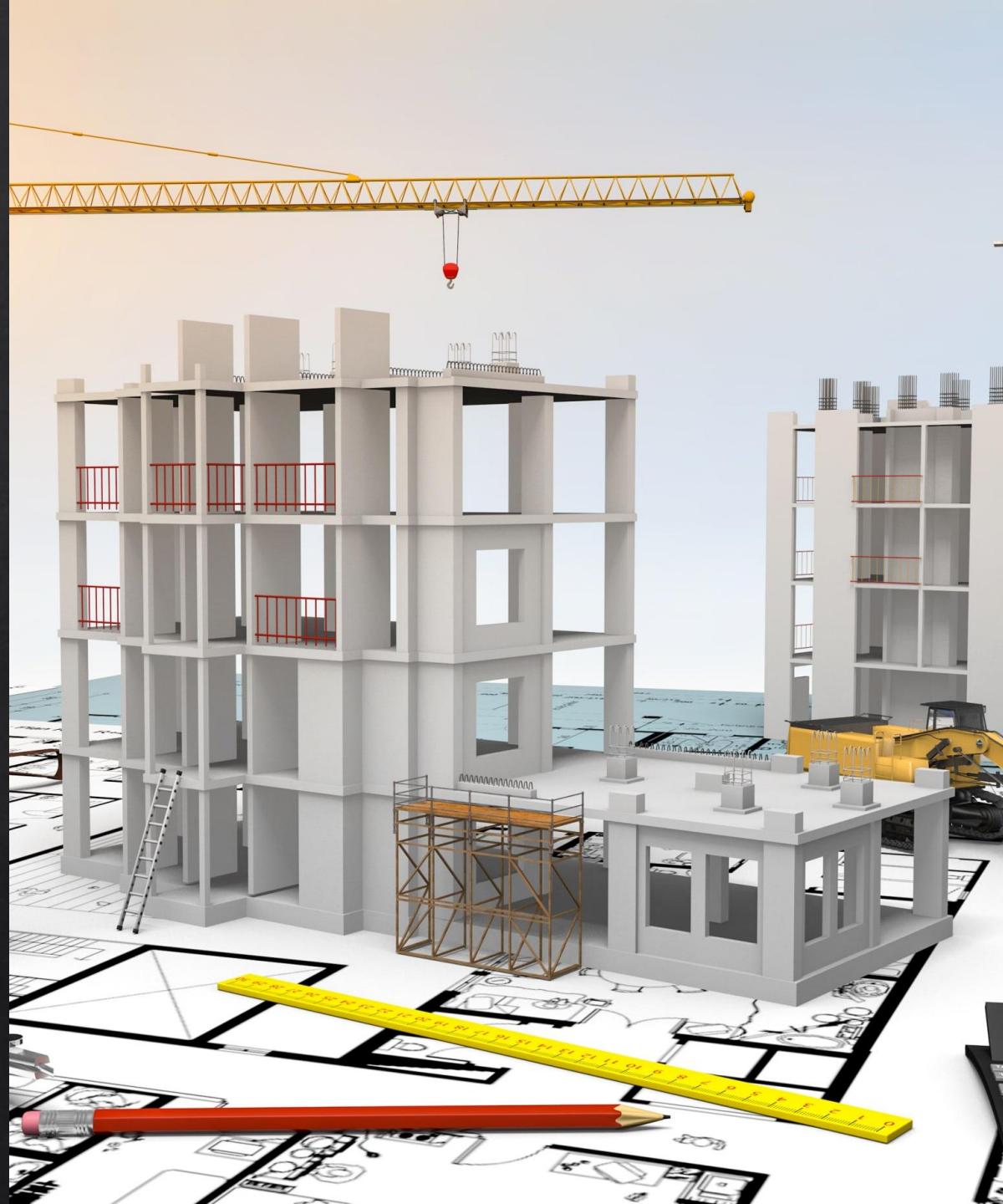


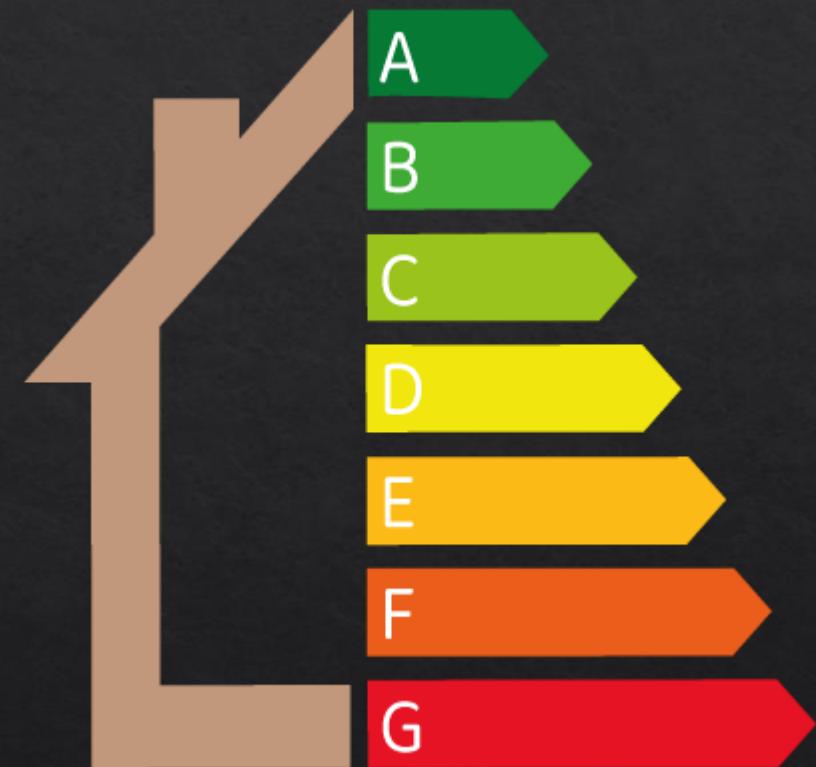
Incorporation des matériaux à changement de phase dans l'isolation thermique d'un bâtiment

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

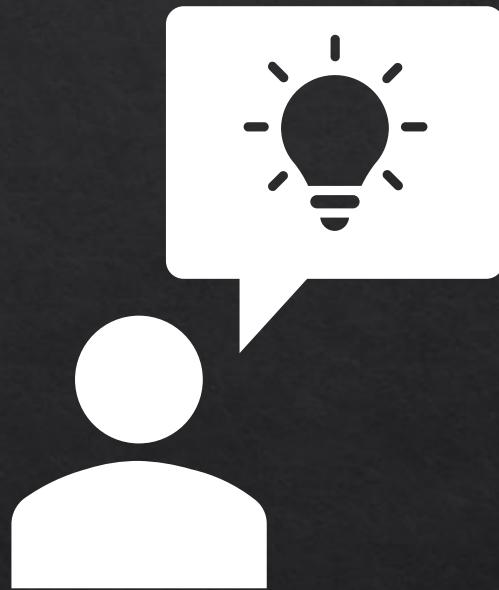


Problématique :
Comment améliorer le confort
thermique d'un bâtiment grâce aux
matériaux à changement de phase
?

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



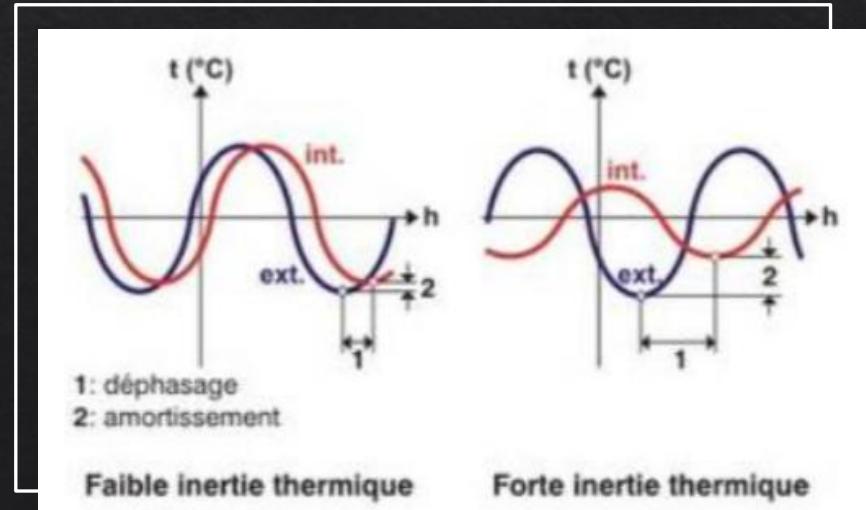
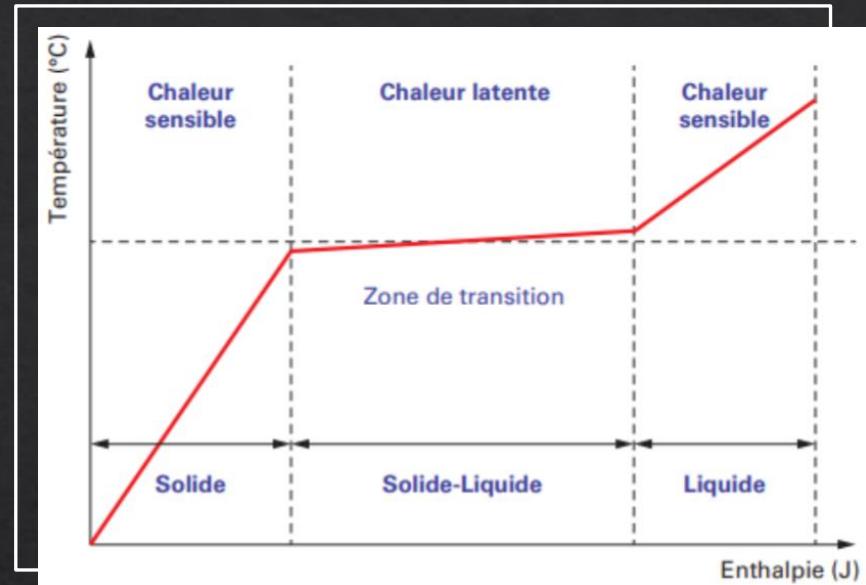


Plan d'étude

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

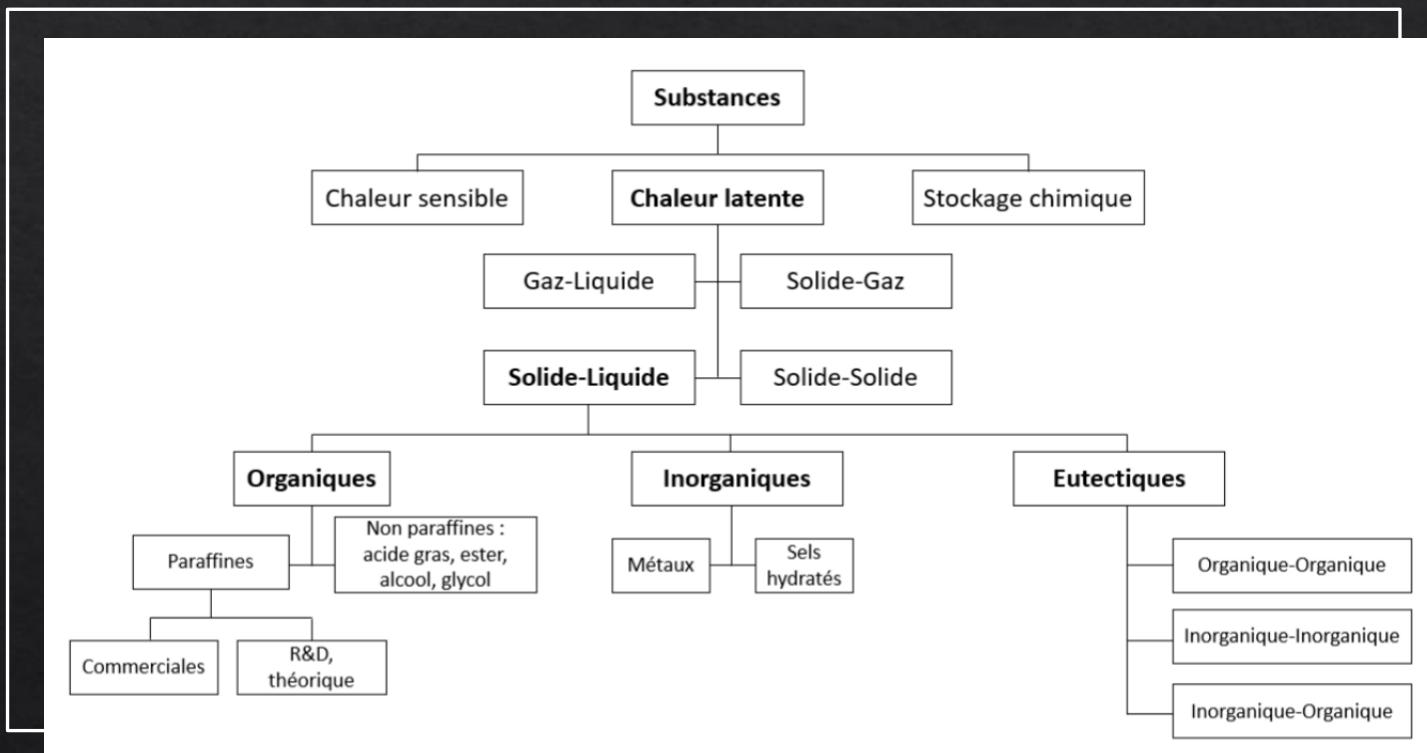
- ❖ Chaleur sensible/chaleur latente
- ❖ Changement de température jour/nuit
- ❖ Augmentation de l'inertie thermique



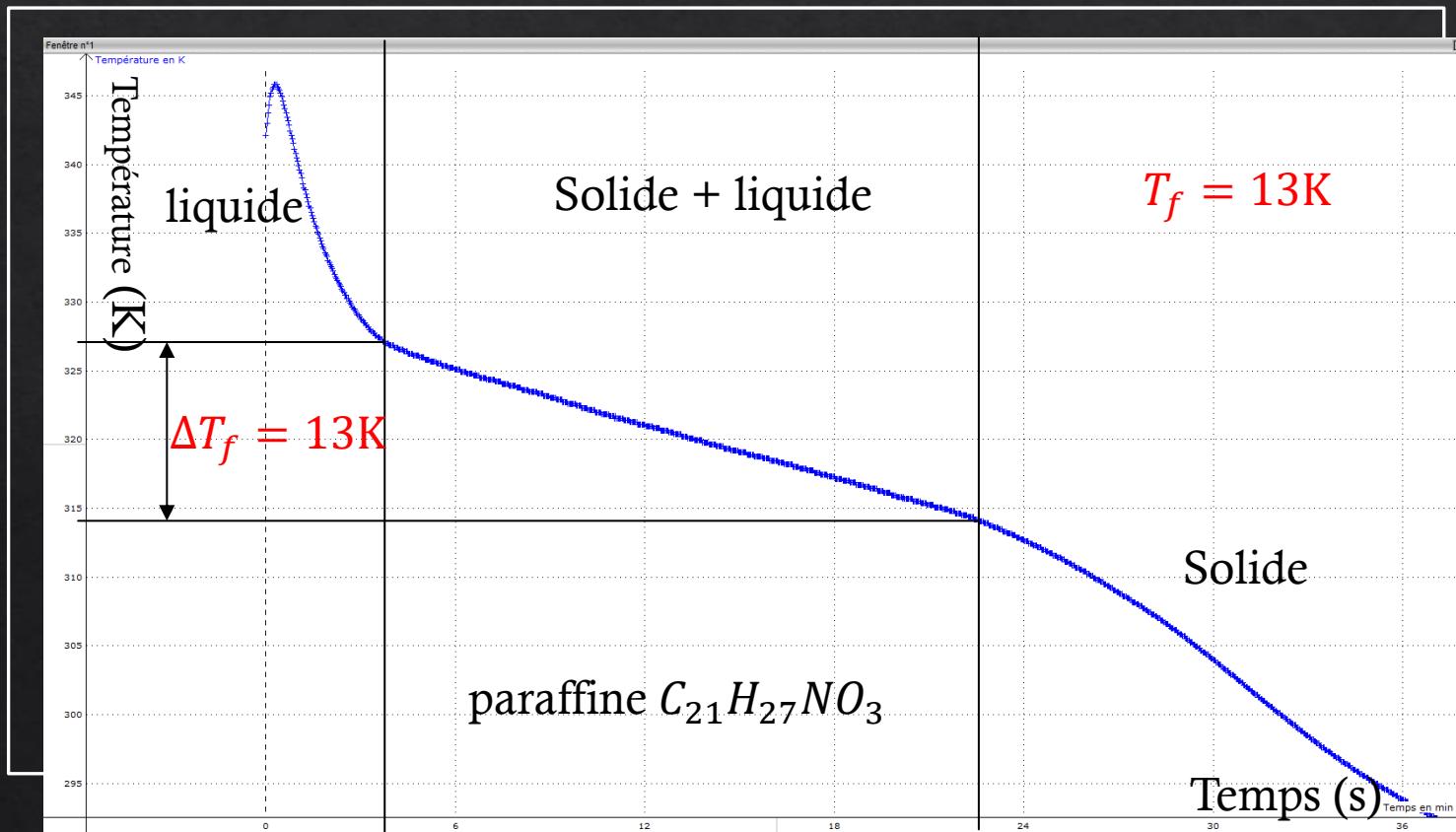
Choix du MCP

Paraffine :

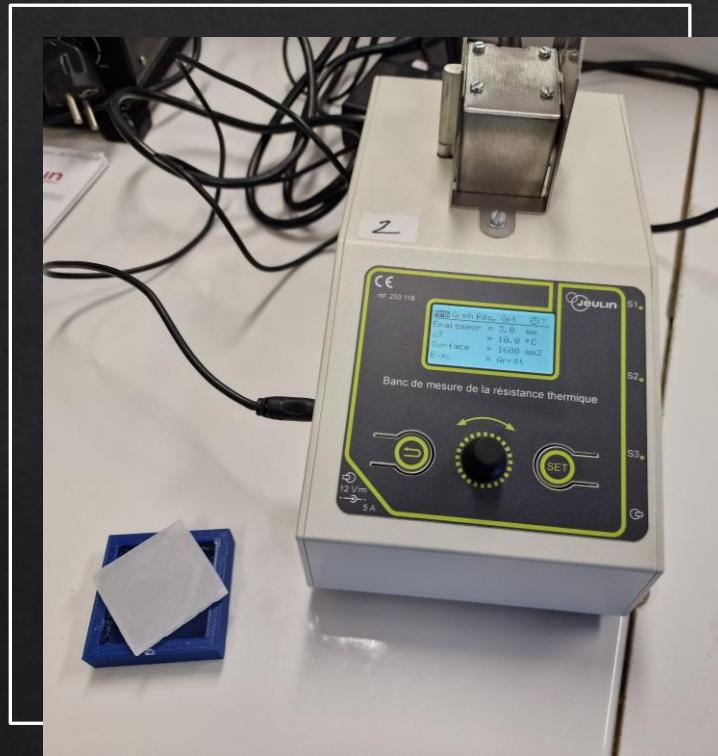
- ❖ Point de fusion bas
- ❖ Bon marché
- ❖ Non toxique



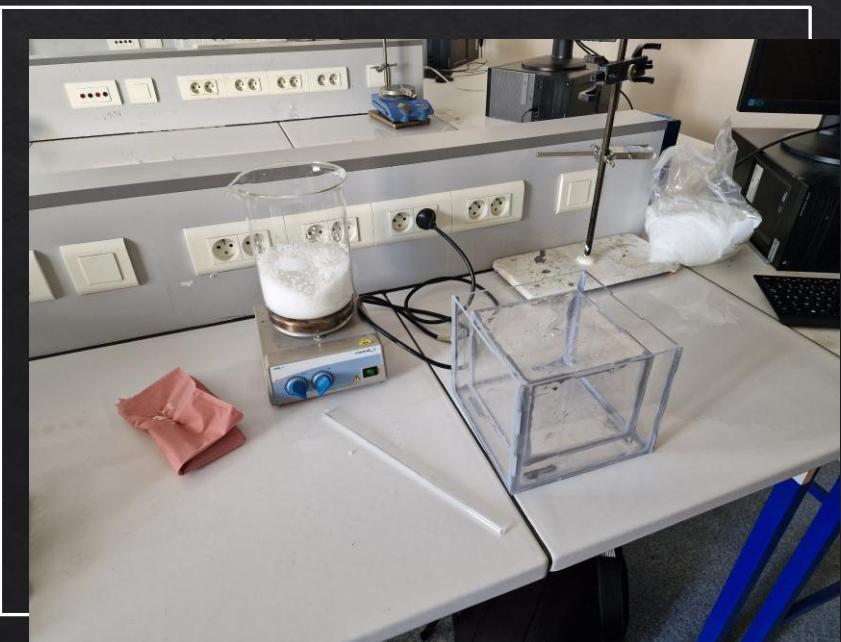
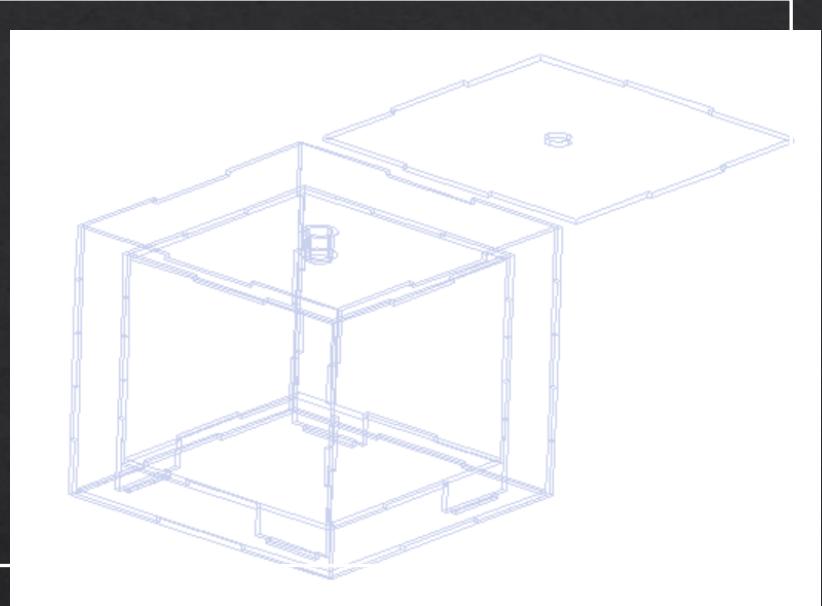
Étude du point de fusion



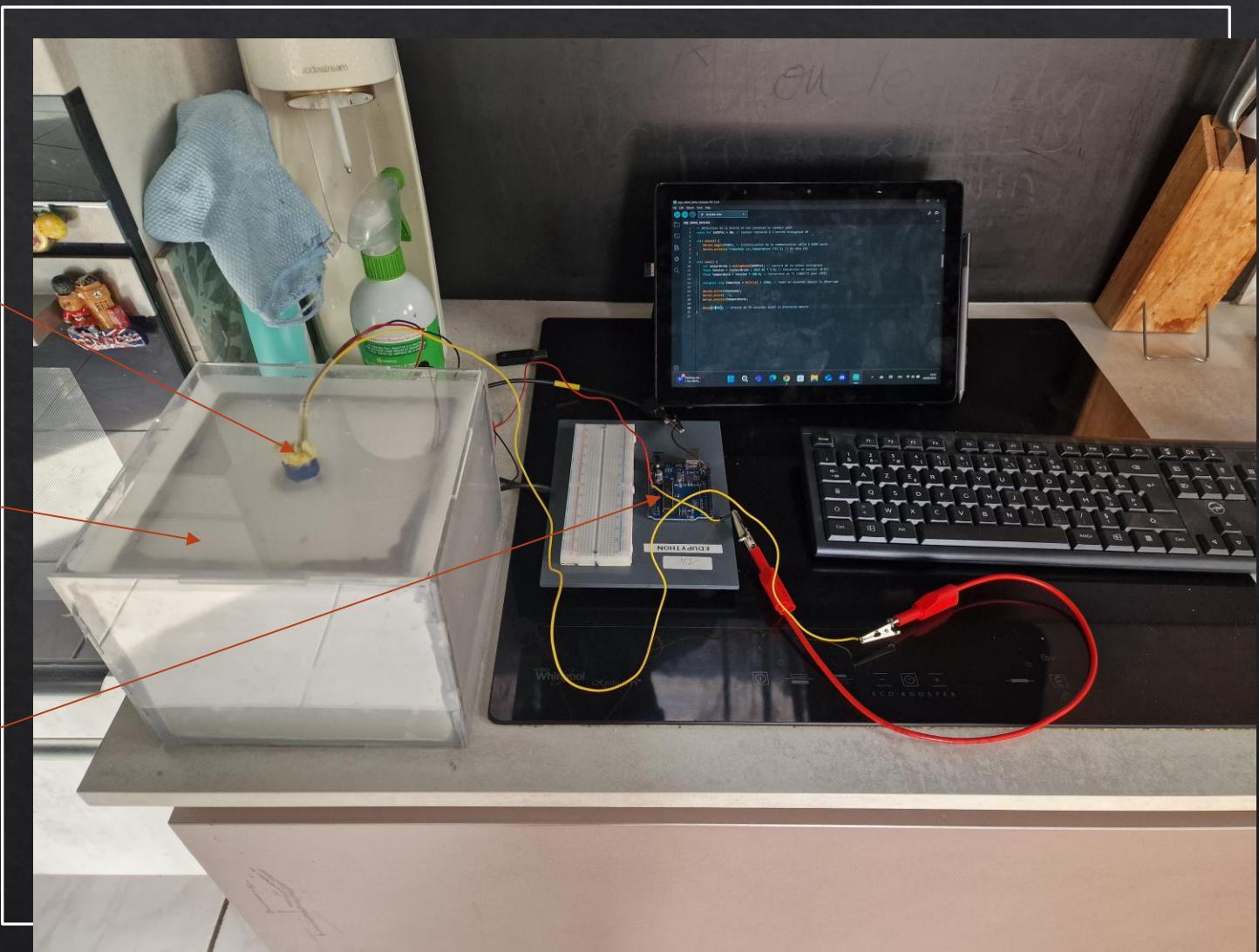
Étude de la conductivité thermique



Réalisation de la maquette



Expérimentation

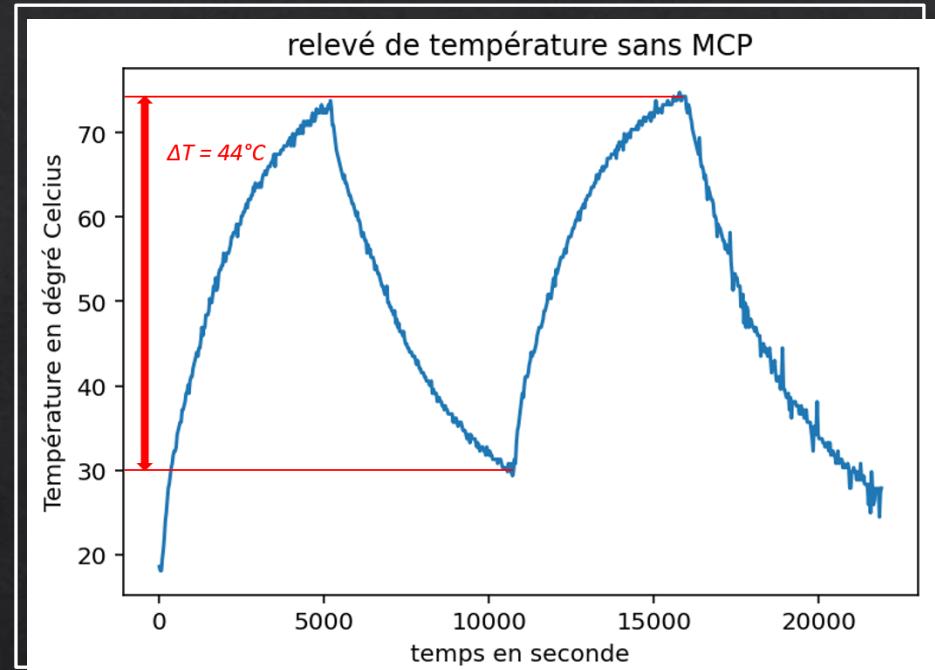
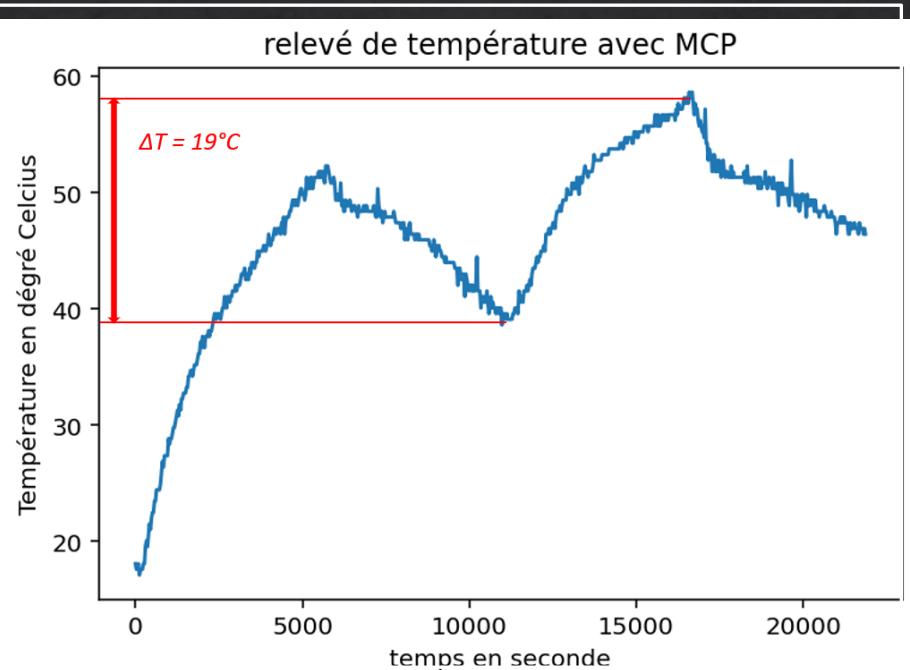


Protocole

On relève la température intérieure lors des différentes étapes



Résultats expérimentaux



- Amortissement fort des pics de températures
- Absence d'introduction d'un déphasage de température

Mise en équation

Equation de diffusion de la chaleur

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

Décomposition de l'enthalpie

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

f : fraction de matériau liquide dans le noeud considéré

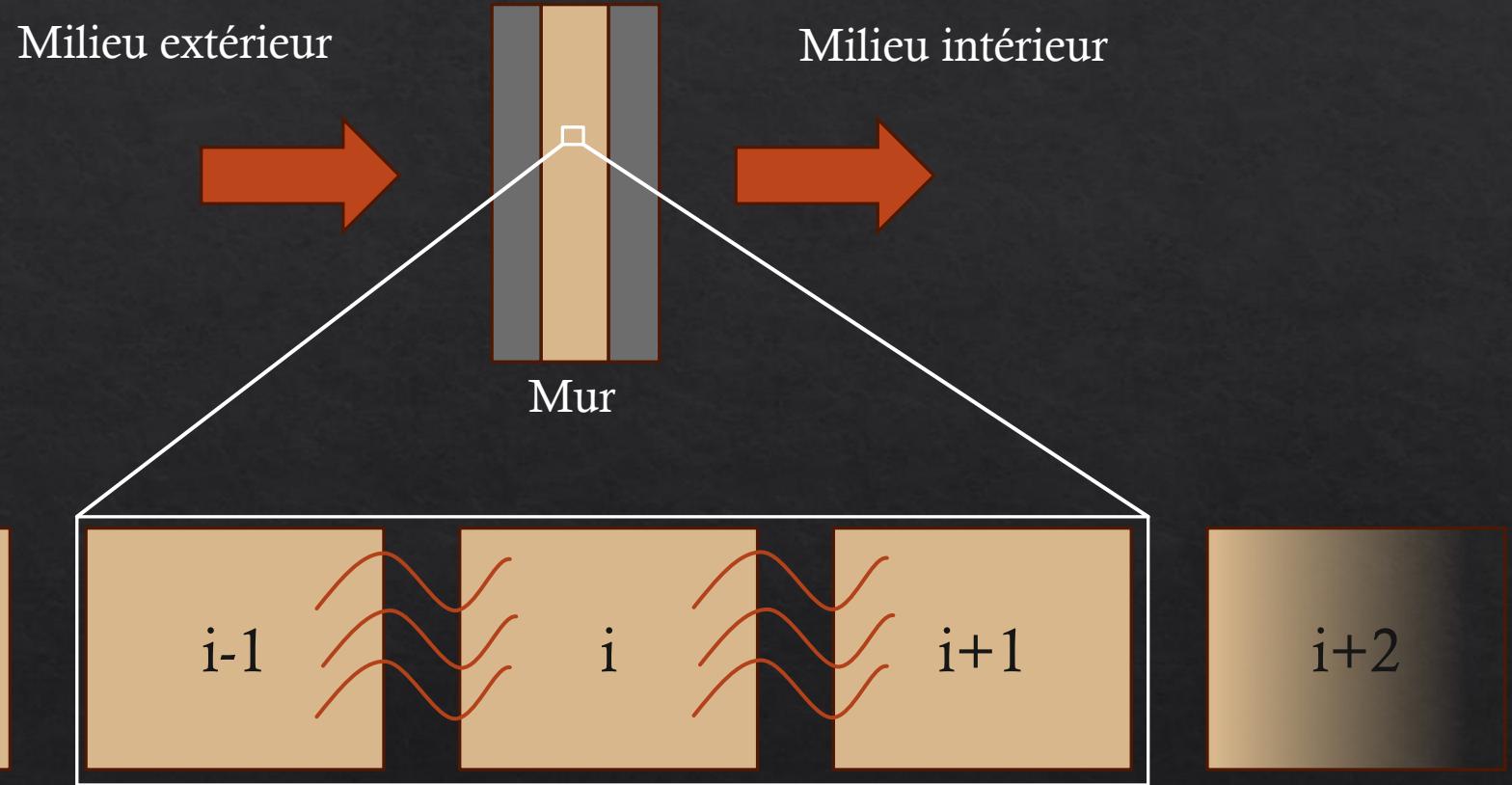
l : enthalpie massique de fusion

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

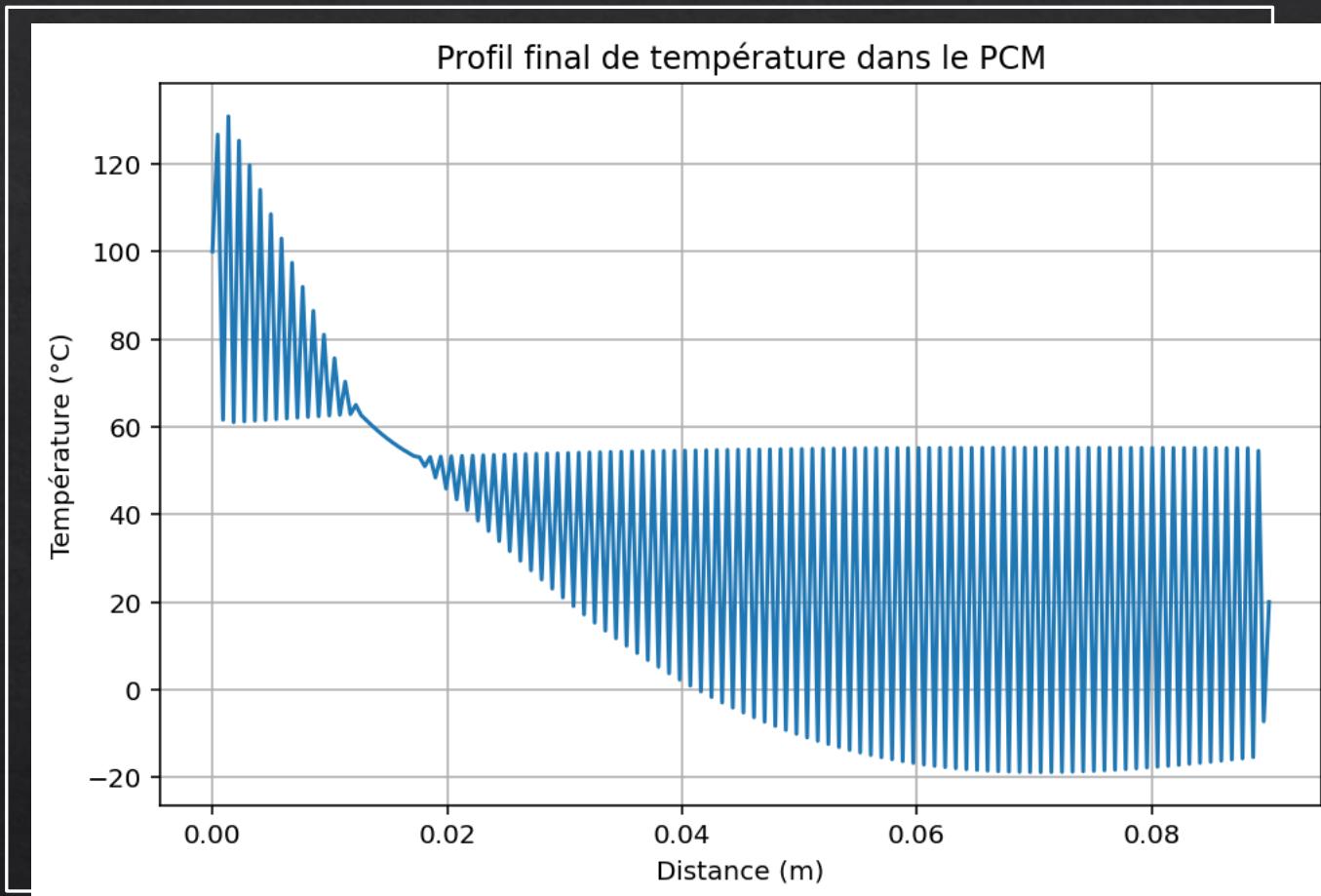
liquide

T

$$T_m - \delta$$

$$T_m + \delta$$

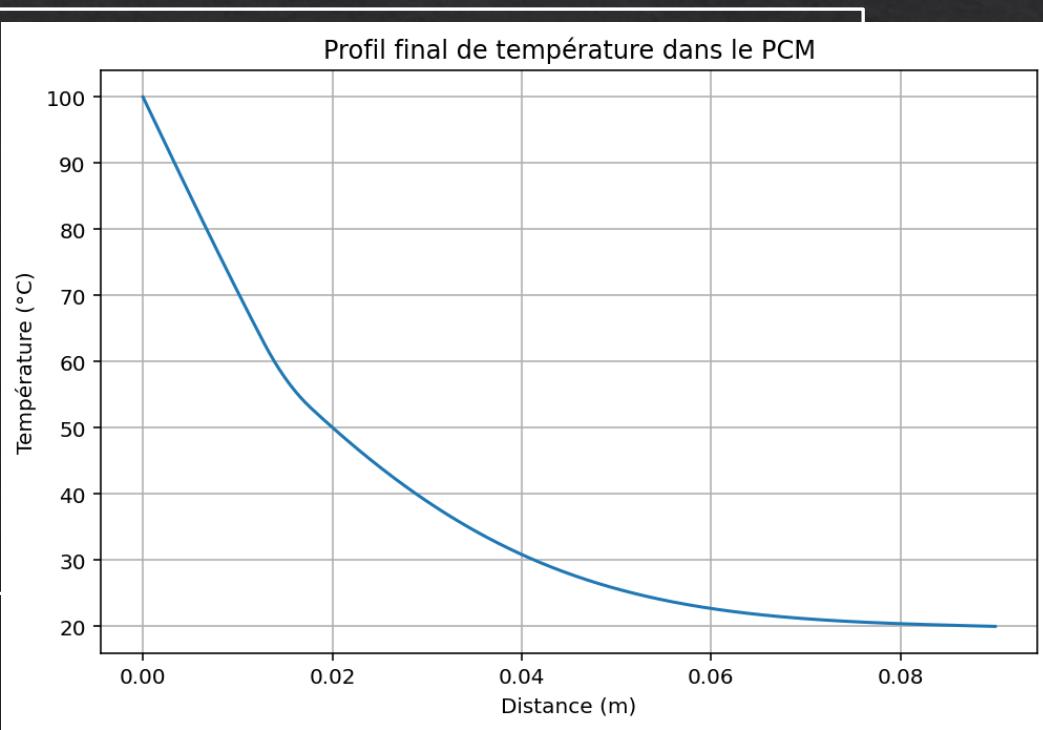
Premier essaie



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

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

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

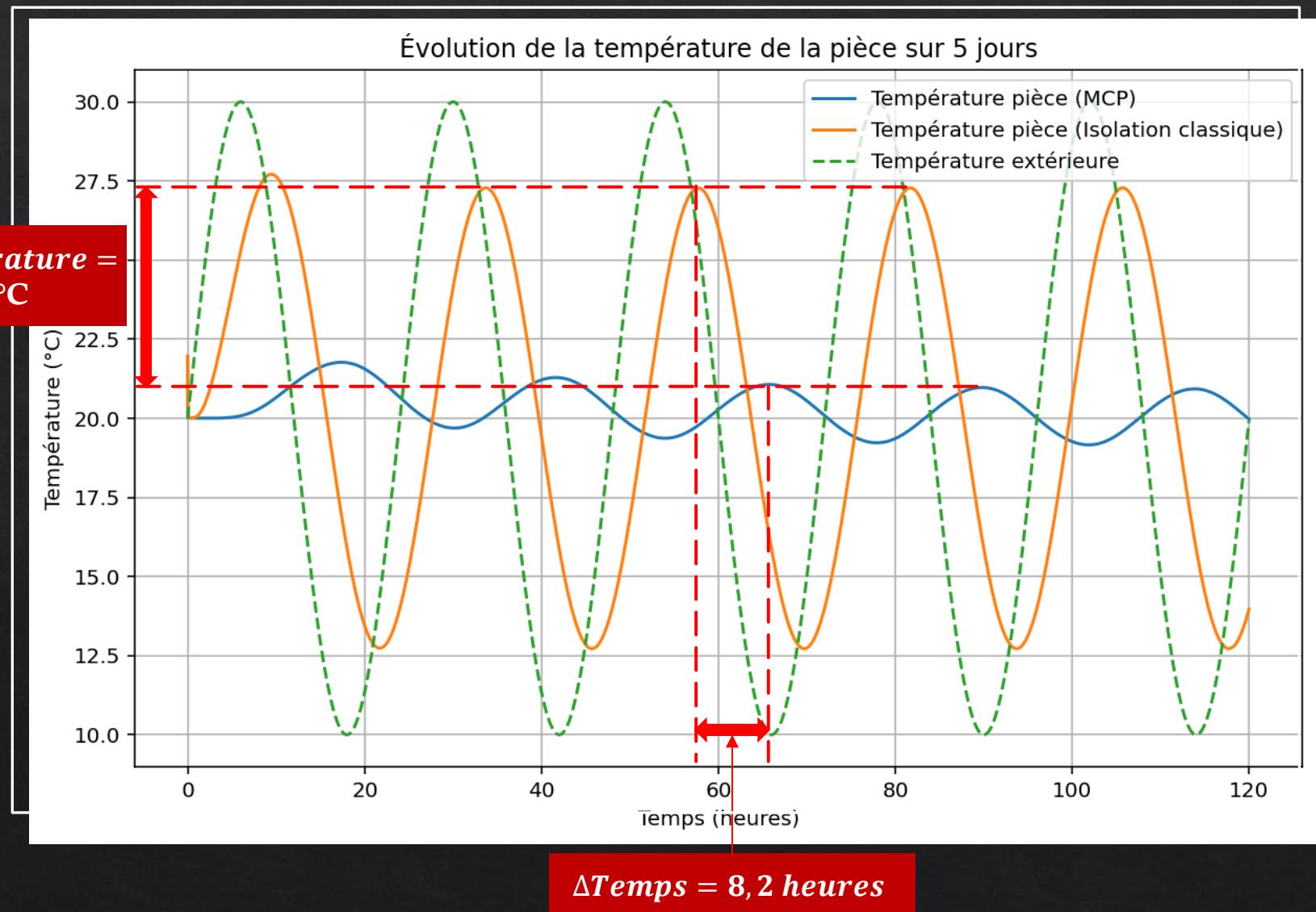


Résultat numérique

Insérer essai expérimental

Résultat expérimental

Évolution de la température dans une pièce avec une isolation MCP



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

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

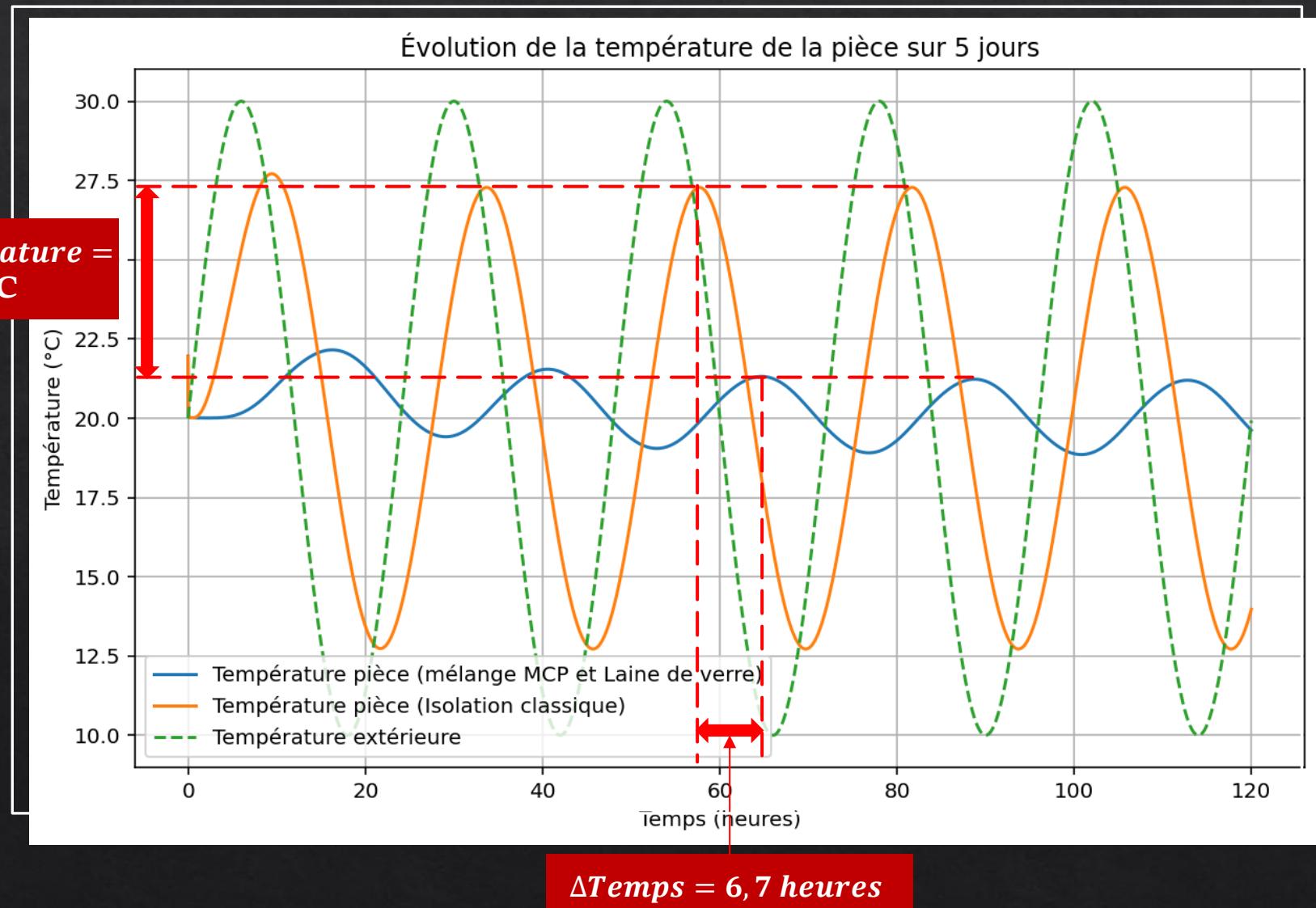
Décomposition de l'enthalpie

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

f : fraction de matériau liquide dans le noeud considéré

l : enthalpie massique de fusion

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



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

```
// 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
}
```

```

def compute_H_pcm(T):
    """
    Calcule l'enthalpie H pour un profil de température T, en tenant compte du PCM.
    """
    H = np.zeros_like(T)
    mask_solid = T < (T_m - delta)
    mask_liquid = T > (T_m + delta)
    mask_mushy = ~(mask_solid | mask_liquid)

    H[mask_solid] = (rho * cp * T[mask_solid]) * p + (rho_v * cp_v * T[mask_solid]) * (1-p)
    H[mask_liquid] = (rho * cp * T[mask_liquid] + rho * L_latent) * p + (rho_v * cp_v * T[mask_liquid]) * (1-p)
    H[mask_mushy] = (rho * cp * T[mask_mushy] \
                     + rho * L_latent * ((T[mask_mushy] - (T_m - delta)) / (2 * delta))) * p \
                     +(rho_v * cp_v * T[mask_mushy]) * (1-p)
    return H

def T_from_H_pcm(H):
    """
    Récupère la température T à partir de l'enthalpie H pour le matériau PCM.
    """
    T = np.zeros_like(H)
    H_low = (rho * cp * (T_m - delta)) * p + (rho_v * cp_v * (T_m - delta)) * (1-p)
    H_high = (rho * cp * (T_m + delta) + rho * L_latent) * p + (rho_v * cp_v * (T_m + delta)) * (1-p)

    mask_solid = H < H_low
    mask_liquid = H > H_high
    mask_mushy = ~(mask_solid | mask_liquid)

    # Zone solide
    T[mask_solid] = H[mask_solid] / ( rho * cp * p + rho_v * cp_v * (1 - p) )
    # Zone liquide
    T[mask_liquid] = (H[mask_liquid] - rho * L_latent * p) / ( (rho * cp * p) + rho_v * cp_v * (1 - p) )
    # Zone mushy (transition)
    T[mask_mushy] = (H[mask_mushy] + (rho * L_latent / (2 * delta)) * (T_m - delta) * p) \
                    / ( (rho * cp + (rho * L_latent / (2 * delta))) * p + rho_v * cp_v * (1 - p) )
    return T

```

```

for step in range(steps):
    t = step * dt
    # Condition extérieure imposée à x = 0
    T_ext_val = T_ext(t)

    # ----- Mise à jour pour la simulation PCM -----
    # Condition limite extérieure
    T_pcm[0] = T_ext_val
    H_pcm[0] = compute_H_pcm(np.array([T_pcm[0]]))[0]

    # Condition convective intérieure (x = L)
    # Approximation par DF:  $T_{pcm}[-1] = (T_{pcm}[-2] + (h_{conv} * dx / k) * T_{room\_pcm}) / (1 + (h_{conv} * dx / k))$ 
    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))))
    H_pcm[-1] = compute_H_pcm(np.array([T_pcm[-1]]))[0]

    # Récupérer T à partir de H pour mettre à jour la conduction
    T_temp_pcm = T_from_H_pcm(H_pcm)
    H_new_pcm = H_pcm.copy()
    # Mise à jour des noeuds internes (schéma explicite)
    for i in range(1, N - 1):
        d2Tdx2 = (T_temp_pcm[i+1] - 2 * T_temp_pcm[i] + T_temp_pcm[i-1]) / dx**2
        #  $dH/dt = k * d2T/dx^2$ 
        H_new_pcm[i] = H_pcm[i] + dt * (k * p + k_v * (1 - p)) * d2Tdx2
    H_pcm = H_new_pcm.copy()
    T_pcm = T_from_H_pcm(H_pcm)

    # ----- Mise à jour pour la simulation en isolation classique -----
    T_classic[0] = T_ext_val # condition extérieure
    T_classic[-1] = (T_classic[-2] + (h_conv * dx / k) * T_room_classic) / (1 + (h_conv * dx / k))
    T_new_classic = T_classic.copy()
    for i in range(1, N - 1):
        d2Tdx2_classic = (T_classic[i+1] - 2 * T_classic[i] + T_classic[i-1]) / dx**2
        # Equation classique:  $\rho * c_p * dT/dt = k * d2T/dx^2$ 
        T_new_classic[i] = T_classic[i] + dt * (k / (rho * cp)) * d2Tdx2_classic
    T_classic = T_new_classic.copy()

    # ----- Mise à jour de la température de la pièce -----
    #  $C_{room} * dT_{room}/dt = h_{conv} * A * (T_{interior\_wall} - T_{room})$ 
    T_room_pcm += dt * (h_conv * A / C_room) * (T_pcm[-1] - T_room_pcm)
    T_room_classic += dt * (h_conv * A / C_room) * (T_classic[-1] - T_room_classic)

```

```
# %%
# -----
# 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)

# %%
# -----
# Paramètres de simulation
# -----
dt = 1             # pas de temps
total_time = 86400*5 # 24 h en secondes
steps = int(total_time / dt)

# Paramètres de la pièce (modèle lumpé)
h_conv = 10          # coefficient convectif intérieur (W/(m^2.K))
A = 24.0             # surface d'échange (m^2)
C_room = 1e4           # capacité thermique de la pièce (J/K)

# Fonction de température extérieure (sinusoïdale sur 24 h)
def T_ext(t):
    # Moyenne 20 °C, amplitude 10 °C
    return 20 + 10 * np.sin(2 * np.pi * t / 86400)
```