Équation de diffusion : méthode de Fourier

1. Méthode

1.a. Hypothèses

On considère l'équation de diffusion à une dimension :

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \tag{1}$$

pour x dans l'intervalle $[-L/2, L/2]$ de largeur L.

On considère deux conditions limites différentes. La première (Dirichlet) consiste à poser un flux nul sur les frontières :

$$\frac{\partial u}{\partial x} = 0 \; pour \; x = \pm\frac{L}{2} \tag{2}$$

La seconde (Neumann) consiste à poser une valeur nulle sur les frontières :

$$u = 0 \; pour \; x = \pm\frac{L}{2} \tag{3}$$

La condition initiale est la fonction $u(x, 0)$.

La méthode de Fourier consiste à rechercher une solution combinaison linéaire de solutions de la forme :

$$u(x,t) = e^{-\frac{t}{\tau}} f(x) \tag{4}$$

Nous allons voir que cette approche conduit à rechercher la série de Fourier de la condition initiale, qui devra être soit paire soit impaire. Cette méthode fait partie de la classe des *méthodes spectrales*, qui consistent à développer la solution sur une base de fonctions. Elle est notamment utilisée en mécanique quantique pour résoudre l'équation de Schrödinger.

1.b. Modes propres

La fonction f vérifie l'équation :

$$\frac{d^2 f}{dx^2} + (D\tau)^{-1} f = 0 \tag{5}$$

On pose $k^2 = (D\tau)^{-1}$. La solution générale est :

$$f(x) = A\cos(kx) + B\sin(kx) \tag{6}$$
$$f'(x) = -Ak\sin(kx) + Bk\cos(kx) \tag{7}$$

Pour l'une ou l'autre des deux conditions initiales, les constantes (A, B) doivent vérifier :

$$A\cos\left(\frac{kL}{2}\right) + B\sin\left(\frac{kL}{2}\right) = 0 \tag{8}$$
$$A\cos\left(\frac{kL}{2}\right) - B\sin\left(\frac{kL}{2}\right) = 0 \tag{9}$$

Le déterminant de ce système doit être nul :

$$2\cos\left(\frac{kL}{2}\right)\sin\left(\frac{kL}{2}\right)=0 \tag{10}$$

La première possibilité est la suivante :

$$kL=2n\pi \tag{11}$$
$$f_D(x)=B\sin(kx) \tag{12}$$
$$f_N(x)=A\cos(kx) \tag{13}$$

où n est un nombre entier. L'indice D est pour la solution avec les conditions limite de Dirichlet. L'indice N est pour la condition de Neumann. La seconde possibilité est :

$$kL=(2n+1)\pi \tag{14}$$
$$f_D(x)=B\cos(kx) \tag{15}$$
$$f_N(x)=A\sin(kx) \tag{16}$$

On voit donc que la solution est soit paire soit impaire. Il en est de même de la condition initiale. En fonction de la parité de celle-ci, et des conditions limites choisies, on prendra l'une de ces quatre fonctions.

Notons $p=2n$ ou $p=2n+1$ l'indice du mode propre. Son temps de relaxation est :

$$\tau_p=\frac{L^2}{\pi^2p^2D} \tag{17}$$

On remarque que les modes de haute fréquence spatiale se relaxent très vite. Le mode le plus lent à se relaxer est le mode fondamental ($p=1$).

1.c. Série de Fourier

Considérons la somme suivante :

$$u(x,t)=\sum_{p=1}^{\infty}e^{-\frac{t}{\tau_p}}\left(A_p\cos\left(\frac{\pi p}{L}x\right)+B_p\sin\left(\frac{\pi p}{L}x\right)\right) \tag{18}$$

Suivant la parité de la condition initiale, et le type de condition limite, on retiendra seulement les coefficients A ou B, qui seront non nuls pour des indices pairs ou impairs (il y a quatre possibilités).

Cette fonction est bien solution de l'équation de diffusion avec les conditions limites choisies. Par ailleurs, la condition initiale s'écrit :

$$u(x,0)=\sum_{p=1}^{\infty}A_p\cos\left(\frac{2\pi p}{2L}x\right)+B_p\sin\left(\frac{2\pi p}{2L}x\right) \tag{19}$$

Il s'agit de la série de Fourier d'une fonction de période $2L$ définie par prolongement de la condition initiale sur l'intervalle $[-L,L]$. Pour faire ce prolongement, on doit tenir compte du fait que les coefficients de Fourier nuls sont soit ceux de rang pair, soit ceux de rang impair. Considérons pour cela l'expression des coefficients de Fourier sous forme complexe :

$$\frac{A_p - iB_p}{2} = C_p = \frac{1}{2L}\int_{-L}^{L} u(x,0)\exp\left(-i\frac{2\pi p}{2L}x\right) \tag{20}$$

La première possibilité est de prolonger la fonction (qui est paire ou impaire) sur l'intervalle $[0, L]$ de la manière suivante :

$$u(L-x,0) = u(x) \tag{21}$$

Le changement de variable $x' = L - x$ permet de simplifier l'intégrale. Pour une fonction paire, on obtient :

$$C_p = \frac{1}{L}(1 + e^{-i\pi p})\int_0^{L/2} u(x,0)\cos\left(\frac{\pi p}{L}x\right)\,dx \tag{22}$$

Pour une fonction impaire :

$$C_p = \frac{1}{L}(1 - e^{-i\pi p})\int_0^{L/2} u(x,0)\sin\left(\frac{\pi p}{L}x\right)\,dx \tag{23}$$

Les coefficients de rang impair sont donc nuls si la fonction est paire. Les coefficients de rang pair sont nuls si la fonction est impaire.

La seconde manière de faire le prolongement est :

$$u(L-x,0) = -u(x) \tag{24}$$

On obtient dans ce cas la règle opposée : les coefficients de rang impair sont nuls si la fonction est impaire ; les coefficients de rang pair sont nuls si la fonction est paire.

On choisira donc l'une ou l'autre règle de prolongement en fonction de la parité de la condition initiale et des termes qui doivent être nuls dans la série de Fourier.

1.d. Algorithme

On choisit tout d'abord :

- ▷ Le type de condition initiale (Neumann ou Dirichlet)
- ▷ La condition initiale sur l'intervalle $[0, L/2]$, et sa parité.

Suivant la parité de la condition initiale et le type de condition limite, on sait quels sont les termes nuls dans la série de Fourier (indices p pairs ou impairs). On prolonge alors la condition initiale sur l'intevalle $[0, L]$ en suivant la règle donnée ci-dessus, puis sur l'intervalle $[-L, L]$ en utilisant sa parité.

Les coefficients de Fourier sont calculés par transformée de Fourier discrète. Il faut pour cela décider d'un indice de troncature P et échantillonner la condition initiale avec $2P+1$ points sur l'intervalle $[-L, L]$. Si la condition initiale présente une ou plusieurs discontinuités, l'indice de troncature doit être pris assez grand pour minimiser le repliement de bande.

Les temps de relaxation des différents modes de Fourier sont calculés. On peut alors calculer $u(x,t)$ sur un échantillonnage de l'intervalle $[-L/2, L/2]$ et pour tout instant, en calculant la somme de Fourier jusqu'au rang P. On peut limiter le calcul de la somme aux modes qui ne sont pas complètement amortis, c'est-à-dire pour lesquels :

$$10\tau_p > t \tag{25}$$

2. Programme python

On pose $L = 1$ et $D = 1$.
fourier.py

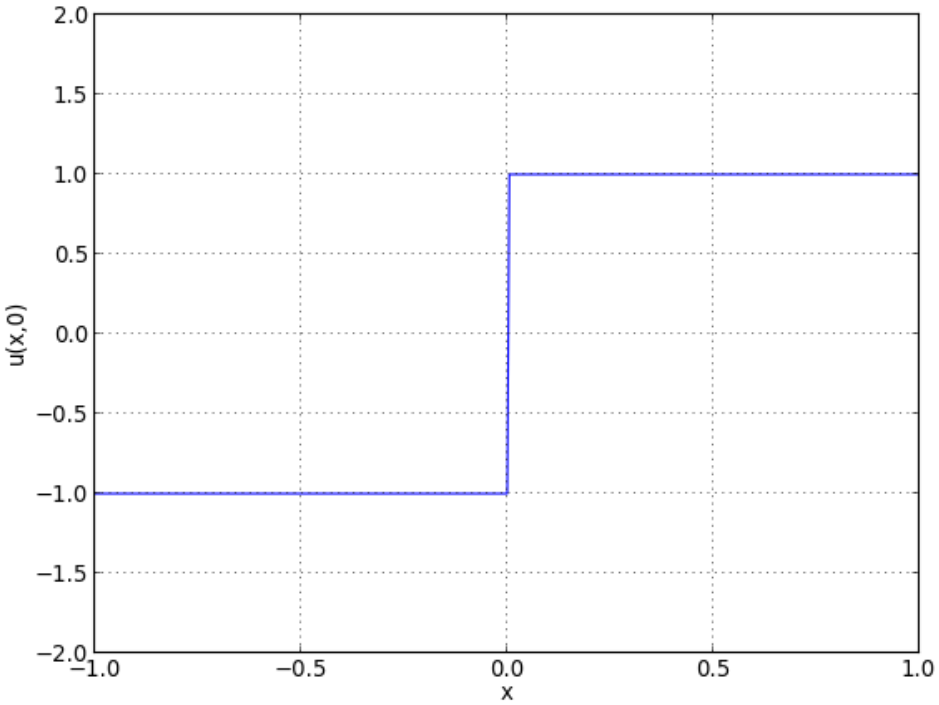
```
import numpy
import numpy.fft

class Fourier:
    def __init__(self,parite,limite,P):
        self.parite = parite
        self.limite = limite
        self.p_impairs = 0
        if self.parite == "impair" and self.limite == "neumann":
            self.p_impairs = 1
        if self.parite == "pair" and self.limite == "dirichlet":
            self.p_impairs = 1
        if self.parite == "pair" and self.p_impairs == 1:
            self.symetrie = -1
        if self.parite == "pair" and self.p_impairs == 0:
            self.symetrie = 1
        if self.parite == "impair" and self.p_impairs == 1:
            self.symetrie = 1
        if self.parite == "impair" and self.p_impairs == 0:
            self.symetrie = -1
        self.P = P
        self.N = 2*P+1
        self.x = numpy.arange(-P,P+1)*1.0/(P)
        self.x1 = self.x + 1.0
        self.u0 = numpy.zeros(self.N)
        self.tau = numpy.zeros(self.P)
        for k in range(self.P):
            self.tau[k] = 1.0/(numpy.pi**2*(k+1)**2)


    def initial(self,f):
        for k in range(0,self.P+1):
            x = self.x[self.P+k]
            if x<=0.5:
                self.u0[self.P+k] = f(x)
                self.u0[self.N-1-k] = self.symetrie*self.u0[self.P+k]
                if self.parite == "pair":
                    self.u0[self.P-k] = self.u0[self.P+k]
                    self.u0[k] = self.u0[self.N-1-k]
                else:
                    self.u0[self.P-k] = -self.u0[self.P+k]
                    self.u0[k] = -self.u0[self.N-1-k]
        self.tfd = numpy.fft.fft(self.u0)/self.N
        self.A = 2*numpy.real(self.tfd)
```

```
        self.B = -2*numpy.imag(self.tfd)


    def valeurs(self,t):
        tau = self.tau[0]
        self.u = numpy.zeros(self.N)
        p = 1
        while tau*10 > t and p<self.P:
            self.u += numpy.exp(-t/tau)*(self.A[p]*numpy.cos(numpy.pi*p*self.x1)+self
            p += 1
            tau = self.tau[p-1]
        return self.u
```

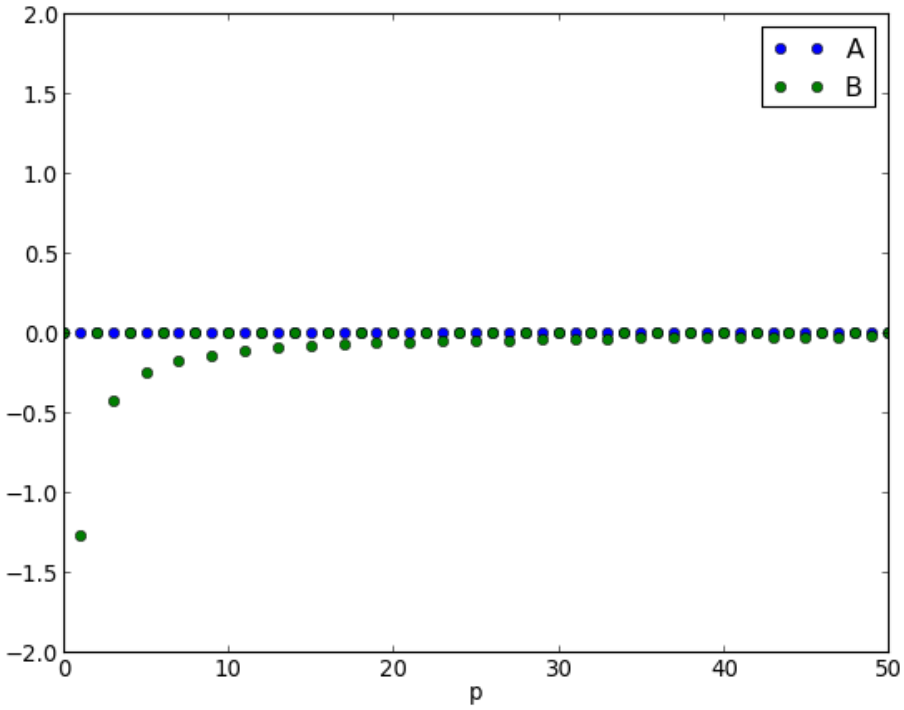
3. Exemples

3.a. Fonction impaire avec conditions de Neumann

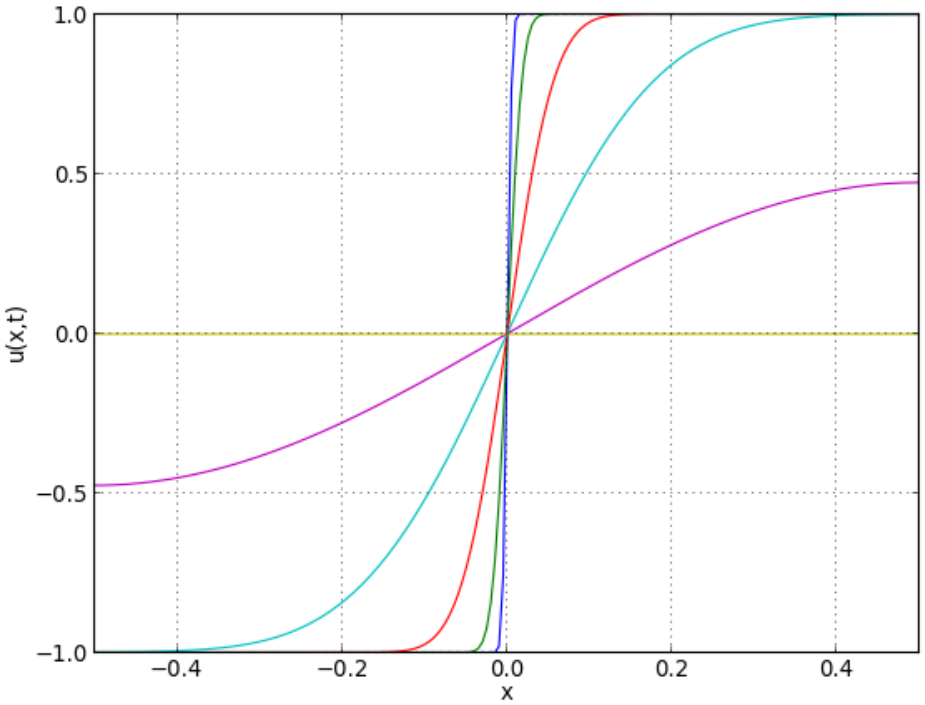
```
from fourier import Fourier
import numpy
from matplotlib.pyplot import *

fourier = Fourier("impair","neumann",200)
def f(x):
    return 1.0
fourier.initial(f)

figure()
plot(fourier.x,fourier.u0)
xlabel('x')
ylabel('u(x,0)')
axis([-1.0,1.0,-2,2])
grid()
```




```
figure()
plot(fourier.A,'o',label='A')
plot(fourier.B,'o',label='B')
xlabel('p')
axis([0,50,-2,2])
legend(loc='upper right')
```

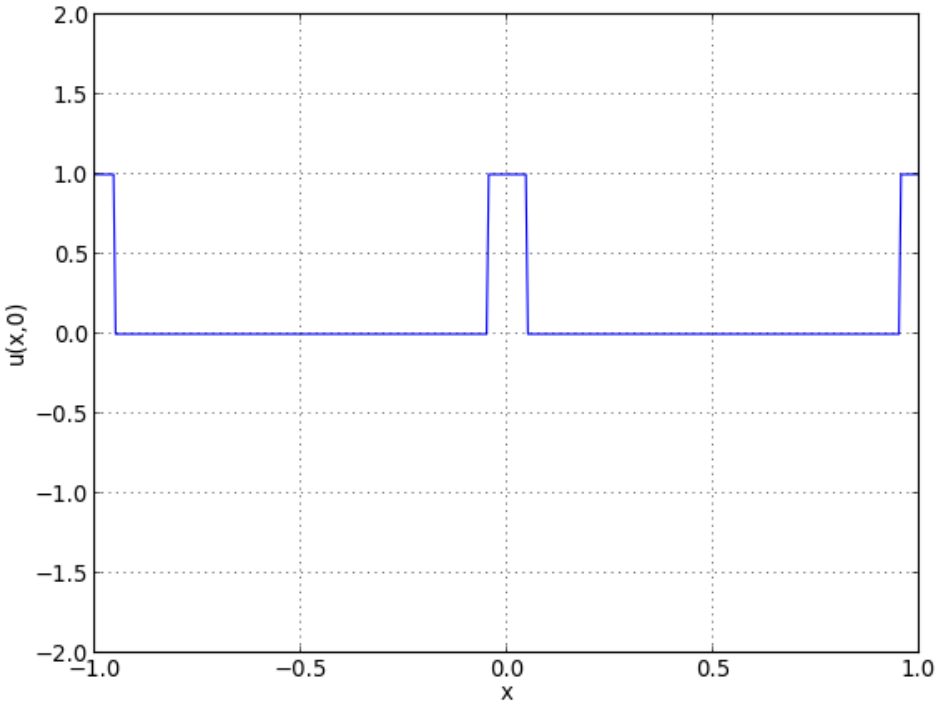



```
u0 = fourier.valeurs(0.00001)
u1 = fourier.valeurs(0.0001)
u2 = fourier.valeurs(0.001)
u3 = fourier.valeurs(0.01)
u4 = fourier.valeurs(0.1)
u5 = fourier.valeurs(1.0)

figure()
plot(fourier.x,u0)
plot(fourier.x,u1)
plot(fourier.x,u2)
plot(fourier.x,u3)
plot(fourier.x,u4)
plot(fourier.x,u5)
xlabel('x')
ylabel('u(x,t)')
axis([-0.5,0.5,-1,1])
grid()
```

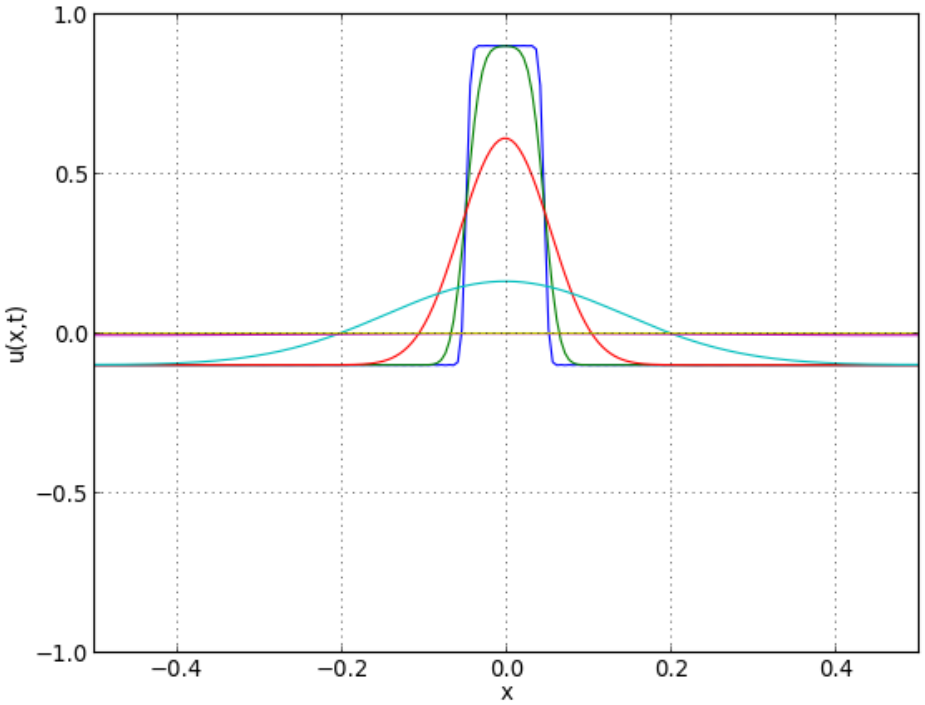


3.b. Fonction paire avec conditions de Neumann

```
fourier = Fourier("pair","neumann",200)
def f(x):
    if x<0.05:
        return 1.0
    else:
        return 0.0
fourier.initial(f)
figure()
plot(fourier.x,fourier.u0)
xlabel('x')
ylabel('u(x,0)')
axis([-1.0,1.0,-2,2])
grid()
```

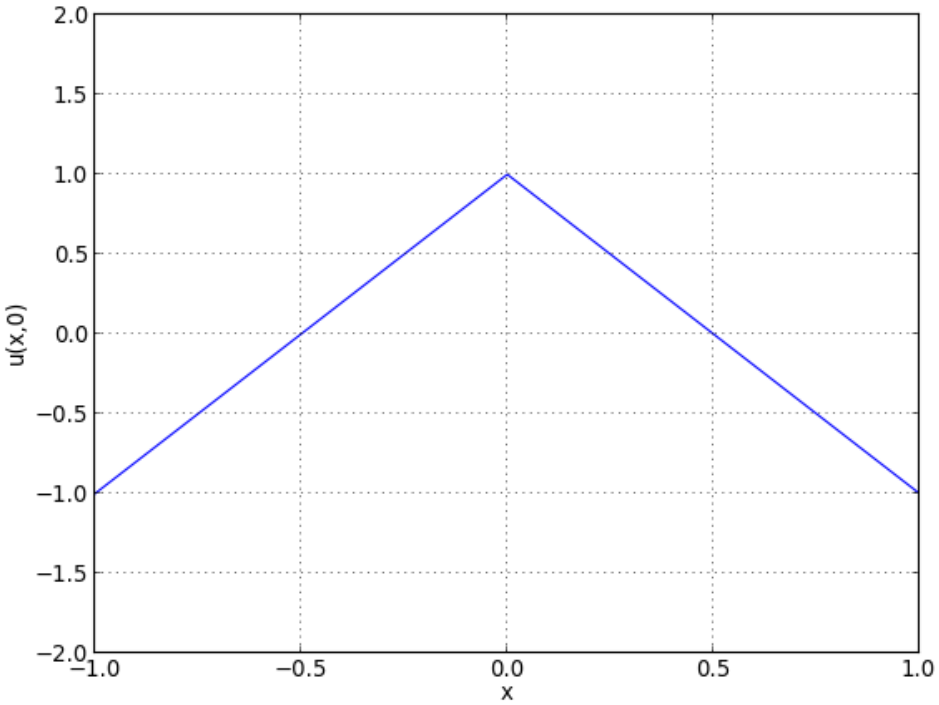



```
u0 = fourier.valeurs(0.00001)
u1 = fourier.valeurs(0.0001)
u2 = fourier.valeurs(0.001)
u3 = fourier.valeurs(0.01)
u4 = fourier.valeurs(0.1)
u5 = fourier.valeurs(1.0)

figure()
plot(fourier.x,u0)
plot(fourier.x,u1)
plot(fourier.x,u2)
plot(fourier.x,u3)
plot(fourier.x,u4)
plot(fourier.x,u5)
xlabel('x')
ylabel('u(x,t)')
axis([-0.5,0.5,-1,1])
grid()
```

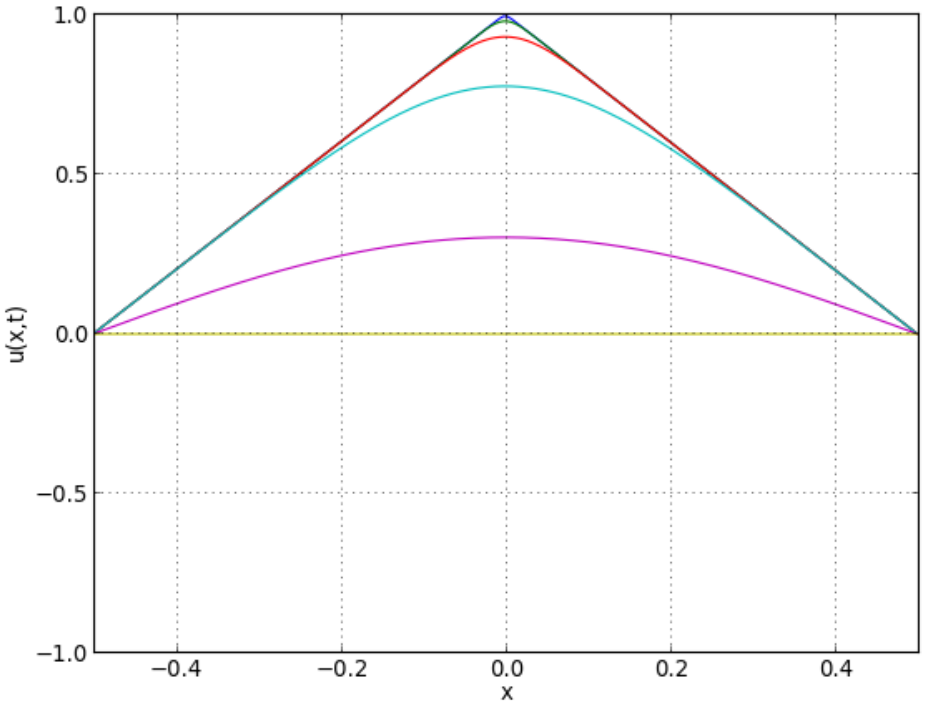


3.c. Fonction paire avec condition de Dirichlet

```
fourier = Fourier("pair","dirichlet",200)
def f(x):
    return 1.0-2.0*x
fourier.initial(f)

figure()
plot(fourier.x,fourier.u0)
xlabel('x')
ylabel('u(x,0)')
axis([-1.0,1.0,-2,2])
grid()
```

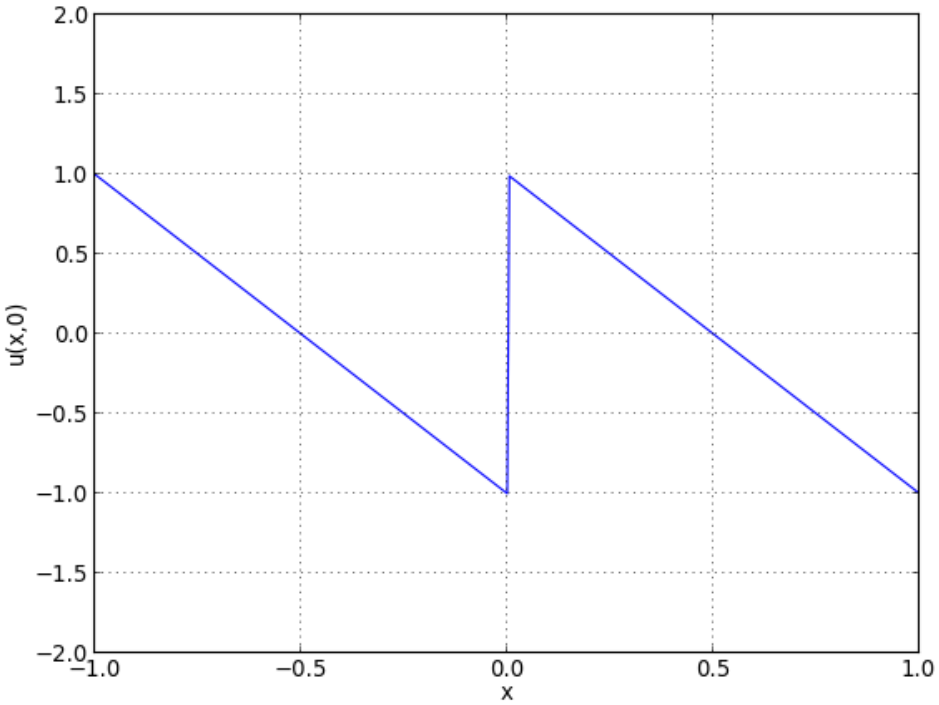



```
u0 = fourier.valeurs(0.00001)
u1 = fourier.valeurs(0.0001)
u2 = fourier.valeurs(0.001)
u3 = fourier.valeurs(0.01)
u4 = fourier.valeurs(0.1)
u5 = fourier.valeurs(1.0)

figure()
plot(fourier.x,u0)
plot(fourier.x,u1)
plot(fourier.x,u2)
plot(fourier.x,u3)
plot(fourier.x,u4)
plot(fourier.x,u5)
xlabel('x')
ylabel('u(x,t)')
axis([-0.5,0.5,-1,1])
grid()
```

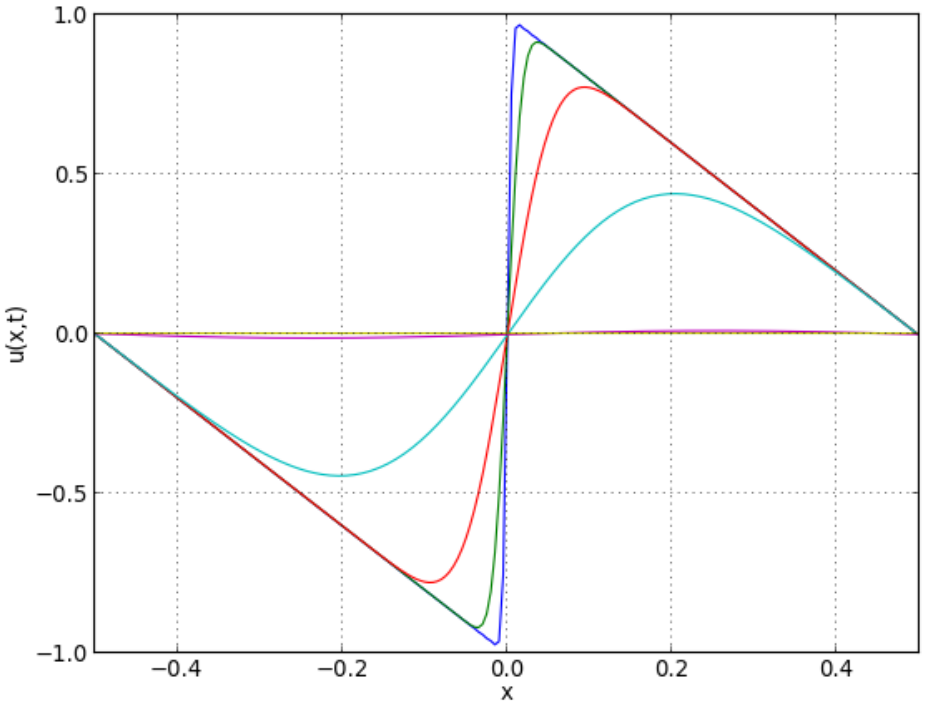


3.d. Fonction impaire avec condition de Dirichlet

```
fourier = Fourier("impair","dirichlet",200)
def f(x):
    return 1.0-2.0*x
fourier.initial(f)

figure()
plot(fourier.x,fourier.u0)
xlabel('x')
ylabel('u(x,0)')
axis([-1.0,1.0,-2,2])
grid()
```




```
u0 = fourier.valeurs(0.00001)
u1 = fourier.valeurs(0.0001)
u2 = fourier.valeurs(0.001)
u3 = fourier.valeurs(0.01)
u4 = fourier.valeurs(0.1)
u5 = fourier.valeurs(1.0)

figure()
plot(fourier.x,u0)
plot(fourier.x,u1)
plot(fourier.x,u2)
plot(fourier.x,u3)
plot(fourier.x,u4)
plot(fourier.x,u5)
xlabel('x')
ylabel('u(x,t)')
axis([-0.5,0.5,-1,1])
grid()
```


1.0
0.5
0.0
−0.5
−1.0
u(x,t)
−0.4
−0.2
0.0
0.2
0.4
x