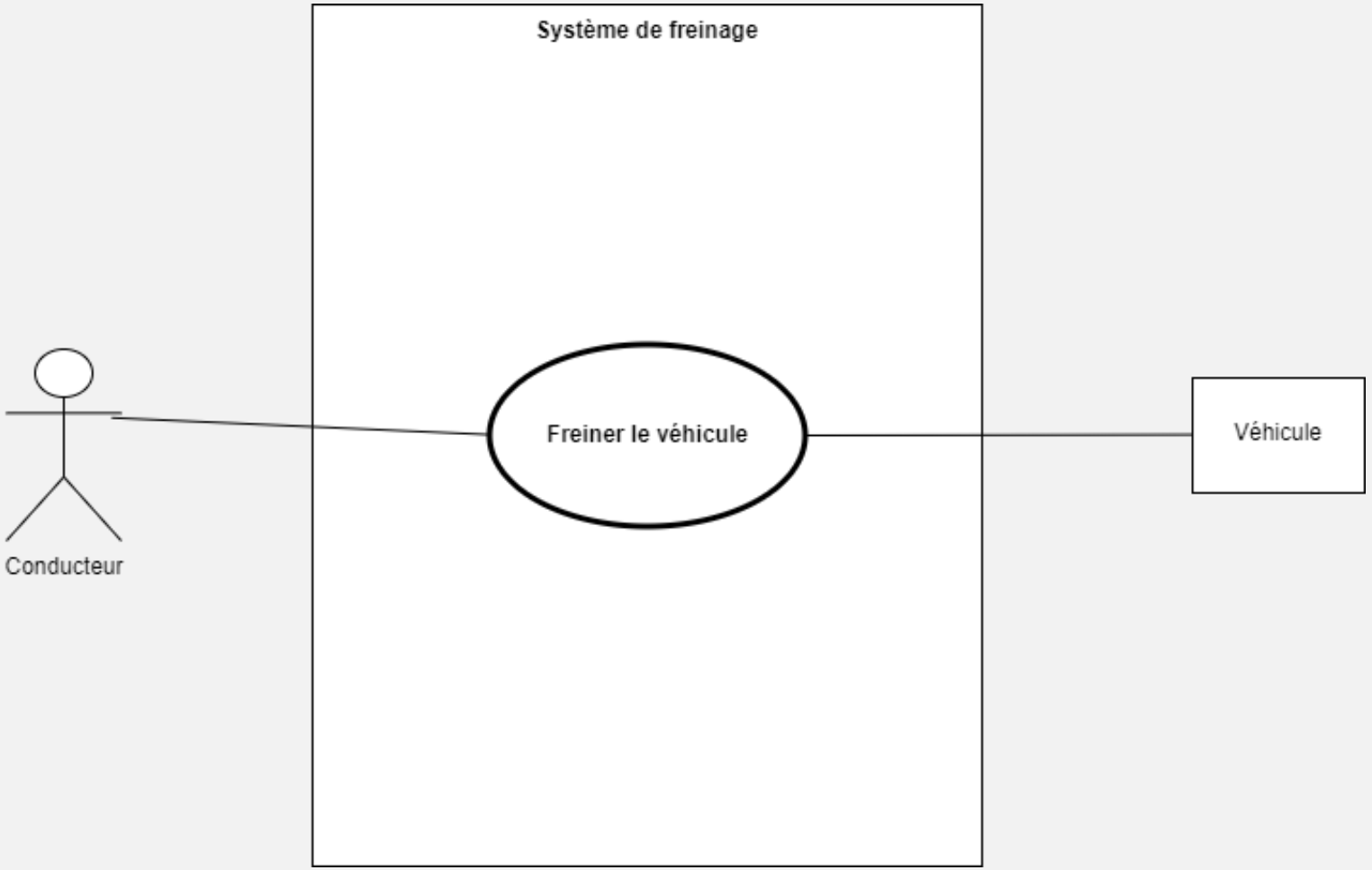
PRÉVENTION DES RISQUES DE SURCHAUFFE DES SYSTÈMES DE FREINAGE AUTOMOBILES

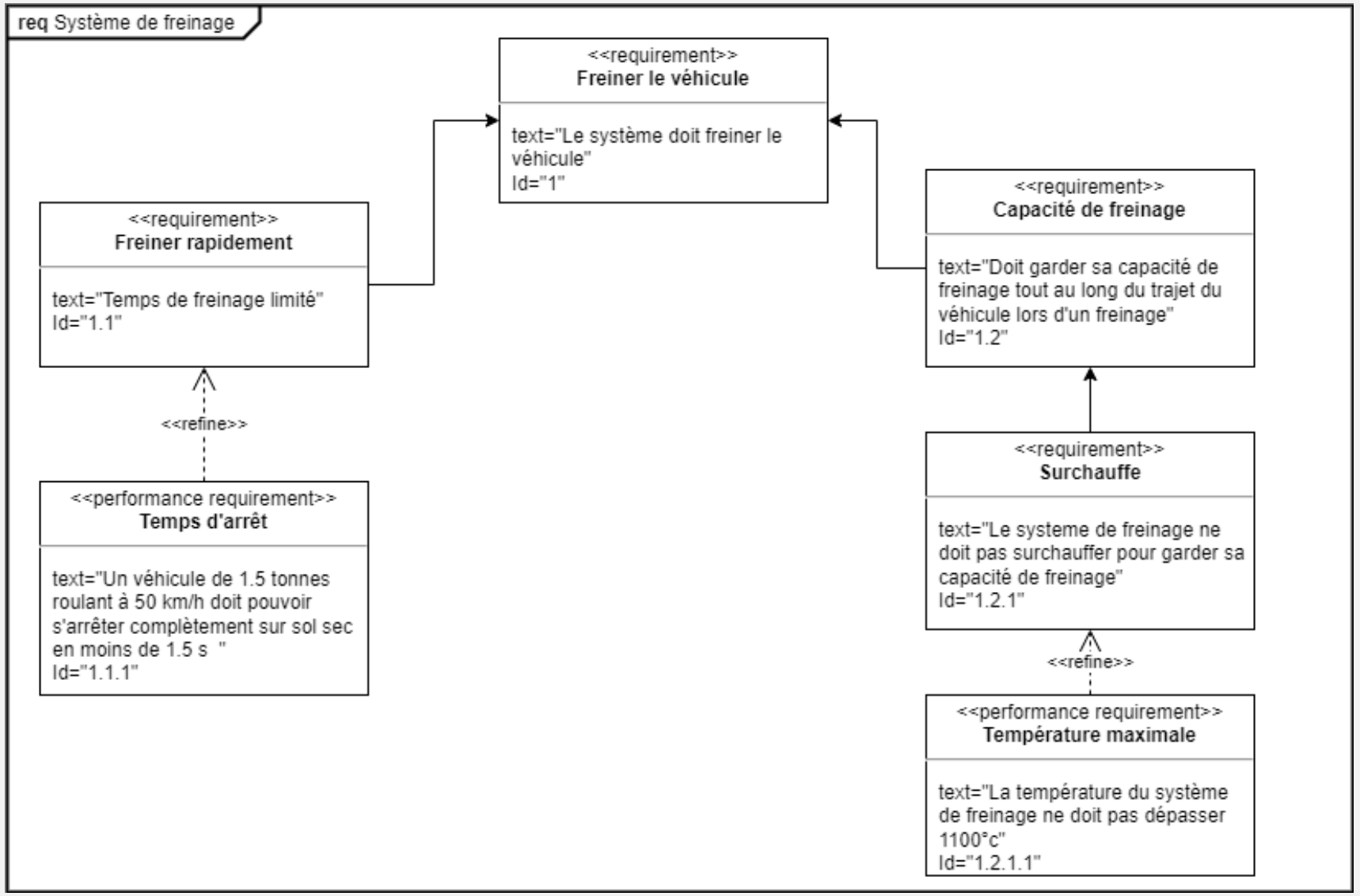
Furnon Clément

N° candidat : 29435

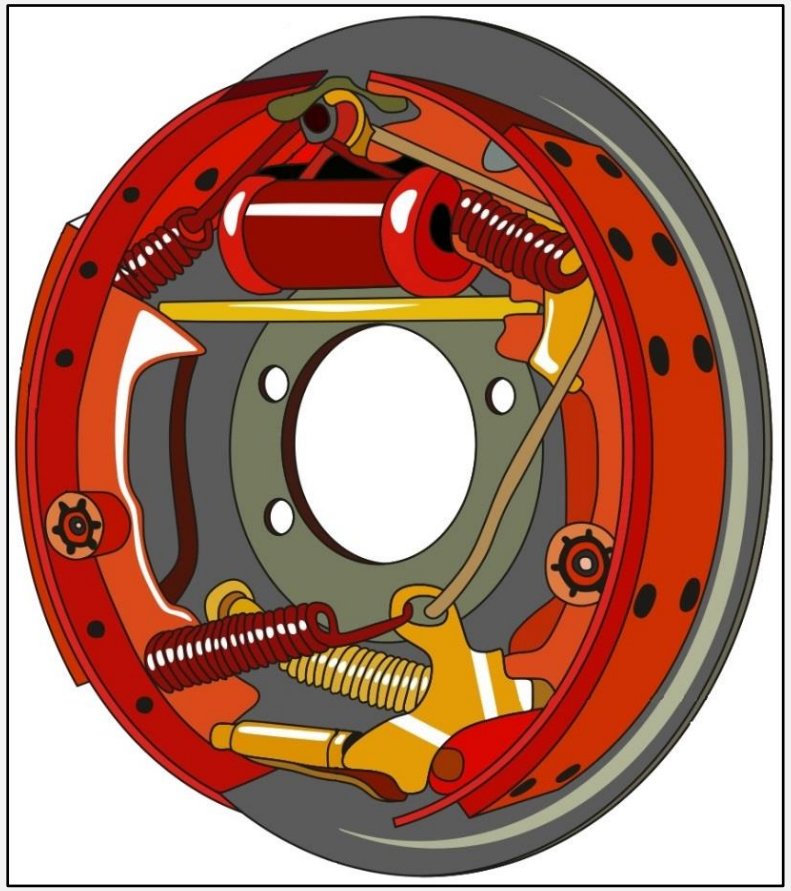
INTRODUCTION

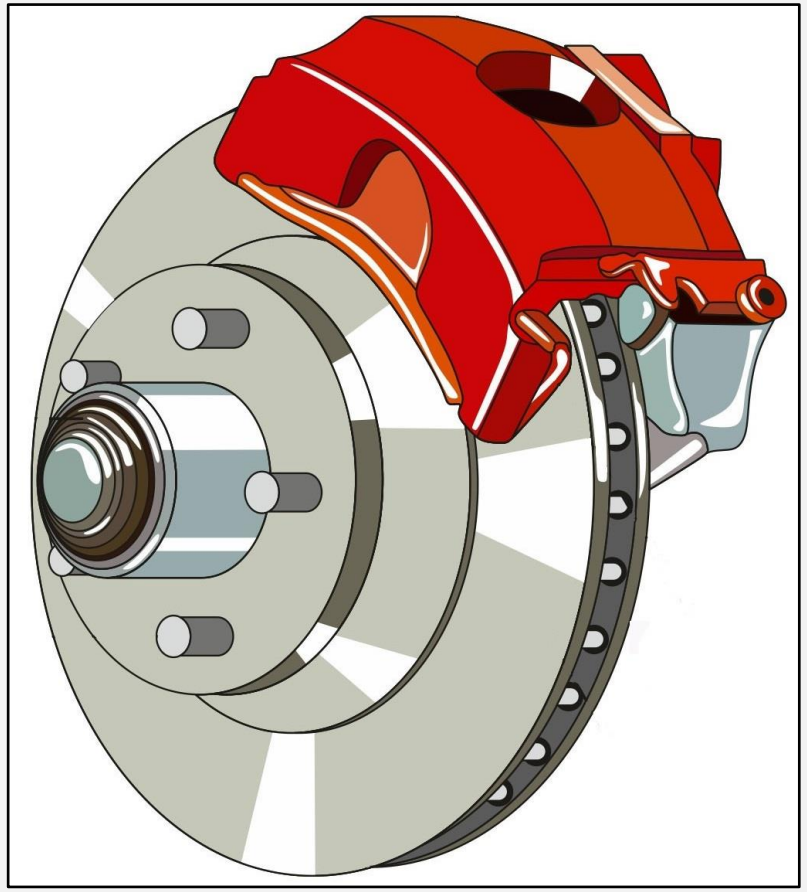


INTRODUCTION



INTRODUCTION

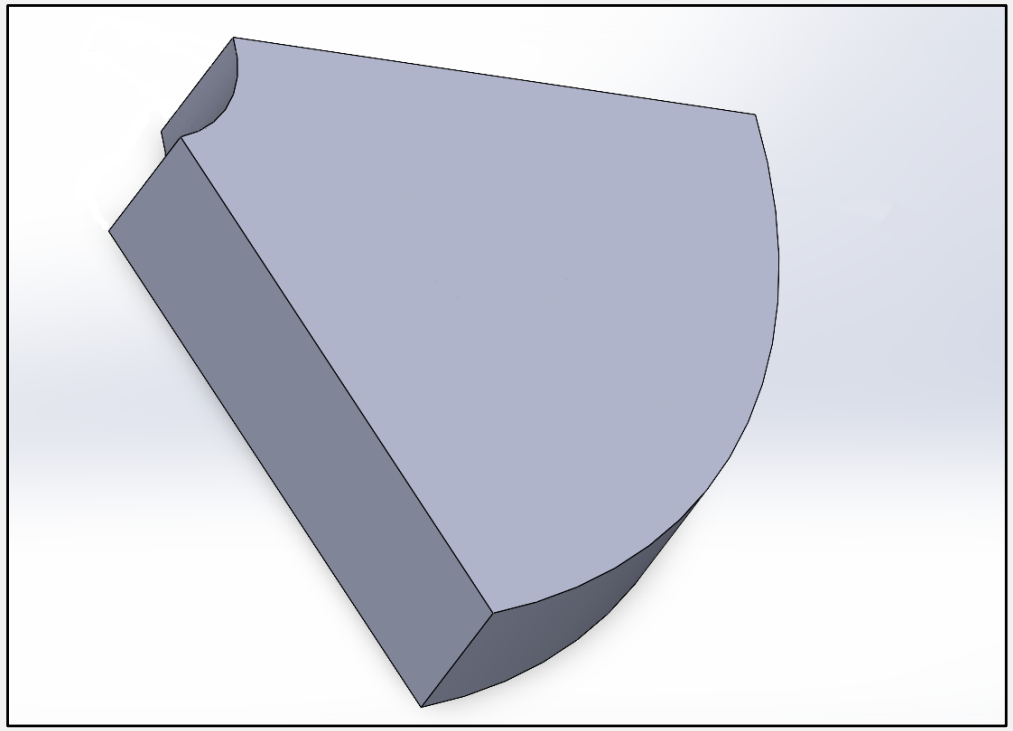




PROBLÉMATIQUE

Comment éviter les risques de surchauffe des dispositifs de freinage ?

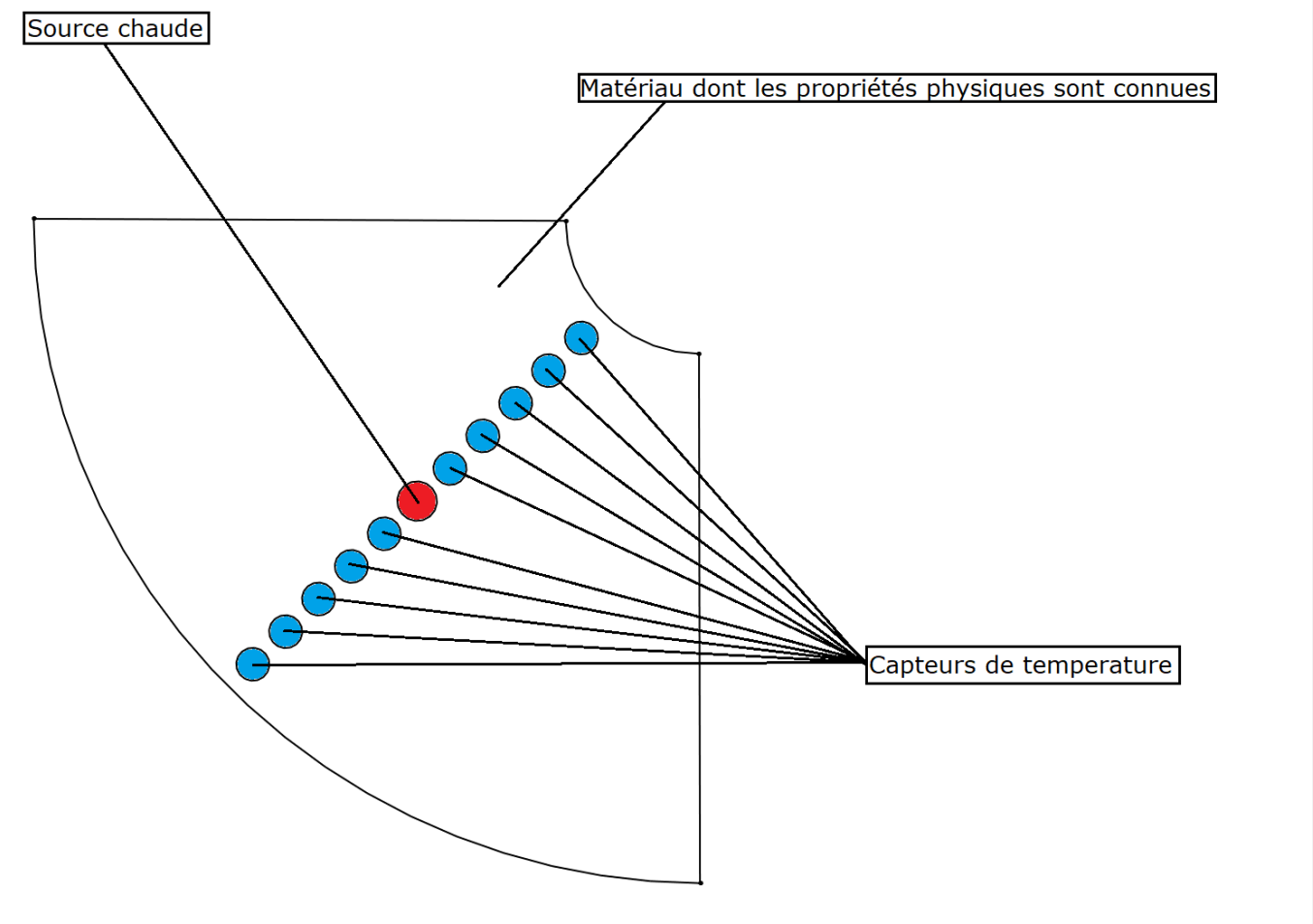
MISE EN PLACE DU MODÈLE



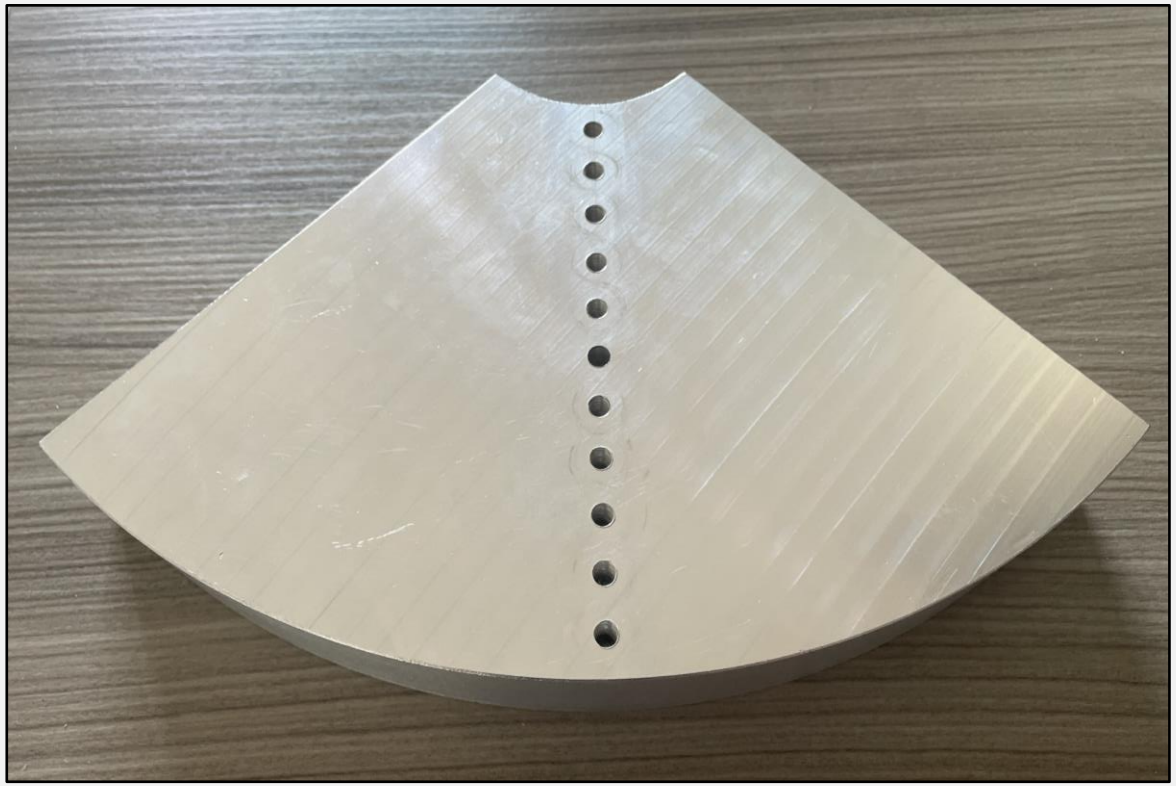
Quart de disque plein

Matériau homogène

PROTOCOLE EXPÉRIMENTALE



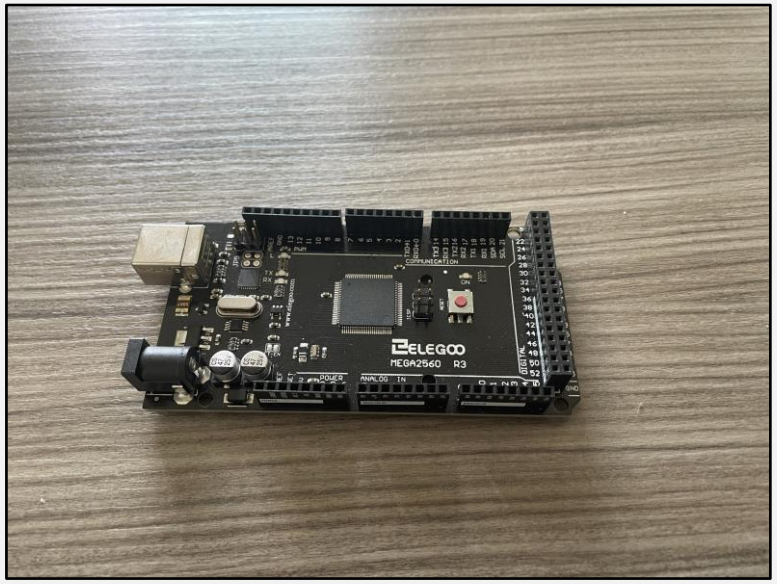
ETUDE EXPÉRIMENTALE



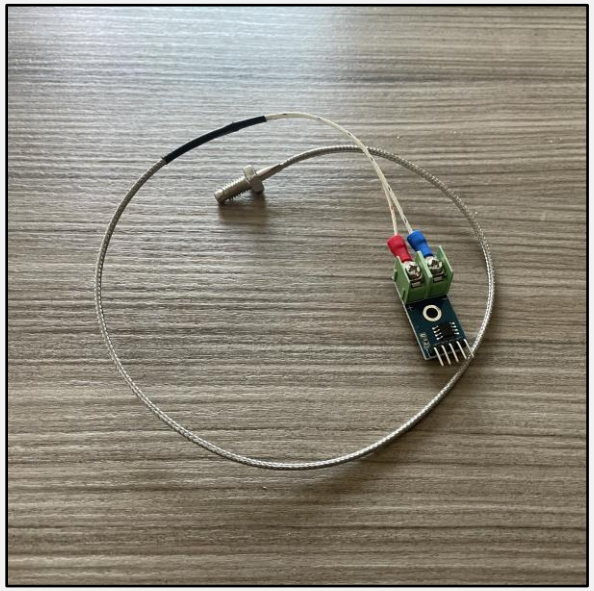
Matériau : Aluminium (AU4G)

ETUDE EXPÉRIMENTALE

Chaine de mesures



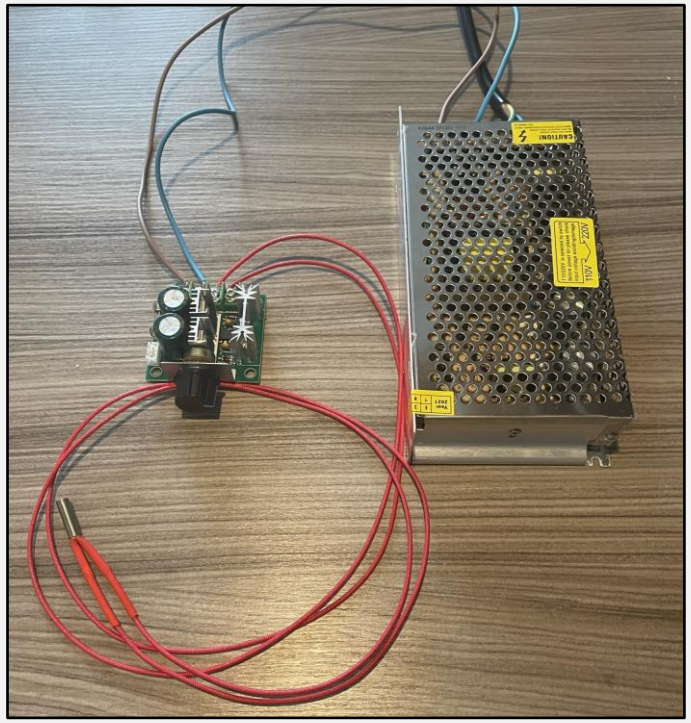
Arduino Méga 2560



Bibliothèque : max6675

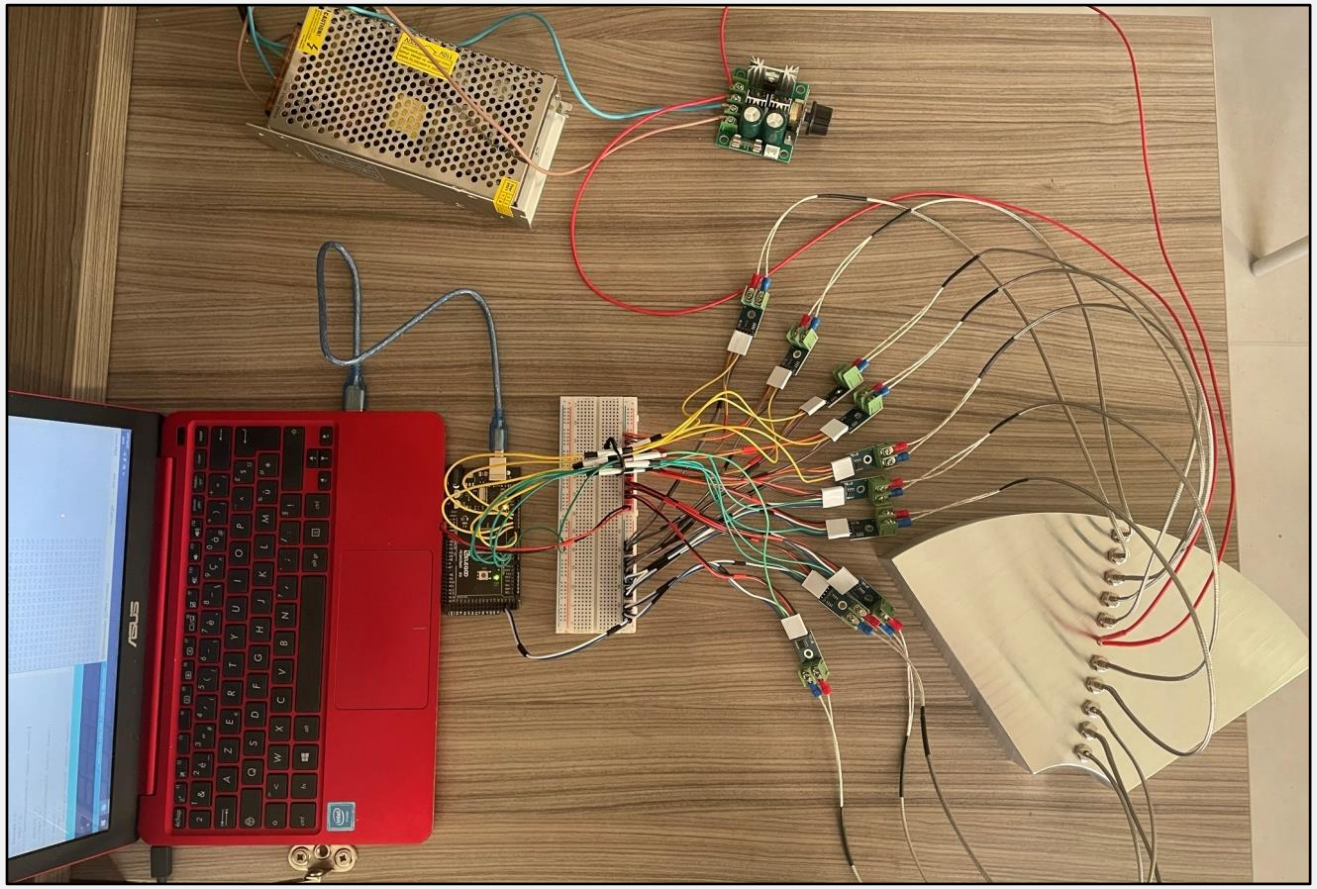
Précision : 0,5°C

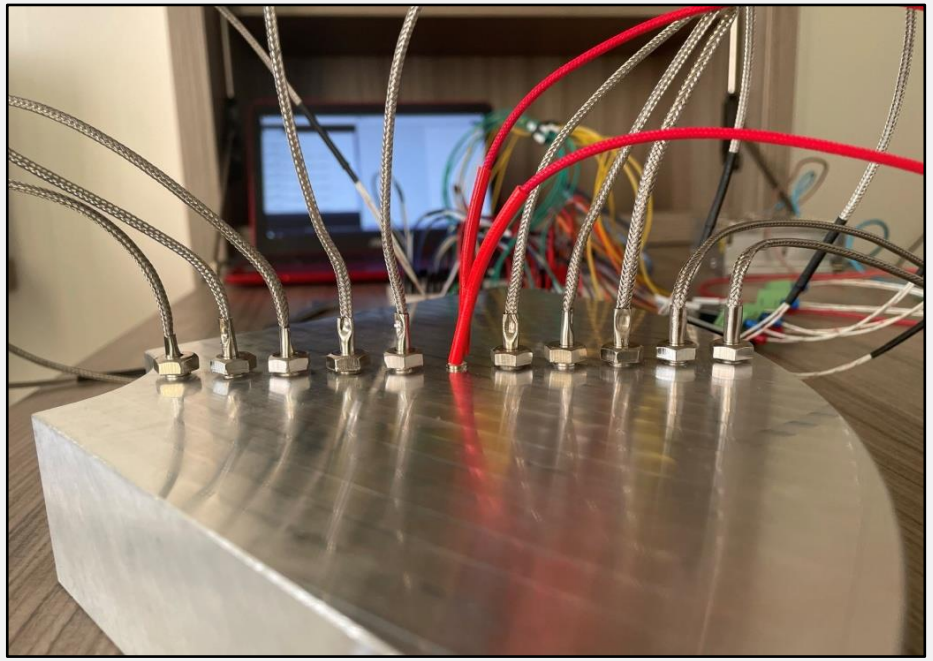
ETUDE EXPÉRIMENTALE



Puissance : Resistance donc $P=RI^2$

ETUDE EXPÉRIMENTALE





TRAITEMENT DES DONNÉES

Sur python avec le module : Py serial

Type bytes : b'$t1(T_1)$ $t2(T_1)$ $t3(T_1)$ $t4(T_1)$ $t5(T_1)$ $t6(T_1)$ $t7(T_1)$ $t8(T_1)$ $t9(T_1)$ $t10(T_1)$'

ou b'nan'

Type float : $t1=[t1(T_1), t1(T_2), \dots, t1(T_n)]$

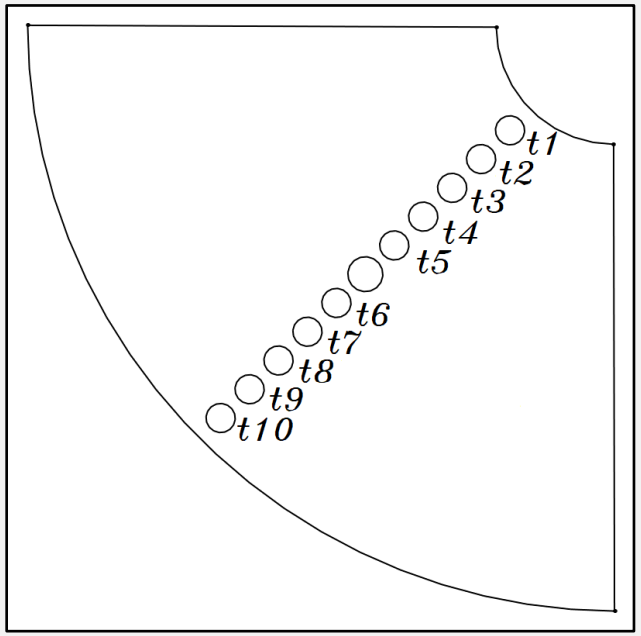
:

:

$t10=[t10(T_1), t10(T_2), \dots, t1(T_n)]$

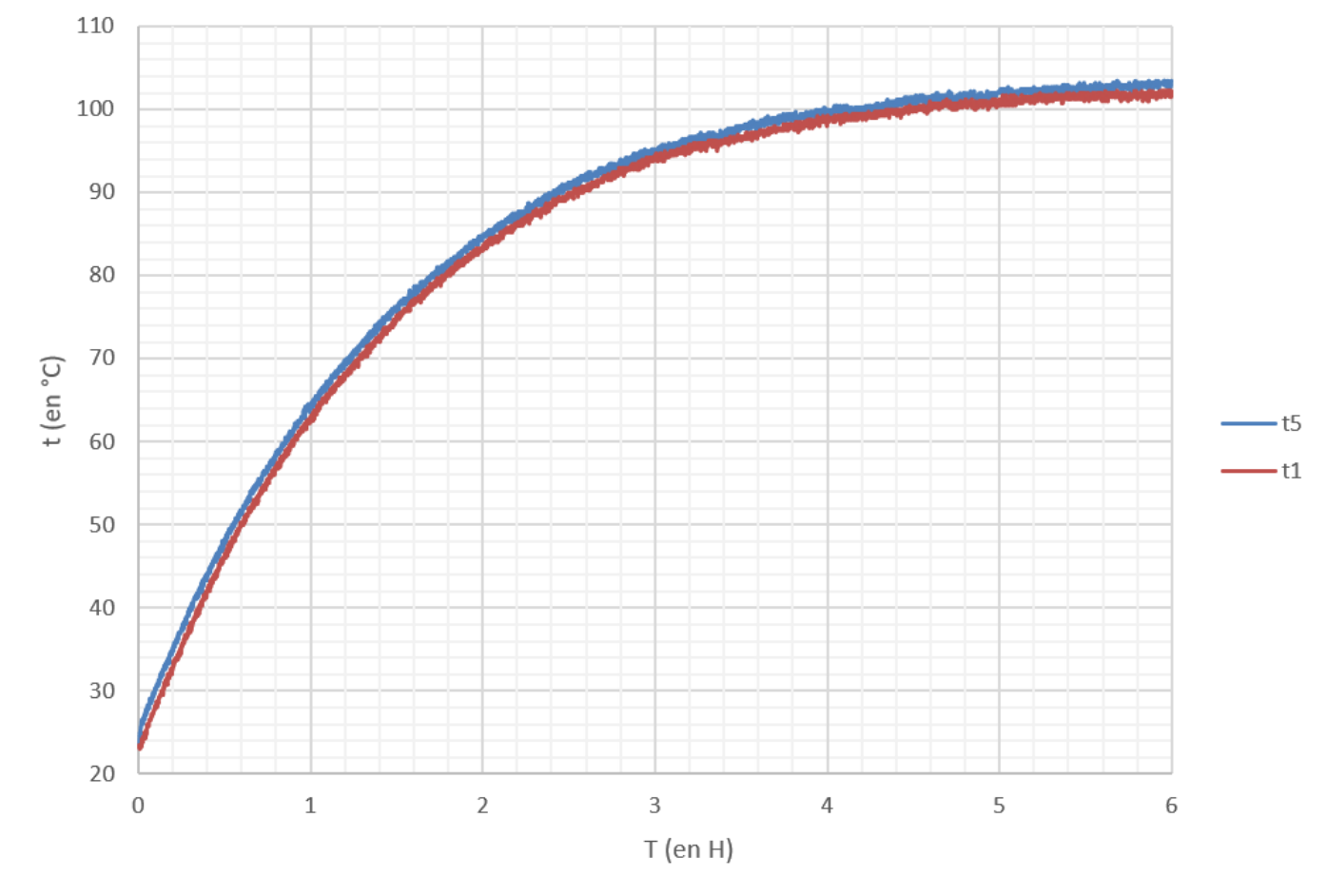
SAUVEGARDES DES DONNÉES

Sur Excel avec le module : pandas



	A	B	C	D	E	F	G	H	I	J	K
1	**T**	**t1**	**t2**	**t3**	**t4**	**t5**	**t6**	**t7**	**t8**	**t9**	**t10**
2	5,23	23,25	23,5	23,25	23,25	23,25	23,5	23	23,5	23,25	23,5
3	10,36	23,25	23,5	22,5	23,75	23,75	23,75	23	22,75	23	23,5
4	15,48	23,5	23,5	23,25	23,5	24,25	24	24	23,75	23,25	23,75
5	20,61	23	23,5	23,5	24	24,5	24,5	24	24	23	23,25
6	25,73	23,5	23,5	23,5	23,5	25	24,5	23,5	24	23	24
7	30,86	23,25	23,75	23,25	24	24,75	25	24,25	24,25	23,25	24
8	35,98	23,25	23,75	22,75	23,75	25,25	25,25	24,75	24,25	23,25	24
9	41,1	23,5	23,5	24	24,5	25,5	26	24,5	24,25	23,5	24
10	46,23	23,5	24	23,75	24,75	25,5	26	25,25	24,75	23,75	24,5
11	51,35	23,25	23,75	24	24,75	25,75	26	25,25	24,5	23,75	23,75
12	56,48	23,25	24,5	24,25	24,75	26,25	26,25	24,75	24,75	23,75	23,75
13	61,6	23,75	24,25	24,25	25,25	26,25	26	25,25	24,75	24	24,5
14	66,73	24,25	24,5	23,5	25	26	26	25,5	24,75	24	24,5
15	71,85	24	24,5	24,5	24,5	26,5	26,5	25,5	25	24,5	24,5
16	76,97	23,75	24,5	24,5	25,25	26,25	26,25	25,5	25	24,25	24,5
17	82,1	24,25	24,75	24	25,25	26,5	26,75	25,75	25,5	24,5	24,25
18	87,22	24,5	24,5	24,75	24,5	26,25	26,25	25,75	24,75	24,5	24,75
19	92,35	24,25	24,75	24,25	25,75	26	26,25	26	25,25	24,75	24,5
20	97,47	24,25	24,75	24,25	25,25	26,75	27	26	25	24,5	25,25
21	107,72	24,5	24,75	25	25,5	26,5	27	25,75	25,5	24,75	25,25

RÉSULTATS



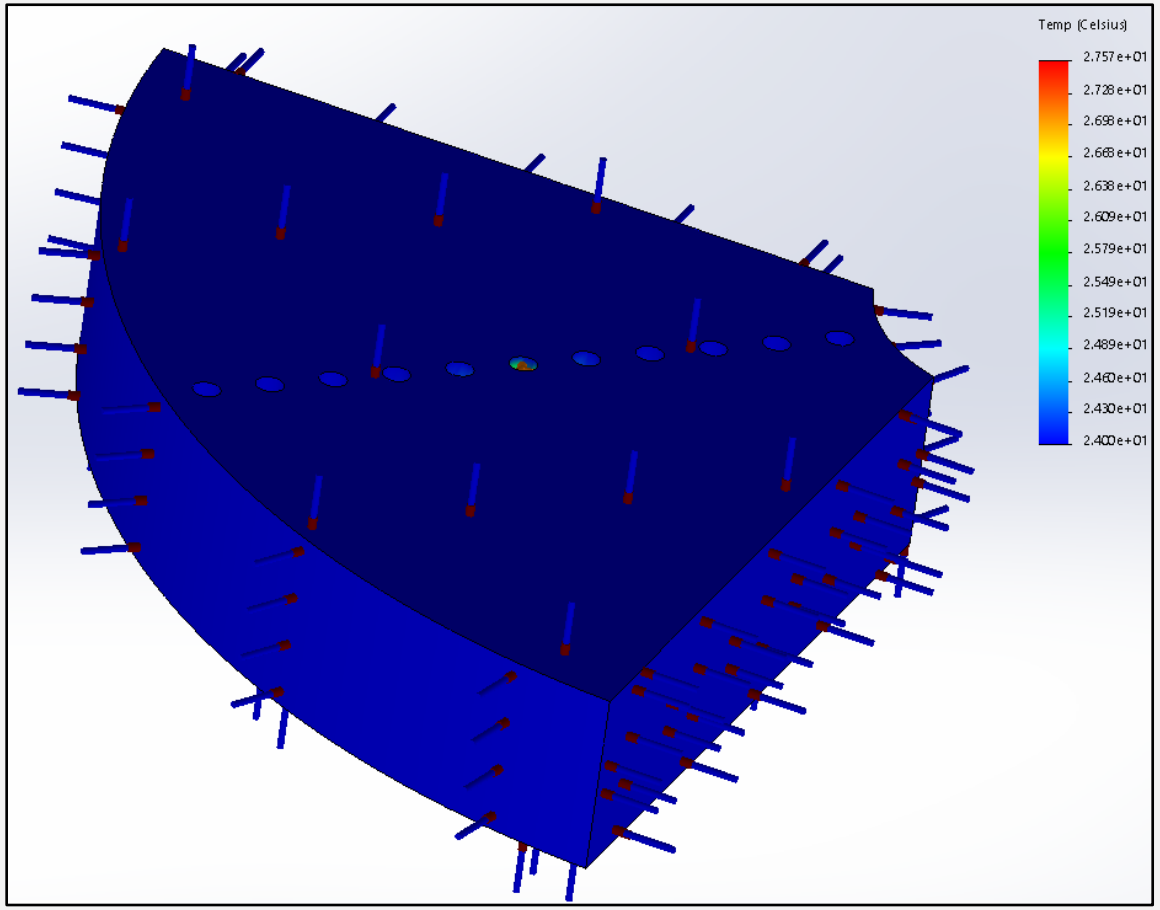
P = 44,6 W

t1 = 102°C
t5 = 103°C

ETUDE SOLIDWORKS

- Puissance 44,6W au milieu

- Température de 24°C sur toutes les faces

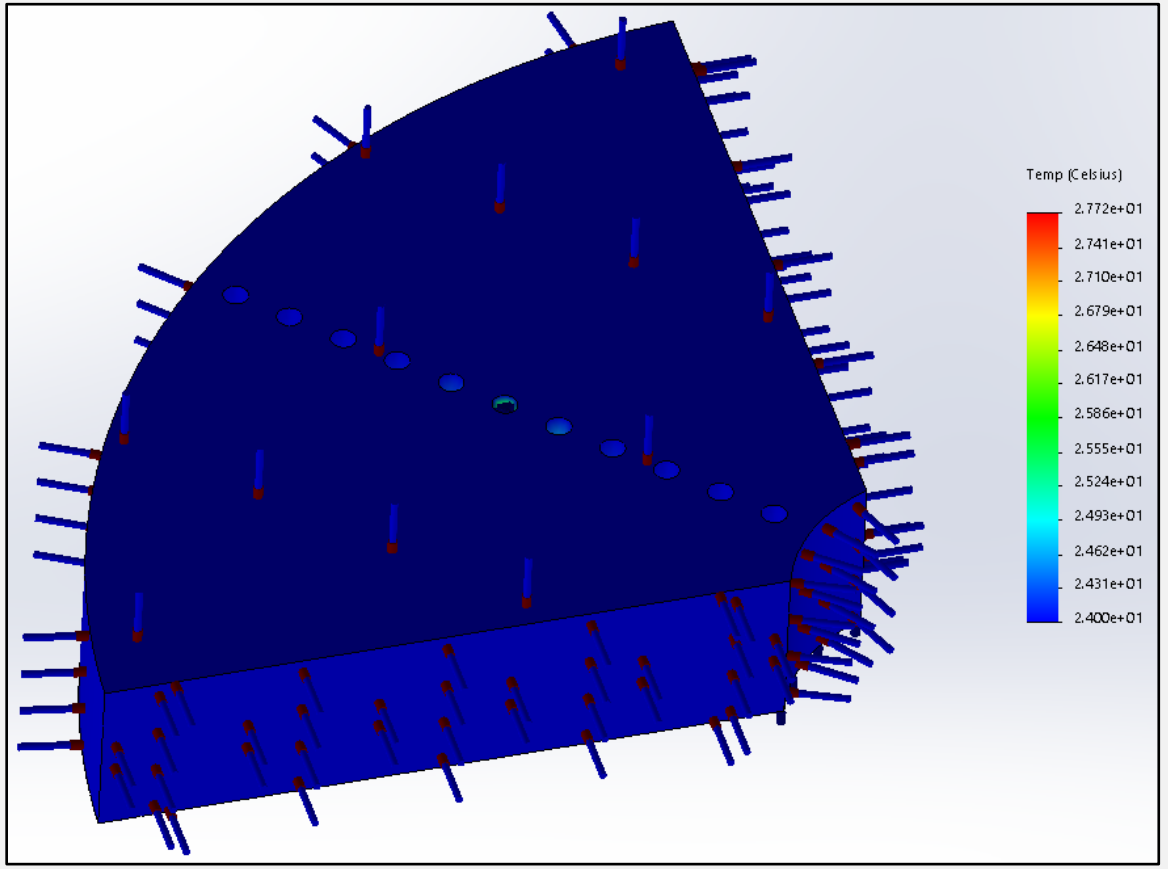


ETUDE SOLIDWORKS

- Puissance 44,6W au milieu

- Température de 24°C sur toutes les faces sauf celle du dessous

- Flux de chaleur nul sur la face du dessous

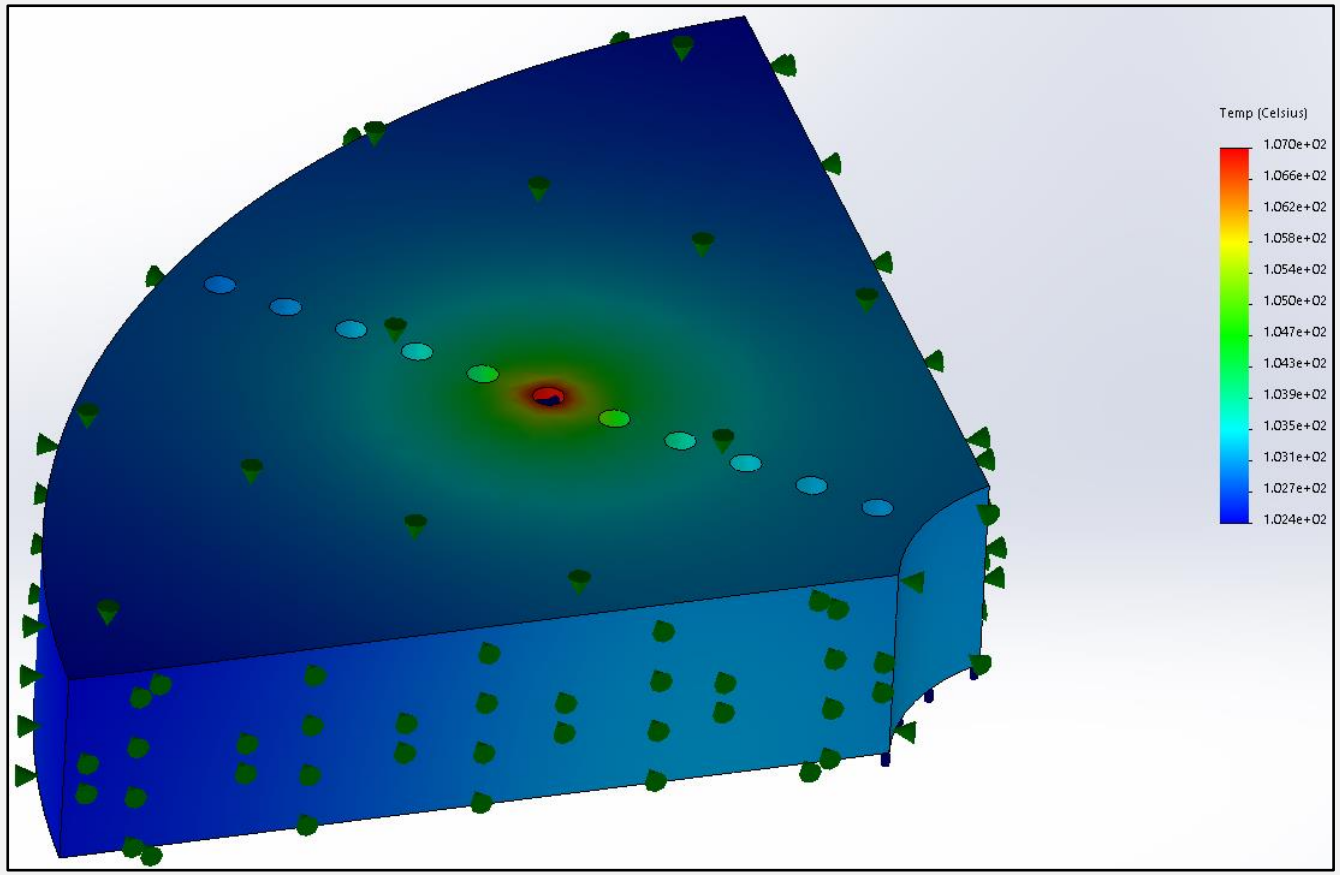


ETUDE SOLIDWORKS

- Puissance 44,6W au milieu

- Convection de 10 W/m².K, température de 24°C sur toutes les faces sauf celle du dessous

- Flux de chaleur nul sur la face du dessous



COMPARAISON DES RÉSULTATS

Températures expérience

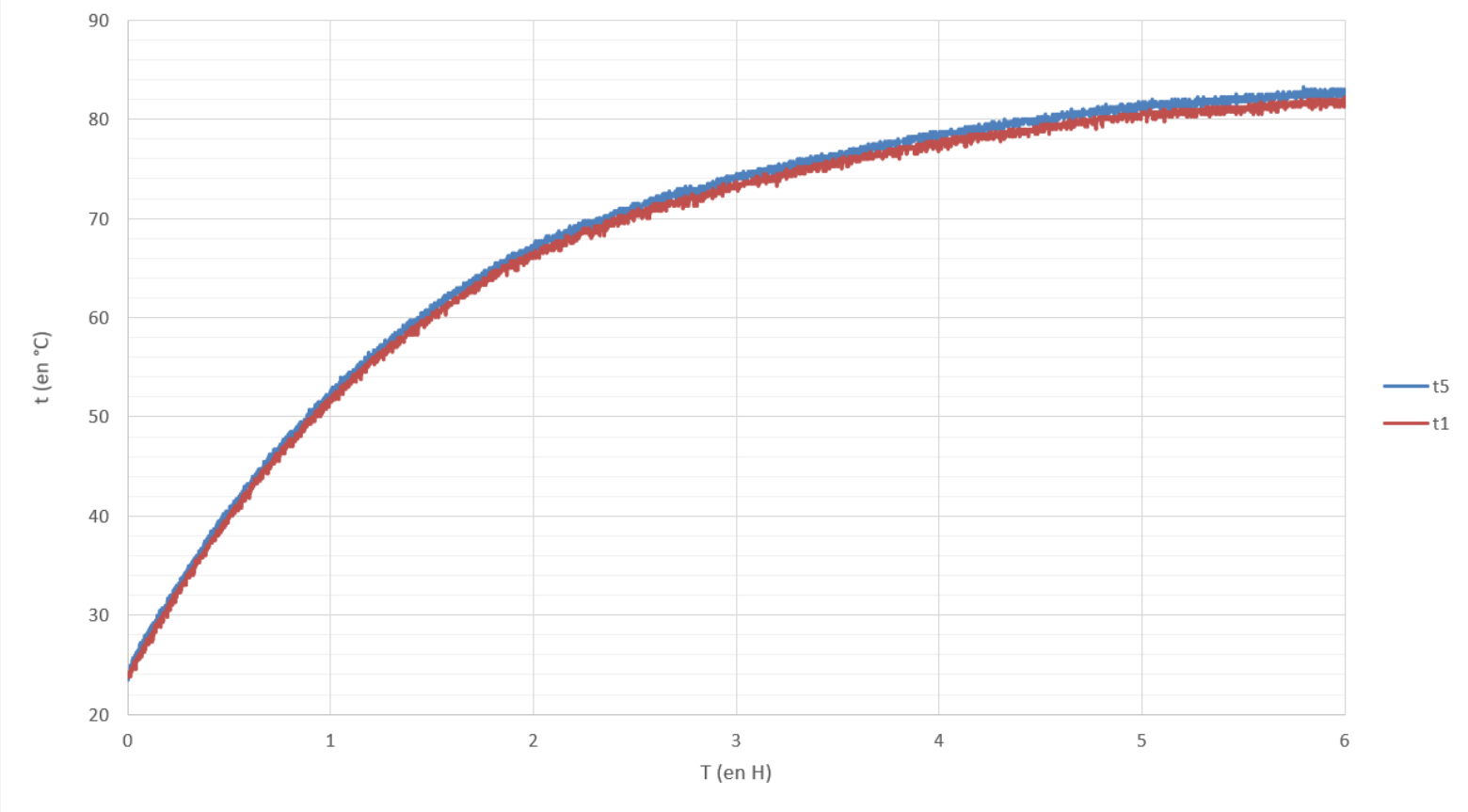
- t1 = 102°C
- t5 = 103°C
- t6 = 104°C
- t8 = 102°C

Températures SolidWorks

- t1 = 102°C
- t5 = 105°C
- t6 = 105°C
- t8 = 103°C

Ecart : 1 °C ± 0,8

ESSAI AVEC UNE PUISSANCE DIFFÉRENTE



P = 22,4W

t1 = 81°C

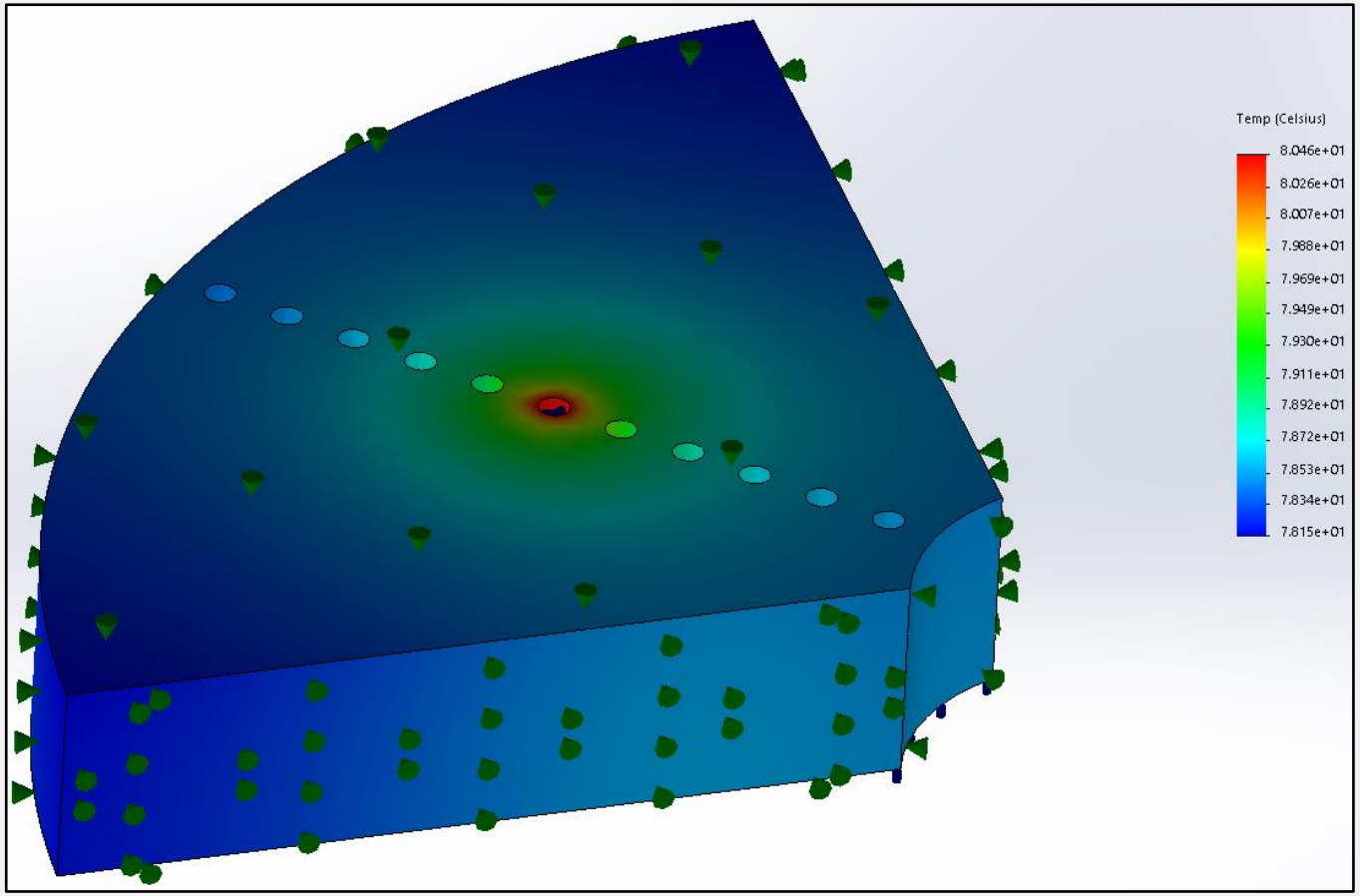
t5 = 83°C

ESSAI AVEC UNE PUISSANCE DIFFÉRENTE

- Puissance 22,4W au milieu

- Convection de 10 W/m².K, température de 24°C sur toutes les faces sauf celle du dessous

- Flux de chaleur nul sur la face du dessous



ESSAI AVEC UNE PUISSANCE DIFFÉRENTE

Températures expérience

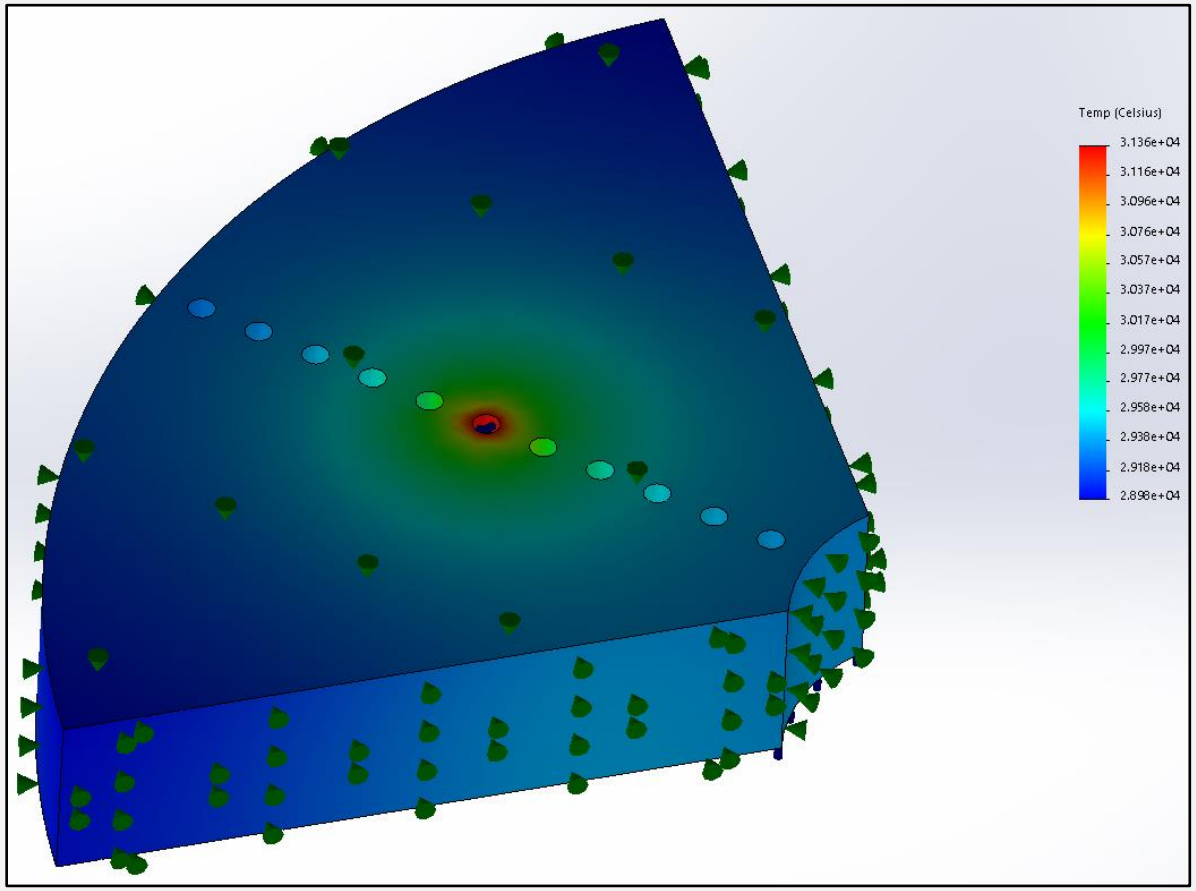
- t1 = 81°C
- t5 = 83°C
- t6 = 83°C
- t8 = 82°C

Températures SolidWorks

- t1 = 78°C
- t5 = 79°C
- t6 = 79°C
- t8 = 78°C

Ecart : 3,75 °C ± 0,5

LIMITES DU MODÈLE SOLIDWORKS



30 000 °C pour une puissance de 15kW

Températures au régime stationnaire

Simulation dynamique sur python

ETUDE THÉORIQUE

Equation de la chaleur :

$$\frac{\partial T}{\partial t} = \frac{\lambda}{c_v \rho} \Delta T + \frac{p}{c_v \rho}$$

ETUDE THÉORIQUE

Discrétisation de l'équation de la chaleur :

$$T(t+\mathrm{d}t) = T(t) + \frac{\lambda}{c_v \rho} \Delta\, T\, \mathrm{d}t + \frac{p}{c_v \rho}\, \mathrm{d}t$$

Discrétisation de d'une dérivée seconde:

$$\frac{\partial^2 f}{\partial x^2} = \frac{f(x+\mathrm{d}x) - 2\,f(x) + f(x-\mathrm{d}x)}{\mathrm{d}x^2}$$

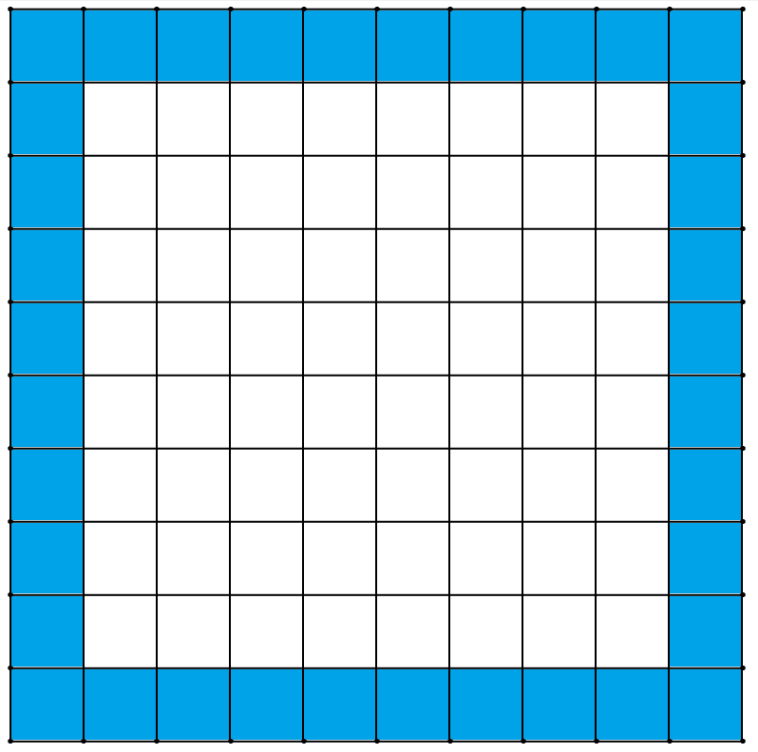
ETUDE THÉORIQUE

Coordonnées cartésiennes (2 dimensions)

$$\Delta f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

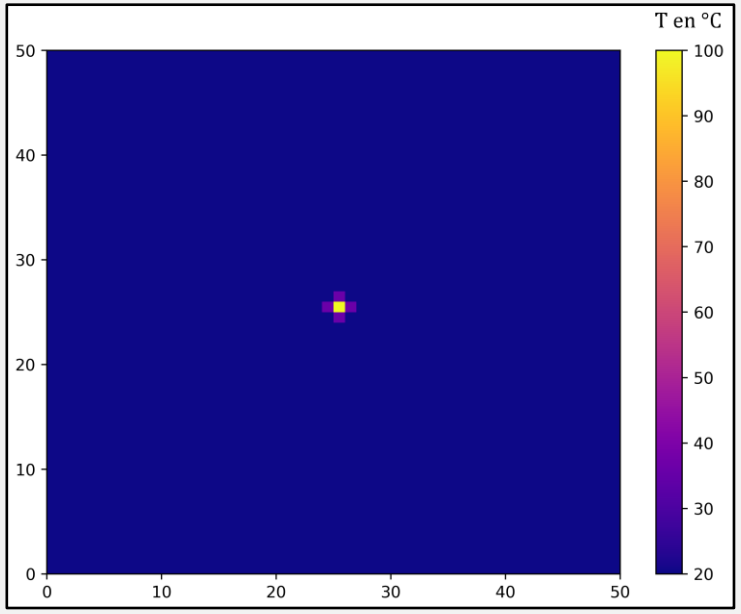
		x,y+dy			
	x-dx,y	x,y	x+dx,y		
		x,y-dy			

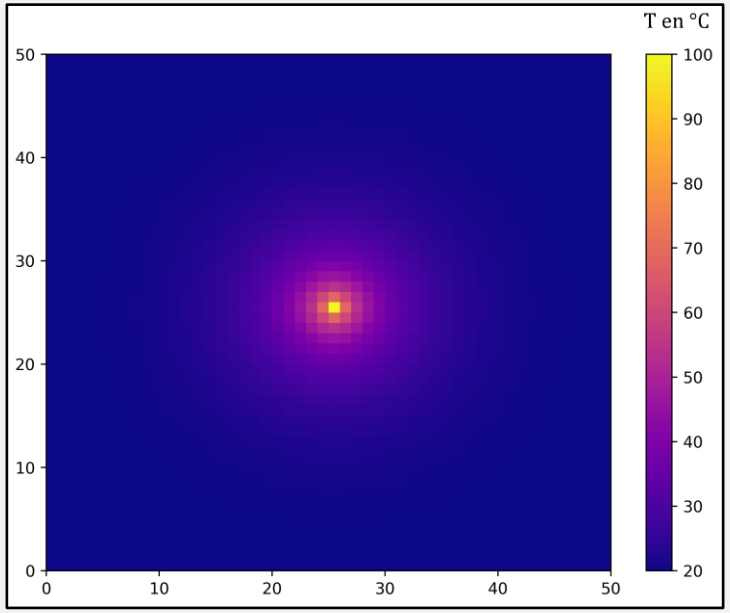
CONDITIONS AUX LIMITES

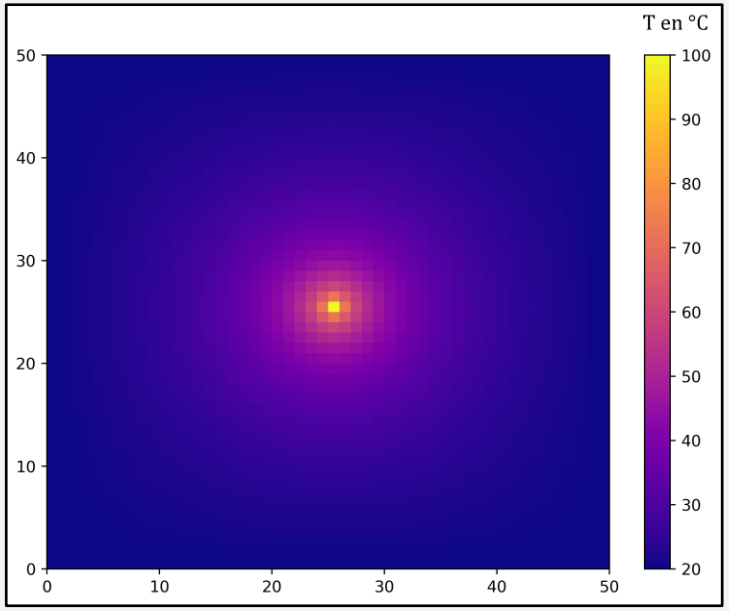


Température extérieure fixe

RÉSULTATS COORDONNÉES CARTÉSIENNES



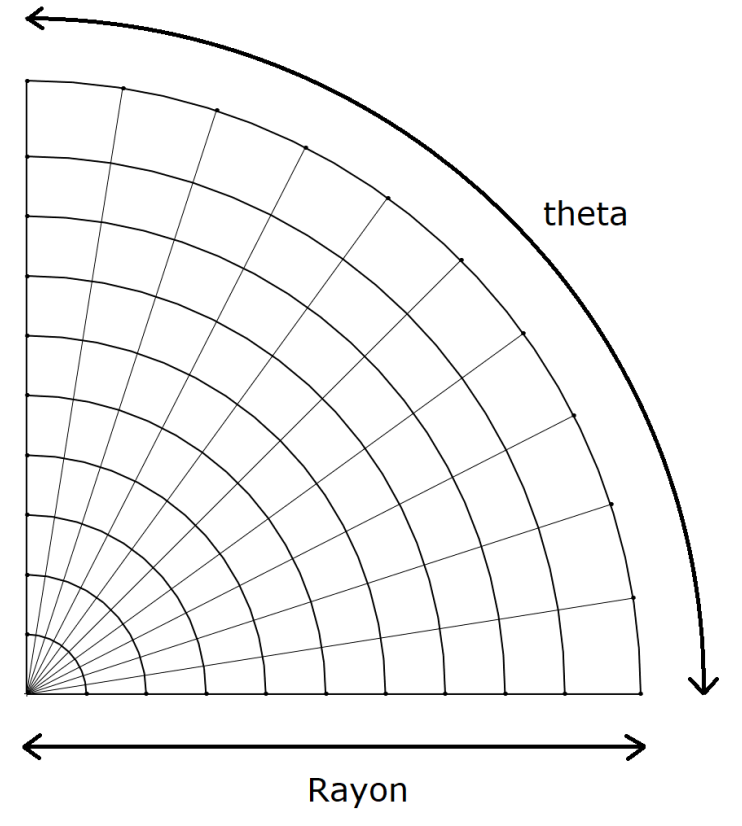


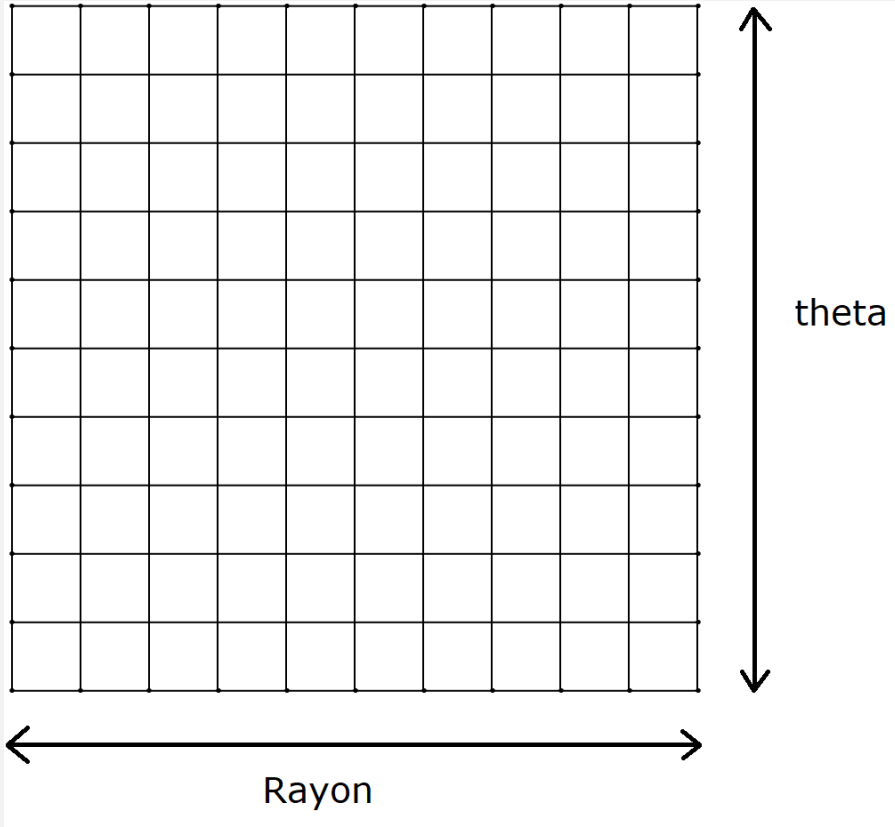


COORDONNÉES POLAIRES

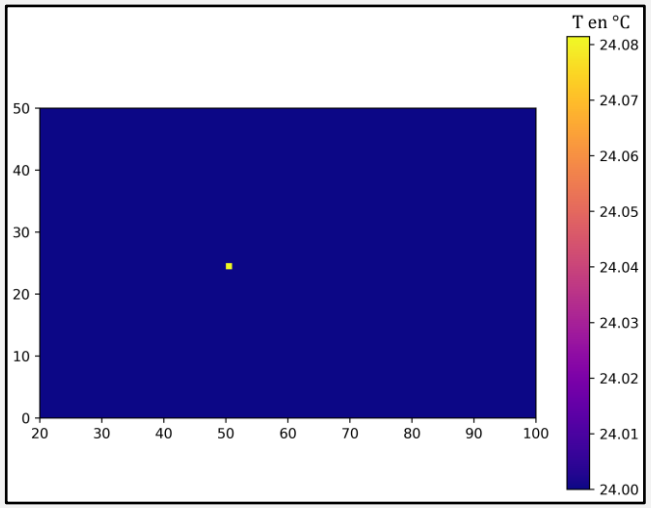
$$\Delta f = \frac{\partial^2 f}{\partial r^2} + \frac{1}{r}\frac{\partial f}{\partial r} + \frac{1}{r^2}\frac{\partial^2 f}{\partial \theta^2}$$

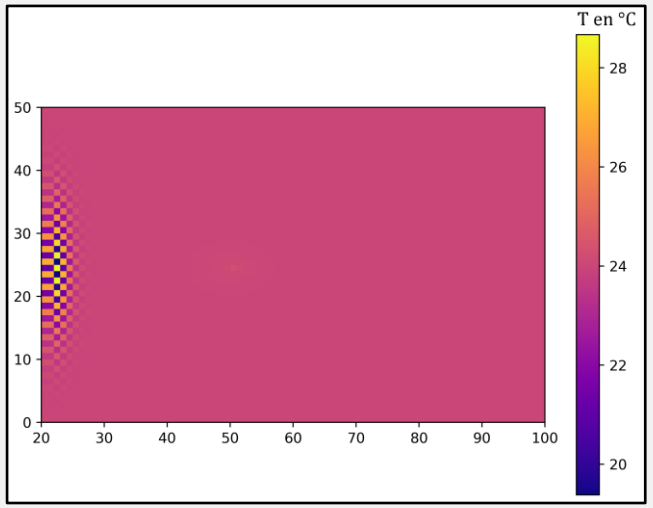
COORDONNÉES POLAIRES

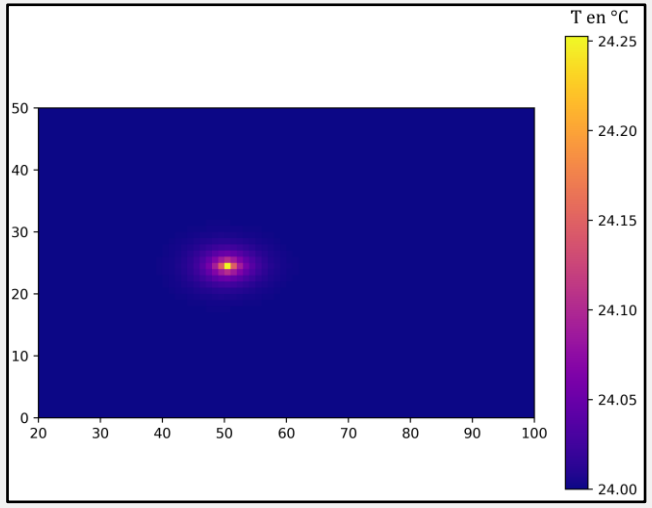


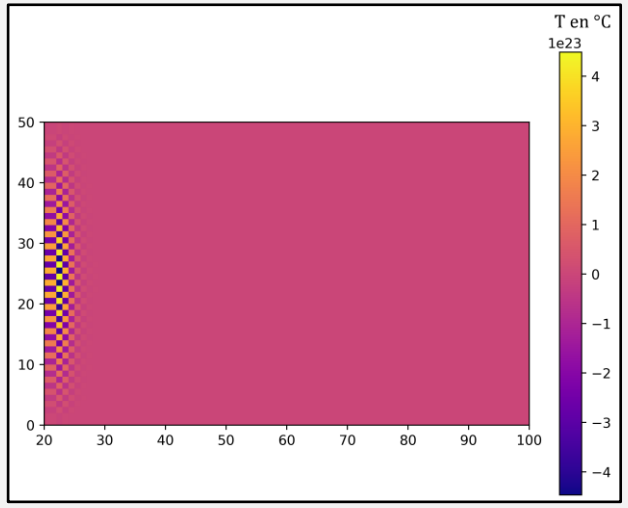


SIMULATION QUI DIVERGE









CORRECTION DU CODE

Paramètres d'influence :

Taille matrice

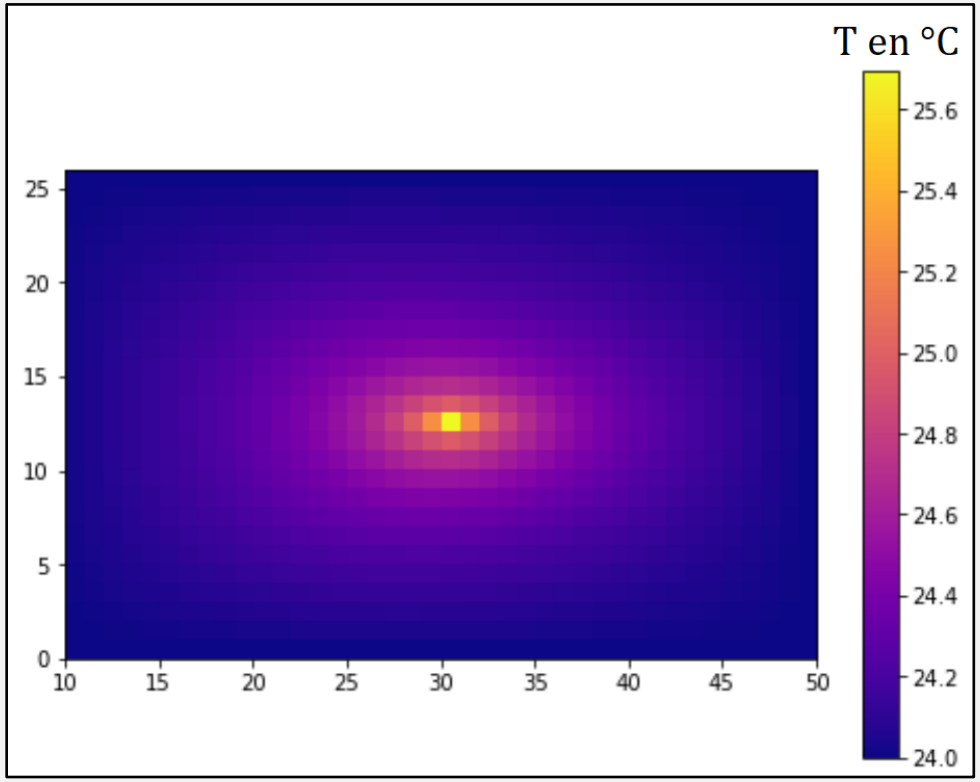
Pas temporelle

Rapport entre taille matrice et pas temporelle

Taille matrice : 50x26

Pas temporelle : 0,01

RÉSULTATS SIMULATION

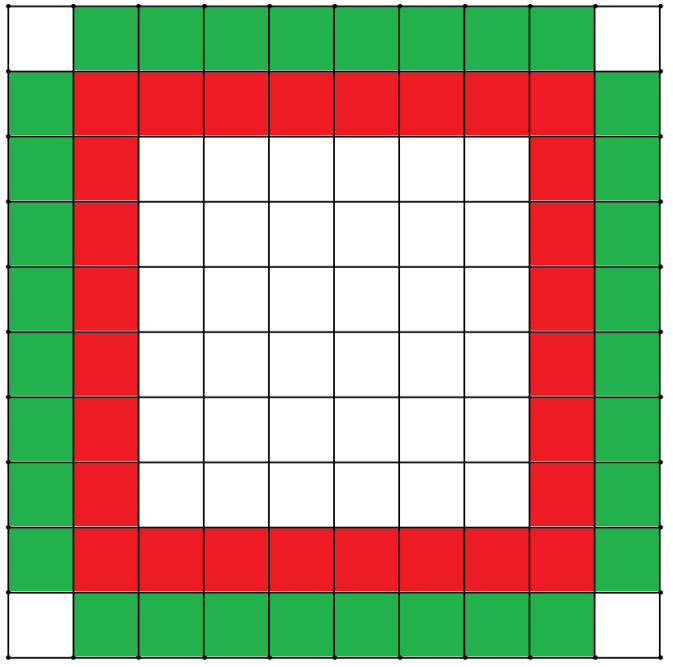


Puissance : 44,6W
Temps : 20 minutes

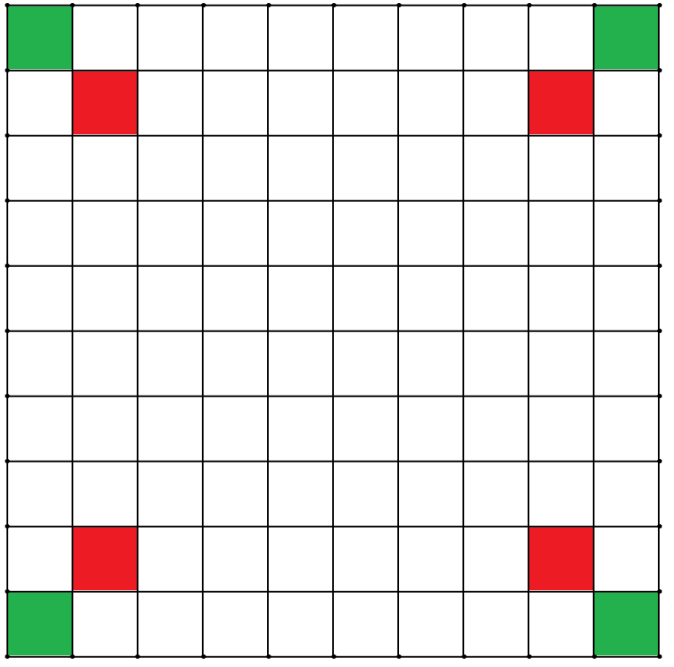
Expérience	Simulation
t1 = 40°C	t1 = 24°C
t5 = 41°C	t5 = 25°C
t10 = 40°C	t10 = 24°C

NOUVELLES CONDITIONS AUX LIMITES

Températures des cases intérieures

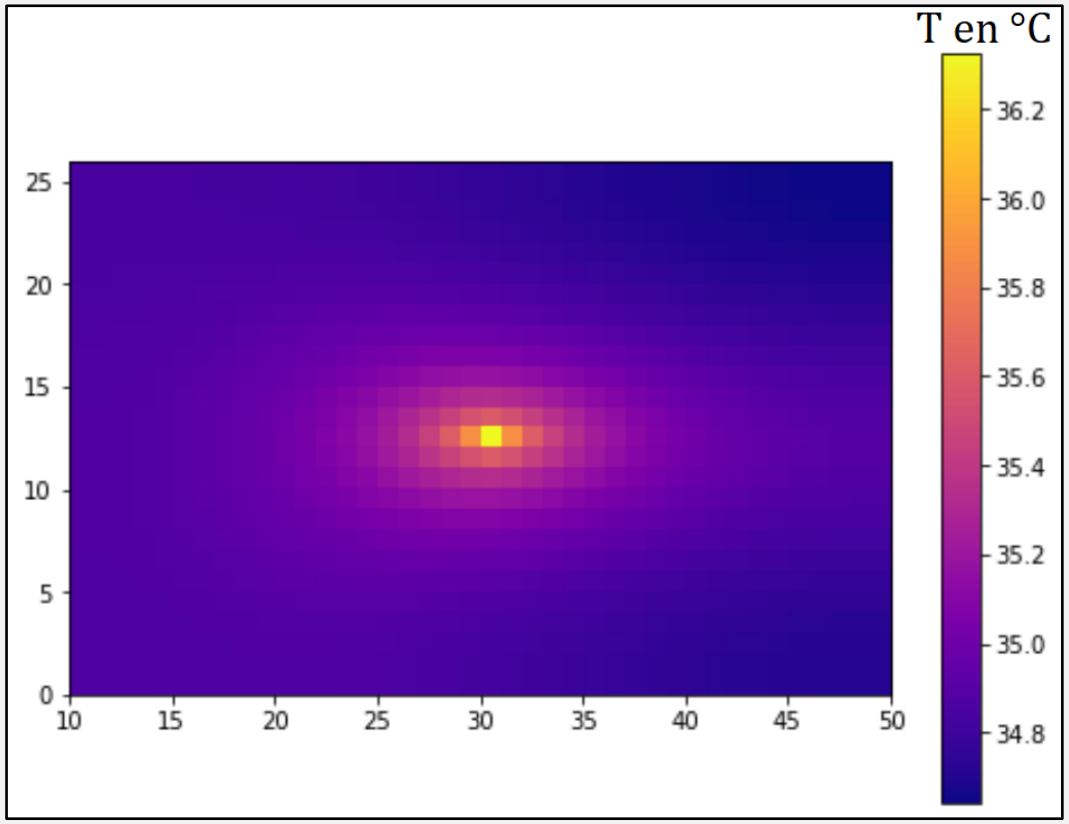


Bords



Coins

RÉSULTATS SIMULATION



Puissance : 44,6W

20 minutes :

Simulation
t1 = 35°C
t5 = 35°C
t10 = 35°C

Expérience
t1 = 40°C
t5 = 41°C
t10 = 40°C

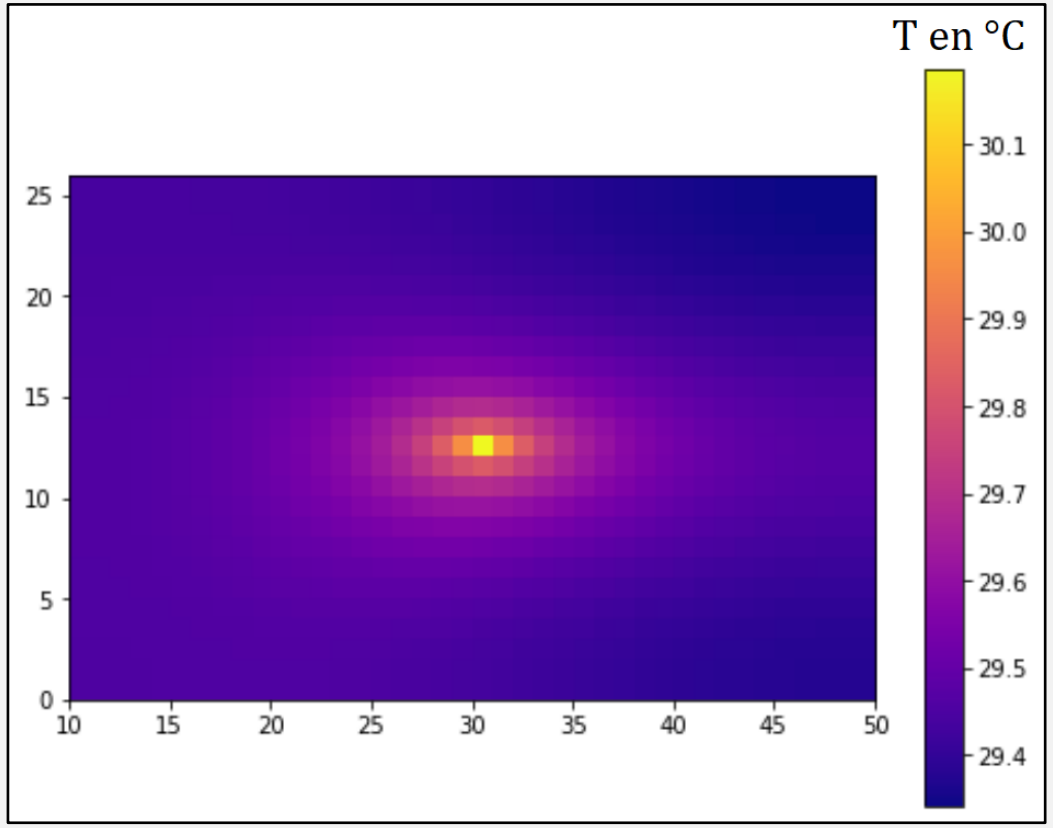
10 minutes :

Simulation
t1 = 30°C
t5 = 30°C
t10 = 30°C

Expérience
t1 = 32°C
t5 = 33°C
t10 = 32°C

Ecart : 2,93°C ± 0,16

RÉSULTATS SIMULATION



Puissance : 22,4W

20 minutes :

Simulation	Expérience
t1 = 29°C	t1 = 35°C
t5 = 30°C	t5 = 36°C
t10 = 29°C	t10 = 34°C

10 minutes :

Simulation	Expérience
t1 = 26°C	t1 = 29°C
t5 = 27°C	t5 = 30°C
t10 = 26°C	t10 = 29°C

Ecart : 3,44°C ± 0,17

SOURCES D'ERREURS

- Flux de chaleur
- Conditions aux limites
- Simulation en 2 dimensions et non en 3 dimensions

SIMULATION

Puissance mise en jeux pour arrêter une voiture de 1,5 tonnes roulant à 50 km/h avec un temps d'arrêt de 1,5 s.

$Em = Ec + Ep$

Avec $Ec = 1/2mv^2$ et $Ep = 0$

$P=Em/t$

P=24kw pour un frein

SIMULATION

Disque :

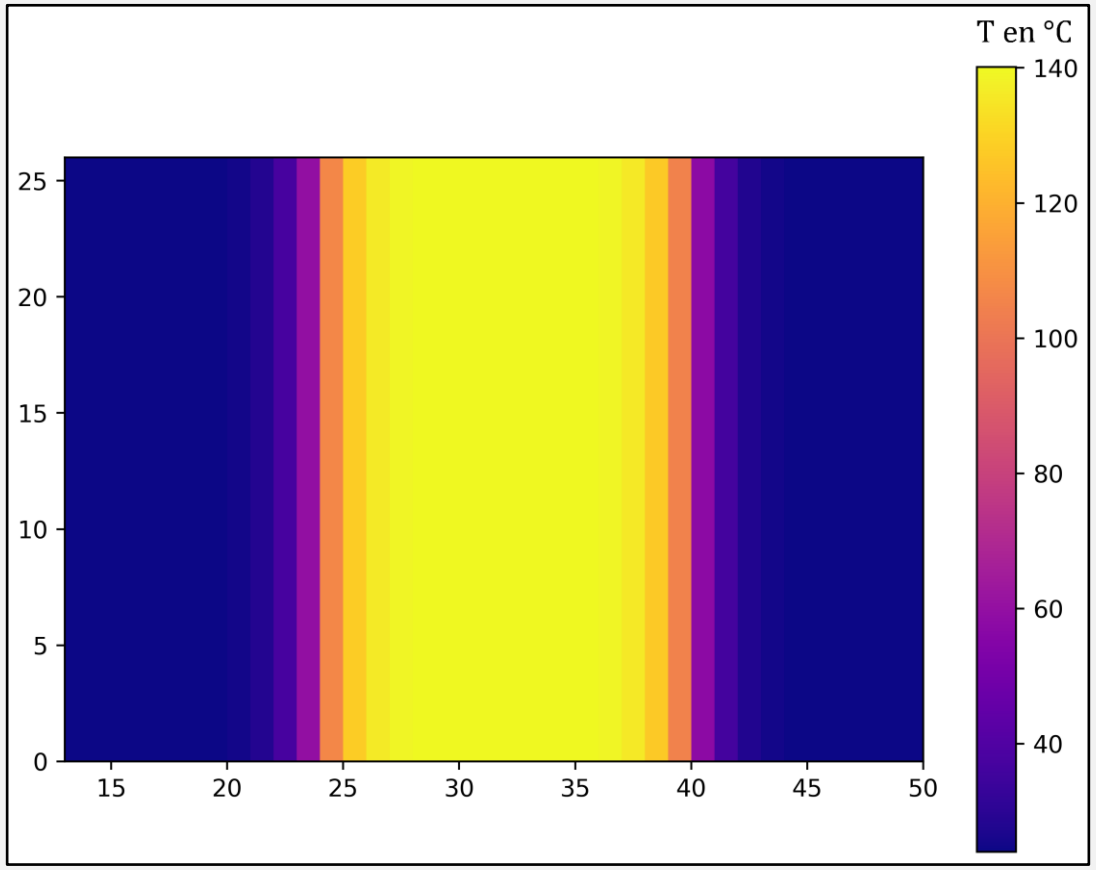
Fonte à graphite sphéroïdal (FGS)

Diamètre : 300 mm

Plaquette :

Volume : $9{,}47*10^{-5}$ m^3

RÉSULTATS SIMULATION



CONCLUSION

- Affiner modèle (3 dimensions, conditions limites…)
- Prévoir surchauffe avec simulation
- Ventilation, matériaux…

ANNEXE : CODE EXPÉRIENCE ARDUINO

```
#include <max6675.h>


int thermoD0 = 24; // so
int thermoCLK = 25; // sck

int thermo1CS = 2;
float temp1 = 0;
MAX6675 thermocouple1(thermoCLK, thermo1CS, thermoD0);

int thermo2CS = 3;
float temp2 = 0;
MAX6675 thermocouple2(thermoCLK, thermo2CS, thermoD0);

int thermo3CS = 4;
float temp3 = 0;
MAX6675 thermocouple3(thermoCLK, thermo3CS, thermoD0);

int thermo4CS = 5;
float temp4 = 0;
MAX6675 thermocouple4(thermoCLK, thermo4CS, thermoD0);

int thermo5CS = 6;
float temp5 = 0;
MAX6675 thermocouple5(thermoCLK, thermo5CS, thermoD0);

int thermo6CS = 7;
float temp6 = 0;
MAX6675 thermocouple6(thermoCLK, thermo6CS, thermoD0);

int thermo7CS = 8;
float temp7 = 0;
MAX6675 thermocouple7(thermoCLK, thermo7CS, thermoD0);

int thermo8CS = 9;
float temp8 = 0;
MAX6675 thermocouple8(thermoCLK, thermo8CS, thermoD0);

int thermo9CS = 10;
float temp9 = 0;
MAX6675 thermocouple9(thermoCLK, thermo9CS, thermoD0);

int thermo10CS = 11;
float temp10= 0;
MAX6675 thermocouple10(thermoCLK, thermo10CS, thermoD0);
```

ANNEXE : CODE EXPÉRIENCE ARDUINO

```
void setup() {
  Serial.begin(115200);
  Serial.setTimeout(1);
}

void loop() {
  temp1 = thermocouple1.readCelsius();
  temp2 = thermocouple2.readCelsius();
  temp3 = thermocouple3.readCelsius();
  temp4 = thermocouple4.readCelsius();
  temp5 = thermocouple5.readCelsius();
  temp6 = thermocouple6.readCelsius();
  temp7 = thermocouple7.readCelsius();
  temp8 = thermocouple8.readCelsius();
  temp9 = thermocouple9.readCelsius();
  temp10 = thermocouple10.readCelsius();
  Serial.print(temp1);
  Serial.print(" ");
  Serial.print(temp2);
  Serial.print(" ");
  Serial.print(temp3);
  Serial.print(" ");
  Serial.print(temp4);
  Serial.print(" ");
  Serial.print(temp5);
  Serial.print(" ");
  Serial.print(temp6);
  Serial.print(" ");
  Serial.print(temp7);
  Serial.print(" ");
  Serial.print(temp8);
  Serial.print(" ");
  Serial.print(temp9);
  Serial.print(" ");
  Serial.print(temp10);
  delay(5000);

}
```

ANNEXE : CODE EXPÉRIENCE PYTHON

```python
import serial
import time
import pandas as pd

arduino = serial.Serial(port='COM5', baudrate=115200, timeout=.1)
Temps_depart=time.time()

#Lecture arduino
#Sortie : bytes
def read():
    data = arduino.readline()
    return data

#Suppression caractère genant sur une chaine de caractère
#Entrée : bytes
#Sortie : string
def supp(a):
    supp="b'na"
    b=str(a)
    for x in range(len(supp)):
        b = b.replace(supp[x],"")
    return (b)

#Supression caractère genant sur des chaines de caractères dans une liste
#Entrée : liste
#Sortie : liste
def supp_liste(a):
    b=[]
    for n in range (len(a)):
        c=a[n]
        d=supp(c)
        b.append(d)
    return (b)

#Suppression chaine de caractère vide dans une liste et position dans la liste d'origne
#Entrée : liste
#Sorties : listes
def supp_vide(a):
    b=[]
    c=[]
    for i in range(len(a)):
        if a[i]=="" or a[i]==" ":
            b.append(i)
        else :
            c.append(a[i])
    return (c,b)

#Suppression temps à la position des temepertatures suprimées de la liste d'origine
#Entrées : listes
#Sortie : lsite
def supp_time(a,b):
    for i in range(len(b)):
        a[b[i]]=""
    c,d=supp_vide(a)
    return(c)

```

ANNEXE : CODE EXPÉRIENCE ARDUINO

```python
#Séparation des temperatures dans 10 listes
#Entrée : liste
#Sorties : listes
def separation(a):
    t1,t2,t3,t4,t5,t6,t7,t8,t9,t10=[],[],[],[],[],[],[],[],[],[]
    for i in range (len(a)):
        b=a[i].split()
        t1.append(float(b[0]))
        t2.append(float(b[1]))
        t3.append(float(b[2]))
        t4.append(float(b[3]))
        t5.append(float(b[4]))
        t6.append(float(b[5]))
        t7.append(float(b[6]))
        t8.append(float(b[7]))
        t9.append(float(b[8]))
        t10.append(float(b[9]))
    return t1,t2,t3,t4,t5,t6,t7,t8,t9,t10

#Création d'une liste contenant les valeurs de l'aduino
#Entrée : entier
#Sorties : listes
def creation_listes_depart(a):
    b=[]
    e=[]
    d=0
    while d<a :
        b.append(read())
        c=time.time()
        d=c-Temps_depart
        e.append(round(d,2))
        time.sleep(5)
    return(b,e)

#Transofrmation d'une liste à 10 listes
def modification_liste(a):
    b=supp_liste(a)
    c,d=supp_vide(b)
    t1,t2,t3,t4,t5,t6,t7,t8,t9,t10=separation(c)
    return d,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10

#Création listes final
#Entrée : entier
#Sorties : listes
def final(temps):
    a,b=creation_listes_depart(temps)
    S,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10=modification_liste(a)
    T=supp_time(b,S)
    return T,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10


T,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10=final(60*60*6)

data = pd.DataFrame({'T':T,'t1':t1,'t2':t2,'t3':t3,'t4':t4,'t5':t5,'t6':t6,'t7':t7,'t8':t8,'t9':t9,'t10':t10})
data.to_excel('Temperatures.xlsx', sheet_name='sheet1', index=False)
```

ANNEXE : CODE SIMULATION CARTÉSIENNES

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

# Constantes et variables

alpha=0.2

t0=20                    #Temperature initiale de la pièce
t1=100                   #Temperature de la case chauffé

nb_cases_x=50
nb_cases_y=50

dx=1                    #Pas d'espace en x
dy=1                   #Pas d'espace en y

position_chaleur_x=25
position_chaleur_y=25


Temps_simu=300             #Temps de l'xperience
dt=1                #Pas temporel

tableau=np.ones((nb_cases_y,nb_cases_x))*t0
tableau[position_chaleur_y,position_chaleur_x]=t1
new_tableau=np.ones((nb_cases_y,nb_cases_x))*t0

#boucle temporelle
for n in range(int(Temps_simu/dt)):

    for y in range (1,nb_cases_y-1):

        for x in range (1,nb_cases_x-1):
            grad=((tableau[y,x+1]-2*tableau[y,x]+tableau[y,x-1])/dx**2)+((tableau[y+1,x]-2*tableau[y,x]+tableau[y-1,x])/dy**2)
            new_tableau[y,x]=tableau[y,x]+dt*alpha*grad

    new_tableau[position_chaleur_y,position_chaleur_x]=t1

    #liste de réécriture pour la nouvelle passe
    for y in range (nb_cases_y):

        for x in range (nb_cases_x):
            tableau[y,x]=new_tableau[y,x]

    fig = plt.figure(figsize=(8,6))
    plt.pcolormesh(tableau,cmap="plasma")
    plt.colorbar()
    plt.show()
```

ANNEXE : CODE SIMULATION POLAIRE

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

#-----------------------------------------------------
# Constantes et variables
#-----------------------------------------------------
#Conductivité thermique en W·m-1·K-1
Lambda=237
#Capacité thermique massique en J K-1 kg-1
Cv=897
#Masse volumique en kg·m-3
rho=2698.9
#Puissance en W
p=44.59
#Volume de la resistance en m3
v=(9*np.pi)/12500000
#Puissance volumique
Pv=p/v
#Constante 1
c1=Lambda/(rho*Cv)
#Constante2
c2=Pv/(rho*Cv)
#Temps en s
Temps_simu=60*20
#Pas temporel
dt=0.001
#Temperature initiale de la pièce
t0=24
#Rayon du disque en m
Rayon=0.2
Nb_cases_rayon=50
Case_rayon_int=int((0.04*Nb_cases_rayon)/Rayon)
#Portion angulaire du disque
Angle=np.pi/2
#Résolution angulaire
Nb_cases_theta=26
#-----------------------------------------------------

#-----------------------------------------------------
#Création liste des rayons
#-----------------------------------------------------
liste_rayon=[]
for i in range (1,Nb_cases_rayon+1):
    liste_rayon.append((Rayon/Nb_cases_rayon)*i)
#-----------------------------------------------------

#-----------------------------------------------------
#Création liste des thetas
#-----------------------------------------------------
liste_theta=[]
for i in range (1,Nb_cases_theta+1):
    liste_theta.append(round((Angle/Nb_cases_theta)*i,10))
#-----------------------------------------------------

```

ANNEXE : CODE SIMULATION POLAIRE

```
56  #-----------------------------------------------------
57  #Calcul des pas
58  #-----------------------------------------------------
59  #Pas angulaire
60  dtheta=(Angle/Nb_cases_theta)
61  #Pas radial
62  dr=(Rayon/Nb_cases_rayon)
63  #-----------------------------------------------------
64
65  #-----------------------------------------------------
66  #Création des matrices
67  #-----------------------------------------------------
68  tableau=np.ones((Nb_cases_theta,Nb_cases_rayon))*t0
69  new_tableau=np.ones((Nb_cases_theta,Nb_cases_rayon))*t0
70  #-----------------------------------------------------
71
72  #-----------------------------------------------------
73  #Position de la case chauffée
74  #-----------------------------------------------------
75  position_chaleur_r=30
76  position_chaleur_theta=13
77  #-----------------------------------------------------
78
79  #-----------------------------------------------------
80  #Creation listes 'Capteurs temperatures', liste temporelle et liste case chauffé
81  #-----------------------------------------------------
82  T,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,=[],[],[],[],[],[],[],[],[],[],[]
83  #-----------------------------------------------------
84
85  #-----------------------------------------------------
86  #Initialisation intervalle temps pour remplisage des listes des 'Capteurs de temperatures'
87  #-----------------------------------------------------
88  Intervalle=0.0
89  #-----------------------------------------------------
90
```

ANNEXE : CODE SIMULATION POLAIRE

```
#-----------------------------------------------------
#Boucle temporelle
#-----------------------------------------------------
for t in range(int(Temps_simu/dt)+1):

    #Boucle des thétas
    for theta in range (1,Nb_cases_theta-1):

        #Boucle des rayons
        for r in range (Case_rayon_int,Nb_cases_rayon-1):
            #calcul en trois temps du gradient
            if theta==position_chaleur_theta and r==position_chaleur_r: #Case "chauffé"
                a=(tableau[theta,r+1]-2*tableau[theta,r]+tableau[theta,r-1])/(dr**2)
                b=(tableau[theta,r+1]-tableau[theta,r])/(dr*liste_rayon[r])
                c=(tableau[theta+1,r]-2*tableau[theta,r]+tableau[theta-1,r])/((dtheta**2)*(liste_rayon[r]**2))
                grad=a+b+c
                new_tableau[theta,r]=tableau[theta,r]+dt*c1*grad+c2*dt
            else:
                a=(tableau[theta,r+1]-2*tableau[theta,r]+tableau[theta,r-1])/(dr**2)
                b=(tableau[theta,r+1]-tableau[theta,r])/(dr*liste_rayon[r])
                c=(tableau[theta+1,r]-2*tableau[theta,r]+tableau[theta-1,r])/((dtheta**2)*(liste_rayon[r]**2))
                grad=a+b+c
                new_tableau[theta,r]=(tableau[theta,r]+dt*c1*grad)


    #Liste de réécriture pour la nouvelle passe
    for theta in range (Nb_cases_theta):

        for r in range (Nb_cases_rayon):
            tableau[theta,r]=new_tableau[theta,r]

    #Remplisage listes 'Capteurs temperatures'
    if round(t*dt,4)==Intervalle:
        Intervalle=Intervalle+1
        T.append(t*dt)
        t1.append(round(tableau[13,12],2))
        t2.append(round(tableau[13,16],2))
        t3.append(round(tableau[13,19],2))
        t4.append(round(tableau[13,23],2))
        t5.append(round(tableau[13,26],2))
        t6.append(round(tableau[13,34],2))
        t7.append(round(tableau[13,37],2))
        t8.append(round(tableau[13,41],2))
        t9.append(round(tableau[13,44],2))
        t10.append(round(tableau[13,48],2))
#-----------------------------------------------------

```

ANNEXE : CODE SIMULATION POLAIRE

```
138  #-----------------------------------------------------
139  #Tableau exel
140  #-----------------------------------------------------
141  data = pd.DataFrame({'T':T,'t1':t1,'t2':t2,'t3':t3,'t4':t4,'t5':t5,'t6':t6,'t7':t7,'t8':t8,'t9':t9,'t10':t10})
142  data.to_excel('Temperatures simulation.xlsx', sheet_name='sheet1', index=False)
143  #-----------------------------------------------------
144
145  #-----------------------------------------------------
146  #Graphique
147  #-----------------------------------------------------
148  fig = plt.figure(figsize=(8,6))
149  plt.imshow(tableau[:,10:],extent=[10,50,0,26],cmap="plasma")
150  plt.colorbar()
151  plt.show()
152  #-----------------------------------------------------
153
```

ANNEXE : CODE SIMULATION POLAIRE

Deuxième conditions aux limites

```
115
116     #Conditions aux limites
117
118     #Limites sauf "coins"
119     for theta in range(1,Nb_cases_theta-1):
120         A=new_tableau[theta,Case_rayon_int+1]
121         new_tableau[theta,Case_rayon_int]=A
122         B=new_tableau[theta,Nb_cases_rayon-2]
123         new_tableau[theta,Nb_cases_rayon-1]=B
124
125     for r in range (Case_rayon_int+1,Nb_cases_rayon-1):
126         C=new_tableau[1,r]
127         new_tableau[0,r]=C
128         D=new_tableau[Nb_cases_theta-2,r]
129         new_tableau[Nb_cases_theta-1,r]=D
130
131     #Limites "coins"
132     E=new_tableau[1,Case_rayon_int+1]
133     new_tableau[0,Case_rayon_int]=E
134
135     F=new_tableau[Nb_cases_theta-2,Case_rayon_int+1]
136     new_tableau[Nb_cases_theta-1,Case_rayon_int]=F
137
138     G=new_tableau[1,Nb_cases_rayon-2]
139     new_tableau[0,Nb_cases_rayon-1]=G
140
141     H=new_tableau[Nb_cases_theta-2,Nb_cases_rayon-2]
142     new_tableau[Nb_cases_theta-1,Nb_cases_rayon-1]=H
143
```

ANNEXE : CODE SIMULATION

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

#-----------------------------------------------------
#Convertir metres en case
#-----------------------------------------------------
def convertir(longueur):
    case=round((longueur*Nb_cases_rayon/Rayon))
    return case
#-----------------------------------------------------

#-----------------------------------------------------
# Constantes et variables
#-----------------------------------------------------
#Conductivité thermique en W·m-1·K-1
Lambda=35
#Capacité thermique massique en J K-1 kg-1
Cv=460
#Masse volumique en kg·m-3
rho=7200
#Puissance en W
p=24110
#Volume de la plaquette en m3 (Epaisseur*Hauteur*Longueur)
v=17*48*116*10**(-9)
#Puissance volumique
Pv=p/v
#Constante 1
c1=Lambda/(rho*Cv)
#Constante2
c2=Pv/(rho*Cv)
#Temps en s
Temps_simu=1.5
#Pas temporel
dt=0.01
#Temperature initiale de la pièce
t0=24
#Rayons du disque en m
Rayon=0.15
Rayon_int=0.04
Nb_cases_rayon=50
#Numero case rayon interieur
Case_rayon_int=convertir(Rayon_int)
#Portion angulaire du disque
Angle=np.pi/2
#Résolution angulaire
Nb_cases_theta=26
#Taille plaquette en m
Taille_plaquette=0.048
#-----------------------------------------------------

```

ANNEXE : CODE SIMULATION

```
#-----------------------------------------------------
#Création liste des rayons
#-----------------------------------------------------
liste_rayon=[]
for i in range (1,Nb_cases_rayon+1):
    liste_rayon.append((Rayon/Nb_cases_rayon)*i)
#-----------------------------------------------------

#-----------------------------------------------------
#Création liste des thetas
#-----------------------------------------------------
liste_theta=[]
for i in range (1,Nb_cases_theta+1):
    liste_theta.append(round((Angle/Nb_cases_theta)*i,10))
#-----------------------------------------------------

#-----------------------------------------------------
#Calcul des pas
#-----------------------------------------------------
#Pas angulaire
dtheta=(Angle/Nb_cases_theta)
#Pas radial
dr=(Rayon/Nb_cases_rayon)
#-----------------------------------------------------

#-----------------------------------------------------
#Création des matrices
#-----------------------------------------------------
tableau=np.ones((Nb_cases_theta,Nb_cases_rayon))*t0
new_tableau=np.ones((Nb_cases_theta,Nb_cases_rayon))*t0
#-----------------------------------------------------

#-----------------------------------------------------
#Position de la case chauffée
#-----------------------------------------------------
Nb_cases_chauffees=convertir(Taille_plaquette)
Jul=Nb_cases_rayon-Case_rayon_int
Jul2=(Jul-Nb_cases_chauffees)/2
print(Nb_cases_chauffees)
Debut_plaquette=round(Case_rayon_int+Jul2)
Fin_plaquette=round(Nb_cases_rayon-Jul2)
print(Debut_plaquette,Fin_plaquette)
#-----------------------------------------------------

#-----------------------------------------------------
#Creation listes 'Capteurs temperatures', liste temporelle et liste case chauffé
#-----------------------------------------------------
T,t1,t2,t3,t4,t5,t6,t7,t8,t9,t1,=[],[],[],[],[],[],[],[],[],[],[],[]
#-----------------------------------------------------
```

ANNEXE : CODE SIMULATION

```
100
101 #-----------------------------------------------------
102 #Boucle temporelle
103 #-----------------------------------------------------
104 for t in range(int(Temps_simu/dt)+1):
105
106     #Boucle des thétas
107     for theta in range (1,Nb_cases_theta-1):
108         #Boucle des rayons
109
110         for r in range (Case_rayon_int,Debut_plaquette):
111             #calcul en trois temps du gradient
112             a=(tableau[theta,r+1]-2*tableau[theta,r]+tableau[theta,r-1])/(dr**2)
113             b=(tableau[theta,r+1]-tableau[theta,r])/(dr*liste_rayon[r])
114             c=(tableau[theta+1,r]-2*tableau[theta,r]+tableau[theta-1,r])/((dtheta**2)*(liste_rayon[r]**2))
115             grad=a+b+c
116             new_tableau[theta,r]=(tableau[theta,r]+dt*c1*grad)
117
118         for r in range (Debut_plaquette,Fin_plaquette):
119             #calcul en trois temps du gradient
120             a=(tableau[theta,r+1]-2*tableau[theta,r]+tableau[theta,r-1])/(dr**2)
121             b=(tableau[theta,r+1]-tableau[theta,r])/(dr*liste_rayon[r])
122             c=(tableau[theta+1,r]-2*tableau[theta,r]+tableau[theta-1,r])/((dtheta**2)*(liste_rayon[r]**2))
123             grad=a+b+c
124             new_tableau[theta,r]=tableau[theta,r]+dt*c1*grad+c2*dt
125
126         for r in range (Fin_plaquette,Nb_cases_rayon-1):
127             a=(tableau[theta,r+1]-2*tableau[theta,r]+tableau[theta,r-1])/(dr**2)
128             b=(tableau[theta,r+1]-tableau[theta,r])/(dr*liste_rayon[r])
129             c=(tableau[theta+1,r]-2*tableau[theta,r]+tableau[theta-1,r])/((dtheta**2)*(liste_rayon[r]**2))
130             grad=a+b+c
131             new_tableau[theta,r]=(tableau[theta,r]+dt*c1*grad)
132
133
```

ANNEXE : CODE SIMULATION

```
132
133
134         #Conditions aux limites
135
136         #Limites sauf "coins"
137         for theta in range(1,Nb_cases_theta-1):
138             A=new_tableau[theta,Case_rayon_int+1]
139             new_tableau[theta,Case_rayon_int]=A
140             B=new_tableau[theta,Nb_cases_rayon-2]
141             new_tableau[theta,Nb_cases_rayon-1]=B
142
143         for r in range (Case_rayon_int+1,Nb_cases_rayon-1):
144             C=new_tableau[1,r]
145             new_tableau[0,r]=C
146             D=new_tableau[Nb_cases_theta-2,r]
147             new_tableau[Nb_cases_theta-1,r]=D
148
149         #Limites "coins"
150         E=new_tableau[1,Case_rayon_int+1]
151         new_tableau[0,Case_rayon_int]=E
152
153         F=new_tableau[Nb_cases_theta-2,Case_rayon_int+1]
154         new_tableau[Nb_cases_theta-1,Case_rayon_int]=F
155
156         G=new_tableau[1,Nb_cases_rayon-2]
157         new_tableau[0,Nb_cases_rayon-1]=G
158
159         H=new_tableau[Nb_cases_theta-2,Nb_cases_rayon-2]
160         new_tableau[Nb_cases_theta-1,Nb_cases_rayon-1]=H
161
162
163
164         #Liste de réécriture pour la nouvelle passe
165         for theta in range (Nb_cases_theta):
166
167             for r in range (Nb_cases_rayon):
168                 tableau[theta,r]=new_tableau[theta,r]
169
170     #-----------------------------------------------------
171
172     #-----------------------------------------------------
173     #Graphique
174     #-----------------------------------------------------
175     fig = plt.figure(figsize=(8,6))
176     plt.imshow(tableau[:,Case_rayon_int:],extent=[Case_rayon_int,Nb_cases_rayon,0,Nb_cases_theta],cmap="plasma")
177     plt.colorbar()
178     plt.savefig("simulation" + ".png",dpi=500)
179     plt.show()
180     #-----------------------------------------------------
```