

```
############---------- Exos -3- du devoir 2 ----------#############
---------- Conversions de devises ----------
Dirham          Euro          USD          CAD
    100          8.70         10.91         14.03
    150         13.04         16.36         21.04
    200         17.39         21.81         28.05
    250         21.74         27.26         35.06
    300         26.09         32.72         42.08
```

Fonction ou Macros : *function*

- Exemple : Calcul simple par l'intermédiaire d'une macro

MaFct4.m

```
function [S1,S2,S3] = MaFct4(a,b,c)
    S1 = a*b;
    S2 = sin(c);
    S3 = sqrt(abs(a*b*c));
```

---------- Exemple d'appel de la fonction ----------

```
>> [S1,S2,S3] = MaFct4(3,10,4)
S1 =
    30

S2 =
   -0.7568

S3 =
   10.9545
```

```
>> [S4,S5,S6] = MaFct4(-2,sqrt(15),3)
S4 =
   -7.7460

S5 =
    0.1411

S6 =
    4.8206
```

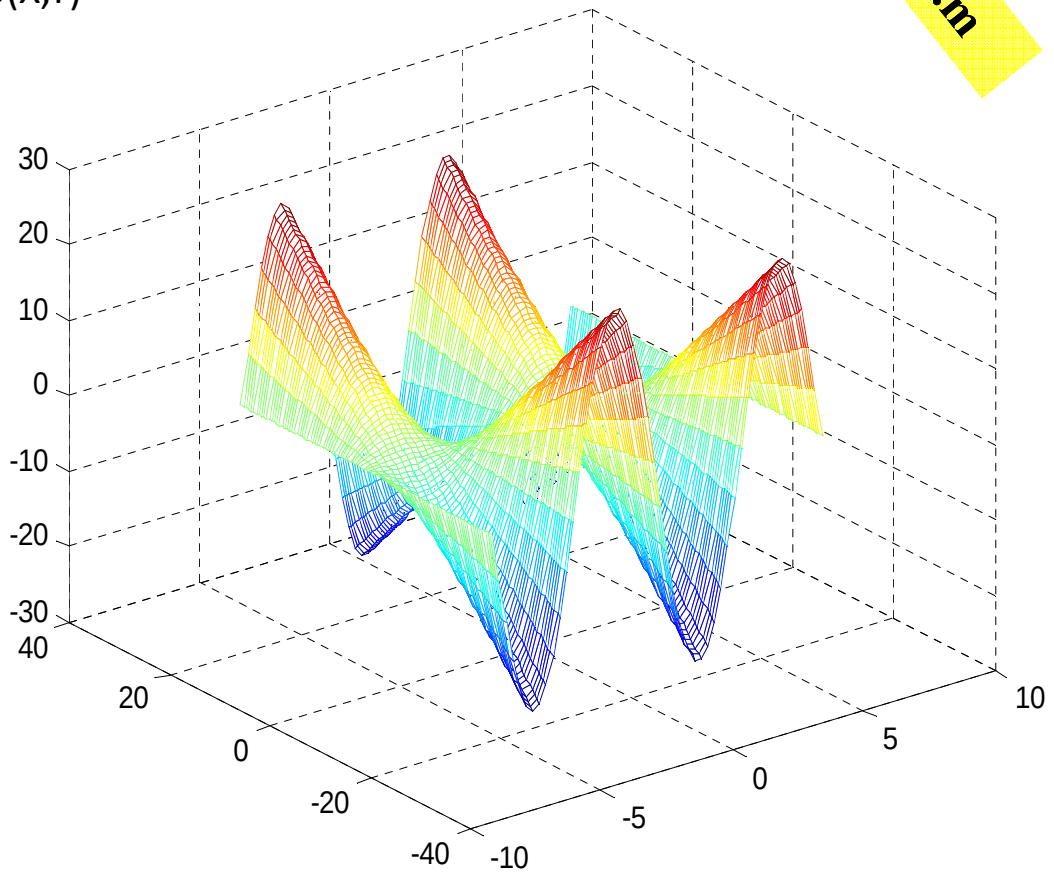
Fonction ou Macros : *function*

- Exemple : Représentation graphique par l'intermédiaire d'une macro

MaFct5.m

```
function Z = MaFct5(X,Y)
Z = sin(X).*Y;
```

```
>> x1 = linspace(-2*pi,2*pi,50);
>> x2 = linspace(-25,25,100);
>> [X,Y] = meshgrid(x1,x2);
>> mesh(X,Y,MaFct5(X,Y))
```



Fonction ou Macros : *function*

```
function  y = MafonctionF1(x)
  if  (x < 0)
     y = sin(2*x);
  elseif  ((x >= 0)  &&  (x < 3))
     y = exp(-x)*sin(2*x);
  else  (x >= 3)
     y = sqrt(sin(2*x)+1);
  end
```

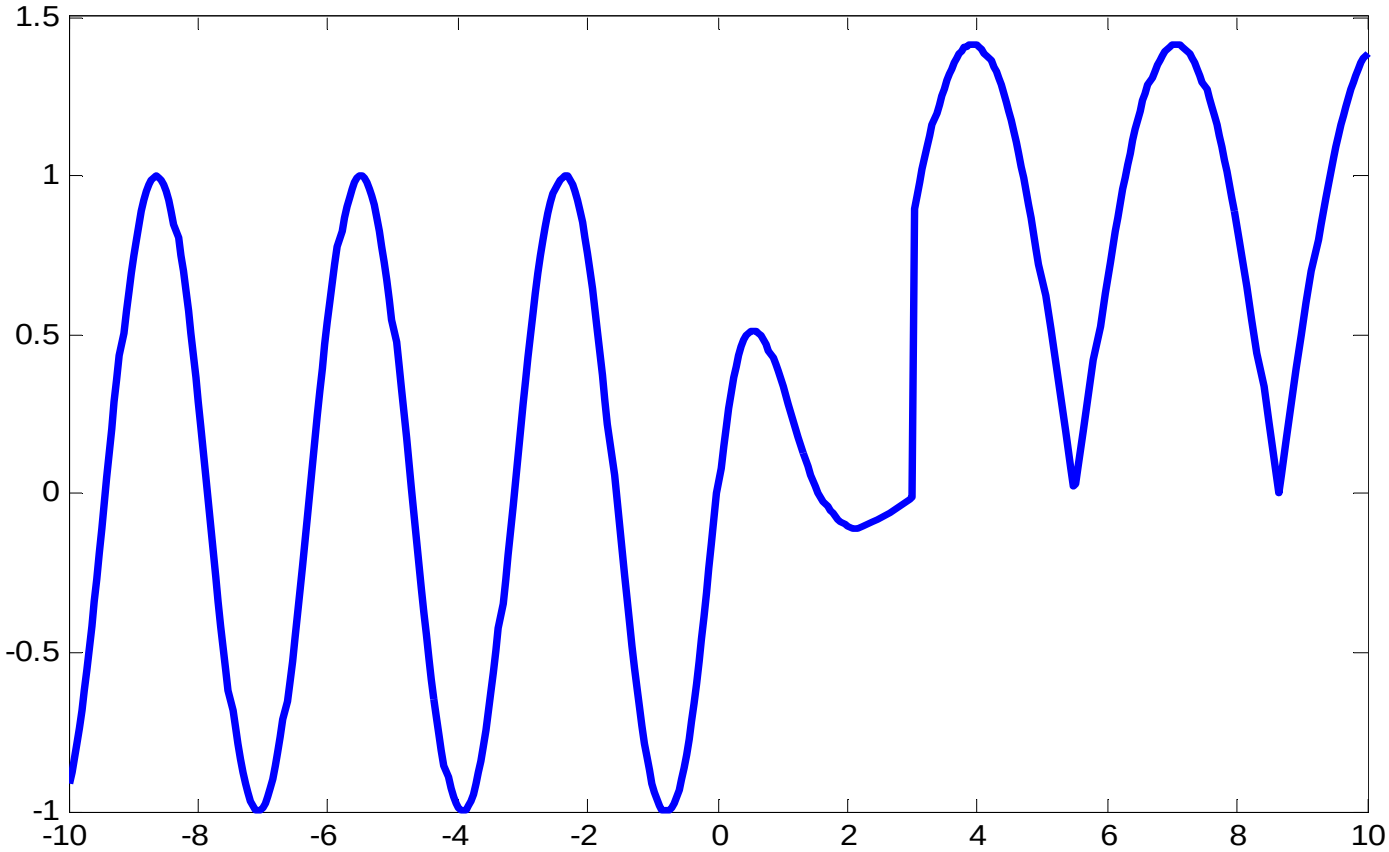
MafonctionF1.m

```
>> MafonctionF1 (-5)
ans =
   0.5440

>> MafonctionF1 (-1)
ans =
  -0.9093

>> MafonctionF1 (0)
ans =
    0
```

```
>> MafonctionF1 (1)
ans =
   0.3345

>> MafonctionF1 (1.7)
ans =
  -0.0467

>> MafonctionF1 (5)
ans =
    0.6753
```

Fonction ou Macros : *function*

```
>> fplot('MafonctionF1',[-10 10]);
```



Fonction ou Macros : *function*

```
>> x = fminsearch(@(x) MafonctionF1(x),0,3) %On cherche la valeur minimale de f2 sur [0 3]
x =
   -0.7854

>> fzero(@(x) MafonctionF1(x),1,3)
ans =
    1.5708

>> quad(@(x) exp(-x).*sin(2*x),0,2) %Calcul d'intégrale (il faut mettre ".")
ans =
    0.4559

>> quad(@(x) sqrt(sin(2*x)+1),4,12) %Calcul d'intégrale en utilisant l'instruction 'quad'
ans =
    7.0017

>> x=4:0.1:12;
>> F=sqrt(sin(2*x)+1); %Calcul d'intégrale en utilisant l'instruction 'trapz'
>> trapz(x,F)
ans =
    7.0017
```

Fonctions anonymes : Anonymous functions

- Une fonction anonyme est une fonction Matlab définie directement, sans la création préalable d'un fichier spécifique.

- Le symbole "**@**", est un pointeur vers la fonction (**handle**), permet la matérialisation de cette création de fonction anonyme.

- L'intérêt majeur des fonctions anonymes réside dans la souplesse que permet leurs syntaxes. Elles sont souvent utilisées dans un contexte de création des raccourcis syntaxiques.

- La syntaxe générale d'une fonction anonyme est :

  Nom de la fonction = @(***Nom de la variable***) Expression de la fonction

  Appel de la fonction anonyme en précisant les arguments Nécessaires

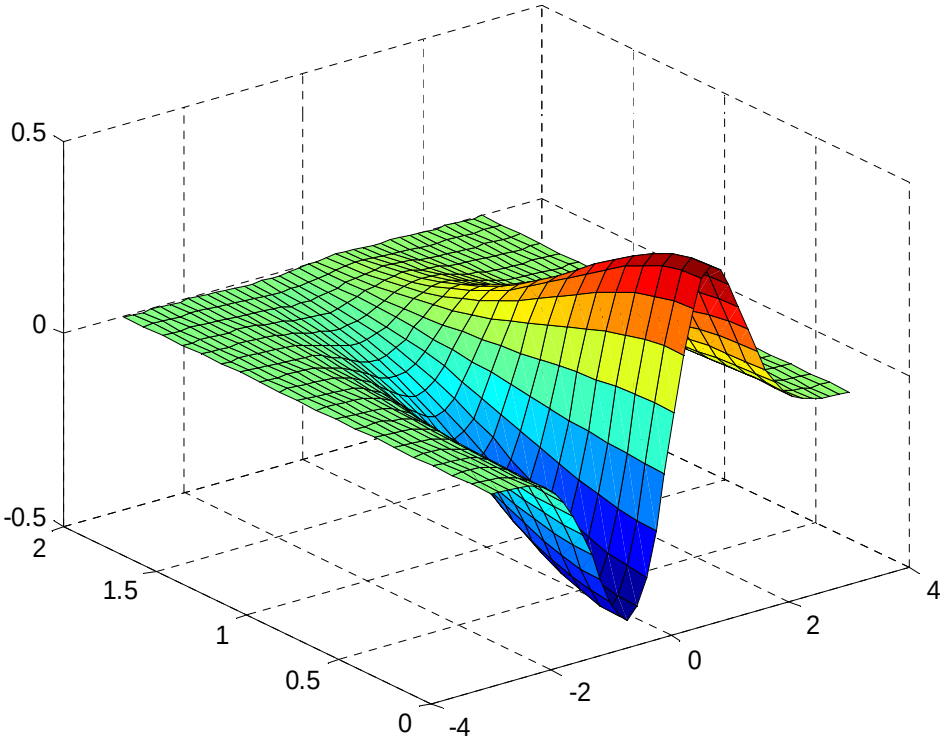
- On peut utiliser aussi l'instruction "*inline*" pour définir une fonction anonyme.

Anonymous functions

```
>> f = inline('sin(2*x)','x')
f =
     Inline function:
     f(x) = sin(2*x)

>> f(1)
ans =
    0.9093

>> f(5)
ans =
   -0.5440

>> g = inline('x.*exp(-x.^2 - y.^2)','x','y');
>> g(0,-1)
ans =
     0
```

L'instruction "inline" est non recommandée dans les versions récentes de Matlab

```
>> x=linspace(-3,3,40);
>> y=linspace(0,2,20);
>> [xx,yy] = meshgrid(x,y);
>> surf(xx,yy,g(xx,yy))
```



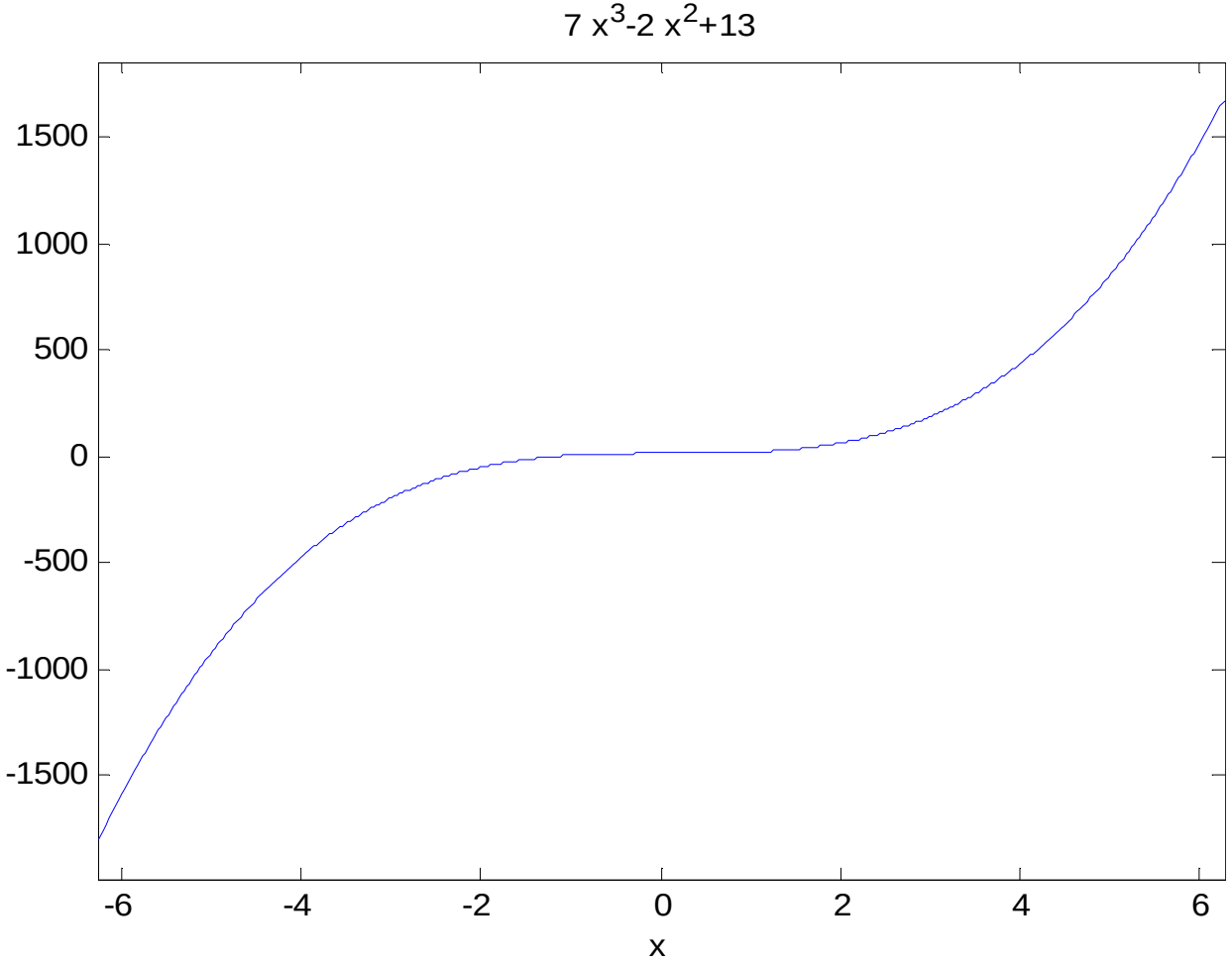
Anonymous functions

```
>> MaFct2 = @(x) 7*x.^3 - 2*x.^2 + 13
MaFct2 =
    @(x)7*x.^3-2*x.^2+13

>> MaFct2(2)
ans =
     61

>> MaFct2(sqrt(3))
ans =
    43.3731

>> MaFct2(-3)
ans =
    -194
```

```
>> Puissancen = @(x,n) x.^n;
>> Puissancen(2,2)
ans =
     4

>> Puissancen(4,3)
ans =
     64

>> Puissancen(1:7,3)
ans =
     1     8    27    64   125   216   343

>> Puissancen(4,3)*MaFct2(-3)
ans =
    -12416
```

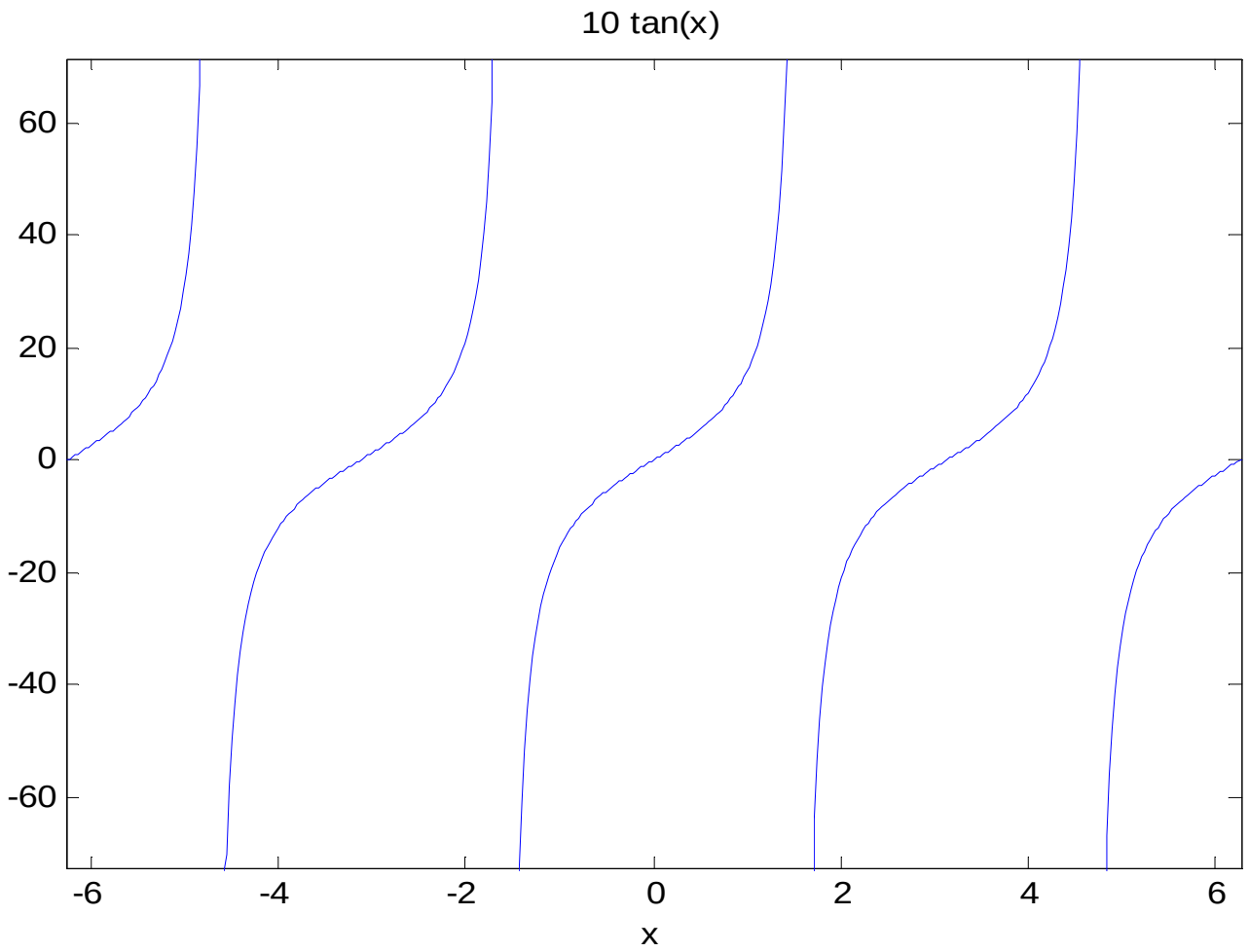
Anonymous functions

>> **ezplot(MaFct2)**



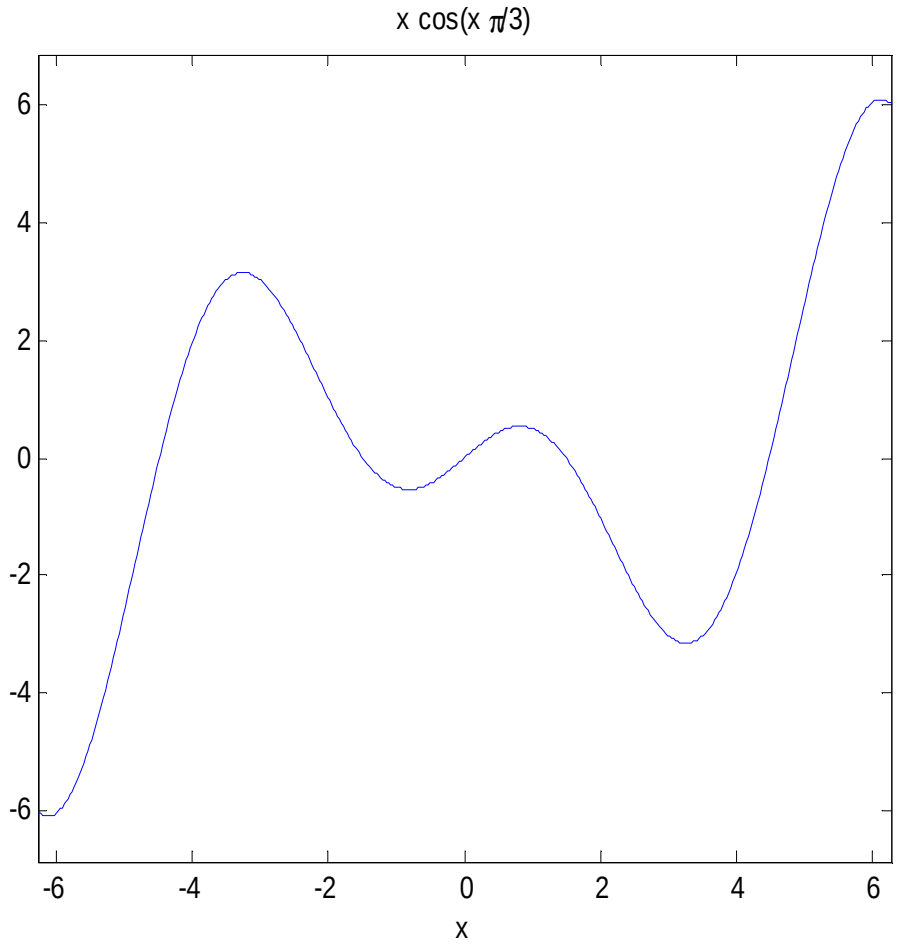
Anonymous functions

```
>> ezplot('10*tan(x)')
```



Anonymous functions

```
>> MaFct3 = @(x)  x.*cos(x*pi/3);
>> MaFct3(10)
ans =
   -5.0000

>> quad(@(x)  x.* cos(x*pi/3),0,2)
ans =
    0.2862

>> quad(@(x)  x.* cos(x*pi/3),0,2*pi)
ans =
    1.7135

>> ezplot(MaFct3)
```



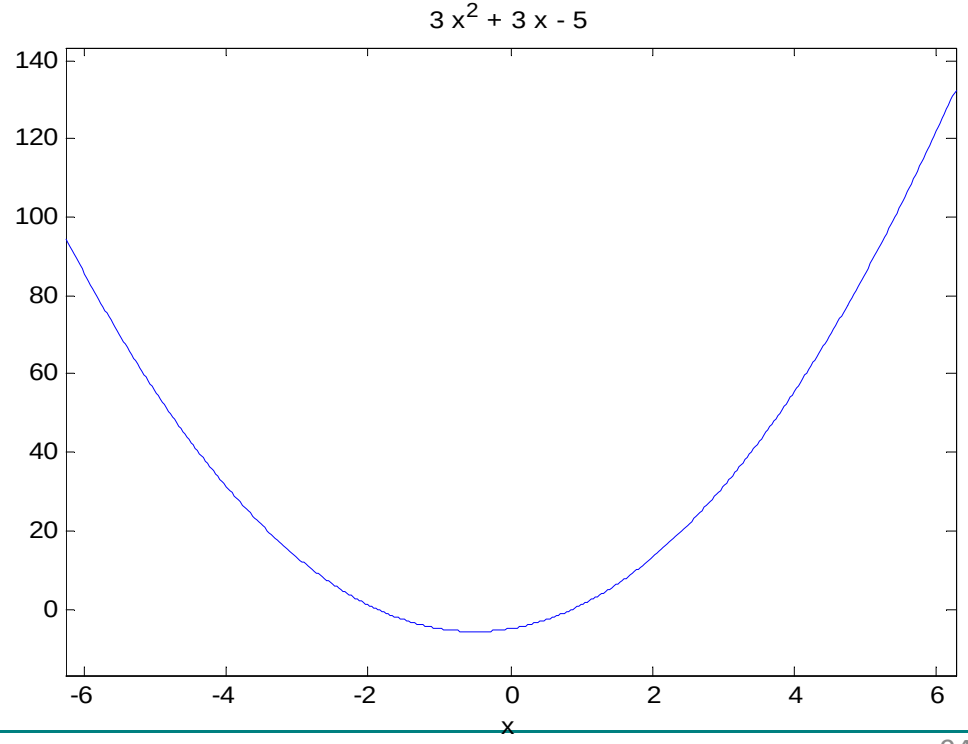
Anonymous functions

- Calculs de l'intégrale suivante pour différentes valeurs du paramètre m

$$f(m) = \int_0^4 (3x^2 + m\,x - 5)\,dx$$

```
>> f = @(m) (quad(@(x) (3*x.^2 + m*x - 5),0,4));
>> f(3)
ans =
    68

>> f(0)
ans =
   44.0000

>> f(-34)
ans =
  -228
```

```
>> ezplot('3*x.^2 + 3*x - 5')
```

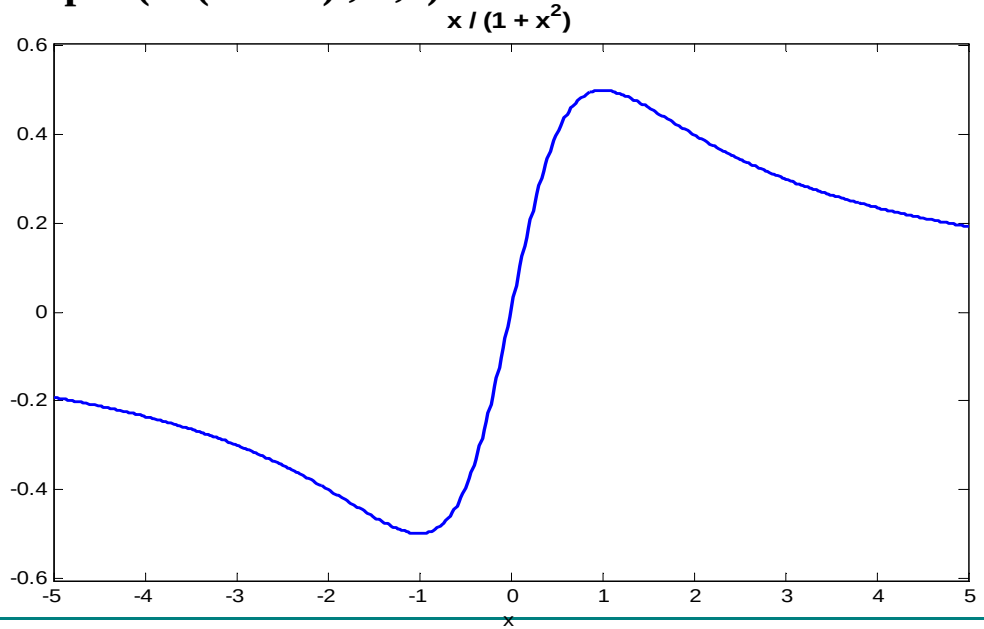


Anonymous functions

$$I = \int_0^1 \frac{x}{1+x^2}\,dx = \int_0^1 \frac{1}{2}\frac{2x}{1+x^2}\,dx = \left[\frac{1}{2}\ln(1+x^2)\right]_0^1 = \frac{1}{2}\ln(2) = 0{,}3466$$

```
>> F_1 = @(x) (x./(1+x.^2));
>> F_1(4)
ans =
   0.2353
```

```
>> ezplot('x/(1+x^2)',-5,5)
```



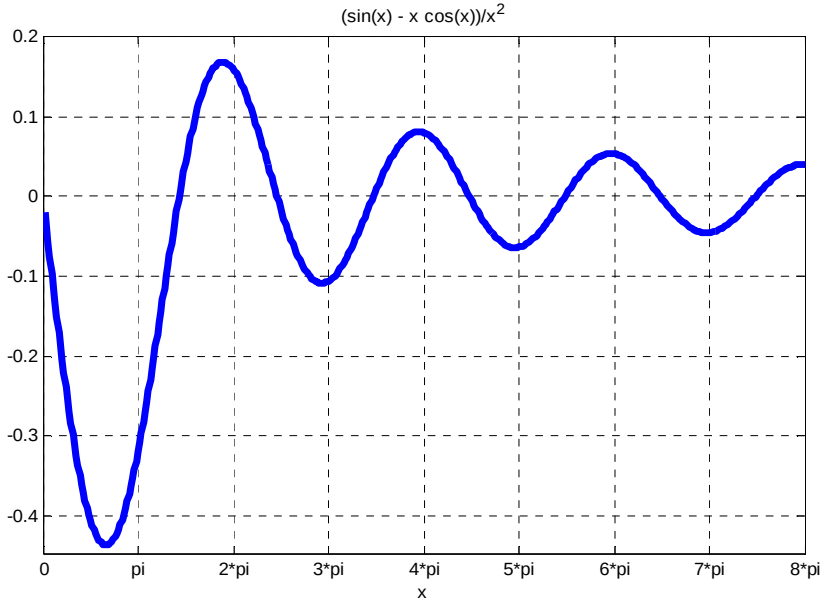

```
>> I_1 = quad(@(x) x./(1+x.^2),0,1)
I_1 =
   0.3466
```

```
>> x = 0 : 0.01 : 1;
>> F_2 = (x ./ (1+x .^2));
>> I_2 = trapz(x,F_2)
I_2 =
   0.3466
```

Anonymous functions

$$I = \int_{\pi/2}^{\pi} \frac{x\cos(x) - \sin(x)}{x^2}\,dx = \int_{\pi/2}^{\pi} \frac{u'v - vu'}{v^2}\,dx = \left[\frac{\sin(x)}{x}\right]_{\pi/2}^{\pi} = \frac{-2}{\pi}$$

```
>> syms x
>> I = int((x.*cos(x) - sin(x))./(x.^2))
I =
    sin(x)/x

>> I_1 = int((x.*cos(x) - sin(x))./(x.^2),pi/2,pi)
I_1 =
    -2/pi   % -2/pi = -0.6366
```




```
>> I_2 = quad(@(x) (x.*cos(x) - sin(x))./(x.^2),pi/2,pi)
I_2 =
  -0.6366
```

```
>> x = pi/2 : 0.01 : pi;
>> F_2 = (x .*cos(x) - sin(x)) ./ (x .^2);
>> I_2 = trapz(x,F_2)
I_2 =
  -0.6364
```

MATLAB POUR L'INGÉNIEUR

STPI-1

ENSAO, 2022 – 2023

Fin de la Partie 2

Prof. **Kamal GHOUMID**