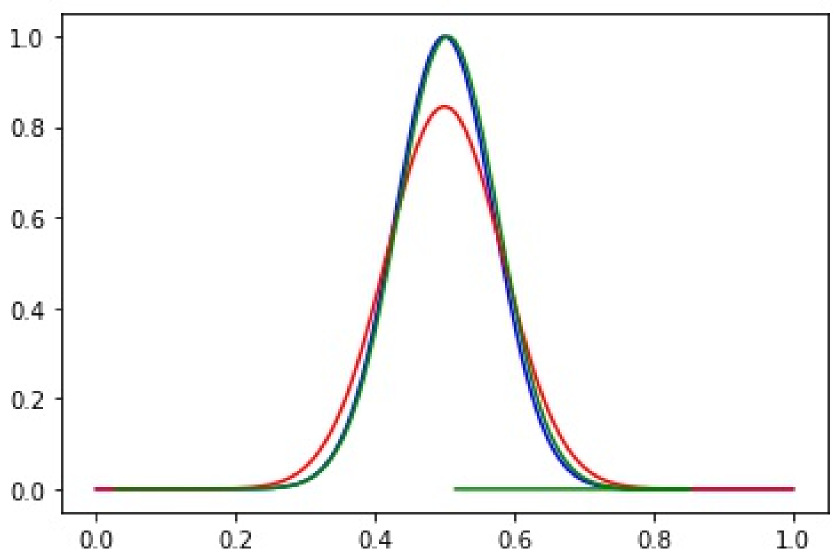
Sur la convolution et son implementation dans la résolution de l'équation de chaleur

Gaggini Lorenzo et Ferreti Thibault

Dans ce rapport, on s'interresse a l'équation de chaleur et ça résolution aux condition initales, que l'on cherche à comparé a la convolution(methode de fourrier), et à la solution donnée par `chalex1dspec` (methode des differences finis)

En modifant adaptant `chalex1dspec`, et en y ajoutant quelques modules, on obtien un algorithme (page 2) nous permettant de visualisé le graphe suivant :



En bleu, la solution la solution initial, en vert la convolution(methode de fourrier), en rouge la solution chalex1dspec(differences finies)

On conclu que ces methodes offres des résultats similaires.

Algorithme 1

```
#Modules ÃČÆŠÃĆ  importer
from math import sin,sqrt,exp,pi
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from scipy.fft import fft, ifft
import scipy.integrate as integrate
#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
#introduire une boucle sur modmax et calculer l'erreur ||u(modmax)||
f=np.zeros(len(x))
u0=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):
    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]

#fb normalise
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]
#verifier l'orthonormalite des fbx ?
#<fbx[m],fbx[n]>=delta_mn
#projection f second membre et u0 condi init sur fbx
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=0.001
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)
plt.plot(x,u,'r')
#convolution python
f1=x
f2=u0
#shift
uconv=np.convolve(f1,f2,'same')/26
plt.plot(uconv,u0,'g')
plt.show()
```