Prise de notes
Algorithmique Mathématiques – José OUIN
Introduction à l'algorithmique en SCILAB.

Hyper Important : Les lignes de commandes commençant par // ne sont pas interprétées par Scilab.

Algorithme : Suite finie d'opérations élémentaires constituant un schéma de calcul ou de résolution d'un problème donné.
Un algorithme n'est pas un programme. Un algorithme résout un problème donné indépendamment des particularités du langage de programmation.
Le programme correspond à l'implémentation d'un algorithme au moyen d'un langage donné. Il s'agit de la mise en œuvre des instructions.

Eléments de base d'un algorithme
Etape 1 : Préparation du traitement
Plusieurs types de données :
Données numériques
- Données saisies au clavier par l'utilisateur
- Lecture d'un fichier contenant des nombres

Données graphiques
- Lecture de la position du pointeur de la souris

Données textuelles
- Données saisies au clavier par l'utilisateur
- Lecture d'un fichier contenant du texte

Etape 2 : Traitement
Il s'agit de déterminer toutes les instructions (ou opérations) à donner pour mettre en œuvre la résolution du problème et obtenir les résultats attendus.

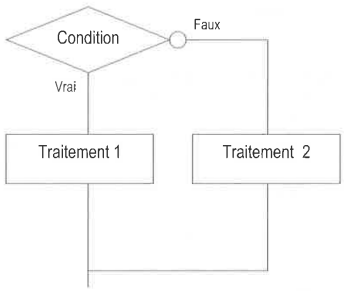
Etape 3 : La sortie des résultats
Plusieurs types de résultats
- Affichages à l'écran (nombres, textes, graphiques-
- Imprimés sur du papier
- Ecrits dans un fichier

Représentation graphique des algorithmes : ISO 5807

Exemple :

L'organigramme ci-contre représente une structure alternative :

Si la "Condition" est vérifiée **Alors**
Effectuer le "Traitement 1".
Sinon
Effectuer le "Traitement 2".
FinSi



Notion de pseudo code :
Le pseudo code est une façon de décrire un algorithme sans référence à un langage de programmation en particulier. Il ressemble cependant à un langage de programmation authentique mais dont on aurait retiré la plupart des problèmes de syntaxe,

Le pseudo code est susceptible de varier légèrement d'un programmeur à un autre.

Exemple :

```
Début
    d prend la valeur b^2-4*a*c
    Si d < 0 Alors
        Afficher : " Aucune solution "
    Sinon
        Afficher : " Une ou deux racines "
    Fin de si
Fin
```

Instruction pour traiter les variables :

Note : le point-virgule " ; " est optionnel. Il permet de ne pas afficher la valeur de la variable précédente dans la console au cours du déroulement du programme.

Affectation :

PC : d prend la valeur 5 ;

Scilab : d = 5 ;

Saisie :

PC : Saisir p ; //p est un réel.

Scilab : p = input(" Saisir la valeur de p ")

Note :

Trois exemples :

```
w = input ("Entrer votre prénom : ","string) ;
u = input ("Entrer les bornes de I " intervalle. [a , b] = ") ;
p = input ("Entrer la valeur de la précision. p = ") ;
```

La variable w est une chaine de caractère (string) contenant le prénom saisi

La variable u est un vecteur contenant deux nombres réels.

La variable p est un nombre réel.

Ecriture :

PC : Afficher " message "

Scilab :

Fonction disp()

disp("message") ;

disp(a) ; //Avec a, une valeur numérique

Fonction printf()

Pour une chaine de caractères

printf(" %s\n ", " f doit changer de signe sur l''intervalle [a,b] ")

La chaine de caractère " %s\n " est appelée chaine de formatage.

%s indique à la fonction printf() qu'il s'agit de l'affichage d'une haine de caractères

\n spécifie un retour à la ligne suite à cet affichage (par retour chariot (backtrack)).

Pour une valeur numérique :

printf(" Une racine double : x = %f\n",x) ;

%f indique à la fonction printf() qu'il s'agit d'e l'affichage d'un nombre réel (float)

\n spécifie un retour à la ligne suite à cet affichage (par retour chariot (backtrack)).

Pour un affichage de plusieurs variables :

```
printf(“ Encadrement : %f\n ”, a ,  “ < x < ”, b) ;
```

Structure de contrôle, boucle et instruction :

Conditionnelle :

```
Si {Condition} Alors
        {Traitement 1}
Sinon
        {Traitement 2}
Fin de Si
```

Ou :

```
Si {Condition 1} Alors

        Si {Condition 2} Alors
                {Traitement 1}
        Sinon
                {Traitement 2}
        Fin de Si
Sinon
        {Traitement 3}
Fin de Si
```

Exemple : Calcul du discriminant d’un polynôme de degré 2.

PS :

```
d prend la valeur b^2 – 4*a*c
Si d > 0 alors
        x1 prend la valeur –(b – racine(d))/(2a)
        x2 prend la valeur –(b + racine(d))/(2a)
        Afficher : “ Deux racines distinctes ”,x1,x2 ;

        SinonSi d = 0 alors
                x1 prend la valeur –(b)/(2a) ;
                Afficher : “ Une racine double ”, x1 ;
Sinon
        Afficher : “ Aucune racine ” ;
Fin de Si.
```

Scilab :

```
d = b^2 – 4 * a * c
if d > 0 then
        x1 = -(b-sqrt(d))/(2*a) ;
        x2 = -(b+sqrt(d))/(2*a) ;
        printf(“ %s\n ”, “ Résultats ”)
        printf(“ Deux racines distinctes : x1 =  %f%s%f\n ”,x1, “ et x2 = ”, x2)
elseif d == 0 then
```

```
            x1 = -(b)/(2*a) ;
            printf(“ %s\n ”, “ Résultats ”)
            printf(“ Une racine double : x1 = %f\n “,x1)
        else
            printf(“ %s\n ”, “ Résultats ”)
            printf(“ %s\n ”, “ Aucune racine ”)
        end
```

Répétitive : Pour

```
Pour k allant de 1 à n Faire //Alternative : Pour k de 1 jusqu’à n Faire
    {Traitement 1}
Fin pour
```

Exemple :

PS :

```
    t prend la valeur 0
    Pour k de 1 jusqu’à n Faire
        x prend la valeur aléatoire strictement comprise entre 0 et 1.
        u prend la valeur aléatoire strictement comprise entre 0 et 1.
        y prend la valeur (1 – x)*u
        z prend la valeur 1 – x – y

        Si ( x < .5 et y < .5 et z < .5) Alors
            t prend la valeur t + 1
        Fin de si
    Fin pour
```

Scilab

```
    t = 0 ;
    For k in 1 :n
        x = rand() ;
        //u = rand() ;
        y = (1-x)*rand() ;//On a factorisé u dans le code
        z = 1 – x – y ;

        If (  x< .5 & y< .5 & z < .5) then
            t = t +1 ;
        end
    end
```

Répétitive : Tant que

```
Tant que {Condition == Vraie} Faire
    {Traitement 1}
    //Mise à jour d’une variable relative à la condition dans le traitement pour éviter de
boucler à l’infinie
Fin Tant que
```

Exemple :

PS :

```
Tant que (b - a) > p faire
        m prend la valeur (b+a)/2

        Si f(a) * f(m)>0 alors
                b prend la valeur m
        Sinon
                a prend la valeur m
        Fin de Si
Fin Tant que
```

Scilab :

```
while (b-a) > p
        m = (b+a)/2 ;

        if f(a)*f(m) > 0 then
                b = m;
        else
                a = m;
        end
end
```

Les fonctions sous Scilab

Input() : saisie d'une variable utilisateur

```
w = input("Entrer votre prénom :","string");
```

```
--> w = input("Entrer votre prénom :","string");
Entrer votre prénom :
Mehdi

--> w
 w  =

 Mehdi
```

```
u = input("Entrer les bornes : [a,b] = ");
```

```
-1-> u = input("Entrer les bornes : [a,b] = ");
Entrer les bornes : [a,b] =
[2,3]

u
 u  =

   2.   3.
```

```
p = input("Entrez le nombre de points p =");
```

```
--> p = input("Entrer le nombre de points : p = ");
Entrer le nombre de points : p =
7

--> p
 p  =

   7.
```

disp() : afficher un résultat dans le console

```
a = 5;
disp(a)
```

```
-1->  a = 5;

-1->  disp(a)

   5.
```

printf() : afficher un résultat dans la console. Plus intéressantes que la fonction disp() du point de vue des fonctionnalités de sortie.

%s%f%\n : chaine de formatage.
%i : affichage d'un nombre entier (integer)
%f : affichage d'un nombre réel (float)
%e : notation exponentielle
%E : notation ingénieur

\n : retour à la ligne après affichage
%0.8f : force l'affichage du réel avec 8 décimales.

```
a = 27/7;
printf("%s%f\n","valeur de a = ",a)
printf("%s%0.2f\n", "valeur à 2 décimales de a = ",a)
printf("%s%0.8f\n", "valeur à 8 décimales de a = ",a)
printf("%s%i\n", "valeur à 2 décimales de a = ",a)
printf("%s%e\n", "valeur au format exponentiel de a = ",a)
printf("%s%E\n", "valeur au format ingénieur de a = ",a)
```

```
--> a = 27/7;

--> printf("%s%f\n","valeur de a = ",a)
valeur de a = 3.857143
```

```
--> printf("%s%0.2f\n", "valeur à 2 décimales de a = ",a)
valeur à 2 décimales de a = 3.86

--> printf("%s%0.8f\n", "valeur à 8 décimales de a = ",a)
valeur à 8 décimales de a = 3.85714286

--> printf("%s%i\n", "valeur à 2 décimales de a = ",a)
valeur à 2 décimales de a = 3

--> printf("%s%e\n", "valeur au format exponentiel de a = ",a)
valeur au format exponentiel de a = 3.857143e+00

--> printf("%s%E\n", "valeur au format ingénieur de a = ",a)
valeur au format ingénieur de a = 3.857143E+00
```

ascii() : convertit une chaîne de caractère en code ascci ou un vecteur de code ascii en chaine de caractères. Par exemple ascii(“ c ”) = 99 et ascii(9) = “ c ”.

```
u = "Mehdi"
disp(u)
v = ascii(u);
disp(v);
w = ascii(v);
disp(w);
```

```
--> u = "Mehdi"
 u  =

 Mehdi


--> disp(u)

 Mehdi


--> v = ascii(u);

--> disp(v);

   77.   101.   104.   100.   105.


--> w = ascii(v);

--> disp(w);

 Mehdi
```

clf() : efface le contenu de la fenêtre graphique courante.
Note : On peut écrire directement dans l'éditeur clf ou clf().
Note : Dans le cadre de plusieurs fenêtres graphiques, clf(n) effacera la fenêtre graphique dont l'identificateur est n. Par défaut, n = 0.

deff() : permet de définir une fonction utilisateur.
deff('[z] = moyenne(x,y)','z=(x+y)/2')
Les paramètres sont des chaines de caractères (que l'on définit à l'aide d'appostrophes " ' " ou de guillemets " ") .
x et y sont les variables d'entrée.
z, la variable de sortie.
Note : cette fonction est utilisée depuis la console Scilab pour définir de nouvelles fonctions (et non depuis l'éditeur). Cependant, elle permet également de définir une fonction au cours d'un programme lorsque l'expression de celle-ci doit être saisie par l'utilisateur.

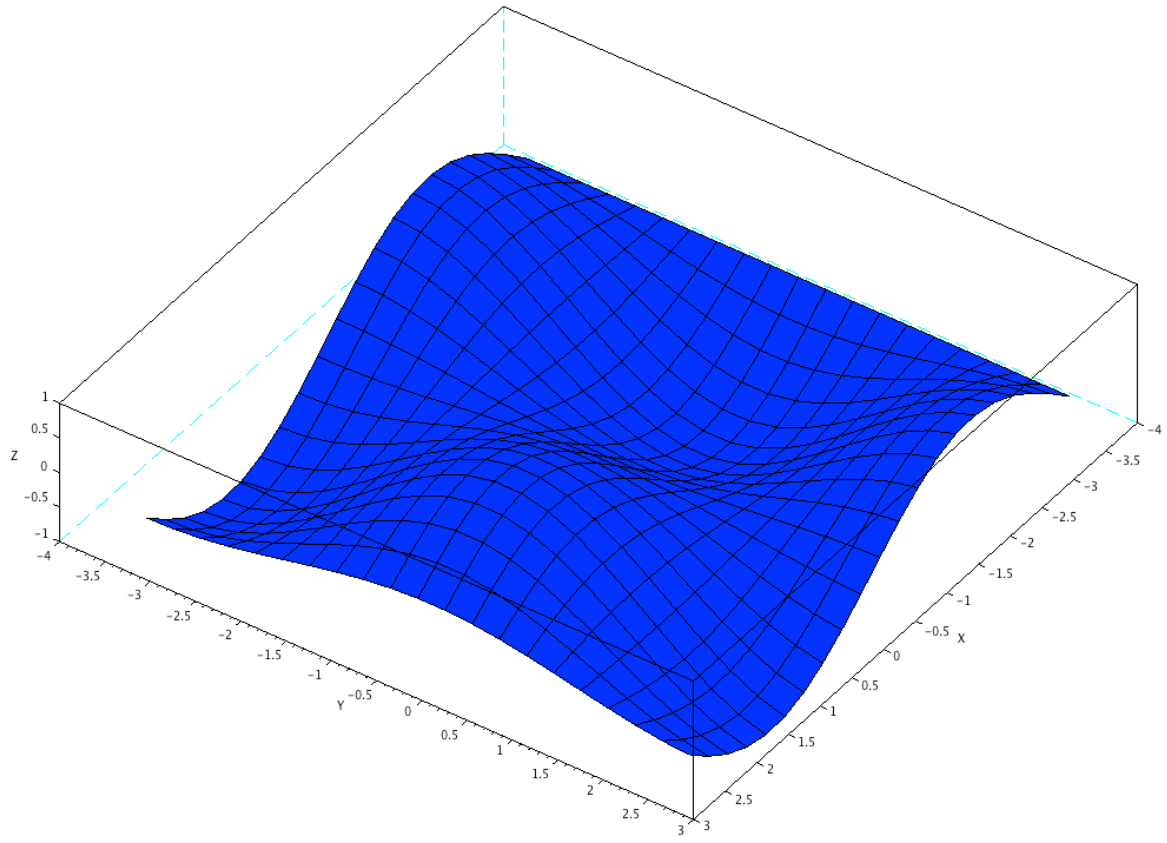
```
deff('[z] = moyenne(x,y)','z=(x+y)/2')
```

```
--> deff('[z] = moyenne(x,y)','z=(x+y)/2')

--> moyenne(6,8)
 ans  =

   7.
```

fplot3d1() : permet de tracer la représentation graphique d'une surface définie par une fonction.

```
deff('z=f(x,y)','z=x^4-y^4')
x = -3:0,2:3;
y=x;
clf();
fplot3d1(x,y,f)
```

```
--> deff('z=f(x,y)','z=x^4-y^4')

--> x = -3:0,2:3;
 x  =

  -3.  -2.  -1.   0.


--> y=x;

--> clf();

--> fplot3d
```



Note : x = -3 :0.2 :3 retourne un vecteur x comportant des nombres compris entre [–3,3] par pas de 0.2 ; c'est à dire : [-3,-2.8, …, 2.8,3].

```
--> x
 x  =

        column 1 to 12

 -3.  -2.8  -2.6  -2.4  -2.2  -2.  -1.8  -1.6  -1.4  -1.2  -1.  -0.8

        column 13 to 24

 -0.6  -0.4  -0.2   0.   0.2   0.4   0.6   0.8   1.   1.2   1.4   1.6

        column 25 to 31

  1.8   2.   2.2   2.4   2.6   2.8   3.

--> length(x)
 ans  =

  31.
```

```
--> max(x)
 ans  =

   3.


--> min(x)
 ans  =

  -3.
```

function() : permet de définit une fonction utilisateur. Sa syntaxe est la suivante :

```
function [arguments_sortie] = nom_fonction(argument_entrée)
      {instructions}
endfunction
```

```
function z=f(x, y)
    z = x^4-y^4
endfunction
disp(f(3,2))
```

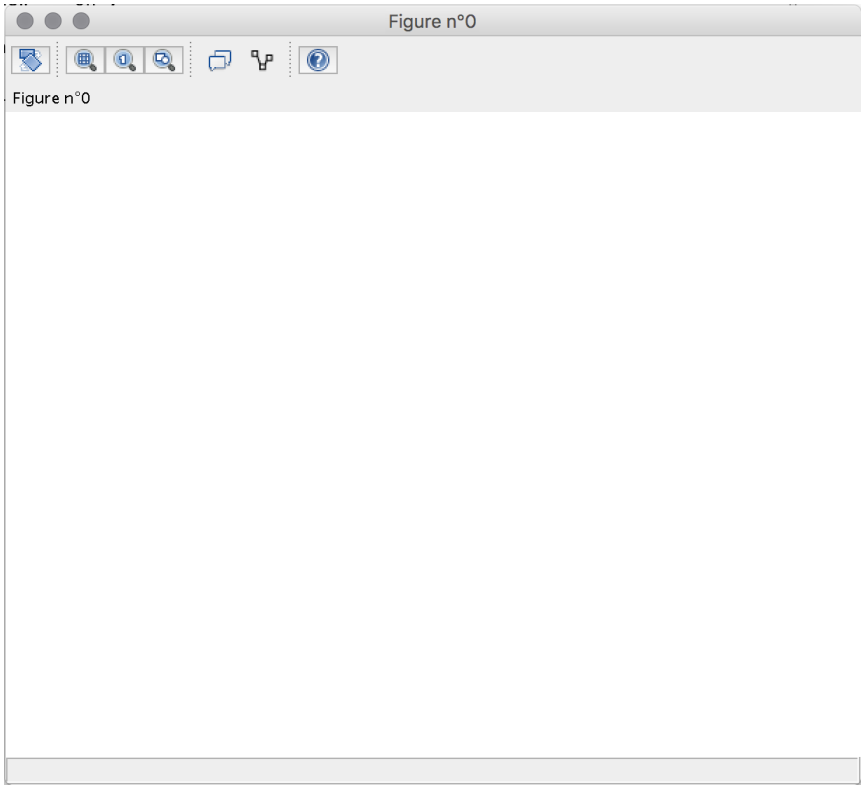
```
--> function z = f(x,y)
  >     z = x^4-y^4
  > endfunction

--> disp(f(3,2))

   65.


--> f(4,5)
 ans  =

  -369.


--> f(6,4)
 ans  =

   1040.
```

gca() et isoview

La fonction gca() retourne l'identificateur de l'axe courant.
L'option isoview permet de définir un repère isométrique (avec les mêmes échelles).

```
a = gca();
a.isoview = "on";
```

```
--> a = gca();

--> a.isoview = "on";
```



int() : Retourne la troncature à l'unité d'un nombre.

```
--> int(5.7)
 ans  =

   5.


--> int(77.0)
 ans  =

   77.


--> int(2)
 ans  =

   2.


--> int(-4)
 ans  =
```

```
 -4.
```

floor() : Retourne la partie entière d'un nombre.

```
--> floor(2)
 ans  =

   2.


--> floor(3.45)
 ans  =

   3.


--> floor(9.99)
 ans  =

   9.


--> floor(-4.55)
 ans  =

  -5.
```

length() : renvoie la taille d'un vecteur

```
--> x = [2,3,5,6]
 x  =

   2.   3.   5.   6.


--> length(x)
 ans  =

   4.
```

linspace() : permet de créer un vecteur de valeurs équidistantes. Sa syntaxe est la suivante : v = linspace(valeur_initiale, valeur_finale, nb_de_valeurs)

```
--> v = linspace(1,5,8)
 v  =
```

```
        column 1 to 6

   1.   1.5714286   2.1428571   2.7142857   3.2857143   3.8571429

        column 7 to 8

   4.4285714   5.


--> v = linspace(1,8,5)
 v  =

   1.   2.75   4.5   6.25   8.


--> v = linspace(-1,-8,5)
 v  =

  -1.  -2.75  -4.5  -6.25  -8.
```

modulo() : Considérons deux entiers naturels a et b. La fonction modulo(a,b) permet de calculer le reste de la division euclidienne de a par b

```
--> modulo(7,22)
 ans  =

   7.


--> modulo(22,7)
 ans  =

   1.


--> modulo(19,5)
 ans  =

   4.
```

Note :

$7 = 7 + 0 \times 22$

$22 = 1 + 3 \times 7$

$19 = 4 + 3 \times 5$

ones() : permet de générer une matrice dont les coefficients sont tous égaux à 1

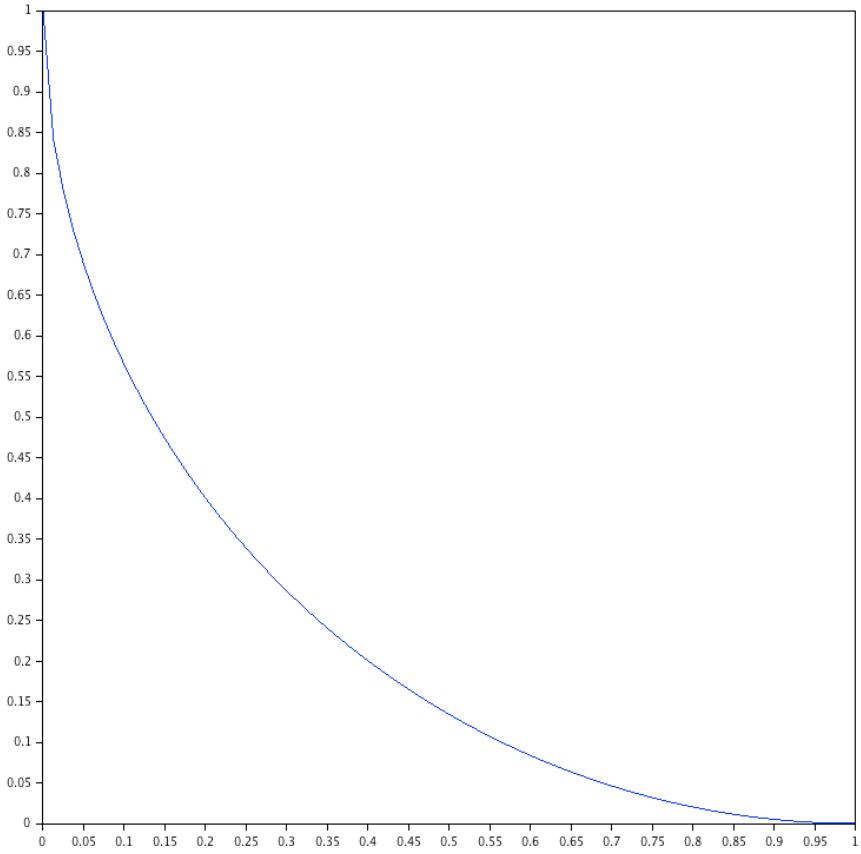
```
--> x = linspace(1,4,4)
 x  =
```

```
  1.   2.   3.   4.

//Variante du livre : x = [1,2,3,4]

--> x
 x  =

  1.   2.   3.   4.


--> y = ones(x)
 y  =

  1.   1.   1.   1.
```

plot() : permet de tracer la courbe représentative d'une fonction donnée ou de représenter un ensemble de points.

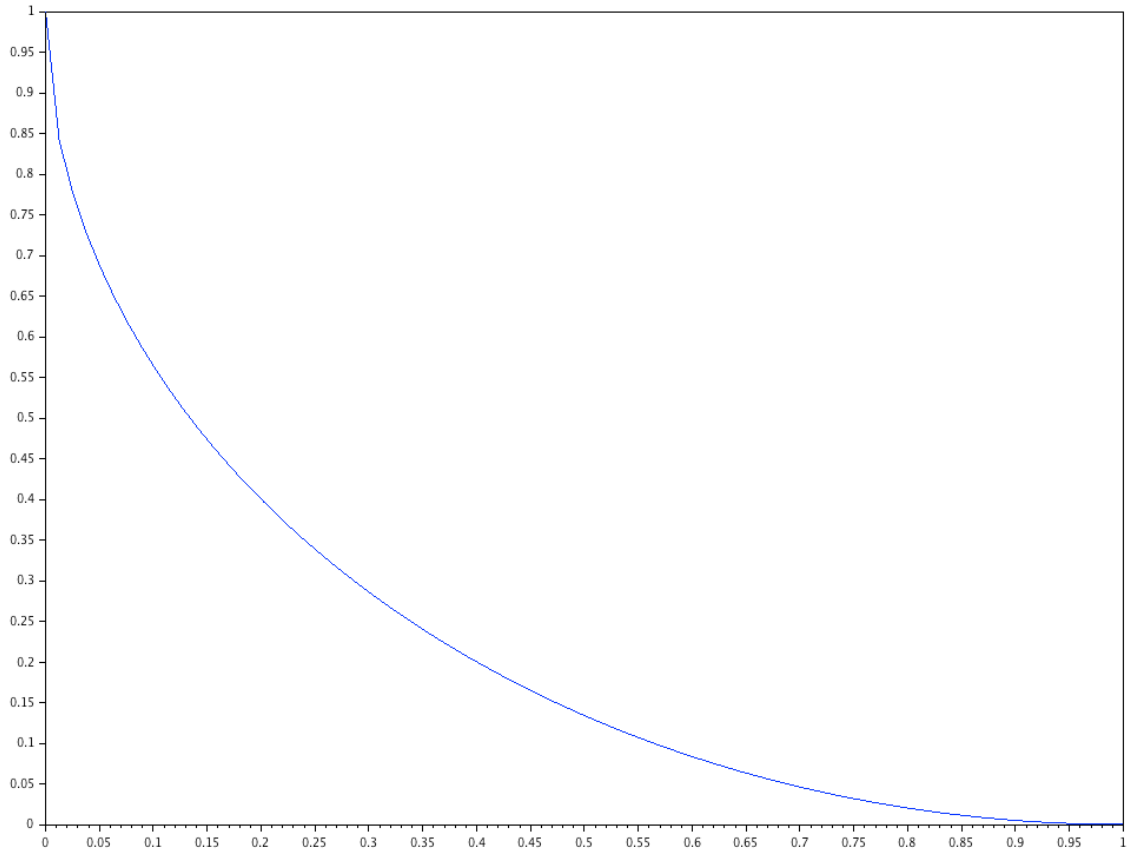
```
deff('[y]=g(x)','y=1-sqrt(1-(x-1)^2)');
x = linspace(0,1,80);
clf()
a = gca();
a.isoview = "on";
plot(x,g,"b")
```

```
--> deff('[y]=g(x)','y=1-sqrt(1-(x-1)^2)');

--> x = linspace(0,1,80);

--> clf()

--> a = gca();

--> a.isoview = "on";

--> plot(x,g,"b")
```




```
deff('[y]=g(x)','y=1-sqrt(1-(x-1)^2)');
x = linspace(0,1,80);
clf()
plot(x,g,"b")
```

```
--> deff('[y]=g(x)','y=1-sqrt(1-(x-1)^2)');

--> x = linspace(0,1,80);

--> clf()

--> plot(x,g,"b")
```

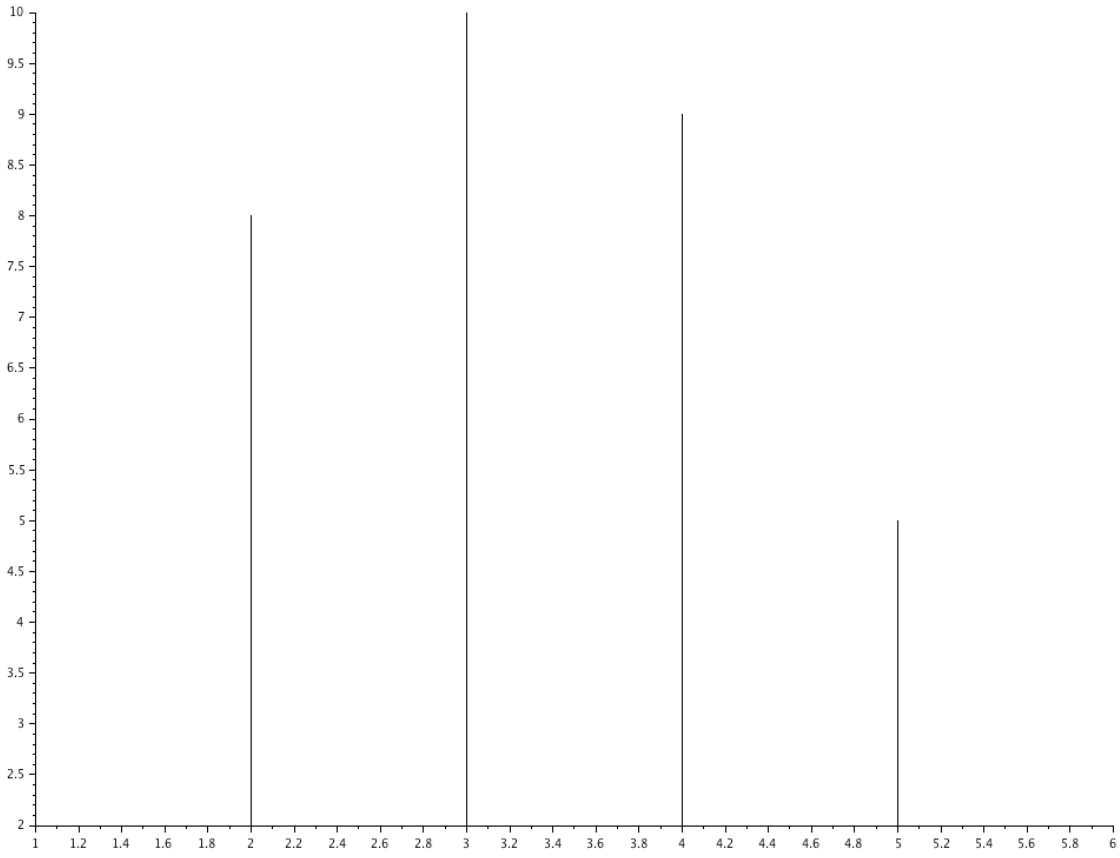


plot2d3() : permet de tracer des diagrammes en bâtons d'un ensemble de données.
plot2d3(x,y) avec x, un vecteur correspondant aux valeurs d'abscisses et y, un vecteur correspondant aux valeurs de coordonnées

```
clf()
//x = linspace(1,6,6);
x = [1,2,3,4,5,6]
y = [2,8,10,9,5,2];
plot2d3(x,y)
```

```
--> clf()

--> //x = linspace(1,6,6);

--> x = [1,2,3,4,5,6]
 x  =

   1.   2.   3.   4.   5.   6.
```

```
--> y = [2,8,10,9,5,2];

--> plot2d3(x,y)
```

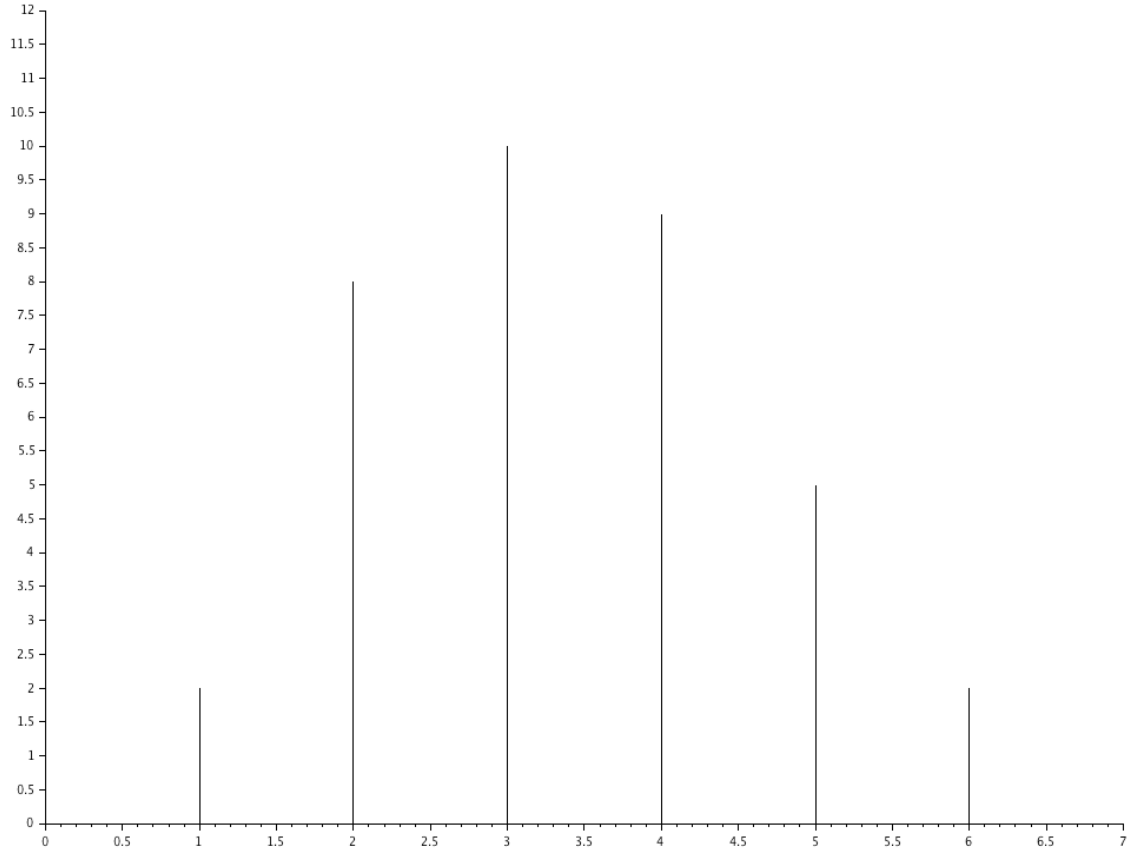


Note : rect = [xmin,ymin,xmax,ymax] de la fonction permet de limiter le tracer de la zone comprise dans le rectangle ABCD défini par les coordonnées A(xmin,ymin) et C(xmax,ymax).

```
clf()
//x = linspace(1,6,6);
x = [1,2,3,4,5,6]
y = [2,8,10,9,5,2];
plot2d3(x,y,rect = [0,0,7,12])
```

```
--> clf()
```

```
--> //x = linspace(1,6,6);

--> x = [1,2,3,4,5,6]
 x  =

   1.   2.   3.   4.   5.   6.


--> y = [2,8,10,9,5,2];

--> plot2d3(x,y,rect = [0,0,7,12])
```



rand() : permet de générer un nombre aléatoire strictement compris entre 0 et 1. Par défaut, la loi sélectionnée pour la distribution des valeurs est la loi uniforme.

```
--> rand()
 ans  =
```

```
  0.2113249


--> rand()
 ans  =

  0.7560439
```

strcat() : permet de concaténer un vecteur de chaines de caractères.

```
w(1) = "Bon";
w(2) = "jour";
w(3) = " Monsieur ";
w(4) = "Mehdi";
r = strcat(w);
disp(r)
```

```
--> w(1) = "Bon";

--> w(2) = "jour";

--> w(3) = " Monsieur ";

--> w(4) = "Mehdi";

--> r = strcat(w);

--> disp(r)

 Bonjour Monsieur Mehdi
```

strsplit() : permet de transformer une chaine de caractères de n éléments en un vecteur de même dimension (ou une matrice). Il faut voir l'aide en ligne.

```
w = "Bonjour !";
r = strsplit(w,[1:length(w)-1]);
disp(r)
```

```
--> w = "Bonjour !";

--> r = strsplit(w,[1:length(w)-1]);

--> disp(r)

!B  !
!   !
```

```
!o  !
!   !
!n  !
!   !
!j  !
!   !
!o  !
!   !
!u  !
!   !
!r  !
!   !
!   !
!   !
!!  !


--> r(1)
 ans  =

 B


--> r(2)
 ans  =

 o


--> r(3)
 ans  =

 n
```

scf() : crée un nouvel objet Scilab de type figure. Cette fonction permet de d'afficher des représentations graphiques dans des fenêtres graphiques différentes.

sum() :
Pour un vecteur, la fonction effectue la somme des éléments du vecteur.
Pour une matrice, il faut voir l'aide en ligne.

```
--> v = [1,2,4,5]
 v  =

   1.   2.   4.   5.


--> sum(v)
 ans  =
```

12.

xset() ; permet de changer les paramètres du contexte graphique.
Exemple : xset('colormap',hotcolormap(128)).
Cette commande permet de définir la table des couleurs hotcolormap (à savoir un dégradé de couleurs allant du rouge foncé au jaune clair).

zeros() : permet de définir une matrice nulle.
Exemple : zeros(1,3) définit un vecteur de dimension 3.

```
--> zeros(1,3)
 ans  =

   0.   0.   0.


--> zeros(3,3)
 ans  =

   0.   0.   0.
   0.   0.   0.
   0.   0.   0.
```

Un souci :

On y trouve l'aide de Scilab

Et sinon des tutos :

http://www.scilab.org/resources/documentation/tutorials