Projet 5, Fourier, produit de convultion et barre infinie

Zouiche Omar et David Czarnecki

8 Mars 2021

Equation de la chaleur, tranformée de Fourier et produit de convultion ***Projet 5*** Zouiche Omar et David Czarnecki

Nous reprenons le même problème de la semaine 4 mais avec une barre infinie et on l'approche avec la transformée de Fourier de la fonction u (représentant la chaleur en un point x à la période t) et en utilisant le produit de convulion de deux fonctions.

Problème : On considère le modèle de la barre infinie donnée par le modèle : \$ $\forall x \in \mathbb{R} \quad \forall t \in [0, T]$\$: $\begin{cases} \frac{\partial u}{\partial u}(x,t) = k\frac{\partial^2 u}{\partial x^2}(x,t) \\ u(x,0) = u_0(x) \end{cases}$ La transformée de Fourier de u est donnée par :

$$F(v,t) = \int_{-\infty}^{+\infty} u(x,t)e^{2ix}dx$$

Sachant qu'avec une propriété de la transformée de Fourier qui est \$F(g*f) = F(g)F(f) \$, avec $g * f$ le produit de convultion de deux fonction g et f, l'expression de u devient :

$$u(x,t) = \frac{1}{2\sqrt{\pi t}} \int_{-\infty}^{\infty} u_0(y)e^{\frac{-(x-y)^2}{4t}} u_0(y)dy$$

L'équation devient :

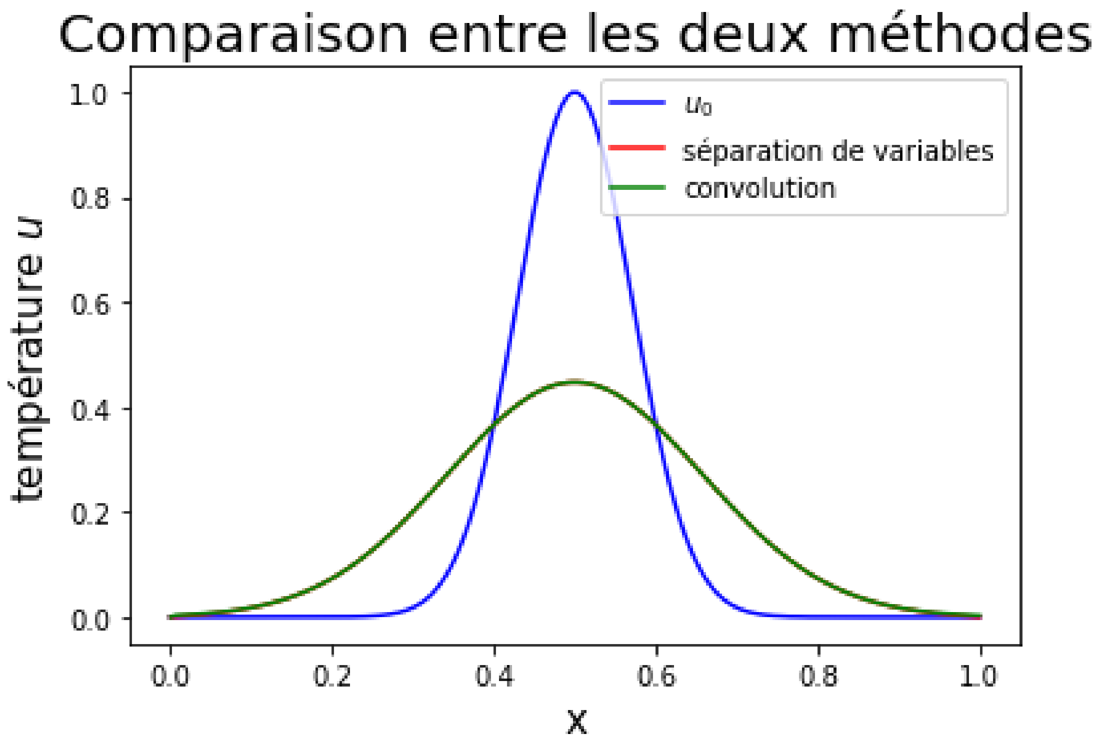
$$\frac{\partial F}{\partial t}(v,t) + 4\pi^2 v^2 F(v,t) = 0$$

Nous allons modifier le code chaleur1dspec.py avec les conditions suivantes :

[24]:
```
from math import sin,sqrt,exp,pi
import numpy as np
import matplotlib.pyplot as plt

#Initialisation
k = 1
s= 100
Lx = 1
Nx = 150 #le maillage spatial sert a la representation de la solution et aux␣
↪calculs des ps par integration numerique
hx = Lx/(Nx-1)
x = np.linspace(0,Lx,Nx)
modmax = 35

f=np.zeros(len(x))
```

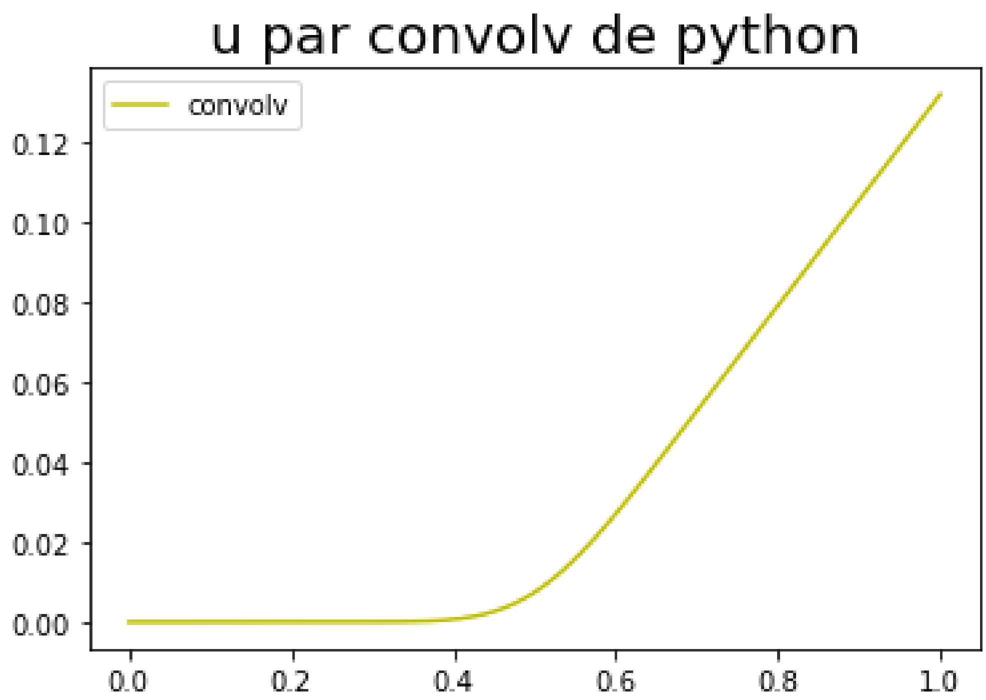
```
u0=np.zeros(len(x))
uc = np.zeros(len(x))
u=np.zeros(len(x))
testx=np.zeros(modmax)
fbx=np.zeros((modmax,Nx))       #phi_n
fmp=np.zeros(modmax)            #projection sur phi_n de f
imp=np.zeros(modmax)            #projection sur phi_n de u0
#rhs
#Coeur du programme

for i in range(1,Nx-1):
    f[i]= 0   #30*exp(-s*((x[i]-Lx/4)**2))  #(100*(x[i]**2)*((Lx-x[i])**2))/
 ↪(Lx**4)
    u0[i]=exp(-s*((x[i]-Lx/2)**2)) #+exp(-2*s*((x[i]-Lx/
 ↪3)**2))+exp(-3*s*((x[i]-2*Lx/3)**2))
    u[i]=u0[i]

plt.plot(x,u0,'b',label = '$u_0$')

#Séparation de variable
for m in range(1,modmax):
    for i in range(1,Nx):
        fbx[m][i]=sin(pi*x[i]/Lx*m)                #phi_n
        testx[m]=testx[m]+fbx[m][i]*fbx[m][i]*hx
    testx[m]=sqrt(testx[m])                        #norme L2 de phi_n
    for i in range(1,Nx):                          #normalisation
        fbx[m][i]=fbx[m][i]/testx[m]

for m in range(1,modmax):
    for i in range(1,Nx):
        fmp[m]+=f[i]*fbx[m][i]*hx     # <f,\phi_n> = f_n
        imp[m]+=u0[i]*fbx[m][i]*hx    # <u0,phi_n> = c_n

#somme serie

temps=0.0     #on doit retrouver la condition initiale
for i in range(1,Nx):
    u[i]=0
for m in range(1,modmax):
    al=(m**2)*(pi**2)/(Lx**2)*k
    coef=imp[m]*exp(-al*temps)
    for i in range(0,Nx-1):
         u[i]+=fbx[m][i]*coef

#plt.plot(x,u,'blue')
```

```
temps=0.01 #la solution a n'importe quel temps sans avoir a calculer les iter␣
 ↪intermediaires
for i in range(1,Nx):
    u[i]=0
for m in range(1,modmax):
    al=(m**2)*(pi**2)/(Lx**2)*k
    coef=imp[m]*exp(-al*temps)
    coeff=fmp[m]*(1-exp(-al*temps))/al
    for i in range(0,Nx):
        u[i]+=fbx[m][i]*(coeff+coef)

# calcul de la convolution
for i in range(1,len(x)):
    uc[i]=0
    z=x[i]

    for j in range(1,len(x)):
        y=x[j]
        uc[i] = uc[i] + exp(-(z-y)**2/(4*k*temps))*u0[j]
    uc[i]= hx * uc[i]/(sqrt(4*temps*k*pi))

plt.plot(x,u,'r',label = 'séparation de variables')
plt.plot(x,uc,'g',label = 'convolution')
plt.xlabel('x',fontsize = 15)
plt.ylabel('température $u$',fontsize = 15)
plt.legend()
plt.title('Comparaison entre les deux méthodes', fontsize = 20)
plt.show()
```



[27]:
```
Convu=np.convolve(u0, x,'full')
plt.plot(x,Convu[:Nx]/(100),'y',label='convolv')
plt.legend()
plt.title("u par convolv de python", fontsize=20)
```

[27]: Text(0.5, 1.0, 'u par convolv de python')



Le produit de convultion et la transformée de Fourier nous on permis de résoudre l'équation de la chaleur de la barre, cette fois infinie avec le code chaleur1dspec.py