Sur le phénomène de déformation et son lien avec l'équation de chaleur

Dans ce rapport, on s'interresse au problème de déformation d'un toit de hangar,

On a un toit accroché sur deux de ses extrémités qui est soumis à une force F. La déformation du toit $u(x, y)$ vérifie l'équation :

$-\operatorname{grad}(u) = F$

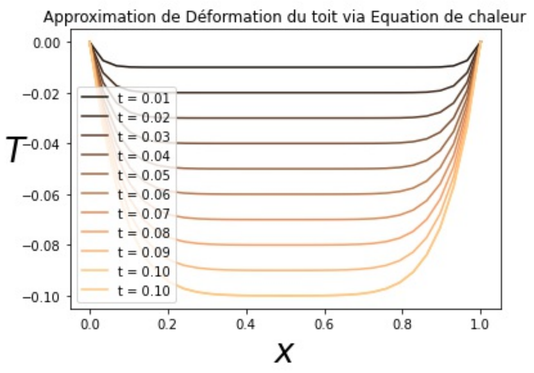
Dans un exemple particulier, on a vu que certain cas peuvent étre résolu imédiatement en faisant appel aux outils vue dans le rapport#1

Mais ici, On cherche une méthode d'approximation pour résoudre cette équation pour tout u et f

Pour ce faire, on considère ce problème comme stationnaire et avec une dérivée en temps qui vaut 0.

En effet, on sais au préalable qu'une équation de chaleur avec une dérivée en temps qui vaut 0 vérifie l'équation d'élasticité

On se sert alors du code heat1d qui modélise la propagation de la chaleur et le modifis légèrement pour obtenir le graphique ci-aprés souhaiter.



Algorithme heat1d.py adapté pour la $d$éformation

```
import numpy as np
import matplotlib.pyplot as plt


#u, t=ku, xx

# PHYSICAL PARAMETERS
K=0.1  #Diffusion coefficient
L=1.0  #Domain size
Time=0.1 #Integration time


# NUMERICAL PARAMETERS
NX=30 #Number of grid points
NT=1000 #Number of time steps
ifre=100
eps=0.001

dx=L/(NX-1) #Grid step (space)
dt=Time/NT  #Grid step (time)
print(NT, dt)

### MAIN PROGRAM ###

# Initialisation
x=np.linspace(0.0, 1.0, NX)
T=np.zeros(NX) #np.sin(2 np.pi x)
F=np.ones(NX)
rest=[]
RHS=np.zeros(NX)
plt.figure()

# Main loop en temps
#for n in range(0, NT):
n=0
res=1
res0=1
while(n<NT and res/res0>eps):

 n+=1
#discretization of the second order derivative (Laplacian)
 res=0
 for j in range (1, NX-1):
   RHS[j]=dt (K (T[j-1]-2 T[j]+T[j+1])/(dx 2)+
F[j])
   res+=abs(RHS[j])
#   T[j]+=RHS[j]
 for j in range (1, NX-1):
   T[j]+=RHS[j]
   RHS[j]=0

 if (n==1):
   res0=res

 rest.append(res)
#Plot every ifre time steps
 if (n%ifre==0 or (res/res0)<eps):
   print(n, res)
   plotlabel="t=%1.2f" %(n dt)
   plt.plot(x, -T, label=plotlabel, color=plt.get_cmap('
copper')(float(n)/NT))



print(n, res)
plotlabel="t=%1.2f" %(n dt)
plt.plot(x, -T, label = plotlabel, color = plt.get_cmap('
copper')(float(n)/NT))

plt.xlabel(u'$x$', fontsize=26)
plt.ylabel(u'$T$', fontsize=26, rotation=0)
plt.title(u'Déformation du toit via Equation de chaleur')
plt.legend()

plt.figure(2)
plt.show()
```