Université Abdelmalek Essaadi

École Nationale des Sciences Appliquées Al Hoceima

Cours d' *Informatique 3 : MATLAB*

# MATLAB POUR L'INGÉNIEUR

## *AP-2*

## *ENSAH-2020-2021*

Partie 2

Pr. *Amina GHADBAN*

Université Abdelmalek Essaadi

École Nationale des Sciences Appliquées Al Hoceima

Cours d' *Informatique 3 : MATLAB*

# MATLAB POUR L'INGÉNIEUR

## *AP-2*

## *Chapitre 5*

## *Programmation MATLAB*

Pr. *Amina GHADBAN*

# Vocabulaires (1/3)

- La programmation consiste à combiner les 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 dédié 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.

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

# Caractères et chaînes de caractères (1/1)

- 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' , 'mot' , c = 'c contient une chaine de caractères' .

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

- **Exemple** :

  Les listes de caractères **['E' 'N' 'S' 'A' 'H']** et **['EN' 'S' 'AH']** sont identiques à la chaîne de caractères **['ENSAH']** :

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

  ans =

  **ENSAH**

# Formats d'affichage et de lecture (1/10)

## 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.

# Formats d'affichage et de lecture (2/10)

## 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
```

# Formats d'affichage et de lecture (3/10)

## Affichage simple '**disp**':

- Exemples:

```
>> disp(['Deuxième' blanks(1) 'année ' blanks(1) ' ENSAH'])
```

Deuxième année ENSAH

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

En deuxième année vous êtes 180 étudiants

```
>> disp(['Le cours de Matlab est programmé à ' num2str(14) ' heures ' num2str(30)])
```

Le cours de Matlab est programmé à 14 heures 30

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

# Formats d'affichage et de lecture (4/10)

## 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'**.

# Formats d'affichage et de lecture (5/10)

## 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 : ENSAH

>> disp('B')
        B

>> disp(B)
        ENSAH
```

# Formats d'affichage et de lecture (6/10)

**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
```

# Formats d'affichage et de lecture (7/10)

## 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.

# Formats d'affichage et de lecture (8/10)

## Impression à l'écran : fprintf

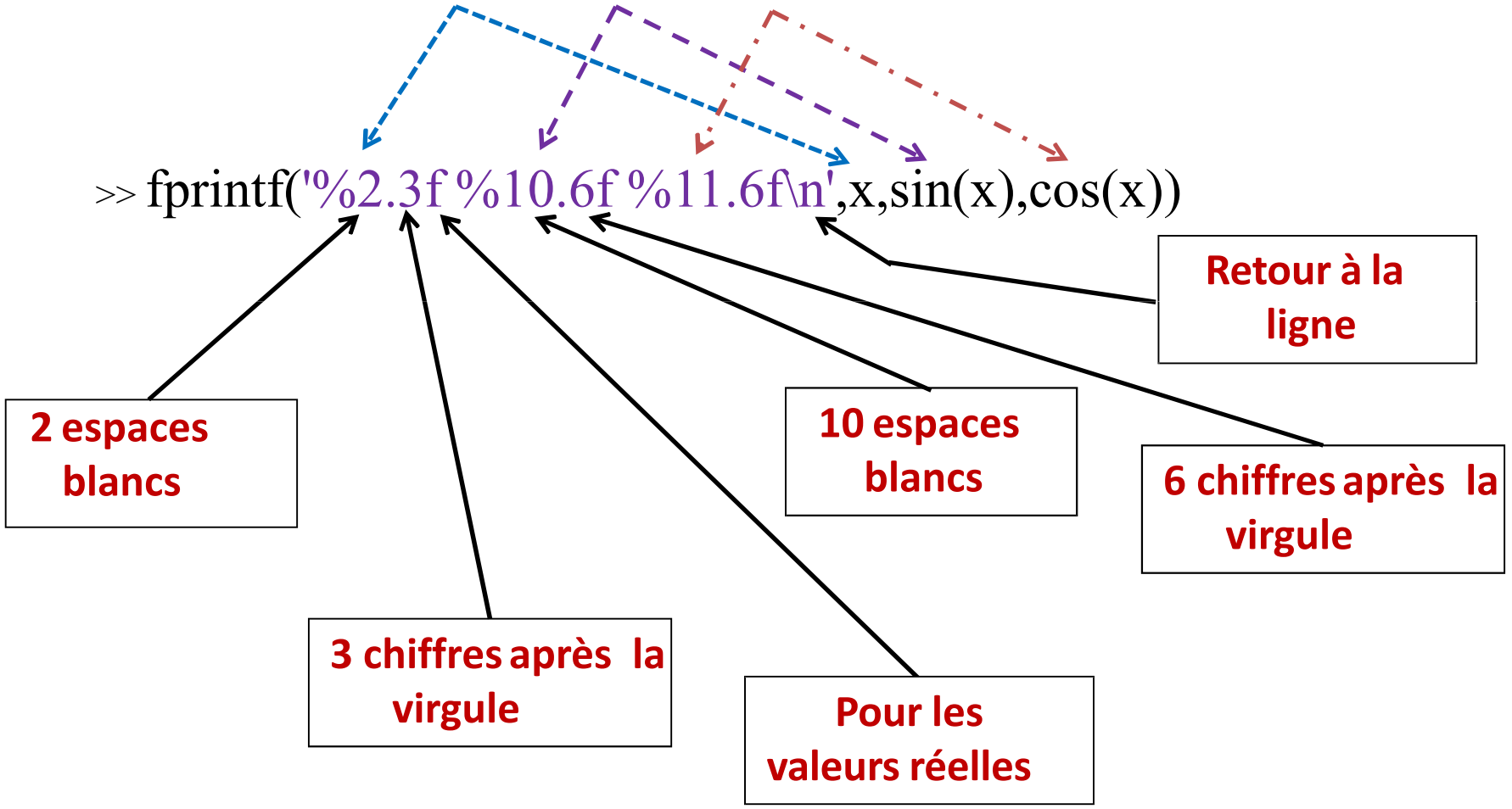
- Exemples:

```
>> 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
```

# Formats d'affichage et de lecture (9/10)

**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
```



# Formats d'affichage et de lecture (10/10)

**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
```

# Programmation et algorithmique avec Matlab (1/20)

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 indispensable et présente dans tous les langages de programmation, mais 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** :

```
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:

```
nb = input('Vous êtes en Baccalauréat, quel est votre âge? :    ');
if (nb < 18)
        disp('Vous avez gagné minimum une année‘);
elseif (nb = = 18)
        disp('Normalement, vous n’avez pas raté d’année');
else
        disp('Vous avez perdu au moins une année');
end
```

## Instruction conditionnelle if

Exemple:

```
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:

```
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:

```
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
```

## Exemple récapitulatif

```
disp('           ');
disp('=============================================================================================================');
disp('   ----      Exemples récapitulatif sur les utilisations de "disp", "input", "if ... end", "switch ... end"       ----- ');
disp('=============================================================================================================');
disp('           ');
disp('En 2eme année vous êtes 120 étudiants.');
disp('Ce petit code vous aide à connaître vos créneaux et salles de TD d"Info3:');
disp('           ');
EtudiantCP2= input('Vous êtes étudiants en AP2(Oui ou Non?) : ','s');
switch EtudiantCP2
   case 'Non'
      disp('Ces affectations de groupes et de salles ne concernent que les étudiants CP2, Excusez-nous SVP.');
   case 'Oui'
         NumCP2 = input('Quel est votre numéro CP2-ENSAH? : ');
      if NumCP2 >= 1 && NumCP2 <=30
            disp(['L"étudiant dont le numéro CP2-ENSAH est ' num2str(NumCP2) ' appartient au groupe A, il a le TD d"Info 3le Lundi après-midi, Salle AE1.']);
         elseif NumCP2 >= 31 && NumCP2<=60
            disp(['L"étudiant dont le numéro CP2-ENSAH est ' num2str(NumCP2 )' appartient au groupe B, il a le TD d"Info 3le Mardi matin, Salle AE2.']);
         elseif NumCP2>= 61 && NumCP2<=90
            disp(['L"étudiant dont le numéro CP2-ENSAH est ' num2str(NumCP2) ' appartient au groupe C, il a le TD d"Info 3le Mercredi après-midi, Salle AE3.']);
         elseif NumCP2 >= 91 && NumCP2<=120
            disp(['L"étudiant dont le numéro CP2-ENSAH est ' num2str(NumCP2) ' appartient au groupe D, il a le TD d"Info 3 le Jeudi matin, Salle AE4.']);
          else
            disp('Le numéro CP2-ENSAH doit être compris entre 1 et 120.');
      end
   otherwise
      disp('Votre réponse n"est pas valide, Vous devriez répondre avec "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.

- 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 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

```
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(12) ...
   num2str(sin(x)^2 + cos(x)^2)]);
end
```

| 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 |

# Formats d'affichage et de lecture (13/20)

## Structure itératives for

```
fprintf([' x' blanks(18) 'sin(x)' blanks(13) 'DévLimité(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
```

| x | sin(x) | DévLimité(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 |

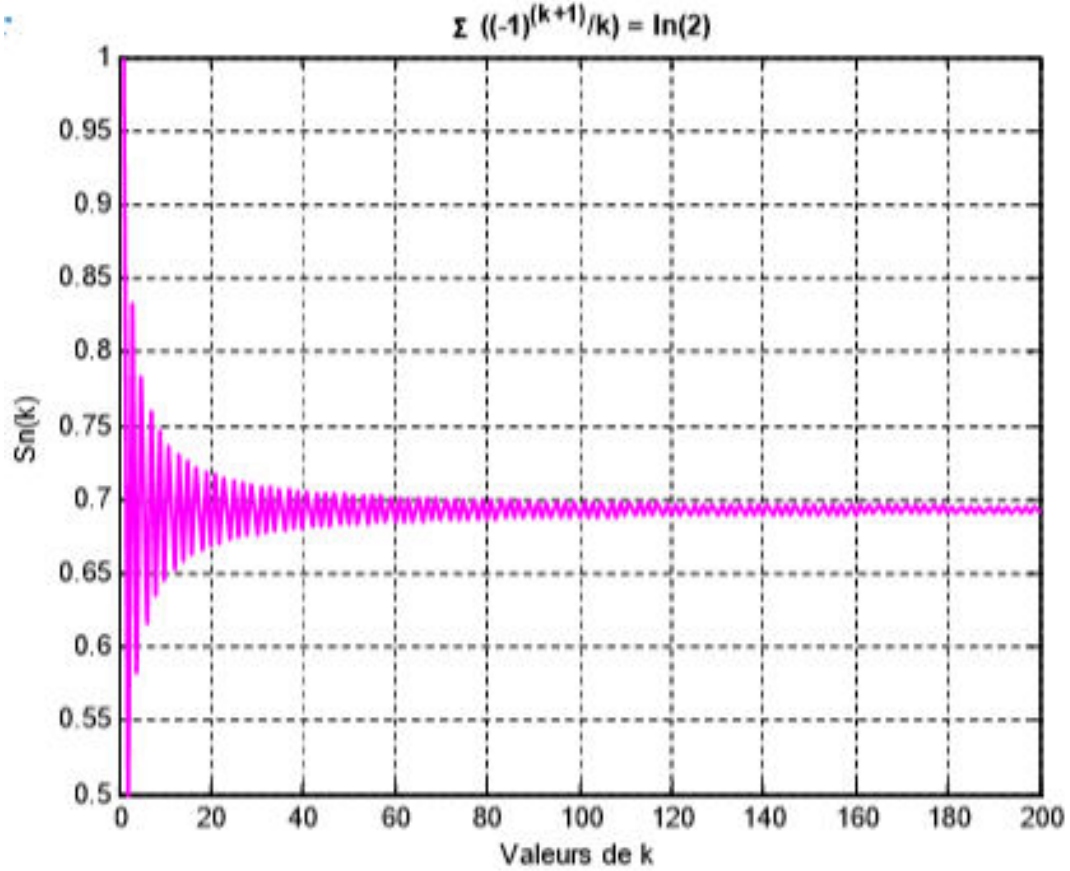
```
disp(['T en ° Celsius' blanks(13) 'T en Kelvin']);
for T = 0:5:25
            K = T + 273.5;
            fprintf('%12i %34.2f\n',T,K);
end
```

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

# Formats d'affichage et de lecture (14/20)

## Structure itératives for

```
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('(k)');
title('\bf \Sigma ((-1)^{(k+1)}/k) = ln(2)');
```




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

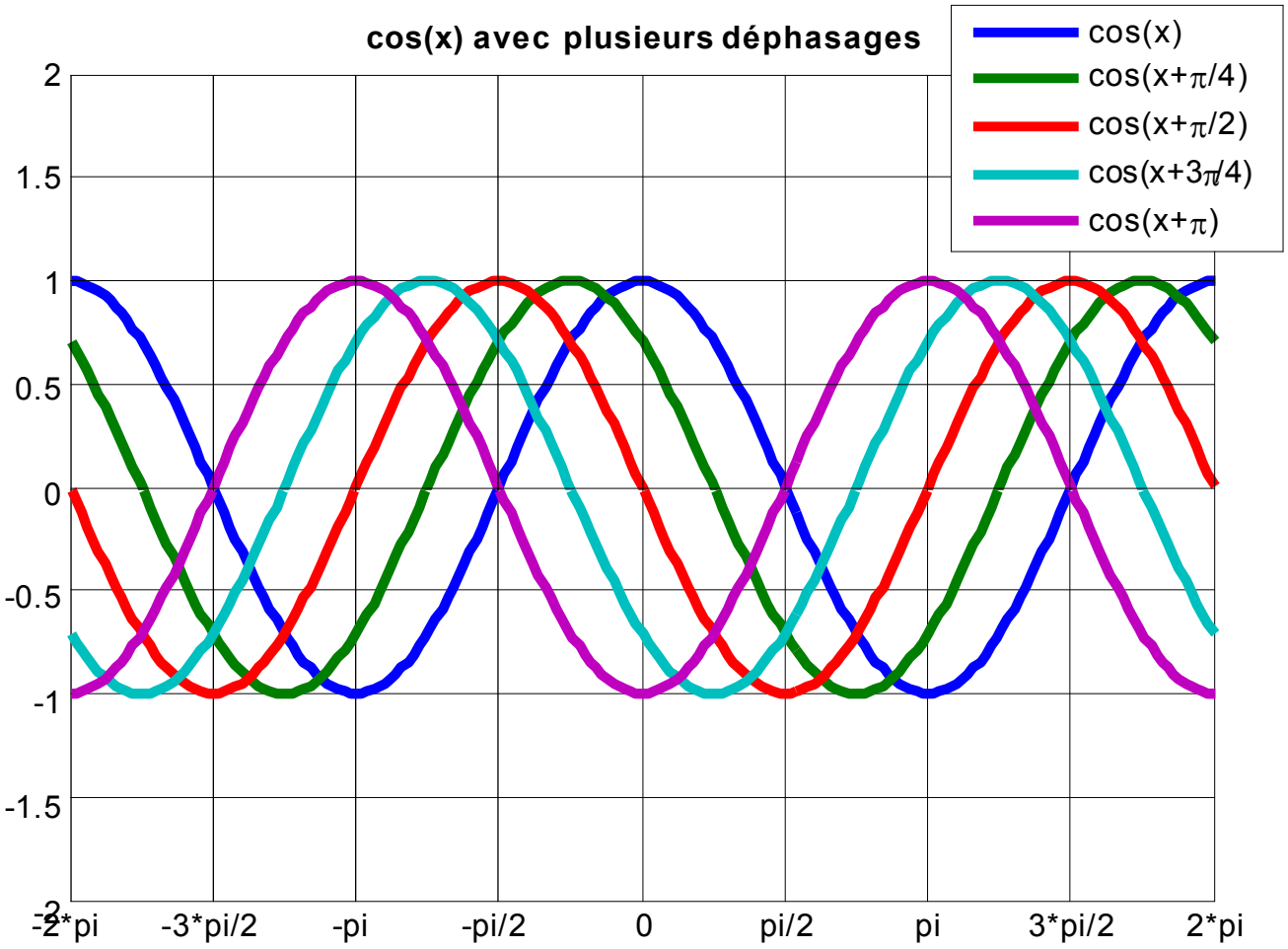
# Formats d'affichage et de lecture (15/20)

## Structure itératives for

```
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');  %  \bf permet de mettre le texte en gras %
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);
```

# Formats d'affichage et de lecture (16/20)

## Structure itératives for




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

## Structure itératives while

- 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 test sur une variable
        Commandes
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

```
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 |

## Opérateurs de comparaison

`==` : égale

`~=` : différent

`>` : strictement supérieur

`<` : strictement inférieur

`>=` : supérieur ou égale

`<=` : inférieur ou égale

## Opérateurs logiques

`&&` : et

`||` : ou

`~` : non