Informatique et Algorithmique – Chapitre 4

Représentation graphique en Scilab

Une remarque pour commencer : lorsque l'on demande à Scilab de tracer des figures, elles s'affichent toutes par défaut sur une seule et même fenêtre graphique. La commande `clf()` efface le contenu de la fenêtre graphique active (et en ouvre une si aucune fenêtre graphique n'est déjà ouverte).

I Représentation de points dans le plan

1) Avec la fonction `plot`

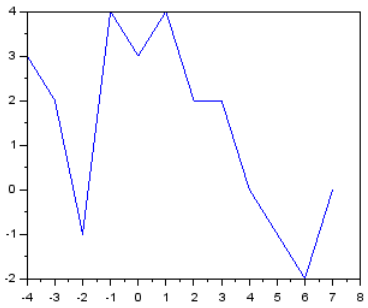
Soient `x` et `y` deux vecteurs de même taille $n \geqslant 2$. Pour tout $i \in [\![1, n]\!]$, notons A_i le point d'abscisse `x(i)` et d'ordonnée `y(i)`. La commande `plot(x,y)` trace les segments $[A_1, A_2], \ldots, [A_{n-1}, A_n]$.

Exemple :

```
-->X=[-4,-3,-2,-1,0,1,2,3,4,5,6,7];

-->Y=[3,2,-1,4,3,4,2,2,0,-1,-2,0];

-->clf(); plot(X,Y)
```



Personnalisation des tracés : On peut ajouter des options à cette commande afin de modifier la couleur, le marquage des points et la forme du trait qui relie les points. La commande est `plot(x,y,'opt')`, où `opt` est composée d'un ou plusieurs des éléments suivants :

- une lettre pour indiquer la couleur parmi

`r`	rouge	`g`	vert	`b`	bleu	`c`	cyan
`m`	magenta	`y`	jaune	`k`	noir	`w`	blanc

- un symbole pour indiquer le marqueur des points parmi (entre autres)

`+`	plus	`x`	croix	`o`	cercle	`.`	point	`*`	astérisque

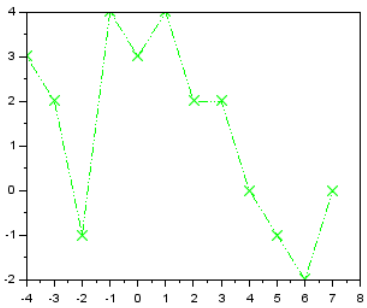
- un symbole pour indiquer la forme du trait qui relie les points parmi

`-`	ligne continue	`--`	pointillés (du type − − − − −)
`-.`	pointillés (du type − · − · −·)	`:`	pointillés (du type − · · − · · − · ·)

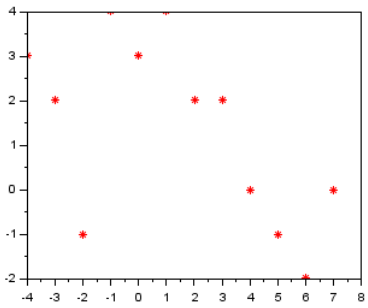
Si on précise le marqueur des points mais pas la forme du trait, alors Scilab trace la figure sans relier les points.

Exemples :

```
-->clf(); plot(X,Y,'gx:')
```



```
-->clf(); plot(X,Y,'r*')
```



Ces options sont d'autant plus pratiques pour différencier des tracés superposés.

Superposition de tracés : Puisque, par défaut, Scilab trace toutes les figures sur une seule fenêtre graphique, il suffit de faire appel à la fonction `plot` consécutivement pour superposer des tracés. Cependant cette méthode peut entraîner une déformation des courbes puisque l'échelle est systématique recalculée pour chaque courbe. On préférera donc la commande

`plot(x1,y1,'opt1',...,xn,yn,'optn')`

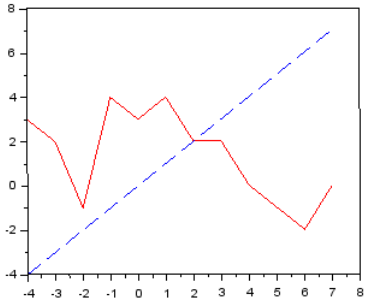
où `x1,y1` sont les vecteurs d'abscisses et d'ordonnées et `opt1` les options pour le premier tracé,... et `xn,yn` sont les vecteurs d'abscisses et d'ordonnées et `optn` les options pour le $n^{\text{ième}}$ tracé (les options étant facultatives).

Si on désire tracer plusieurs courbes ayant le même vecteur d'abscisses `x`, et des vecteurs d'ordonnées respectives `y1,...,yn` (tous nécessairement de même taille que `x`), alors on peut utiliser la syntaxe [1] `plot(x1,[y1;...;yn])`. Naturellement Scilab proposera différentes couleurs pour les tracés successifs. Avec cette syntaxe, on ne peut pas préciser d'options différente pour chaque tracé.

Exemples :

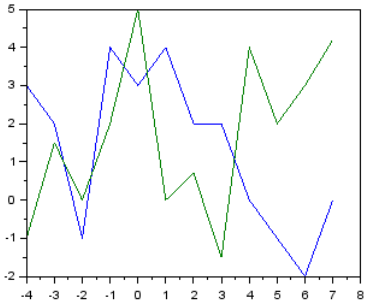
```
-->S=[-4,7];

--> clf(); plot(X,Y,'r',S,S,'b--')
```



```
-->Z=[-1,1.5,0,2,5,0,0.7,-1.5,4,2,3,4.2];

-->clf(); plot(X,[Y;Z])
```



2) Avec la fonction `plot2d`

Soient `x` et `y` deux vecteurs de même taille $n \geqslant 2$. Pour tout $i \in [\![1, n]\!]$, notons A_i le point d'abscisse `x(i)` et d'ordonnée `y(i)`. La commande `plot2d(x,y)` trace les segments $[A_1, A_2], \ldots, [A_{n-1}, A_n]$.

A première vue, la fonction `plot2d` semble identique à la fonction `plot`. Elles fonctionnent essentiellement de la même manière mais avec des différences notables. Notamment `plot2d` est plus élaborée et on peut davantage la personnaliser [2] (on peut lui ajouter plus d'options).

Personnalisation des tracés : Pour ajouter des options, la syntaxe est `plot2d(x,y,options)`. Les principales options sont :

- `style=k` où k est un entier entre -14 et 33 qui détermine le style de tracé. Plus précisément :
  - Si $k \in [\![-14, 0]\!]$, les points ne sont pas reliés et le marqueur des points est modifié. Citons par exemple :

0	point	−1	plus	−2	croix	−9	cercle	−10	astérique

  - Si $k \in [\![1, 33]\!]$, c'est la couleur qui est modifié. Citons par exemple :

1	noir	2	bleu	3	vert clair	4	cyan
5	rouge	6	magenta	13	vert foncé	7	jaune

    La commande `getcolor()` affiche un tableau avec la liste des couleurs.

  Si on superpose `n` tracés avec des indices de styles respectifs `k1,...,kn`, alors on utilise la commande `style=[k1,...,kn]`.
- `rect=[xmin,ymin,xmax,ymax]` décrit la zone du graphique visible (il s'agit d'un rectangle dont les coordonnées de l'angle inférieur gauche est `(xmin,ymin)` et celles de l'angle supérieur droit est `(xmax,ymax)`).

1. En Scilab, [y1 ;... ;yn] est une matrice dont les lignes sont `y1,...,yn`. Nous y reviendrons.
2. Mais par exemple on ne peut pas faire un tracé en pointillé de façon simple...

- Si on superpose `n` tracés, la commande `leg="legende_1@...@legende_n"` ajoute des légendes aux différents tracés. Nous y reviendrons ultérieurement dans ce chapitre.

Superposition des tracés : Pour tracer plusieurs courbes ayant le même vecteur d'abscisses `x`, et des vecteurs d'ordonnées respectifs `y1`,...,`yn` (tous nécessairement de même taille que `x`), alors on utilise la syntaxe

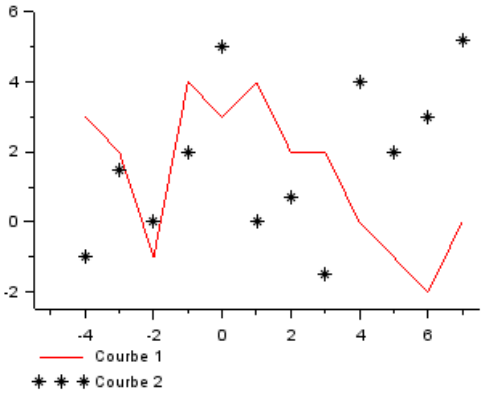
```
plot(x',[y1',...,yn'],options)
```


La fonction `plot2d` fonctionne avec des vecteurs colonnes et non des vecteurs lignes. Ainsi, si on dispose de vecteurs lignes (ce qui est le cas en général), il ne faut pas oublier de mettre les ' qui permettent de les transposer (c'est-à-dire les transformer en vecteurs colonnes).

Exemple :

```
-->clf();

-->plot2d(X',[Y',Z'],style=[5,-10],
rect=[-5.5,-2.5,7.5,6],
leg='Courbe 1@Courbe 2')
```



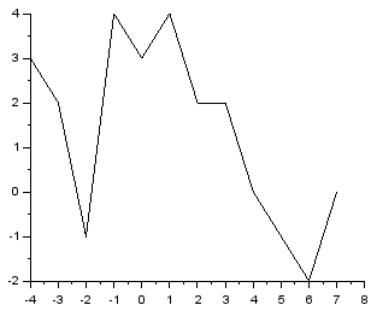
3) Avec les fonctions `plot2d2` et `plot2d3`

Il s'agit de variantes de `plot2d` qui fonctionnent de la même façon. Elles diffèrent dans le type de tracé :

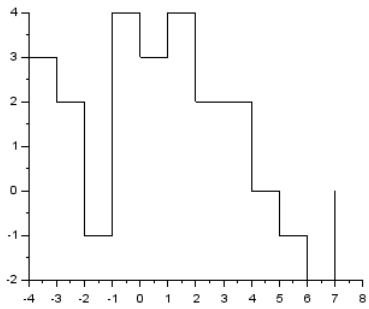
- `plot2d2` trace des fonctions constantes par morceaux à gauche (c'est-à-dire elle relie les points par un segment horizontal suivi d'un segment vertical).
- `plot2d3` trace des barres verticales.

Exemples :

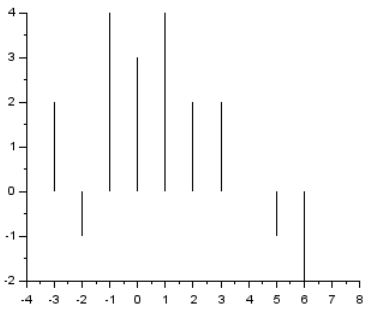
```
-->clf(); plot2d(X,Y)
```



```
-->clf(); plot2d2(X,Y)
```



```
-->clf(); plot2d3(X,Y)
```



II Tracé de courbes représentatives de fonctions

Tracer la courbe représentative d'une fonction sur un intervalle, revient à tracer une infinité de points. Or il n'est évidemment pas possible pour un ordinateur de réaliser une infinité d'instructions. L'approche de Scilab pour tracer une courbe est de représenter un nombre fini de points et de les relier par une ligne continue. La puissance de calcul des ordinateurs permet de considérer un très grand nombre de points et d'obtenir ainsi une très bonne approximation de la courbe par une ligne brisée dont les abscisses sont très proches afin de donner l'illusion de courbe.

Tracé avec `plot` ou `plot2d` : Supposons que l'on ait codé une fonction f en Scilab dans un script nommé `f.sci`. Soient a et b des réels tels que $a < b$. Pour tracer la courbe représentative de f sur l'intervalle $[a, b]$, on peut utiliser l'une des commandes suivantes :

```
x=linspace(a,b,n)
plot(x,f(x),'opt')
```

```
x=linspace(a,b,n)
plot(x,f,'opt')
```

```
x=linspace(a,b,n)
plot2d(x,f(x),options)
```

```
x=linspace(a,b,n)
fplot2d(x,f,options)
```

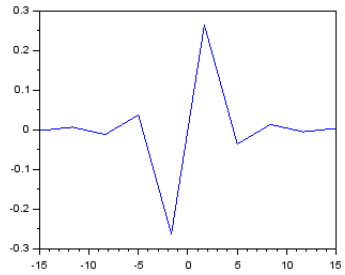
selon que l'on désire utiliser `plot` ou `plot2d`. L'entier `n` détermine le nombre de points utilisés pour effectuer le tracé. Sachant que deux points sont reliés par une ligne continue, il faut prendre `n` grand pour que la courbe apparaisse suffisamment lisse.

Remarques :

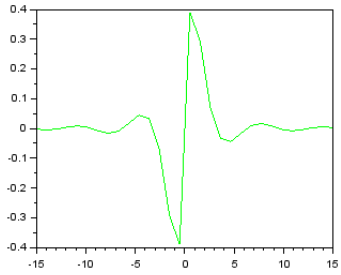
- On peut remplacer `linspace(a,b,n)` par `a:1/n:b`. Ces deux commandes ne fournissent pas le même vecteur mais, si `n` est grand, la différence n'est pas visible lors du tracé de la courbe par Scilab.
- Les commandes à retenir principalement sont `plot(x,f(x))` et `plot2d(x,f(x))`. Elles permettent de faire plus facilement le lien avec la méthode de tracé (les courbes sont approchés par des courbes de fonctions affines par morceaux). Ces commandes sont usuelles et elles ont des analogues dans de nombreux autres langages de programmation.
- Les commandes `plot(x,f)` et `plot2d(x,f)` sont moins universelles mais elles sont très pratiques. En effet, si on les utilise, alors on est pas obligé de rendre compatibles les fonctions Scilab avec les opérations vectorielles (pas besoin de `.*`, `./`, `.^`, ni de boucle `for` si la fonction présente des structures conditionnelles).

Exemple 1 : Représentons la fonction f définie au chapitre précédent (et codée dans le script `f.sci`).

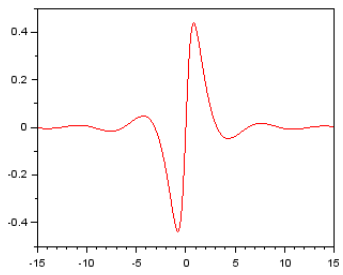
```
-->x=linspace(-15,15,10);
-->y=f(x);
-->clf(); plot(x,y,'b')
```



```
-->x=linspace(-15,15,30);
-->y=f(x);
-->clf(); plot(x,y,'g')
```

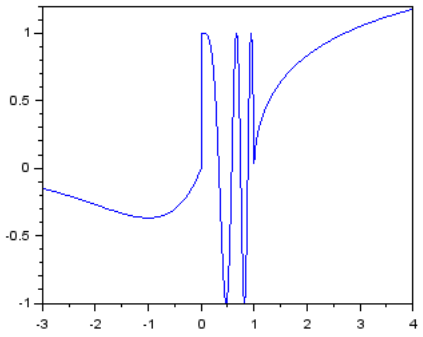


```
-->x=linspace(-15,15,1000);
-->y=f(x);
-->clf(); plot(x,y,'r')
```

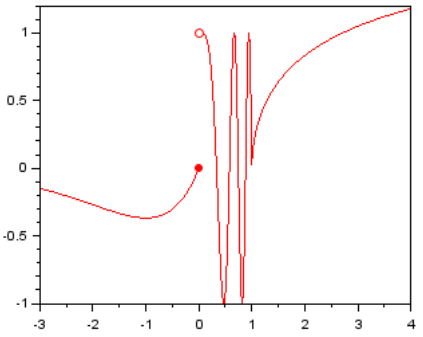


Exemple 2 : Représentons la fonction g définie au chapitre précédent (et codée dans le script `g.sci`). On pourra modifier le tracé pour prendre en compte la discontinuité en 0.

```
-->exec('f.sci', -1)
-->x=linspace(-3,4,1000); y=g(x);
-->clf(); plot(x,y)
```



```
-->x1=linspace(-3,0,1000); y1=g(x1);
-->x2=linspace(0.01,4,1000); y2=g(x2);
-->clf(); plot(x1,y1,'r',x2,y2,'r',
                [0],[0],'r.',[0],[1],'ro')
```



Superposition de courbes : Si on désire superposer les courbes représentatives sur $[a,b]$ de plusieurs fonctions codées en Scilab par les scripts `f1.sci`,..., `fk.sci`, alors généralement on se donne un vecteur d'abscisses commun `x=linspace(a,b,n)` puis on utilise l'une des commandes suivantes :

- `plot(x,[f1(x);...;fk(x)])`
- `plot(x,f1(x),'opt1',...,x,fk(x),'optk')`
- `plot(x,f1,'opt1',...,x,fk,'optk')`
- `plot2d(x',[f1(x)',...,fk(x)'],options)`

Exemple :

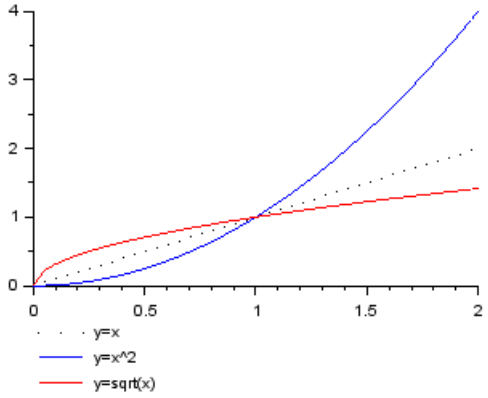
```
-->x=linspace(0,5,100);

-->y=x.^2; z=sqrt(x);

-->clf();

-->legende='y=x@y=x^2@y=sqrt(x)';

-->plot2d(x',[x',y',z'],style=[0,2,5],
            rect=[0,0,2,4],leg=legende);
```



III Diagrammes à barres et histogrammes

1) Diagrammes à barres

Un diagramme à barres est un graphique permettant de représenter la répartition de données avec des barres verticales. Plus précisément :

- chaque barre a la même largeur et son abscisse représente une catégorie,
- la hauteur d'une barre est égale à l'effectif (ou la fréquence) de la catégorie qu'elle représente.

Si `x` est un vecteur contenant des abscisses des catégories et si `y` est un vecteur contenant les effectifs respectifs de ces catégorie, alors la syntaxe `bar(x,y)` permet de construire le diagramme à barre correspondant. Comme pour `plot`, on peut lui ajouter des options.

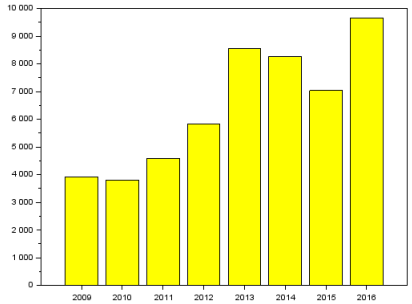
Si on dispose de vecteurs `y1`,...,`yk` représentant des données différentes mais relatives aux mêmes catégories (dont les abscisses respectives sont contenues dans un vecteur `x`), alors on peut superposer les `k` diagrammes à barres en utilisant la syntaxe `bar(x,[y1',y2',...,yk'])`. L'ajout de l'option `'stacked'` permettant d'empiler les diagrammes.

Exemple : Le patron d'une entreprise crée en 2009 a compilé son nombre de ventes par année dans le tableau suivant :

Année	2009	2010	2011	2012	2013	2014	2015	2016
Nb de ventes	3923	3805	4578	5814	8547	8260	7033	9665

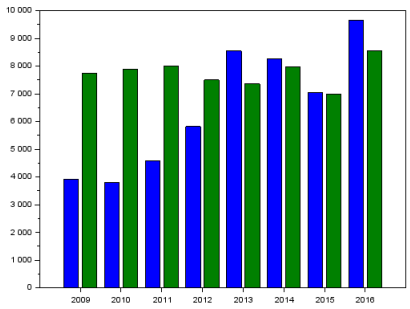
Il trace un diagramme à barre à partir de ses données (les catégories étant les années) :

```
-->vente2=[3923,3805,4578,5814,
           8547,8260,7033,9665];
-->clf(); bar(an,vente,'y')
```

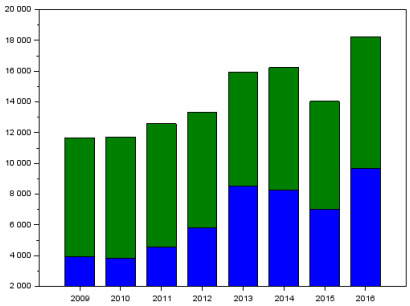


Il dispose également des effectifs moyens des entreprises de la même branche et souhaite les comparer graphiquement avec ses propres effectifs.

```
-->-->vente2=[7746,7892,8003,7502,
              7361,7963,6995,8561];
-->clf(); bar(an,[vente',vente2'])
```




```
-->clf(); bar(an,[vente',vente2'],'stacked')
```



2) Histogrammes

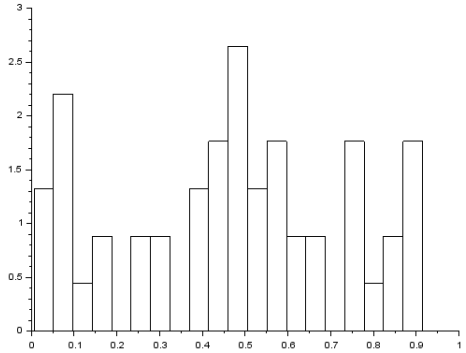
Un histogramme est un graphique permettant de représenter la répartition de données avec des colonnes verticales. Plus précisément, chaque colonne représente le nombre (ou la fréquence) de termes appartenant à une classe ou une catégorie.

Si les données sont des réels, alors chaque classe correspond à un intervalle de $\mathbb{R}$ et on représente alors cette classe par un rectangle vertical dont l'aire est proportionnel à son effectif (et donc à sa fréquence).

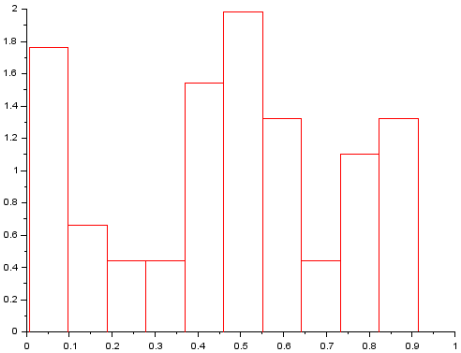
Si `x` est un vecteur contenant des données réelles et si `n` est un entier naturel strictement positif, alors la syntaxe `histplot(n,x)` permet de construire l'histogramme des données du vecteur `x` avec `n` classes de même amplitude. Comme pour `plot2d`, on peut lui ajouter des options.

Exemples :

```
-->x=rand(1:50);
-->clf(); histplot(20,x)
```




```
-->clf(); histplot(10,x,style=5)
```

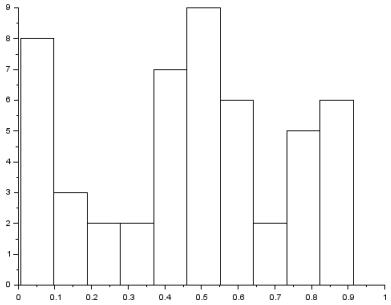


 Par défaut, Scilab construit un histogramme normalisé (c'est-à-dire de telle sorte que la somme des aires des rectangles soit égale à 1). Nous verrons dans les TP de probabilités que c'est très pratique. Néanmoins avec un tel histogramme, on ne peut pas lire graphiquement l'effectif par classe. Pour construire un histogramme non renormalisée on rajoutera l'option `normalization=%f`.

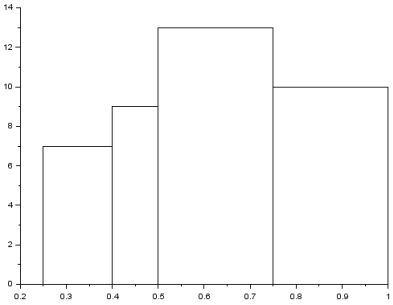
On peut également paramétrer l'amplitude de chaque classe : si `a1,a2,a3,...,ak` sont des réels distincts rangés dans un ordre croissant, alors `histplot([a1,a2,a3,...,ak],x)` construit l'histogramme des données du vecteur `x` avec `k-1` classes d'amplitudes respectives `a2-a1`, `a3-a2`, etc.

Exemples :

```
-->clf();
-->histplot(10,x,normalization=%f)
```



```
-->c=[0.25,0.4,0.5,0.75,1]
-->clf();
-->histplot(c,x,normalization=%f)
```



IV Personnalisation des figures et gestion des fenêtres graphiques

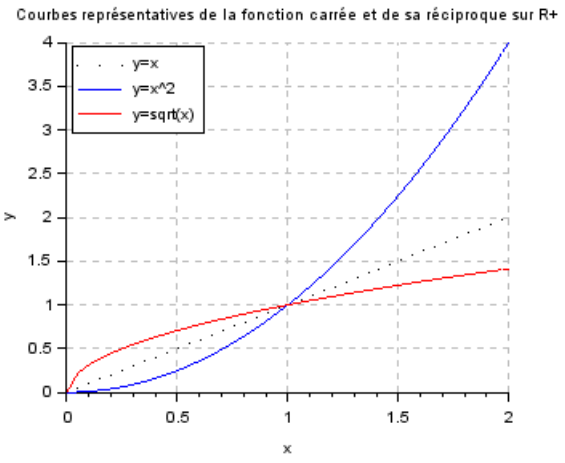
1) Personnalisation des figures

Sans annotations, une figure n'a pas de sens sortie de son contexte. De plus des légendes sont indispensables lorsque l'on superpose plusieurs tracés sur une seule figure. Le tableau suivant décrit plusieurs commandes permettant d'ajouter des titres, des légendes, d'annoter les axes, de gérer les échelles, etc.

`square(xmin,ymin,xmax,ymax)`	la zone du graphique visible devient le rectangle dont les coordonnées de l'angle inférieur gauche est `(xmin,ymin)` et celles de l'angle supérieur droit est `(xmax,ymax)`
`xgrid(k)`	ajoute un quadrillage (dont le numéro de couleur est `k`)
`title('titre_graphique')`	ajoute le titre `titre_graphique` au graphique
`xlabel('titre_abs')`	nomme `titre_abs` l'axe des abscisses
`ylabel('titre_ord')`	nomme `titre_abs` l'axe des ordonnées
`legend('leg1',...,'legn',pos)`	ajoute la légende `leg1` pour la première courbe, ... et la légende `legn` pour la $n^{\text{ième}}$ courbe. L'option `pos` est un entier qui vaut 1 (resp. 2, 3, 4, ...) pour placer la légende dans le coin supérieur droit (resp. supérieur gauche, inférieur gauche, inférieur droit, ...).
`xstring(x,y,'texte')`	ajoute `texte` à la position `(x,y)`.

Exemple :

```
-->clf();
-->plot2d(x',[x',y',z'],style=[0,2,5])
-->square(0,0,2,4)
-->xgrid(33)//grille de couleur grise
-->title('Courbes représentatives
de la fonction carrée et de sa
réciproque sur R+')
-->legend('y=x','y=x^2','y=sqrt(x)',2)
-->xlabel('x'); ylabel('y')
```



2) Gestion des fenêtres graphiques

Commençons par quelques commandes permettant de gérer les différentes fenêtres graphiques.

`clf()`	efface le contenu de toutes les fenêtres graphiques (elle ouvre une si aucune n'existe)
`clf(n)`	efface le contenu de la fenêtre graphique portant le numéro `n`

`scf()`	ouvre une nouvelle fenêtre graphique qui devient la fenêtre active (celle où les figures seront tracées)
`scf(n)`	rend active la fenêtre portant le numéro `n` (si elle n'existe pas, elle crée une fenêtre graphique et lui attribue le numéro `n`)

On peut également afficher plusieurs figures sur une même fenêtre graphique « côte à côte ». Plus précisément, la commande `subplot(n,p,k)` découpe la même fenêtre graphique active en un tableau de $n \times p$ sous fenêtres et sélectionne la $k^{\text{ième}}$ sous fenêtre (qui se trouve à la $i^{\text{ième}}$ ligne et la $j^{\text{ième}}$ colonne telles que $k = n(i-1)+j$).

Exemple : Traçons quelques courbes représentatives de fonctions usuelles.

```
clf();

subplot(2,3,1)
x=linspace(-4,4,1000)
plot2d(x,abs(x),style=2)
legend('y=|x|',4)

subplot(2,3,2)
x=linspace(-4,4,1000)
plot2d(x,floor(x),style=1)
legend('y=[x]',2)

subplot(2,3,3)
x1=linspace(-5,-0.1,1000)
x2=linspace(0.1,5,1000)
plot(x1,x1.^(-1),'g',x2,x2.^(-1),'g')
legend('y=1/x',4)

subplot(2,3,4)
x1=linspace(-4.5,2,1000)
x2=linspace(0.01,7,1000)
plot(x1,exp,'c',x2,log,'m')
legend('y=exp(x)','y=log(x)',2)
square(-4.5,-4.5,7,7)
```

```
subplot(2,3,5)
x=linspace(-10,10,1000)
plot2d(x,[sin(x)',cos(x)'],
                  style=[13,32])
legend('y=sin(x)','y=cos(x)',2)

subplot(2,3,6)
x1=linspace(-1.32,1.32,1000)
x2=linspace(-4,4,1000)
plot(x1,tan,'r',x2,atan,'b')
legend('y=tan(x)','y=Arctan(x)',2)

//Ajout d'une grille grise à chaque
//graphique et annotation des axes
for k=1:6
    subplot(2,3,k)
    xlabel('x')
    ylabel('y')
    xgrid(33)
end
```