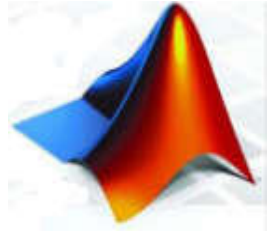
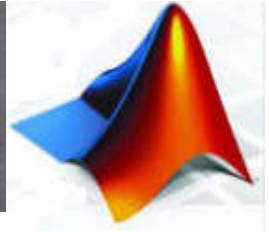


Programmer avec Matlab



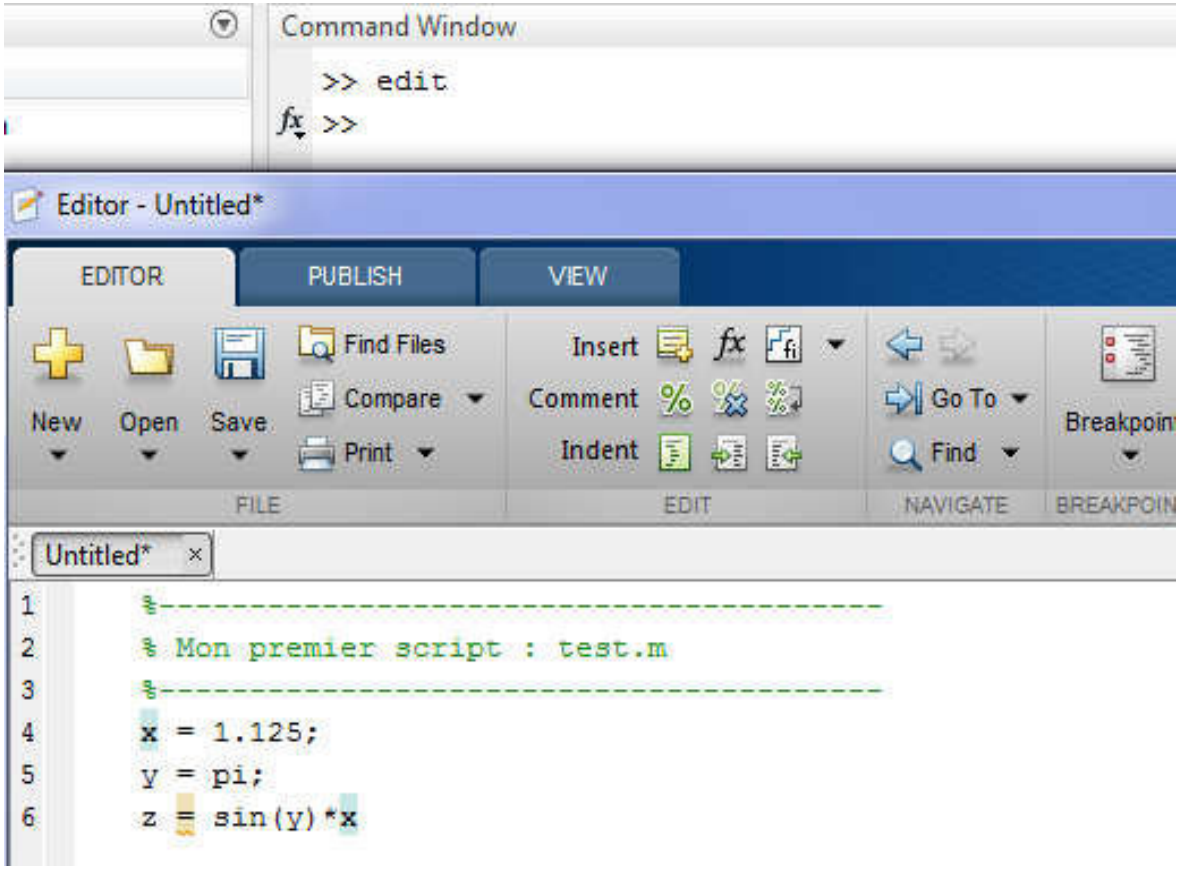
Mode de programmation

1. Scripts et m-files:

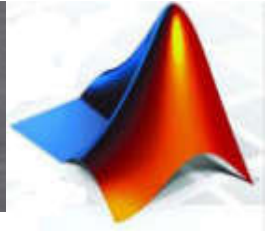
- ✓ Un **script** est assez fastidieux, est un ensemble d'instructions MATLAB qui joue le rôle du programme principal
- ✓ MATLAB permet d'enregistrer le texte d'un **script** sous forme d'un fichier de texte appelé **m-file**.
- ✓ Les **m-files** servent en particulier à définir de nouvelles fonctions.
- ✓ Une grande partie des fonctions prédéfinies de MATLAB sont stockées sous forme de m-files dans la toolbox matlab.
- ✓ Les **m-files** peuvent être crées par n'importe quel éditeur. Cependant, dans les versions récentes de MATLAB, il existe un éditeur intégré que l'on peut appeler à partir du menu file ou à partir de la barre de menu de la fenêtre de commande (avec la commande **edit**). >> edit

Mode de programmation

1. Scripts et m-files (exemple):



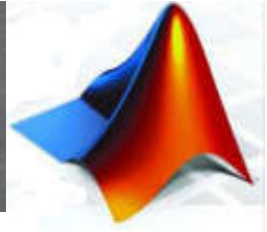

```
>> test

z =

    1.3777e-16

>>
```



Mode de programmation

2. Fonctions définies par l'utilisateur :

- ✓ Les fichiers de fonctions permettent à l'utilisateur de définir des fonctions **(appelées fonctions utilisateur)** qui ne figurent pas parmi les fonctions Matlab incorporées et de les utiliser de la même manière que ces dernières.
- ✓ Ils sont également un élément important dans la programmation d'applications où les fonctions jouent le rôle des fonctions et procédures des langages de programmation usuels.
- ✓ Les fonctions sont des enchaînements de commandes Matlab regroupées sous un nom de fonction permettant de commander leur exécution.
- ✓ On peut mettre dans une fonction un groupe de commandes destine a être exécuté plusieurs fois au cours du calcul avec éventuellement des valeurs de paramètres différents.
- ✓ La fonction peut aussi être chargée de réaliser un calcul avec un certain algorithme, qui pourra être remplacé par un autre plus rapide ou plus précis, en changeant simplement le nom de la fonction dans le programme appelant.



Mode de programmation

2. Fonctions définies par l'utilisateur - Suite - :

✓ Enfin, dès que le programme est un peu long et compliqué, il est souhaitable de le découper en fonctions, correspondant a des étapes pour améliorer la lisibilité et la compréhension de l'algorithme.

la syntaxe de définition d'une fonction externe est la suivante :

function [$vars_1$,$vars_2$,...,$vars_n$] = *nom_fonction* ($vare_1$,$vare_2$,...$vare_m$)

Séquence d'instructions

- **$Vars_1$,$vars_2$,...,$vars_n$:** sont les variables de sortie de la fonction
- **$Vare_1$,$vare_2$,...,$vare_m$:** sont les variables d'entrée de la fonction
- **Séquence d'instruction :** est le corps de la fonction
- **nom_fonction :** est le nom de la fonction

Mode de programmation

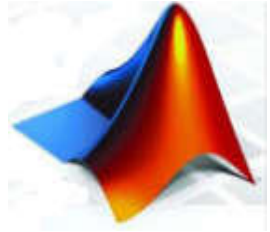
2. Fonctions définies par l'utilisateur (exemple) :

modulo.m*

```
1   function [r,q] = modulo(a,n)
2
3   % Calcule la valeur de a modulo n en prenant pour systeme de residus
4   % 1, ... , n au lieu de 0, ... , n-1.
5   %
6   % appel : [r,q] = modulo(a,n)
7   %
8   % Arguments de sortie :
9   %  r : le residu
10  %  q : le quotient
11
12  q = floor(a./n); % floor pour retourner un nombre d'entier approximative du a./n
13  r = a - n*q;
14
15  % si le reste de la division entiere vaut 0, le residu vaut par convention n
16  if r == 0, r = n;
17  end
```

Command Window

```
>> a = 10 ; n = 4;
>> [r,q]=modulo(a,n)

r =

     2


q =

     2
```



Graphisme

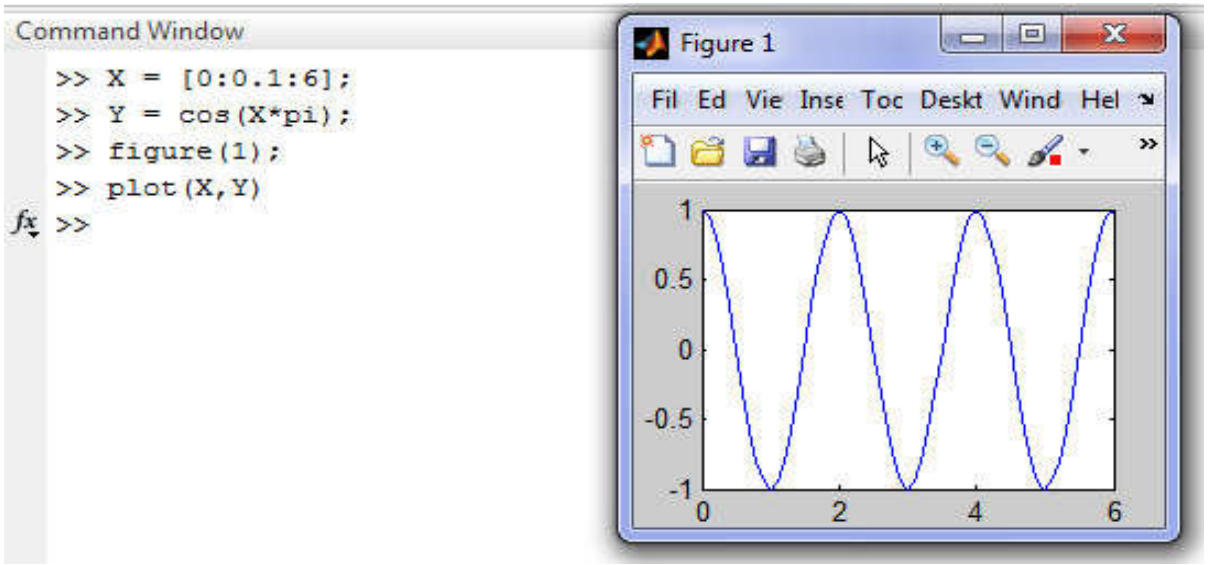
Graphisme

1. Graphisme 2D :

Le but de ce cours est de vous donner les outils permettant d'utiliser Matlab pour rédiger des documents scientifiques. Une compétence importante est donc la capacité à réaliser des graphs de qualité présentant clairement vos données.

1.2. Graphique d'une fonction :

Le graph le plus simple est l'affichage d'une fonction d'une variable réelle. Voyons comment tracer f(x) = cos pi*x.



Graphisme

1. Graphisme 2D :

1.2. Graphique de plusieurs fonctions :

Si l'on souhaite placer deux courbes sur le même graph on utilise la commande **hold on**

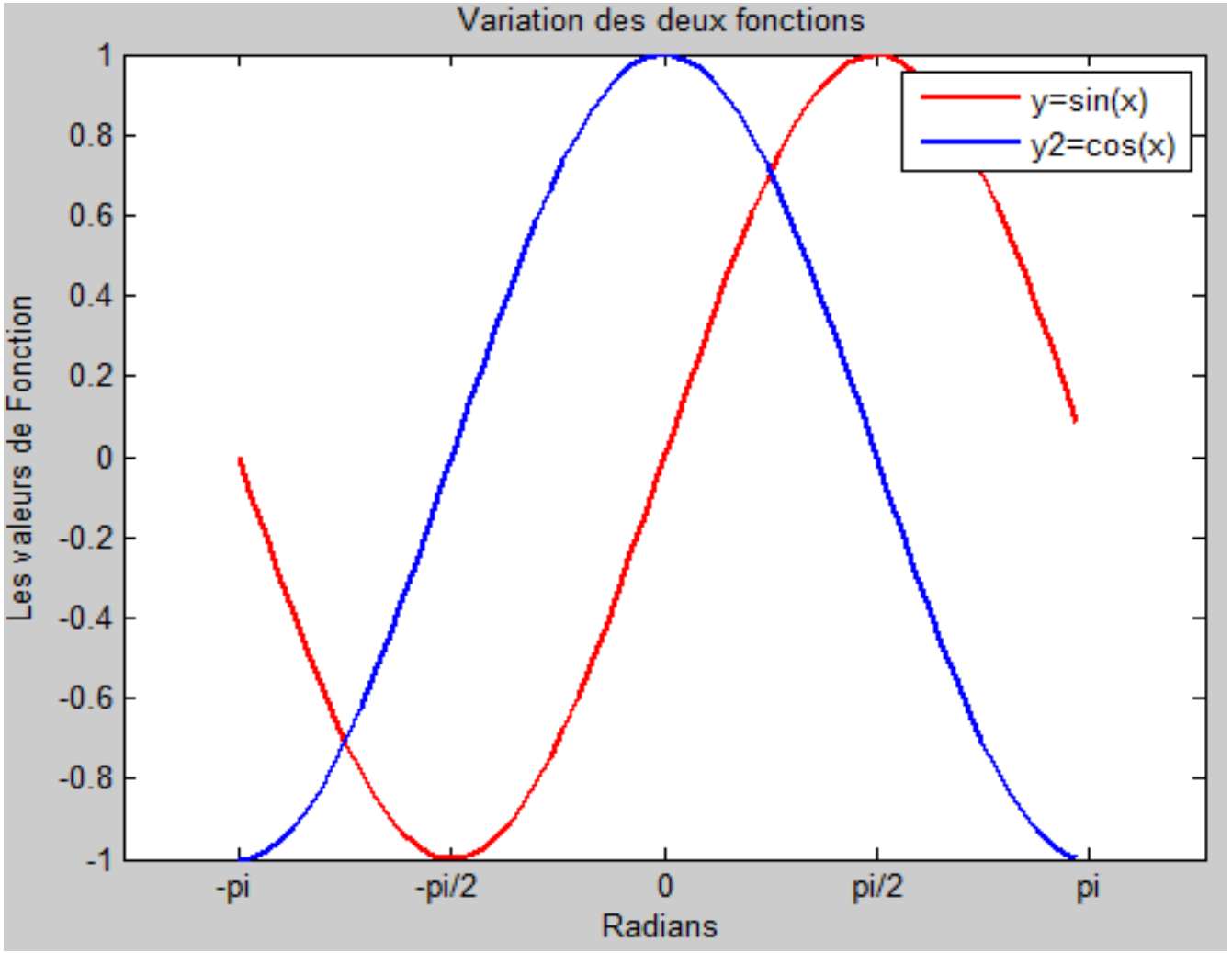
Command Window

```
>> x = [-pi:.1:pi];
y = sin(x);
y2 = cos(x);
p1=plot(x,y);
hold on
p2=plot(x,y2);
set(p1,'Color','red','LineWidth',2)
set(p2,'LineWidth',2)
set(gca,'XTick',-pi:pi/2:pi)
set(gca,'XTickLabel',{'-pi','-pi/2','0','pi/2','pi'})
xlabel('Radians'); title('Variation des deux fonctions');
ylabel('Les valeurs de Fonction'); legend('y=sin(x)','y2=cos(x)');
```

Graphisme

1. Graphisme 2D :

1.2. Graphique de plusieurs fonctions :



Graphisme

1. Graphisme 2D :

1.2. Graphique de plusieurs fonctions :

Commandes de la mise en forme des graphes	Définition
Hold on	Pour rassembler plusieurs courbes dans la même figure
xlabel(' ')	Pour donner un label à l'axe des abscisses
Ylabel(' ')	Pour donner un label à l'axe des cordonnées
Legend (' ')	Pour mettre une légende
Title(' ')	Pour mentionner un titre au figure
LineWidth	largeur de la ligne. La valeur est un nombre réel (en point)
LineStyle	type de ligne. La valeur est un code par exemple : '--' pour pointillés.
Color	Pour mentionner la couleur de la ligne

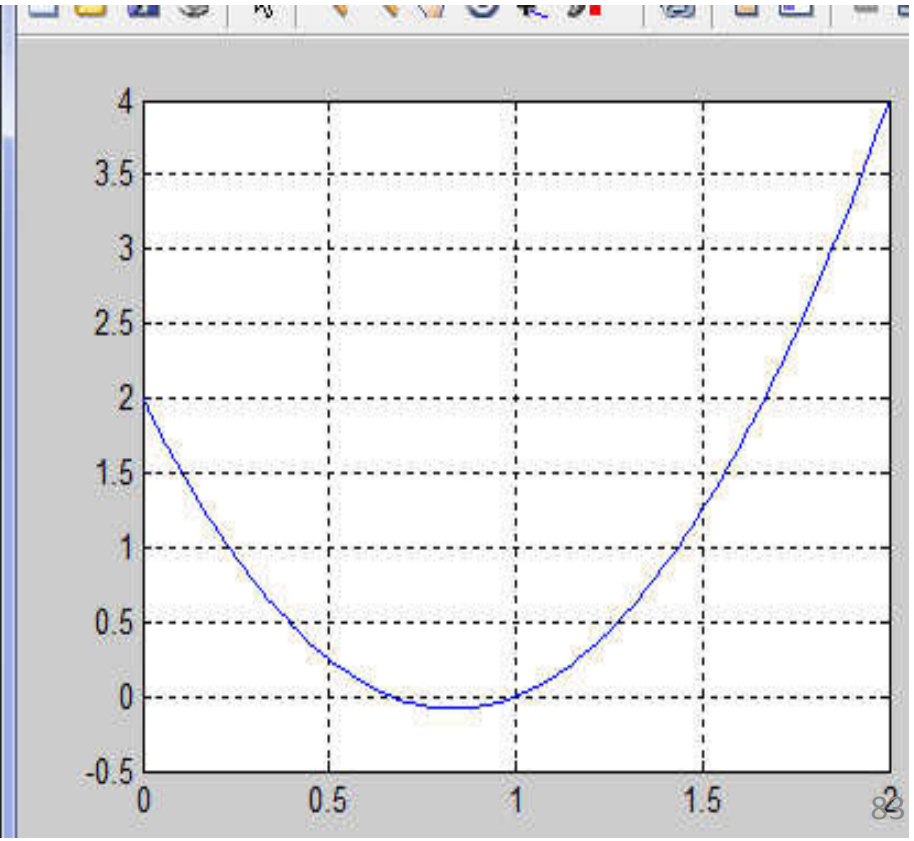
Graphisme

1. Graphisme 2D :

1.3. Commande fplot :

Première méthode : >> fplot ('f(x)' , [intervalle de x])

```
Command Window
>> fplot ( '3*x^2 - 5*x + 2' , [ 0 2 ] )
>> grid on
>>
```



Graphisme

1. Graphisme 2D :

1.3. Commande fplot :

Deuxième méthode : Il faut créer le fichier .m de la fonction :

fon.m

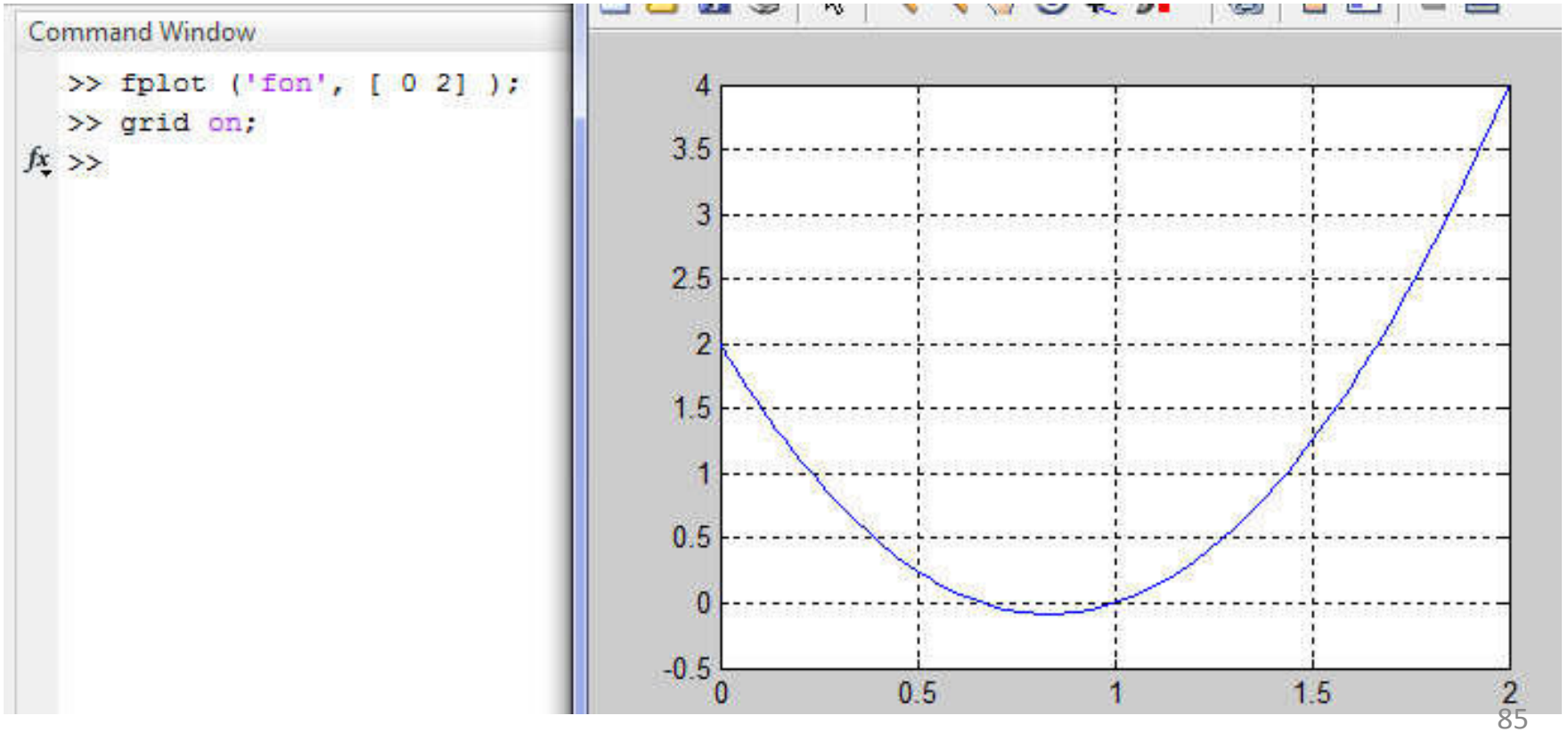
```
1   function y=fon(x)
2   p = [3 -5 2];
3   y = polyval (p,x);
4   end
```

Graphisme

1. Graphisme 2D :

1.3. Commande fplot :

Deuxième méthode :

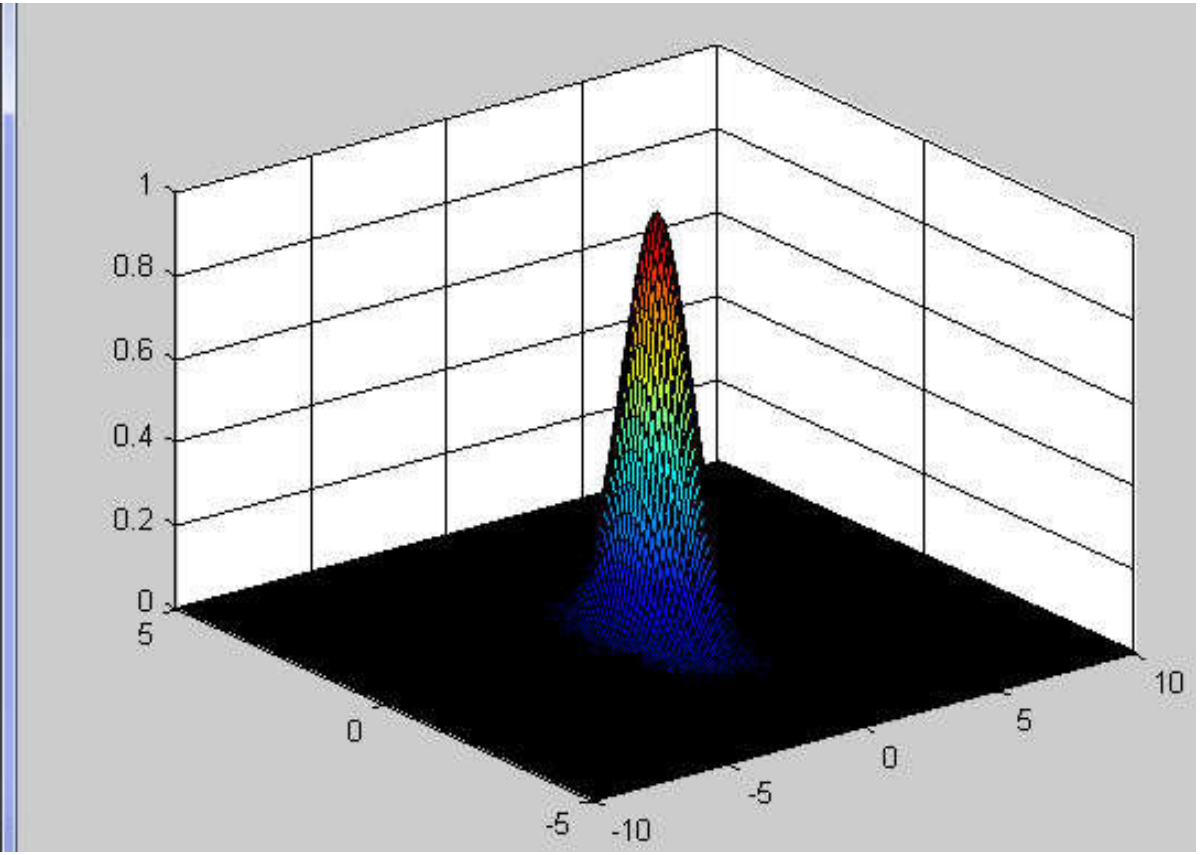


Graphisme

2. Graphisme 3D :

- Graphe Surfacique -

```
Command Window
>> x = -10:0.1:10;
y = -5:0.1:5;
[X,Y] = meshgrid(x,y);
>> g = exp(-X.^2).*exp(-Y.^2);
>> figure(1)
surf(x,y,g)
fx >>
```

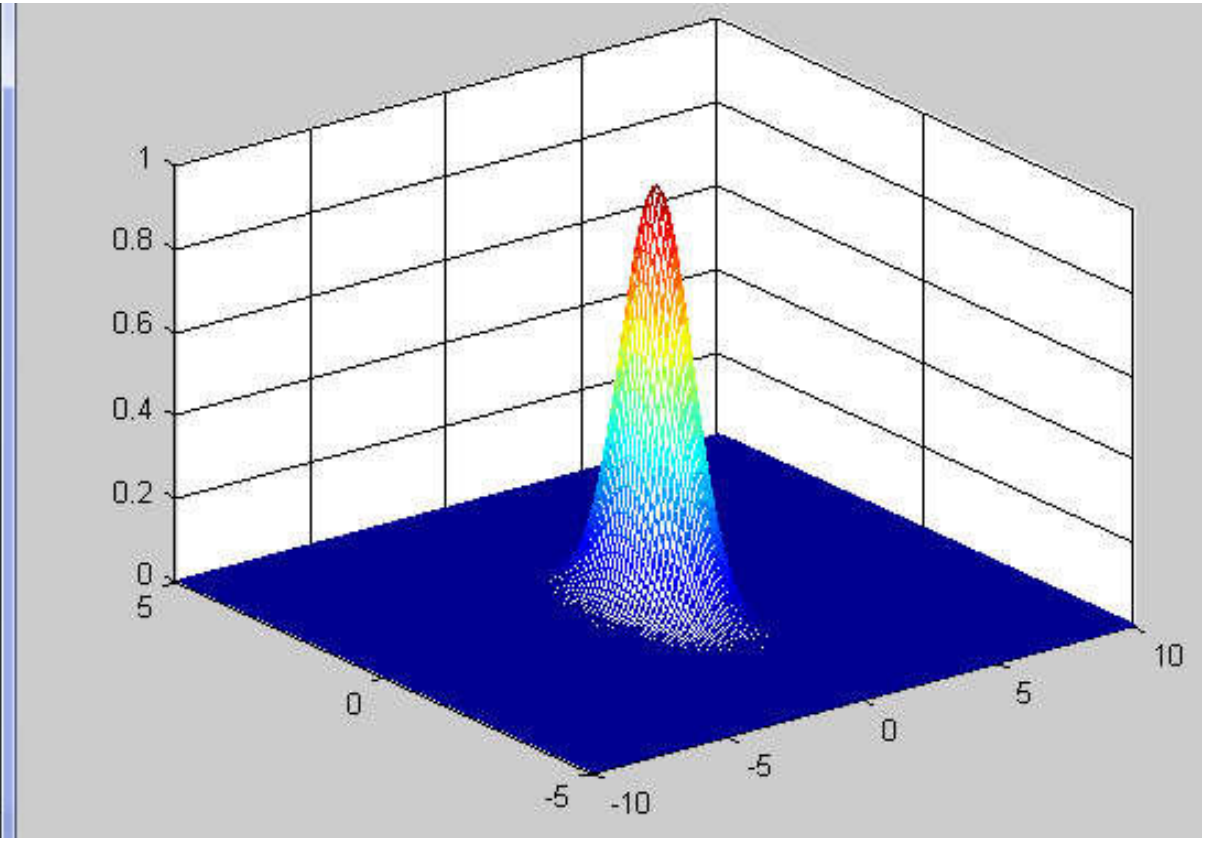


Graphisme

2. Graphisme 3D :

- Graphe en mesh -

```
Command Window
>> x = -10:0.1:10;
>> y = -5:0.1:5;
>> [X,Y] = meshgrid(x,y);
>> g = exp(-X.^2).*exp(-Y.^2);
>> mesh(x,y,g)
>>
```

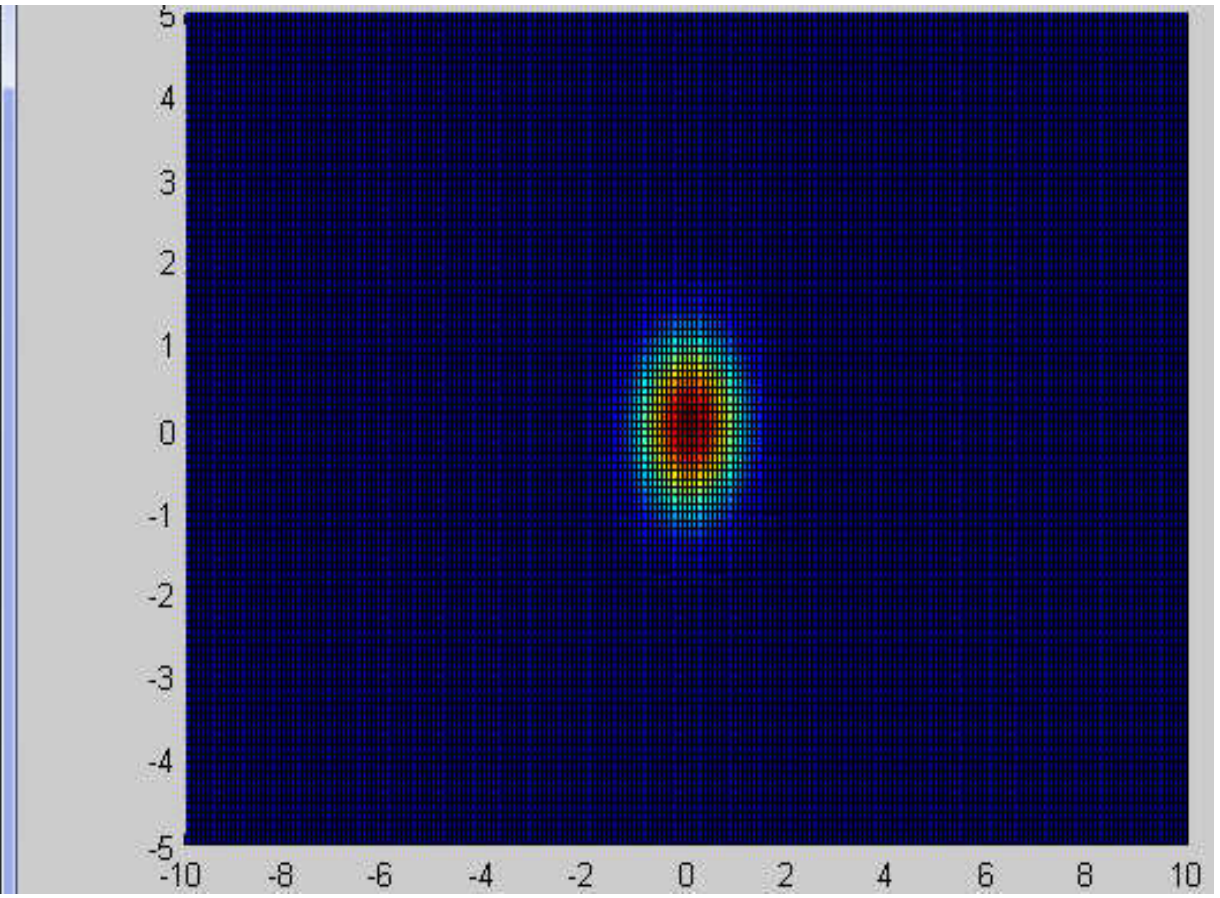


Graphisme

2. Graphisme 3D :

- Graphe en pcolor -

```
Command Window
>>  x = -10:0.1:10;
y = -5:0.1:5;
[X,Y] = meshgrid(x,y);
g = exp(-X.^2).*exp(-Y.^2);
>> pcolor(x,y,g)
>>
```



Graphisme

2. Graphisme 3D :

- Graphe en contour -

```
Command Window
>>   x = -10:0.1:10;
y = -5:0.1:5;
[X,Y] = meshgrid(x,y);
g = exp(-X.^2).*exp(-Y.^2);
>> contour(x,y,g)
fx >>
```