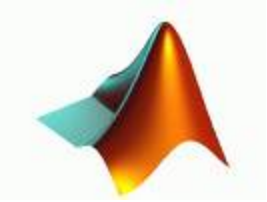


Formation
Initiation à Matlab

Coriandre Vilain

GIPSA-Lab

Janvier 2014

coriandre.vilain@gipsa-lab.fr

Demandez le programme!

1) Ouverture de Matlab
2) Matrices

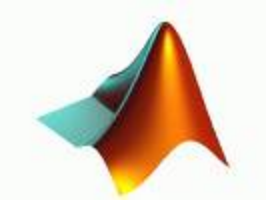
Séance 1

3) Chaînes de caractères
4) Visualisation graphique simple

Séance 2

5) Programmes et Fonctions
6) Bases de programmation
7) Gestion des fichiers textes

Séance 3



Formation Matlab

Premiers pas

Présentation de Matlab

Matlab (Matrix Laboratory) est un logiciel de calcul **matriciel** à syntaxe 'simple' (relativement à des langages évolués comme C, C++). MATLAB est un **interpréteur de commandes**: les instructions sont interprétées et exécutées ligne par ligne (pas de compilation avant de les exécuter).

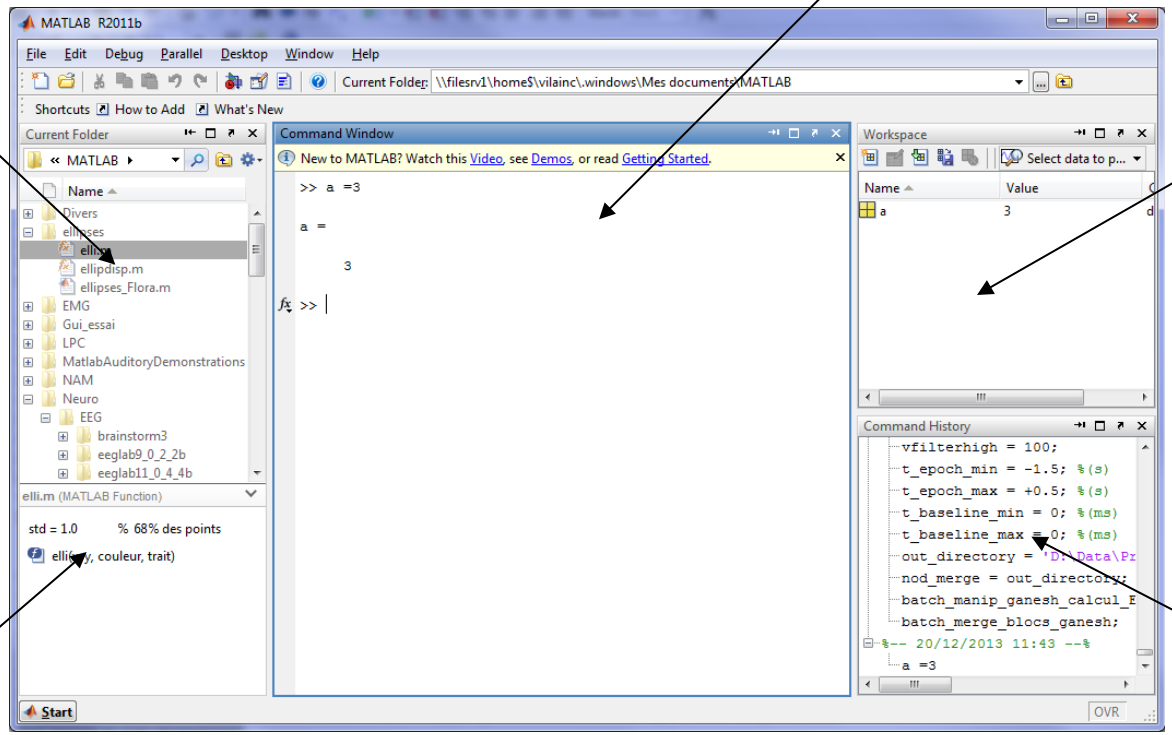
Il existe deux modes de fonctionnement:

- **mode interactif:** MATLAB exécute les instructions au fur et à mesure qu'elles sont données par l'usager.
- **mode exécutif:** MATLAB exécute ligne par ligne un fichier ".m" (programme en langage MATLAB).

1-Ouverture de Matlab (R2011b)

Fenêtre principale : **Editeur de commandes** (pour taper les instructions)

Arborescence
Du dossier en cours



Espace de travail : liste des variables déjà définies

Aide du fichier séléctionné (si elle existe)

Historique des dernières commandes

1-Ouverture de Matlab

- Les différentes fenêtres :
  - Dans la fenêtre d'édition de commande taper x = 1 et visualiser les conséquences sur la fenêtre d'espace de travail et d'historique de commande.

- Les variables
  - La variable **x** a été définie très simplement sans avoir à préciser le type de donnée qu'elle contient ni sa taille : **c'est un des grands avantages de MATLAB (mais parfois aussi un inconvénient. `**

Ex: pi = 18, sin = 3, affectation de valeurs à des variables déjà connues de Matlab au risque de se tromper par la suite dans l'interprétation**);**

  - Une variable est définie par sa classe (son type). Ex : double, char… Pour avoir une information sur une variable x : whos x
  - Pour effacer la variable x de l'espace de travail, taper clear x
  - Pour effacer toutes les variables de l'espace de travail taper clear all

2-Matrices

a) Définition d'une matrice

Sous matlab, les données sont généralement définies comme des **matrices**, ie des tableaux à 1, 2 … n dimensions. On ne considérera ici que des tableaux à 1 ou 2 dimensions.

Exemple :

- Si on tape A = 335, on définit une variable A correspondant à une matrice à 1 ligne et 1 colonne contenant le nombre 335.
- Si on tape A = [12 15 138], on définit une variable A correspondant à une matrice à 1 ligne et 3 colonnes contenant les nombre 12, 15 et 138 (On peut parler dans ce cas de vecteur). Les espaces entre les nombres permettent de délimiter les colonnes (on peut aussi utiliser des , pour cela) .
- Si on tape A = [1 3 5 ; 2 4 6], on définit une matrice A à 2 lignes et 3 colonnes. Les caractères ; permettent de passer à la ligne et les espaces délimitent les colonnes.
- Concaténation : si on a A = [1 2 3] et B = [4 5 6 7], on peut concaténer les 2 variables en une seule : C = [A B], on aura alors C = [1 2 3 4 5 6 7];

*NB : Si l'on ne souhaite pas voir afficher la valeur de la variable qu'on définit, il suffit de terminer la commande par le caractère « **;** ». La variable est alors bien définie (on le vérifie dans l'espace de travail) mais elle ne s'affiche pas. Si on veut vérifier la valeur de cette variable il suffit de taper le nom de cette variable sans « ; ».*

2-Matrices

Bon à savoir

- Pour créer un vecteur A (matrice à 1 ligne ou 1 colonne) contenant des nombres régulièrement espacés d'un pas p, commençant par le nombre u et finissant par v, il suffit de taper : A = u : p : v

Exemples:

- Vecteur commençant par 0 et allant jusqu'à 100 : A = 0:100
- vecteur commençant par 3 et finissant par 27 par pas de 3 : A = 3 :3 :27 (on aura alors A = [3 6 9 12 15 18 21 24 27]).

- Pour transformer un vecteur ligne en vecteur colonne ou inversement, il suffit d'utiliser le caractère ' (apostrophe) après le vecteur à transformer.

Exemple :

- Si on définit A = [1 4 2 6] (vecteur ligne), alors A' est égal à [1 ;2 ;3 ;4] (vecteur colonne).

***NB** : le caractère ' est en fait le symbole de transposition, il est applicable à une matrice quelconque. Si A = [1 2 3 ; 4 5 6 ; 7 8 9] alors A' = [1 4 7 ; 2 5 8 ; 3 6 9]*

2-Matrices

b) Eléments d'une matrice

- Chaque élément d'une matrice est accessible à condition de spécifier sa place dans la matrice. Pour cela, il suffit de donner le numéro de ligne et de colonne entre ().

Exemple :

- Si on a A = [1 5 2 6] alors A(1,2) = 5 (1ère ligne, 2ème colonne).
  NB : en fait ici, vu que la matrice A ne contient qu'1 seule ligne on peut se contenter d'écrire A(2) pour obtenir 5
- Si on a A = [1 5 6 ; 2 3 6] alors A(2,2) = 3 (2ème ligne, 2ème colonne).
- La variable **end** permet de récupérer les dernier élément d'un vecteur. Si on a A = [1 5 2 6] alors A(end) = 6

- Pour récupérer plusieurs éléments d'une matrice, il suffit de préciser l'ensemble des numéros de lignes et de colonnes des éléments à prélever. En particulier, pour récupérer l'ensemble des éléments d'une ligne ou d'une colonne on utilise le caractère ' :'.

Exemple :

- A = [1 2 4 ; 2 6 8 ; 3 5 4].
  - Pour récupérer les 2 premiers éléments de la 2ème ligne : A(2,[1 2]) = [2,6]
  - pour récupérer la 3ème ligne : A(3, :) = [3 5 4]
  - pour récupérer la 2ème colonne : A(:,2) = [2 ;6 ;5]

2-Matrices

c) Opération sur les matrices

- **Addition / soustraction :** 2 matrices de mêmes dimensions (même nombre de lignes et de colonnes) s'additionnent ou se soustraient **terme à terme**.

Exemple:

- A = [1 2 3 ; 4 5 6] , B = [2 4 6 ; 7 8 -1]
  A+B = [3 6 9 ; 11 13 5]
  A-B = [-1 -2 -3 ; -3 -3 7]

- **Multiplication ou division :** la multiplication ou division de matrice obéit à des règles spéciales qu'on ne détaille pas ici. En revanche on peut multiplier ou diviser 2 matrices **terme à terme** par l'emploi des symboles « .* » ou « ./ »

Exemple:

- A = [1 2 3 ; 4 5 6] , B = [2 4 6 ; 7 8 -1]
  A.*B = [2 8 18 ; 28 40 -6]
  A./B = [0.5 0.5 0.5 ; 0.57 0.62 -6]

2-Matrices

c) Opération sur les matrices

- fonctions trigonométriques : **cos, acos, sin, asin, tan, atan, cot, acot**
- fonctions logarithmiques : **exp, log** (log Népérien), **log10** (log de base 10)
- fonctions somme et produits : **sum, cumsum, prod, cumprod**
- génération de nombres aléatoires : **rand, randn …**
- Permutations aléatoires : **randperm**
- transformée de Fourier discrète : **fft, ifft …**
- Arrondis : **round, floor, ceil…**

NB : pour connaître la syntaxe d'une fonction : taper ***help nom_de_la_fonction***

Exemples :

- A = [1:10];

  sin(A) = [0.84 0.91 0.14 -0.76 -0.96 -0.28 0.66 0.99 0.41 -0.54]

  log(A) = [0 0.69 1.10 1.39 1.61 1.79 1.95 2.08 2.20 2.30]

  sum(A) = 55 (sum(A) = 1+2+3+4+5+6+7+8+9+10)

  cumsum(A) = [1 3 6 10 15 21 28 36 45 55] (**somme cumulée**)

- rand(1,10) = [0.95 0.23 0.61 0.49 0.89 0.76 0.46 0.02 0.82 0.44] (par exemple)
  (vecteur à 1 ligne et 10 colonnes obtenu par **tirage aléatoire** avec une **loi de probabilité uniforme** : chaque élément compris entre 0 et 1 a la même probabilité de sortir).

2-Matrices

d) Exercices

EX 1 :

La formule permettant de calculer rapidement la valeur de la somme des n premiers entiers naturels est la suivante : sn =1+2+3+4+…+n = n*(n+1)/2.

Vérifier cette formule pour différentes valeurs de n : n = 100, n = 100 000.

EX 2 :

1) Générer un vecteur **x** à 1 ligne et 30 colonnes rempli de 3 en utilisant la fonction **ones().**

2) Calculer la somme cumulée de x (fonction **cumsum()**) et l'affecter à la variable y.

3) Prélever un échantillon sur 9 de y et placer ces échantillons dans un vecteur **z.**

EX 3 :

1) Générer un vecteur **x** à 1 colonne et 1000 lignes rempli de nombres aléatoires distribués uniformément entre 0 et 1 en utilisant la fonction **rand()**

2) Calculer la moyenne et l'écart type du vecteur x en utilisant **mean()** et **std()**

3) Créer un générateur de 'lancer de dé' ie un générateur de nombre aléatoire **entier** compris entre 1 et 6 avec une probabilité uniforme (avec **rand()** et **ceil()** par exemple)

3-chaînes de caractères

a) Définition d'une chaîne de caractère

- Les chaînes de caractères sont des matrices de caractères!

- Pour définir une chaîne de caractère on utilise les apostrophes

  Exemple

  - nom = 'dupont';
  - phrase = 'ceci est une phrase';
  - rg = ['dupont';'dupond']

3-chaînes de caractères

b) Eléments d'une chaîne de caractères

- Pour récupérer certains caractères d'une chaîne de caractères, il suffit de préciser les indices des numéros de lignes et de colonnes correspondant.

Exemple:

```
Nom_du_capitaine = 'Archibald Haddock';
```

Pour prélever son prénom et le mettre dans la variable prenom_du_capitaine, on peut faire

```
Prenom_du_capitaine = nom_du_capitaine(1:9);
```

On aura alors prenom_du_capitaine = 'Archibald'

NB : D'autres possibilités existent si on ne connait pas à l'avance la longueur de la chaîne de caractère correspondant au prénom. On peut par exemple rechercher tous les caractères situés avant l'espace situé entre le prénom et le nom. On verra plus loin!

3-chaînes de caractères

c) Opération sur les chaînes de caractères

- **Concaténation :** pour concaténer 2 chaînes de caractères, on peut utiliser les symbole [];

Exemple:

```
a = 'Tryphon'
b = ' fait le zouave'
c = [a b];
```

Que croyez vous qu'il arrive si on affiche la valeur de c?

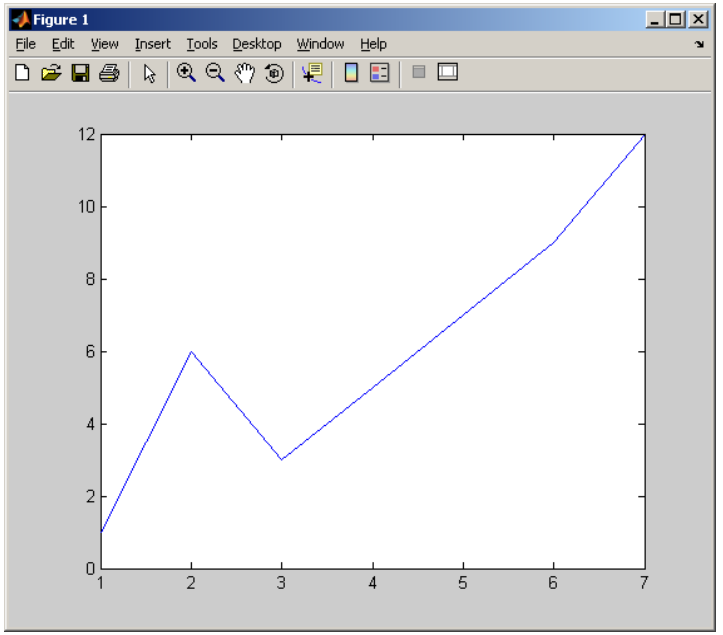
- **Transposition :** on peut transposer une chaîne de caractères avec le symbole « ' »

NB : *d'autres opérations sont possibles mais nous les détaillerons dans la partie avancée.*

4-Visualisation graphique simple

a) Affichage des valeurs d'un vecteur

- Pour afficher les valeurs d'un vecteur A = [1 6 3 5 7 9 12], il suffit de taper plot(A). On obtient la figure ci-contre.



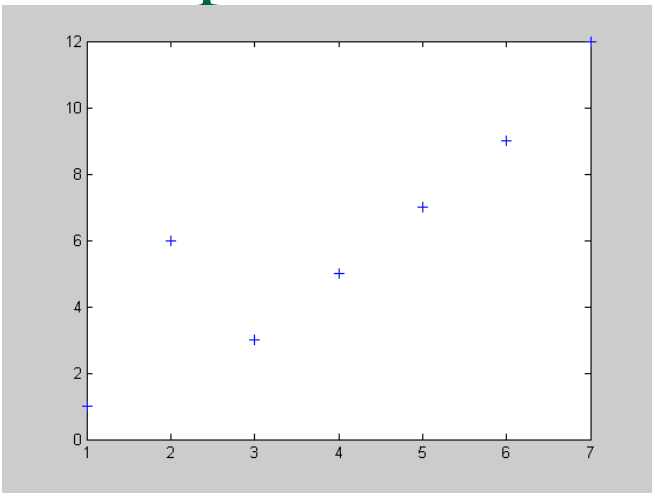
En abscisse, la valeur représentée correspond au **numéro de l'échantillon**.

En ordonnée, la valeur représentée est l'élément de A associé au numéro d'échantillon mentionné en abscisse

NB : Par défaut, matlab relie les points affichés par des lignes. On peut alors avoir **l'impression trompeuse d'un signal continu alors que seules les valeurs des échantillons sont connues (signal discret)!!!**

4-Visualisation graphique simple

- On peut s'affranchir du problème en représentant chaque point par un marqueur.

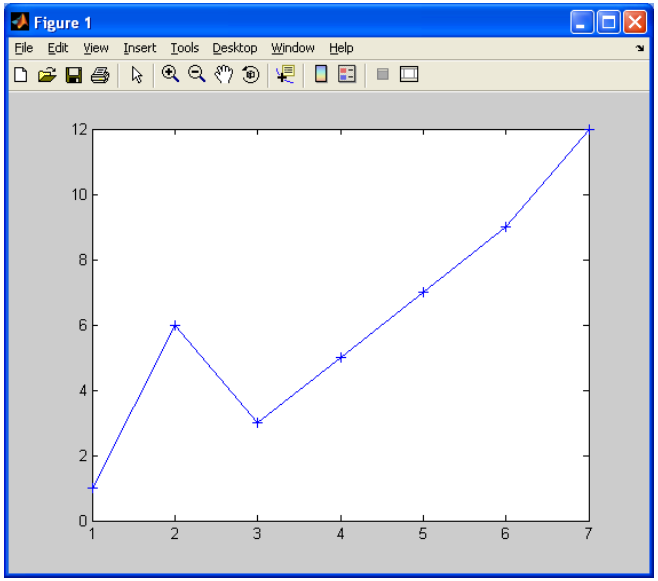


Exemple :

plot(A,'+') pour utiliser un marqueur de type '+'

plot(A,'-+') pour utiliser un marqueur de type '+' ajouté à une ligne continue

NB : on peut aussi utiliser le gestionnaire de figure de Matlab en sélectionnant la flèche de saisie d'objet dans les icones situés au dessus de la figure puis en faisant un click droit sur la figure puis *Show Property Editor*



4-Visualisation graphique simple

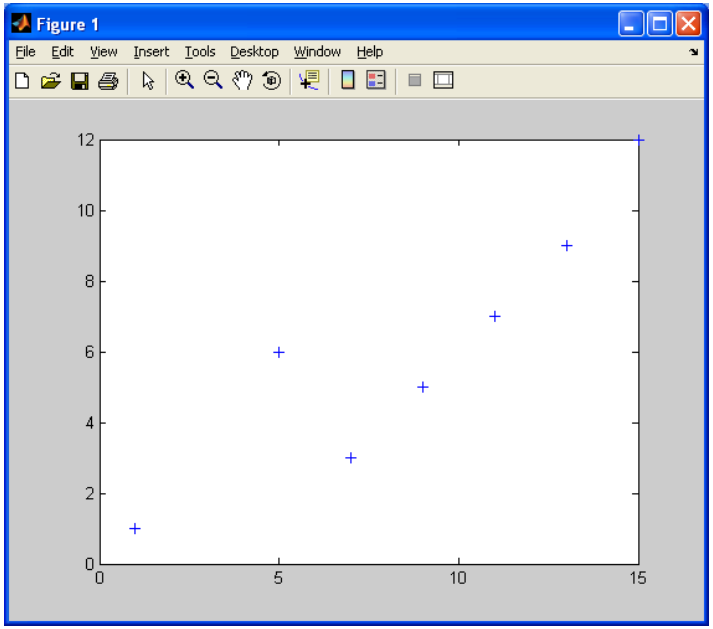
b) Affichage des valeurs d'un vecteur en fonction d'un autre vecteur

- Si on veut afficher un vecteur A en fonction d'un autre vecteur X, il suffit de taper : plot(X,A).

Matlab interprète le premier vecteur comme celui correspondant à l'axe des abscisses et le second vecteur comme celui correspondant à l'axe des ordonnées.

- Exemple :

```
X = [1 5 7 9 11 13 15],
A = [1 6 3 5 7 9 12],
plot(X, A,'+')
```



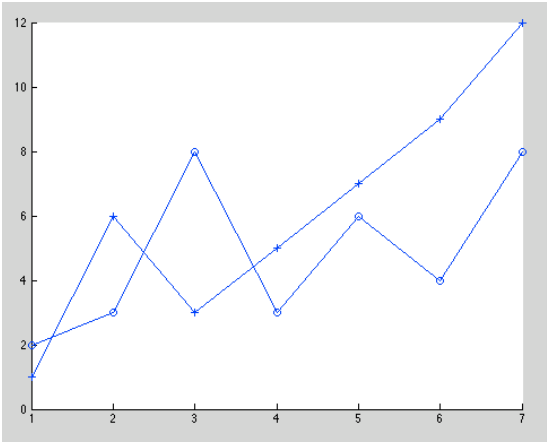
4-Visualisation graphique simple

c) Affichage de plusieurs vecteurs

- Si on veut afficher un vecteur A et un vecteur B sur une même fenêtre, il faut activer le mode **hold** dans l'éditeur de commande en tapant: hold on (par défaut, ce mode est à off).

Exemple :

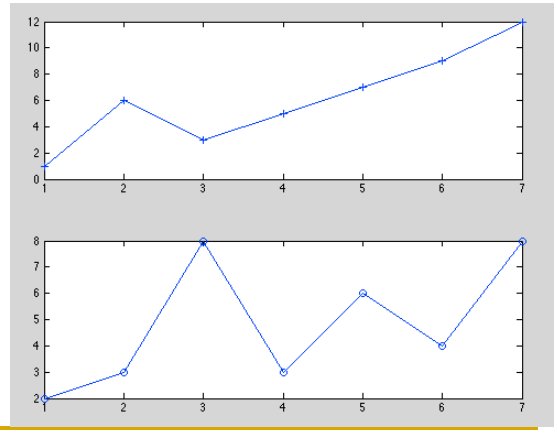
```
A = [1 6 3 5 7 9 12];
B = [2 3 8 3 6 4 8];
hold on
Plot(A,'+-'), plot(B,'o-')
```



- Si on veut afficher un vecteur A et un vecteur B sur une fenêtre séparée en 2 sous fenêtres, il faut utiliser la commande subplot.

Exemple :

```
A = [1 6 3 5 7 9 12];
B = [2 3 8 3 6 4 8];
subplot(2,1,1), plot(A,'+-')
subplot(2,1,2), plot(B,'o-')
```



4-Visualisation graphique simple

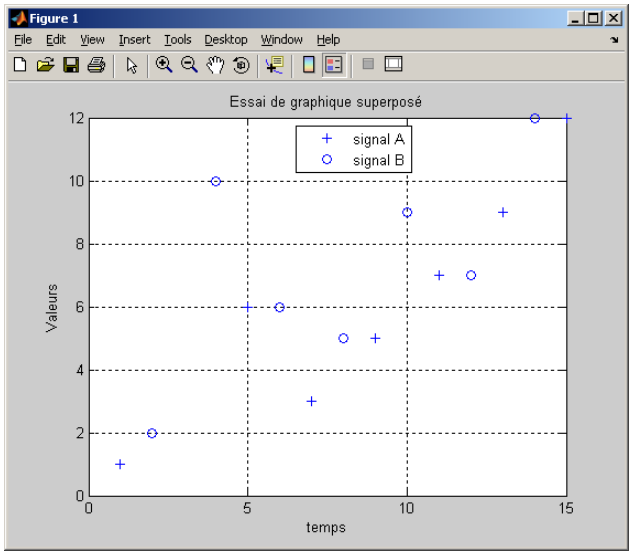
c) Labels, grilles et légendes :

En mode de commande, on peut paramétrer les propriétés d'un graphique :

- pour mettre un label sur l'axe des x : **xlabel**('texte à afficher')
- pour mettre un label sur l'axe des y : **ylabel**('texte à afficher')
- pour mettre un titre à la figure : **title**('texte à afficher')
- pour rajouter un quadrillage : **grid** on
- pour afficher une légende : l**egend**('courbe 1','courbe 2'…)

Exemple :

```
X = [1 5 7 9 11 13 15] ;
A = [1 6 3 5 7 9 12] ;
plot(X, A,'+');
hold on
Y = [2 4 6 8 10 12 14] ;
B = [2 10 6 5 9 7 12] ;
plot(Y,B,'o');
xlabel('temps')
ylabel ('Valeurs')
title('Essai de graphique superposé')
grid on
legend('signal A', 'signal B') ;
```



4-Visualisation graphique simple

d) Création, fermeture des figures

- Pour créer une nouvelle figure : **figure** ou **figure**(num_de_figure)

- Pour effacer la figure courante (en la laissant ouverte): **clf** (clear figure)

- Pour fermer une figure :
  - close : efface la figure courante
  - close(num_de_figure) : ferme la figure de num_de_figure
  - close all : ferme toutes les fenêtres ouvertes

  NB : A sa création, une figure est associée à un identifiant (handle) défini par matlab. Pour récupérer l'identifiant de la figure courante : gcf

4-Visualisation graphique simple

e) Enregistrement des figures

- Pour sauver une figure, il existe deux possibilités:

1. Mode interactif : aller dans la fenêtre associée à la figure et cliquer sur *>fichier>enregistrer sous* puis rentrer le type de format de sortie de la figure:
   - Format graphique simple (bmp, jpg, tiff…)
   - Format matlab (.fig) qui permettra de retravailler la figure à posteriori si besoin

2. Mode commande : rester dans la fenêtre d'éditeur de commande et taper saveas(gcf, 'nom_du_fichier', 'type_de_fichier_de_sortie').

Exemple: saveas(gcf, 'figure_cv','fig') permet de sauver la figure courante (gcf) en format Matlab

4-Visualisation graphique simple

f) Exercice :

EX 4 :

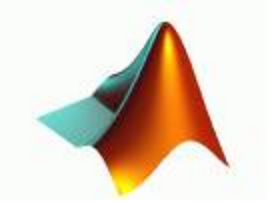
Visualisation d'un signal sinusoïdal de fréquence f = 100 Hz, échantillonné à 20 kHz pendant 10 secondes

1 - Générer un vecteur t contenant les dates en secondes des échantillons d'un signal de durée 10 sec et échantillonné à 20 kHz.

2 - Calculer les valeurs du signal sinusoïdal grâce à la formule : y = sin(2*pi*f*t).

3 - Afficher le signal y en fonction du temps en marquant d'une croix les valeurs des échantillons.

4 - Placer sur le graphique un titre, des labels d'abscisses et d'ordonnée.



Formation Matlab

Deuxième séance

Récapitulatif de la leçon 1:

- Suite à la leçon 1 vous devez savoir :
  - Générer un vecteur allant de 1 à 100 par pas de 2 : A = 1:2:100
  - Prélever dans un vecteur un échantillon tous les 3 échantillons : B = A(1:3:end);
  - Additionner 2 matrices A et B de mêmes dimensions : C = A+B
  - Multiplier 2 matrices A et B de même dimension **terme à terme** : C = A.*B;
  - Transposer une matrice : B = A';
  - Gérer de façon simple les chaînes de caractères
  - Génération et manipulation de graphiques
- La leçon 2 va aborder les thèmes suivants :
  - Programmes et fonctions
  - Bases de programmation

5-Programmes et fonctions

a) Programmes

- Un programme matlab est une suite d'instructions enregistrées dans un fichier texte ayant l'extention '.m'.
- Dans un programme, chaque instruction est effectuée **ligne par ligne** comme si elle était tapée dans la fenêtre d'édition de commande.
- Toutes les variables utilisées par le programme sont disponibles dans la fenêtre d'éditeur de commande.

5-Programmes et fonctions

a) Programmes

- Pour exécuter un programme, Matlab doit pouvoir le localiser. Pour cela il existe la variable d'environnement path. C'est l'ensemble de tous les emplacements des dossiers susceptibles de contenir des fichiers « .m ».

- Par défaut, Matlab cherche si le programme à exécuter est situé dans le répertoire courant. S'il ne le trouve pas, il cherche dans tous les dossiers spécifiés dans la variable path. S'il ne le trouve nulle part, il affichera un message d'erreur de type : ***file not found***

- Pour ajouter un répertoire dans la liste du path on peut utiliser la commande addpath()

5-Programmes et fonctions

a) Programmes

- La structure d'un programme devrait toujours être grosso-modo la suivante

```
% toto.m
%-----------------------------------------
% Description du programme
%   blablabla
% -----------------------------------------
% Variables d'entrée:
%   Mettre ici les variables requises par le programme si besoin
% Variables de sortie
%   Mettre ici le nom des variables générées par le programme si besoin
%-----------------------------------------
% Auteur(s), date de programmation, dates de modifs

Lignes de programme
% commentaires sur ce que fait le programme
Lignes de programme
                ...
```

NB : Les lignes commençant par « % » sont des **commentaires** non interprétés par Matlab mais **fort utiles** pour comprendre la structure du programme.

Les premières lignes de commentaires sont affichées si on tape dans la fenêtre d'édition de commande: **help nom_du_programme.**

5-Programmes et fonctions

a) Programmes

Exemple de programme:

Calcul de la moyenne et de l'écart-type d'un vecteur donné.

1. Ouvrir l'éditeur de matlab
2. Ecrire le programme suivant

```
% calc_moy_ec_type
% programme calculant la moyenne
% et l'écart type d'un vecteur x
% Input :
%   * x : vecteur dont on veut connaître la moy, et l'et.
% Output:
 %   * moy, ec_typ : moyenne et écart type de x
%  C.V. le 10/10/06

moy = mean(x);
ec_typ = std(x);
```

3. Sauver le programme sous le nom que vous voulez, ex: calc_moy_et.m
4. Visualier l'aide de ce programme (help calc_moy_et)
5. Exécuter le programme: calc_moy_et après avoir défini le vecteur x

5-Programmes et fonctions

b) Fonction

- Une fonction matlab ressemble à un programme matlab. Deux différences cependant :
  - La fonction accepte des variables d'entrée et de sortie lors de son appel dans l'éditeur de commande
  - Les variables utilisées dans la fonction sont indépendantes des variables utilisées dans l'éditeur de commande (on parle de variable locales).

Exemple de fonction : Calcul de la somme de 2 nombres

```
function y = somme (a,b)
%function y = somme (a,b)
% calcule la somme de a et b
y=a+b
```

- Une fonction est souvent utilisée pour réaliser un calcul court, utilisé par un autre programme
- Une fonction commence **toujours** par une ligne du type

  function [Variables de sortie] = nom_de_la_fonction([Variables d'entrée])

5-Programmes et fonctions

b) Fonction

- La structure d'une fonction devrait toujours être grosso-modo la suivante

```
function [Variables de sortie] = toto ([Variables d'entrée])
% function [Variables de sortie] = toto ([Variables d'entrée])
% -----------------------------------------
% Description de la fonction
%   blablabla
% -----------------------------------------
% Variables d'entrée:
%   Mettre ici les variables requises par la fonction si besoin
% Variables de sortie
%   Mettre ici le nom des variables générées par la fonction si besoin
%-----------------------------------------
% Auteur(s), date de programmation, dates de modifs

Lignes de programme
% commentaires sur ce que fait la fonction
Lignes de programme
```

- L'appel de la fonction dans l'éditeur de commande se fait alors par

  [var_out1, var_out2,…] = toto (var_in1, var_in2,…)

5-Programmes et fonctions

Exemple de fonction:

Calcul de la moyenne et de l'écart-type d'un vecteur donné.

```
function [moy,ec_typ] = calc_moy_ec_typ(x)
moy = mean(x);
ec_typ = std(x);
```

1. Enregistrer la fonction ci-dessus (par ex: calc_moy_et_func.m)
2. Dans l'éditeur de commande, créer un vecteur **u** dont on veut calculer la moyenne et l'écart type. Exemple : u = 1:10;
3. Taper dans l'éditeur de commande : [**m**, **et**] = calc_moy_ec_typ(**u**).

   *Dans ce cas, la variable **m** prendra le résultat de la moyenne calculée dans la fonction sous le nom de variable locale **moy**. La variable ec prendra le résultat de l'écart type calculé sous le nom de **ec_typ.***
4. Visualiser la valeur des variables x, moy et ec_typ

   *Les variables x, moy et ec_typ sont **locales.** Elles n'existent pas pour l'éditeur de commande. Seules existent les variables **m** et **et***

Exercices récapitulatifs

EX 5

Renommage de nom de fichier :

Soit la variable nom_fich = 'fichier_1.txt';

a) Définir une variable contenant le nom du fichier sans son extension

b) Ajouter à cette variable le suffixe '_new.txt' par concaténation de chaîne de caractère.

c) Générer une fonction *change_extension* qui accepte en variable d'entrée des chaînes de caractère de type nom_de_fichier.extension et qui transforme automatiquement le nom de l'extension (à 3 caractères) en « dat ». La valeur de la sortie étant alors nom_de_fichier.dat.

EX 6

Gestion de matrices de chaines de caractères

Générer une variable nom_fich contenant sur 3 lignes 3 noms de fichiers : toto_1.txt, toto_2.txt, toto_3.txt.

Que se passe t'il si l'on y concatène la chaîne 'toto_10.txt'?

6-Bases de programmation

a) Les expressions logiques (ou expressions booléennes).

En français :	Sous Matlab :
« Est ce que A est égal à B »	A==B (noter le double =)
« Est ce que A est supérieur à B »	A>B (noter le simple >)
« Est ce que A est un vecteur vide »	isempty(A)
…	…
Expressions dont la réponse est Oui ou Non	Expressions dont la réponse est 1 (oui) ou 0 (non)

Exemples :

- 1) a = 2, a == 1
- 2) a = [1 2 3 1 4 6], a==1
- 3) a = 'toto', a == 'titi'
- 4) ~~a = [1 2], a== [1 2 3]~~

NB : Les termes à comparer doivent être de même taille (cas 1 et 3) sinon erreur (cas 4). Exception si l'un des termes est un vecteur de taille 1, dans ce cas Matlab se débrouille (cas 2).

6-Bases de programmation

Principales fonctions utilisant des expressions logiques :

- If … else …end

```
If expression logique
    effectuer actions 1
else
    effectuer actions 2
end
```

L'expression logique a la valeur 0 ou 1:

- si elle vaut 1 alors seule l'action 1 est effectuée,
- sinon, seule l'action 2 est effectuée.

Exemple:

```
rep = input('Réponse 1 ou 2?');
if rep == 1
   disp('Vous avez tapé la réponse 1');
else
  disp('vous avez tapé la réponse 2') ;
end
```

6-Bases de programmation

- **find :** find(Z) donne les valeurs des numéros d'échantillons du vecteur Z différents de 0

  En particulier, si Z est le résultat d'une opération logique (ie un vecteur de 0 ou de 1) , find(Z) donne les numéros d'échantillons pour lesquels cette opération est vraie (ie pour lequel Z =1).

Find est très utilisée pour détecter des seuils sur un signal.

Exemple :

A = [1 2 4 5 6]

find(A==4) donne le résultat 3 *(c'est l'échantillon associé au cas où la condition A==4 est vraie)*

find (A>2) donne le résultat [3 4 5]; *(même principe avec A>2)*

ech_seuil = min(find (A>2)) donne le résultat 3, *(c'est le numéro du 1er échantillon dont la valeur est au dessus du seuil)*

6-Bases de programmation

- **Strfind:** strfind(s, 'chaîne à chercher dans s') donne la valeur de l'échantillon de s correspondant au début de la chaîne cherchée

Exemple :

- s = 'toto est un sot';

  strfind(s,'sot') donne le résultat 13

  strfind(s,'titi') donne le résultat [] (ensemble vide)

- Nom_du_capitaine = 'Archibald Haddock';

Pour prélever son prénom et le mettre dans la variable prenom_du_capitaine, on peut aussi faire :

```
Prenom_du_capitaine = nom_du_capitaine(1:strfind(nom_du_capitaine,' ')-1);
```

On aura alors prenom_du_capitaine = 'Archibald'
Ici on n'a pas besoin de connaître le nombre de lettres exactes du prénom ou du nom. On prend tout ce qui est avant l'espace pour le prénom.

6-Bases de programmation

b) Les boucles

Elles permettent de répéter un ensemble d'actions pendant un nombre de fois prédéfini.

Sous matlab, il existe 2 types de boucles :

- La boucle for…end
- La boucle while …end

6-Bases de programmation

- La boucle for ***variable = expression*** … end

Elle permet de répéter une action utilisant une variable spécifiée après le for.

Ex :
```
for i = 1:3
        tâche 1
        tâche 2
        …
end
```

Les tâches peuvent faire appel (ou pas) à la variable i qui vaudra ici :

- 1 lors du 1er passage de la boucle,
- 2 lors du 2nd passage
- 3 lors du dernier passage

Exemple :

Pour calculer la somme des éléments d'un vecteur x on peut calculer:

```
som = 0; % initialisation
for i = 1:length(x)
    som = som+x(i);
end
```

6-Bases de programmation

- La boucle while ***expression logique*** … end

Elle permet de répéter une action tant que l'expression logique est vraie.

```
while i<10 & j>10
        tâche 1
        tâche 2
        …
end
```

Les tâches seront effectuées tant que i sera inférieur à 10 **et** j sera supérieur à 10

Exemple : Pour calculer la somme du vecteur x comme précédemment

```
som = 0; % initialisation
fin = 0; % variable logique qui nous servira à arrêter la boucle
While not(fin)
    som = som+x(i);
    If i < length(x)
        i = i+1;
    else
        fin = 1; % pour arrêter la boucle
    end
end
```

6-Bases de programmation

EX 7

Boucle for et nombre d'or….

1. Générer un vecteur A de taille 100 dont chaque échantillon A(i) est égal à la somme des 2 échantillons précédents et que A(1) = A(2) = 1:

   A(1) = 1,

   A(2) = 1.

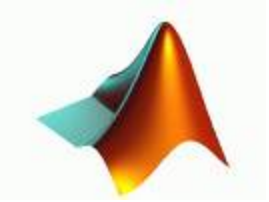
   A(i) = A(i-1)+A(i-2).

2. Afficher sur un graphique la variable r définie par r(i) = A(i)/A(i-1);

EX 8

Boucle while

Demander à l'utilisateur de donner un nombre entre 0 et 10 (en utilisant la fonction **input** et le faire recommencer tant que le nombre donné est différent de 8)



Formation Matlab

Troisième séance

- Suite à la leçon 2 vous devez savoir :
  - Effectuer des opérations logiques sur des nombres ou des chaînes de caractère
  - Générer des programmes et des fonctions utilisant des boucles for… end ou while… end
- Programme de la 3ème séance :
  - Gestion des fichiers textes
  - Opérations avancées sur les chaînes de caractères
  - Travaux Pratiques

7-Gestion des fichiers textes

a) Introduction

- Un fichier texte est une suite de caractères enregistrés sur le disque dur de l'ordinateur.
- Matlab utilise le code ASCII ISO LATIN-1 pour coder chaque caractère: un caractère est codé sous la forme d'un nombre de 2 octets (donc choisi parmi 2^16 possibilités).
- Sous Matlab, la correspondance entre un caractère et son code ascii se fait simplement par les commandes **double** ou **char**.

- Exemple:
  - le code ascii du caractère 'a' est donné par double('a').
  - le caractère « a » correspondant au code ascii 98 est obtenu par char(98).

7-Gestion des fichiers textes

b) Retour sur la gestion des chaînes de caractères

- formatage de chaînes :

  En plus de la concaténation de chaînes avec [] ou **strcat()**, il existe des outils qui permettent d'intégrer dans une chaîne de caractères des données de tout types (nombres à virgule flottante, entier,…). C'est ce que permet la commande **sprintf.**

- Ecriture de chaînes formatées avec sprintf :

  sprintf s'utilise de la manière suivante :

  A = sprintf('texte incluant les données d1 : *%format-de-d1* et d2 : *%format-de-d2* et des caractères spéciaux comme le retour à la ligne \n',*valeur de d1, valeur de d2)*

NB : Pour qu'une ' soit incluse dans le texte il faut la doubler, sinon matlab l'interprète comme la fin de la ligne de texte et affiche une erreur.

```
A = sprintf('l''âge du capitaine est %d ans.\n',55);
```

7-Gestion des fichiers textes

- formatage de chaînes (suite):

Le type de données intégrées à la chaîne est précisé avec le symbole %. On peut citer par exemple :

- %f : float (plus généralement **%a.bf** avec a = nombre minimum de chiffres à afficher, b: nombre de décimales à afficher)
- %d : entier (plus généralement **%.ad** avec a = nombre minimum de chiffres à afficher, remplit de 0 si l'entier contient moins de chiffre que a)
- %c : char
- %s : string of char

- Exemple :
  - a = sprintf('0+0 = %f',3)
  - a = sprintf('0+0 = %s','la tête à toto')
  - a = sprintf('toto a %d amis qui s''appellent %s et %s',2, 'titi', 'tutu')

NB: '' le caractère % étant réservé au formatage, si l'on souhaite l'inclure dans une chaîne formatée, il faut utiliser %%

7-Gestion des fichiers textes

- formatage de chaînes (fin):

  En plus des données formatées utilisant le caractère %, on peut inclure dans les chaînes formatées certains 'caractères' spéciaux comme le retour à la ligne, la tabulation… :
  - **\t : tabulation**
  - **\n : nouvelle ligne**
  - **\r : retour charriot**

- Exemple :
  - sprintf('le résultat est \n\ta = 2\n\tb=2')

7-Gestion des fichiers textes

- Lecture de chaînes formatées :

Pour lire des chaînes contenant différents type de données on peut utiliser la fonction **strread.**

Supposons qu'une chaîne de caractères A contienne les données suivantes : *nom_de_fichier valeur_1 valeur_2* avec :

- nom_de_fichier : chaîne de caractères,
- valeur_1 : float
- valeur_2 : entier

On peut alors récupérer les valeurs de nom_de_fichier, valeur_1 et valeur_2 de A par la commande strread : [nom_fich, v1, v2] = strread(A,'%s%f%d')

Exemple :

a = 'fichier1.txt 2.34 4'; (équivalent à a = sprintf('%s %f %d', 'fichier1.txt', 2.34, 4))

[nom_de_fichier,val1,val2] = strread(a,'%s%f%d')

NB : la variable nom_de_fichier n'est pas de format char mais de format cell (cellule). Pour la convertir en char il faut faire : nom_de_fichier = char(nom_de_ficher)

7-Gestion des fichiers textes

c) Ouverture/ Fermeture d'un fichier texte

- Ouverture en **lecture** d'un fichier nommé nom_fich.txt :

  fid1 = fopen('nom_fich.txt','rt')

  Le fichier nom_fich.txt est ouvert en lecture.

  NB : fid1 est l'identifiant du fichier texte ouvert. On peut ouvrir simultanément plusieurs fichiers (en lecture et/ou écriture à condition de définir pour chacun un nom d'identifiant différent.

- Ouverture en **écriture** d'un fichier nom_fich.txt :

  fid2 = fopen('nom_fich.txt','wt')

  Le fichier nom_fich.txt est ouvert en écriture

- Fermeture du fichier : fclose(fid).

Penser à toujours fermer un fichier en fin de manipulation!!!

7-Gestion des fichiers texte

d) Lecture d'un fichier texte

- La lecture d'un fichier se fait de façon cursive : les caractères ou les lignes de caractères sont lus à la suite les uns des autres.

- Lecture d'un caractère :
  - fread(fid,1,'char') : donne le code ascii du caractère courant
  - fscanf(fid,'%c',1) : donne le caractère courant

- Lecture d'un ensemble de n caractères
  - fread(fid,n,'char)
  - fscanf(fid,'%c',n)
  - fgetl(fid) : Lecture de la ligne courante du fichier

7-Gestion des fichiers texte

d) Lecture d'un fichier texte

Test pour savoir si le curseur est à la fin du fichier : feof(fid)

feof(fid) = 1 si l'on a atteint la fin du fichier
feof(fid) = 0 sinon

Exemple :

Lecture de chaque ligne d'un fichier jusqu'à la fin du fichier

```
while not(feof(fid))
        l = fgetl(fid);
end
```

7-Gestion des fichiers texte

d) Lecture d'un fichier texte

- Lecture d'une suite de caractères et/ou de nombres disposés en colonnes :
  - **load** : Si les données ne contiennent que des nombres, on peut utiliser la commande load : S = load('fichier.txt');
  - **textread :** Si les données sont des colonnes de nombres et de chaînes de caractères, on peut utiliser la commande textread

Exemple :

Si le fichier toto.txt contient 3 colonnes de données séparées par un espace :

nom (chaîne de caractères) val1 (float) val2 (int), On peut récupérer ces données de la façon suivante :

```
[nom,v1,v2] = textread('toto.txt','%s%f%d')
```

7-Gestion des fichiers texte

d) Lecture d'un fichier texte

EX 9

Créer avec le bloc-note un texte simulant une suite de données de différents types séparées par des tabulations :

- nom de famille (chaîne de caractères)
- Age (entier)
- note (nombre flottant)

Récupérer ces informations sous Matlab à l'aide des commandes d'ouverture et de lecture de fichiers textes

7-Gestion des fichiers texte

e) Ecriture d'un fichier texte

- L'écriture d'un fichier se fait de façon cursive : les caractères ou les lignes de caractères sont écrits à la suite les uns des autres.

- Ecriture d'une chaîne de caractères formatée :
  - fprintf(fid,' toto est un sot \n')
  - fprintf(fid,'%s %f %f',nom_de_fichier,val1, val2)

7-Gestion des fichiers texte

e) Ecriture d'un fichier texte

EX 10 : Ecriture de fichier texte

Reprendre le fichier texte généré pour l'exercice 9. Lire l'ensemble des notes de chaque élève pour les augmenter d'un point et écrire le nouveau fichier contenant le nom, l'âge et la note modifiée.

EX 11 : Lecture de fichiers Texgrids

Cf annexes