Révisions : Outils graphiques de Scilab

Table des matières

1 Représentation graphique d'une fonction **2**
1.1 Définir une fonction 2
1.2 Courbes du plan 2
1.3 Surfaces (ou nappes) de l'espace 4

2 Représentations graphiques de données **5**
2.1 La fonction tabul 5
2.2 Diagramme en bâtons : la commande bar 5
2.3 Diagramme circulaire : la commande pie 6
2.4 Histogramme : la commande histplot 7

Le logiciel *Scilab* possède de nombreuses de commandes permettant de tracer des courbes. On a ainsi la possibilité de visualiser immédiatement et simplement des résultats numériques. On donne dans ce chapitre un aperçu élémentaire de quelques commandes graphiques.

1 Représentations graphiques d'une fonction

1.1 Définir une fonction

Proposition 1.1 : *Définir une fonction*

L'instruction

```
--> function z=g(x), ..., endfunction
```

permet de créer la fonction nommée `g` qui à chaque réel x associe le réel z décrit dans ... par `z=...`.

La commande suivante permet de définir f comme étant la fonction $x \mapsto \frac{1}{1+e^x}$.

```
--> function z=f(x); z=1/(1+exp(x)); endfunction
```

On peut alors demander à Scilab de donner, par exemple, la valeur de $f(0)$.

```
--> f(0)
 ans  =

    0.5
```

On peut également créer des fonctions à deux variables.

```
--> function d=dollars(e,t); d=e*t; endfunction
--> dollars(200,1.1)
 ans  =

    220.
```

1.2 Courbes du plan

On va voir deux façons principales de tracer la courbe représentative d'une fonction, soit sans définir la fonction, soit en définissant la fonction.

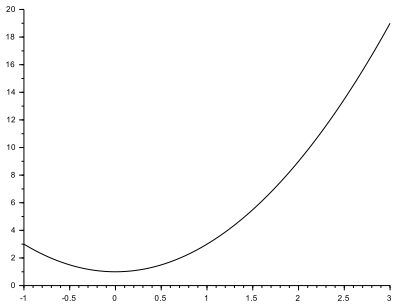
La commande `plot` est la plus simple et la moins paramétrable des fonctions graphiques. Sa syntaxe est de la forme :

Proposition 1.2 : *Tracer un graphique facilement*

On construit le vecteur x des abscisses, on définit le vecteur $y = f(x)$, puis on utilise la commande `plot(x,y)`.

```
--> x=-1:0.01:3; y=2*x.^2+1; plot(x,y)
```

Cette commande permet de tracer la parabole d'équation $y = 2x^2 + 1$ sur un rectangle dont les abscisses sont comprises entre -1 et 3. On a pris un pas pour le vecteur x assez petit (0.01 est assez petit) pour que la ligne brisée représentée soit suffisamment proche de la courbe espérée. Dans ce dernier exemple, vous pouvez changer le pas pour un pas plus grand (par exemple 0.5 au lieu de 0.01), vous observerez une nette différence avec la courbe précédente.



On peut aussi écrire directement

```
--> x=-1:0.01:3; plot(x,2*x.^2+1)
```

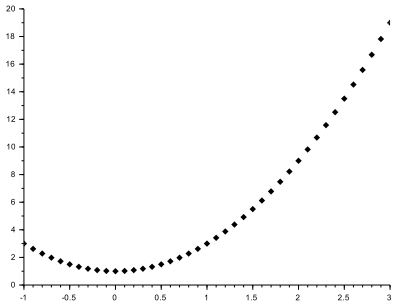
L'instruction `plot` est la plus simple pour tracer un graphique. Cependant si l'on veut paramétrer la forme des courbes en 2 dimensions, il faut utiliser `plot2d`.

Proposition 1.3 : *Tracer un graphique*

On construit le vecteur x des abscisses, on définit le vecteur $y = f(x)$, puis on utilise la commande `plot2d(x,y,style=Z)` où `Z` est une des valeurs suivantes :

-6	-5	-4	-3	-2	-1	0	1	2	3	4	5
△	◇	♦	⊕	×	+	·	noir	bleu foncé	vert	bleu clair	rouge

```
--> x=-1:0.1:3; plot2d(x,2*x.^2+1,style=-4)
```



Remarque 1.4 : *Pour avoir du style...*

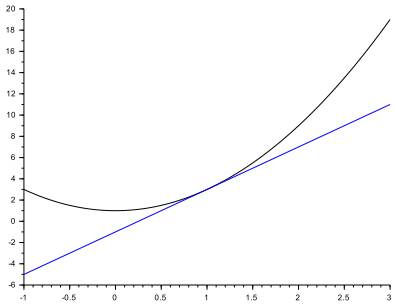
- Si aucun style n'est indiqué, *Scilab* trace une ligne brisée de couleur noire reliant les points.
- Si une valeur positive est donnée à `style`, on peut choisir la couleur de la ligne brisée.
- Si une valeur négative est donnée à `style`, *Scilab* trace l'ensemble des points sans les relier et la forme des points peut être choisie.

La commande `plot2d` permet de tracer plusieurs courbes sur un même graphique. On peut faire autant d'appels de `plot2d` qu'il y a de courbes à tracer.

```
--> x=-1:0.01:3; plot2d(x,2*x.^2+1); plot2d(x,4*x-1)
```

On peut aussi tracer deux courbes en un seul appel de `plot2d` (avec cette syntaxe, `x` doit être un vecteur colonne d'où l'instruction `x=x'`).

```
--> x=-1:0.01:3; x=x'; plot2d(x,[2*x.^2+1,4*x-1])
```



Proposition 1.5 : *Tracer un graphique en déclarant une fonction*

On construit le vecteur x des abscisses, on déclare la fonction f dont on veut tracer la courbe, puis on utilise la commande `fplot2d(x,f)`.

On peut tracer la même parabole que précédemment avec la commande suivante.

```
--> function y=f(x); y=2*x^2+1; endfunction; x=-1:0.01:3; fplot2d(x,f)
```

1.3 Surfaces (ou nappes) de l'espace

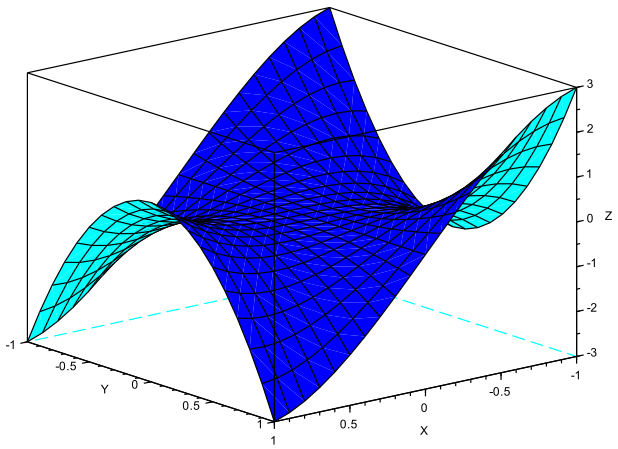
Cette partie n'a pas été traitée en première année, on va prendre ici un peu d'avance. On a les deux mêmes façons de faire pour dessiner des nappes de l'espace que pour dessiner les courbes du plan, mais avec les fonctions `plot3d` et `fplot3d`. Cependant, on évitera d'utiliser `plot3d` qui est une commande bien trop laborieuse à mettre en place, on lui préférera `fplot3d`.

Proposition 1.6 : *Tracer une nappe en déclarant une fonction*

On construit le vecteur x des abscisses et le vecteur y des ordonnées, on déclare la fonction f dont on veut tracer la nappe, puis on utilise la commande `fplot3d(x,y,f)`.

Afin de tracer la fonction $f : (x, y) \mapsto x^3 - 4\,x\,y^2$ sur le domaine $[-1, 1] \times [-1, 1]$, on utilise la commande suivante

```
--> function z=f(x,y); z=x^3-4*x*y^2, endfunction
--> x=-1:0.1:1;y=x;fplot3d(x,y,f)
```



Remarque 1.7 : *Rotation de la nappe*

Dans le cas d'une nappe, la figure proposée par *Scilab* n'est pas toujours très "lisible". Il faut cliquer sur l'icône "pivoter" en haut à gauche de la fenêtre graphique, puis faire tourner la figure en maintenant le clic droit enfoncé sur le graphique jusqu'à obtenir une vue correcte.

2 Représentations graphiques de données

2.1 La fonction tabul

Afin d'analyser et visualiser rapidement les propriétés d'une liste de données, souvent considérée comme un échantillon (i.e. une réalisation) d'une variable aléatoire discrète, on est souvent amené à vouloir connaitre les fréquences d'apparition de chacune des valeurs composant cet échantillon.

Dans l'exemple suivant, on va créer un vecteur composé uniquement de valeurs entières prises aléatoirement.

```
--> x=floor(10*rand(1,12)),
 x  =

    5.    4.    2.    6.    4.    9.    0.    4.    2.    4.    2.    1.
```

La fonction `tabul` va nous permettre de donner les valeurs distinctes prises par la variable dans la première colonne et le nombre d'apparitions de chacune de ces valeurs dans la seconde colonne.

```
--> m=tabul(x)
 m  =

    9.    1.
    6.    1.
    5.    1.
    4.    4.
    2.    3.
    1.    1.
    0.    1.
```

Dans cet exemple, on a une fois le 9, une fois le 6, une fois le 5, quatre fois le 4, trois fois le 2, une fois le 1 et une fois le 0.

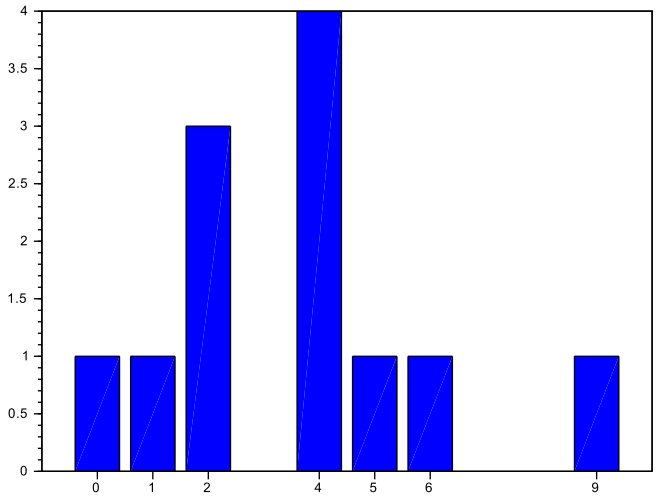
2.2 Diagramme en bâtons : la commande bar

Proposition 2.1 : *Tracer un diagramme en bâtons*

Pour dessiner un diagramme en bâtons, on utilise la commande `bar(v,w)` où v désigne la liste des valeurs distinctes prises par la variable et w la liste du nombre d'apparitions de chacune de ces valeurs.

On reprend le x et le m définis dans l'exemple précédent.

```
--> bar(m(:,1),m(:,2))
```



`m(:,1)` (la première colonne du vecteur `m`) donne les abscisses, ce sont la liste des valeurs distinctes prises. `m(:,2)` (la deuxième colonne du vecteur `m`) donne les ordonnées, c'est le nombre d'apparitions de chacune de ces valeurs.

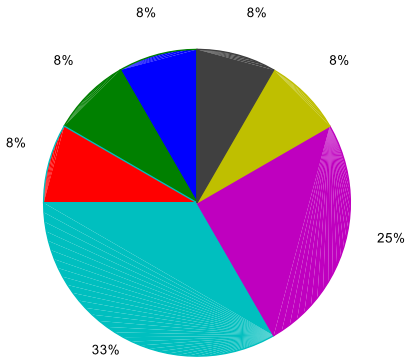
2.3 Diagramme circulaire : la commande pie

Proposition 2.2 : *Tracer un diagramme circulaire*

Pour dessiner un diagramme circulaire (*i.e.* diagramme en camembert), on utilise la commande `pie(w)` où w désigne la liste du nombre d'apparitions des valeurs.

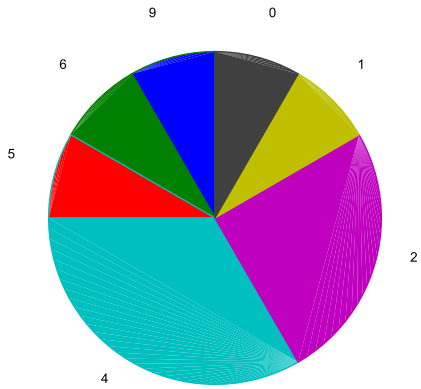
On reprend le x et le m définis dans l'exemple précédent.

```
--> pie(m(:,2))
```



Cette commande ne permet pas de distinguer quelle part désigne le 9, laquelle le 6, laquelle le 5.... on utilise alors la commande `pie(w,['x1',...,'xp'])` où `'x1',...,'xp'` sont les légendes. Ce qui nous donne dans l'exemple précédent

```
--> pie(m(:,2),['9','6','5','4','2','1','0'])
```



2.4 Histogramme : la commande histplot

Pour étudier les propriétés d'un échantillon d'une variable aléatoire discrète, on est souvent amené à regrouper ses valeurs par classes.

Proposition 2.3 : *Tracer un histogramme*

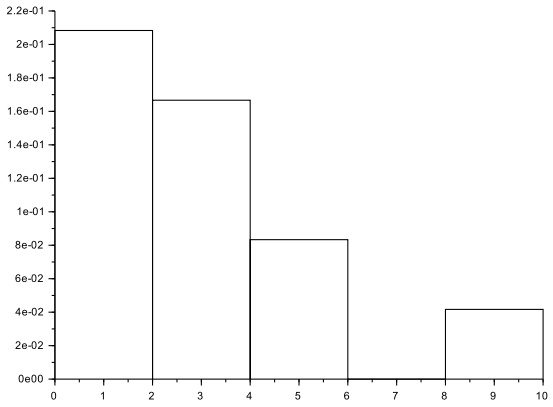
Pour dessiner un histogramme, on utilise la commande `histplot(c,x)` où c désigne une liste de classes et x les données.

Reprenons le x défini dans l'exemple précédent. On va découper l'intervalle [0,10] en 5 intervalles.

```
--> c=linspace(0,10,6)
 c  =

    0.    2.    4.    6.    8.    10.

--> histplot(c,x)
```



Dans cet exemple, l'histogramme a groupé nos données par classes de la manière suivante

classes	[0,2]	[2,4]	]4,6]	]6,8]	]8,10]
effectifs	5	4	2	0	1