

# Incorporation des matériaux à changement de phase dans l'isolation thermique d'un bâtiment

---

VIELLEPEAU Axel  
TIPE 2025



## Introduction – Contexte

- ❖ Augmentation des températures dû à la crise climatique
- ❖ Importance du secteur du bâtiment concernant les gaz à effet de serre
- ❖ Système de climatisation active énergivore

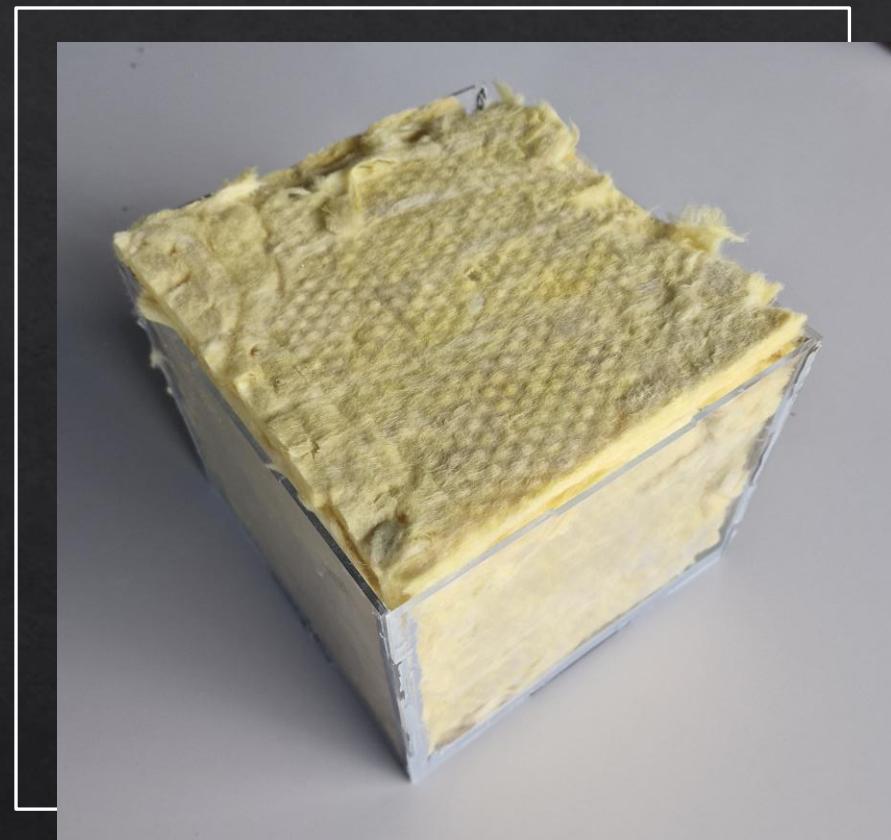


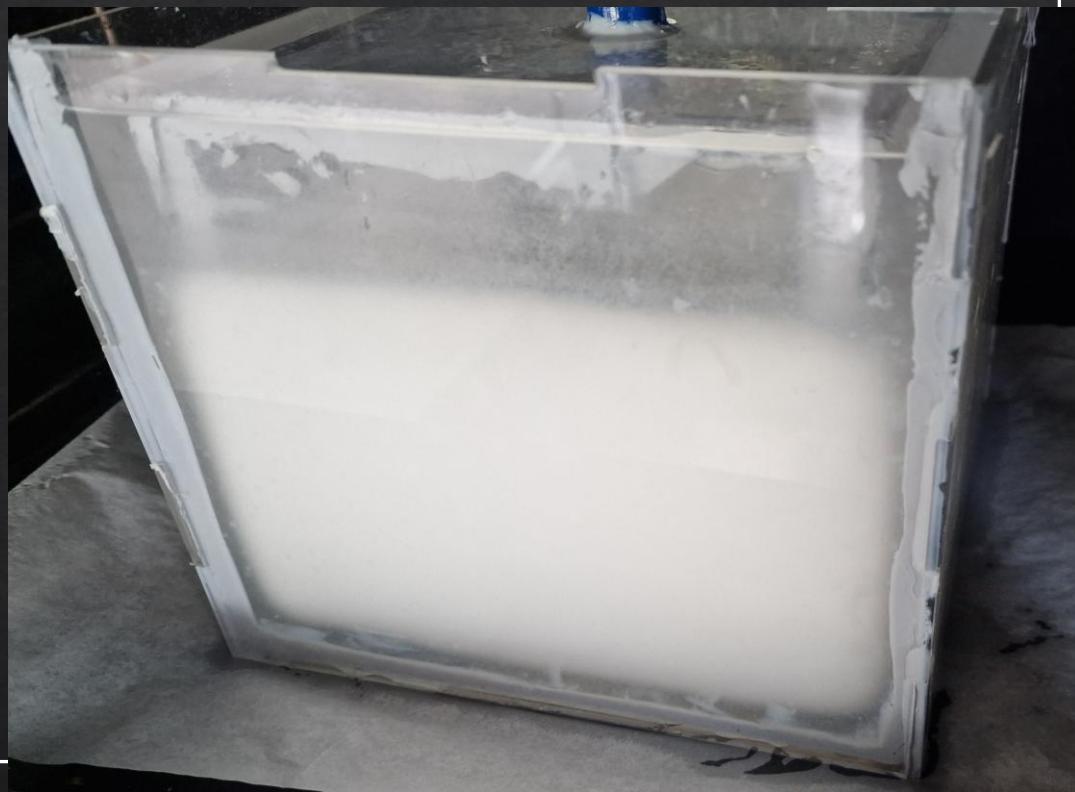
Problématique :  
Comment améliorer le confort  
thermique d'un bâtiment grâce aux  
matériaux à changement de phase  
?

## Les Objectifs

---

- étudier l'inertie thermique avec le MCP
- Réaliser une maquette de pièce isolé avec et sans MCP
- Développé un modèle numérique de l'équation de la chaleur incorporant des MCP





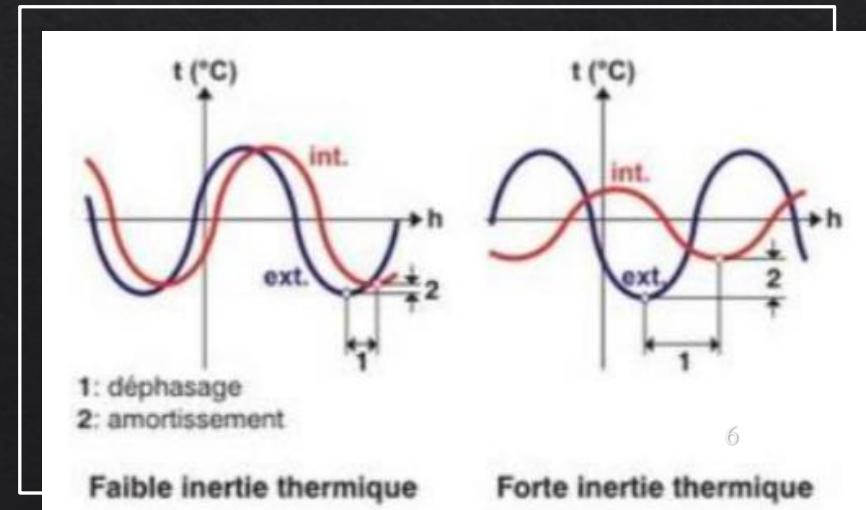
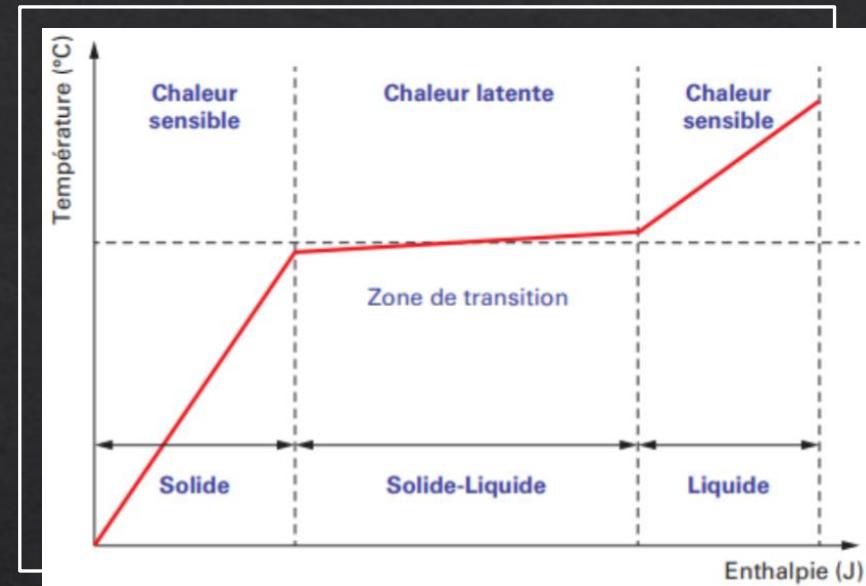
# Plan d'étude

---

- 1.La Théorie des MCP
2. Caractérisation du MCP
3. Expérimentation
- 4.Mise en équation
- 5.Modèle numérique
- 6.conclusion

## Le MCP dans tous ses états

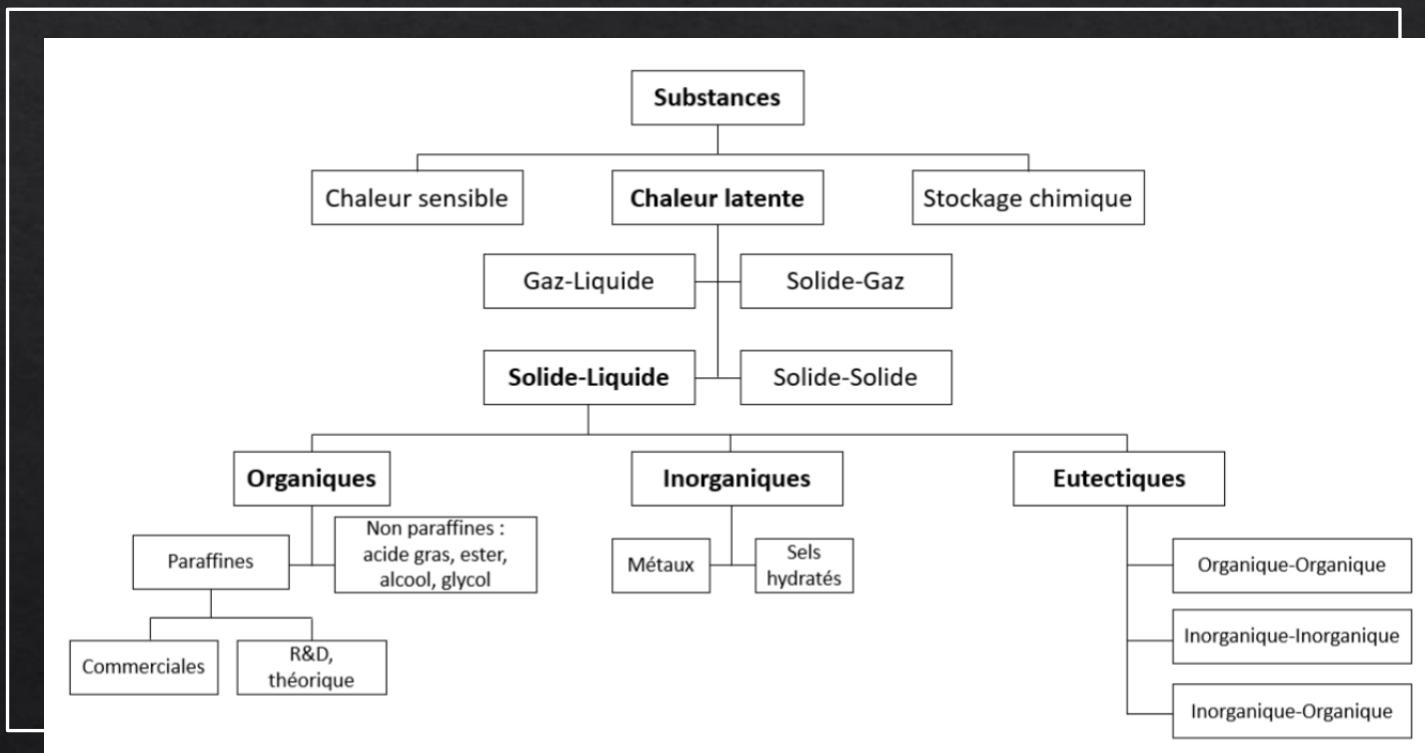
- ❖ Chaleur sensible/chaleur latente
- ❖ Changement de température jour/nuit
- ❖ Augmentation de l'inertie thermique



## Choix du MCP

Paraffine :

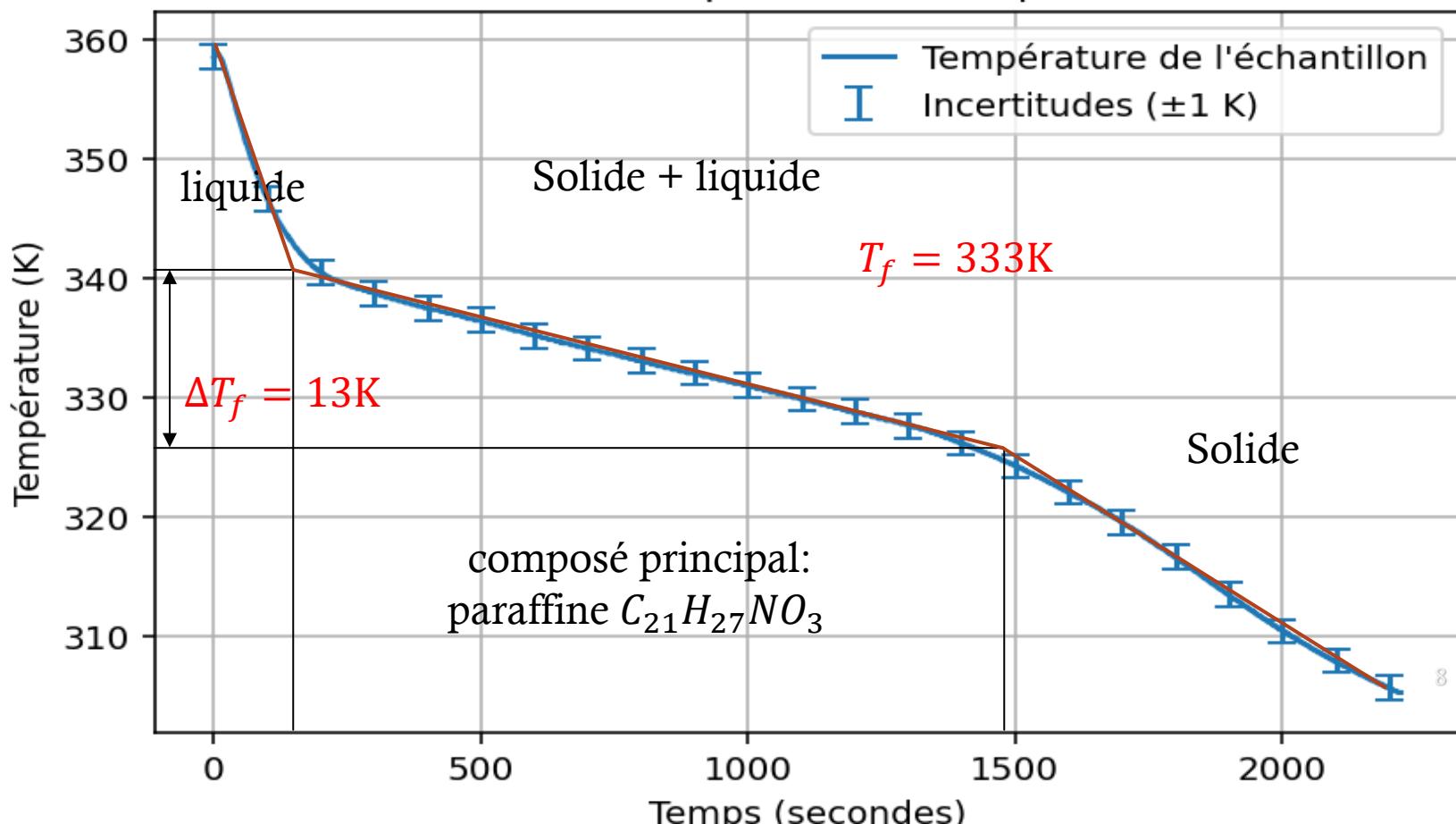
- ❖ Point de fusion bas
- ❖ Bon marché
- ❖ Non toxique



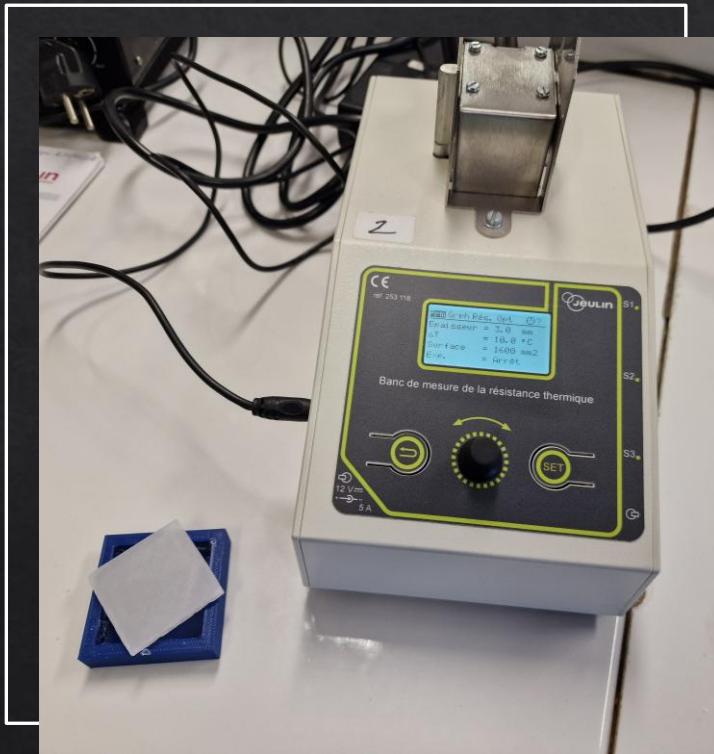
## Étude du point de fusion



relevé de température de la paraffine

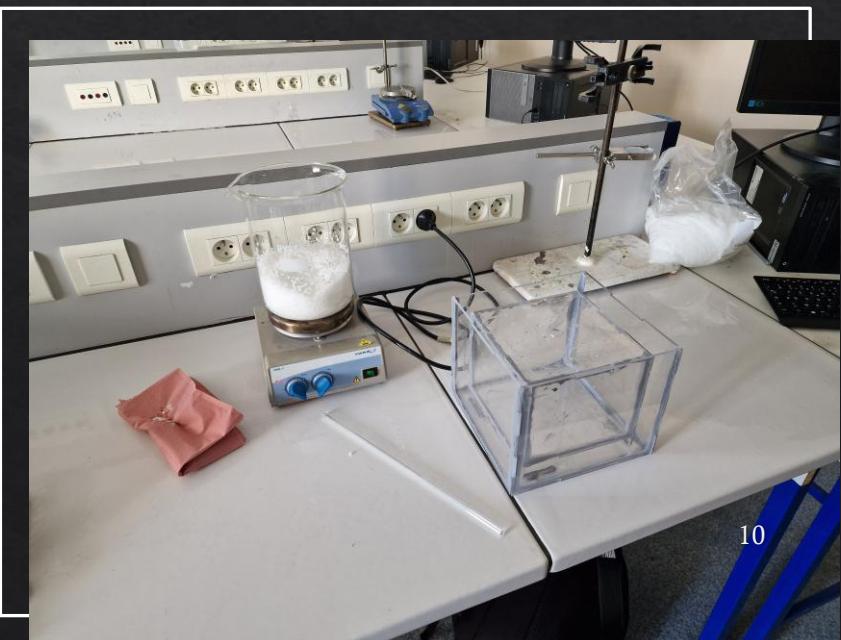
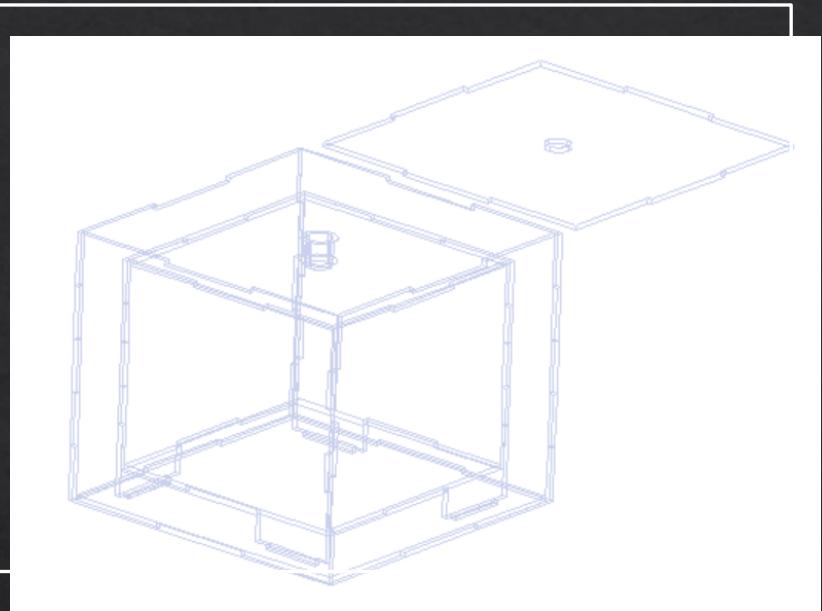


# Étude de la conductivité thermique



# Expérimentation

## Réalisation de la maquette



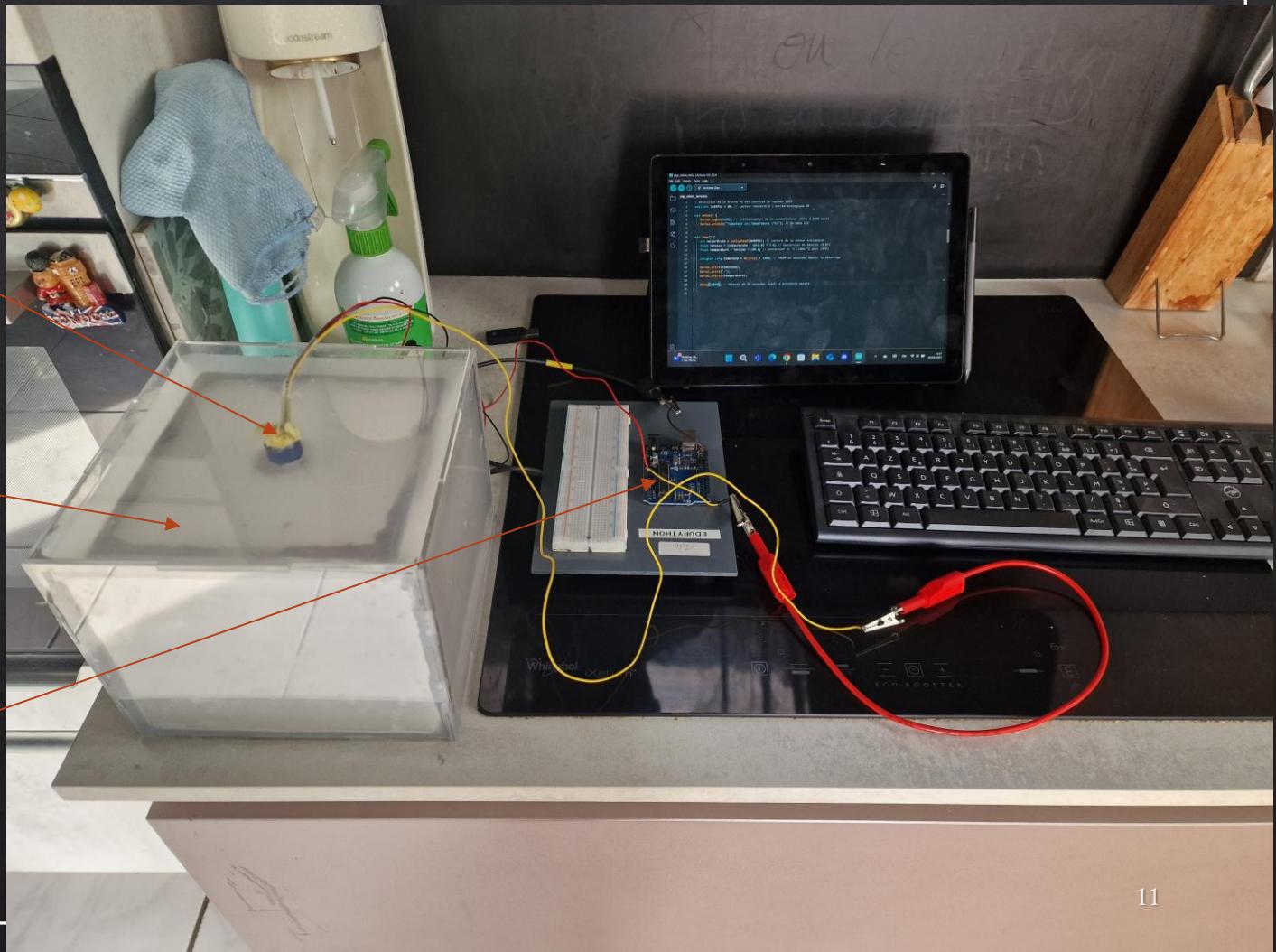
# Expérimentation

## Expérimentation

Capteur de température

maquette

Carte Arduino



# Expérimentation

## Protocole

On relève la température intérieure lors des différentes étapes

Réchauffement de la pièce soumise à une température extérieure de 80°C (four)



1h 30min



Refroidissement de la pièce soumise à une température extérieure de 20°C (air libre)



1h 30min

Refroidissement de la pièce soumise à une température extérieure de 20°C (air libre)



1h 30min



Réchauffement de la pièce soumise à une température extérieure de 80°C (four)

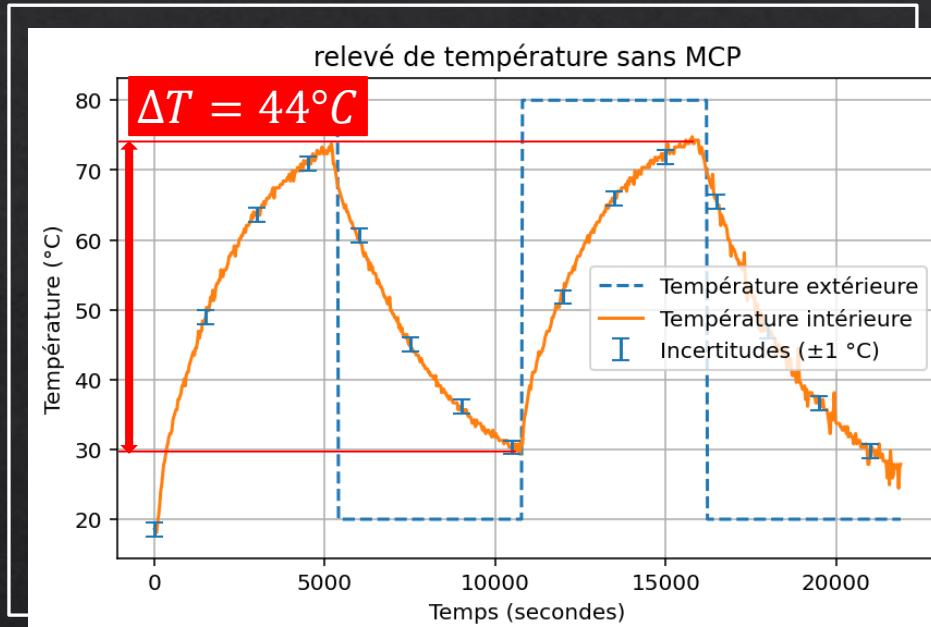
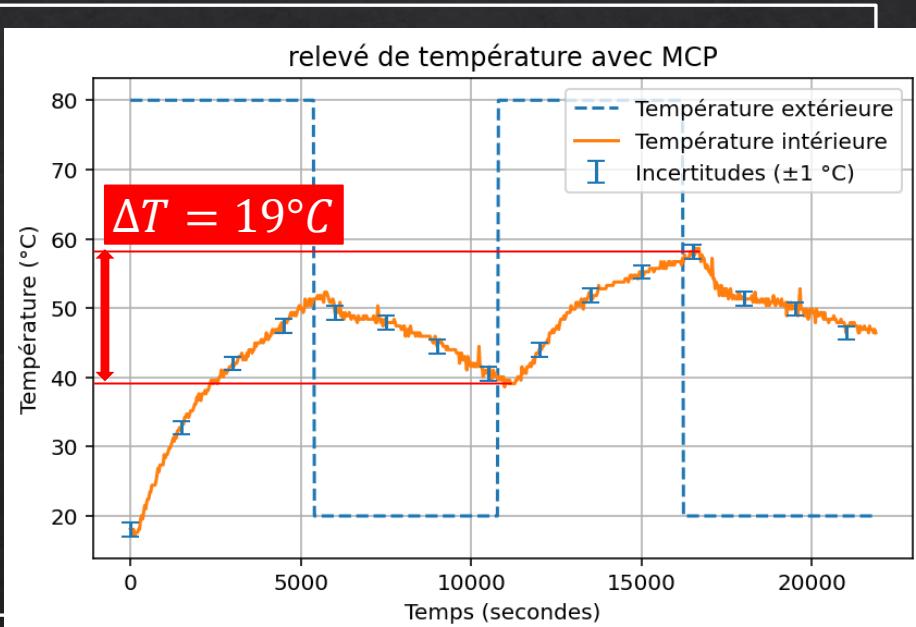


1h 30min

12

# Expérimentation

## Résultats expérimentaux



- Amortissement fort des pics de températures
- Absence de l'augmentation du déphasage de température

# Mise en équation

---

*Equation de diffusion de la chaleur*

$$\frac{\partial H_v(x, t)}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2}$$

*Décomposition de l'enthalpie*

$$H_v = \rho c T + \rho l f$$

*f : fraction de matériau liquide dans le noeud considéré*

*l : enthalpie massique de fusion*

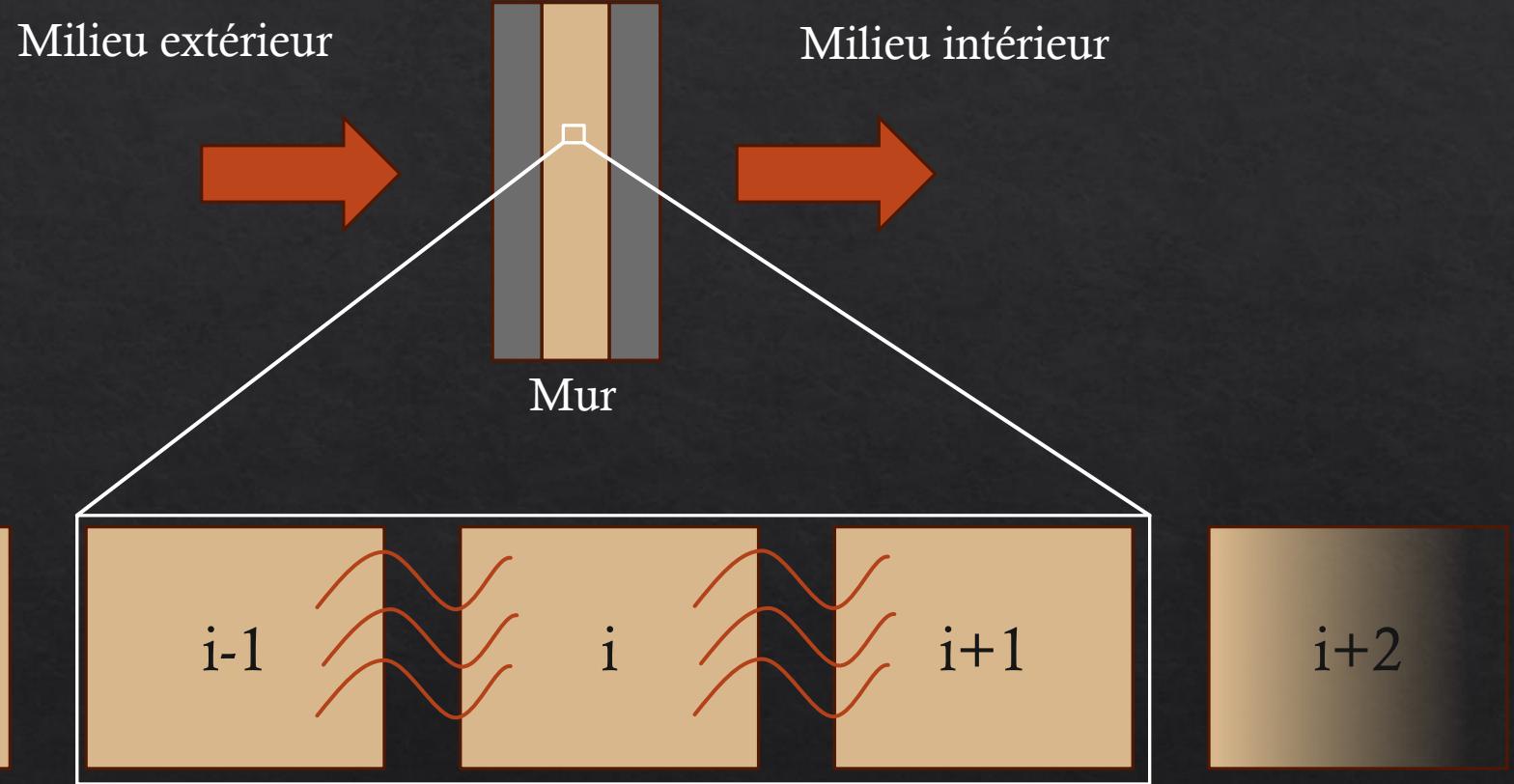
## Modèle numérique

---

- ❖ Hypothèses
  - ❖ Propriété thermo-physique constantes
  - ❖ Transfert de chaleur unidimensionnel et réalisé par la conduction
  - ❖ Mur entièrement fait de MCP



# Discrétisation de l'expression



$$H_i^{n+1} = H_i^n + \frac{\lambda \Delta t}{\Delta x^2} (T_{i-1}^n - 2T_i^n + T_{i+1}^n)$$

## Déroulé du code

N

 $H_i^n$ 

N+1

Mise à  
jour de  $T_i^n$  $H_i^{n+1}$ 

H

$$H_v = \rho c_p T$$

$$H_v = \rho c_p T + \rho l \frac{T - (T_m - \delta)}{2\delta}$$

$$H_v = \rho c_p T + \rho l$$

État

solide

solide + liquide

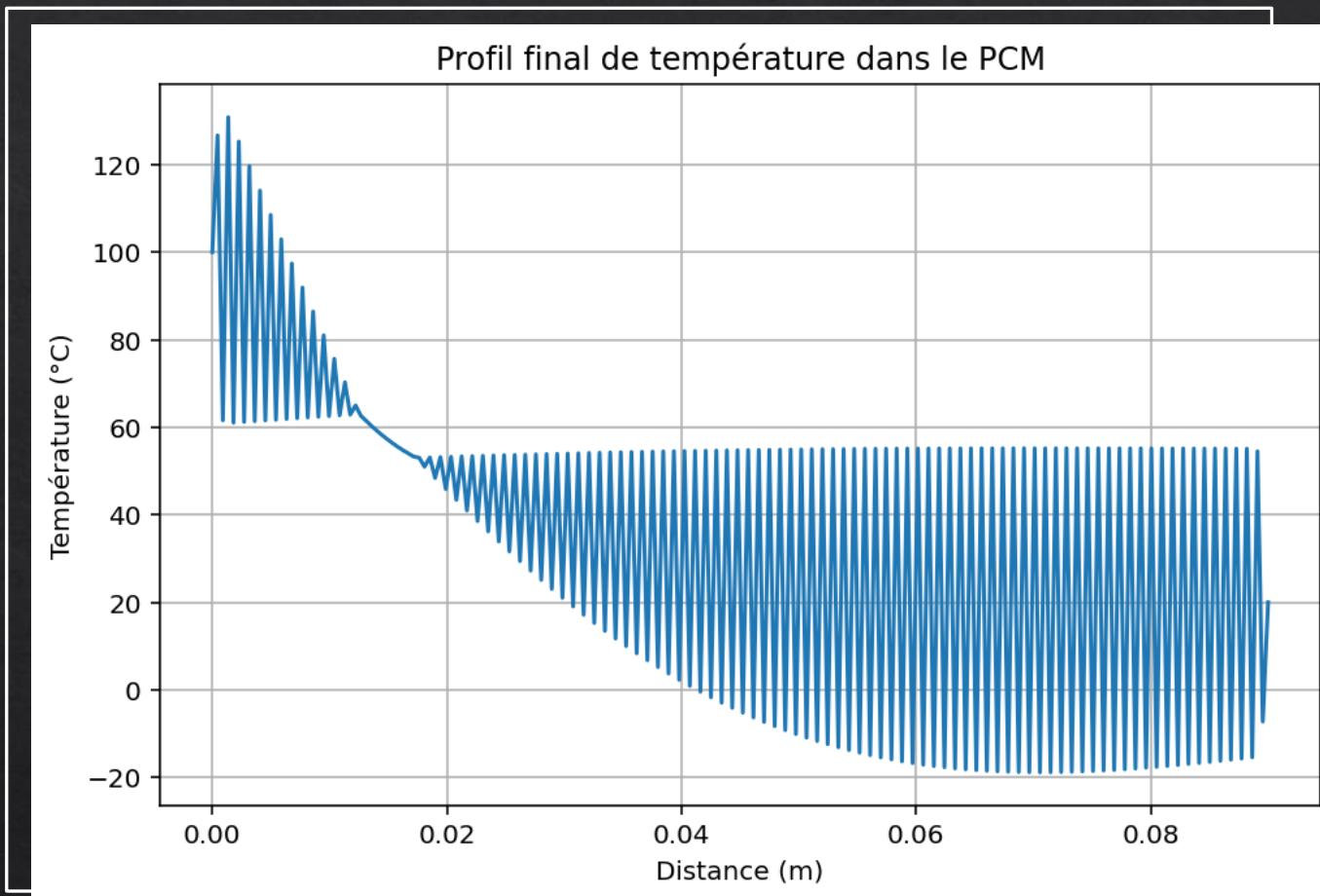
liquide

T

$$T_m - \delta$$

$$T_m + \delta$$

## Premier essaie



---

condition de stabilité  $\Delta t \leq \frac{\rho c \Delta x^2}{2k}$

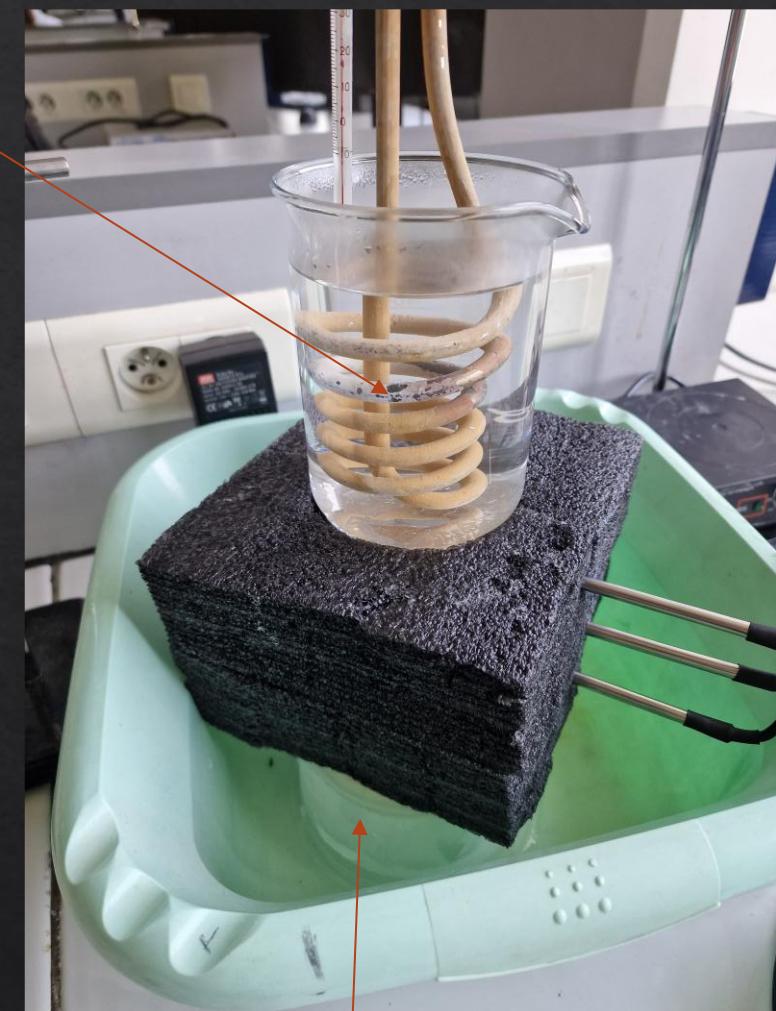
$$\Delta t = 1\text{s} \text{ et } \frac{\rho c \Delta x^2}{2k} = 0,81\text{s}$$

## Validation expérimentale du modèle



paraffine

Capteur de température

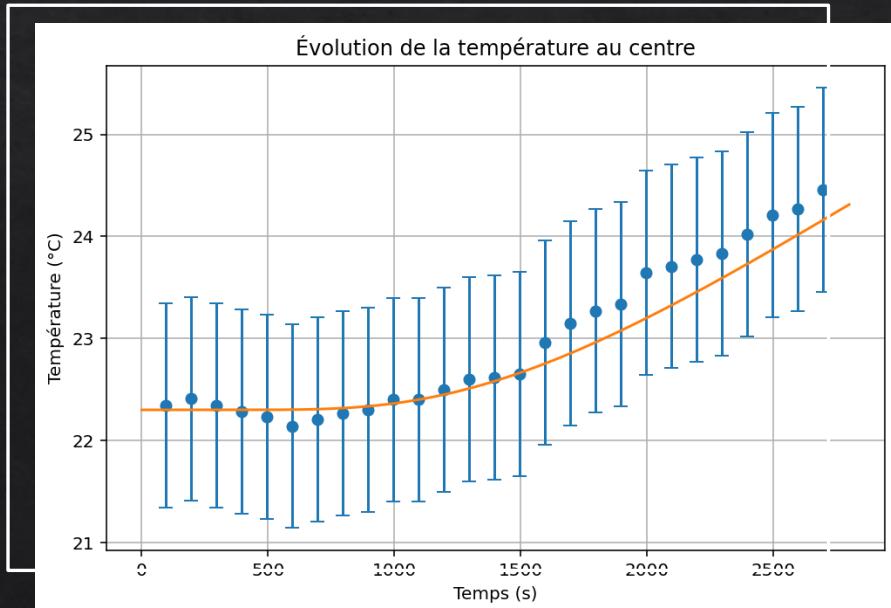
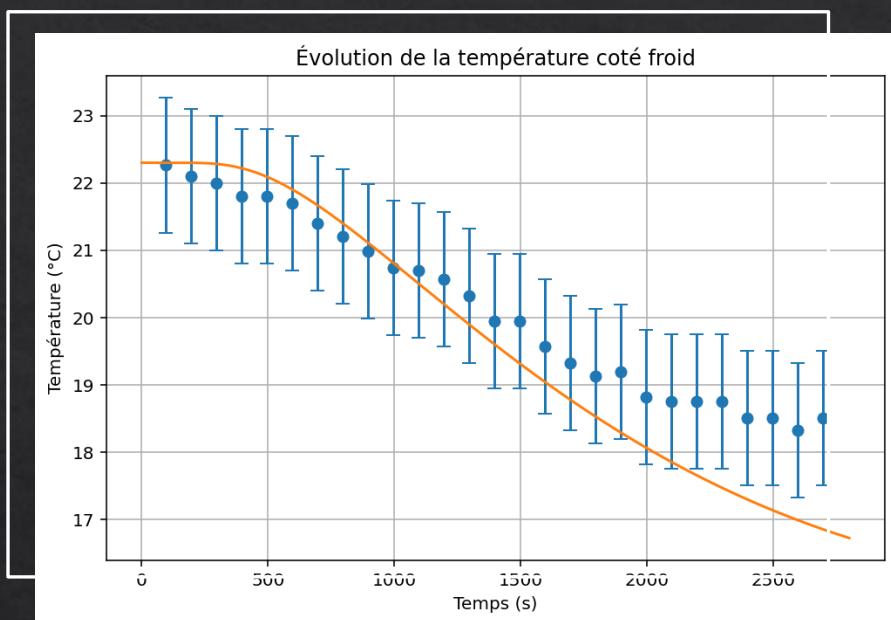
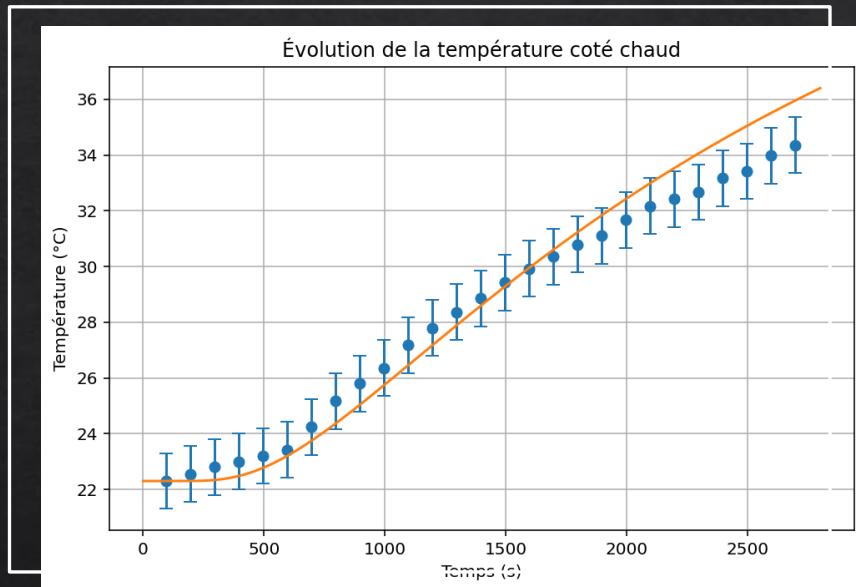


Thermostat à 0°C

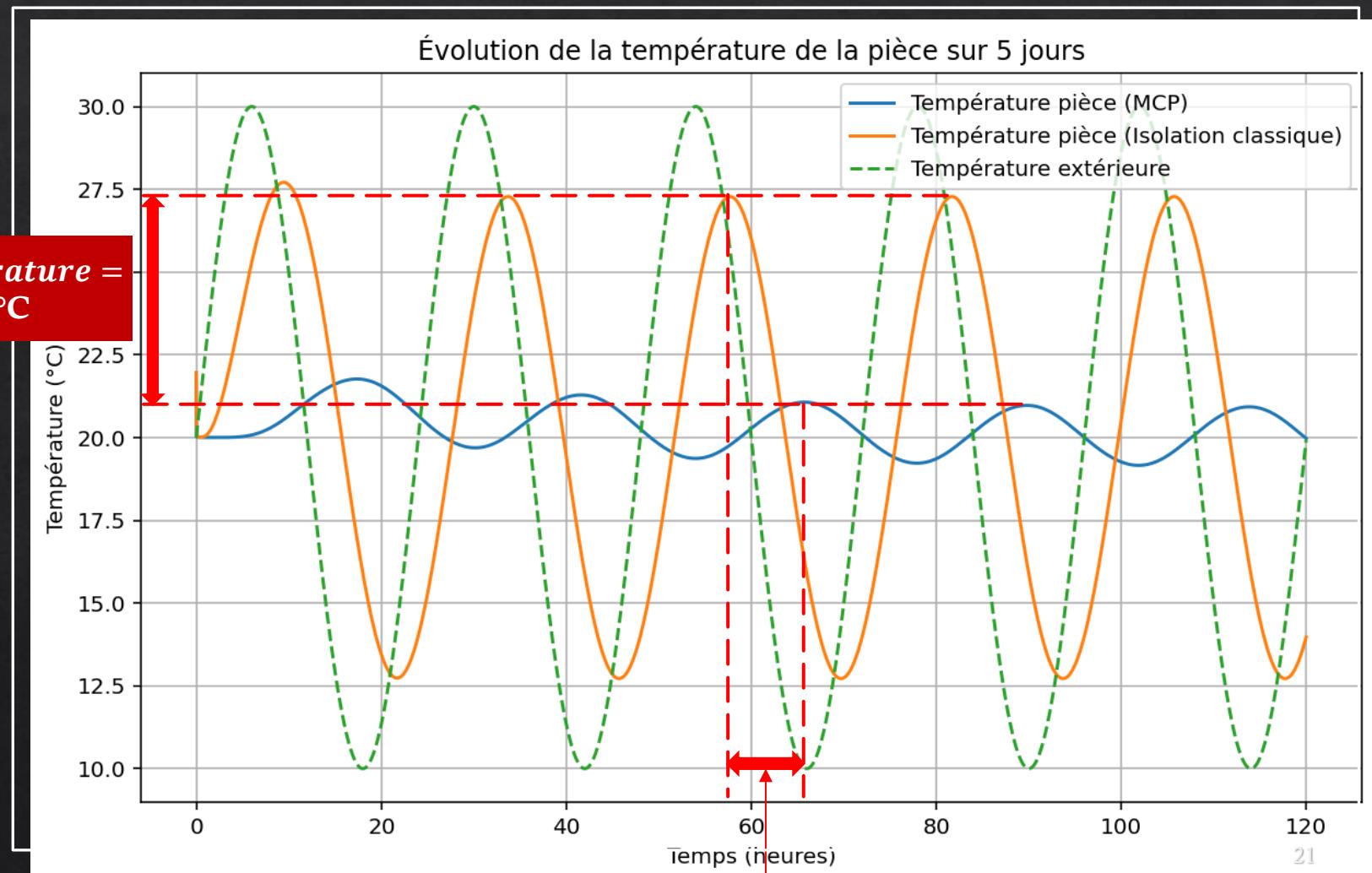
# Modèle numérique

Condition de stabilité respecté :

$$\Delta t = 0,01\text{s} \text{ et } \frac{\rho c \Delta x^2}{2k} = 0,085\text{s}$$



## Évolution de la température dans une pièce avec une isolation MCP



$\Delta\text{Temps} = 8,2 \text{ heures}$

## Modèle amélioré

---

- ❖ Hypothèses
  - ❖ Propriétés thermo-physique constantes
  - ❖ Transfert de chaleur unidimensionnel et réalisé par la conduction
  - ❖ Mur constitué d'un mélange de Laine de verre et de MCP à une proportion p
  - ❖ Chaque nœud possède la même proportion de MCP et de Laine de Verre
  - ❖ Chaque nœud est uniformément chauffé

# Equations du modèle amélioré

---

*Equation de diffusion de la chaleur*

$$\frac{\partial H(x, t)}{\partial t} = (\lambda_{MCP} p + \lambda_{MCP} \times (1 - p)) \times \frac{\partial^2 T}{\partial x^2}$$

*Décomposition de l'enthalpie*

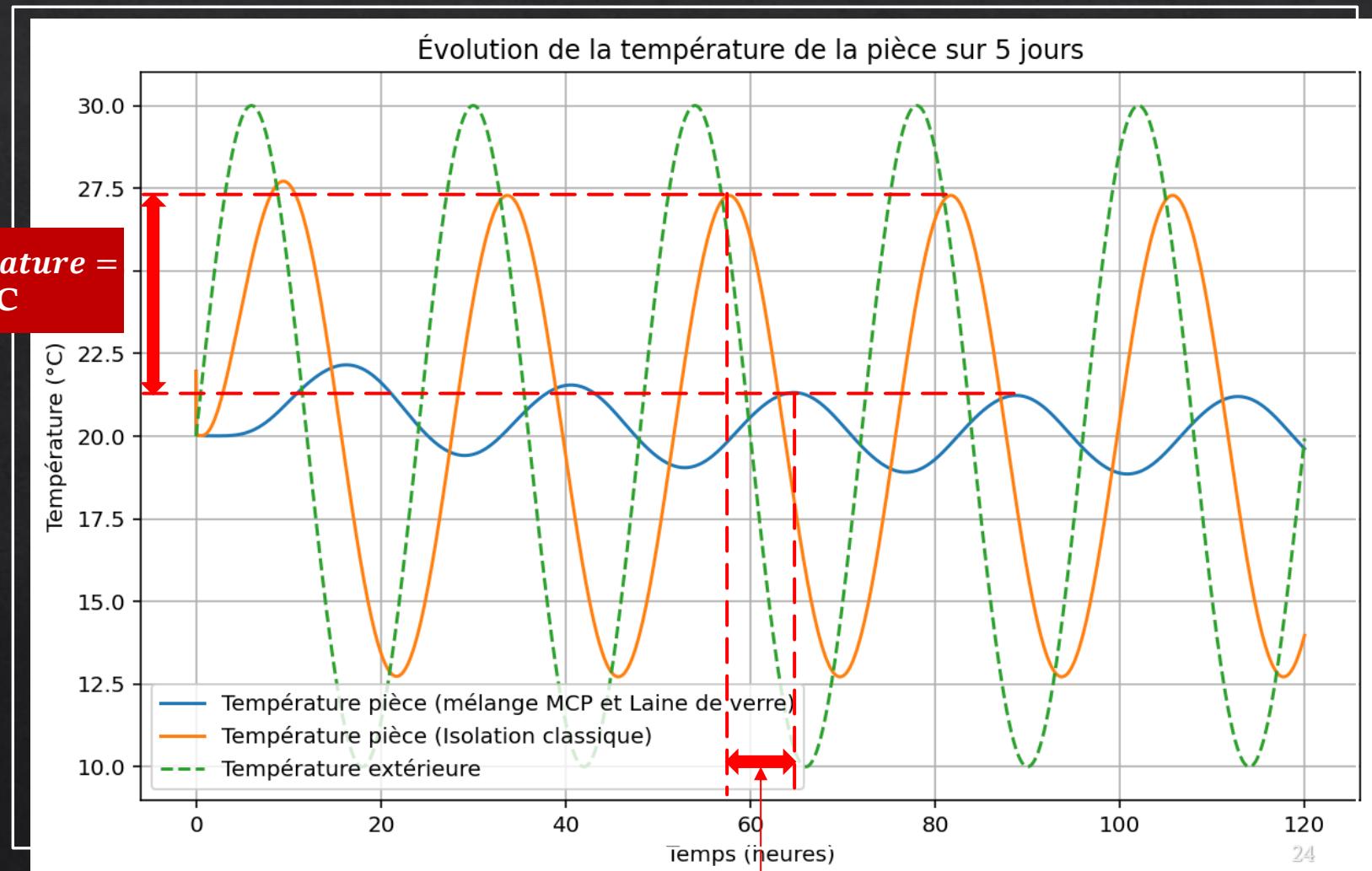
$$H_v = (\rho_{MCP} c_{MCP} T + \rho_{MCP} l f) \times p + \rho_{MCP} c_{MCP} T \times (1 - p)$$

*f : fraction de matériau liquide dans le noeud considéré*

*l : enthalpie massique de fusion*

# Modèle Numérique

## Évolution de la température dans une pièce avec une isolation MCP et Laine de Verre avec $p=0,3$



$\Delta\text{Temps} = 6,7 \text{ heures}$

# conclusion

---

- Grande augmentation de l'inertie thermique :
  - Pics de chaleurs diminuées de 6°C
  - Déphasage thermique augmenté de 6,7 heures
- Problème d'utilisation:
  - Efficacité réduite avec le temps
  - Renouvellement de l'isolation

# Annexe 1

```
// Constantes
const int pinLM35 = A0; // Broche connectée au capteur LM35

void setup() {
    Serial.begin(9600); // Initialisation de la communication série
}

void loop() {
    // Lecture de la valeur brute (0-1023) sur le port analogique
    int valeurBrute = analogRead(pinLM35);

    // Conversion de la valeur brute en tension (en volts)
    float tension = valeurBrute * (5.0 / 1023.0);

    // Conversion de la tension en température (en °C)
    float temperature = tension * 100.0; // LM35 : 10 mV = 1°C

    // Affichage de la température dans le moniteur série
    Serial.print("Température : ");
    Serial.print(temperature);
    Serial.println(" °C");

    // Attente avant la prochaine mesure
    delay(1000); // 1 seconde
}
```

## Annexe 2

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

# %%
# -----
# Paramètres physiques et géométriques
# -----
L = 0.1          # épaisseur du mur (m)
N = 50           # nombre de points spatiaux
dx = L / (N - 1)
p = 1            #fraction de MCP dans un noeud

#MCP
rho = 800         # masse volumique (kg/m^3)
cp = 2000          # capacité thermique (J/(kg.K))
k = 0.5            # conductivité thermique (W/(m.K))

#Laine de verre
rho_v = 25         # masse volumique (kg/m^3)
cp_v = 1030          # capacité thermique (J/(kg.K))
k_v = 0.046          # conductivité thermique (W/(m.K))

# Paramètres spécifiques au PCM
L_latent = 15e4    # chaleur latente (J/kg)
T_m = 20.0           # température de fusion (°C)
delta = 5.0           # intervalle autour de T_m pour la transition (°C)
```

## Annexe 4

```
49 # -----
50 # Fonctions pour la méthode enthalpique (PCM)
51 # -----
52 def compute_H_pcm(T):
53     """
54     Calcule l'enthalpie H pour un profil de température T, en tenant compte du PCM.
55     """
56     H = np.zeros_like(T)
57     mask_solid = T < (T_m - delta)
58     mask_liquid = T > (T_m + delta)
59     mask_mushy = ~(mask_solid | mask_liquid)
60
61     H[mask_solid] = (rho * cp * T[mask_solid]) * p + (rho_v * cp_v * T[mask_solid]) * (1-p)
62     H[mask_liquid] = (rho * cp * T[mask_liquid] + rho * L_latent) * p + (rho_v * cp_v * T[mask_liquid]) * (1-p)
63     H[mask_mushy] = (rho * cp * T[mask_mushy] \
64                     + rho * L_latent * ((T[mask_mushy] - (T_m - delta)) / (2 * delta))) * p \
65                     +(rho_v * cp_v * T[mask_mushy]) * (1-p)
66
67     return H
68
69 def T_from_H_pcm(H):
70     """
71     Récupère la température T à partir de l'enthalpie H pour le matériau PCM.
72     """
73     T = np.zeros_like(H)
74     H_low = (rho * cp * (T_m - delta)) * p + (rho_v * cp_v * (T_m - delta)) * (1-p)
75     H_high = (rho * cp * (T_m + delta) + rho * L_latent) * p + (rho_v * cp_v * (T_m + delta)) * (1-p)
76
77     mask_solid = H < H_low
78     mask_liquid = H > H_high
79     mask_mushy = ~(mask_solid | mask_liquid)
80
81     # Zone solide
82     T[mask_solid] = H[mask_solid] / ( rho * cp * p + rho_v * cp_v * (1 - p) )
83     # Zone liquide
84     T[mask_liquid] = (H[mask_liquid] - rho * L_latent * p) / ( (rho * cp * p) + rho_v * cp_v * (1 - p) )
85     # Zone mushy (transition)
86     T[mask_mushy] = (H[mask_mushy] + (rho * L_latent / (2 * delta)) * (T_m - delta) * p) \
87                     / ( (rho * cp + (rho * L_latent / (2 * delta))) * p + rho_v * cp_v * (1 - p) )
88
89     return T
```

## Annexe 4

```
90 # -----
91 # Initialisation des profils muraux et de la pièce
92 # -----
93 T_init_wall = 20.0    # température initiale du mur (°C)
94
95 # Pour la simulation PCM
96 T_pcm = np.ones(N) * T_init_wall
97 H_pcm = compute_H_pcm(T_pcm)
98
99 # Pour l'isolation classique (aucun PCM, mise à jour directe de T)
100 T_classic = np.ones(N) * T_init_wall
101
102 # Températures initiales de la pièce (pour chaque cas)
103 T_room_pcm = 22.0
104 T_room_classic = 22.0
105
106 # Pour l'enregistrement des résultats
107 time_arr = []
108 T_room_pcm_arr = []
109 T_room_classic_arr = []
110 T_interior_pcm_arr = []      # température à la face intérieure du mur (x = L) - PCM
111 T_interior_classic_arr = [] # même pour isolation classique
112 T_ext_arr = []
```

# Annexe 5

```
115 # -----
116 # Boucle temporelle de simulation
117 #
118 for step in range(steps):
119     t = step * dt
120     # Condition extérieure imposée à x = 0
121     T_ext_val = T_ext(t)
122
123     # ----- Mise à jour pour la simulation PCM -----
124     # Condition limite extérieure
125     T_pcm[0] = T_ext_val
126     H_pcm[0] = compute_H_pcm(np.array([T_pcm[0]]))[0]
127
128     # Condition convective intérieure (x = L)
129     # Approximation par DF: T_pcm[-1] = ( T_pcm[-2] + (h_conv * dx / k)*T_room_pcm ) / (1 + (h_conv * dx / k))
130     T_pcm[-1] = (T_pcm[-2] + (h_conv * dx / (k * p + k_v * (1-p))) * T_room_pcm) / (1 + (h_conv * dx / (k * p + k_v * (1-p))))
131     H_pcm[-1] = compute_H_pcm(np.array([T_pcm[-1]]))[0]
132
133     # Récupérer T à partir de H pour mettre à jour la conduction
134     T_temp_pcm = T_from_H_pcm(H_pcm)
135     H_new_pcm = H_pcm.copy()
136     # Mise à jour des noeuds internes (schéma explicite)
137     for i in range(1, N - 1):
138         d2Tdx2 = (T_temp_pcm[i+1] - 2 * T_temp_pcm[i] + T_temp_pcm[i-1]) / dx**2
139         # dH/dt = k * d2T/dx^2
140         H_new_pcm[i] = H_pcm[i] + dt * (k * p + k_v * (1 - p)) * d2Tdx2
141     H_pcm = H_new_pcm.copy()
142     T_pcm = T_from_H_pcm(H_pcm)
143
144     # ----- Mise à jour pour la simulation en isolation classique -----
145     T_classic[0] = T_ext_val # condition extérieure
146     T_classic[-1] = (T_classic[-2] + (h_conv * dx / k) * T_room_classic) / (1 + (h_conv * dx / k))
147     T_new_classic = T_classic.copy()
148     for i in range(1, N - 1):
149         d2Tdx2_classic = (T_classic[i+1] - 2 * T_classic[i] + T_classic[i-1]) / dx**2
150         # Equation classique: rho*cp * dT/dt = k * d2T/dx^2
151         T_new_classic[i] = T_classic[i] + dt * (k / (rho * cp)) * d2Tdx2_classic
152     T_classic = T_new_classic.copy()
153
154     # ----- Mise à jour de la température de la pièce -----
155     # C_room * dT_room/dt = h_conv * A * (T_interior_wall - T_room)
156     T_room_pcm += dt * (h_conv * A / C_room) * (T_pcm[-1] - T_room_pcm)
157     T_room_classic += dt * (h_conv * A / C_room) * (T_classic[-1] - T_room_classic)
158
159     # Enregistrement des résultats tous les 100 pas
160     if step % 100 == 0:
161         time_arr.append(t / 3600.0) # temps en heures
162         T_room_pcm_arr.append(T_room_pcm)
163         T_room_classic_arr.append(T_room_classic)
164         T_interior_pcm_arr.append(T_pcm[-1])
165         T_interior_classic_arr.append(T_classic[-1])
166         T_ext_arr.append(T_ext_val)
```

## Annexe 5

```
169 # -----
170 # Visualisation des résultats
171 # -----
172 plt.figure(figsize=(10, 6))
173 plt.plot(time_arr, T_room_pcm_arr, label='Température pièce (MCP)')
174 plt.plot(time_arr, T_room_classic_arr, label='Température pièce (Isolation classique)')
175 plt.plot(time_arr, T_ext_arr, label='Température extérieure', linestyle='--')
176 plt.xlabel('Temps (heures)')
177 plt.ylabel('Température (°C)')
178 plt.title('Évolution de la température de la pièce sur 5 jours')
179 plt.legend()
180 plt.grid(True)
181 plt.show()
182
183 plt.figure(figsize=(10, 6))
184 plt.plot(time_arr, T_interior_pcm_arr, label='Temp. intérieure du mur (MCP)')
185 plt.plot(time_arr, T_interior_classic_arr, label='Temp. intérieure du mur (Isolation classique)')
186 plt.plot(time_arr, T_ext_arr, label='Température extérieure', linestyle='--')
187 plt.xlabel('Temps (heures)')
188 plt.ylabel('Température (°C)')
189 plt.title('Évolution de la température à la face intérieure du mur')
190 plt.legend()
191 plt.grid(True)
192 plt.show()
193
```