Université de Picardie Jules Verne
Licence 2
Scilab
2017-2018

Fonctions - Représentation et étude

1 Approximation des fonctions

1.1 Etude locale

La fonction "zoom" sur la fenêtre graphique de Scilab permet de grossir une région sélectionnée à l'aide de la souris (ou du pad), touche 'loupe'. Elle permet donc d'illustrer les études locales de fonctions.

Exemple
soit $f(x) = \sin(x)$. on étudie f au voisinage de 0 en comparant la fonction 'a ses d.l. pris à des ordres croissants

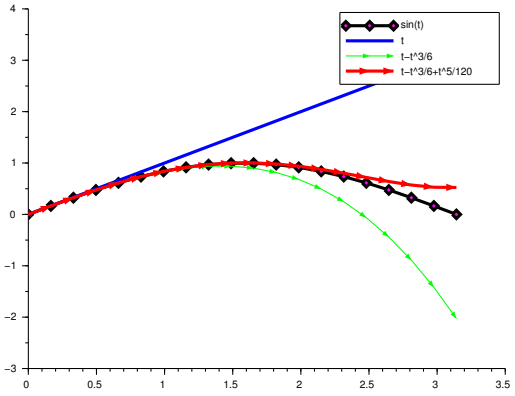

Figure 1: *la fonction* $\sin(x)$ *et ses d.l.*

1.2 Approximation des fonctions par des polynômes

Théorème 1 *Soit f une fonction de $\mathcal{C}([a,b])$. Alors, pour tout $\epsilon > 0$, il existe un polynôme P tel que*

$$\max_{x\in[a,b]} |f(x) - P(x)| < \epsilon.$$

On dispose en fait de l'estimation

$$\max_{x\in[a,b]} |f(x) - P(x)| < cw(\frac{1}{\sqrt{n}})$$

où w désigne un module de continuité de f.

Preuve. Par exemple, en se ramenant à l'ensemble $\{f \in \mathcal{C}([0,1]), f(0) = f(1) = 0\}$, on démontre de manière constructive le théorème de Stone-Weierstrass à l'aide des polynômes de Bernstein

$$B_n(f)(x) = \sum_{k=0}^{n} C_k^n x^k (1-x)^{n-k} f(\frac{k}{n})$$

On montre que la suite $B_n(x)$ converge uniformément vers f. ■
Cette convergence peut être très lente.

Remarque 2 *On pourra consulter le livre de M. Schatzman [1] pour la preuve, ou bien S. Lang [?], p 211. le théorème et sa démonstrations par les polynmes de Bernstein (alternative par la méthode des noyaux, cf S. Lang, voir annexe.*

Voici une autre démonstration On rappelle la

Définition 3 *(Suite de noyaux de Dirac)*
Soit K_n une suite de fonctions réelles définies sur $\mathbb{R}$. *On dit que (K_n) forme une suite de Dirac (*Dirac sequence*) si*

- $K_n(x) \geq 0, \forall x \in \mathbb{R}$
- *Chaque $K_n(x)$ est continu et* $\int_{-\infty}^{+\infty} K_n(x)dx = 1, \forall n \geq 0$
- *Etant donné $\epsilon > 0$ et $\delta > 0$ il existe N tel que $\forall n \geq N$*

$$\int_{-\infty}^{-\delta} K_n(x)dx + \int_{\delta}^{+\infty} K_n(x)dx < \epsilon$$

On montre le

Théorème 4 *Soit*

$$K_n(x) = \begin{cases} \frac{(1-t^2)^n}{c_n} & si \ -1 \leq t \leq 1 \\ 0 & sinon \end{cases}$$

Soit $f \in \mathcal{C}([0,1)]$ telle que $f(0) = f(1) = 0$. Alors la suite de fonctions $f_n(x) = K_n * f(x)$ est une suite de polyn mes convergeant uniformément vers f.

Preuve. Voir l'exercie en annexe. ■

1.3 Interpolation polynomiale

Soit f continue sur $[a,b]$ et x_i, $i=1,\cdots,n+1$, $n+1$ points 2 à 2 disctincts. On cherche à construire un polynôme $p(x)$ tel que

$$(\mathcal{P}) \quad f(x_i)=p(x_i), i=1,\cdots,n+1.$$

Ce problème équivaut à déterminer les réels $(a_i)_{i=0}^n$ tels que

$$p(x_i)=\sum_{,=0}^{n} a_j x_i^j.$$

Nous avons le

Théorème 5 *Le probléme $(\mathcal{P})$ admet une solution unique si et seulement si les réels $(x_i)_{i=1}^{n+1}$ sont distincts deux à deux.*

Le polynôme p peut se construire de diverses manières, par exemple en résolvant le Vandermonde de matrice

$$V=\begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_1^n \\ \vdots & & & & \\ 1 & x_k & x_k^2 & & x_k^n \\ \vdots & & & & \\ 1 & x_{n+1} & x_{n+1}^2 & \cdots & x_{n+1}^n \end{pmatrix},$$

On préfèrerera la construction de p dans la Base de Langrange et surtout dans celle de Newton, on se réfèrera au cours d'analyse numérique.

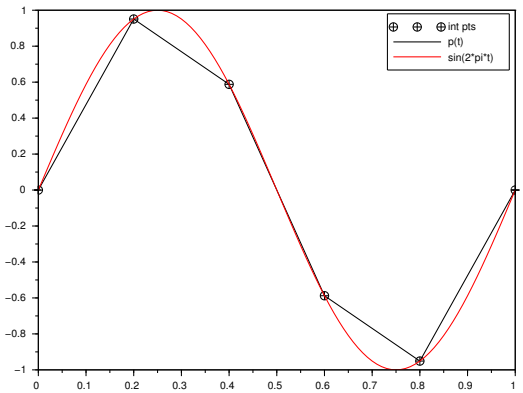

Figure 2: *Interpolation linéaire par morceaux de* $\sin(2\pi x)$

1.4 Représenter et évaluer un polynôme en un point

Les polynômes sont les fonctions les plus simples que l'on puisse construire à l'aide des 3 opérations $+,-,*$. La façon la plus simple de représenter un polynôme p de degré inférieur ou égal à n est de

l'exprimer dans la base canonique $\{1, x, x^2, \cdots, x^n\}$ comme

$$p(x) = \sum_{k=1}^{n} a_k x^k.$$

Pour autant cette écriture ne suggère pas la méthode la plus efficace pour calculer (évaluer, donner une valeur) $p(x)$ pour x donné. Faisons le décompte des opérations.
Pour calculer $a_j x^j$, on doit effectuer j multiplications. Le nombre totale de multiplications est donc

$$N_{mult} = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

Il faut ajouter à cela n additions. Si une addition a le même coût qu'une multiplication, alors le nombre totale d'opérations est de $N_{tot} = n + \frac{n(n+1)}{2} \simeq \frac{n^2}{2}$ quand n est grand. Il est à noter que ce calcul est intimement liè à la forme de l'expression de p. Pour changer de mode de calcul, réarrangeons les termes. Nous pouvons écrire

$$p(x) = a_0 + x\left(a_1 + a_2 x + \cdots + a_n x^{n-1}\right)$$

et en appliquant une fois cette factorisation partielle à l'interieur de la parenthèse, nous avons

$$p(x) = a_0 + x\left(a_1 + x\left(a_2 + \cdots + a_n x^{n-2}\right)\right).$$

Bien sûr ce procédé peut être appliqué récursivement jusqu'à obtenir

$$p(x) = a_0 + x\left(a_1 + x\left(a_2 + \left(a_3 + \cdots \left(a_{n-1} + a_n x\right)\right)\cdots\right)\right).$$

Cette écriture suggère de calculer $p(x)$ de proche en proche en commençant par le terme le plus à l'intérieur de cette expression. L'algorithme de Horner peut se résumer ainsi

`Algorithme de Horner`

`Initialisation`	$b_0 = a_n$
`Pour` $k = 1, \cdots n$	
`poser`	$b_k = x b_{k-1} + a_{k-1}$
`poser`	$p(x) = b_n$

Faisons maintenant le décompte des opérations. A chaque ligne on effectue exactement une addition et une multiplication. Avec n lignes, nous obtenons

$$N_{tot}^{Horner} = 2n$$

ce qui est asymptotiquement une réduction très importante du nombre d'opérations. En effet si $n = 1000$, avec la méthode classique nous devrons effectuer environ 500000 opérations tandis que celle d'Horner n'en recquiert que 2000. Les opérations sont réduites, le calcul est plus rapide mais aussi plus sûr car sur ordinateur, en précision finie, chaque opération génère des erreurs, d'arrondis notamment.

2 Suites de fonctions

2.1 Drôles de dessins

2.1.1 Lignes très brisées

Marche de l'ivrogne
On se donne $n \in \mathbb{N}$ et x un vecteur de taille n dont les composantes tirées au sort équiprobablement valent 1 ou -1. On considère la suite des sommes partielles

$$S_k = \sum_{k=1}^{n} x_i$$

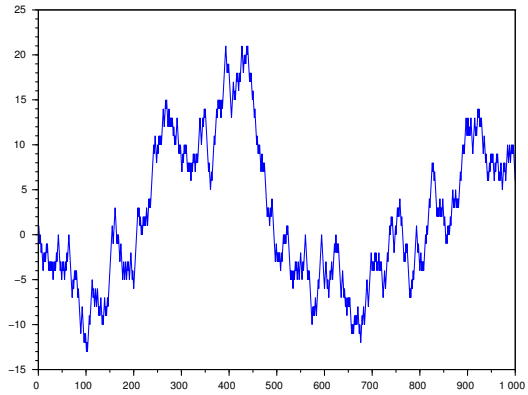

Figure 3: *Marche de l'ivrogne nmax* = 100

2.1.2 Fonctions bizarres : continues et nulle part dérivables

La fonction de Weierstrass On la définie comme

$$f_{a,b}(x) = \sum_{n=0}^{+\infty} a^n \cos(b^n \pi x)$$

avec $0 < a < 1$ et $ab > 1 + \frac{3\pi}{2}$
Le graphe de cette fonction devient de plus en plus erratique au fur et à mesure que b augmente, on peut le voir sur la figure (2.1.2). Dans ce cas, $f(x)$ st continue te nulle part dérivable.

Nous allons étudier ce type de fonctions exprimées sous la forme

$$f(x) = \sum_{n=0}^{+\infty} \frac{\sin(2\pi N^n x)}{c^n}$$

avec N un entier pair vérifiant $1 < c < \frac{N}{1+6\pi}$

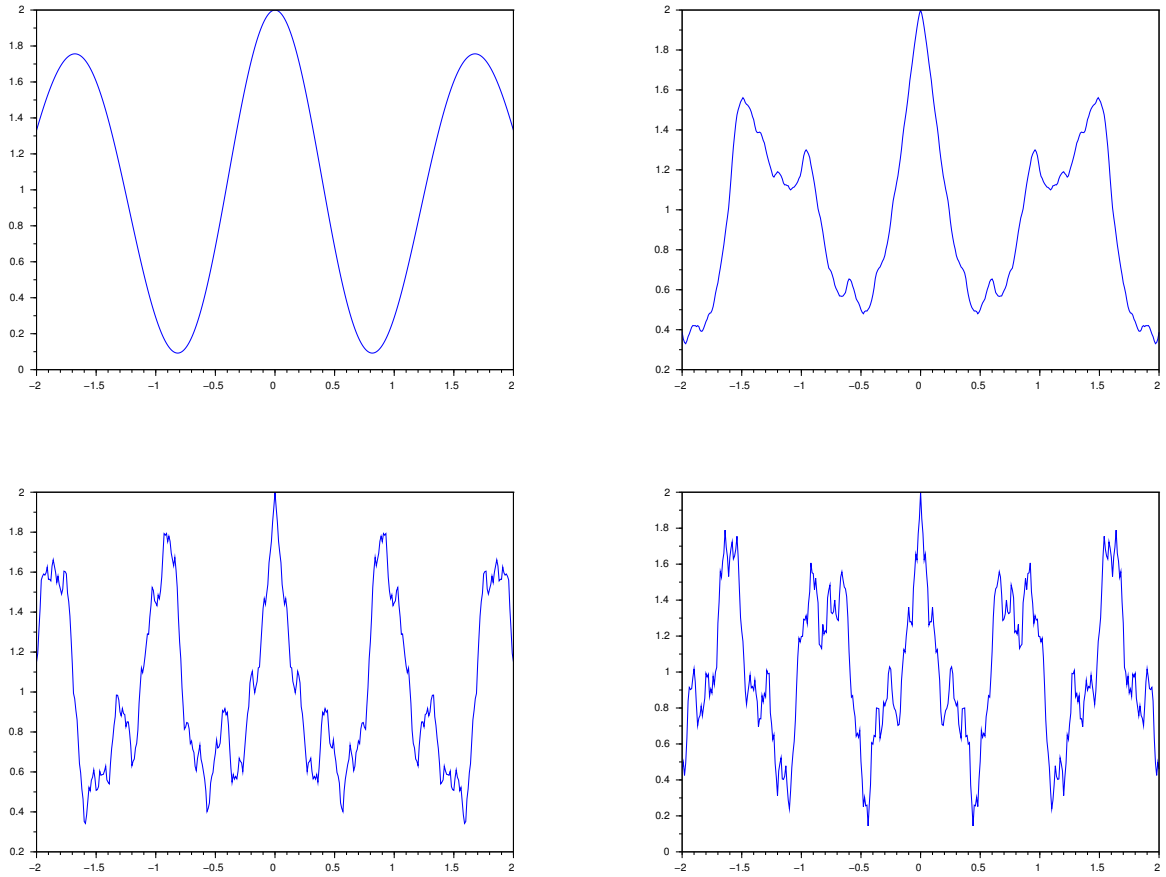

Figure 4: *Fonction de Weiertrass pour* $a = 0.5, b = 1.1$ *(à gauche, en haut),* $a = 0.5, b = 1.6$ *(à droite, en haut),* $a = 0.5, b = 2.1$*, (à gauche, en bas),* $a = 0.5, b = 2.5$ *(à droite, en bas),* $nmax = 600$

Théorème 6 *La fonction* $f(x)$ *est continue sur* $\mathbb{R}$ *mais nulle part dérivable.*

Pour établir ce résultat, nous allons au préalable utiliser le lemme suivant :

Lemme 7 *Si* f *est dérivable en* a*, et si* (x_n) *et* (y_n) *sont deux suites convergeant vers* a *telles que pour tout* n*,* $y_n \leq a \leq x_n$ *et* $y_n < x_n$*, alors* $\lim_{n\to+\infty} \dfrac{f(x_n) - f(y_n)}{x_n - y_n} = f'(a)$.

Preuve. Si f est drivable en a, on peut crire $f(a+h) = f(a) + hf'(a) + h\epsilon(h)$), où $\epsilon(.)$ est une fonction telle que $lim_{h\to 0}\epsilon(h) = 0$. On introduit à présent les suites

$$x_n = a + \alpha_n \text{ et } y_n = a - \beta_n$$

Nous avons

$$\frac{f(x_n) - f(y_n)}{x_n - y_n} = \frac{(\alpha_n + \beta_n)f'(a) + \alpha_n\epsilon(\alpha_n) + \beta_n\epsilon(\beta_n)}{\alpha_n + \beta_n} = f'(a) + r_n$$

avec $r_n = \dfrac{\alpha_n\epsilon(\alpha_n) + \beta_n\epsilon(\beta_n)}{\alpha_n + \beta_n}$. Il reste à montrer que r_n tend vers 0 lorsque n tend vers $+\infty$. Nous avons

$$|r_n| \leq \frac{\alpha_n}{\alpha_n + \beta_n}|\epsilon(\alpha_n)| + \frac{\beta_n}{\alpha_n + \beta_n}|\epsilon(\beta_n)| \leq |\epsilon(\alpha_n)| + |\epsilon(\beta_n)|$$

d'où le résultat. ■

Nous pouvons maintenant démontrer le théorème

Preuve. du théorème

Tout d'abord, notons que f est bien continue en tout point de $\mathbb{R}$ comme série normalement convergente de fonctions continues, puisque $|\frac{\sin(2\pi N^n x)}{c^n}| \leq \frac{1}{c^n}$, avec $c > 1$. Soit $a \in \mathbb{R}$ fixé. Nous procédons par l'absurde et supposons que f est dérivable en a. Nous allons construire deux suites (x_n) et (y_n) convergentes vers a, vérifiant les hypothèses du lemme, et telles que $\dfrac{f(x_n) - f(y_n)}{x_n - y_n} \geq c^n$, ce qui montrera que f ne peut être dérivable. Soit k_n le plus grand entier relatif tel que

$$k_n \leq aN^n + \frac{1}{4\pi}$$

On pose

$$x_n = \frac{4k_n + 5}{4N^n} \text{ et } y_n = \frac{4k_n - 1}{4N^n}.$$

Les suites (x_n) et (y_n) vérifient $y_n \leq a < x_n$: par construction $y_n < x_n$ et d'autre part, par définition de k_n

$$k_n + 1 > aN^n + \frac{1}{4\pi}$$

donc

$$x_n = \frac{4(k_n + 1) + 1}{4N^n} > \frac{4(k_n + 1)}{4N^n} > a + \frac{1}{\pi N^n}.$$

Par ailleurs,

$$4k_n - 1 \leq 4aN^n + \frac{1}{\pi} - 1\text{d' où} y_n = \frac{4k_n - 1}{4N^n} \leq a + \frac{1}{4a^n}\left(\frac{1}{\pi} - 1\right) \leq a.$$

Enfin $x_n - y_n = \dfrac{3}{2N^n} \to 0$ quand $n \to +\infty$.

Posons $f_k(x) = \dfrac{\sin(2\pi N^k x)}{c^k}$ et considérons maintenant l'expression

$$\rho_n = \sum_{k=0}^{+\infty} \frac{f_k(x_n) - f_k(y_n)}{x_n - y_n}$$

Nous remarquons que pour $k > n$

$$f_k(x_n) = \frac{\sin(2\pi N^k x_n)}{c^k} = \frac{\sin(2\pi N^{k-n}\frac{4k_n + 5}{4})}{c^k} = \frac{1}{c^k}\sin(N^{k-n}(2\pi k_n + 5\frac{\pi}{2})) = \frac{1}{c^k}\sin(N^{k-n}5\frac{\pi}{2}) = 0$$

car N est un entier pair. On montrer de même que pour $k > n$ $f(y_n) = 0$. Lorsque $k = n$, $f_k(x_n) = 1$ et $f_k(y_n) = -1$. Du coup, l'expression de ρ_n devient

$$\rho_n = \frac{2}{c^n(x_n - y_n)} + \sum_{k=0}^{n-1} \frac{f_k(x_n) - f_k(y_n)}{x_n - y_n}.$$

Du théorème des accroissements finis appliqué $\mathrm{f}_k(x)$, nous déduisons

$$\frac{|f_k(x_n) - f_k(y_n)|}{x_n - y_n} \leq |f_k'(c_n)| = 2\pi \left(\frac{N}{c}\right)^n |\cos(2\pi N^n c_n)| \leq 2\pi \left(\frac{N}{c}\right)^n$$

Ainsi

$$\rho_n \geq \frac{2}{c^n(x_n - y_n)} - \sum_{k=0}^{n-1} \frac{|f_k(x_n) - f_k(y_n)|}{x_n - y_n} \geq \frac{(N/c)^n}{3} - \sum_{k=0}^{n-1} 2\pi \left(\frac{N}{c}\right)^n$$

soit

$$\rho_n \geq \frac{(N/c)^n}{3} - 2\pi \frac{(N/c)^n - 1}{(N/c) - 1}.$$

L'hypothèse $c < \frac{N}{1+6\pi}$ implique que la suite ρ_n est divergente et donc que la fonction f ne peut être dérivable en a. ■

La fonction Blanc-manger Il s'agit de la courbe de Takagi (1903) dont le nom blanc manger évoque un entremet du même nom.

$$b(x) = \sum_{n=0}^{+\infty} \frac{E(2^n x)}{2^n}$$

où $E(x) = \min_{y \in Z} |x - y|$

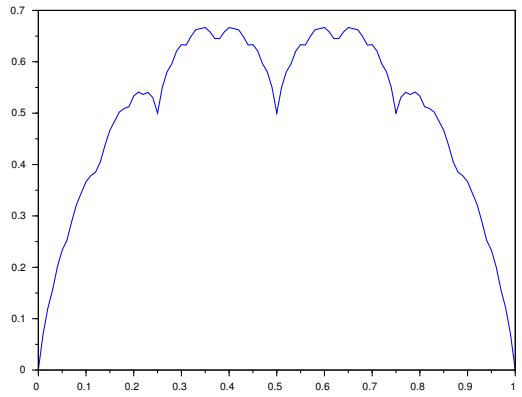

Figure 5: *Fonction de Blanc Manger nmax* = 400

La fonction de Bolzano La fonction de Bolzano est définie comme la limite d'une suite de fonctions $B_k(x)$, qui se construit par récurrence. On se donne un intervalle de définition $[a, b]$ et un intervalle image $[A, B]$. ON part de $B_0(x)$, fonction affine envoyant $[a, b]$ dans $[A, B]$, soit

$$B_0(x) = A + \frac{B - A}{b - a}(x - a)$$

La fonction $B_1(x)$ est linéaire par morceaux et se définit sur 4 sous-intervalles de même taille :

$$B_1(a) = a,\ B_1(a + \tfrac{3}{8}(b-a)) = A + \tfrac{5}{8}(b-a),$$
$$B_1(\tfrac{1}{2}(a+b)) = A + \tfrac{1}{2}(B-A),\ B_1(a + \tfrac{7}{8}(b-a)) = B - \tfrac{1}{8}(B-A),$$
$$B_1(B) = b.$$

La fonction B_2 sear aussi linéaire par morceaux, on appliquera sur chaque morceaux la transformation utilisée pour définir B_1.

3 annexe

3.1 Noyau de Dirac

Exercice 1 *1. Soit f une fonction continue sur un compact S. On pose $f_n(x) = f * K_n$ où K_n est une suite de Dirac.*

(a) Montrer que

$$|f(x) - f_n(x)| \leq \int_{|y|>\delta} |f(x) - f(x-y)|K_n(y)dy + \int_{|y|\leq\delta} |f(x) - f(x-y)|K_n(y)dy$$

puis que

$$\int_{|y|>\delta} |f(x) - f(x-y)|K_n(y)dy \leq 2M \int_{|y|>\delta} K_n(y)dy, \ \forall\delta > 0$$

où M est un majorant de $\|\|_\infty$ sur S.

(b) Montrer que pour $x \in S$ Alors pour $\epsilon > 0$ petit, il existe $\delta > 0$ tel que

$$|y| \leq \delta \Rightarrow |f(x) - f(x-y)| < \epsilon.$$

(c) En déduire que

$$|f(x) - f_n(x)| \leq \epsilon 2M \left(\int_{|y|>\delta} K_n(y)dy + \epsilon \int_{|y|\leq\delta} K_n(y)dy \right)$$

et conclure.

2. Pour démontrer le théorème de Weierstrass, on se place tout d'abord sur $[-1, 1]$ et on considère les fonctions continues f telles que $f(1) = f(-1) = 0$, ce qui ne fait rien perdre en généralité. On introduit

$$K_n(x) = \begin{cases} \frac{(1-t^2)^n}{c_n} & si \ -1 \leq t \leq 1 \\ 0 & sinon \end{cases}$$

(a) Montrons que $K_n(x)$ est une suite de Dirac

i. Montrer que

$$\frac{c_n}{2} \geq \frac{1}{n+1}$$

ii. Montrer que pour $\delta > 0$ on a

$$\int_\delta^1 K_n(x)dx \leq \frac{n+1}{2}(1-\delta^2)^n(1-\delta)$$

iii. Conclure

(b) Montrer que $K_n(x)$ est un polynôme.

(c) Conclure.

3.2 Interpolation

- `interpln` interpolation linéaire
- `smooth` interpolation par une spline
- `bsplin3val` 3d spline arbitrary derivative evaluation function
- `cshep2d` bidimensional cubic shepard (scattered) interpolation
- `eval_cshep2d` bidimensional cubic shepard interpolation evaluation
- `interp` cubic spline evaluation function
- `interp1` one-dimension interpolation function
- `interp2d` bicubic spline (2d) evaluation function
- `interp3d` 3d spline evaluation function
- `linear_interpn` n dimensional linear interpolation
- `lsq_splin` weighted least squares cubic spline fitting
- `splin` cubic spline interpolation
- `splin2d` bicubic spline gridded 2d interpolation
- `splin3d` spline gridded 3d interpolation

3.3 Graphiques avec Scilab

Graphiques avec Scilab

- Graphe simple

```
x = linspace(-% pi,% pi,11); // subdivision [-pi,pi] avec 11 points.
y = cos(x)./(1+x.^ 2); // (% pi est une constante Scilab)
clf() // efface la fenêtre graphique courante
plot(x,y,'b')
```

On peut modifier le script en jouant avec la longueur de l'intervalle et/ou le paramètre de discrétisation et/ou la couleur des traits `('b', 'g', 'r', 'k', 'c', 'm')` correspondent respectivement aux couleurs bleu, vert, rouge, noir, cyan, magenta).

- Plus généralement la fonction `plot` peut être utilisée pour dessiner une ou plusieurs courbes :
`plot(x1,y1[,style1],x2,y2[,style2], ....)`
où style est une option (couleur, type de trait ou symbol)

Exemple

```
x = linspace(0,2% pi,31);
y1 = sin(x); y2 = cos(x);
scf(0); // selectionne 0 comme label ou numéro de la fenêtre graphique par défaut
clf(); // efface la fenêtre graphique
plot(x,y1,"b-",x,y2,"r--"); // lignes seules
scf(1); // selectionne 1 comme label ou numéro de la fenêtre graphique par défau
clf(); // efface la fenêtre graphique
```

```
subplot(2,1,1); // organise la fenêtre graphique comme une matrice 2×1 et utilise
la sous-fenêtre 1
plot(x,y1,"ro",x,y2,"bx"); // symboles seuls
subplot(2,1,2); // utilise la sous-fenêtre 2
plot(x,y1,"r--o",x,y2,"g-x"); // lignes et symboles
```

- Plot3d `plot3d(x,y,z,<optargs>)`
  Exemple graphique simple avec z=f(x,y)

```
t=[0:0.3:2*% pi]';
z=sin(t)*cos(t');
plot3d(t,t,z)
```

couleurs				types de ligne		symboles			
k	black	c	cyan	-	solid	+	+	d	♢
b	blue	m	magenta	--	dashed	x	×	v	▽
r	red	y	yellow	:	dotted	o	◯	s	□
g	green	w	white	-.	dashdot	*	∗	*wedge*	△

3.4 Polynômes

Scilab permet de manipuler simplement les polynômes en les définissant par leurs coefficienst dans la base canonique ou bien par leurs racines.

La command `roots` (racines) permet de calculer numériquement les racines d'un polynôme, comme décrit dans l'exemple suivant :

```
//Ici un polynôme défini par ses racines
p = poly([1 2 3],"x")
roots(p)
// Là, on donne directement les coefficients du polynôme
p = [3 2 1] roots(p) // Ici un polynôme avec des racines complexes
p=poly([0,10,1+%i,1-%i],'x');
roots(p)
// racines du polynôme caractéristique d'une matrice
A=rand(3,3);
p = poly(A,'x')
roots(p)
spec(A)
```

Pour se convaincre de la difficulté de déterminer les racines d'un polynôme, voici e célèbre exemple du à Wilkinson :

```
n=10;
p=poly(1:n,'x');
r=roots(p)
// Représentons maintenant les racines dans le plan complexe
```

```
plot(real(r),imag(r),'*')
```

Jusqu'à $n = 20$ les racines calculées par Scilab sont très proches de celles du polyôme. Les difficultés apparaissent pour de plus grandes valeurs de n. On illustre cela en procédant à l'animation :

```
for n=5:5:40
p=poly(1:n,'x');
r=roots(p)
// On représente les racines dans le plan complexe
plot(real(r),imag(r),'*')
drawnow
//pause en microsec
xpause(1000000)
end
```

3.5 Weierstrass

```
//weierstrass
a=0.5;
b=2.1;
x=-2:0.01:2;
s=1.;
for k=1:200 // for i=(k-1)*10+1:10*k
s=s+a^k*cos(b^k*%pi*x);
end
plot(x,s)
```

On peut visualiser le comportement de plus en plus "chaotique en faisant varier b de 0.1 à 5 et en maintenant fixe la valeur de a, par exemple $a = 0.5$ `//weierstrass`

```
a=0.5;
b=0.1;
x=-2:0.01:2;
for b=0.1:0.1:5 s=1.;
for k=1:200 // for i=(k-1)*10+1:10*k
s=s+a^k*cos(b^k*%pi*x);
end
plot(x,s)
xpause(100000)
end
```

References

[1] M. Schatzman,*Analyse numérique pour la licence*, InterEditions, Paris, 1991.