Chapitre 0
Introduction à Matlab

Plan du cours

1. Présentation de Matlab
2. Syntaxe, les premiers pas
3. Calcul matriciel
4. Graphiques 2D et 3D
5. Fonctions et boucles

Présentation

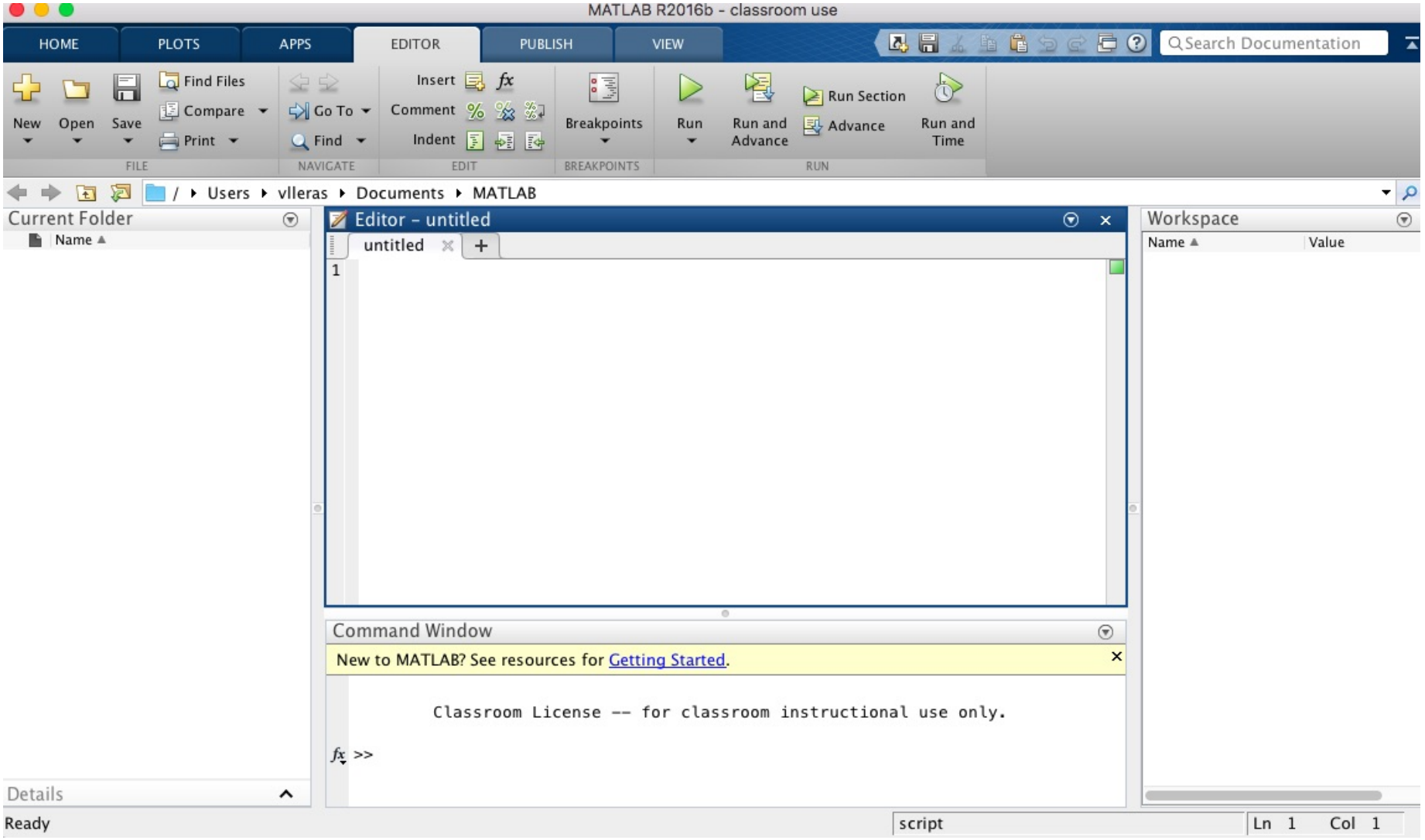
Matlab signifie MATrix LABoratory :

- C'est un logiciel de calcul matriciel numérique, langage de programmation pour le calcul scientifique.
- Il y a la librairie graphique intégrée
- Des fonctions de haut niveau sont prédéfinies : fonctions mathématiques usuelles, fonctions plus spécifiques du signal...
- Il est basé sur la représentation matricielle des données.
- Il fonctionne dans plusieurs environnements tels que Linux, Windows, Macintosh.

Matlab est un interpréteur de commandes (par opposition aux langages compilés tels que C ou fortran) que l'on peut utiliser :

- **en mode interactif** : les instructions sont exécutées au fur et à mesure de leur entrée dans la fenêtre de commande.
- **en mode exécutif** : les instructions sont lues dans un fichier `fichier.m` et exécutées une à une par l'interpréteur de commande.

Présentation



Présentation

L'environnement de travail Matlab est composé de plusieurs sous-parties :

— **La fenêtre de commandes** est repérée par le prompt >>. C'est après le prompt qu'il faut taper les commandes ou éxécuter les fonctions. Pour taper une nouvelle commande on le fait à la suite, on ne peut pas revenir en arrière.
Si le symbole >> ne revient pas à la fin d'une instruction, c'est qu'il y a un problème et Matlab donne des indications sur l'origine et le moyen de le résoudre.
Exemple : `>> a=1, b=3`
Pour effacer une variable b, on utilisera `clear b`

— **L'onglet Command History** est visible par défaut et indique les dernières commandes effectuées.
Matlab conserve l'historique des commandes. Il est donc possible de récupérer des instructions déjà saisies (et ensuite de les modifier dans le but de les réutiliser)

Présentation

— **L'espace de travail** est un espace de la mémoire vive où sont stockés les tableaux de nombres que nous utilisons. La fenêtre `workspace` recense toutes les variables assignées et donne différentes informations : taille en mémoire, valeur minimale et maximale des tableaux.
On peut également obtenir la liste des variables en tapant la commande `who` ou `whos` pour l'affichage détaillé.
Si vous quittez Matlab, vous perdez toutes les variables définies sauf si vous les avez enregistrer dans un fichier .mat.

— **Le Current Directory** gère les fichiers et sera utile pour les m-files. A chaque début de session, penser à parcourir les dossiers jusqu'au dossier où se trouvent vos programmes.

Présentation

— **L'aide en ligne** est l'outil de base de Matlab. On peut y accéder dans le menu déroulant `help` ou par ligne de commande :
`>> help plot` pour avoir de l'aide sur la fonction plot.
Pour voir comment utiliser l'aide : `>> help help`
Pour ouvrir la fenêtre d'aide : `>> helpwin`

Matlab à la maison

Pour travailler sur Matlab chez soi, voici quelques solutions :

- Acheter une version étudiante de Matlab
- En trouver une sur Internet...
- Aller sur les ordinateurs de la fac
- Installer le logiciel octave (https ://www.gnu.org/software/octave) qui est une sorte de clone de Matlab un peu plus lent et il y a quelques variantes mais très peu.

Syntaxe

— Les nombres réels peuvent être écrits sous différents formats : `-235.07`, `0.5E-10`
— Les nombres complexes peuvent être écrits sous forme cartésienne (`2.5+9.7*i`) ou polaire (`1.25*exp(i*0.246)`)
— On attribue une valeur numérique à une variable en tapant directement son expression. Le symbole d'affectation de valeur à une variable est le signe `=`.
`>> a=3.2`
— ; en fin de ligne permet de ne pas afficher le résultat de l'instruction. `>> a=3.2;`
— la variable `ans` stocke le résultat de la dernière commande
— % permet de commenter
— On peut afficher un message : `>> disp('hlma310')`
— On peut demander à entrer une valeur avec le clavier : `x=input('entrer une valeur de x')`
— Pour voir le temps d'éxécution, on met `tic` au début du programme et `toc` à la fin.
— des constantes sont prédéfinies : `pi`, `i`...
— `inf` traduit l'infini et `NaN` signifie not a number (0/0)

Syntaxe

— **Opérateurs logiques** : == (égal à), ~= (différent de), < > <= >=, & (et), | (ou), ~ (non)

— **Opérations classiques :**

>> b=1+9 addition

>> c=1-9 soustraction

>> d=b*c multiplication

>> 4/8 division

>> 4^2 puissance

— **Fonctions mathématiques usuelles :**

min, max, mean (valeur moyenne), std (écart type), cov (covariance), sum (somme), abs (valeur absolue), real (partie réelle), imag (partie imaginaire), conj (conjugué d'un complexe), round (arrondir), sqrt (racine carrée), exp (exponentielle), log (logarithme base e), log10 (logarithme base 10), sin, cos, tan, asin, acos, atan, sinh, cosh, tanh, asinh, acosh, atanh

Syntaxe

— **Limite, dérivation, intégration :**

Opérateur	Description
`syms`	définit la variable
`lim(func,x,a)`	donne la limite de la fonction quand x tend vers a
`diff(func,n)`	Calcule la n-ème dérivée de la fonction `func`
`int(func)`	intègre la fonction `func`
`int(func,a,b)`	intègre la fonction `func` entre a et b

```
>> syms x
>>int(x*log(1 + x), 0, 1)
>>int(-2*x/(1 + x^2)^2)
>>diff(1/(x^2 + 1))
>>diff(x^2 + 1,2)
>>limit(1/x, x, 0, 'right')
>>limit(1/x, x, 0, 'left')
```

Syntaxe

— **Calcul sur les polynômes :**

Opérateur	Description
`polyval(p,x)`	Calcule les valeurs de p(x)
`roots(p)`	Calcule les racines de p
`conv(p1,p2)`	calcule les coefficients du polynôme p1p2
`polyder(p)`	calcule les coefficients de p'

```
>> x=linspace(-2,2,50)
>>c=[1 0 1 0 -1]
>>y=polyval(c,x)
>>plot (x,y)
```

On a tracé $x^4 + x^2 - 1$ sur [-2,2]

```
>> polyder(c)
>>roots(c)
```

Calcul matriciel

Opérateur	Description
`x=first:last`	Crée un vecteur ligne x commençant par `first` finissant par `last` (ou avant) avec un pas de 1.
`x=first:h:last`	Crée un vecteur ligne x commençant par `first` finissant par `last` (ou avant) avec un pas de `h`.
`x=linspace(first,last,n)`	Crée un vecteur ligne x commençant par `first` finissant par `last` (ou avant) avec `n` éléments.
`A(r,c)`	Rend le coefficient sur la ligne r et la colonne c de la matrice A
`A(r,:)` ou `A(:,c)`	les `:` prennent en compte toutes les lignes ou toutes les colonnes
`size(A)`	renvoie le nombre de lignes et le nombre de colonnes de A
`length(x)`	renvoie la taille d'un vecteur x.
`zeros(n,m)`	écrit une matrice nulle de taille (n,m)
`eye(n)`	écrit une matrice identité de taille (n,n)
`ones(n,m)`	écrit une matrice ne contenant que des 1 de taille (n,m)
`rand(n,m)`	écrit une matrice aléatoire de taille (n,m) avec des nombres compris entre 0 et 1
`inv(A)`	renvoie la matrice inverse de A
`trace(A)`	somme les éléments diagonaux de A
`eig(A)`	donne la matrice diagonale et la matrice des vecteurs propres
`det(A)`	calcule le déterminant de A

Calcul matriciel

- Définition d'une matrice : `>> A=[2,0,1;1,3,2;0,4,2]` Les coefficients d'une ligne sont séparés par des virgules ou des espaces. Les différentes lignes comportent le même nombre d'éléments et sont séparés par des points virgule.
- Définition d'une matrice aléatoire : `>> B=rand(3,3)`
- Définition d'un vecteur ligne : `>> b=[1,3,2]` ou `>> b=[1;3;2]'`
  ' correspond à l'opérateur de transposition.
- Définition d'un vecteur colonne : `>> b=[1;3;2]`
- Définition d'un vecteur contenant les valeurs de 0 à 1 avec un pas de 0.1 :
  `>> x=[0:0.1:1]`
  `>> y=[0:1]` crée un vecteur avec un pas unitaire.
- Définition d'un vecteur contenant les 6 valeurs de 0 à 1 avec un pas régulier :
  `>> x=linspace(0,1,6)`
  Remarque : `linspace(a,b,n)` est équivalent à `[a:`$\frac{b-a}{n-1}$`:b]`
- Matrice identité de dimension (n,n) : `>> eye(3)` pour $n = 3$.
- Matrice de dimension (n,m) ne contenant que des 1 : `>> ones(3,4)` pour $n = 3, m = 4$.
- Matrice de dimension (n,m) ne contenant que des zéros : `>> zeros(3,4)` pour $n = 3, m = 4$.
- Matrice ayant les termes du vecteur b sur la diagonale : `>> A=diag(b)`

Calcul matriciel

Sur les matrices de dimensions compatibles :

- somme : `A+B`
- transposition : `A'`
- produit matriciel : `A*B`
- inversion de matrices : `inv(A)`
- résolution de systèmes linéaires $Ax = b$, entrer la commande : `x=A\b`
- application de fonctions usuelles : `sin(A), exp(A), sqrt(A)`...

- nombre de lignes : `>> size(A,1)`
- nombre de colonnes : `>> size(A,2)`
- taille de A : `>> size(A)` renvoie `n m` si A est de taille (n,m)
- taille d'un vecteur x : `>>length(x)`

- extraction de termes, affectation :
  Notez qu'en Matlab, on commence la numérotation des lignes et des colonnes d'une matrice à 1.
  - obtenir le ième terme d'un vecteur x : `>>x(3)` pour $i = 3$
  - obtenir le terme a_{ij} : `>> A(2,2)` pour $i = 2, j = 2$
  - extraire la troisième ligne de A : `>> A(3,:)`
  - comment extraire la deuxième colonne de A ?
  - extraire la sous matrice allant de la 2ième à la 3ième ligne et de la 1ère colonne à la 3ième colonne : `>> A(2:3,1:3)`
  - supprimer les deux premières lignes de A : `A([1,2],:)=[]`

Calcul matriciel

— **distribution des opérations** :opérations élément par élément
On ne peut pas calculer les valeurs de la fonction x^2 et tracer son graphe en faisant

```
>> x=[0:0.01:1]
>> y=x^2
>> plot(x,y)
```

Il faut spécifier à Matlab que l'on veut distribuer l'opération ^ sur les nombres du tableau en remplaçant ^ par .^

```
>> x=[0:0.01:1]
>> y=x.^2
>> plot(x,y)
```

De la même manière il existe les opérations `.*` et les opérations `./`. Le problème ne se pose pas pour l'addition et la soustraction.

Exemple : `>> A=[1 2 3 4 5]`
`>> B=[1 4 5 2 2]`
`>> C=A.*B` D'après vous, quel est le résultat affiché ?
Attention, les matrices doivent avoir la même dimension.

Calcul matriciel

Exercice :

On considère les vecteurs et les matrices suivantes :

$$l = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \end{pmatrix}, v = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix}, A = \begin{pmatrix} 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 \end{pmatrix}$$

1. Extraire la première ligne de la matrice A, la deuxième colonne et la sous matrice $(a_{ij})_{1 \leq i \leq 3, 1 \leq j \leq 5}$

2. Calculer A^t et A est-elle inversible ? Si oui calculer son inverse.

3. Résoudre le système Ax=v.

4. Comment déterminer la taille de l,v,A ?

5. Comparer ce que produisent les instructions : `1:0.2:5`, `1:0.2:5.1`, `linspace(1,5,21)` et commenter.

Calcul matriciel

Graphiques 2D

L'affichage graphique le plus courant se fait à l'aide de la commande `plot`.

Exemple :

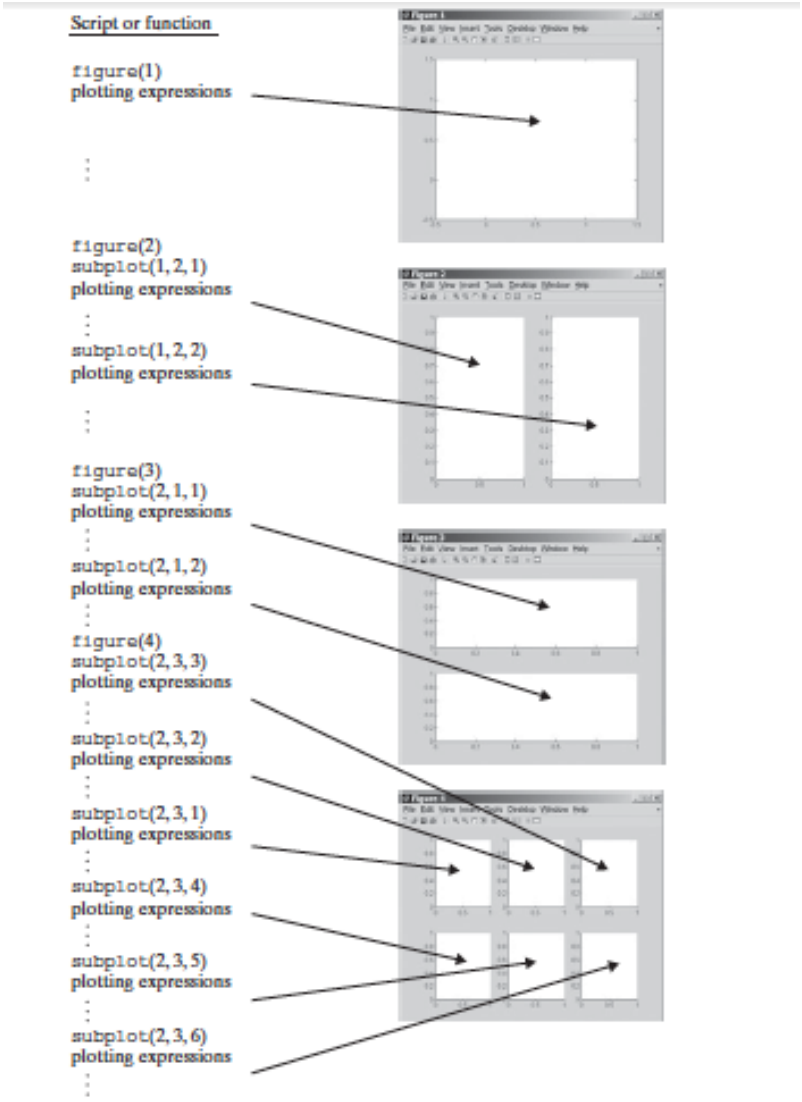
```
x=[0:0.1:5*pi]
y=sin(x)
plot(x,y)
```

La première commande construit en mémoire un tableau de nombres allant de 0 à 5π par pas de 0.1. Ce tableau est appelé x et ses éléments par `x(i)` où i est un indice entier.
La deuxième ligne applique la fonction sinus aux nombres du tableau x qui est alors considéré comme un tableau de nombres et non comme une matrice.
La dernière commande trace le graphe composé de points d'abscisses `x(i)` et d' ordonnées `y(i)`.

Graphiques 2D

`plot(x,y)`	trace le vecteur y en fonction du vecteur x.
`polar(theta,r)`	trace $r(\theta)$ en coordonnées polaires
`bar(x,y)`	trace $y(x)$ sous forme de barres
`loglog(x,y)`	trace $y(x)$ en échelles logarithmiques
`grid`	ajoute une grille
`hold on`	permet de tracer plusieurs graphes sur la même figure
`hold off`	recommence un nouveau graphe en restant dans la même fenêtre
`figure(2)`	ouvre une deuxième fenêtre graphique sans fermer la première
`subplot`	permet de diviser la fenêtre en plusieurs parties
`title`	donne un titre au graphe
`xlabel`	définit l'axe des abscisses
`ylabel`	définit l'axe des ordonnées
`legend`	ajoute une légende
`clf`	efface la figure
`axis[xmin xmax ymin ymax]`	délimite les axes du graphique

Graphiques 2D



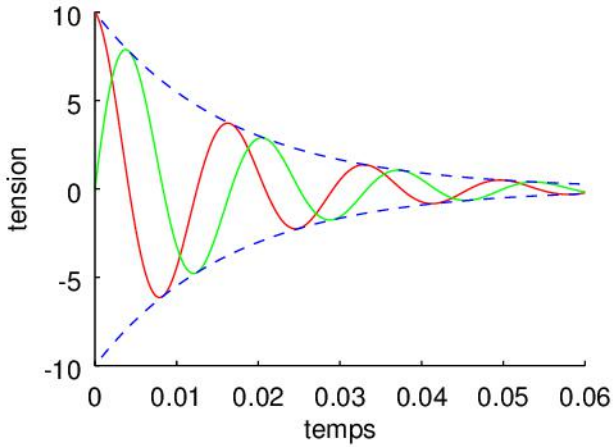
Graphiques 2D

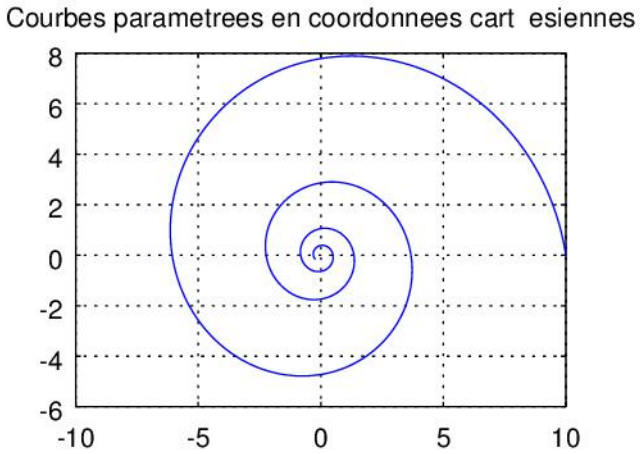
On peut étudier les instructions les plus courantes à travers la séquence suivante :

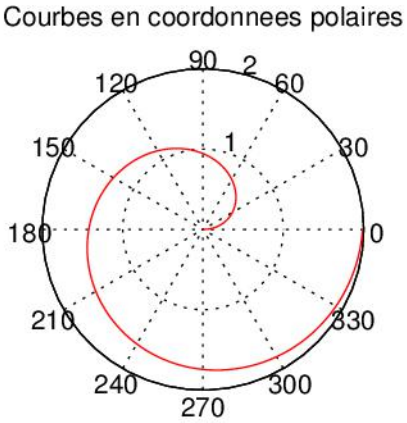
```
 t=[0:0.01e-3:0.06];
y=10*exp(-60*t).*cos(120*pi*t);
z=10*exp(-60*t).*sin(120*pi*t);
a=10*exp(-60*t);
figure(1);
subplot(2,2,1);
hold on;
plot(t,y,'r',t,z,'g');
plot(t,a,'b--',t,-a,'b--');
hold off;
xlabel('temps');  ylabel('tension');
subplot(2,2,2);
plot(y,z); grid;
title('Courbes parametrees en coordonnees cartésiennes');
subplot(2,2,3);
```

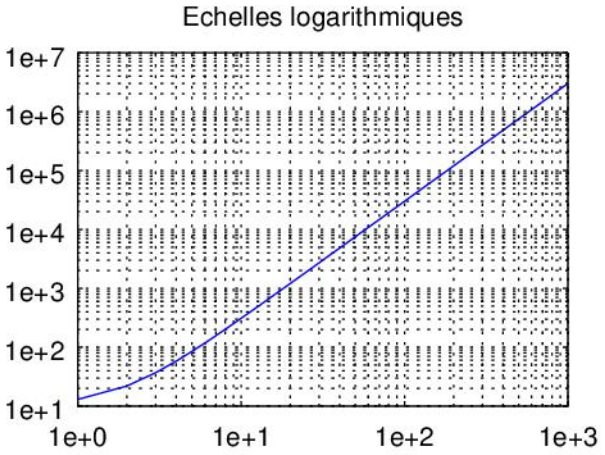
```
theta=0:0.01:2*pi;
r=log(1+theta);
polar(theta,r,'r-');
title('Courbes en coordonnees polaires');
subplot(2,2,4);
t=0:1:1000;
x=3*t.^2+10;
loglog(t,x); grid;
title('Echelles logarithmiques');
```

Graphiques 2D









Graphiques 2D

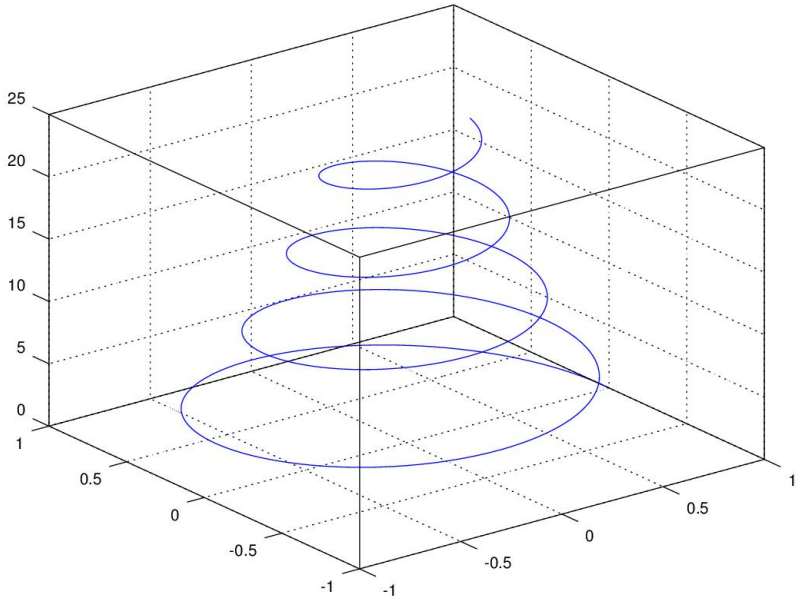
Exercice :

Représenter sur une figure à 3 cadrans, les fonctions sinus, exponentielle et logarithme.

Graphiques 3D

Pour une courbe paramétrée :

```
 t=[0:0.05:25];
x=exp(-0.05*t).*cos(t);
y=exp(-0.05*t).*sin(t);
z=t;
plot3(x,y,z); grid;
```

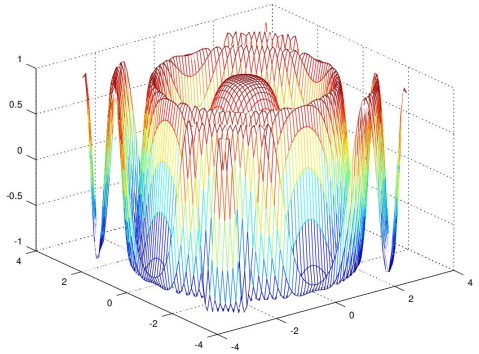


Graphiques 3D

Tracer une surface nécessite la création d'une grille régulière d'abscisses et d'ordonnées : `meshgrid` est à la base de leur construction. Ensuite pour le tracé, on peut utiliser une des fonctions suivantes :

`meshgrid(x,y)`	génere des matrices x,y pour les fonctions 3D
`contour3`	lignes de niveau 3D
`mesh`	surface définie par une grille 3D
`meshc`	mesh + contour
`quiver`	visualisation de champs de vecteurs
`surf`	surface ombrée
`surfc`	surface + contour
`waterfall`	tracé en chute d'eau

```
 x=[-pi:0.5:pi];
y=[-pi:0.5:pi];
[X,Y]=meshgrid(x,y);
Z=cos(X.^2+Y.^2);
mesh(X,Y,Z);
```



Programmation en Matlab

Quand on souhaite écrire un programme complet, on utilise l'éditeur de script de Matlab.

— **Définition de fonctions**

Matlab permet d'éxécuter des suites de commandes listées dans un fichier. Les fichiers à éxécuter doivent porter le suffixe $.m$
Pour que l'interpréteur de commandes trouve le fichier et reconnaisse la ligne de commande, il faut se placer dans le bon répertoire.
Pour créer des m. files , il faut aller dans l'éditeur de texte.

Une fonction est une procédure qui reçoit en entrée des arguments et renvoie en sortie d'autres tableaux de nombres. Elles sont écrites suivant la syntaxe suivante :

```
1 function z=mafonction(x,y)
2 % commentaires: arguments d'entree: x,y,vecteurs
3 % arguments de sortie: z vecteur
4 z=cos(x.^2+y.^2).*exp(-5*x)
5
6 end
```

Il suffit ensuite d'enregistrer le fichier sous `mafonction.m` et d'éxécuter en ligne de commande, par exemple, `>> mafonction(1,2)`

Programmation en Matlab

— **boucle conditionnelle**

```
if condition1
...
elseif condition2
...
else...
end
```

```
1 x=input('entrer une valeur pour x');
2 y=input('entrer une valeur pour y');
3 if x>0 && y>0
4 disp ('les deux valeurs sont positives')
5 end
```

Programmation en Matlab

Le mot clé `elseif` qui accompagne toujours un `if` permet d'éxecuter un code si les conditions du `if` ou des autres `elseif` situés auparavant sont fausses et que sa propre condition est vraie.
Le `else` ne possède aucune condition l'accompagnant : il est éxécuté si tous les autres choix ont été rejetés.

```
1 x=input('entrer une valeur pour x');
2 y=input('entrer une valeur pour y');
3 z=input('entrer votre choix d operation');
4 if z==1
5 disp(x+y);
6 elseif z==2
7 disp (x*y);
8 elseif z==3
9 disp(x-y);
10 elseif z==4
11 disp(x/y)
12 else disp ('mauvais choix');
13 end
```

Programmation en Matlab

— **boucle for**

```
for i = a : b : c
...
end
```

```
1  for i=1:10
2      x(i)=2*i;
3 end
4  x
```

Que va nous rendre Matlab ?

— boucle while

```
while condition
...
end
```

Il est imperatif que le bloc de code après le while fasse évoluer la condition sinon le programme ne sortira jamais de la boucle.
En général si on sait à l'avance combien de fois la boucle doit se répéter on utilisera une boucle `for`. Sinon on utilisera un `while`.

```
1  n=1
2 while 2^n<100
3  n=n+1;
4 end
5 n
```

Que va nous rendre Matlab ?

Programmation en Matlab

Exercice :

Définir la fonction f :

$$f(x) = \begin{cases} 0 \text{ si } x \leq -1 \\ x^2 \text{ si } -1 \leq x \leq 1 \\ 1 \text{ si } 1 \leq x \leq 4 \\ 0 \text{ si } x \geq 4 \end{cases}$$

et la dessiner sur [-5, 5].

Programmation en Matlab

Programmation en Matlab

Exercice :

Ecrire une fonction Trouve qui prend en argument un vecteur v et un nombre x, et qui retourne 1 si x est un élément du vecteur v, et 0 sinon.

Programmation en Matlab

Exercice :

1. Ecrire des scripts qui permettent de calculer N ! (factoriel de N) en utilisant la boucle while puis la boucle for.

2. A l'aide de `tic` et `toc`, calculez le temps pris par chacune de ces boucles pour calculer les factoriels de `N=100`.