Chapitre 1 MATLAB POUR LE COURS D'ANALYSE NUMERIQUE DE LA LICENCE 3 PHYSIQUE NUMERIQUE

Docteur DIALLO

UNIVERSITE ALIOUNE DIOP DE BAMBEY

18 Avril 2020

mamadou.salif.diallo@aims-senegal.org

mamadou.salif.diallo@aims-senegal.org

1. Matlab est un logiciel de calcul numérique commercialisé par la société *MathWorks*[1]. Il a été initialement développé à la fin des années 70 par Cleve Moler, pour permettre aux étudiants de travailler à partir d'un outil de programmation de haut niveau et sans apprendre le Fortran ou le C.
2. Matlab signifie Matrix laboratory. Il est un langage pour le calcul scientifique, l'analyse de données, leur visualisation, le développement d'algorithmes. Son interface propose, d'une part, une fenêtre interactive type console pour l'exécution de commandes, et d'autre part, un environnement de développement intégré (IDE) pour la programmation d'applications.
3. Matlab trouve ses applications dans de nombreuses disciplines. Il constitue un outil numérique puissant pour la modélisation de systèmes physiques.

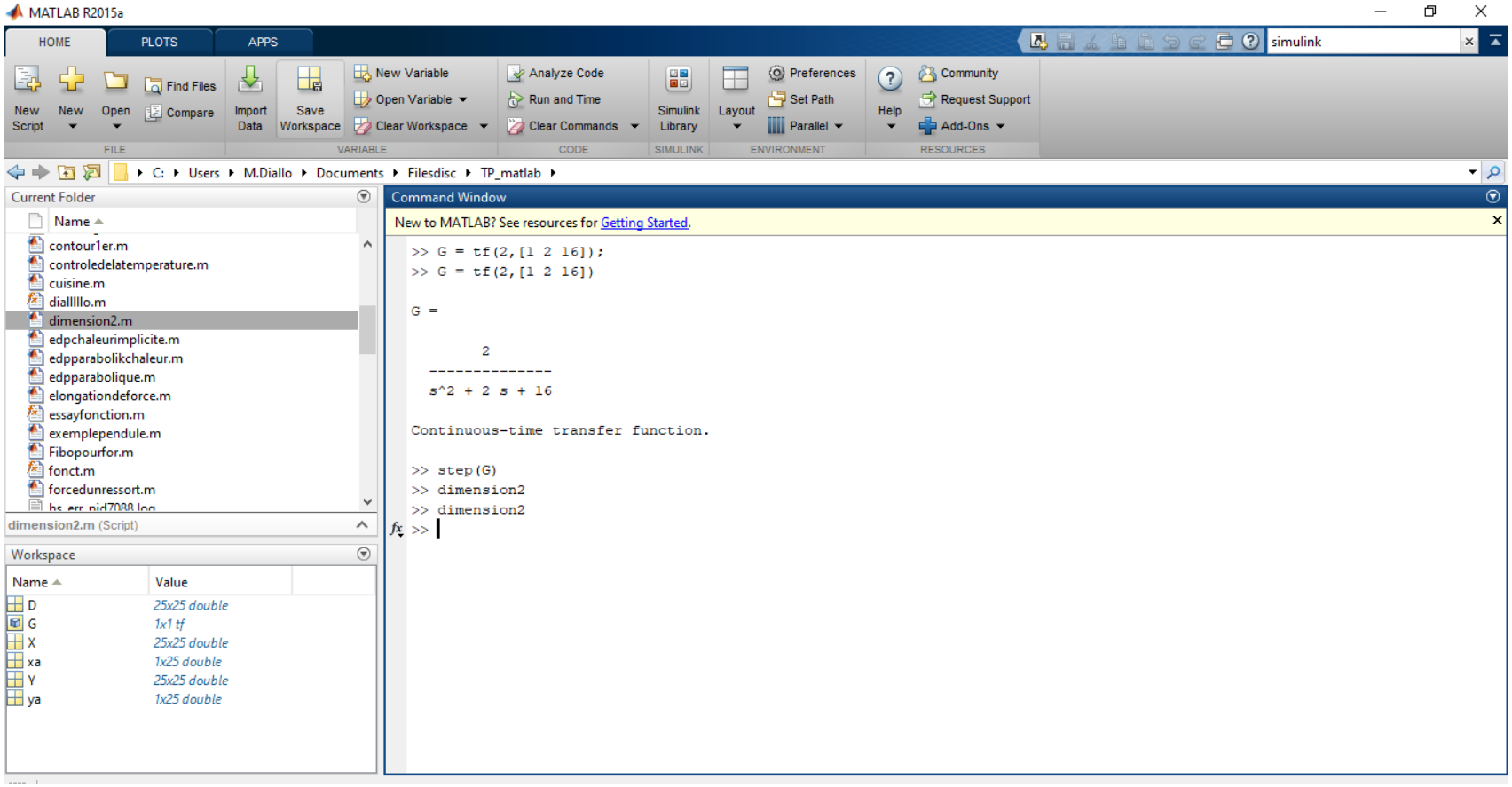
1. Matlab est un logiciel de calcul numérique commercialisé par la société *MathWorks*[1]. Il a été initialement développé à la fin des années 70 par Cleve Moler, pour permettre aux étudiants de travailler à partir d'un outil de programmation de haut niveau et sans apprendre le Fortran ou le C.
2. Matlab signifie Matrix laboratory. Il est un langage pour le calcul scientifique, l'analyse de données, leur visualisation, le développement d'algorithmes. Son interface propose, d'une part, une fenêtre interactive type console pour l'exécution de commandes, et d'autre part, un environnement de développement intégré (IDE) pour la programmation d'applications.
3. Matlab trouve ses applications dans de nombreuses disciplines. Il constitue un outil numérique puissant pour la modélisation de systèmes physiques.

1. Matlab est un logiciel de calcul numérique commercialisé par la société *MathWorks*[1]. Il a été initialement développé à la fin des années 70 par Cleve Moler, pour permettre aux étudiants de travailler à partir d'un outil de programmation de haut niveau et sans apprendre le Fortran ou le C.
2. Matlab signifie Matrix laboratory. Il est un langage pour le calcul scientifique, l'analyse de données, leur visualisation, le développement d'algorithmes. Son interface propose, d'une part, une fenêtre interactive type console pour l'exécution de commandes, et d'autre part, un environnement de développement intégré (IDE) pour la programmation d'applications.
3. Matlab trouve ses applications dans de nombreuses disciplines. Il constitue un outil numérique puissant pour la modélisation de systèmes physiques.

1. Matlab est un logiciel de calcul numérique commercialisé par la société *MathWorks*[1]. Il a été initialement développé à la fin des années 70 par Cleve Moler, pour permettre aux étudiants de travailler à partir d'un outil de programmation de haut niveau et sans apprendre le Fortran ou le C.
2. Matlab signifie Matrix laboratory. Il est un langage pour le calcul scientifique, l'analyse de données, leur visualisation, le développement d'algorithmes. Son interface propose, d'une part, une fenêtre interactive type console pour l'exécution de commandes, et d'autre part, un environnement de développement intégré (IDE) pour la programmation d'applications.
3. Matlab trouve ses applications dans de nombreuses disciplines. Il constitue un outil numérique puissant pour la modélisation de systèmes physiques.

Le logiciel de base peut être complété par de multiples toolboxes. Nous pouvons citer par exemple :
l'Automatique, le traitement du signal, l'analyse statistique, l'optimisation... Ce cours propose une introduction à Matlab et développe un ensemble de fonctionnalités spécifiques à certains domaines des sciences de l'ingénieur. En plus de l'aide intégrée à l'environnement et des nombreux ouvrages dédiés, une quantité abondante de ressources sont disponibles sur Internet
Documentation en ligne de MathWorks
:http://www.mathworks.com/help/techdoc/
Developpez.com :*http : //matlab.developpez.com/*
Matlab Central :*http : //www.mathworks.com/matlabcentral/*
http : //nte.mines − albi.fr/MATLAB/co/Matlab$_{web}$.html
www.personnel.isae.fr/sites/personnel/IMG/pdf/InitiationMatLab.pdf
www.perso.telecom − paristech.fr/ prado/enseignement/polys/matlab.html

Au lancement de Matlab, l'interface suivante apparait



Les interfaces

1. Command Window (console d'exécution) : à l'invite de commande(>>), l'utilisateur peut entrer les instructions à exécuter. Il s'agit de la fenêtre principale de l'interface
2. Workspace (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire...
3. Command History :historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande.
4. Current Folder(répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable.

Les interfaces

1. Command Window (console d'exécution) : à l'invite de commande(>>), l'utilisateur peut entrer les instructions à exécuter. Il s'agit de la fenêtre principale de l'interface
2. Workspace (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire...
3. Command History :historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande.
4. Current Folder(répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable.

Les interfaces

1. Command Window (console d'exécution) : à l'invite de commande(>>), l'utilisateur peut entrer les instructions à exécuter. Il s'agit de la fenêtre principale de l'interface
2. Workspace (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire...
3. Command History :historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande.
4. Current Folder(répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable.

Les interfaces

1. Command Window (console d'exécution) : à l'invite de commande(>>), l'utilisateur peut entrer les instructions à exécuter. Il s'agit de la fenêtre principale de l'interface
2. Workspace (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire...
3. Command History :historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande.
4. Current Folder(répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable.

Les interfaces

1. Command Window (console d'exécution) : à l'invite de commande(>>), l'utilisateur peut entrer les instructions à exécuter. Il s'agit de la fenêtre principale de l'interface
2. Workspace (espace de travail) : permet de visualiser les variables définies, leur type, la taille occupée en mémoire...
3. Command History :historique des commandes que l'utilisateur a exécutées. Il est possible de faire glisser ces commandes vers la fenêtre de commande.
4. Current Folder(répertoire courant) : permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Les programmes de l'utilisateur doivent être situés dans ce répertoire pour être visible et donc exécutable.

Préliminaires

Les commandes peuvent se taper directement suite au prompt de Matlab. L'opération est alors immédiatement effectuée et le résultat retourné. Si la commande se termine par un point virgule, la commande est effectuée, mais le résultat obtenu n'est pas retourné.

L'aide

help *func* affiche l'aide concernant la fonction *func*. Voir *help help*...

demo Matlab contient de nombreuses démo. de ses capacités. Taper demo, et naviguer...

Commandes générales

cd change/affiche le répertoire courant

which affiche le chemin complet d'une fonction Matlab

Préliminaires

path variable Matlab contenant la liste des répertoires connus, dans lesquels Matlab recherche une fonction lors de son appel
addpath permet d'ajouter un chemin dans le path
who liste des variables de l'espace de travail
clear var supprime la variable de l'espace de travail
clear all supprime toutes les variables
close all ferme tous les graphiques

Opérations usuelles

= affectation d'une valeur à une variable (ex. a = 2)

+,-,*,/ opérations usuelles sur les variables ou valeurs numériques

1i nombre complexe $(1i)^2 = -1$

abs, angle, real, imag opérations usuelles sur les nombres complexes

Définitions et opérations sur les vecteurs et matrices

[, , ; , ,] définition manuelle d'une matrice (ex. $A = [1, 2, 3; 4, 5, 6]$)

deb:pas:fin définition d'un vecteur régulier balayant l'intervalle [*deb*, *fin*] avec le pas (ex. $A = 1 : 16$); par défaut le pas est égal à 1 s'il est omis (ex. $A = 1 : 6$)

linspace(deb,fin,N) définition

Ligne de Commande

Vous saisissez ces lignes de commande dans le console (command window) et sur la feuille de script (current folder).

```
%section définition du cours SalifDiallo
% Definitions de scalaires |||||||||||||||||||| %
a = 4 ;
A = 7 ;
A = A * 2 ;
disp(a) ; % a = 1
disp(A) ; % A = 4
z = -5 + 4*i ; % Nombre complexe
```

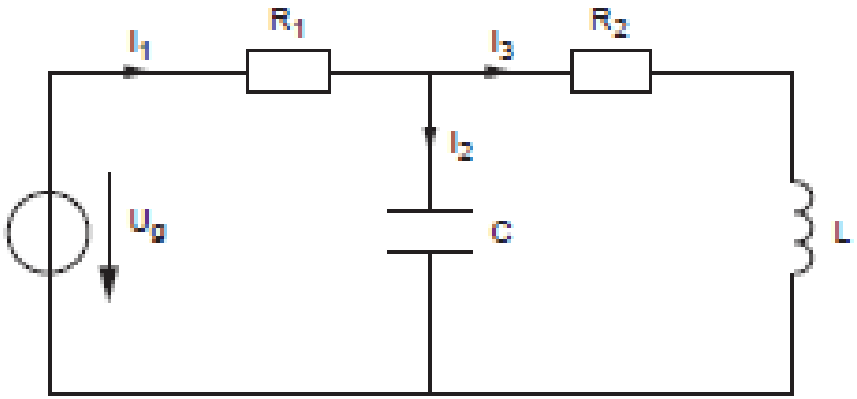
```
% Declaration de Vecteurs  %
vecA = [ 1 2 3 4 5] ; % Vecteur horizontal
vecB = [ 1 2 3 4 5]' ; % Vecteur vertical
vecB = [1;2;3;4;5];
% | Declaration de Matrices %
matM = [ 1 2 3 ; % Ligne 1
4 5 6 ] ; % Ligne 2
%  Matrices Complexes %
Z = [1 2 ; 3 4] + i * [5 6 ; 7 8]

%  Definitions de matrices speciale %
eye(m,n) % Matrice unitée
ones(m,n) % Matrice dont tous les éléments sont égaux à un
zeros(m,n) % Matrice dont tous les éléments sont égaux à zéro
rand(m,n) % Matrice d'éléments aléatoires
diag(m,n) % Matrice diagonale
meshgrid(m :n,k :l) % Renvoie deux matrices de grilles défnissant un quadrilla
```

```
 3       %Transposition '
 4 -     + % Addition
 5 -     - % Soustraction
 6 -     * % Multiplication
 7 -     / % Division à droite
 8 -     \ % Division à gauche
 9 -     ^ % Puissance
10       %%%%%%%%%%%%%% Fonctions sur les scalaires %%%%%%%%%%%%%%%%%%%%
11 -     abs(z) % Valeur absolue ou module complexe
12 -     conj(z) % Complexe conjugu□e
13 -     imag(z) % Partie imaginaire
14 -     real(z) % Partie r□eelle
15 -     norm(z) % Norme
16 -     prod(z) % Produit de tous les □el□ements d'un argument vectoriel
17 -     sum(z) % Somme de tous les □el□ements d'un argument vectoriel
18 -     round(z) % Arrondit tous les □el□ements aux entiers les plus proches
```

```
%-----------FONCTIONS SUR LES MATRICES--------- %
cumsum(V) % Somme cumulée
cumprod(V) % Produit cumulé
sum(V) % Soustraction
prod(V) % Multiplication
tril(A) % Extraction de la partie triangulaire inférieure d'une matrice
triu(A) % Extraction de la partie triangulaire supérieure d'une matrice
sort(A) % Trie la matrice
iplr(A) % Retourne la matrice horizontalement
ipud(A) % Retourne la matrice verticalement
rank(A) % Rang de la matrice
trace(A) % Somme des éléments diagonaux
%------OpErations élément par élément-----------%
.+ % Addition
.- % Soustraction
.* % Multiplication
./ % Division
.^ % Puissance
```

Exemple Résoudre le circuit suivant en utilisant Matlab Étant donné le circuit de la figure suivante, on souhaite calculer les valeurs efficaces et phases des courants lorsque
$U_g = 220[V]$, $f_g = 50[Hz]$, $R_1 = 10[\Omega]$,$R_2 = 3[\Omega]$,$C = 10[\mu F]$,
$L = 100[mH]$



Utiliser la loi des mailles et la loi des noeuds pour générer toutes les équations reliant les courants, tensions et résistances.
Écrire les équations sous forme de système puis calculer les valeurs efficaces à l'aide d'un programme sous Matlab.

Programme du circuit

```
% Resolution d'un circuit electrique
R1=10;
R2=3;
c=10*10^-6;  Ug=200;  L=0.1;  fg=50;  w=2*pi*fg;
%Declaration de la matrice obtenue
C=[R1+1/(j*c*w) -1/(j*c*w);-1/(j*c*w) R2+j*L*w+1/(j*c*w)];
b=[Ug;0];
I=inv(C)*b; I1=1.9564 - 4.9748i; I3= 2.1127 - 5.5417i;
I2=I1-I3; I=[I1;I2;I3]; Imax=max(I); Ieff=Imax/sqrt(2);
%phaseI=angle
```

$>> t = [0 \quad 0.5 \quad 1 \quad 1.5 \quad 2 \quad 2.5 \quad 3 \quad 3.5 \quad 4 \quad 4.5 \quad 5 \quad 5.5 \quad 6]$

Cette ligne de commande définit un tableau de 13 valeurs (allant de 0 à 6 par incrément de 0.5) nommé «t»

$>> z = \frac{2*\pi}{6}$

$>> y = sin(w * t)$

Le terme « pi » est une constante prédéfinie et donne donc la valeur de π. Le tableau (ou vecteur) « y », de même dimension que « t », contient les valeurs résultantes de l'opération appliquée à chaque composante de « t ». « y » représente donc la fonction sinus de période 6 (de pulsation « w »). Il est possible de représenter graphiquement les points du tableau à l'aide de la fonction suivante :

$>> plot(t, y, '*');$

Deux courbes sur un même repère

```
% tracer de fonction sinus et cosinus
% commence par sinus
%on declare le domaine de travail
t=[-pi:pi/4:pi];   g=sin(t);  plot(t,g,'g','linewidth',2)
hold on;
%on trace sur le meme axe la fonction cosinus
h=cos(t);
plot(t,h,'r','linewidth',3)
hold off;
xlabel('abscisse(m)');
ylabel('ordonnée(m)');
title('Fontion trigonometrique');
legend('cosinus', 'sinus')
```

Dans les précédentes sections, nous avons présenté des séries de commandes lancées depuis la command window. Pour des calculs complexes et répétitifs, il est préférable (ou plutôt indispensable) de rassembler l'ensemble des commandes dans un fichier qui constituera le programme à exécuter. On distingue deux types de fichiers dans Matlab, également appelés m-files : les scripts et les fonctions. Bien que l'environnement de Matlab propose son propre éditeur (fenêtre Editor), ces fichiers sont de simples fichiers textes avec une extension .m. Vous pouvez donc utiliser votre éditeur de texte préféré pour créer vos programmes (sans oublier de modifier l'extension). A partir de Matlab, un m-file est créé ou ouvert, soit depuis le menu Fichier (New > M-File), soit depuis l'invite en tapant : **edit monfichier**

1. Tracer la fonction sinus dans l'intervalle $[-\pi, \pi]$ avec un pas de 0.01.
2. Tracer deux courbes, sur un même graphe en utilisant les commandes *holdon* et *holdoff* telle que $x = [-20, 20]$ de taille 1000 éléments, $y = x \times sin(x)$ et $y^2 = -x$
3. Soit un domaine horizontal $x = [-1, 1]$ avec 100 éléments de nœud.
   1. Utiliser la commande lispace pour représenter la longueur de x puis tracer le graphe $f(x) = x$
   2. Sur le même graphe, tracer les courbes $f_2(x) = x^2, f_3(x) = x^3$; mettre en valeur les graphes sur le fond, la forme et leur épaisseur.

1. Tracer la fonction sinus dans l'intervalle $[-\pi, \pi]$ avec un pas de 0.01.
2. Tracer deux courbes, sur un même graphe en utilisant les commandes *holdon* et *holdoff* telle que $x = [-20, 20]$ de taille 1000 éléments, $y = x \times sin(x)$ et $y^2 = -x$
3. Soit un domaine horizontal $x = [-1, 1]$ avec 100 éléments de nœud.
   1. Utiliser la commande lispace pour représenter la longueur de x puis tracer le graphe $f(x) = x$
   2. Sur le même graphe, tracer les courbes $f_2(x) = x^2, f_3(x) = x^3$; mettre en valeur les graphes sur le fond, la forme et leur épaisseur.

1. Tracer la fonction sinus dans l'intervalle $[-\pi, \pi]$ avec un pas de 0.01.
2. Tracer deux courbes, sur un même graphe en utilisant les commandes *holdon* et *holdoff* telle que $x = [-20, 20]$ de taille 1000 éléments, $y = x \times sin(x)$ et $y^2 = -x$
3. Soit un domaine horizontal $x = [-1, 1]$ avec 100 éléments de nœud.
   1. Utiliser la commande lispace pour représenter la longueur de x puis tracer le graphe $f(x) = x$
   2. Sur le même graphe, tracer les courbes $f_2(x) = x^2, f_3(x) = x^3$; mettre en valeur les graphes sur le fond, la forme et leur épaisseur.

Programme des trois graphes

```
% afficher de plusieurs courbes sur un meme graphe
x=linspace(-1,1,100);
y1=x.^1;
y2=x.^2;
y3=x.^3;
% afficher les fonctions
fig =figure('color','w');
plot(x,y1,'-r','LineWidth',5);
hold on;
plot(x,y2,'-g','LineWidth',3);
plot(x,y3,'--b','LineWidth',3);
hold off;
% ajouter legende
legend('x','x.^2','x.^3', 'Location','SouthEast');
```

Programme de l'exercice 2

```
x = linspace(-20,20,1000);
y = x.*sin(x);
plot(x,y)
hold on;
figure(2);
y2=-x;
plot(x,y2,'r')
grid on;
xlabel('valeur x')
ylabel('valeur y')
title('mon graphique')
legend('ma courbe')
grid on
```

Animation

```
%Définir d'abord le domaine
x=linspace(0,10,1000);
for k=2:length(x)
y1=sin(x);
y2=cos(x);
plot(x,y1,x,y2,'-g','linewidth',4)
comet(x,y1)
comet(x,y2)
end
```

Boucles FOR et WHILE

Les instructions de contrôle sous MATLAB sont très proches de celles existant dans d'autres langages de programmation.

Boucle FOR : parcours d'un intervalle

Une première possibilité pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle pour les valeurs d'un indice, incrémenté à chaque itération, variant entre deux bornes données.

Syntaxe :

$$for \quad indice = borne_i : borne_s$$

$$squence \quad d'instructions$$

$$end$$

Exemple : Utiliser une boucle FOR pour calculer n factoriel

Boucle WHILE : parcours d'un intervalle

Une seconde possibilité pour exécuter une séquence ... consiste à effectuer une boucle tant que qu'une condition est vérifiée. On arrête de boucler dès que cette condition n'est plus satisfaite.

Boucle FOR

Syntaxe :

while expression logique

squence d' instructions

end

Exemple : Utiliser une boucle FOR pour calculer les n termes de la suite de Fibonacci

```
% UTILISATION DE BOUCLE FOR POUR CALCULER LES TERMES DE LA SUITE
% fIBONACCI
n=197;
fibo=[1,1];
for i=3:n
fibo(i)=fibo(i-1)+fibo(i-2);
end
```

Boucle WHILE

Syntaxe :

while expression logique

sequence d'instructions

end

Exemple : Utiliser une boucle WHILE pour calculer n factoriel

```
n=6;
nfac = 1;
for k = 1:n
nfac = nfac*k;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Applications Utilisation de WHILE

On lance un ballon avec une vitesse de $v_0 = 20m/s$ qui fait un angle α avec l'horizontal. A l'instant $t = 0s$ les coordonnées spatiales sont nulles; on donne $g = 9,81m/s^2$.

1) Donner le modèle traduisant ce problème si l'angle α est nul.

2) Pour tout $y \geq 0$, donner la position de la balle pour chaque temps $t = 0,1s$.

3) Récupérer les données puis tracer la trajectoire de la balle en fonction du temps, utiliser les commandes qui soignent bien la courbe.

Programmation du ballon lancé

```
%Montrer la position d'un ballon pour chaque 0,1 seconde jusqu'a
% on déclare les données
v0=20;
g=9.8;
t=0;
y=0;
% boucle avec la condition y positive ou nulle
while(y>=0);
disp(['Ã  t=',num2str(t),'balle=',num2str(y)]);
t=t+0.1;
y=v0*t-g*t^2/2;
end
```

Les polynômes

L'objectif principal de l'interpolation est d'interpoler des données connues à partir des points discrets. Dans ce cas, la valeur de la fonction entre ces points peut être estimée. Cette méthode d'estimation peut être étendue et utilisée dans divers domaines ; à savoir la gestion de données qui regroupe plusieurs domaines de l'ingénierie.
Dans MATLAB, les polynômes sont représentés sous forme de vecteurs lignes dont les composantes sont données par ordre des puissances décroissantes. Un polynôme de degré n est représenté par un vecteur de taille $(n+1)$.
Exemple Le polynôme $8x^5 + 2x^3 - 3x^2 + 4x - 2$ est représenté par :
$p = [8 \quad 0 \quad 2 \quad -3 \quad 4 \quad -2]$
La fonction $conv(f, g)$ donne le produit de convolution de deux polynômes. L'exemple suivant montre l'utilisation de cette fonction
La fonction $deconv(p, q)$ donne le rapport de convolution de deux polynômes (déconvolution des coefficients du polynôme).

Interpolation directe

```
% comment approcher des données après une expérience au laborato
x = 0:10;
y = sin(x);
xi = 0:.25:10;
yi = interp1(x,y,xi,'line');
plot(x,y,'o',xi,yi)
```

Interpolation linéaire

```
% Utilisation de la fonction interp1 *
%*********************************
clear all;  clc;
x=0:10;   y=cos(x); % Points à interpoler
z=0:0.25:10; % Le pas du vecteur z est inférieur à celui de x
% Interpolation linéaire
figure(1);  f=interp1(x,y,z);
% Tracé des valeurs réelles et de la courbe d'interpolation
plot(x,y,'*r',z,f);   grid on;
xlabel('Interpolation');
% Interpolation par splines cubiques
figure(2);  f=interp1(x,y,z,'spline');
plot(x,y,'*r',z,f);
grid on;
xlabel('Interpolation par splines cubiques');
```

Évaluation d'un polynôme

Pour évaluer le polynôme $P(x)$ en un point donné, on doit utiliser la fonction $polyval(P, 1)$. On évalue ce polynôme pour x=1, par exemple :
On déclare P puis x et on évalue le polynôme
Dans le domaine de l'analyse numérique des données, on a souvent besoin d'établir un modèle mathématique liant plusieurs séries de données expérimentales. L'interpolation polynomiale consiste à approcher la courbe liant les deux séries de mesures par un polynôme. Les coefficients optimaux de ce polynôme sont ceux qui minimisent la variance de l'erreur d'interpolation. Ce principe (voir cours du Pr Mbaye) est connu sous le nom de la méthode des moindres carrés. La fonction $polyfit(x, y, n)$ retourne le polynôme P de degré n permettant d'approcher la courbe au sens des moindres carrés.

Exemple

```
>>x=[1.1 2.3 3.9 5.1];
>>y=[3.887 4.276 4.651 2.117];
>>a=polyfit(x,y,length(x)-1)
a =-0.2015 1.4385 -2.7477 5.4370
```

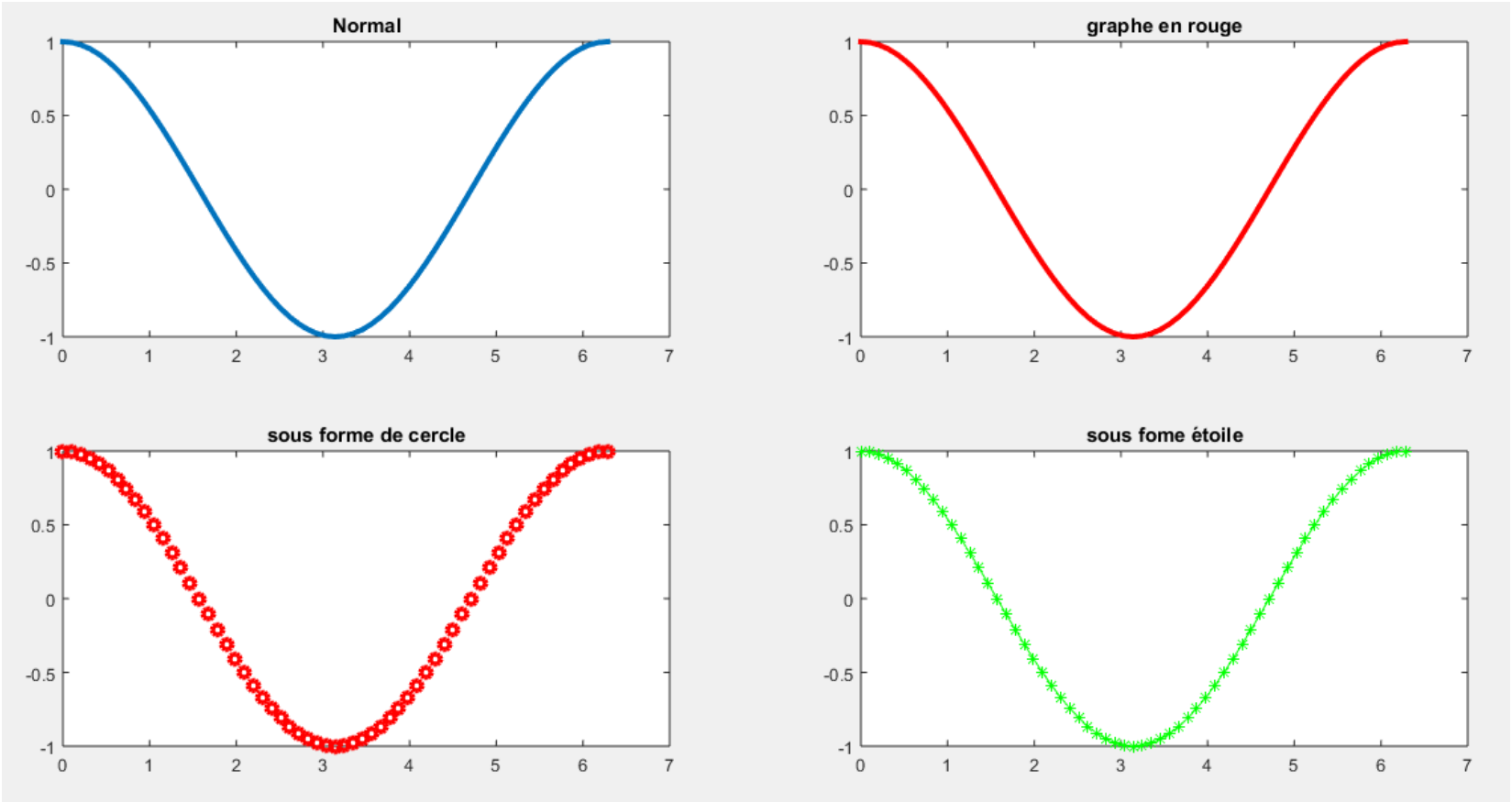
Ainsi, le polynôme d'interpolation de y (d'ordre length(x)-1=3)

Application à la démographie

```
annee = [1900 1910 1920 1930 1941 1950 1960 1970 1980 1990 2000]
population = [3315 3753 3880 4066 4266 4715 5429 6270 6366 6874
p10 = polyfit(annee, population,1);
annee_sample = [1900:2:2010];
vp2 = polyval(p10, annee_sample);
plot(annee, population, '*r',annee_sample, vp2, 'b', 'LineWidth'
xlabel('Annee');
ylabel('Population');
title('EVOLUTION DEMOGRAPHIQUE');
```

Commande subplot

Nous voulons placer sur une page un certain nombre de figures sans utiliser les ajustements de word c'est à dire à partir du programme



mamadou.salif.diallo@aims-senegal.org

Programme de subplot

```
% un graphe avec subplot de cos(t)
% on veut tracer le cos(t) dans l'intervalle [0 2pi]
t=0:pi/30:2*pi;
y=cos(t);
subplot(2,2,1)
plot(t,y,'Linewidth',3);
title('Normal');
subplot(2,2,2)
plot(t,y,'r','Linewidth',3);
title('graphe en rouge');
subplot(2,2,3)
plot(t,y,'or','linewidth',3)
title('sous forme de cercle');
subplot(2,2,4)
plot(t,y,'*-g')
title('sous fome étoile');
```

Projet Labo-mécanique

On considère ici un ressort que l'on tend plus ou moins tout en mesurant la force qui lui est appliquée à l'aide d'un dynamomètre dont la précision est d'environ 0.5 [N]. Les résultats que l'on a obtenus sont les suivants

Longueur [cm]	4.2	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.0	14.0
Force [N] $+1/-0.5$ [N]	0.0	1.1	2.0	3.2	3.9	4.6	5.8	7.0	8.3	9.0	9.5

1. Tracer le graphe de la force en fonction de la longueur avec les barres d'erreurs.
2. Mettre en valeur le graphe à l'aide d'un titre et d'informations portées sur l'abscisse
et l'ordonnée. 3. Rechercher une loi polynomiale représentant au mieux cette caractéristique ; celle d'un ressort est généralement linéaire, éventuellement cubique.
4. Mesurer la qualité des modèles proposés pour représenter le ressort.
5. Afficher les informations sur le graphe lui-même.

Programme du projet

```
% Exemple de traitement des donnees
clear all ; close all ; format compact ; format short g ;
% elongation d'un ressort : valeurs mesurees
x = [4.2 5.0 6.0 7.0 8.0 9.0 10.0 11.0 12.0 13.0 14.0] ; % [cm]
force = [0.0 1.1 2.0 3.2 3.9 4.6 5.8 7.0 8.1 9.0 9.5] ; % [N]
% vecteurs des erreurs
ErrNeg = -0.5*ones(size(force)) ;
ErrPos = +1.0*ones(size(force)) ;
% graphes des mesures
plot(x,force,'o') ;
% regressions lineaire et cubique
coeff1 = polyfit(x,force, 1) % coeff. du polynôme d'ordre 1
coeff3 = polyfit(x,force, 3) % coeff. du polynôme d'ordre 3
```

Programme du projet

```
% calcul des graphes des modeles d'ordre 1 et 3
xfit = linspace(0,20,201) ; % abscisse plus fine (200 points)
F1 = polyval(coeff1,xfit) ; % valeurs du polynôme d'ordre 1
F3 = polyval(coeff3,xfit) ; % valeurs du polynôme d'ordre 3
% tracage des mesures avec erreurs et des modeles
errorbar(x,force,ErrNeg, ErrPos,'o') ;
hold on ; grid on ; % maintien du graphe et pose d'une grille
plot (xfit,F1, xfit,F3,'--') ;
axis ([0 20 -5 13]) ;  title('Force d''un ressort') ;
xlabel('x [cm]') ; ylabel('F(x) [N]') ;
% affichage des infos
texte = ['F_1(x) = ', num2str(coeff1(1),3), ' x + '] ;
texte = [texte, num2str(coeff1(2),3)] ;
text(1,11, texte) ;
texte = ['F_3(x) = ', poly2str(coeff3,'x')] ;
text(2.1,-3, texte) ;
```

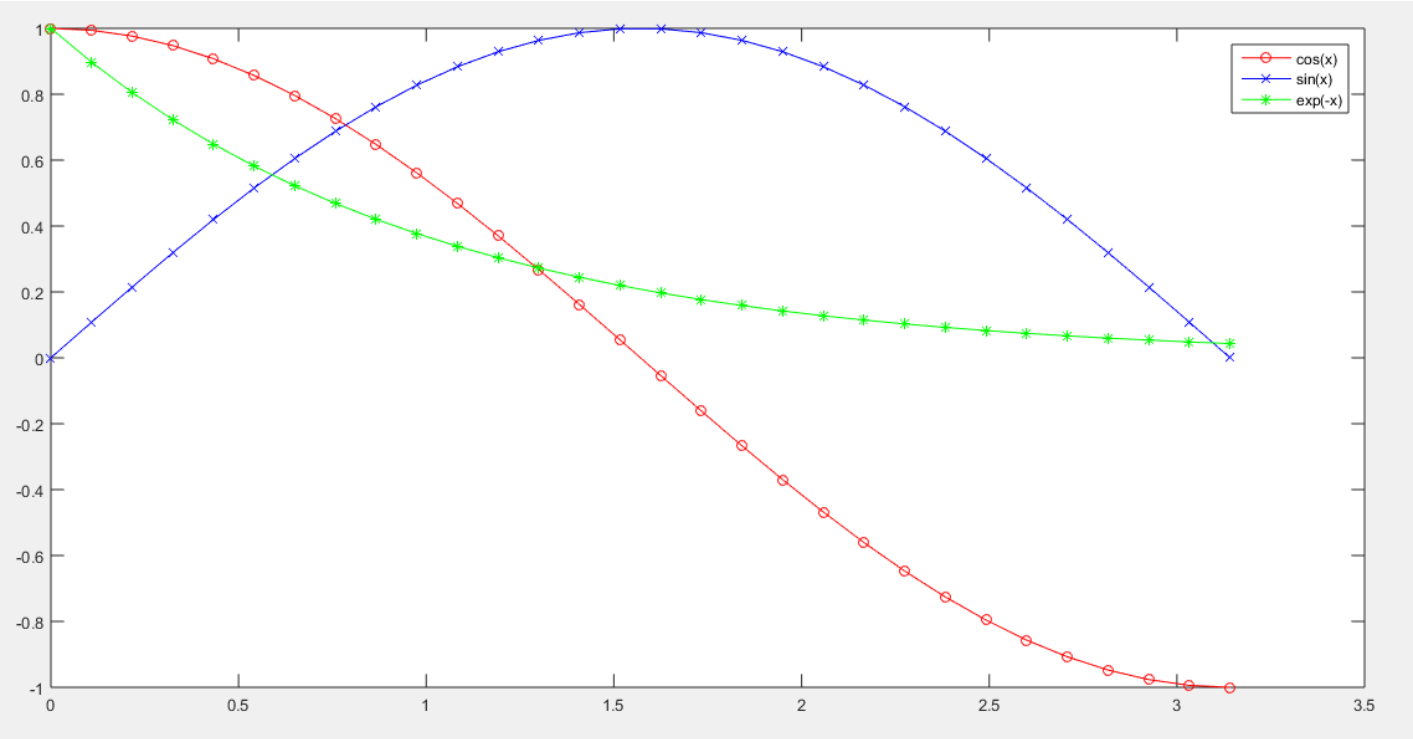
mamadou.salif.diallo@aims-senegal.org

Programme suite

```
% ecart type = sigma = mesure de la dispersion des ecarts
F1 = polyval(coeff1, x) ; % valeurs du modele d'ordre 1
ecart1 = force - F1 ;
eqm1 = sum(ecart1 .^ 2)/length(ecart1) ;
sigma1 = sqrt(eqm1) ;
text (1,10.2, ['\sigma _1 = ', num2str(sigma1,3), ' [N]']) ;
F3 = polyval(coeff3, x) ; % valeurs du modele d'ordre 3
ecart3 = force - F3 ;
eqm3 = sum(ecart3 .^ 2)/length(ecart3) ;
sigma3 = sqrt(eqm3) ;
text (2.1,-3.8, ['\sigma _3 = ', num2str(sigma3,3), ' [N]']) ;
print -deps ressort.eps
```

Graphique multiples sans hold on

Il est possible de tracer plusieurs courbes sur le même graphique. Pour ce faire, une méthode consiste à mettre l'ensemble des fonctions à tracer dans les parenthèses qui suivent la com-mande plot.



Programme correspondant

```
% Autre façon de tracer des courbes sur un meme repère
figure ; % Nouvelle fenetre
% Fonction du temps
x = linspace(0,pi,30) ;
plot(x,cos(x),'o-r',x,sin(x),'x-b',x,exp(-x),'*-g')
legend('cos(x)','sin(x)','exp(-x)')
```

Instruction if

L'instruction **if** est la plus simple et la plus utilisée des structures de contrôle de flux.
Elle permet d'orienter l'exécution du programme en fonction de la valeur logique d'une condition. Sa syntaxe générale est la suivante :

if (condition)
instruction 1
instruction 2
instruction 3
...
end ou bien

if (condition)
ensemble d'instruction
else
ensemble d'instruction
end

Si la condition est évaluée à vrai (true), les instructions entre le **if** et le **end** seront exécutées, sinon elles ne seront pas (oi si un **else** existe les instructions entre le **else** et le **end** seront exécutées). S'il est nécessaire de vérifier plusieurs conditions au lieu d'une seule, on peut utiliser des clauses **elseif** pour chaque nouvelle condition, et à la fin on peut mettre un **else** dans le cas ou aucune condition n'a été évaluée à vrai.

```
if (expression_1)
      Ensemble d'instructions 1
elseif (expression_2)
      Ensemble d'instructions 2
            ....
elseif (expression_n)
      Ensemble d'instructions n
else
      Ensemble d'instructions si toutes les expressions étaient fausses
end
```

Exemple d'utilisation de if

Faire un programme qui résout le problème suivant

1. $y = x$ si $x < 0$
2. $y = x^2$ si $x > 0$
3. $y = 10$ si $x = 0$

```
x=input('introduire la valeur de x ');
Introduire la valeur de x 6
if x<0
y=x;
end
if x>0
y=x^2;
end
if x==0
y=10;
end
disp('la valeur de y est: '),y
```

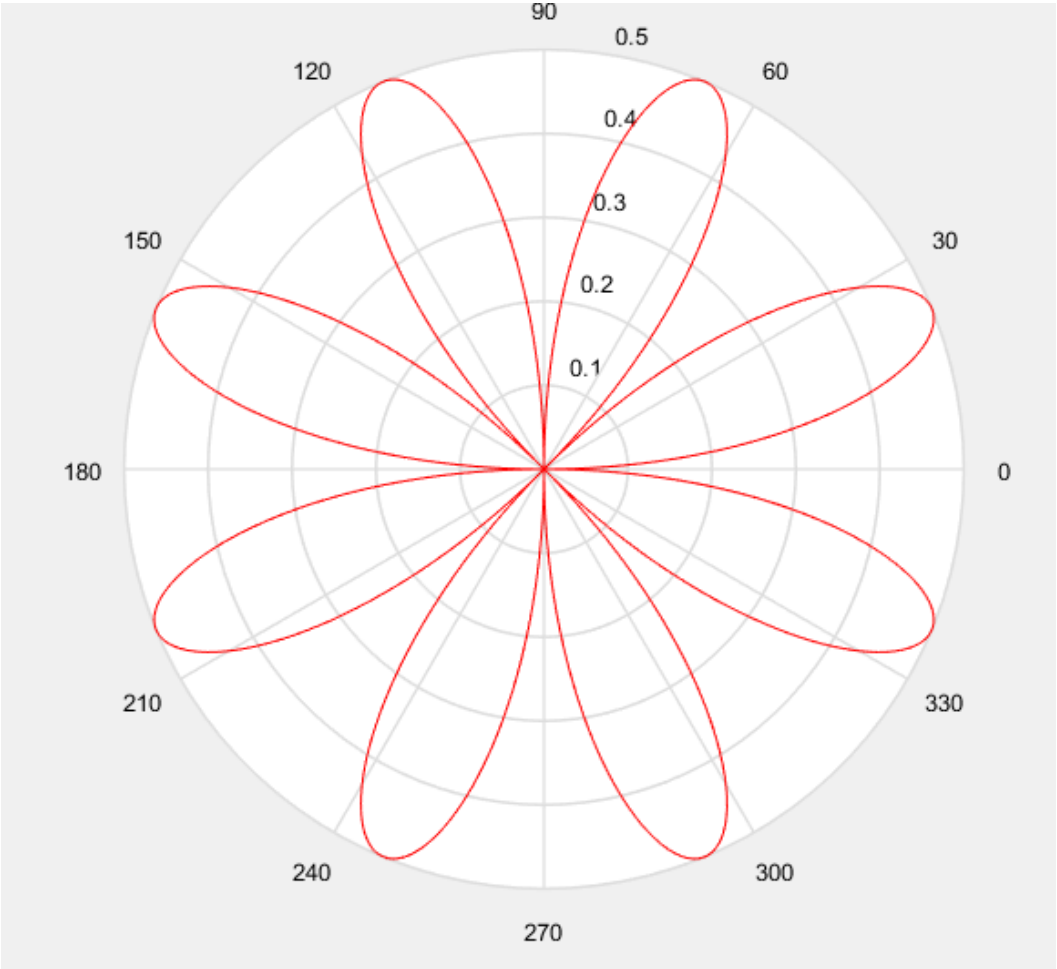
Exemple d'utilisation de if

Écrire un programme qui résout l'équation du second degré à valeurs réelles

```
% Programme de résolution de l'équation a*x^2+b*x+c=0
a = input ('Entrez la valeur de a : ');       % lire a
b = input ('Entrez la valeur de b : ');       % lire b
c = input ('Entrez la valeur de c : ');       % lire c
delta = b^2-4*a*c ;                           % Calculer delta
if delta<0
   disp('Pas de solution')                    % Pas de solution
elseif delta==0
   disp('Solution double : ')                 % Solution double
   x=-b/(2*a)
else
   disp('Deux solutions distinctes: ')                    %
Deux solutions
   x1=(-b+sqrt(delta))/(2*a)
   x2=(-b-sqrt(delta))/(2*a)
end
```

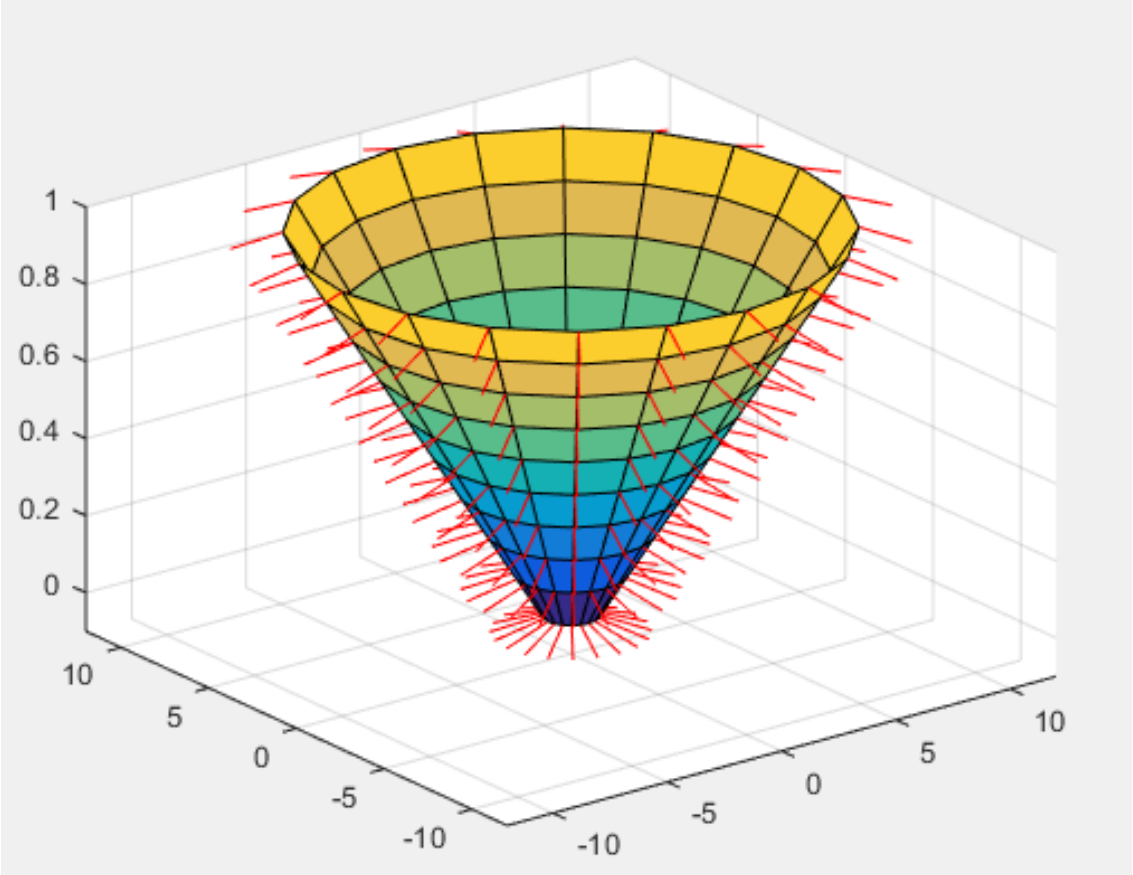
Graphique polaire

Il est aussi possible de dessiner des graphiques polaires sous Matlab.



Normales de surfaces

il est possible sous Matlab de représenter des surfaces à trois dimensions ainsi que



mamadou.salif.diallo@aims-senegal.org

Programme de la représentation d'une cône

```
%  Normales de surfaces
% Nouvelle Fenetre
figure ;
% Creation d'un cylindre 3 dimensions
[x,y,z] = cylinder(1 :10) ;
% Affichage des surfaces et normes
surfnorm(x,y,z)
% Etalonnage des axes
axis([-12 12 -12 12 -0.1 1]);
```