

Caractérisation et optimisation d'un processus de chauffage micro-ondes

Études et exploitation
des inhomogénéités du champ micro-ondes

Motivation et point de départ de l'étude

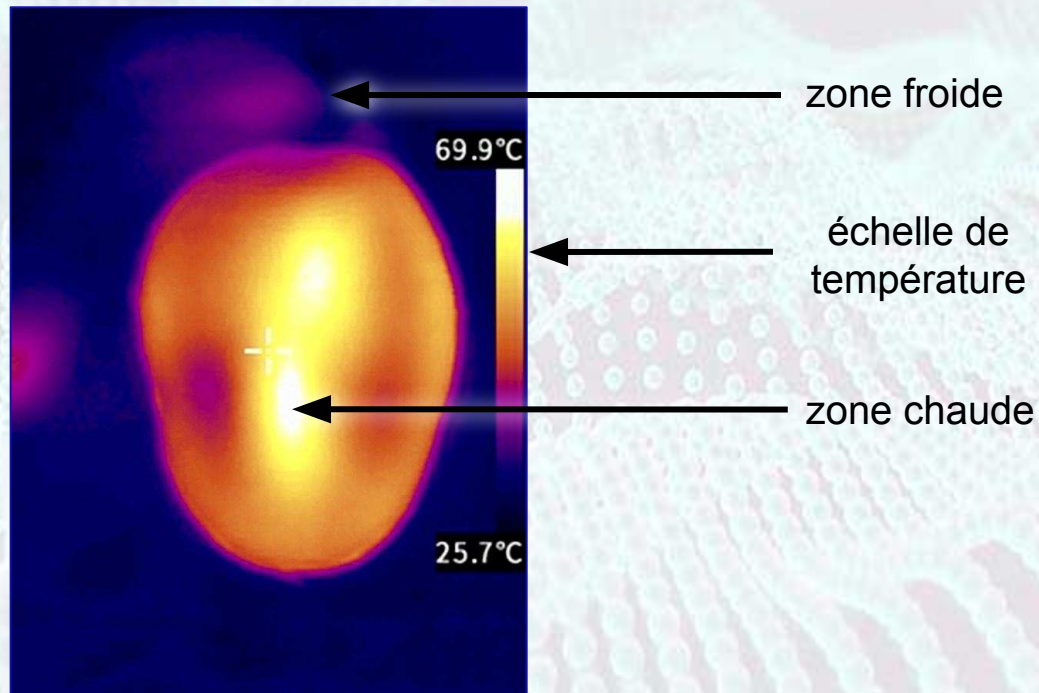


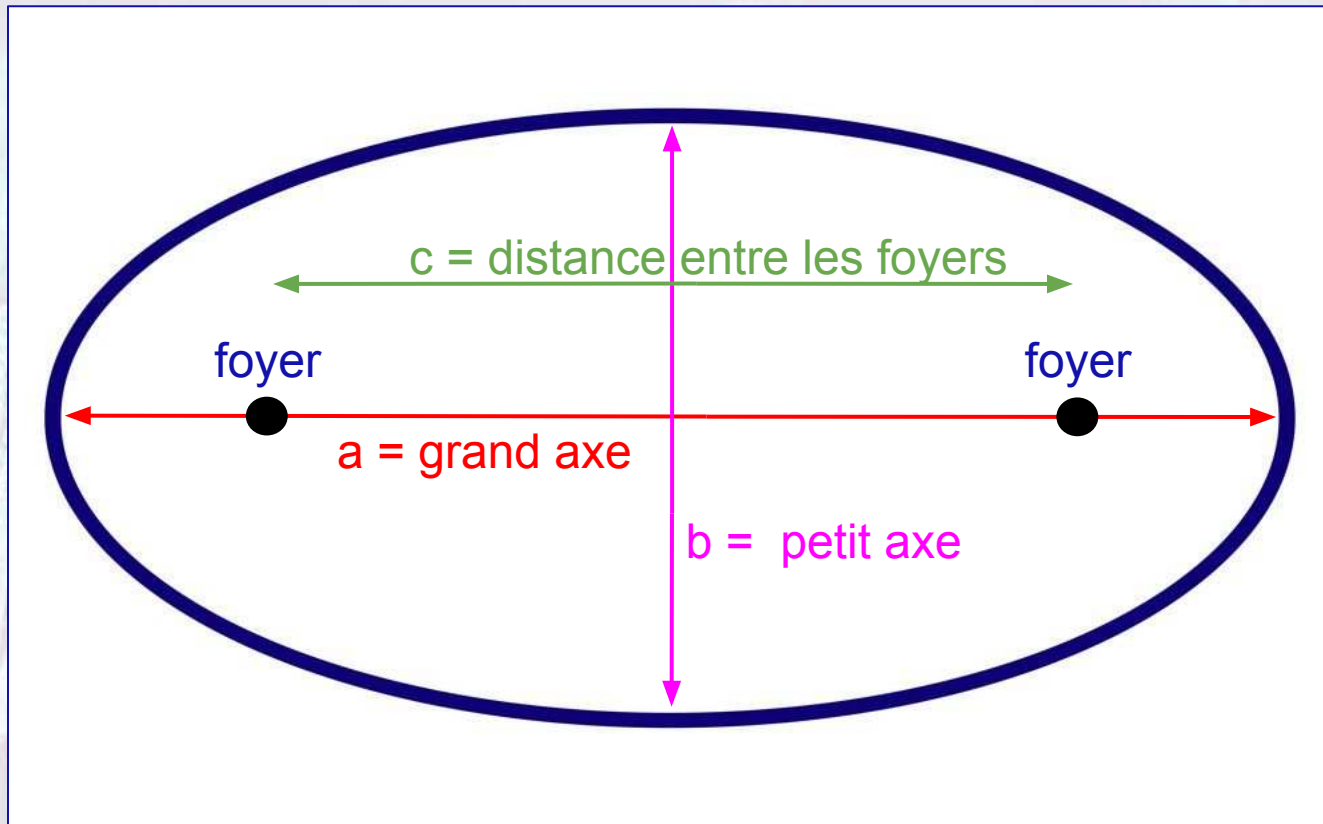
*photo de l'expérience
que nous avons reproduite*

Sommaire

- I - Invalidation de la théorie des foyers**
- II - Réalisation de guides d'ondes reproductibles**
- III - Recherche de solutions analytiques**
- IV - Validation expérimentale des modèles analytiques**
- V - Étude énergétique**
- VI - Application à une synthèse chimique au micro-ondes**

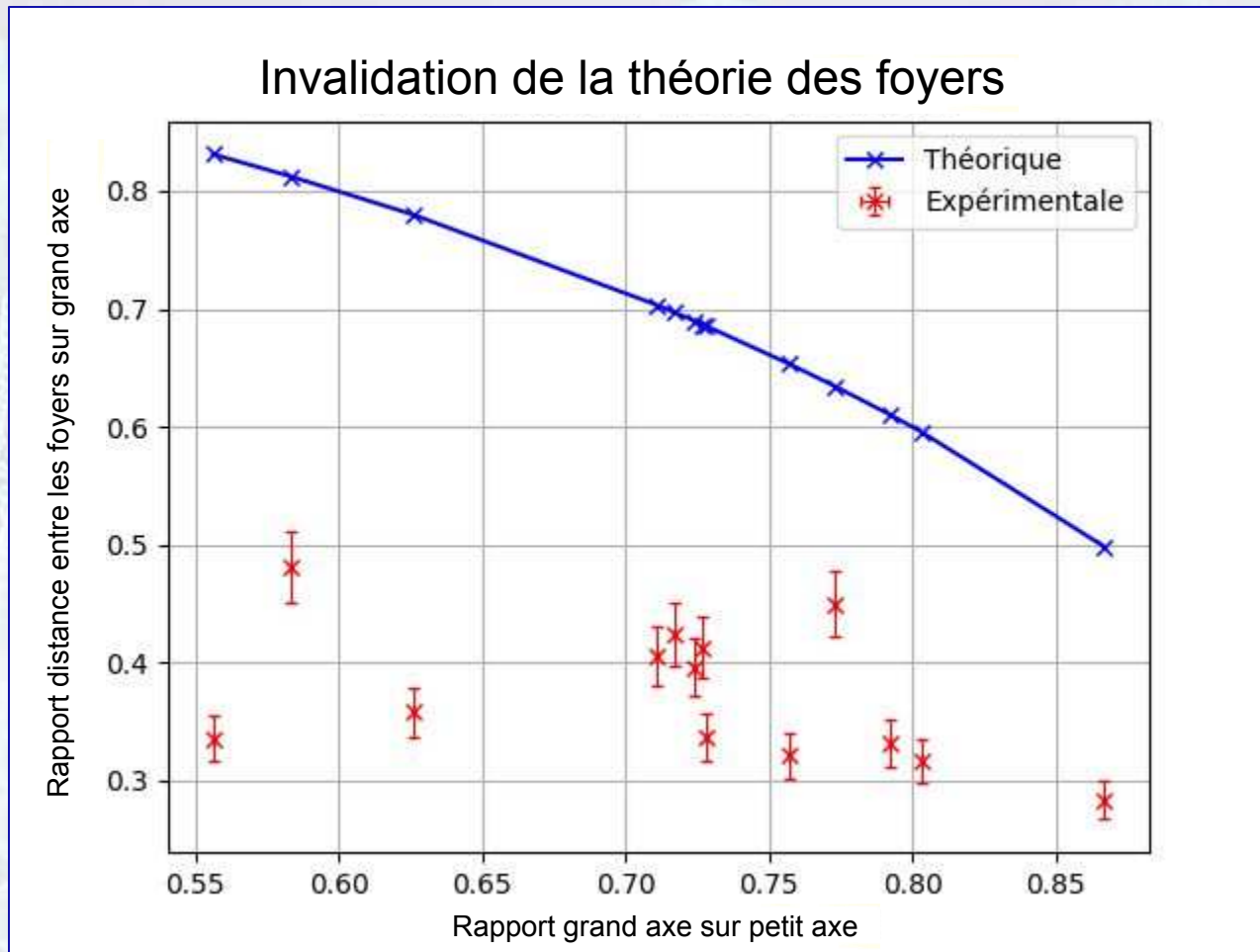
I - Invalidation de la théorie des foyers

Invalidation de la théorie des foyers



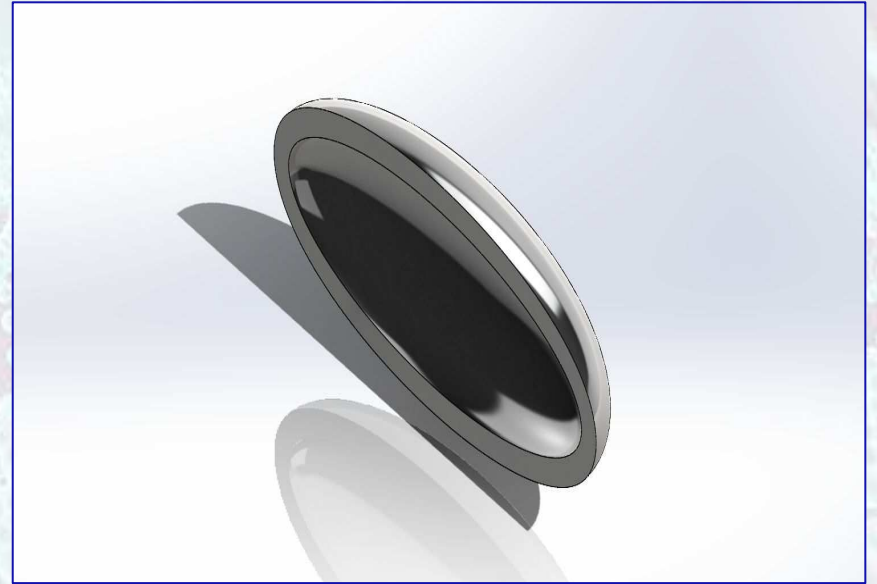
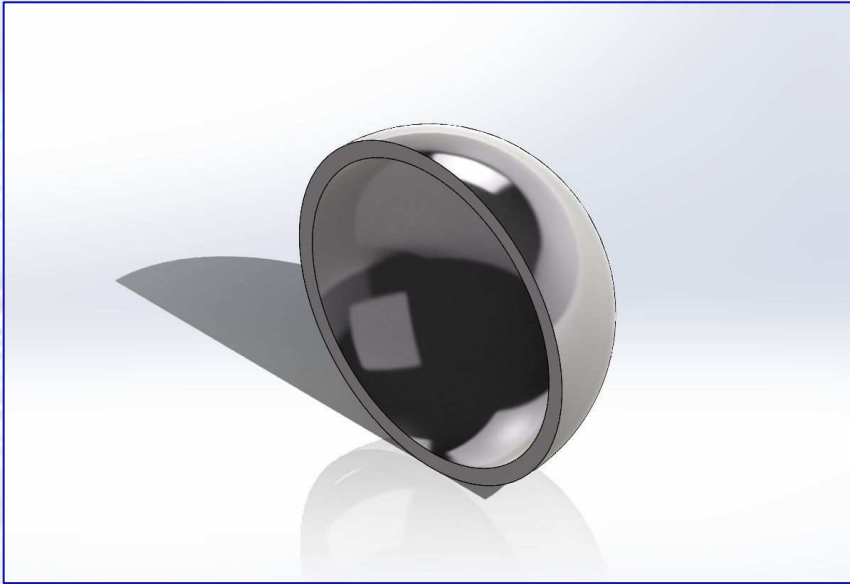
Relation entre les différents paramètres : $\sqrt{\left(\frac{c}{a}\right)^2 + \left(\frac{b}{a}\right)^2} = 1 \Rightarrow \frac{c}{a} = \sqrt{1 - \left(\frac{b}{a}\right)^2}$

Invalidation de la théorie des foyers



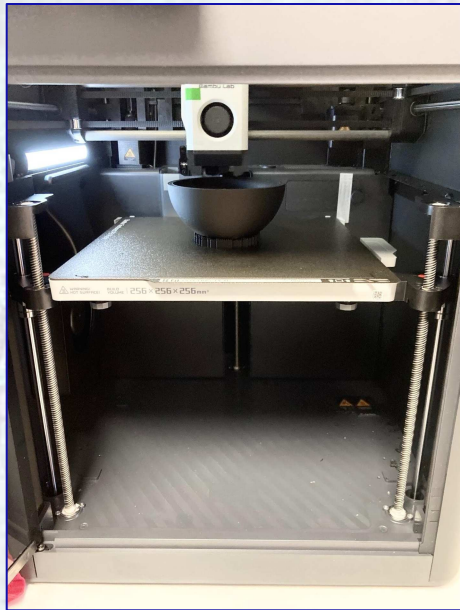
II - Réalisation de guides d'ondes reproductibles

Réalisation de guides d'ondes reproductibles



images de la modélisation des moules

Réalisation de guides d'ondes reproductibles



impression 3D des coques



différentes formes de coques



*pied pour stabiliser la coque
lors du moulage*

Réalisation de guides d'ondes reproductibles

Solution d'agar-agar à 20 g.L⁻¹.

Certaines propriétés physiques de l'agar-agar sont proches de celles de l'eau.



*moule
rempli*



résultat



*conservation
de la forme sphérique*

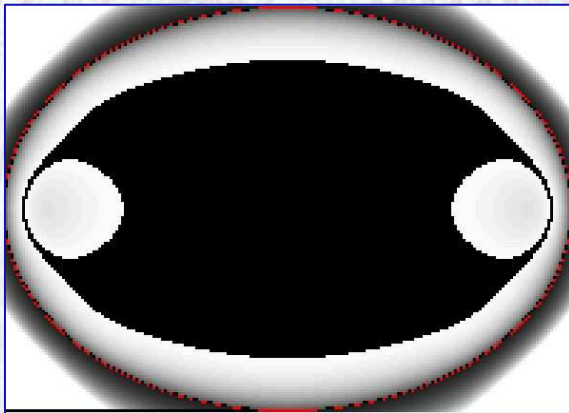
III - Recherche de solutions analytiques

Recherche de solutions analytiques

$$\frac{\partial^2 E(r, z, t)}{\partial r^2} + \frac{1}{r} \frac{\partial E(r, z, t)}{\partial r} + \frac{\partial E(r, z, t)}{\partial z} = -\mu \varepsilon \omega^2 E(r, z, t)$$

Solution développable en série entière, tronquée à l'ordre 10 et vérifiant la condition à la limite.

Résolution de l'équation matricielle par notre code Python.



*représentation du champ électrique
trouvé comme solution par notre code*

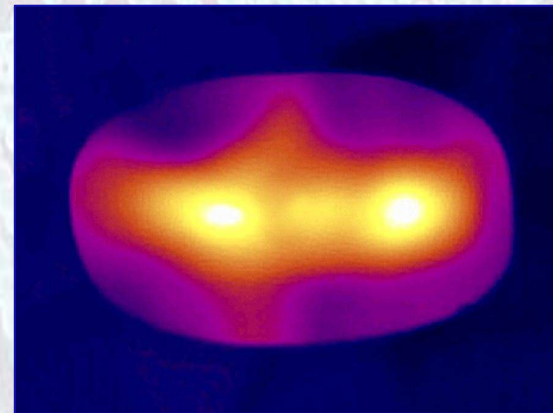
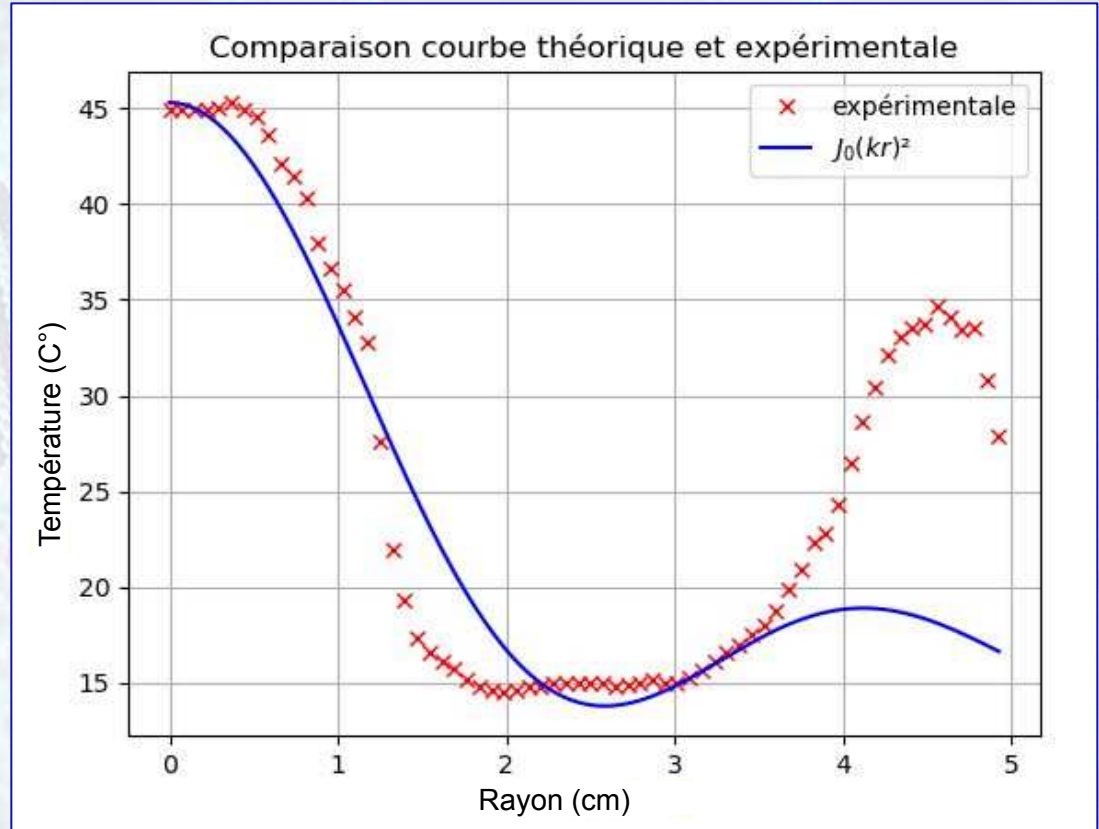
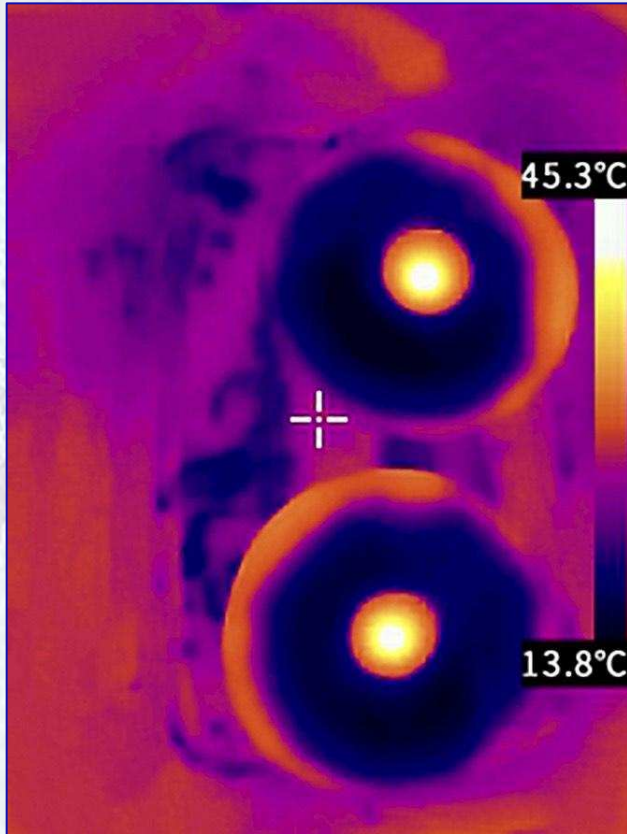


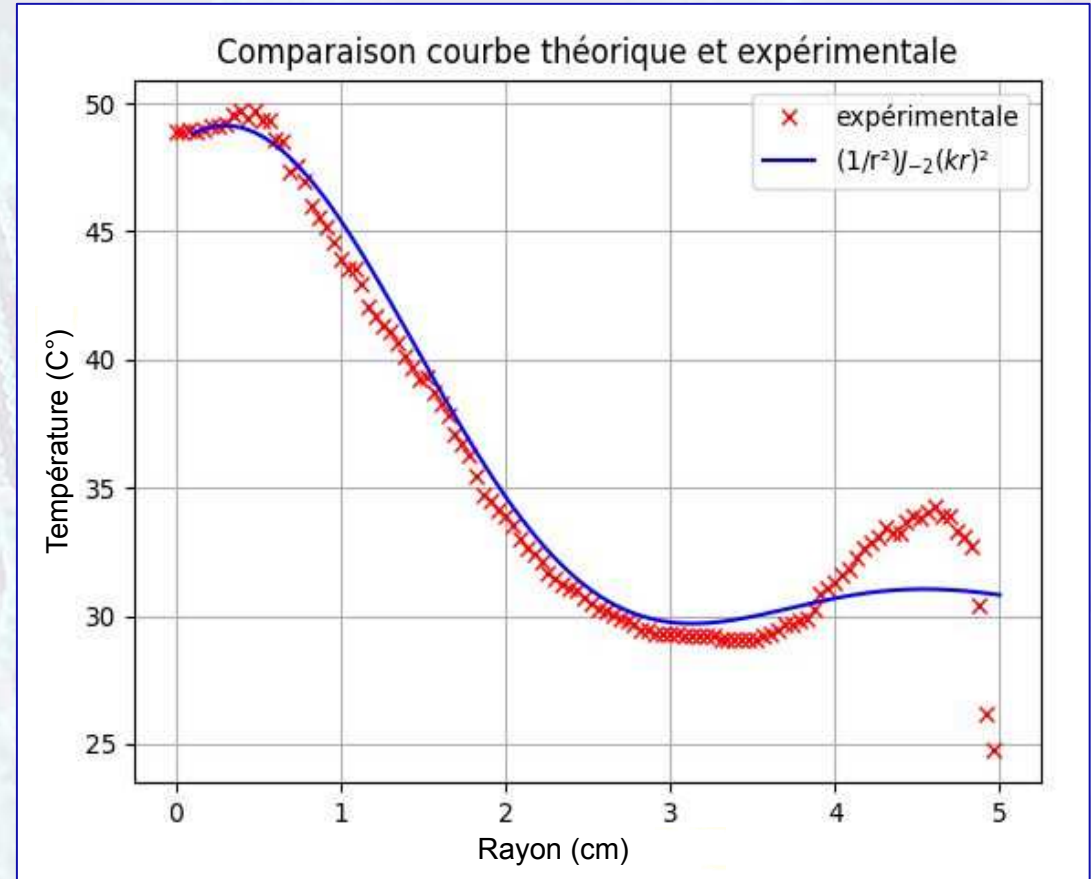
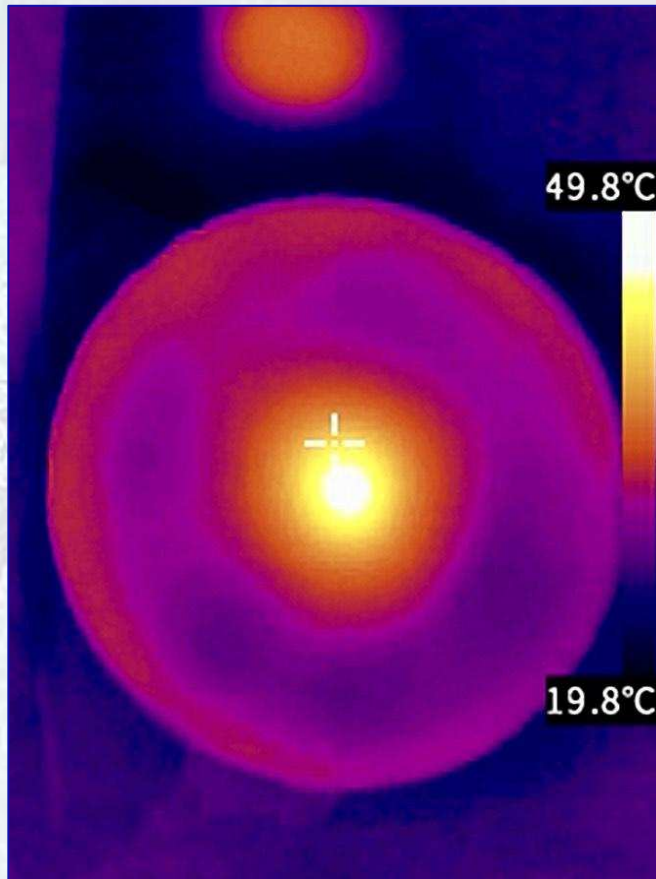
image du profil de température dans notre expérience

IV - Validation expérimentale des modèles analytiques

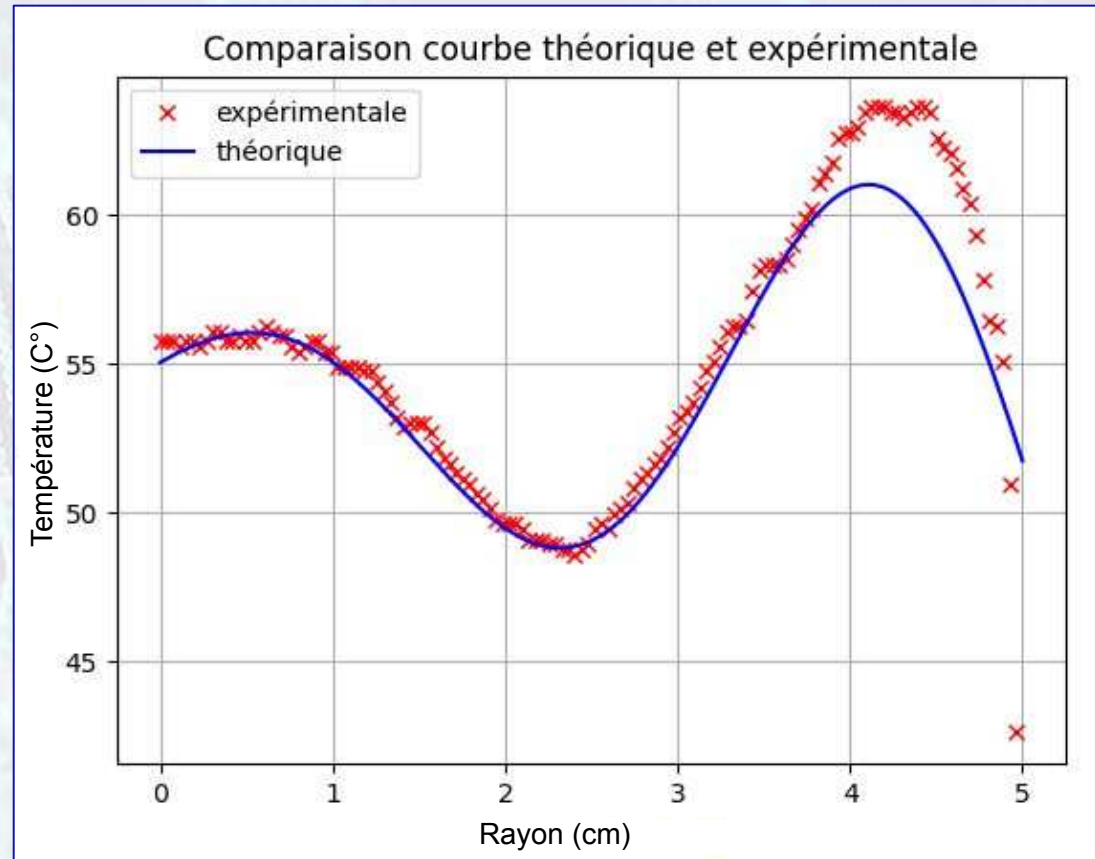
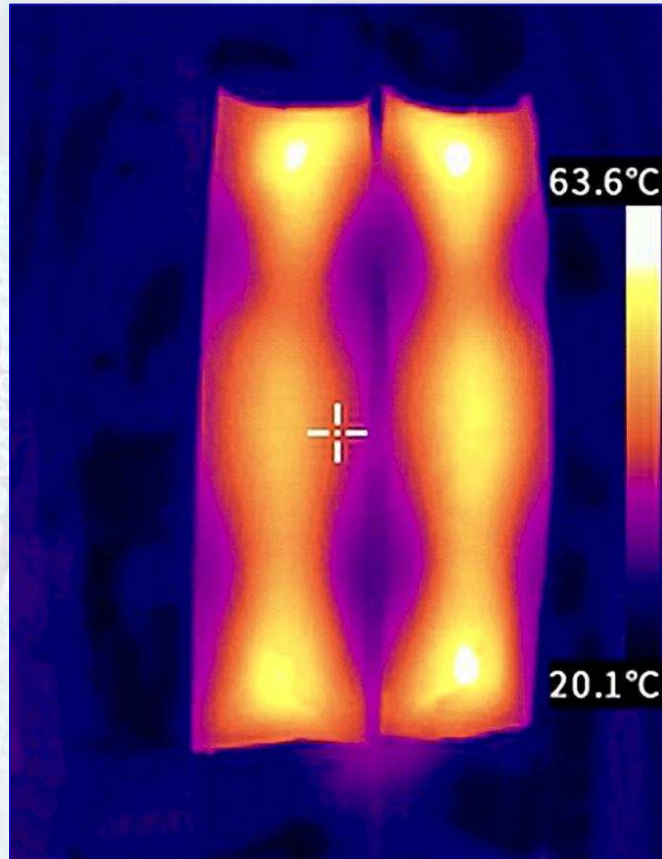
Validation du modèle calculé pour des sphères



Validation du modèle calculé pour des cylindres



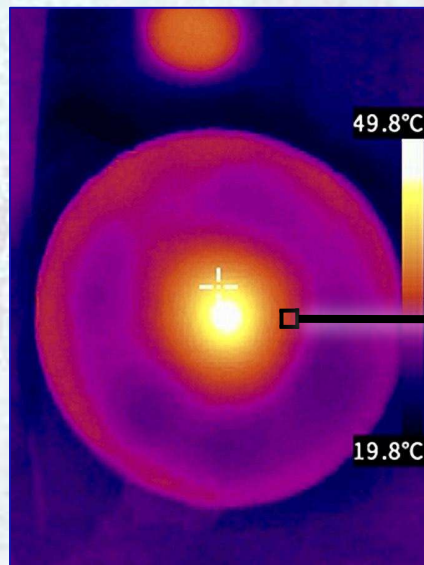
Validation du modèle calculé pour des cylindres



V - Étude énergétique

Étude énergétique

Fonctionnement de notre code Python :



Conversion couleur/
température grâce à l'
échelle de température

Utilisation de la
symétrie
sphérique du
guide d'ondes

Sommation du premier
principe de la
thermodynamique sur
chaque coque

Variation d'énergie
interne totale dans
le guide d'ondes

Étude énergétique

Résultats du code Python sur plusieurs formes de guide d'ondes :

Cylindre épais : $u = 3,4 \text{ J.cm}^{-3}$

Cylindre fin : $u = 5,1 \text{ J.cm}^{-3}$

Sphère : $u = 4,0 \text{ J.cm}^{-3}$

Ellipsoïde : $u = 7,1 \text{ mJ.cm}^{-3}$

Il y aurait donc une forme permettant une chauffe optimale à volume donné.



VI - Application à une synthèse chimique au micro-ondes

Synthèse chimique au micro-ondes

Idée de synthèse micro-ondes :

	Protocole B	Protocole C
	Synthèse au four à micro-ondes	Synthèse avec un appareil de Dean-Stark
Modifications de l'étape 2	Chauffage avec une puissance de 800 W pendant 30 s.	Chauffage à l'aide de l'appareil de Dean-Stark permettant d'extraire l'eau au cours de sa formation, en présence de cyclohexane jouant le rôle de solvant.
Rendement	87 %	85 %

extrait du sujet B du Bac 2023

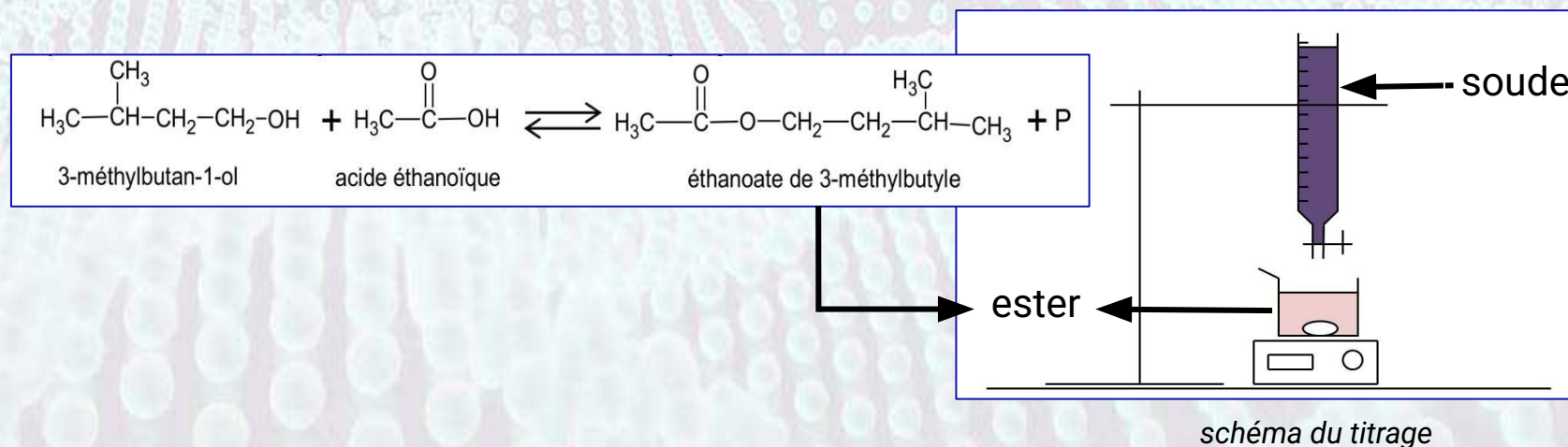
Synthèse chimique au micro-ondes

Protocole :

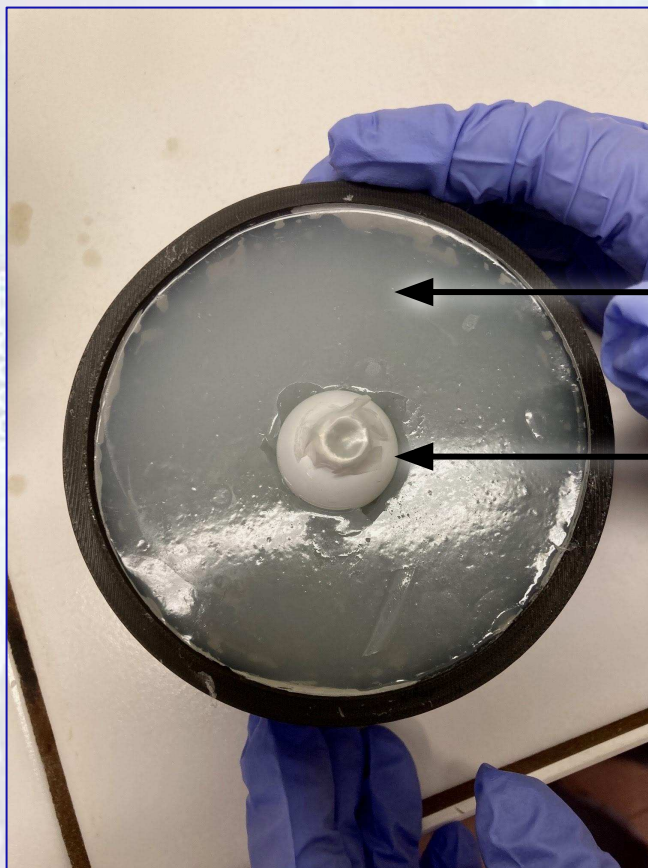
Estérification de Fischer avec de l'acide éthanoïque et du 3 méthylbutan-1-ol.

Chauffage pendant 1 minute à 570 W au micro-ondes.

Titration de l'acide restant par la soude avec un indicateur coloré.



Synthèse chimique au micro-ondes

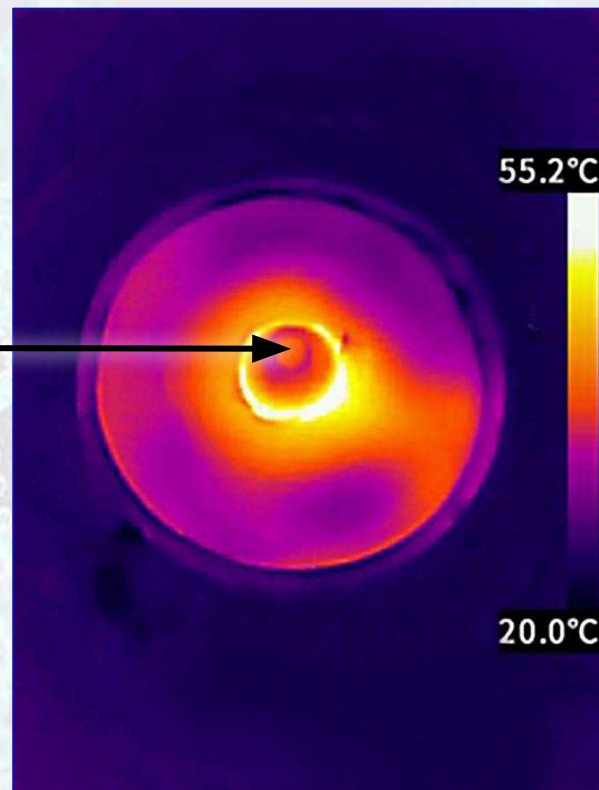


guide d'ondes

boule de guirlande en verre
contenant les réactifs

Synthèse chimique au micro-ondes

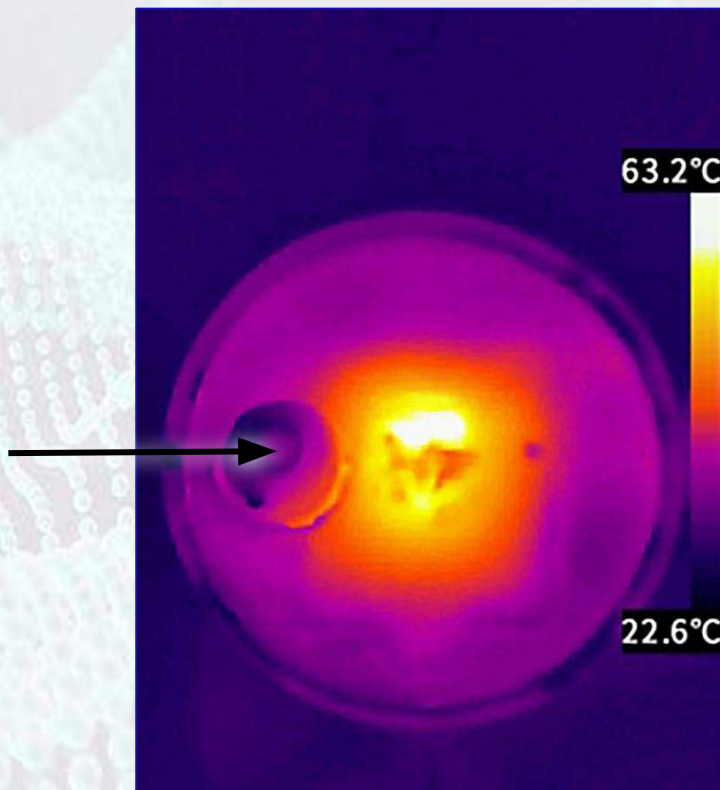
réactifs dans la boule de
guirlande au centre
du guide d'ondes



Rendement de 75 % dans cette configuration.

Synthèse chimique au micro-ondes

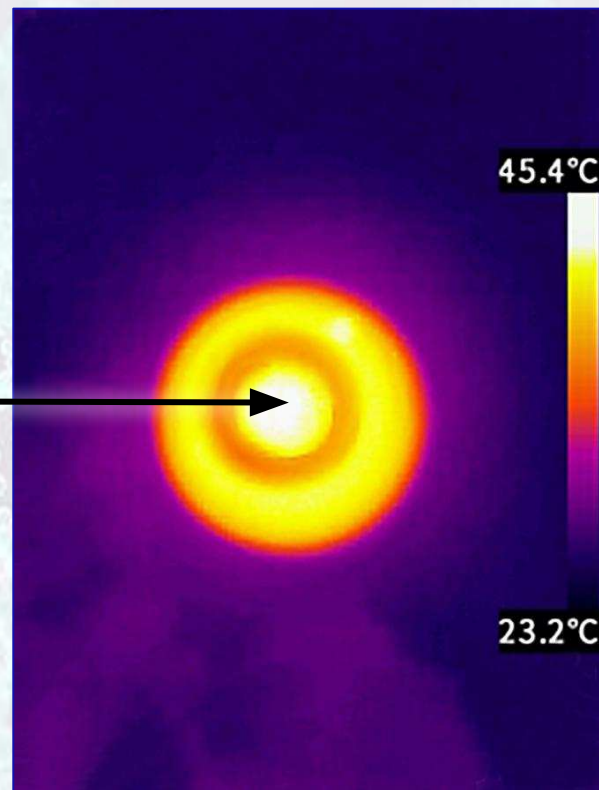
réactifs dans la boule de
guirlande au bord du guide
d'ondes



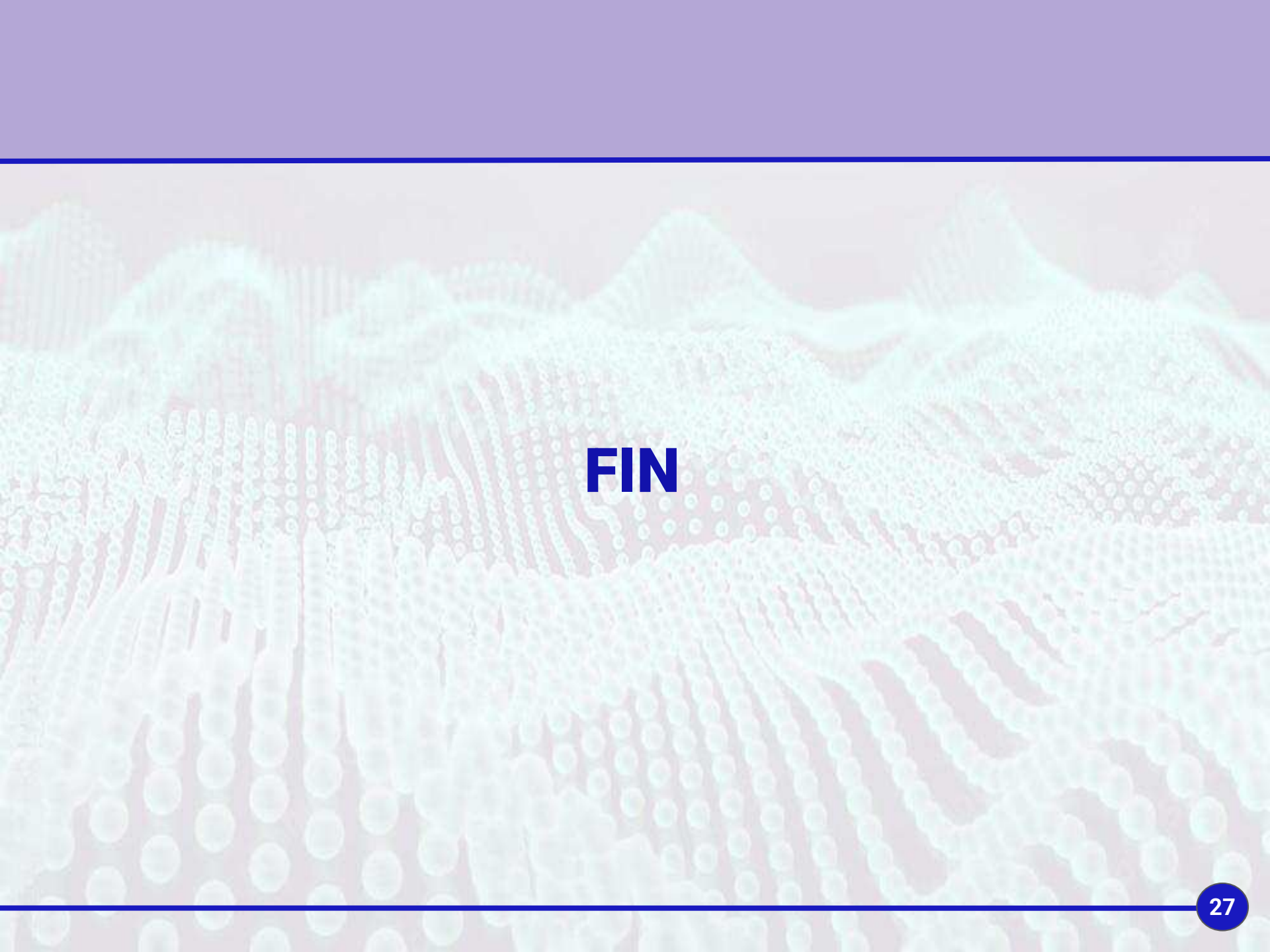
Rendement de 49 % dans cette configuration.

Synthèse chimique au micro-ondes

réactifs dans la boule
de guirlande sans
guide d'ondes



Rendement de 60 % dans cette configuration.



FIN

Annexes

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3
4 def graphique(data):
5     r_vals = np.array([triplet[0] for triplet in data])
6     d_vals = np.array([triplet[1] for triplet in data])
7     t_vals = np.array([triplet[2] for triplet in data])
8
9     y_exp = d_vals
10    y_theo = np.sqrt(1 - r_vals**2)
11    incertitude_d = 0.15 * np.mean(d_vals)
12    dy = np.abs(d_vals / np.sqrt(1 - d_vals**2)) * incertitude_d
13    plt.errorbar(r_vals, y_exp, yerr=dy, xerr=0, fmt='x', color='red', ecolor='red', elinewidth=0.8, capsize=2,
14    label='Expérimentale')
15    plt.plot(r_vals, y_theo, color='blue', marker='x', label='Théorique')
16
17    plt.xlabel("Rapport petit axe sur grand axe")
18    plt.ylabel("Rapport distance entre les foyers sur grand axe")
19    plt.title("Invalidation de la théorie des foyers")
20    plt.legend()
21    plt.grid(True)
22    plt.show()

```

code traçant la courbe d'invalidation de la théorie des foyers

```

1 from math import *
2 from PIL import Image as Img
3 import numpy as np
4 import scipy.integrate as integrate
5 import scipy.special as sp
6
7 kv=pi
8 N=10
9 tableau=np.full((N,N,N),np.array(N*[0]))
10 E0=10
11 a=7
12 b=5
13
14 for k in range(0,N):
15     tableau[0][k]=np.array((k*[0]+[1]+(N-k-1)*[0]))*20000
16 for j0 in range(N-1):
17     j=j0
18     i=1
19     while j>=0:
20         tableau[i][j]=-((2*j+2)*(2*j+1)/(i*2+2)**2)*tableau[i-1][j+1]-(kv**2/(i*2+2)**2)*tableau[i-1][j]
21         j+=-1
22         i+=1
23
24 def f(k,r,z):
25     mat=np.transpose(np.array(((k)*[0]+[1]+(N-k-1)*[0])))
26     S=0
27     for i in range(N):
28         for j in range(N):
29             S+=np.matmul(tableau[i][j],mat)*(r**(2*i))*(z**(2*j))
30     return(S)
31 print(tableau)
32
33 def A():
34     A=np.zeros((N,N))
35     for i in range(N):
36         for j in range(N):
37             A[i][j]=integrate.quad(lambda r: f(i,r,a*sqrt(1-(r/b)**2))*f(j,r,a*sqrt(1-(r/b)**2)),0,b)[0]
38     return(A)

```

code simulant la solution du champ électrique (partie 1)

```

40 def B():
41     B=np.zeros((N,1))
42     for k in range(N):
43         B[k][0]=integrate.quad(lambda r: f(k,r,a*sqrt(1-(r/b)**2))*E0,0,b)[0]
44     return(B)
45
46 coeffs=np.linalg.solve(A(),B())
47 print(coeffs)
48
49 def E(r,z):
50     S=0
51     for k in range(N):
52         S+=coeffs[k]*f(k,r,z)
53     return(S)
54
55 def simu(N):
56     L=N
57     C=int((b/a)*N)
58     image=np.zeros([L+1,C+1]+[3],dtype='uint8')
59     for l in range(0,int((L+1)/2)+1):
60         for c in range(0,int((C+1)/2)+1):
61             r=abs((C/2-c))*(b/C)*2
62             z=abs((L/2-l))*(a/L)*2
63             val=abs(E(r,z))
64             if val>255:
65                 val=255
66             image[l][c]= val
67     for l in range(int((L+1)/2)+1):
68         for c in range(C,int((C+1)/2),-1):
69             image[l][c]=image[l][int(C/2-(c-C/2))]
70     for l in range(int((L+1)/2),L):
71         for c in range(C):
72             image[l][c]=image[int(L/2-(l-L/2))][c]
73     for l in range(L+1):
74         for c in range(C+1):
75             if ((c-C/2)/(C/2))**2+((l-L/2)/(L/2))**2<=1+0.01 and ((c-C/2)/(C/2))**2+((l-L/2)/(L/2))**2>=1-0.01:
76                 image[l][c]=(255,0,0)
77             if ((c-C/2)/(C/2))**2+((l-L/2)/(L/2))**2<1:
78                 val=image[l][c]*150
79                 image[l][c]=val
80     retour=Img.fromarray(image)
81     retour.show()

```

code simulant la solution du champ électrique (partie 2)

```

1 import matplotlib.pyplot as plt
2 import numpy as np
3 from scipy.special import jn # Fonction de Bessel d'ordre n
4 from math import *
5 from PIL import Image as Img
6
7 ##création de la liste de tuples (température, rayon)
8 Tmin=13.8 # en °C
9 Tmax=45.3 # en °C
10 larg,long=10,5 #largeur et longueur réelle de la photo en cm
11
12 def liste(image):
13     matimage=np.array(image)
14     L,C=matimage.shape[0],matimage.shape[1]
15     x=larg/C #largeur réelle d'un pixel en cm
16     y=long/L #longueur réelle d'un pixel en cm
17     matimagegris=np.zeros([L,C], dtype='uint8')
18     cmin,cmax=matimagegris[0,0],matimagegris[0,0]
19     for i in range(L):
20         for j in range(C):
21             matimagegris[i,j]=int(matimage[i,j][0]/2 + matimage[i,j][1]/2)
22             if matimagegris[i,j]>cmax:
23                 cmax=matimagegris[i,j]
24             if matimagegris[i,j]<cmin:
25                 cmin=matimagegris[i,j]
26     imagegris=Img.fromarray(matimagegris)
27     V=[]
28     for i in range(L):
29         index_ligne = L-i-1
30         if 0 <= index_ligne < L:
31             valeur_gris = matimagegris[index_ligne, C // 2]
32             temperature = ((Tmax - Tmin) / (cmax - cmin)) * valeur_gris + (Tmax - ((Tmax - Tmin) / (cmax - cmin)) * cmax)
33             V.append((temperature, i*y))
34     return(V)
35
36 def tracer_points(l: list, n: int = 0):
37     Y = []
38     X = []
39     for el in l:
40         Y.append(el[0])
41         X.append(el[1])
42
43     plt.figure()
44     plt.plot(X, Y, 'rx', label='expérimentale')
45     x_vals = np.linspace(min(X), max(X), 400)
46     y_bessel_squared = (Tmax-Tmin)*(jn(n, 0.93*x_vals)**2+Tmin
47     plt.plot(x_vals, y_bessel_squared, 'b-', label=f'$J_{n}(kr)^2$')
48
49     plt.xlabel("rayon")
50     plt.ylabel("T")
51     plt.title("Comparaison courbe théorique et expérimentale")
52     plt.legend()
53     plt.grid(True)
54     plt.show()

```

code traçant la courbe comparaison théorique et expérimentale

```

1 from math import *
2 from PIL import Image as Img
3 import numpy as np
4
5 ##création d'une image en nuances de gris
6
7 image=Img.open("image puissance test.jpg")
8 matimage=np.array(image)
9
10 L,C=matimage.shape[0],matimage.shape[1]
11
12 matimagegris=np.zeros([L,C], dtype='uint8')
13
14 cmin,cmax=matimagegris[0,0],matimagegris[0,0]
15
16 for i in range(L):
17     for j in range(C):
18         matimagegris[i,j]=int(matimage[i,j][0]/2 + matimage[i,j][1]/2)
19         #définition des couleurs extrémales de l'image
20         if matimagegris[i,j]>cmax:
21             cmax=matimagegris[i,j]
22         if matimagegris[i,j]<cmin:
23             cmin=matimagegris[i,j]
24
25 imagegris=Img.fromarray(matimagegris)
26 #imagegris.show()
27
28 ##conversion de la couleur en température
29
30 Tmin=20          # en °C
31 Tmax=60          # en °C
32
33 #conversion de la couleur du pixel en température à l'aide d'une fonction affine  $T(\text{couleur})=a*\text{couleur}+b$  et telle que
34  $T_{\text{max}}=a*c_{\text{max}}+b$  ;  $T_{\text{min}}=a*c_{\text{min}}+b$  d'où :
35  $a=((T_{\text{max}}-T_{\text{min}})/(c_{\text{max}}-c_{\text{min}}))$ 
36  $b=(T_{\text{max}}-a*c_{\text{max}})$ 
37
38 ##intégration de l'énergie dans la boule
39
40 larg,long=0.03,0.1      #largeur et longueur réelle de la photo en m
41 x=larg/C                #largeur réelle d'un pixel en m
42 y=long/L                #longueur réelle d'un pixel en m
43 Ceau=4.18*(10**3)       #J.kg-1.K-1
44 peau=1000               #kg.m-3
45 U=0
46
47 for i in range(L):
48     for j in range(C):
49         T=a*matimagegris[i,j]+b          #calcul de la température du pixel
50         U+=(Ceau)*(T-Tmin)*(2*pi*(j*x))*(x*y)*peau      #calcul de l'énergie interne d'un anneau
51
52 U=C*delta(T)*2*pi*rayon*surface_pixel*peau
53 print("la variation d'énergie interne de l'ellipsoïde est de",U,"J")

```

code calculant la variation d'énergie interne dans les guides d'ondes