26

Compléments sur les représentations graphiques

Remarque. Dans la suite, on suppose que les *imports* usuels pour PYTHON on été réalisés. □

```
from math import *
from cmath import *
import scipy as sp
import numpy as np
import matplotlib.pyplot as plt
```

◇ Pour faire un tracé, on utilise en général un vecteur x qui représente des abscisses et un vecteur y qui représente les ordonnées associées. Il y a deux possibilités pour obtenir le vecteur y à partir de x.

◇ On peut utiliser les opérations disponibles avec NUMPY. Par exemple pour représenter la fonction $x \mapsto \sin(x)^2$ sur $[-5,5]$:

```
x=np.linspace(-5,5,100)
y=np.sin(x)**2
```

◇ On peut également définir la fonction considérée à part et utiliser la construction `for ... in ...`. Remarque : lorsque x est un vecteur, on utilisera la notation xi pour désigner l'une de ses composantes.

```
def f(x):
    return sin(x)**2
x=np.linspace(-5,5,100)
y=np.array([f(xi) for xi in x])
```

◇ Avec SCILAB le principe est le même. Première possibilité :

```
x=linspace(-5,5,100);
y=sin(x).^2;
```

◇ Deuxième possibilité :

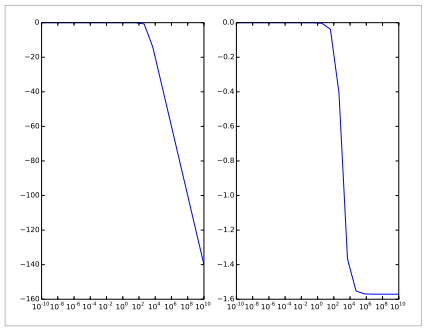
```
deff('y=f(x)','y=sin(x)^2')
x=linspace(-5,5,100);
y=feval(x,f);
```

◇ On présente les tracés en échelle semi-logarithmique en abscisses (on peut également avoir des échelles semi-logarithmiques en ordonnées ou logarithmiques sur les deux axes).

Exemple. On veut tracer le diagramme de Bode de la fonction de transfert d'un circuit RC ci-contre. On commence par définir les fonctions correspondant au gain $G(\omega) = 20\log\left(|H(j\omega)|\right)$ (logarithme en base 10) et à la phase $T(\omega) = \arg(H(j\omega))$. Pour simplifier, on note $h(\omega) = H(j\omega)$.

$$H(j\omega) = \frac{1}{1+\frac{j\omega}{\omega_0}}$$
$$\omega_0 = \frac{1}{RC} = 10^3 \text{ rad}\cdot\text{s}^{-1}$$

```
def h(w):
    w0=1e3
    return 1.0/(1+1j*w/w0)
def G(w):
    return 20*log10(abs(h(w)))
def T(w):
    return phase(h(w))
ww=np.logspace(-10,10,20)
plt.subplot(1,2,1)
plt.semilogx(ww,[G(w) for w in ww])
plt.subplot(1,2,2)
plt.semilogx(ww,[T(w) for w in ww])
plt.show()
```



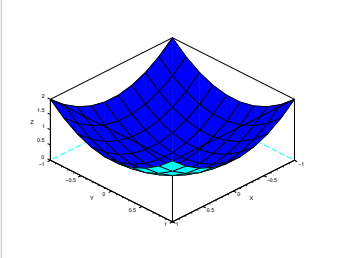
Remarque. La commande **`logspace(a,b,n)`** donne n valeurs espacées *logarithmiquement* de 10^a à 10^b. La fonction **`semilogx`** réalise un tracé semi-logarithmique en abscisses. □

◇ Ci-dessous, on donne une traduction pour SCILAB (on obtient la même représentation graphique). On obtient une échelle logarithmique en utilisant la commande **`plot2d`** avec l'option **`logscale="ln"`** (pour une abscisse logarithmique et une ordonnée normale, pour l'inverse on préciserait **`"nl"`** et **`"ll"`** pour que les deux échelles soient logarithmiques).

```
w0=1e3;
deff('z=h(w)','z=1/(1+%i*w/w0)')
deff('g=G(w)','g=20*log10(abs(h(w)))')
deff('t=T(w)','t=atan(imag(h(w)),real(h(w)))')
ww=logspace(-10,10,20);
subplot(1,2,1)
plot2d(ww,feval(ww,G),logflag="ln")
subplot(1,2,2)
plot2d(ww,feval(ww,T),logflag="ln")
```

◇ Un exemple de représentation 3D avec SCILAB :

```
x=linspace(-1,1,10);
y=linspace(-1,1,10);
deff('z=f(x,y)','z=x.^2+y.^2')
plot3d(x,y,feval(x,y,f))
```



Remarque. Pour une représentation 3D avec PYTHON, se reporter par exemple à http://matplotlib.org/mpl_toolkits/mplot3d/. □