Support du cours

Interfaces graphiques sous Matlab

Préparé par : Mr N.Nasri

1 - Introduction :

MATLAB (MATrix LABoratory) est un logiciel basé sur le calcul matriciel. Son noyau est composée de librairies écrites au début en Fortran puis en C++. Il permet :
– le calcul numérique.
– le tracé de graphiques.
– la programmation.
– l'interfaçage avec d'autres langages (Fortran, C++).
– le développement d'interface utilisateur (GUI).
– l'utilisation es boites à outils (Toolboxes) spécialisées.

MATLAB présente deux modes de fonctionnement :
1. Le mode interactif : exécution des instructions entrées par l'utilisateur dans la fenêtre de commande de MATLAB (le Workspace).

2 . L e mode exécutif : exécution de ″programmes″ ou ″scripts″ qui sont des fichiers regroupant une suite d'instructions (les fichiers M).

2 - Première utilisation de Matlab :

Une façon efficace de découvrir Matlab est d'utiliser son aide en ligne.

- **help :** "help" tout seul donne la liste des aides générales possibles.
- **helpwin :** ouvre une fenêtre et donne accès à une aide détaillée.
- **help + nom d'une commande** : indique la syntaxe des graphes en 2D.
  Exemple : >> help plot
- **demo** : lance une démo générale de Matlab .
- **help demos :** donne une liste des démos existantes.

Quelques commandes importantes:

- **lookfor + nom de commande** : donne une liste de toutes les commandes qui ont un rapport avec la nom de commande écrite .

- **who** et **whos :** donne les noms de variables présentes dans l'espace de travail.
- **clear :** supprime les variables de l'espace du travail.
- **clc :** efface l'écran (sans toucher aux variables !)

- **exit** ou **quit :** permet de sortir de Matlab
- **CTRL** + **C** (touches du clavier) permet d'interrompre un programme (ie : récupérer la main).

Format des variables:

x = [4/3 1.2345e-6]

format short	1.3333	0.0000
format short e	1.3333e+000	1.2345e-006
format short g	1.3333	1.2345e-006
format long	1.33333333333333	0.00000123450000
format long e	1.333333333333333e+000	1.234500000000000e-006
format long g	1.33333333333333	1.2345e-006
format bank	1.33	0.00
format rat	4/3	1/810045
format hex	3ff5555555555555	3eb4b6231abfd271

Nota : on commence d'abord par définir le format, puis l'opération à effectuer ou le nombre à affiché.

3 - Les matrices dans Matlab :

Création de matrices :

>> A = [a_{11} …. a_{1m}; …. ; a_{n1} …. a_{nm}] **Exemple :** >> A=[1 2 3 4;2 3 4 1;3 4 1 2;4 1 2 3]

Quelques matrices prédéfinies :

>> **zeros**(i,j) : donne une matrice de zéros.
>> **ones**(i,j) : donne une matrice de uns .
>> **eye**(i,j) : donne une matrice identité.
>> **rand**(i,j) ou >> **randn**(i,j) : donne une matrice d'éléments aléatoires.

- i,j sont des entiers.
- pour randn : les éléments sont choisis avec la loi normale.

Quelques commandes importantes:

>> A(i, :) : désigne la $i^{iéme}$ ligne de la matrice A.
>> A(:, j) : désigne la $j^{ième}$ colonne de la matrice A.

>> A(i:j , :) : désigne la sous matrice formées des $i^{iéme}$ et $j^{ième}$ lignes de la matrice A.
>> A(:, i:2:j) : désigne la sous matrice formée des colonnes impaires de la matrice A.
Nota : le 2 est le pas considéré.

>> **size**(A) : permet d'obtenir la taille de la matrice A.
>> **max** (A) : désigne l'élément maximal de la matrice A.
>> **min** (A) : désigne l'élément minimal de la matrice A.
>> **mean** (A) : désigne la moyenne des éléments de matrice A.
>> **median** (A) : désigne la valeur médiane des éléments de matrice A.

>> **sort** (A) : tri par ordre croissant des éléments de matrice A.
>> **prod** (A) : produit des éléments de matrice A.
>> **sum** (A) : somme des éléments de matrice A.
>> **expm** (A) : exponentielle de matrice A. (**important :**A doit être carrée)
>> **sqrtm** (A) : racine carrée de la matrice A. (**important :**A doit être carrée)

>> **inv (A)** **:** la matrice inverse de A.
>> **det** (A) : déterminant de la matrice A. (**important :**A doit être carrée)
>> **eig** (A) : valeurs propres et vecteurs propres de la matrice A.
>> **null** (A) : noyau de la matrice A.
>> **rank** (A) : rang de matrice A.

Application 1 : Construction élément par élément
Ecrire l'exemple sous Matlab et commenter le résultat.
Exemple:
>>for i=1:3,for j=1:4, F(i,j)=i+(j-1)*3;end;end;
>> F

4 - Opérations sur les polynômes :

Nota : un polynôme se déclare comme un vecteur qui contient ses coefficients.
Exemple 1 : $B(x)=5.X^4+3.X^3+2.X$
Sous Matlab : >> B=[5 3 0 2 0] ou bien : >> B=[5,3,0,2,0]

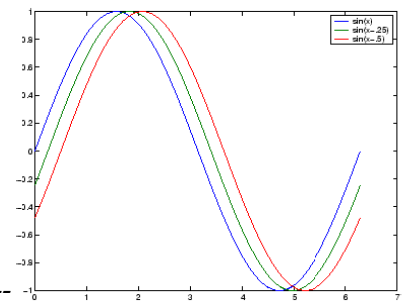
Quelques commandes importantes:
>> Roots (B) **:** racines du polynôme B.
>> **polyval (B,x)** **:** évaluer le polynôme (**exp:** avant de représenter son graphe).
>> **poly (v)** **:** reconstruit un polynôme à partir de ses racines.

5 – Graphisme dans Matlab :

Gestion des graphiques à 2d :
>> **Plot (y)** **:** représente le graphe de la fonction y.
>> **title** (' expression ') : affiche le titre du graphe.
>> **Xlabel** ('expression ') : affiche l'étiquette à l'axe des abscisses.
>> **Ylabel** ('expression ') : affiche l'étiquette à l'axe des ordonnées.
>> **grid on** : affiche la grille.
>> **grid off** : masque la grille.
>> **hold on** : affiche une seconde courbe dans une meme figure.
>> **clf** : efface le graphe .
>> **stem (y)** **:** représente la séquence de données (discrètes) y.

Application :

```
x = 0:pi/100:2*pi;
y = sin(x);
y2 = sin(x-.25);
y3 = sin(x-.5);
plot(x,y,x,y2,x,y3)
legend('sin(x)','sin(x-.25)','sin(x-.5)')
```



Le style et Couleurs des lignes :
>> **plot** (x,y,'*style_couleur_marker*')

• **C**ouleur : 'c', 'm' 'y' 'r' 'g' 'b' 'w' 'k'
[cyan, magenta, yellow, red, green, blue, white et black].

• **S**tyle : '-' '--' ':' '-.' **Exemple** : >> **plot** (y, '**linestyle**','—') ;

• **M**arker : '+' 'o' ' * ' 'x' 's' 'd' ' ^' 'v' '>' '<' 'p' 'h'

Utilisation du Handle :
Quand Matlab crée des objets, il leurs affecte un identifiant appelé "handle". Ce dernier est utilisé pour acceder aux propriétés de l'objet grace aux instructions : "**set** " et "**get** ".

Exemple :
```
>> x = 1:10;
>> y = x.^3;
>> h = plot(x,y);
Pour  changer de couleur , on ecrit :
set(h,'Color','red')
Pour  changer le style de ligne , on ecrit :
set(h,'LineWidth', 6);
```

6 - Gestion des Images :
Lire et afficher une Image : I = **imread** ('nom_image.ext');
figure, **imshow** (I)

Redimensionner une Image : J = imresize(I,0.6);
figure, **imshow** (J) % --- Affichage ---*
(Le 0.6 est le facteur d'échelle)

Retourner une Image : K = **imrotate** (I,30);
figure, imshow(K)
(Le 30 est le facteur d'échelle)

7 - Gestion de l'audio :
Lecture d'un son :

>> y = **wavread**('Nom.wav'); : Nom.wav Fichier qui se trouve dans le dossier Work.
>> **wavplay**(y, F) : F représente la fréquence.
% (Poser F=20000, puis changer de valeur pour voir son effet).
>> **sound**(y, F) : Convertir le signal y en un son.
>> **aviread** (y) : lire les fichiers AVI (Audio / Video Interleaved).

Enregistrer un son :

>> **wavwrite** (y, F, 'nom_fichier') : sauvegarde le signal y dans un fichier au format Wav.

Nota :

>> beep : émit un beep sonore.

8 - Programmation en Matlab :

Les fonctions élémentaires :

Un certain nombre de fonctions élémentaires sont prédéfinies : sin, cos, abs,...

```
>> x=[0:0.5:pi]
>> sin(x)

>> x=[0:0.2:10]
>> exp(x)
```

Fonctions et script :

- **Exemple de fonction :** écrire ce programme et l'enregistrer sous nom : **trinome.m**

```
% --- calcul des solutions de l'équation a*x^2+b*x+c=0 ---*

function r =trinome(a,b,c)
delta=b^2-4*a*c;
if delta==0
   r(1)=-b/2;

else delta > 0
   r(1)=(-b-sqrt(delta))/2*a;
   r(2)=(-b+sqrt(delta))/2*a;

end
disp(['delta == ', num2str(delta)]);
disp(['la solution est ', num2str(r)]);
```

- **Utilisation de la fonction dans un script:** écrire ce programme et l'enregistrer sous nom : **trinome3.m**

```
% --- calcul des solutions de l'équation a*x^2+b*x+c=0 ---*
disp ('Ceci est un script qui calcul des solutions de l''équation a*x^2+b*x+c=0');

disp ('Donner la valeur de a')
a=input ('a = ');
disp ('Donner la valeur de b')
b=input ('b = ');
disp ('Donner la valeur de c')
c=input ('c = ');
trinome (a,b,c);          % appel de la fonction aux parameters a,b,c
```

Remarque :

Cette fonction ne prend pas en considération le cas ou (a=0, b=0, c=0=).

- Modifier trinome2.m afin d'avoir une solution complète.

- **Exemple de script:** écrire ce programme et l'enregistrer sous nom : **trinome2.m**

```
% --- calcul des solutions de l'équation a*x^2+b*x+c=0 ---*

disp ('Ceci est un script qui calcul des solutions de l''équation a*x^2+b*x+c=0');

disp ('Donner la valeur de a')
a=input ('a = ');
disp ('Donner la valeur de b')
b=input ('b = ');
disp ('Donner la valeur de c')
c=input ('c = ');

delta=b^2-4*a*c;
if delta==0
   r(1)=-b/2;

else delta >0
   r(1)=(-b-sqrt(delta))/2*a;
   r(2)=(-b+sqrt(delta))/2*a;

end

   disp (['delta == ', num2str(delta)]);
   disp (['la solution est ', num2str(r)]);
```

9 - Conception de l'interface graphique GUI : [Graphical User Interface]

Avantage des GUI : La productivité.

Basculer entre plusieurs applications sans perdre beaucoup du temps.
(Exemple : dans l'environnement Windows, passer entre Word, Explorer, Excel, Media player, .. se fait de manière très rapide).

Etapes de Conception :

Il clair qu'il ne suffit pas de s'asseoir et de se mettre à programmer. Il faut passer par certaines étapes :

Prendre un cahier et un crayon

- **Planifier :** définir le but de l'interface
- **Analyse :** définir le problème à résoudre.
  (Rassembler les équations mathématiques nécessaires)
- **Le Cahier de charge :** définir les fonctions et taches en détails.
  (Enumérer les différentes taches voulues et rassembler les données nécessaires)
- **La Conception :** compléter le cahier de charge avec les détails de conception :
  - *Développer le design.*
  - *Exemples d'écran (taille des fenêtres...).*
  - *Disposition des boutons.*
  - *Les couleurs, la forme d'écriture.*

  (Ces détails dépendent de votre cible [A qui est destiné votre interface).

Maintenant on passe à l'étape de programmation :

- **Le Codage :** qui ne se termine jamais (ie : il y a toujours des modifications).
- **Le Test :** tester les différentes fonctions avec différentes « inputs ».
  (Il conseillé de refaire le test plusieurs fois).

Principes de Conception :

Garder à l'esprit **les sept principes** qui ont guidé l'écriture de Windows.

- **L'utilisateur doit avoir le contrôle :** l'utilisateur doit sentir que c'est lui qui à l'initiative d'une action au lieu de réagir à l'ordinateur.

- **Etre direct :** une image vaut mille mots. (les différentes fonctions doivent être visuellement intuitives).

- **La Cohérence :** votre interface doit être cohérente avec Windows, l'utilisateur ne doit pas passer son temps à apprendre votre application(c'est la cohérence externe).
  (L'interface doit présenté une cohérence interne : entre les différentes fenêtres).
  Exemple : Si le raccourci clavier **Ctrl + C** : sert à copier dans une fenêtre, il faut pas l'utiliser pour une autre fonction dans une autre fenêtre.

- **Etre magnanime** : si l'on est sur le point d'effectuer une action qui formatera le disque dur, une boite de dialogue apparaît est prévient de ce qui risque d'arriver.
  ie : Toutes les actions de l'utilisateur doivent être réversible ou corrigible ;et il doit être prévenu à l'aide des boites du dialogue.

- **Du répondant :** « faire toujours savoir à l'utilisateur ce qui se passe » .
  - *Barre de chargement*
  - *Boite de dialogue*
  - *Indices visuels et sonores*
  - *Messages d'erreur*
  - *Changement de la forme du curseur*

- **De l'esthétique :**
  - *Couleur*
  - *Design*
  - *Disposition des objets*

- **De la simplicité :** la GUI doit être facile d'apprentissage et d'usage . ie : permettre l'accès à toutes les fonctions et informations de l'application, tout en gardant la manipulation aussi simple que possible.
  Eviter de trop écrire : écrivez **Nom** au lieu de **Nom du client**.
  (Il faut s'exprimer avec un minimum de mots sans pour autant perdre en sens).

Exemple de GUI par programmation :

On veut reprendre la fonction **trinome.m** et l'utiliser dans une interface.
Ecrire ce code et sauvegarder le sous le nom : second_degre.m .

```
% --- calcul des solutions de l'équation a*x^2+b*x+c=0 ---*
close all
clear all

%création de la fenêtre graphique
figure (1)

% titre principal
txt_principal = uicontrol(gcf, ...
                 'style', 'text', ...
                'position',[30 340 500 45],...
                'string',' Calcul des solutions de l''équation : a*x^2+b*x+c=0') ;

%  création des zones éditables
ed_a = uicontrol(gcf, ...
                'style', 'edit', ...
                'position',[275 230 100 20])

ed_b = uicontrol(gcf, ...
                'style', 'edit',...
                'position',[275 200 100 20])

ed_c = uicontrol(gcf, ...
                'style', 'edit', ...
                'position',[275 170 100 20])

ed_delta = uicontrol(gcf, ...
                'style', 'edit',...
                'position',[275 110 200 20])

ed_sol = uicontrol(gcf,...
                'style', 'edit', ...
                'position',[275 80 200 20])

% création des boutons
% le bouton Fermer
clabel = 'Fermer'
cpos = [370 30 180 20] ;
ccallback = 'close all ; clear all' ;


bp_fermer = uicontrol(gcf, ...
                'style', 'push', ...
                'position',cpos,...
```

```
                'string',clabel,...
                'callback', ccallback) ;

% le bouton Résolution
clabel = 'Résolution'
cpos = [130 140 180 20] ;

ccallback=['a=str2num(get(ed_a,''string'')),'...
        'b=str2num(get(ed_b,''string'')),'...
        'c=str2num(get(ed_c,''string'')),'...
       'trinome(a,b,c),'...        % le fichier modifié
       'load delta,'...
       'load solution,'...
       'set(ed_delta,''string'',num2str(delta)),'...
       'set(ed_sol,''string'',num2str(r))']

 bp_resolution = uicontrol(gcf, ...
                        'style', 'push',...
                        'position',cpos,...
                        'string',clabel,...
                        'callback',ccallback);
```

Nota :

La fenêtre peut être personnalisé :

```
Hfenetre= figure ('Name', 'nom_fenetre' , …
                        'NumberTitle', 'off', …
                        'Resize' , 'off', …
                        'Position', [x y largeur hauteur])
```

10 - Création d'une interface graphique avec MATLAB GUIDE

L'environnement GUIDE de Matlab permet de développer des GUI, des interfaces graphiques. Un GUI est défini dans Matlab par deux fichiers dépendants, respectivement une figure et un script. La programmation d'un GUI utilise des callbacks, et les échanges de données s'opèrent avec des handlers.

Éléments de base

- Fiche et composants visuels (aspect visuel).
- Code définissant fonctionnement de l'interface (aspect fonctionnement).

Création d'une nouvelle interface

– Ligne de commande MATLAB : tapez **« guide »**
– Dans la fenêtre **« GUIDE Quick Start »,** choisissez l'onglet **« Create New GUI ».**
– Une fiche vierge quadrillée apparaît à l'écran sur laquelle vous placerez les composants visuels de l'interface graphique.

Palette des composants visuels :

- « Push buttons » : (boutons poussoirs)
  - Un seul état (stable).
  - Identifié avec un court texte (paramètre 'String').
  - Événement à traiter: clic ou relâchement du bouton gauche de la souris.

- « Checkboxes » : (cases à cocher)
  - Deux états possibles.
  - Boutons d'un même groupe sont indépendants (plusieurs options peuvent être sélectionnées simultanément).

- « Radio buttons » : (boutons radio)
  - Deux états (comme case à cocher).
  - Boutons d'un même groupe sont mutuellement exclusifs (une seule option peut être sélectionnée à la fois).

- « Frames » : (cadres)
  - Bordure rectangulaire délimitant un groupe de contrôles.

- « Static text » : (champ de texte fixe)
  - Affichage de texte.

- « Edit text » : (champ de texte éditable)
  - Saisie de texte.
  - Pas pour affichage.

- « Pop-up menus » : (menus déroulants)
  - Choix d'un item parmi une liste.
  - Seul l'item sélectionné est affiché.
  - Événement à traiter : item cliqué par la souris.

- « Sliders » : (barres de défilement)
  - Choix d'une valeur numérique à l'intérieur d'un intervalle.

- Peuvent être orientées horizontalement ou verticalement.
- Événement à traiter : déplacement de la barre.

- « Listboxes » : (listes)
  - Choix d'un item parmi une liste.
  - Un groupe d'items, dont l'item sélectionné, est affiché.
  - Événement à traiter: item choisi par clic de la souris.

- Menus : (à partir du MenuEditor)
  - Choix d'un item parmi une liste d'options permises ou non.
  - Possibilité de clé de raccourcis.
  - Possibilité de sous-menus.
  - Événement à traiter: item de menu choisi par clic de souris.

- Axes :
  - Permet d'afficher un graphique tracé par **MATLAB**
  - Permet d'afficher des fichiers image
  - Événement à traiter: l'utilisateur clique sur la souris lorsque le curseur se trouve à l'intérieur des bornes du composant.

Accès aux données associées aux composants visuels

À tout objet d'interface, Matlab associe un pointeur qui permet d'accéder aux propriétés, le handler.

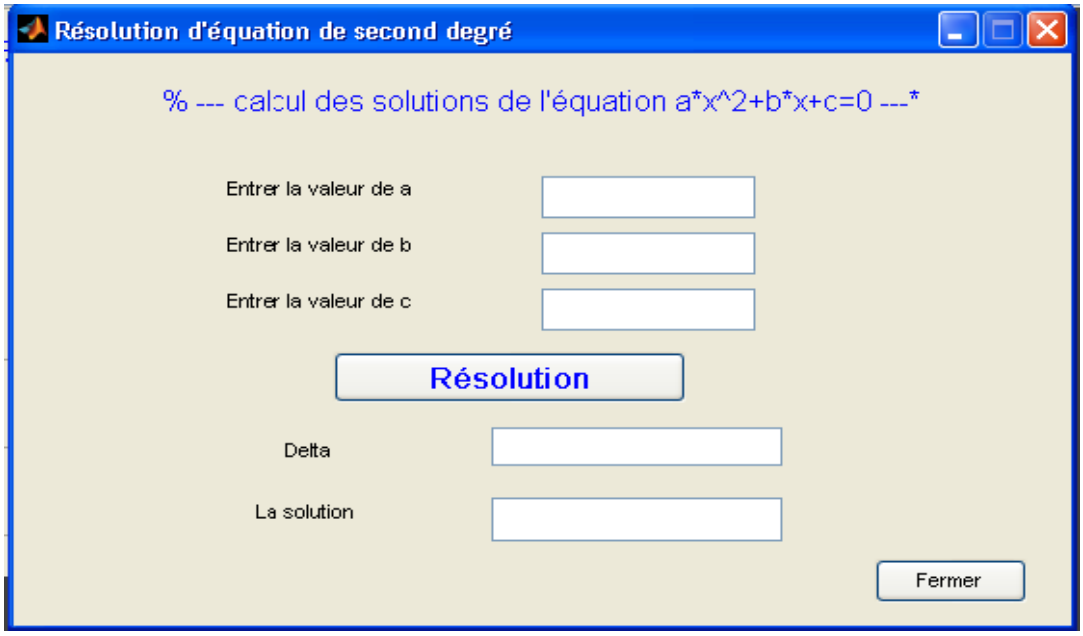
Certains handlers sont déjà définis par défaut :

- **gcf** : handler de la figure courante (get current figure).
- **gca** : axes courants de tracé (get current axes).
- **gcbf** : la figure cliquée (get callback figure).
- **gcbo** : l'objet qui appelle (celui sur lequel on a cliqué).
- **0** : la fenêtre interpréteur Matlab, ou racine.

Certaines instructions nécessitent un handler :

- **get** et **set :** respectivement pour lire et pour changer les valeurs des propriétés.
- **findobj** : pour retrouver le handler d'un objet.
- **propedit** : pour éditer les propriétés d'un objet.
- **delete** : pour effacer un objet.

- **Exemple de GUI :** construire sous GUIDE la fenêtre suivante et l'enregistrer sous nom : **second –degre2.m**



Ecrire au niveau du bouton « Résolution » le programme suivant :

```
h=findobj (gcbf,'tag','edit_a');
a=str2num (get (h,'string'));
h=findobj (gcbf,'tag','edit_b');
b=str2num (get (h,'string'));
h=findobj (gcbf,'tag','edit_c');
c=str2num (get (h,'string'));

trinome(a,b,c)     % utilisation de la fonction  programmée auparavant.

load delta;
load solution;
% --- Affichage -----*
h=findobj (gcbf,'tag','edit_delta');
set (h,'string',num2str(delta));
h=findobj (gcbf,'tag','edit_sol');
set (h,'string',num2str(r));
```

Modifier le fichier **trinome.m** en ajoutant ces deux instructions **:**

```
save delta delta;
save solution r;
```

Annexe :

Exemple : les vecteurs

Sur un ensemble de notes entre 0 et 20.

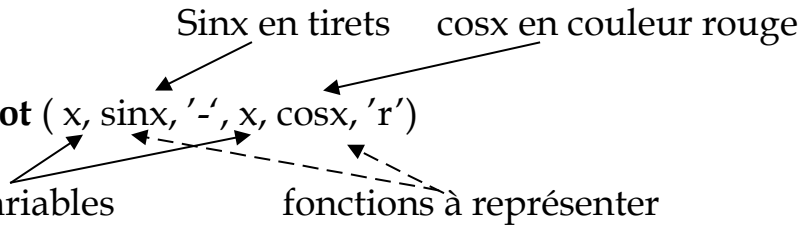
- Calculer le nombre de notes>= 10.
- Calculer le nombre de notes >=moyenne des notes.

Le programme :

```
disp (' *** N''oubliez pas les crochets ***') ;
Note = input('Note == ');

Taille= length(Note);
Somme=0;

for i=1:Taille
        Somme=Somme+Note(i);
end

moy=Somme/Taille ;
N10=0;
Nmoy=0;

for i =1:Taille
    if  Note (i)>=10
            N10=N10+1;
    end
    if Note (i)>=moy
            Nmoy=Nmoy+1;
    end
 end

disp (['nombre de notes sup à 10 :' num2str(N10)]);
disp (['nombre de notes sup à la moyonne :' num2str(Nmoy)]);
```

Exemple : Graphe (légende)

X=0 :0.1 :1



Xlabel (' titre') % titre = fonction sinx/cosx
ylabel (' titre') % titre = x
legend ('sinx','cosx',**1**) % loin droit haut.
2 % loin gauche haut.
3 % loin droit bas.
4 % loin gauche bas.
5 % hors l'axe du graphe.

Exemple : diagramme circulaire

x= [100 10 25 45]
detache= [1 0 0 1]
etiquette= {'investissement','taxe','frais', bénéfice'}
pie(x, detache, etiquette)

3D-Looking :
pie3(x, detache, etiquette)

Exercice : Paraboloïde

Tracer le paraboloïde d'équation : $z^2=x^2+y^2$
Pour : $-50 \le x \le 50$ et $-50 \le y \le 50$

Solution 1 : (02 boucles)

```
N=50 ;
x=-N:N ;
y=-N:N ;
figure ;
for k=1:2*N+1
    for l=1 :2*N+1
       z1 (k, l)=sqrt(x (k) ^2+y (l) ^2) ;
    end
end
meshc(x,y,z1) ;
xlabel ('x') ; ylabel ('y') ; zlabel ('z') ;
title ('Exemple de tracé d'un paraboloïde');
```