# Prévention des accidents dus à la fatigue du métal dans le transport aérien

Numéro d'inscription:
14209

TIPE 2018-2019

TRANSPORT

# Problématique

-Comment se propagent les fissures dans un matériau conducteur ?

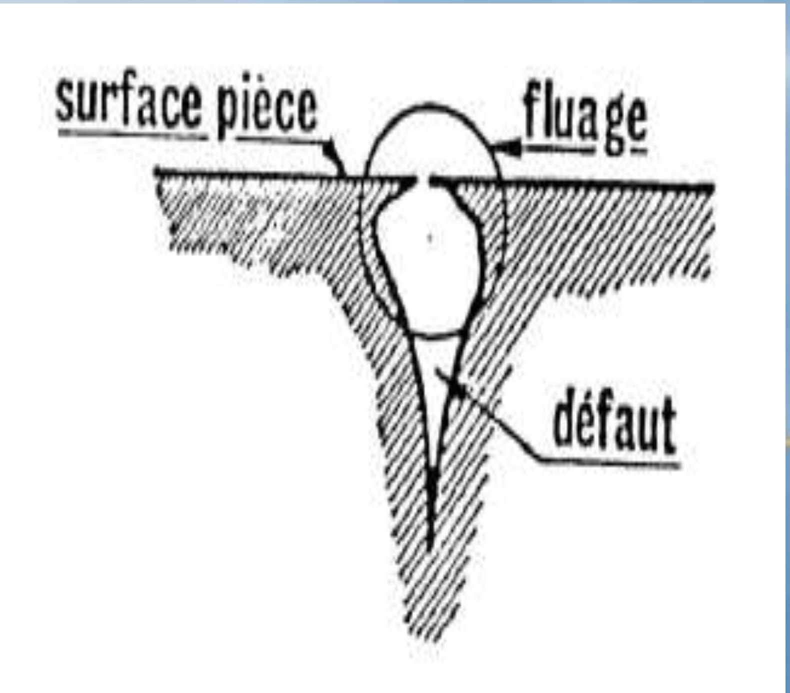
- Comment peut-on les détecter grâce au contrôle par courant de Foucault ?

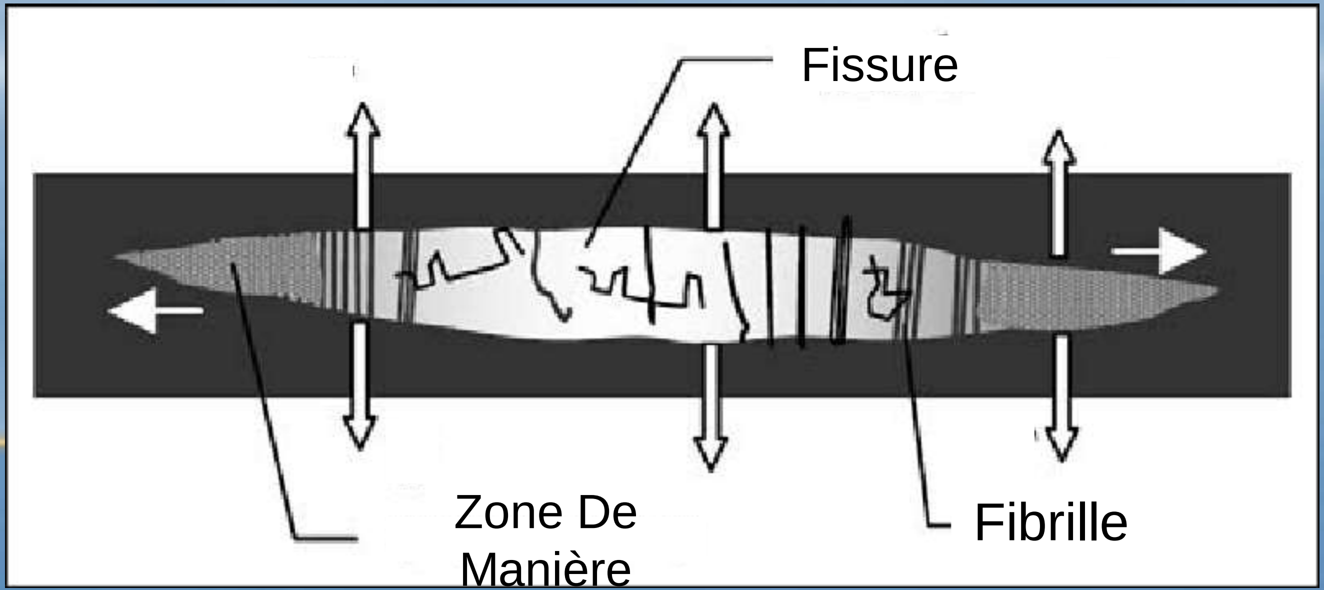
# Plan

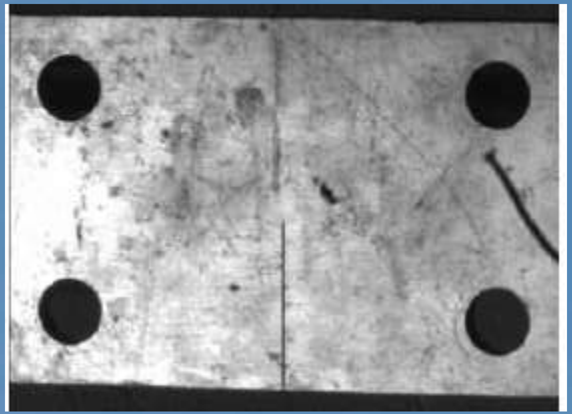
# Contributions

- Prise de contact avec des usines et laboratoires en Tunisie
- Rencontre avec le physicien et le chercheur ATEF BOULILA, travaillant sur la propagation des fissures et le physicien EDOUARD TANTART
- Implémentation de codes python traduisant la diffusion de la fissure dans le métal
- Implémentation d’un code python permettant de visualiser la résolution analytique de l’équation de la chaleur par transformée de Fourier.

# Introduction



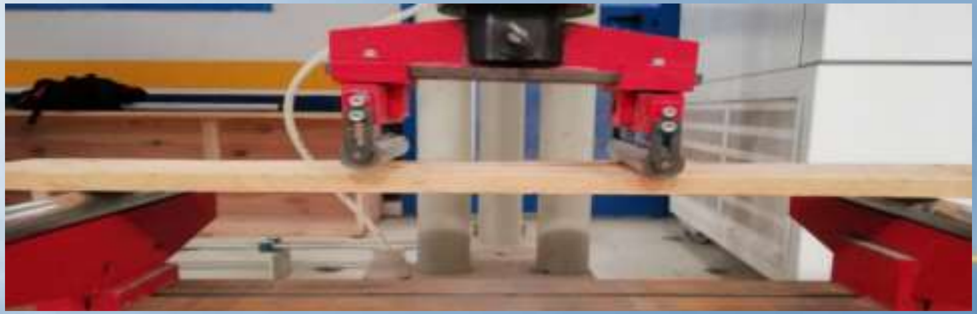



# Principe de la propagation des fissures

1ère expérience :

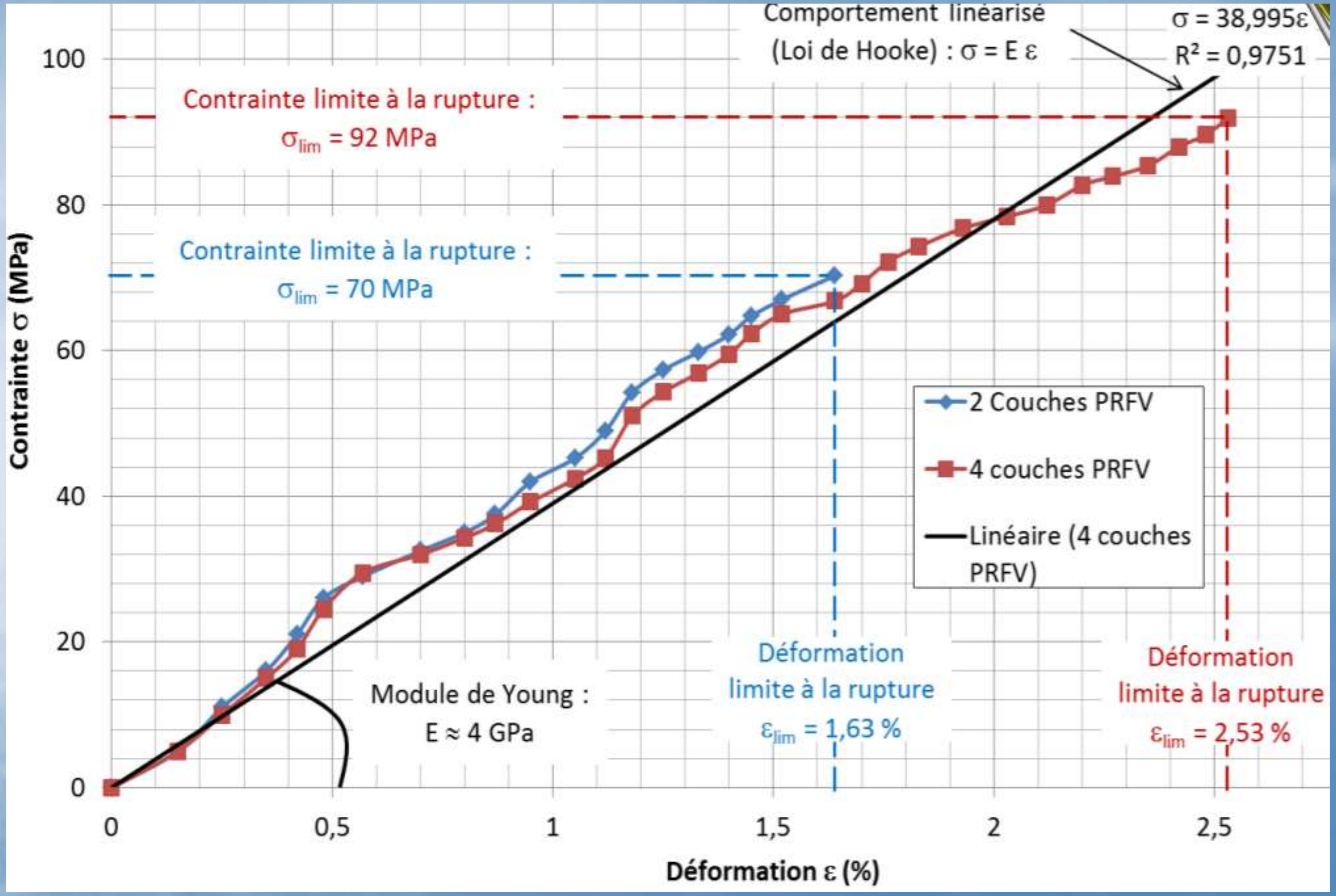
* Poutre Homogène




# * Poutre Composite


ZOOM

Remarque: effort exercée plus important




On peut remarquer que dans des milieux différents on obtient une différence dans le mode de transport des fissures , d’où la nécessité d’une étude fine

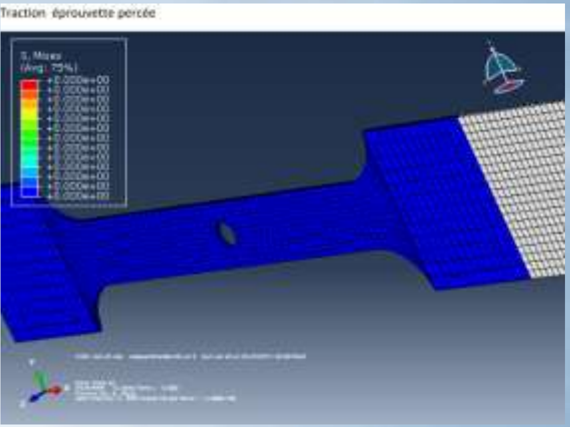
# 2ème expérience: une force de traction appliquée à une éprouvette métalliques percées

Des éprouvettes métalliques percées

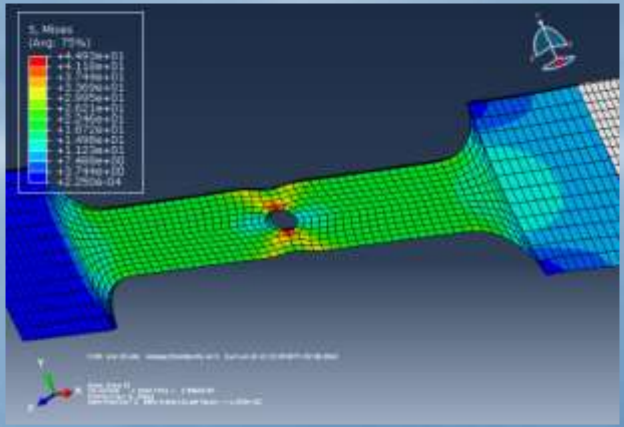

Machine de traction

# Observation:

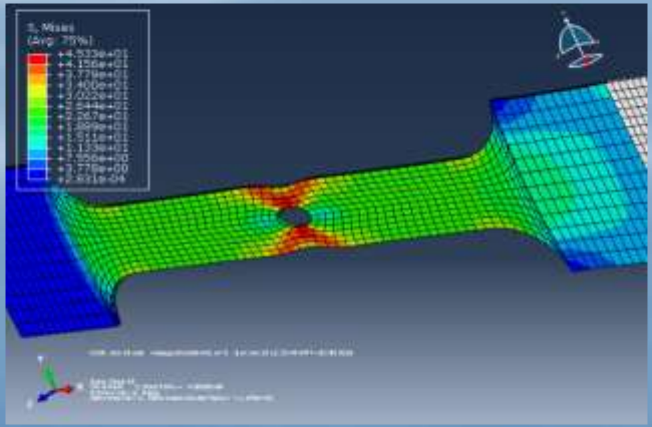
## Comportement du matériau pendant l'essai de traction



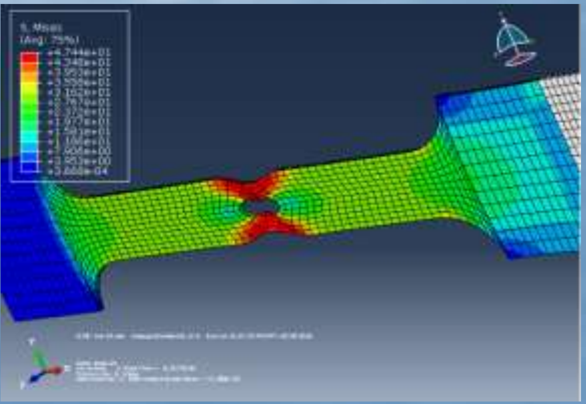

État initial



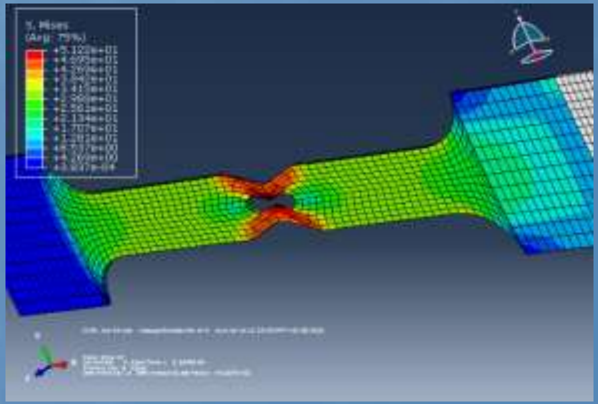
Zone élastique



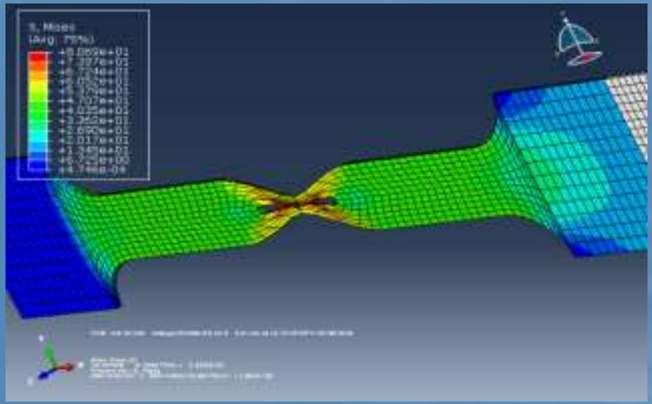
Début de la zone plastique



Diffusion de la chaleur

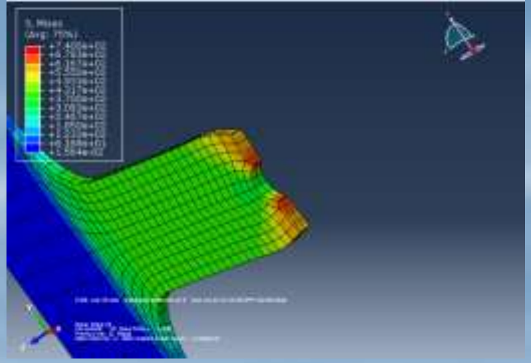


Apparition des fissures



État final

# Modélisation



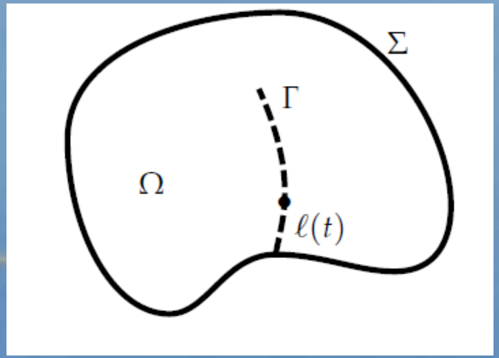
## Hypothèse

* Milieu solide : énergie potentielle

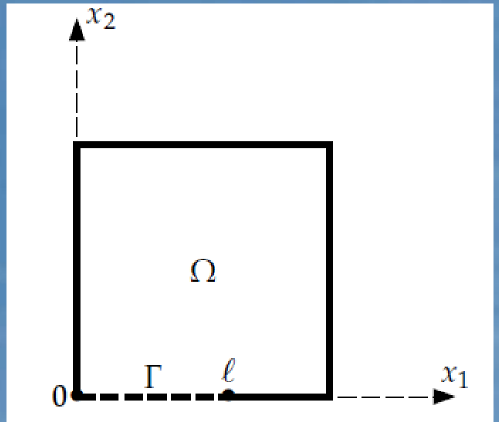
** la dissipation d'énergie d'une fissure en propagation

***Ω= ]0, 1[x] − 1, 1[, avec une fissure le long de l'axe y = 0 $\Gamma=[0,\ell]\times\{0\}$

et x = (x1, x2)




$$\begin{cases} -\Delta u(x) = f(x) & dans\ \ \Omega \\ u(x) = 0 & \text{sur } \partial\Omega \,/\, \Gamma \\ \dfrac{\partial u(x)}{\partial x_2} = 0 & \text{sur } \Gamma \end{cases}$$



# Résolution analytique

Equation de Griffith dans l'espace réel $\xrightarrow{\text{T.F}}$ Equation de Griffith dans l'espace de phase

↓

Solution réelle $\xleftarrow{\text{T.F.I}}$ Solution dans l'espace de phase

* Il suffit de fixer les conditions initiales sur le déplacement et sa dérivée première pour atteindre des solutions dans l'espace de phase

* Pour des conditions initiales et aux limites quelconques de la forme :

$$\begin{cases} u(0,t)=0 & \forall t \\ u(\infty,t) = 0 & \forall t \\ u(x,0) = e^{-i\omega t\frac{x^2}{2\sigma}} & \forall x \end{cases}$$

## * Cas de: f(x) = $\frac{\partial u(x,t)}{\partial t}$

$$\boxed{-\Delta \boldsymbol{u}(\boldsymbol{x}, \boldsymbol{t}) = \frac{\partial \boldsymbol{u}(\boldsymbol{x},\boldsymbol{t})}{\partial \boldsymbol{t}}} \quad \textbf{(*)}$$

On résout l'équation de la chaleur en considérant les paramètres constants et un déplacement unidimensionnelle

On applique à l'équation (*) une transformée de Fourier (T.F) et son inverse avec :

$\hat{h}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} h(x) e^{-ikx} dx$

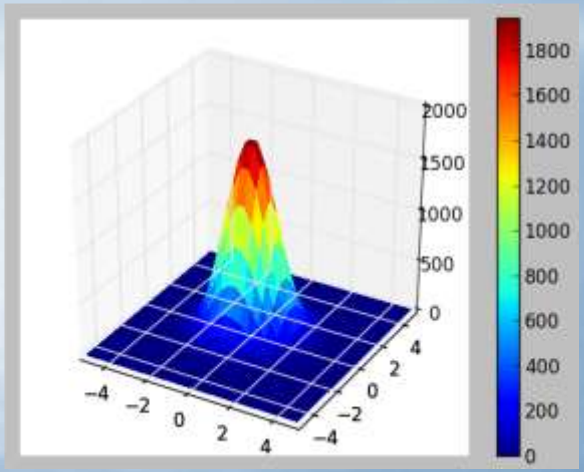
Avec pour condition initiale une impulsion de Dirac u(x,0)=f(x)

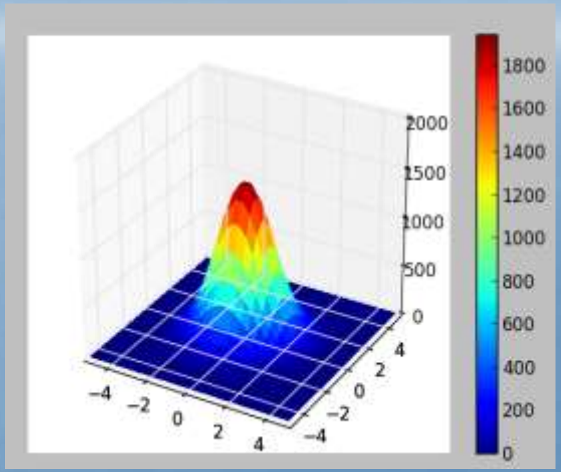
$$\frac{\partial u}{\partial t} = -\frac{\partial^2 u}{\partial x^2} \Rightarrow \tilde{u}(k,t) = \frac{1}{\sqrt{2\pi}} e^{-k^2 t} \Rightarrow u(x,t) = \frac{1}{2\sqrt{\pi t}} e^{\frac{-x^2}{4t}}$$

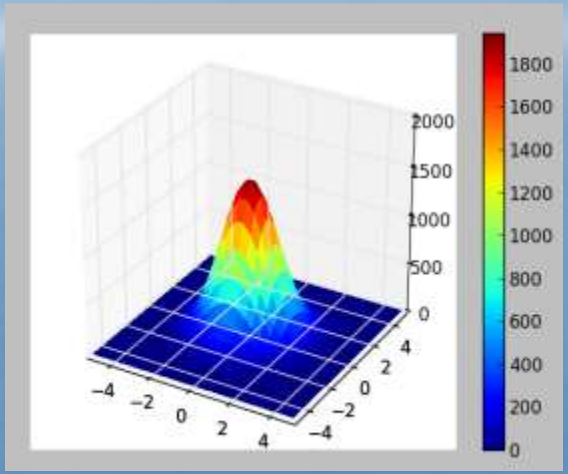
Ne représente pas vraiment un phénomène physique , on utilise alors une gaussienne comme condition initiale:

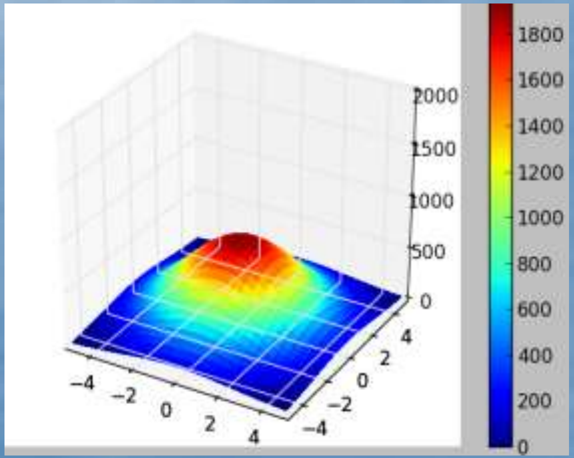
$$\tilde{u}(k,t) = \sqrt{\sigma} e^{-\frac{k^2 \sigma}{2}} e^{-k^2 t} \Rightarrow \boxed{u(x,t) = \frac{\sqrt{2\sigma}}{\sqrt{4t+2\sigma}} e^{\frac{-x^2}{4t+2\sigma}}}$$

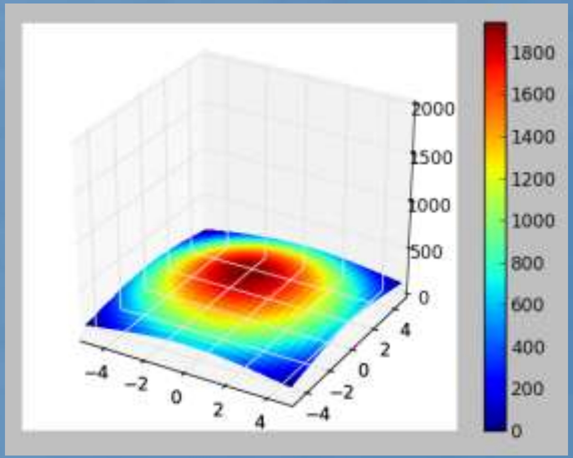
# Visualisation de la solution avec Python




➡ Diffusion de l’énergie dans la zone de la fissure
Provoquant une fissuration

➡ Phénomène de transport d’énergie

# Résolution numérique

On résout l'équation de la chaleur à 2 dimensions en utilisant la méthode des différences finies.

On définit d'abord les pas $x_i$ , $y_j$ $et$ $t_k$: $x_i = x_0$+i$\Delta x$, $y_j$= $y_0$+j$\Delta y$, $t_k$= $t_0$+k$\Delta t$

On détermine ensuite $\frac{\partial u}{\partial t}$, $\frac{\partial^2 u}{\partial x^2}$ et $\frac{\partial^2 u}{\partial y^2}$ en appliquant Taylor respectivement à l'ordre 1 en $t_k$ et à l'ordre 2 en $x_i$ et $y_j$. On pose $u_{i,j}^{k}$= u(x,y,k)

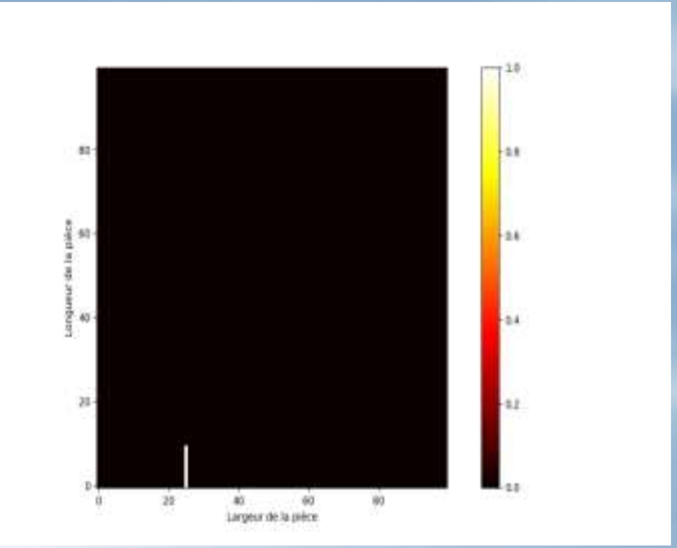
$$u(x_i+\Delta x, y_j, t_k) = u(x_i) + \Delta x \frac{\partial u}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2} + O(\Delta x^3) \Rightarrow \boxed{\frac{\partial^2 u}{\partial x^2} \cong \frac{u_{i+1,j}^{k}+u_{i-1,j}^{k}-2u_{i,j}^{k}}{\Delta x^2}}$$

$$u(x_i-\Delta x, y_j, t_k) = u(x_i) - \Delta x \frac{\partial u}{\partial x} + + \frac{\Delta x^2}{2}\frac{\partial^2 u}{\partial x^2} + O(\Delta x^3) \Rightarrow \boxed{\frac{\partial^2 u}{\partial y^2} \cong \frac{u_{i,j+1}^{k}+u_{i,j-1}^{k}-2u_{i,j}^{k}}{\Delta y^2}}$$

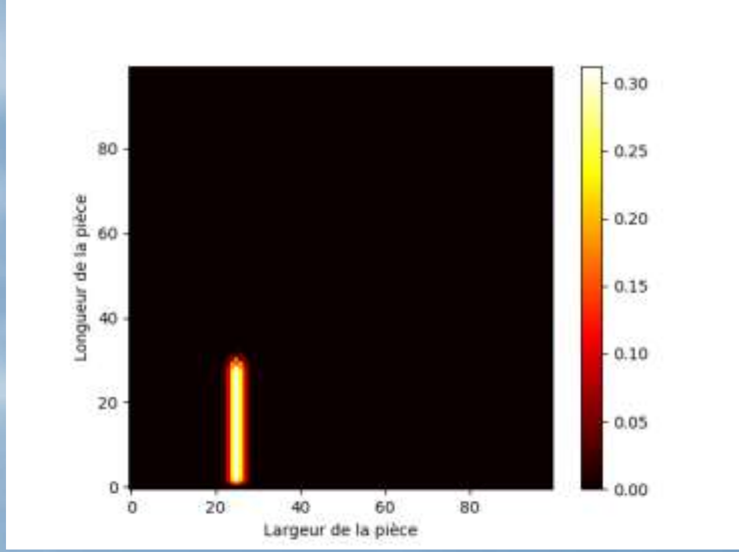
$$u(x_i, y_j, t_k+\Delta t) = u(x_i, y_j, t_k) + \Delta t \frac{\partial u}{\partial t} + O(\Delta t^2) \Rightarrow \boxed{\frac{\partial u}{\partial t} \cong \frac{u_{i,j}^{k+1}-u_{i,j}^{k}}{\Delta t}}$$

On remplace ensuite dans l'équation (*)

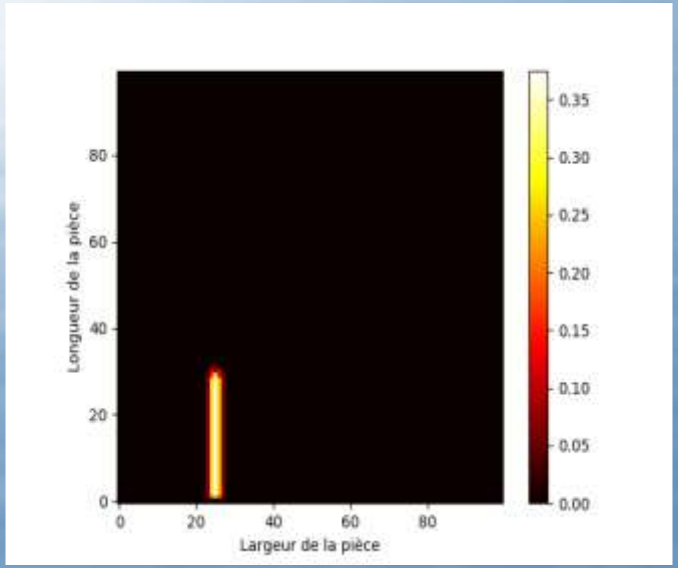
$$\frac{u_{i,j}^{k+1}-u_{i,j}^{k}}{\Delta t} = \frac{u_{i+1,j}^{k}+u_{i-1,j}^{k}-2u_{i,j}^{k}}{\Delta x^2} + \frac{u_{i,j+1}^{k}+u_{i,j-1}^{k}-2u_{i,j}^{k}}{\Delta y^2} \Rightarrow \boxed{u_{i,j}^{k+1} = u_{i,j}^{k}+\Delta t\left(\frac{u_{i+1,j}^{k}+u_{i-1,j}^{k}-2u_{i,j}^{k}}{\Delta x^2} + \frac{u_{i,j+1}^{k}+u_{i,j-1}^{k}-2u_{i,j}^{k}}{\Delta y^2}\right)}$$



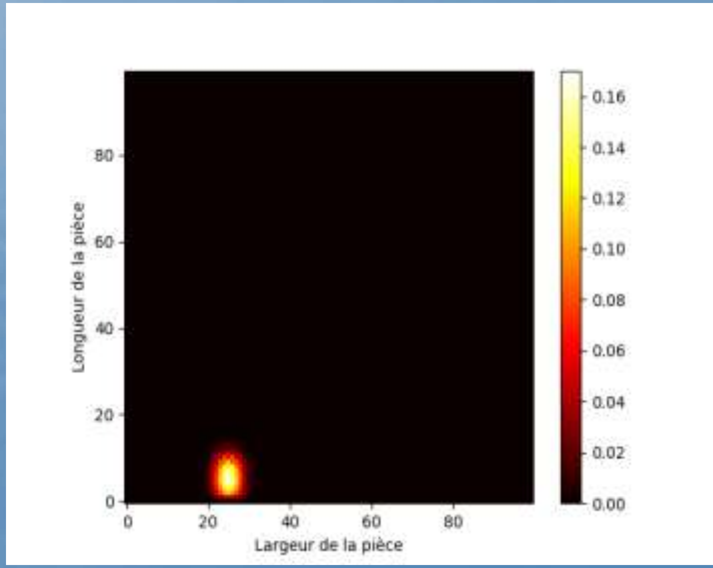














## Observation:

La simulation numérique n'est pas fidèle à la réalité physique.
Elle reproduit la diffusion de l'énergie mais ne représente pas la propagation des fissures d'où la nécessité d'un modèle plus élaboré

# Principe du contrôle non destructif

**Emetteur**

- caractériser l’état d’une pièce à contrôler
- Excitation

**Pièce à contrôler**

- Perturbation de la pièce à contrôler

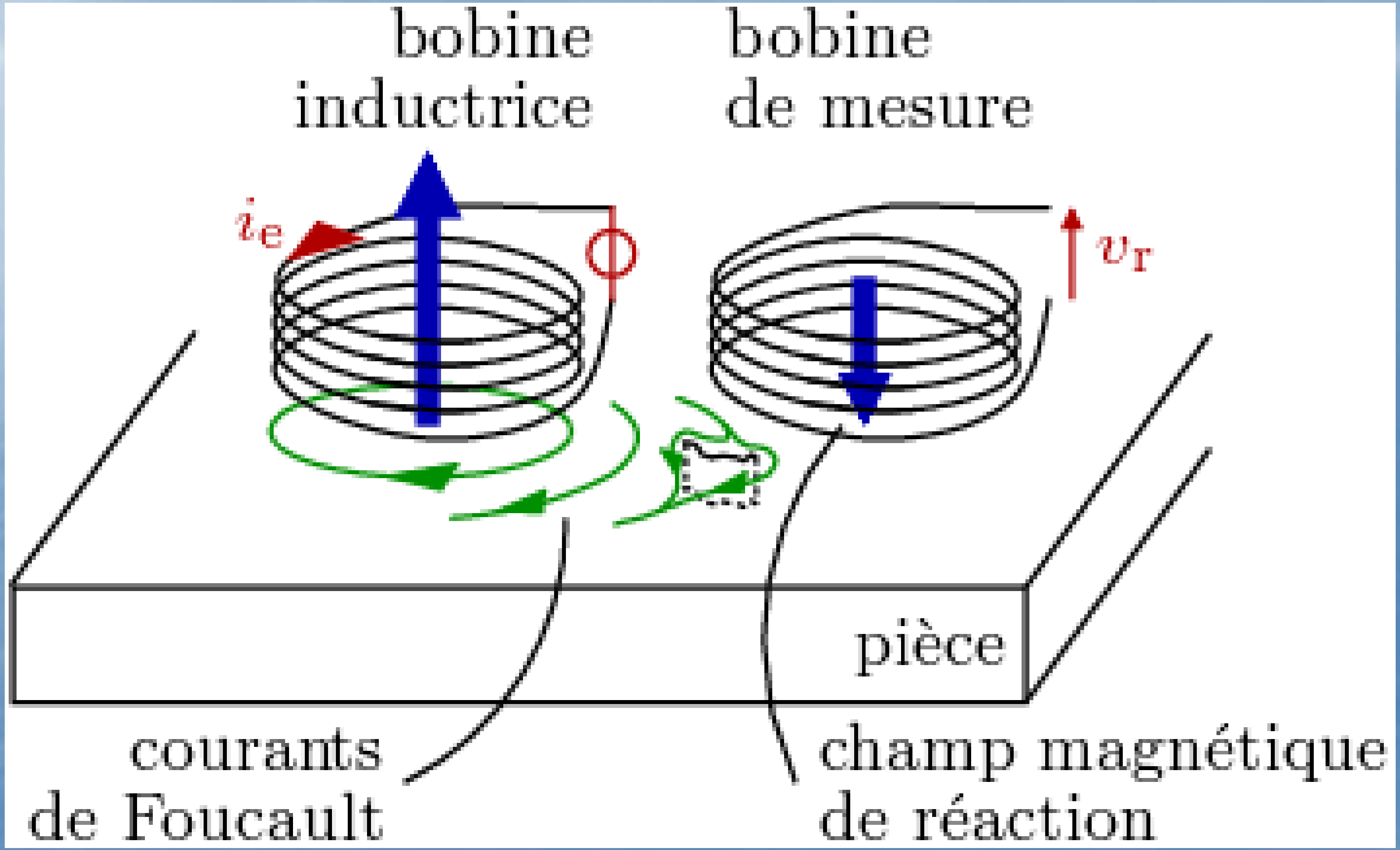
**Récepteur**

- Réponse
- caractériser les défauts superficiels de la pièce à contrôler

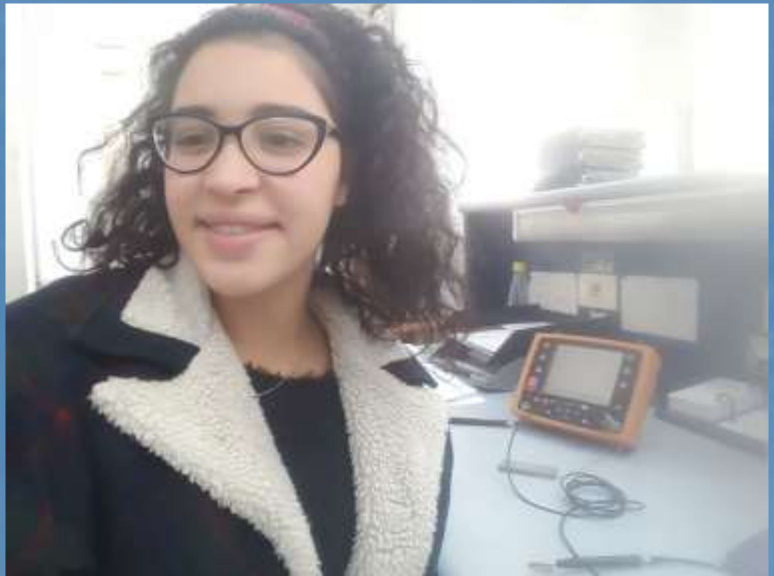
# Différentes techniques de CND

- Émission acoustique
- Courants de Foucault
- Étanchéité
- Magnétoscope
- ressuage
- Radiographie
- Ultrasons
- Examen visuel
- Thermographie

# Méthode de contrôle par courants de Foucault



# 3ème expérience



# Modélisation

**Hypothèses**

$\overrightarrow{rot}\left(\vec{E}\right) = -\frac{\partial \vec{B}}{\partial t}$

$\overrightarrow{rot}\left(\vec{H}\right) = \vec{j}$

div$(\vec{j}) = 0$

$\vec{j} = \overrightarrow{j_0} + \overrightarrow{j_s}$

$\overrightarrow{j_0} = \gamma\vec{E}$

$\vec{B} = \mu\vec{H}$

div$(\vec{B}) = 0$

$\exists \vec{A} \backslash \vec{B} = \overrightarrow{rot}(\vec{A})$ *avec* $\vec{A} \leq$ *potentiel vecteur magnétique*

$\overrightarrow{rot}\left(\vec{E}\right) = -\frac{\partial \vec{B}}{\partial t} \Rightarrow \overrightarrow{rot}\left(\vec{E}\right) = -\frac{\partial}{\partial\ t}\overrightarrow{rot}(\vec{A})$

$\overrightarrow{rot}\left(\vec{E} + \frac{\partial \vec{A}}{\partial\ t}\right) = \vec{0}$

$\Rightarrow \exists$ V\ $\vec{E} + \frac{\partial \vec{A}}{\partial\ t}$ = -$\overrightarrow{grad}$ (V)

$\Rightarrow \vec{E}$ = -$\overrightarrow{grad}$ (V)- $\frac{\partial \vec{A}}{\partial\ t}$

Et on a div$(\vec{j}) = 0$ *et* $\overrightarrow{rot}\left(\vec{H}\right) = \vec{j}$ ainsi $\overrightarrow{rot}\left(\vec{H}\right) = \overrightarrow{j_0} + \overrightarrow{j_i}$ avec $\overrightarrow{j_i} = \gamma\vec{E}$

$\Rightarrow \overrightarrow{rot}\left(\vec{H}\right) = \overrightarrow{j_0} + \gamma\vec{E} \Rightarrow \overrightarrow{rot}\left(\frac{\vec{B}}{\mu}\right) = \overrightarrow{j_0} - \gamma(\overrightarrow{grad}(V) - \frac{\partial \vec{A}}{\partial\ t})$

$\Rightarrow \overrightarrow{rot}\left(\frac{1}{\mu}\overrightarrow{rot}(\vec{A})\right) = \overrightarrow{j_0} - \gamma(\overrightarrow{grad}(V) - \frac{\partial \vec{A}}{\partial\ t})$

$\Rightarrow -\frac{1}{\mu}\vec{\Delta}(\vec{A}) + \gamma\frac{\partial \vec{A}}{\partial\ t} = \overrightarrow{j_0} - \overrightarrow{grad}\left(\frac{1}{\mu}\text{div}(\vec{A}) + \gamma V\right)$

$\Rightarrow$ $-\frac{1}{\mu}\vec{\Delta}(\vec{A})+\gamma\frac{\partial\vec{A}}{\partial\ t}=\ \vec{j_0}-\overrightarrow{grad}\,(\frac{1}{\mu}\text{div}(\vec{A})+\gamma V)$

On a div($\vec{A}$)=0 d'après la jauge de Coulomb

On projette suivant x:

On aura $-\frac{1}{\mu}\Delta(A_x)+\gamma\frac{\partial A_x}{\partial t}=\ \vec{j_0}-\gamma\frac{\partial V}{\partial x}$

On suppose que $\frac{\partial V}{\partial x}$=0

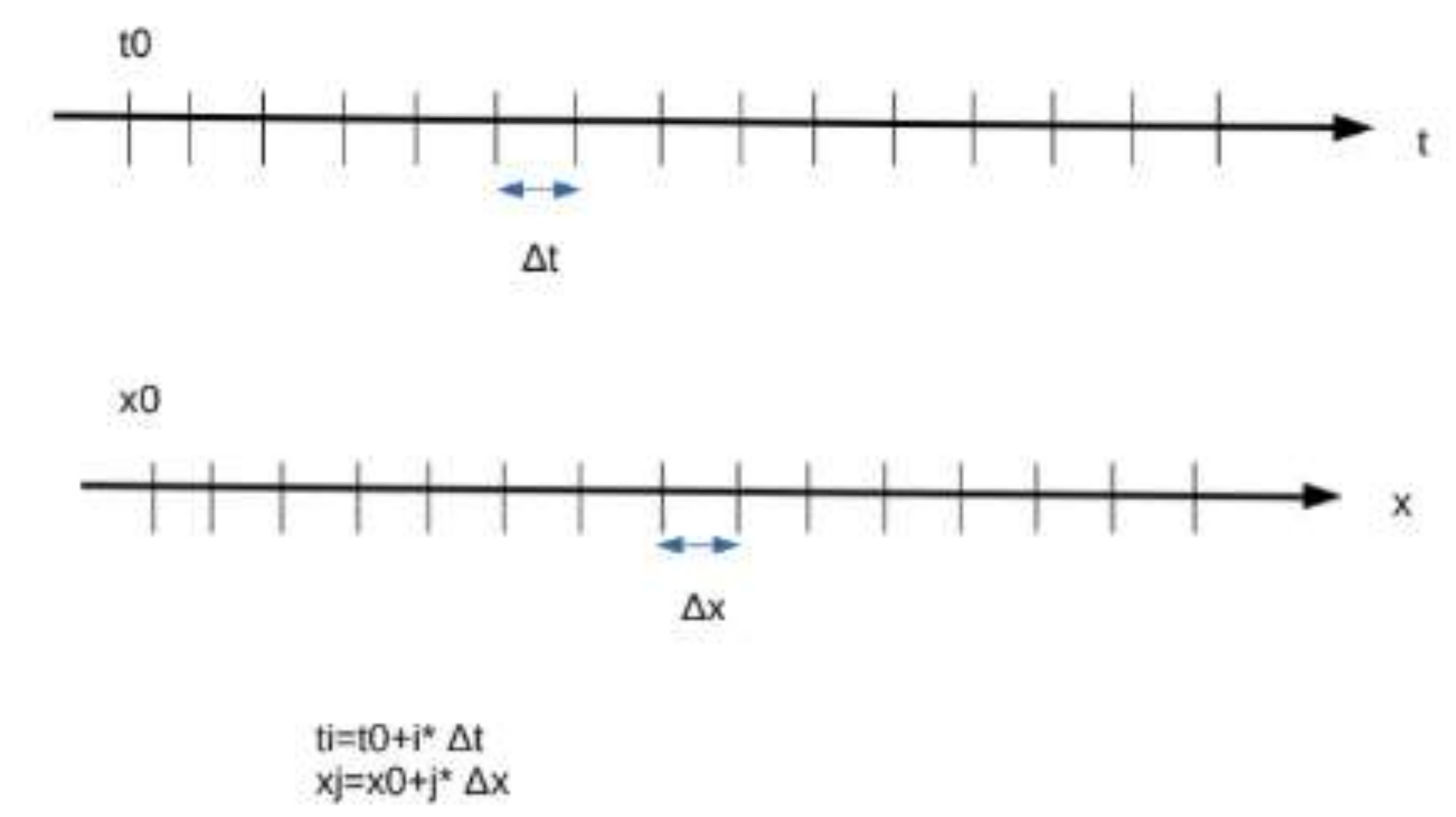
$\Rightarrow$ $-\frac{1}{\mu}\Delta(A_x)+\gamma\frac{\partial A_x}{\partial t}=\ \vec{j_0}$

On a $\Delta(A)=\frac{\partial^2 A}{\partial x^2}+\frac{\partial^2 A}{\partial y^2}+\frac{\partial A}{\partial z^2}$

On suppose que $\Delta(A)=\frac{\partial^2 A}{\partial x^2}$

$-\frac{1}{\mu}\frac{\partial^2 A}{\partial x^2}+\gamma\frac{\partial A_x}{\partial t}=\ \vec{j_0}-\gamma\frac{\partial V}{\partial x}$

# Résolvons cette équation avec la méthode de différences finies



A est de classe C1 par rapport au temps .

Effectuons un développement de Taylor à l'ordre 1

$\Delta t \ll 0$

$\Delta t_i \to 0$

$$A(x, t_i + \Delta t) = A(x, t_i) + \Delta t \frac{\partial A}{\partial t} + O(\Delta t^2)$$

$$\Rightarrow \frac{\partial A}{\partial t} = \frac{A(x, t_i + \Delta t) - A(x, t_i)}{\Delta t}$$

On note également: $\frac{\partial A}{\partial t} = \frac{A_{i+1}^{j} - A_i^j}{\Delta x}$

A est de classe C2 par rapport au temps .

Effectuons un développement de Taylor à l'ordre 2

$\Delta x \ll 0$

$\Delta x \to 0$

$$A(x_j + \Delta x, t) = A(x_j, t) + \Delta x \frac{\partial A}{\partial x} + \frac{\Delta x^2}{2!} \frac{\partial^2 A}{\partial^2 x} + O(\Delta x^3)$$

$$A(x_j - \Delta x, t) = A(x_j, t) - \Delta x \frac{\partial A}{\partial x} + \frac{\Delta x^2}{2!} \frac{\partial^2 A}{\partial^2 x} + O(\Delta x^3)$$

$$\Rightarrow \frac{\partial^2 A}{\partial x^2} = \frac{A(x_j + \Delta x, t) + A(x_j) - 2A(x_j, t)}{\Delta x^2}$$

On note également : $\frac{\partial^2 A}{\partial x^2} = \frac{A_i^{j+1} + A_i^{j-1} - 2A_i^j}{\Delta x^2}$

$$- \frac{1}{\mu}\Delta(A_x) + \gamma\frac{\partial A_x}{\partial t} = \overrightarrow{j_0}$$

$$- \frac{1}{\mu}\frac{A_i^{j+1}+A_i^{j-1}-2A_i^j}{\Delta x^2} + \gamma\frac{A_{i+1}^j-A_i^j}{\Delta x} = j_0$$

$$\Rightarrow \quad \boxed{A_{i+1}^j = \left(j_0 + \frac{1}{\mu}\frac{A_i^{j+1}+A_i^{j-1}-2A_i^j}{\Delta x^2}\right) . \frac{\Delta t}{\gamma} + A_i^j}$$

# Conclusion

- ✓ La propagation des fissures est un phénomène physique complexe, d’où la nécessité d’un couplage entre la méthode expérimentale avec la modélisation physique et résolution numérique.
- ✓ La technique différence finie ne permet que de reproduire la diffusion de la chaleur par contre je n’arrive pas à simuler la fissure.
- ✓ Des méthodes numériques plus élaborées sont nécessaires pour comprendre finement le phénomène.

# Annexes

```
#3D heat equation

#from math import pi, exp, sqrt
from numpy import pi,exp,sqrt
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import axes3d
import matplotlib.animation as animation

fig = plt.figure()
fig.set_dpi(100)
ax1 = fig.gca(projection='3d')

x = np.linspace(-5,5,30)
y = np.linspace(-5,5,30)

X,Y = np.meshgrid(x,y)

#Initial time
t0 = 0

#Time increment
dt = 0.05

#Initial displacement at (0,0) at t0=0
u = 2000

#Sigma squared
s = 2

#displacement function
def f(x,y,t):
    return (u/sqrt(1+4*t/s))*exp(-(x**2+y**2)/(s+4*t))


#def f(x,t):

# (u/sqrt(1+4*t/s))*exp(-(x**2)/(s+4*t))

a = []

# 500
for i in range(1000):
    z = u(X,Y,t0)
    t0 = t0 + dt
    a.append(z)

m = plt.cm.ScalarMappable(cmap=plt.cm.jet)
#m = plt.cm.ScalarMappable(cmap=plt.cm.hot)
m.set_array(a[0])
cbar = plt.colorbar(m)

k = 0


def animate(i):
    global k
    temp = a[k]
    k += 1
    ax1.clear()

    ax1.plot_surface(X,Y,temp,rstride=1, cstride=1,cmap=plt.cm.jet,linewidth=0,antialiased=False)
    #ax1.plot_surface(X,Y,temp,rstride=1, cstride=1,cmap=plt.cm.jet,linewidth=0,antialiased=False)

    #ax1.contour(x,y,temp)
    ax1.set_zlim(0,T)
    ax1.set_xlim(-5,5)
    ax1.set_ylim(-5,5)

anim = animation.FuncAnimation(fig,animate,frames=220,interval=20)
plt.xlabel("X en km")
plt.ylabel("Y en km")
plt.show()
```

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

dx = 0.01
dy = 0.01
a = 0.5    # Diffusion constant.
timesteps = 200
nx = int(1/dx)
ny = int(1/dy)

dx2 = dx**2
dy2 = dy**2

# Pour des considérations de stabilité numérique du schéma numérique
# on prend cette valeur maximale pour le pas temporel
dt = dx2*dy2/(2*a*(dx2+dy2))


u =  np.zeros([nx,ny,timesteps])

"""
for i in range(nx):
    for j in range(ny):
        if ( ( (i*dx - 0.5)**2 + (j*dy-0.5)**2 <= 0.1)
        & ((i*dx-0.5)**2 + (j*dy-0.5)**2 >= 0.05 ) ):
            u[i,j,0] = 1.


from numpy.random import random

for i in range(nx):
    for j in range(ny):
        u[-1,j,0] = 1


from numpy.random import random

for i in range(nx):
    for j in range(ny):
        u[0,j,0] = 1.

from numpy.random import random

for i in range(nx):
    for j in range(ny):
        u[i,j,0] = random()

###############
for i in range(nx):
    for j in range(ny):
        u[0,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[10,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[20,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[30,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[40,j,0] = 10.

############################
```

```
############################
for i in range(nx):
    for j in range(ny):
        u[0,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[10,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[20,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[30,j,0] = 10.

for i in range(nx):
    for j in range(ny):
        u[40,j,0] = 10.


for i in range(nx):
    for j in range(ny):
        u[80,j,0] = 5.


for i in range(nx):
    for j in range(ny):
        u[90,j,0] = 5.

import random as r

LX = [r.randrange(0,100) for x in range(100)]
LY = [r.randrange(0,100) for x in range(100)]

for x in LX:
    for y in LY:
        u[x,y,:] = 100.*r.random()

"""
for i in range(0,30):
    for j in range(25,28):
        u[i,j,0] = 10.


def calcul(u):
    global nx,ny
    for k in range(timesteps-1):
        print("Computing u for k= ",k)
        for i in range(1,nx-1):
            for j in range(1,ny-1):
                u[i,j,k+1] = u[i,j,k] + dt*a*((u[i+1,j,k] - 2*u[i,j,k] + u[i-1,j,k])/dx2
                 +(u[i,j+1,k] - 2*u[i,j,k] + u[i,j-1,k])/dy2)
```