Graphiques 2D avec Scilab

16 septembre 2015

Pour tracer "à la main" la courbe $y = f(x)$, on choisit quelques points d'abscisses x_0, x_1, x_2, puis on calcule les ordonnées correspondantes : y_0, y_1, y_2. Ensuite, on joint ces points avec habileté et quelques connaissances sur la fonction (notamment sur la dérivée). Plus le nombre de points est élevé, meilleure est l'allure de la courbe. Scilab fait le même raisonnement avec la commande `plot`, l'avantage est qu'il peut utiliser un nombre élevé de points.

1 L'instruction `plot`

L'instruction graphique la plus simple est :

```
plot(x,y)
```

où `x` et `y` sont des vecteurs contenant respectivement les abscisses et les ordonnées des différents points considérés.
Voici, en exemple, comment tracer la fonction $x \to \sin(x)$ sur l'intervalle $[-2\pi, 2\pi]$:

```
--> x = -2*%pi:%pi/100:2*%pi;
--> plot(x,sin(x));
--> plot(x,cos(x));
```

Contrairement à Matlab, l'exécution d'une deuxième instruction `plot` n'écrase pas la courbe déjà tracée.

1.1 Couleur et Style

On peut spécifier la couleur et le style de la courbe par l'instruction :

```
--> plot(x,y,'S')
```

où S est un des symboles donnés dans le tableau suivant :

Couleur		Symbole		Style de la ligne	
b	bleu	.	point	-	solid
g	vert	o	cercle	:	dotted
r	rouge	x	x-mark	-.	dashdot
c	cyan	+	plus	−−	dashed
m	magenta	*	étoile	(none)	no line
y	jaune	s	carré		
k	noir	d	diamand		
		v	triangle (down)		
		^	triangle (up)		
		<	triangle (left)		
		>	triangle (right)		
		p	pentagone		
		h	hexagone		

Exemple :

```
--> plot(x,sin(x),'r+-.')
```

1.2 Epaisseur du trait et du marqueur

```
--> plot(x,sin(x),'o-','markersize',15,'thickness',3)
```

1.3 Habiller son graphique

`xlabel('texte')`	ajoute une légende sur l'axe x
`ylabel('texte')`	ajoute une légende sur l'axe y
`title('texte')`	ajoute un texte en haut du graphique
`xstring(x,y,'texte')` :	positionne un texte au point de coordonnées (x,y)
`legend(['leg1', 'leg2', ...],pos)`	légende avec pos=1,2,3 ou 4 selon la position souhaitée
`xgrid`	(voir l'aide pour les paramètres) Ajoute une grille
`replot([xmin,ymin,xmax,ymax])`	change les axes
`zoomrect([xmin,ymin,xmax,ymax])`	change les axes

1.4 Changer l'allure des courbes a posteriori

Le "Axes handle" courant (=les propriétés des axes actuelles) est obtenu grâce à la commande :

```
aa=gca();
```

On peut en modifier les propriétés. En voici quelques unes :

`aa.data_bounds`=[xmin ymin ; xmax ymax]	axes de la figure
aa.grid=[cx cy]	cx, cy=-1 si pas de grille sinon 0,1,2 ... selon la couleur
`aa.font_style` = 6	police de caractères
`aa.font_size` = 3	taille des caractères
`aa.font_color` = 2	couleur des caractères
`aa.isoview="on"`	axes orthonormés
`aa.log_flags = "lnn"`	échelle logarithmique en x
`aa.log_flags = "nln"`	échelle logarithmique en y

Les propriétés de la courbe `i` sont stockées dans la variable :

```
cc= aa.children(1).children(i).
```

On peut également modifier les propriétés des courbes a posteriori :

`cc.thickness`=2	epaisseur du trait
`cc.children(i).foreground`=5	couleur du trait
`cc.line_style`=4	style du trait
`cc.mark_style` = 0	style du marqueur
`cc.mark_size`=2	taille du marqueur

Les valeurs par défaut du Axes handle ou du graphique sont retrouvées grâce aux commandes `sda()` et `sdf()`.

2 Gérer plusieurs graphiques

Si on veut observer plusieurs courbes en même temps, trois possibilités s'offrent à nous :

— on peut tracer toutes les courbes sur un même graphique et donc sur une même figure. Dans ce cas, on exécute les instructions `plot` les unes après les autres, les courbes se superposeront (ceci est différent dans Matlab). Exemple :

```
-->  x = [-%pi:0.01:%pi];
-->  plot(x,sin(x),'r*-');
-->  plot(x,cos(x),'b--');
```

— on peut tracer chaque courbe sur une figure différente. Dans ce cas l'instruction `figure()` permet d'ouvrir une nouvelle fenêtre graphique. Exemple :

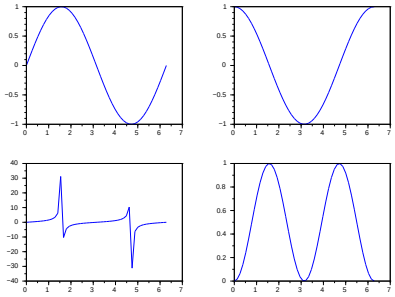
```
--> plot(x,sin(x),'r*-');
--> figure()
--> plot(x,cos(x),'b--');
```

— on peut subdiviser une figure en une matrice de sous-fenêtres en utilisant la commande `subplot`. Elle s'utilise de la manière suivante :

`subplot(m,n,p)`

Cette commande symbolise le découpage de la figure en une matrice m lignes et n colonnes. Le nombre p est compris entre 1 et $n*m$, il désigne le numéro de la sous-fenêtre où l'on veut tracer son graphe. Par exemple, si on veut tracer sin, cos, tan et $\sin^2$ sur une même figure mais sur des graphiques différents :

```
--> x = linspace(0,2*%pi,50);
--> subplot(2,2,1)
--> plot(x,sin(x))
--> subplot(2,2,2)
--> plot(x,cos(x))
--> subplot(2,2,3)
--> plot(x,tan(x))
--> subplot(2,2,4)
--> plot(x,sin(x).^2)
```



Quand plusieurs fenêtres sont ouvertes, voici quelques commandes qui permettent de les gérer :

`winsid()`	donne la liste des fenêtres ouvertes
`figure()`	crée une nouvelle fenêtre
`figure(5)`	crée la figure 5
`xdel()`	ferme la figure courante
`xdel(5)`	ferme la figure 5
`clf()`	efface le contenu de la figure courante

Exemple :

```
--> figure(0);          // cree la figure 0
--> figure(2);         // cree la figure 2
--> ff=winsid()        // cherche toutes les fenetres ouvertes
  ff  =

     0.    2.           // les fenetres 0 et 2 ont ete ouvertes
--> xdel(ff(2))        // ferme la deuxieme fenetre
--> xdel()             // ferme la figure courante, ici 0
--> figure(0);figure(12);
--> xdel(winsid())     // ferme toutes les fenetres ouvertes
--> figure(12);
--> plot(x,sin(x))
--> clf(12)            // efface le contenu de la fenetre 12
```

3 Sauver une figure sous le format .eps ou .gif

```
--> xs2eps(gcf(),'foo.eps')
--> xs2gif(gcf(),'foo.gif')
```

4 Animation

Une animation est une suite de graphiques affichés à l'écran. En voici un exemple :

```
f=gcf();
f.children(1).data_bounds=[0 -1 ; 1 1];
for i=1:100
    drawlater()
    clf()
    f.children(1).data_bounds=[0 -1 ; 1 1];
    plot(x',sin(i*x'/10));
    drawnow()
    xpause(100000)
end
```

5 Astuces

Pour tracer le logarithme d'une fonction qui s'annule :

```
ieee(1)
x=[0:0.01:1];
plot(x,log10(x))
```

Ceci produira un 'Warning' mais executera tout de même l'instruction.