Introduction à Scilab (4)

6 - Graphique

Scilab dispose de fonctionnalités afin de tracer des graphiques à partir de données sous forme de tableaux de valeurs. En MAP101, on utilisera essentiellement l'instruction `plot` sous l'une des formes suivantes

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

`plot(`x`,`y`,`*options_graphiques*`)`

6.1 - Représentation de points du plan

Les deux premiers arguments x et y de l'instruction `plot` sont deux vecteurs-lignes de même taille définissant les coordonnées de points dans le plan, le vecteur x correspond aux abscisses des points et le vecteur y correspond aux ordonnées des points.

On peut alors représenter un ensemble de points (x_i, y_i), $1 \leq i \leq n$ du plan, en définissant un vecteur `x` $= [x_1\ x_2\ \ldots\ x_n]$ et un vecteur `y` $= [y_1\ y_2\ \ldots\ y_n]$, chaque vecteur contenant n valeurs.

Exemple : Pour représenter les 4 points $(2,1)$, $(0,0)$, $(3,-1)$ et $(4,2)$, d'abord définir les deux vecteurs pour les abscisses et les ordonnées :

```
x1 = [ 2  0  3  4]
y1 = [ 1  0 -1  2]
```

puis effectuer le tracé :

```
scf(); // creer une nouvelle fenetre graphique
plot(x1,y1,".") // option "." : tracé de points
```

⇨ on remarque que les limites du repère correspondent aux limites des données soit l'intervalle $[0,4]$ en abscisse et l'intervalle $[-1,2]$ en ordonnée, et certains points sont peu visibles.

Pour modifier les limites du repère, il suffit d'utiliser l'instruction `replot([xmin,ymin,xmax,ymax])`

Exemple : Testez les instructions suivantes en observant le résultat après chacune d'elles :
`replot([-1 -2 5 3])` , `replot([-10 -10 10 10])` , `replot([0 0 4 4])`

On peut modifier le mode de tracé en modifiant le troisième paramètre de la procédure `plot` qui est une option de tracé sous forme d'une chaîne de caractères.

Exemple : Testez les instructions suivantes en observant le résultat après chacune d'elles et observez notamment le résultat visuel des différentes options :

```
scf(), plot(x1,y1,"g-"), replot([-3 -2 5 3])
scf(), plot(x1,y1,"k--"), plot(x1,y1,'ro'), replot([-3 -2 5 3])
scf(), plot(x1,y1), plot(x1,y1,'c.'), replot([-3 -2 5 3])
```

⇨ on peut effectuer différents tracés dans une même fenêtre graphique en effectuant plusieurs instructions `plot` à la suite.

Exemple : Rajoutez les instructions suivantes :

```
x2 = [-1  0  1  0 -1]
y2 = [ 0  1  0 -1  0]
plot(x2,y2,'c:'), plot(x2,y2,'m*')
```

⇨ les points des vecteurs `x2` et `y2` correspondent aux sommets d'un carré, alors que la représentation graphique ne donne pas nécessairement un carré (on voit plutôt un losange). Pour avoir une représentation graphique plus juste, il faut faire en sorte que le repère soit normalisé (même échelle en abscisse et en ordonnée).
Exécutez l'instruction suivante :

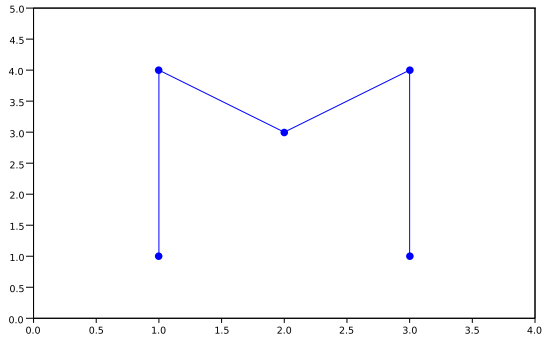
```
set(gca(),"isoview","on")
```

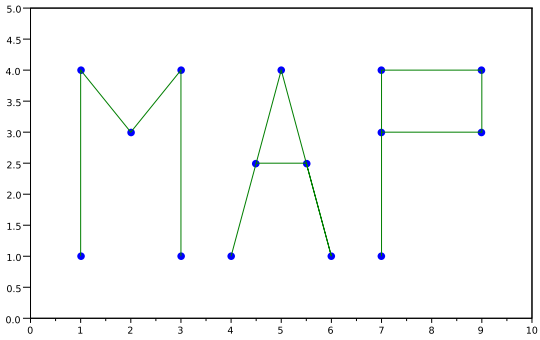
⇨ pour plus d'info sur l'instruction `plot`, tapez l'instruction `help plot` .

⇨ pour supprimer toutes les fenêtres graphiques, utiliser l'instruction `xdel(winsid())` .

Exercice 1 :

écrire les instructions Scilab afin d'obtenir les deux figures suivantes :





6.2 - Tracé de courbes représentatives de fonctions

Dans ce paragraphe, nous allons voir comment tracer le graphe $y = f(x)$ d'une fonction f dont on connait l'expression en fonction de x.

Le but de cette séance est que vous fassiez votre propre formulaire avec les représentations graphiques des principales fonctions usuelles.

Exemple : tracer le graphe de $f(x) = x^2$. Le domaine de définition de f étant $\mathbb{R}$, on ne peut pas représenter le graphe pour toutes les valeurs x de $\mathbb{R}$ mais seulement pour un intervalle $I = [a, b]$. De plus, on ne peut pas calculer la fonction f sur toutes les valeurs de l'intervalle I (car il y en a une infinité), mais seulement avec un nombre fini de valeurs entre a et b

Créez un fichier script Scilab nommé `trace_fonction1.sce` avec les instructions suivantes :

```
a = -2; b = 2; // les bornes de l'intervalle I
x = linspace(a,b,10); // choisir 10 valeurs de x entre a et b
deff("y = f(x)","y = x .^ 2"); // définir la fonction f
y = f(x); // calculer les valeurs y correspondant à x
scf(); // ouvrir une fenetre graphique
plot(x,y,'.');
```

et exécutez ce script. Seuls les 10 points du graphe de f correspondant aux 10 valeurs du tableau `x` sont affichés.

Pour avoir un tracé continu, il faut relier les plans entre eux : modifiez le script précédent en remplaçant l'instruction `plot(x,y,'.')` par `plot(x,y,'-')` et ré-exécutez-le. Le graphique présente une courbe continue mais avec des points anguleux (la courbe a un rendu visuel *non lisse*).

Pour avoir une courbe avec un rendu visuel lisse, il faut augmenter le nombre de valeurs dans le tableau `x` : modifiez le script précédent en remplaçant l'instruction `x = linspace(a,b,10)` par `x = linspace(a,b,1000)` et ré-exécutez-le.

Ensuite, on peut éventuellement enrichir et personnaliser la représentation : rajouter à la fin du script les instructions suivantes :

```
replot([-2,-1,2,5]); // modifier les bornes du repère
xtitle("f(x) = x^2") // titre du graphique
axes = gca(); // le repère-axes graphique
axes.x_location = "origin"; // repère-axes passant par l'origine
axes.y_location = "origin";
axes.box = "off"; // supprimer la boite englobant le repère-axes
axe.isoview = "on"; // normaliser le repère
```

Exemple : : tracer le graphe de $g(x) = \left\{ \begin{array}{ll} \exp(x) & \text{si } x < 0 \\ -x^2 + x + 1 & \text{si } x \geq 0 \end{array} \right\}$ pour x entre -1 et 1.

On peut utiliser la méthode précédente en définissant g sous forme d'une *procédure Scilab* (cf. exemple du paragraphe 5.7). Créez un fichier script Scilab nommé `trace_fonction2.sce` avec les instructions suivantes puis exécutez-le :

```
// définition de la fonction g(x)
function y=g(x)
        if x<0 then
                y = exp(x);
        else
                y = -x^2+x+1
        end
endfunction

// calcul de la fonction g pour des valeurs x entre -1 et 1
n = 1000; // nombre de valeurs pour x
x = linspace(-1,1,n);
y = zeros(1,n); // créer un vecteur de meme taille que x
for k=1:n
        y(k) = g(x(k));
end

// tracé de g
scf();
plot(x,y);
```

ou bien définir et tracer g pour $x < 0$ puis définir et tracer g pour $x \geq 0$. Créez un fichier script Scilab nommé `trace_fonction3.sce` avec les instructions suivantes puis exécutez-le :

```
scf();

// définition et tracé de g pour x < 0
deff("y = g(x)" , "y = exp(x);");
```

```
x = linspace(-1,-10^(-8),1000);
y = g(x);
plot(x,y,"b-");

// définition et tracé de g pour x >= 0
deff("y = g(x)" , "y = - x .^ 2 + x + 1;");
x = linspace(0,1,1000);
y = g(x);
plot(x,y,"r-");
```

⇨ pour tracer g sur l'intervalle $[-1, 0[$, on le trace sur l'intervalle $[-1, 0 - \varepsilon]$ avec ε très petit (ici $\varepsilon = 10^{-8}$).

⇨ notez les avantages et inconvénients des deux méthodes.

Exercice 2 :

Cet exercice fera l'objet d'un compte-rendu noté.

Le but de cet exercice consiste en la création de votre propre formulaire avec les graphiques des fonctions usuelles.

Pour cela, inspirez-vous des exemples précédents afin de tracer dans des fenêtres séparées les courbes représentatives des différentes fonctions.

Ensuite il faut rassembler les différentes figures des fenêtres graphiques dans un document de type *traitement de texte* : démarrez sous Windows l'application *LibreOffice*, puis par le menu *Fichier*, créez un nouveau document de type texte. Indiquez au début du document vos noms et prénoms.

Ensuite, faites un copier-coller des différentes figures dans ce document : pour chaque fenêtre graphique faite avec Scilab, copiez le contenu dans le presse-papier (menu *Fichier*, item *Copier dans le presse-papier*) puis allez dans votre document texte *LibreOffice* et collez la figure à l'endroit voulu. Vous pouvez ensuite réduire la taille de la figure si vous le souhaitez.

Une fois votre formulaire terminé, sauvegardez-le puis exportez-le au format PDF (menu *Fichier*, item *Exporter au format PDF*) puis envoyez ce fichier PDF par e-mail à votre enseignant de TP en indiquant comme sujet du message :

`[MAP101] - Compte-rendu TP 1 -` *noms du binôme*