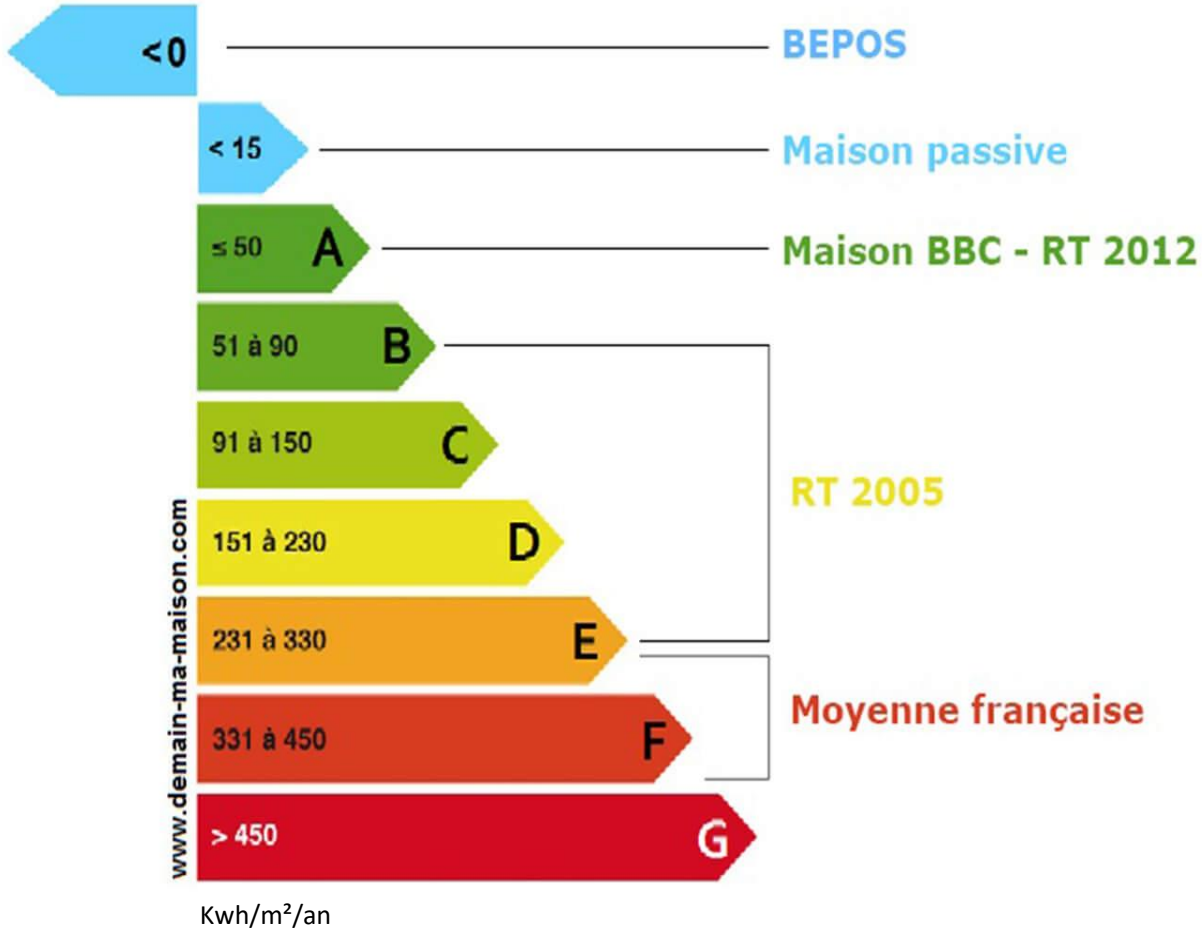
Paul Maréchal

N° candidat : 15085

Energétique des bâtiments

2.

Comment rendre un bâtiment performant énergétiquement ?



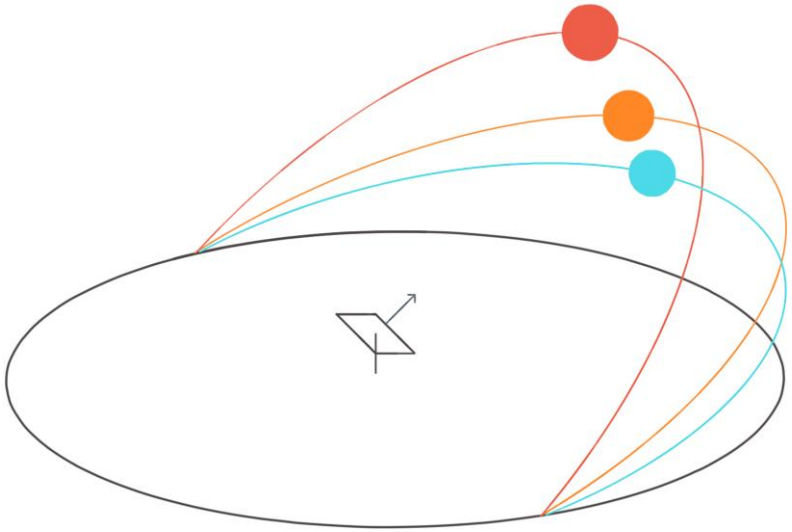
Sommaire :

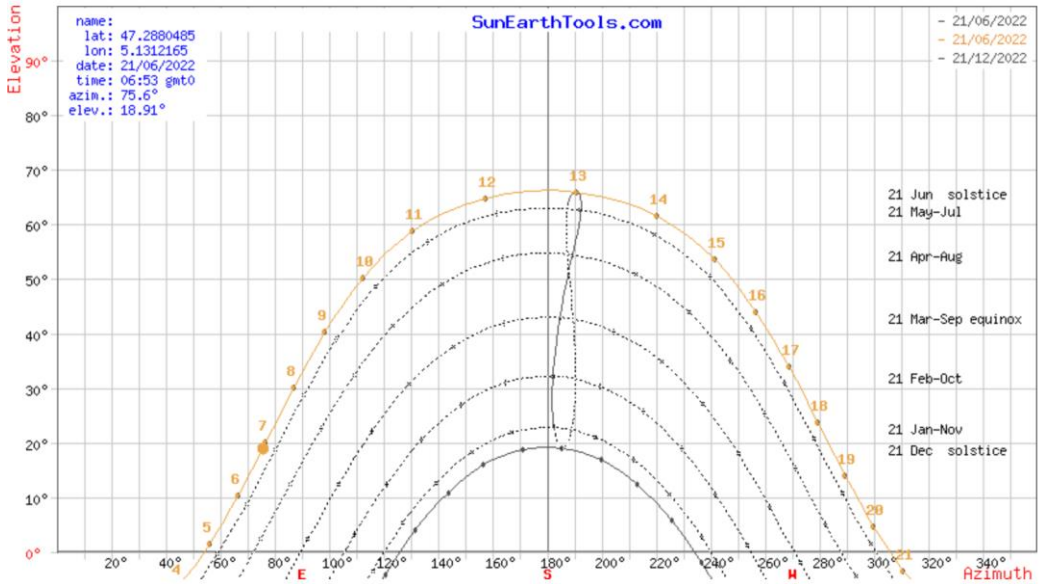
4.

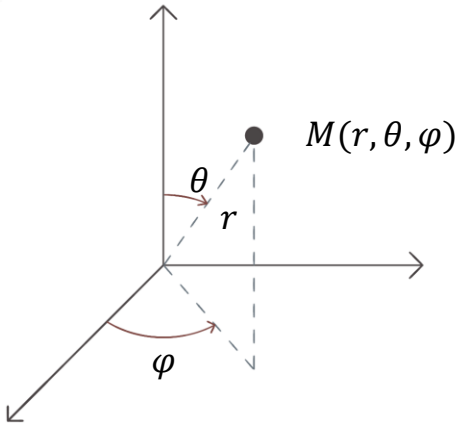
Apports énergétiques d’un bâtiment

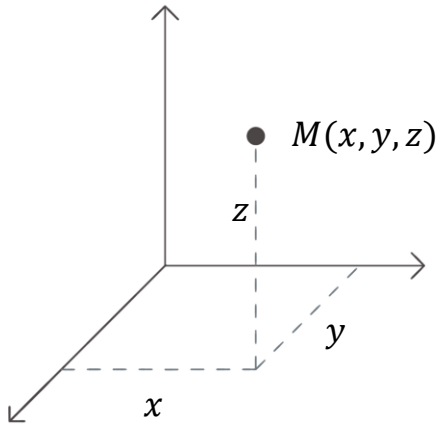
5.

Modélisation



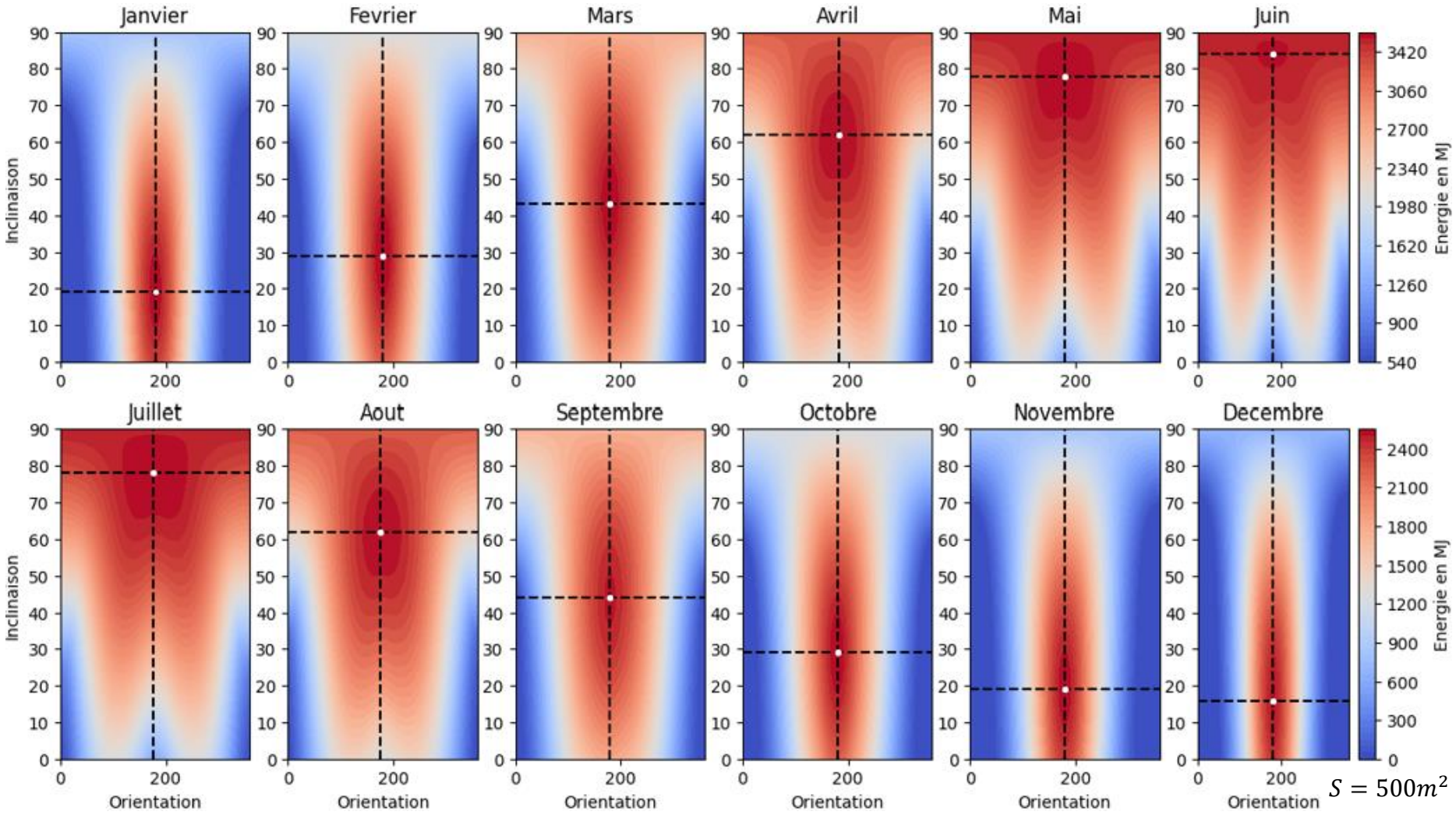






6.

Énergie produite en fonction de l'orientation et de l'inclinaison des panneaux

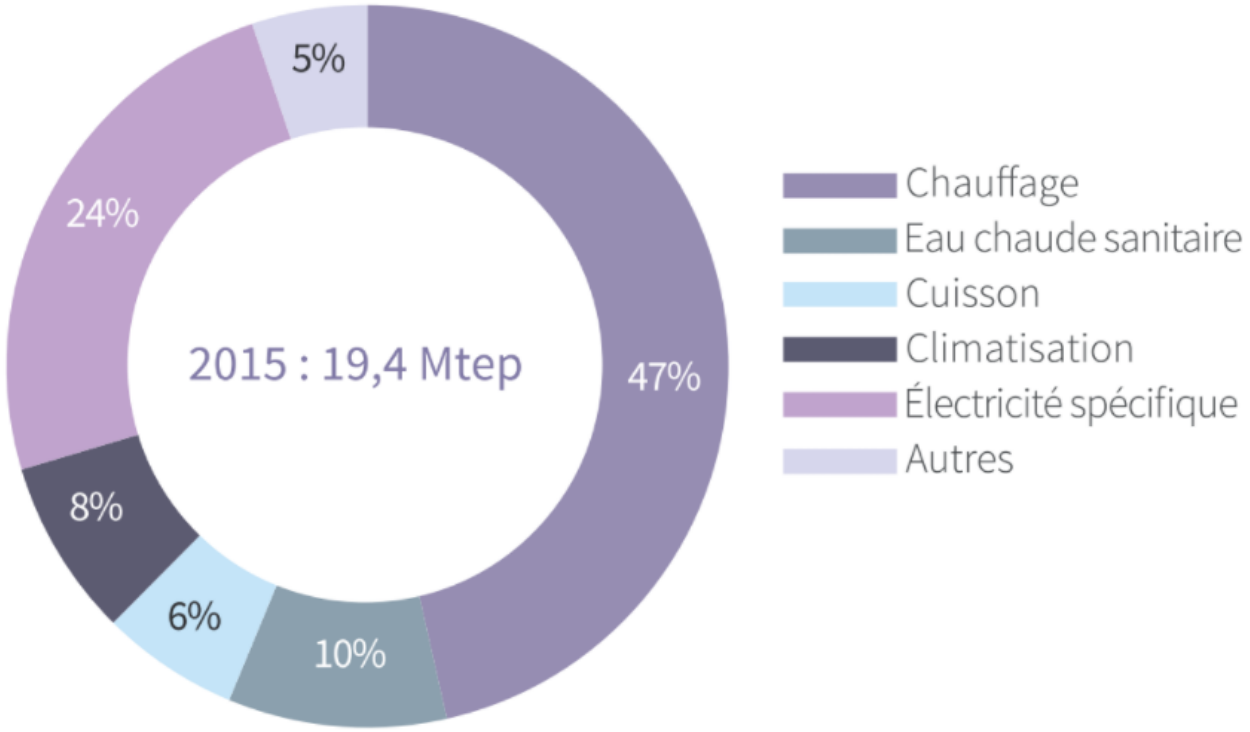


Inclinaison optimale : 43° Orientation optimale : 180°

7.

Consommation énergétique d’un bâtiment

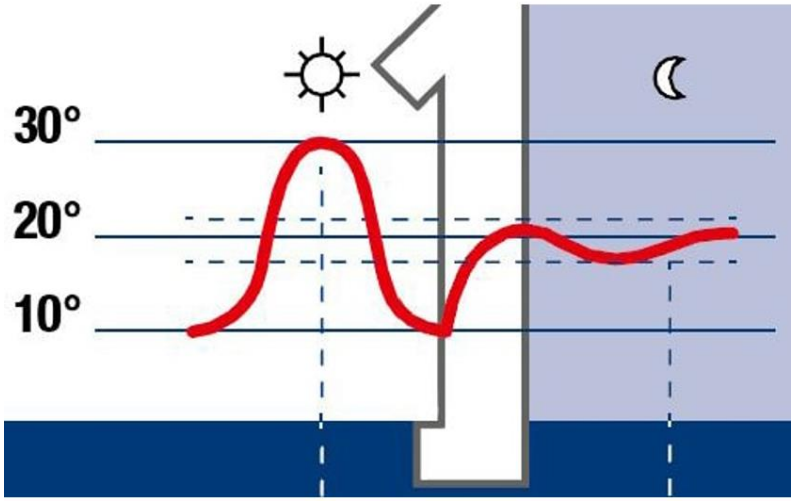
Consommation des bâtiments

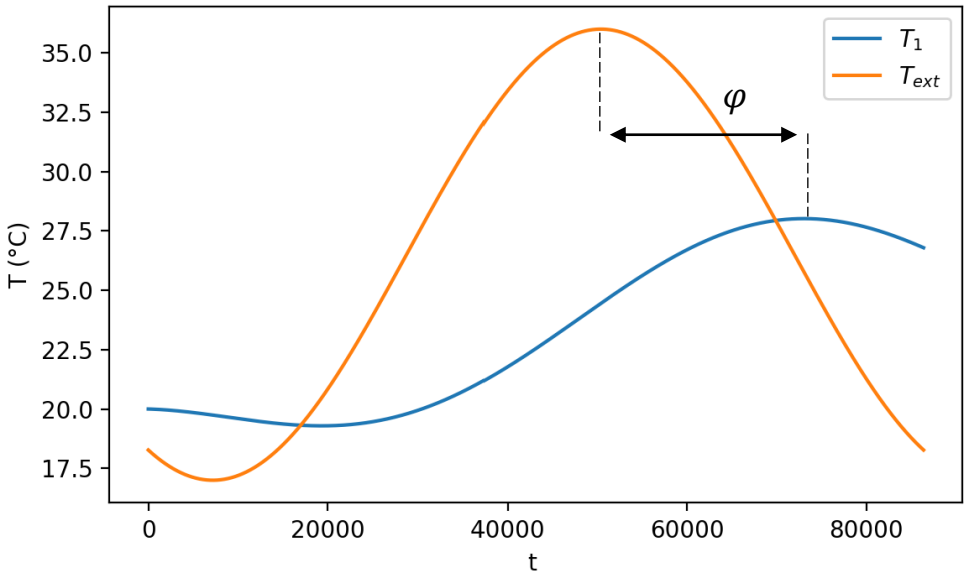


Source : CEREN (Centre d'études et de recherches économiques sur l'énergie)

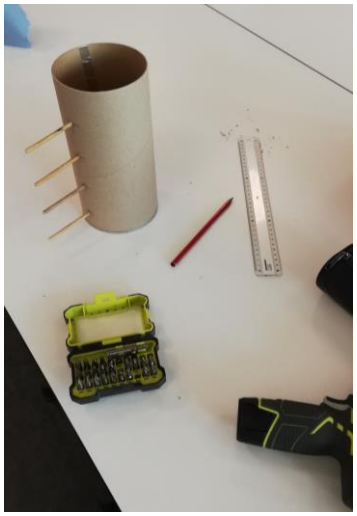
Réduction de la consommation

Intérêt du déphasage thermique



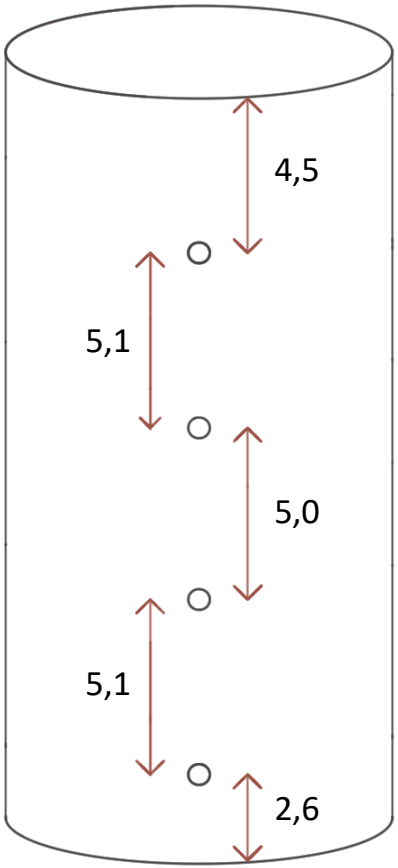


Mise en place de l’expérience :

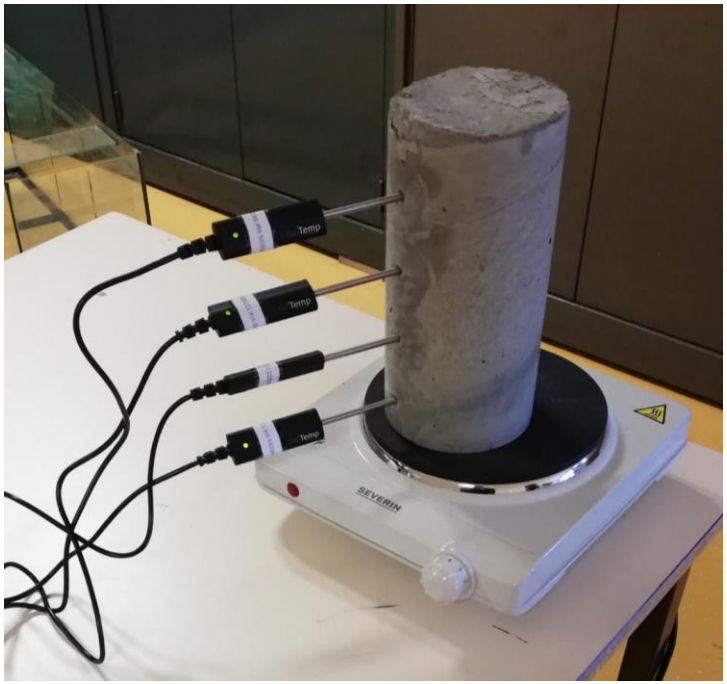


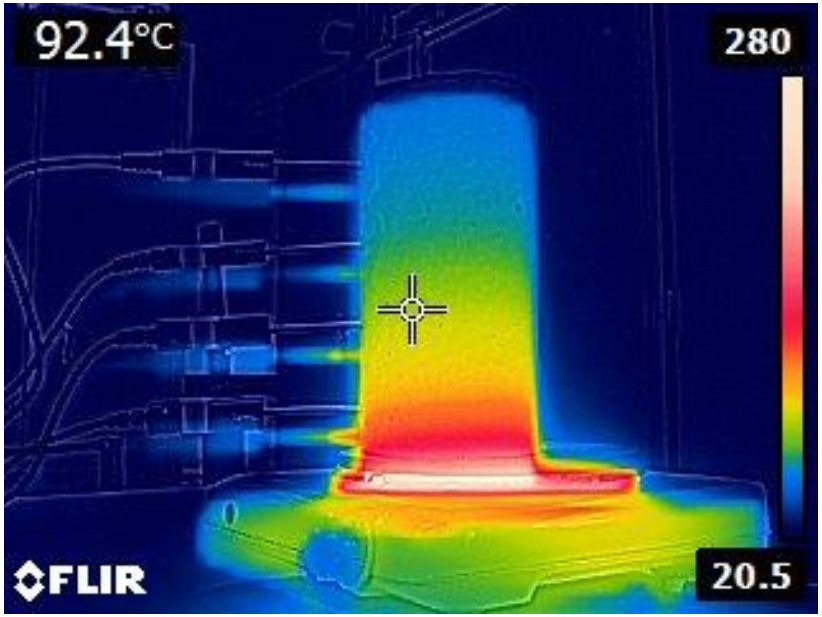
11.

Expérience :



Longueurs en cm

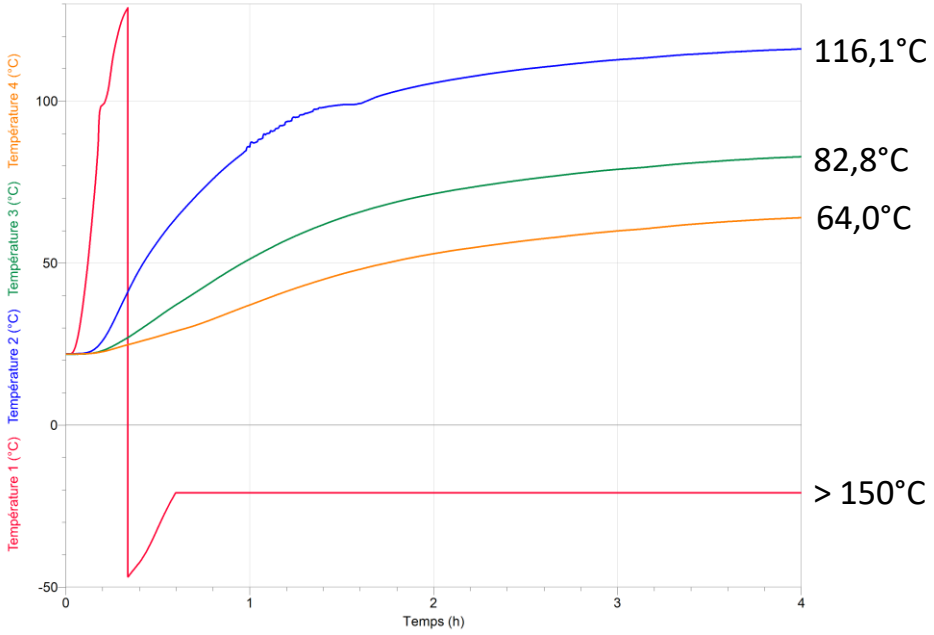




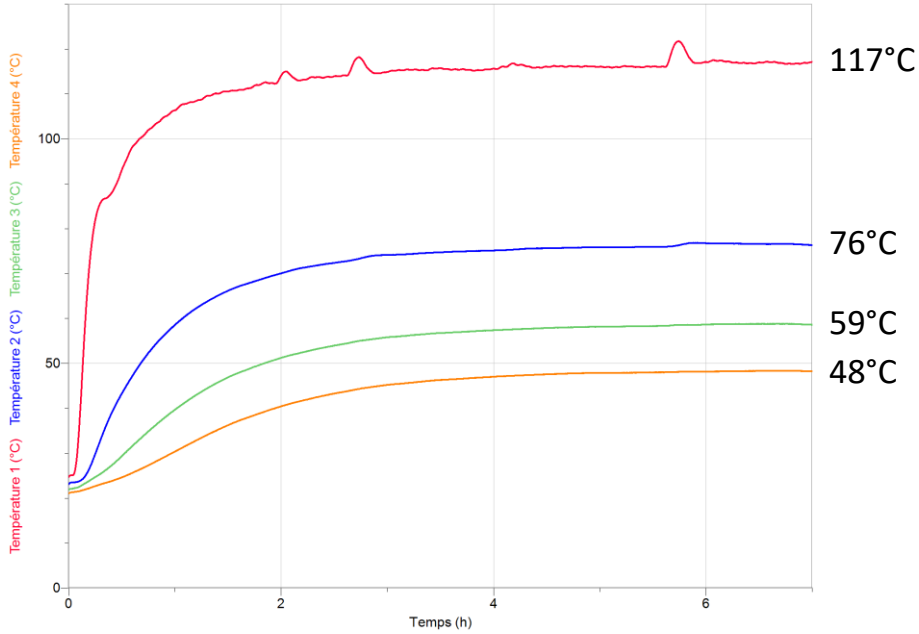
12.

Résultats :

Température en fonction du temps



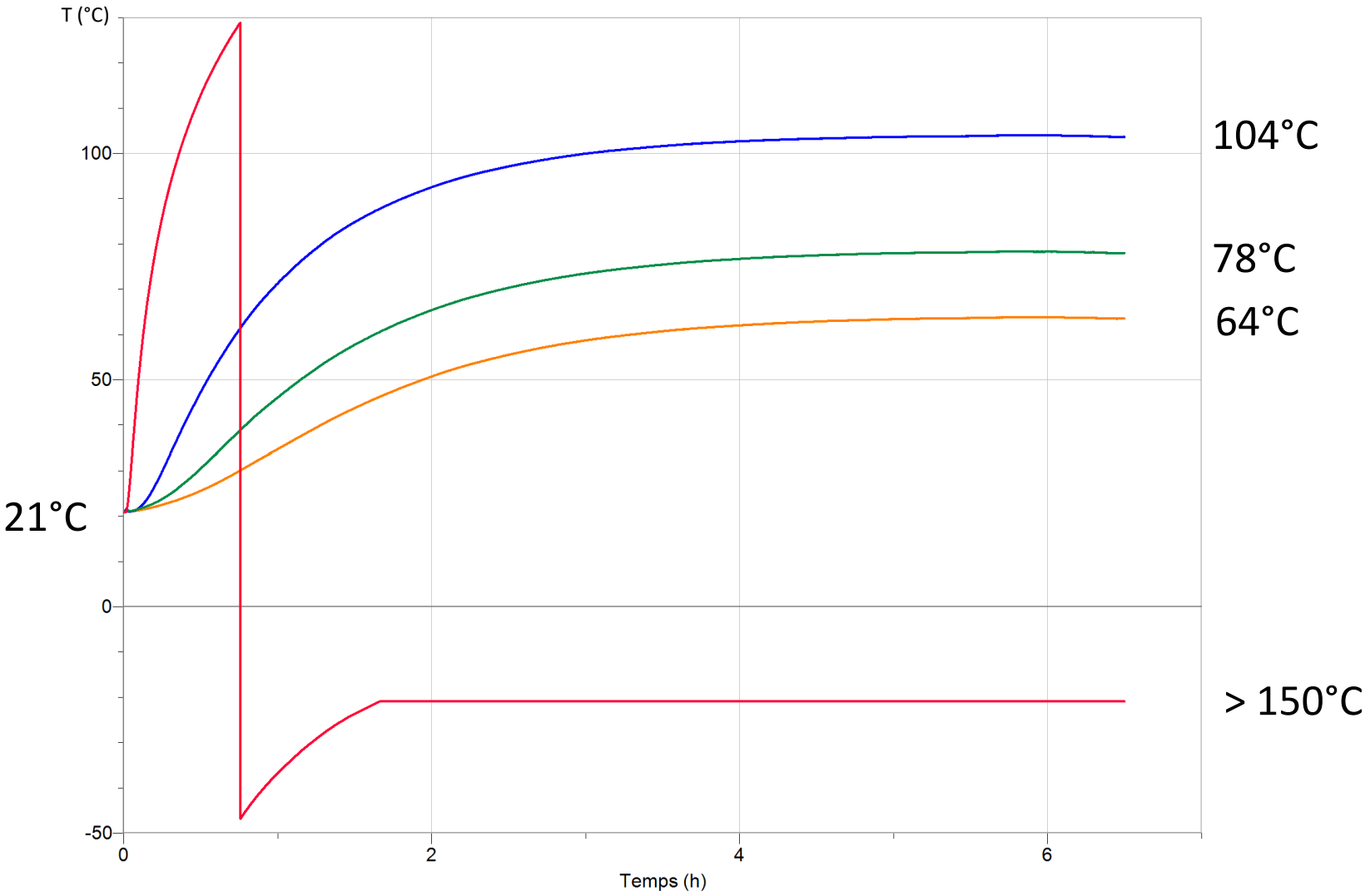
1er essai



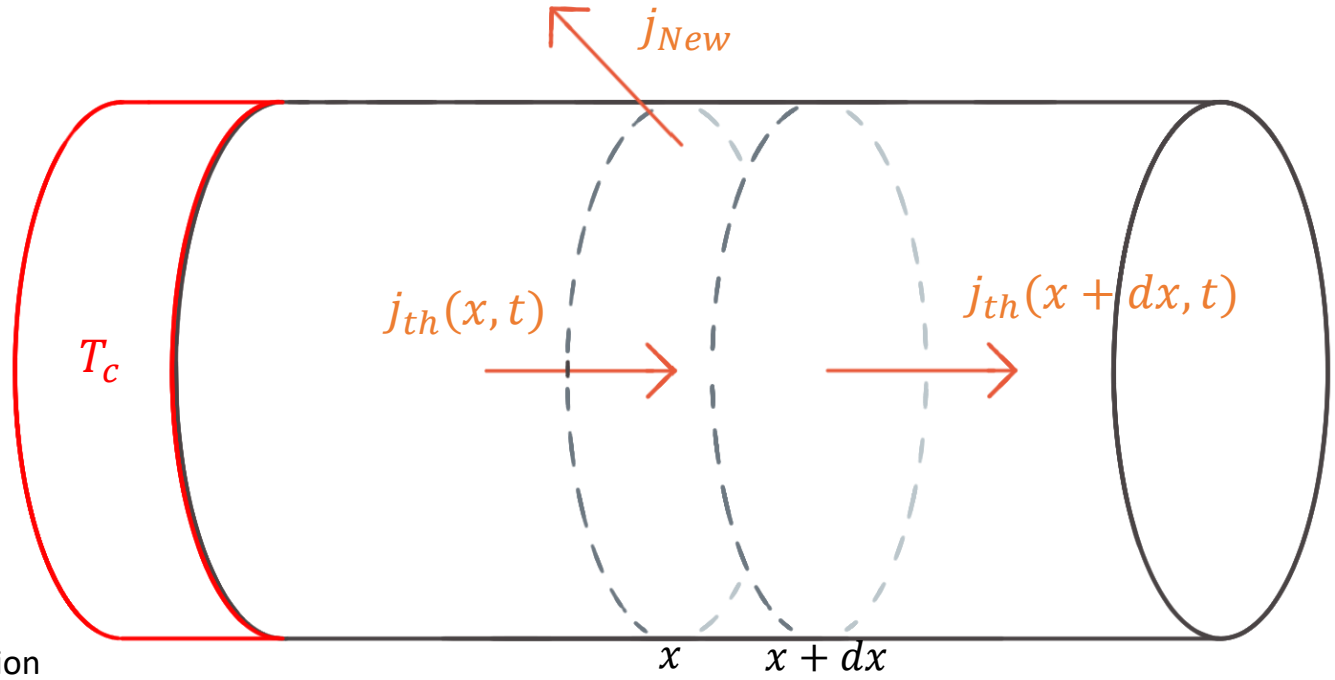
2ème essai

13.

Température en fonction du temps



Bilan d'énergie thermique dans le bloc



λ : Conductivité thermique
ρ : Masse volumique
c_p : Capacité thermique massique
h_{cc} : Coefficient de conducto-convection

$$\rho c \frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2} + h_{cc}(T_{ext} - T)\frac{2}{r}$$

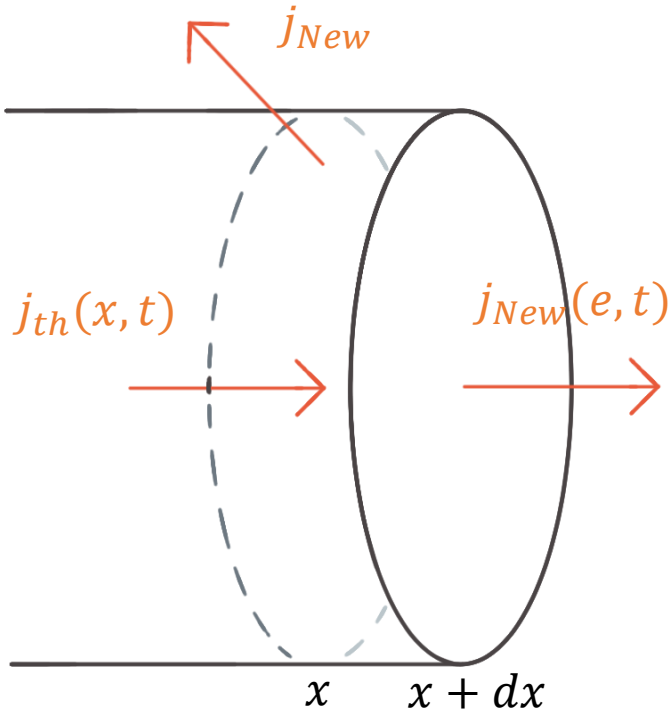
$$\rho c \frac{T_x^{t+1} - T_x^t}{n_t} = \lambda \frac{T_{x+1}^t - 2T_x^t + T_{x-1}^t}{n_x^2} - h_{cc}(T_x^t - T_{ext})\frac{2}{r}$$

$$T_x^{t+1} = T_x^t \left(1 - \frac{2\lambda n_t}{\rho c n_x^2} - \frac{2h_{cc}n_t}{\rho c r}\right) + T_{x+1}^t \frac{\lambda n_t}{\rho c n_x^2} + T_{x-1}^t \frac{\lambda n_t}{\rho c n_x^2} + T_{ext}\frac{2h_{cc}n_t}{\rho c r}$$

15.

Bilan d'énergie thermique à l'extrémité du bloc

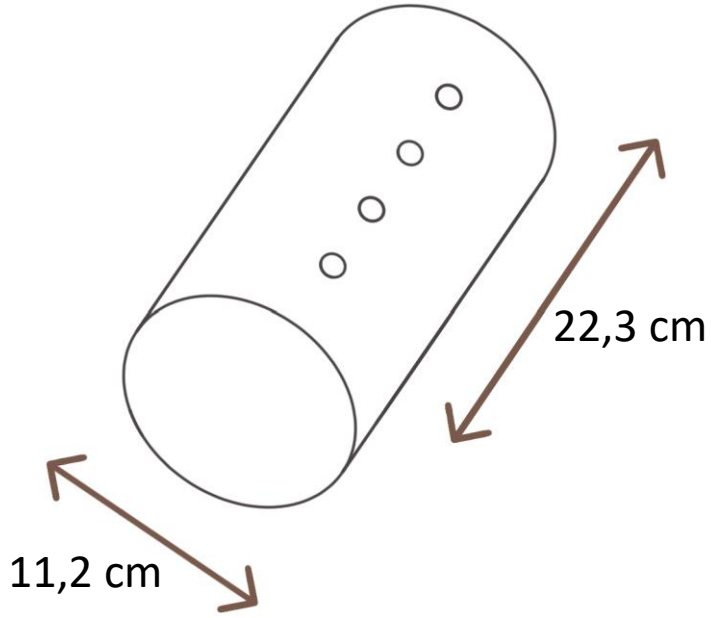
$$\rho c \frac{\partial T}{\partial t} = -\frac{\lambda}{dx}\frac{\partial T}{\partial x} + \frac{h_{cc}(T_{ext} - T)}{dx} + h_{cc}(T_{ext} - T)\frac{2}{r}$$



$$\rho c \frac{T_x^{t+1} - T_x^t}{n_t} = -\lambda \frac{T_x^t - T_{x-1}^t}{n_x^2} + \frac{h_{cc}(T_{ext} - T_x^t)}{n_x} + h_{cc}(T_{ext} - T_x^t)\frac{2}{r}$$

$$T_x^{t+1} = T_x^t \left(1 - \frac{\lambda n_t}{\rho c n_x^2} - \frac{2h_{cc}n_t}{\rho c r} - \frac{h_{cc}n_t}{\rho c n_x}\right) + T_{x-1}^t \frac{\lambda n_t}{\rho c n_x^2} + T_{ext}\left(\frac{2h_{cc}n_t}{\rho c r} + \frac{h_{cc}n_t}{\rho c n_x}\right)$$

Caractéristiques du bloc de béton

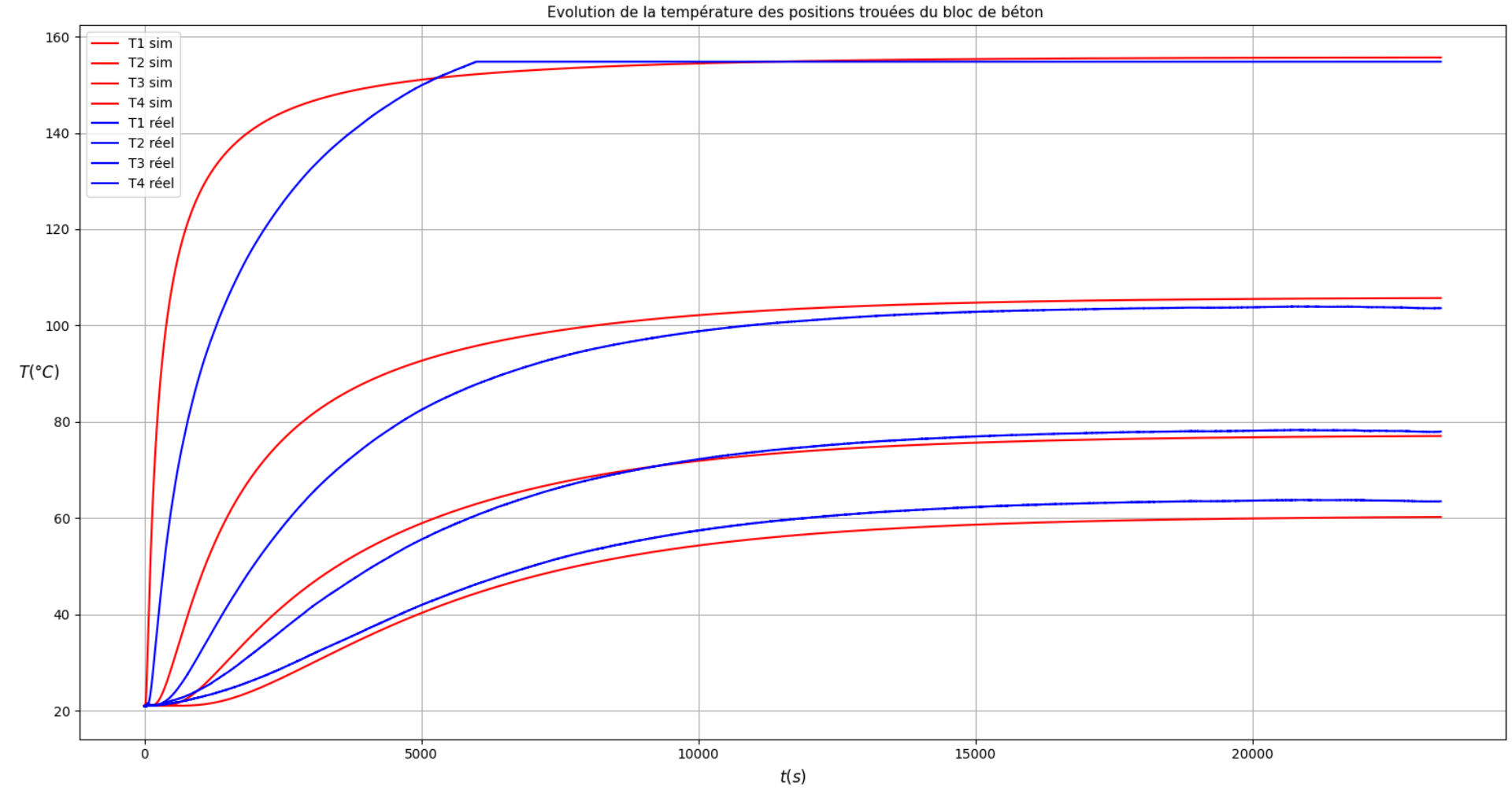


- Béton : $\lambda = 4\ \mathrm{W}.K^{-1}.m^{-1}$
- $\rho = \frac{m}{V} = 2401\ kg.m^{-3}$

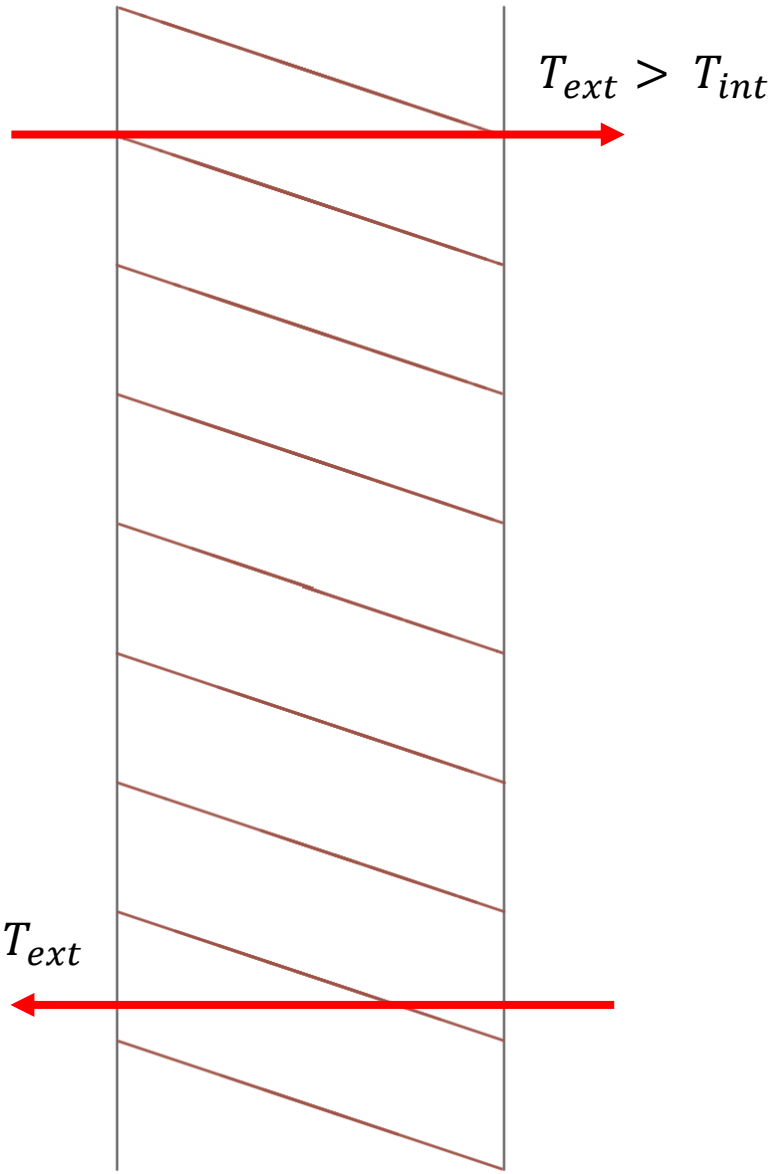
$(V = 2{,}197.10^{-3}\ m^3)$

- Béton : $c_p = 1100\ J.K^{-1}.kg^{-1}$

17. # Résolution numérique :



Adaptation pour un mur



T_{ext}

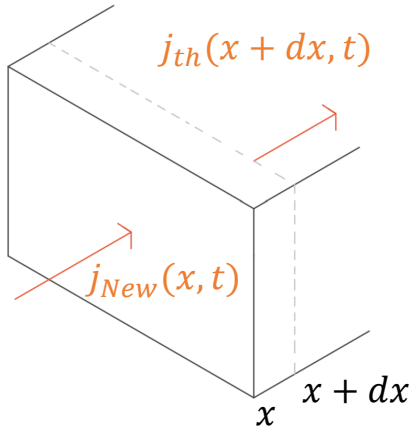
$= T_{moy} + \frac{\Delta T}{2} \sin\left(t \frac{2\pi}{86400}\right)$

T_{int}

Bilan d'énergie thermique d'un mur

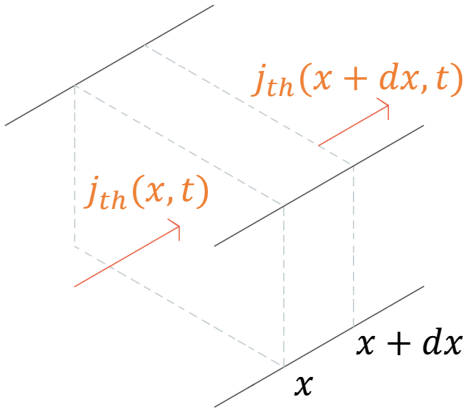
$$\frac{\partial T}{\partial t} = \frac{h_{cc}(T_{ext} - T)}{\rho\ c\ dx} + \frac{\lambda}{\rho\ c\ dx}\frac{\partial T}{\partial x}$$

$$T_x^{t+1} = T_x^t \left(1 - \frac{\lambda n_t}{\rho c n_x^2} - \frac{h_{cc} n_t}{\rho c n_x}\right) + T_{x+1}^t \frac{\lambda n_t}{\rho c n_x^2} + T_{ext} \frac{h_{cc} n_t}{\rho c n_x}$$



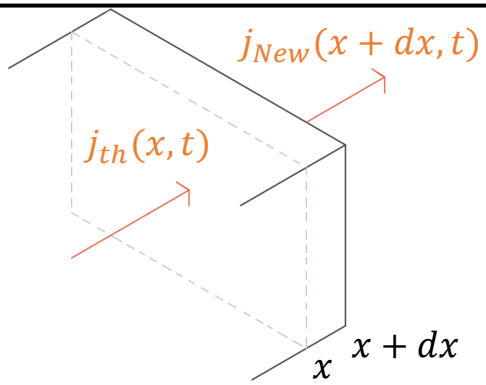
$$\rho\ c\ \frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2}$$

$$T_x^{t+1} = T_x^t \left(1 - \frac{2\lambda n_t}{\rho c n_x^2}\right) + T_{x+1}^t \frac{\lambda n_t}{\rho c n_x^2} + T_{x-1}^t \frac{\lambda n_t}{\rho c n_x^2}$$



$$\frac{\partial T}{\partial t} = -\frac{\lambda}{\rho\ c\ dx}\frac{\partial T}{\partial x} + \frac{h_{cc}(T_{int} - T)}{\rho\ c\ dx}$$

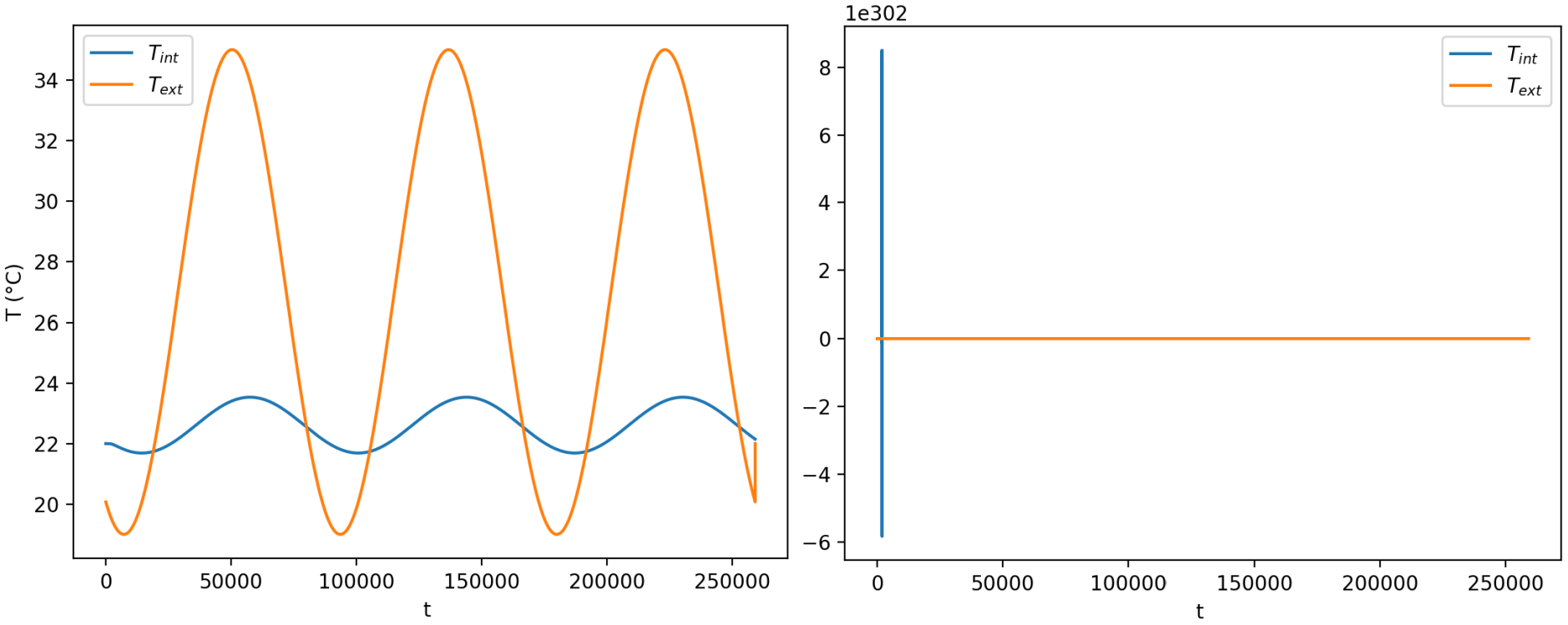
$$T_x^{t+1} = T_x^t \left(1 - \frac{\lambda n_t}{\rho c n_x^2} - \frac{h_{cc} n_t}{\rho c n_x}\right) + T_{x-1}^t \frac{\lambda n_t}{\rho c n_x^2} + T_{int} \frac{h_{cc} n_t}{\rho c n_x}$$



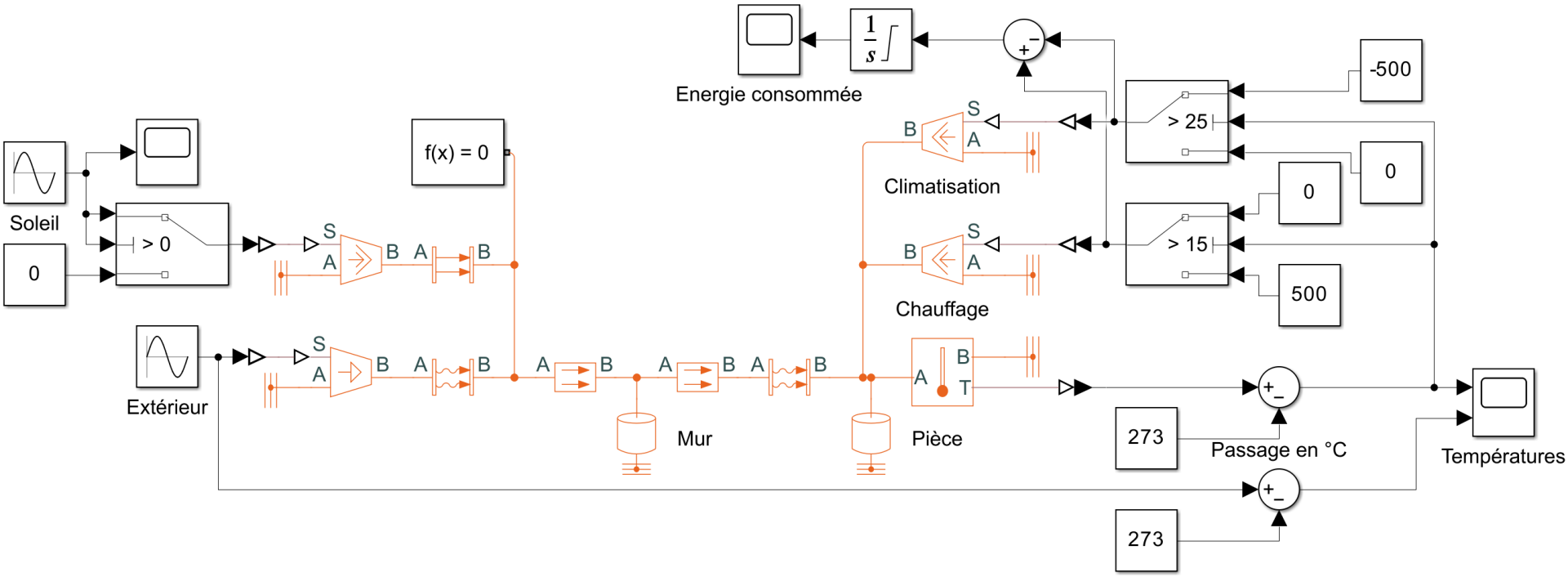
20.

Résolution numérique

Températures intérieure et extérieure en fonction du temps



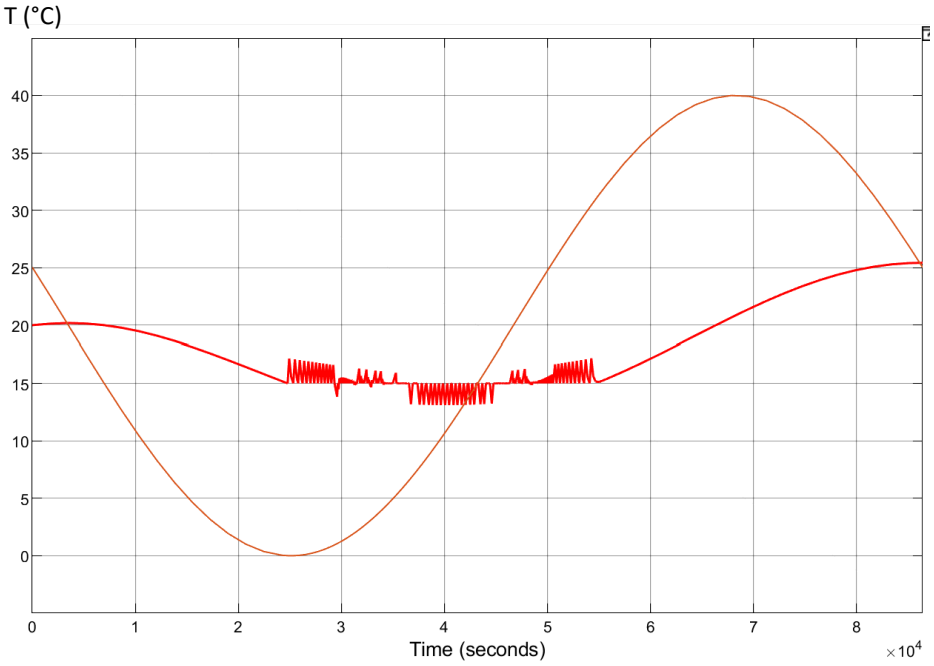
Simulation


Energie consommée
1/s
Soleil
0
> 0
f(x) = 0
S
A
B
Climatisation
Chauffage
> 25
> 15
-500
0
0
500
Extérieur
Mur
Pièce
T
273
273
Passage en °C
Températures

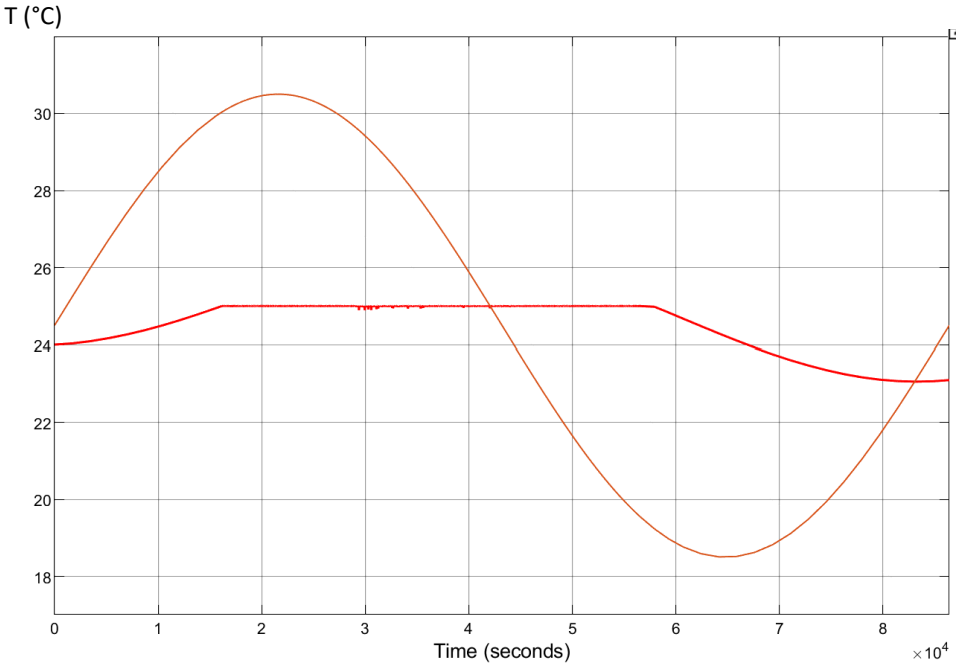
22.

Simulation de l’évolution de la température en fonction du temps

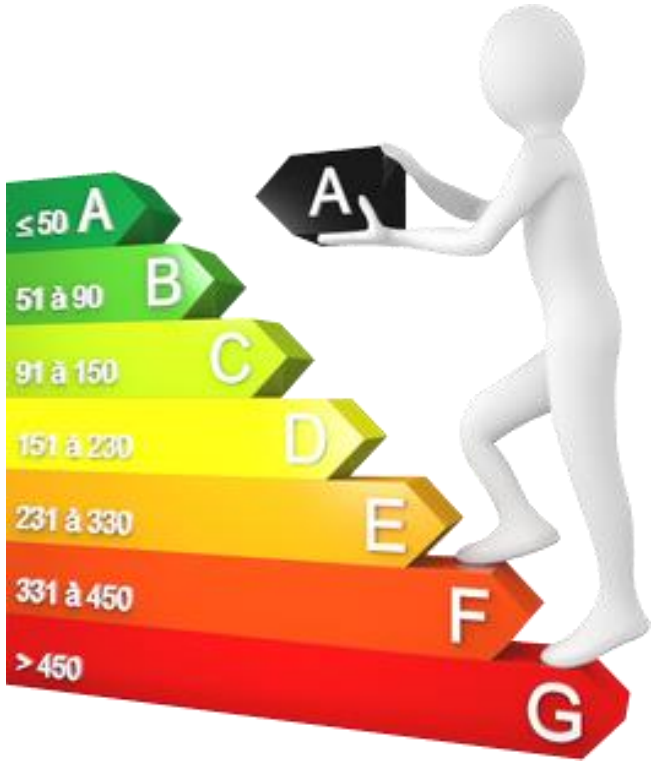
Chauffage à 15 °C



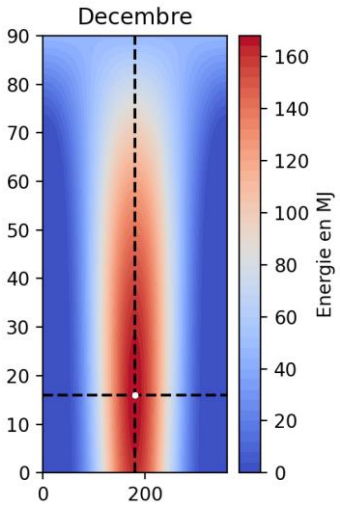
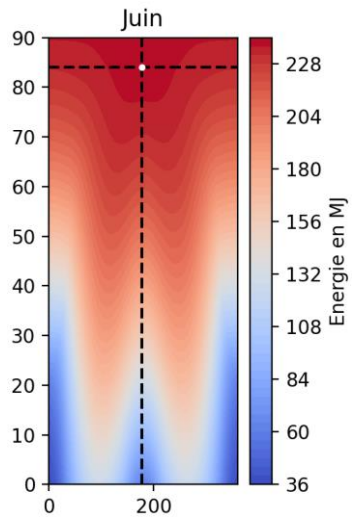
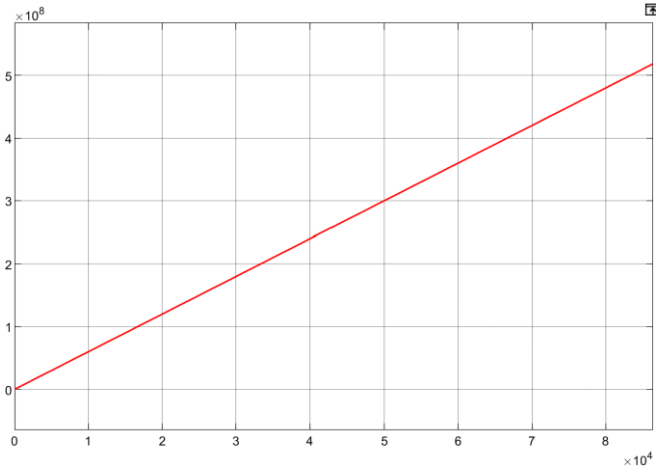
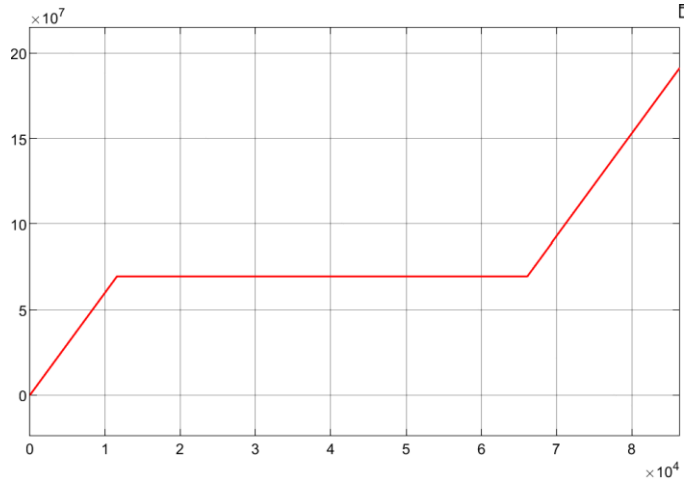
Climatisation à 25 °C



Bilan énergétique

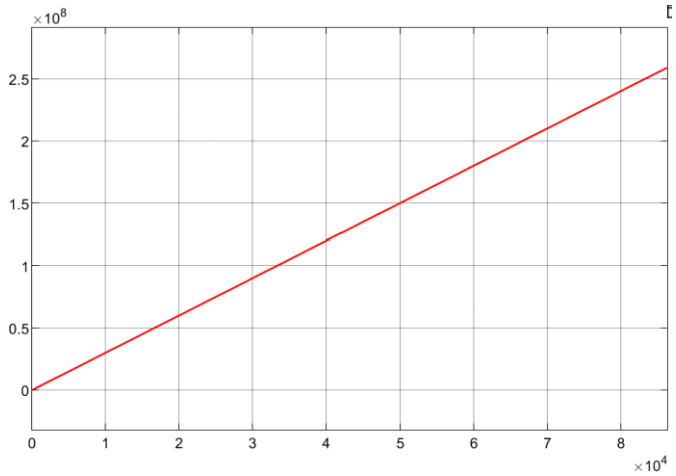


Bilan pour un bâtiment de 80m²

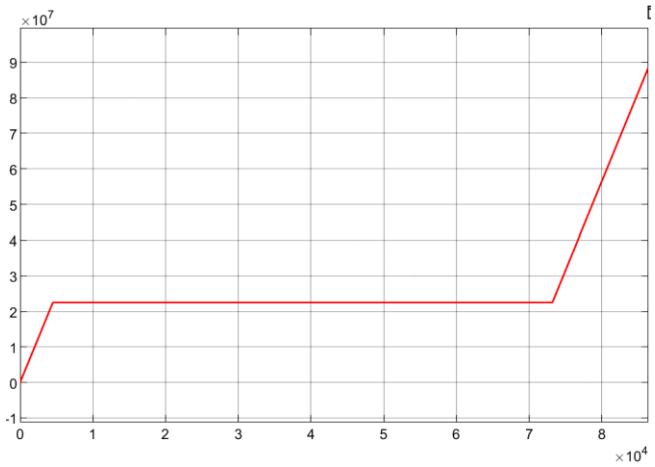
Saison	Hiver 5°C – 15°C	Été 18°C – 35°C
Apport énergétique	 $E_a = 160\ MJ$	 $E_a = 240\ MJ$
Dépense Énergétique Sans optimiser le déphasage	 $E_{cons} = 518\ MJ$	 $E_{cons} = 191\ MJ$

25.

Dépense Énergétique avec déphasage



$E_{cons} = 259\ MJ$



$E_{cons} = 88\ MJ$

Bilan

Sans optimisation

$E_{bil} = 160 - 518 = -358\ MJ$

$E_{bil} = 240 - 191 = +49 MJ$

Avec optimisation

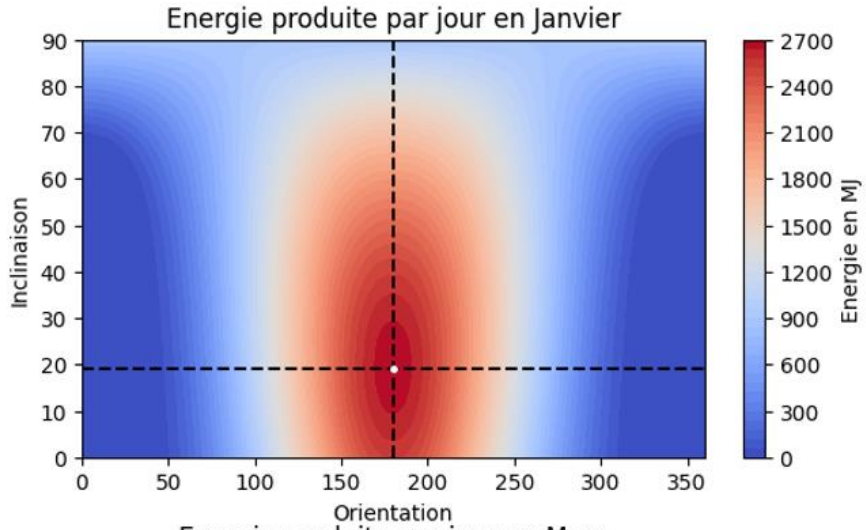
$E_{bil} = 160 - 259 = -99\ MJ$

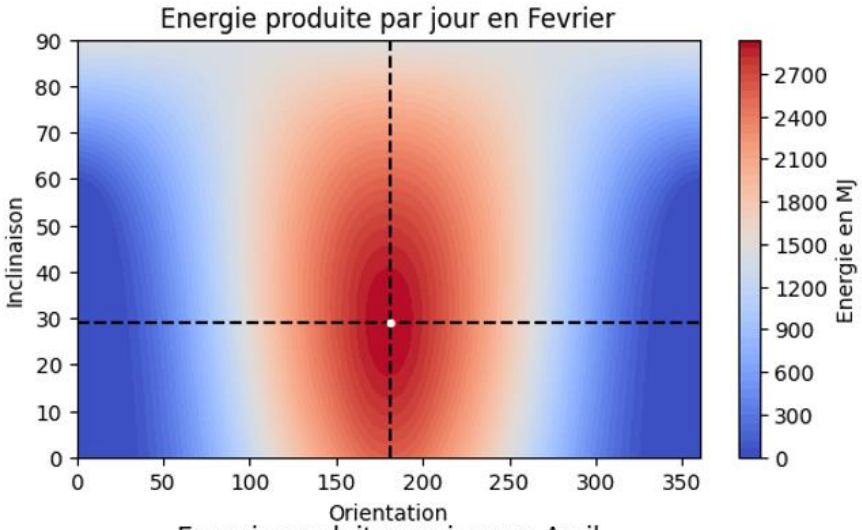
$E_{bil} = 240 - 88 = +152\ MJ$

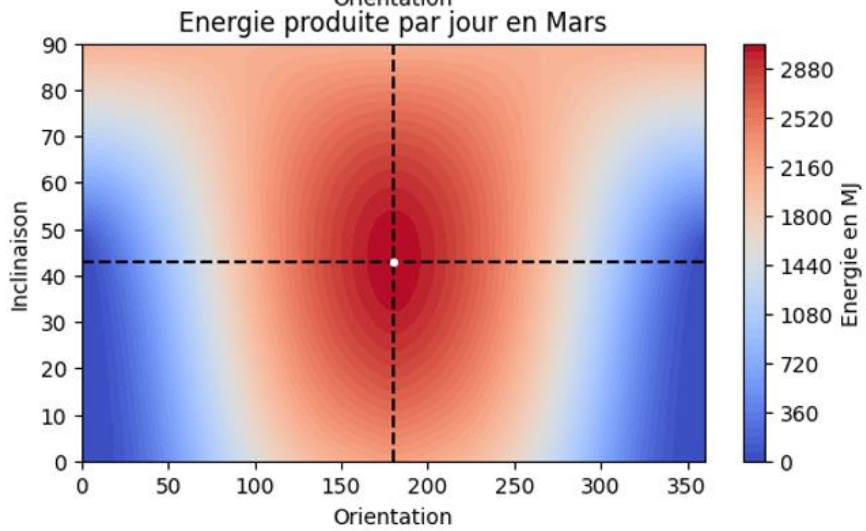
Conclusion

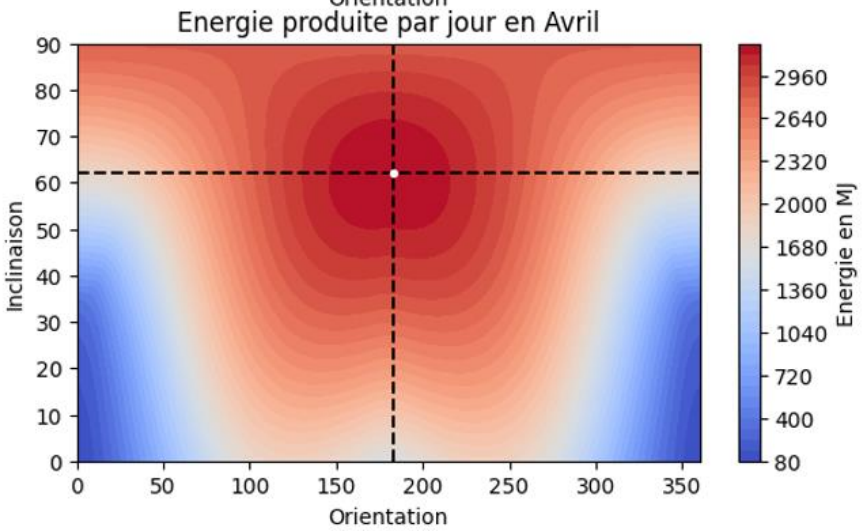
Remerciements : Ludovic Avril - ESTP

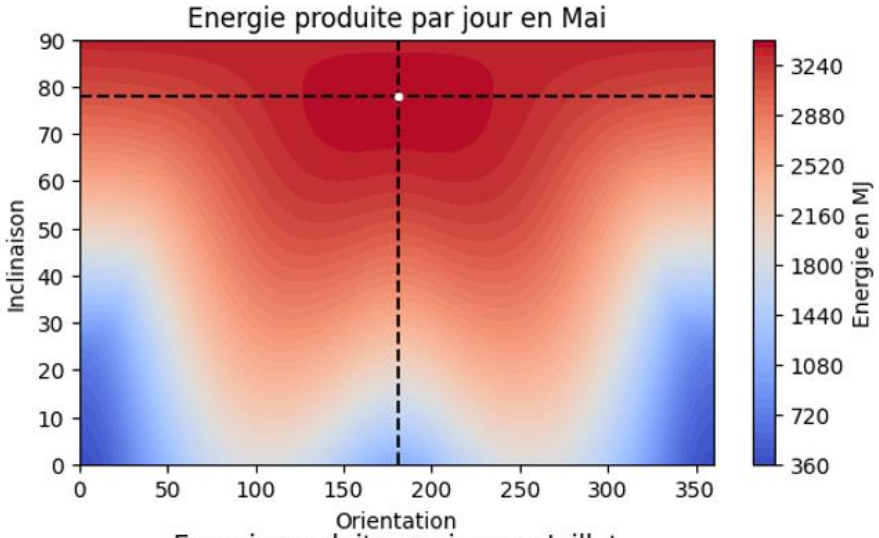
Annexe

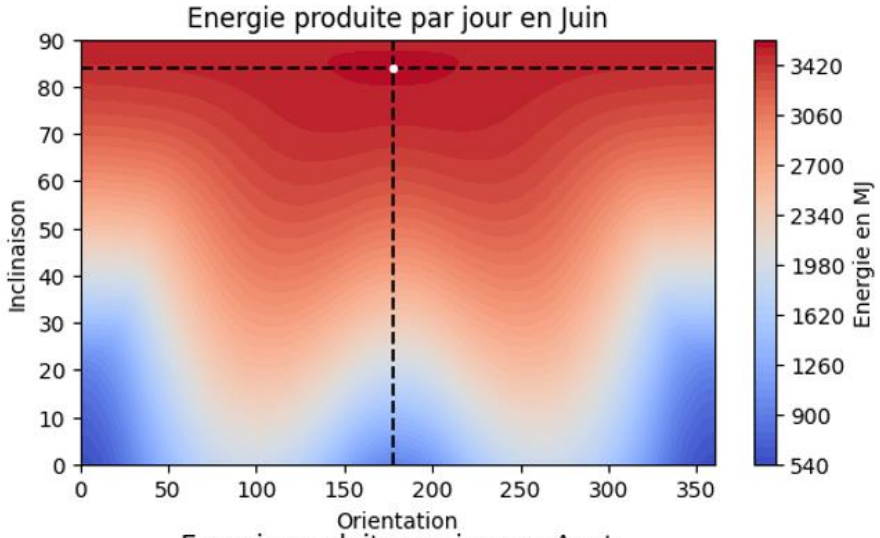

Energie produite par jour en Janvier
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
0
300
600
900
1200
1500
1800
2100
2400
2700

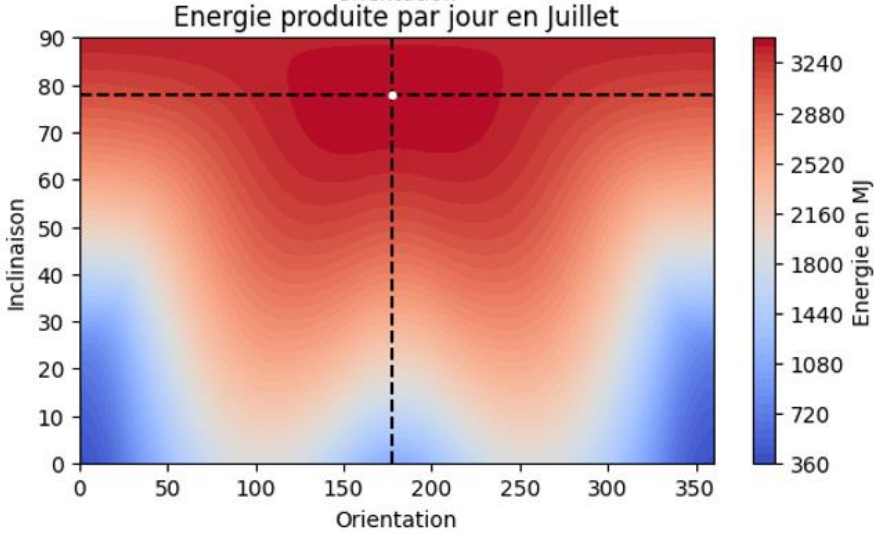

Energie produite par jour en Fevrier
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
0
300
600
900
1200
1500
1800
2100
2400
2700

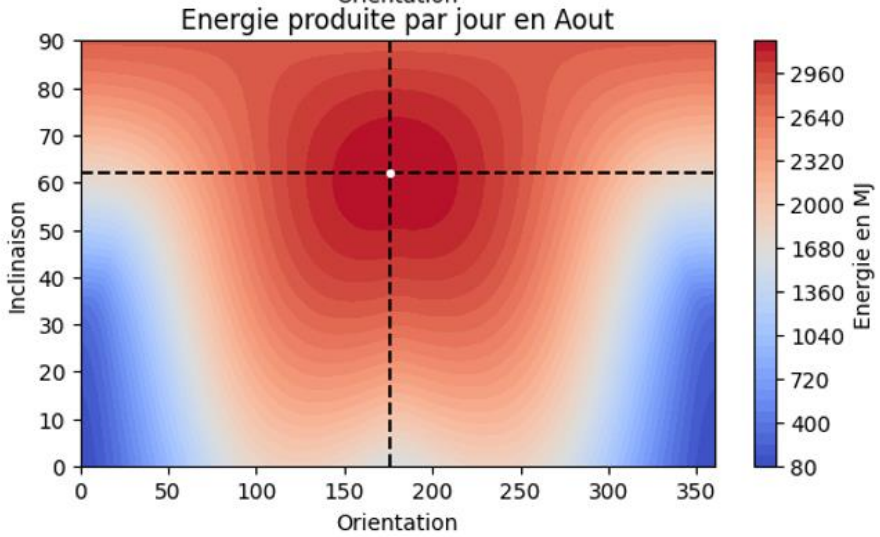

Energie produite par jour en Mars
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
0
360
720
1080
1440
1800
2160
2520
2880

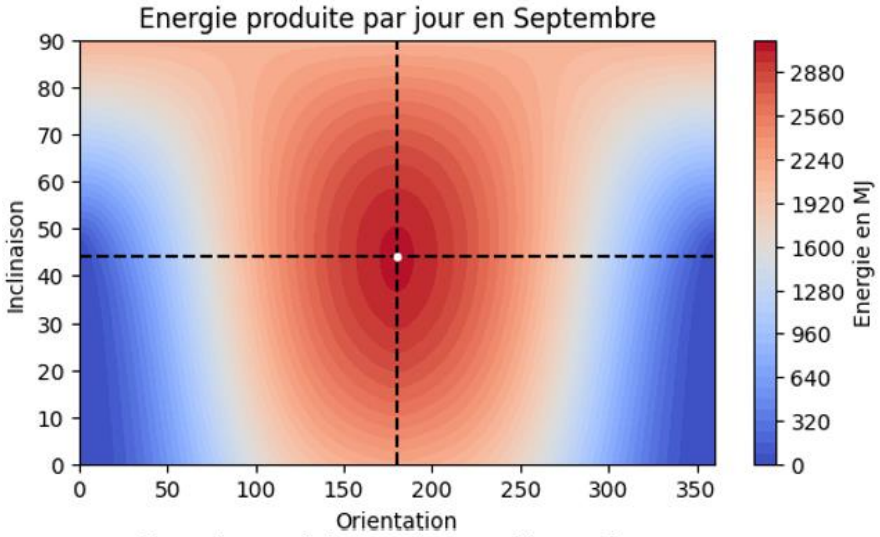

Energie produite par jour en Avril
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
80
400
720
1040
1360
1680
2000
2320
2640
2960

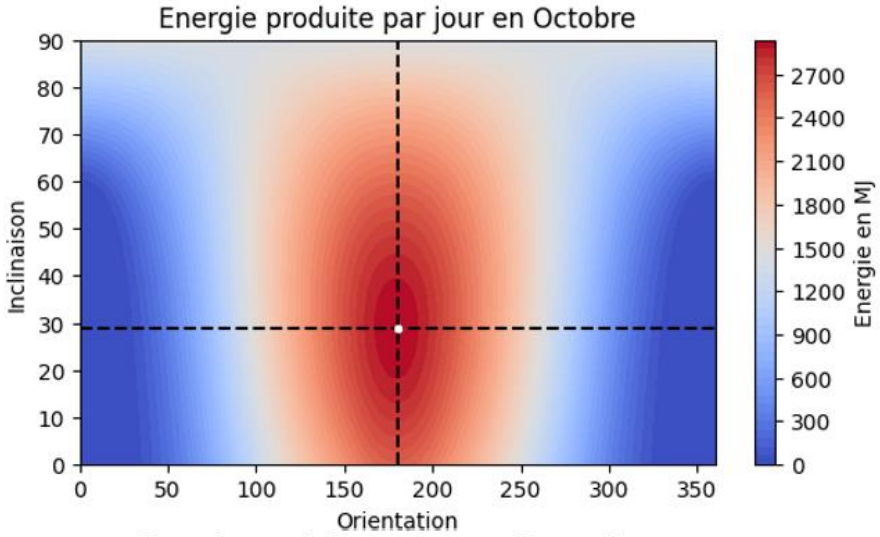

Energie produite par jour en Mai
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
360
720
1080
1440
1800
2160
2520
2880
3240

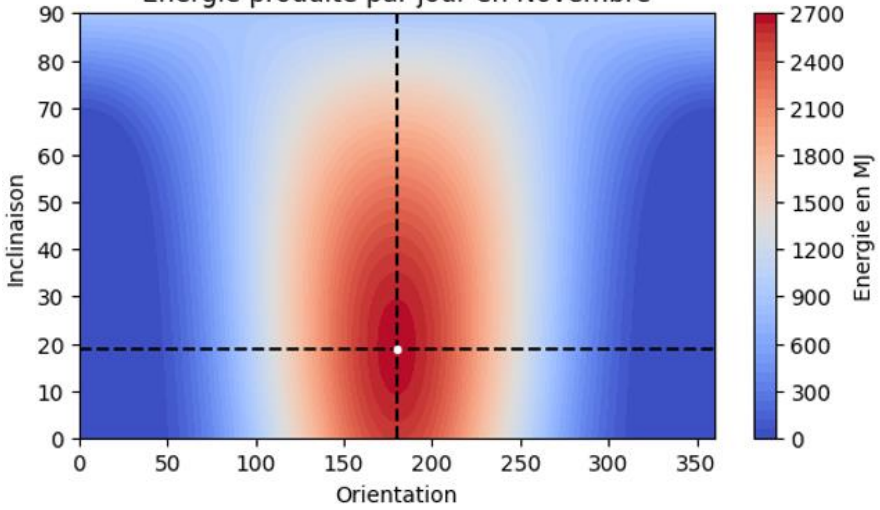

Energie produite par jour en Juin
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
540
900
1260
1620
1980
2340
2700
3060
3420

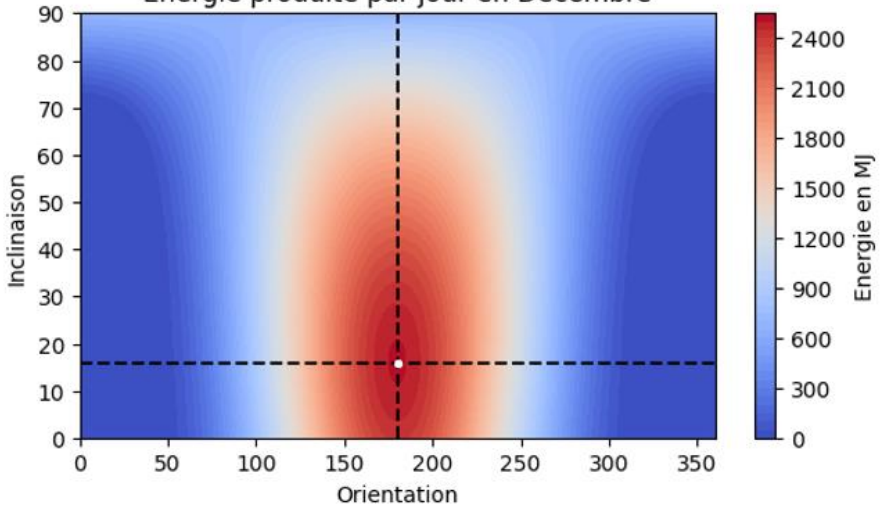

Energie produite par jour en Juillet
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
360
720
1080
1440
1800
2160
2520
2880
3240


Energie produite par jour en Aout
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
80
400
720
1040
1360
1680
2000
2320
2640
2960


Energie produite par jour en Septembre
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
0
320
640
960
1280
1600
1920
2240
2560
2880


Energie produite par jour en Octobre
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
0
300
600
900
1200
1500
1800
2100
2400
2700


Energie produite par jour en Novembre
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
0
300
600
900
1200
1500
1800
2100
2400
2700


Energie produite par jour en Decembre
Inclinaison
Orientation
Energie en MJ
0
10
20
30
40
50
60
70
80
90
0
50
100
150
200
250
300
350
0
300
600
900
1200
1500
1800
2100
2400

Inclinaisons :

Janvier :	19°	Juillet :	78°
Février :	29°	Aout :	62°
Mars :	43°	Septembre :	44°
Avril :	62°	Octobre :	29°
Mai :	78°	Novembre :	19°
Juin :	84°	Décembre :	16°

Inclinaison moyenne : 47°

Inclinaison optimale par rapport à la normale : 43°

Orientations :

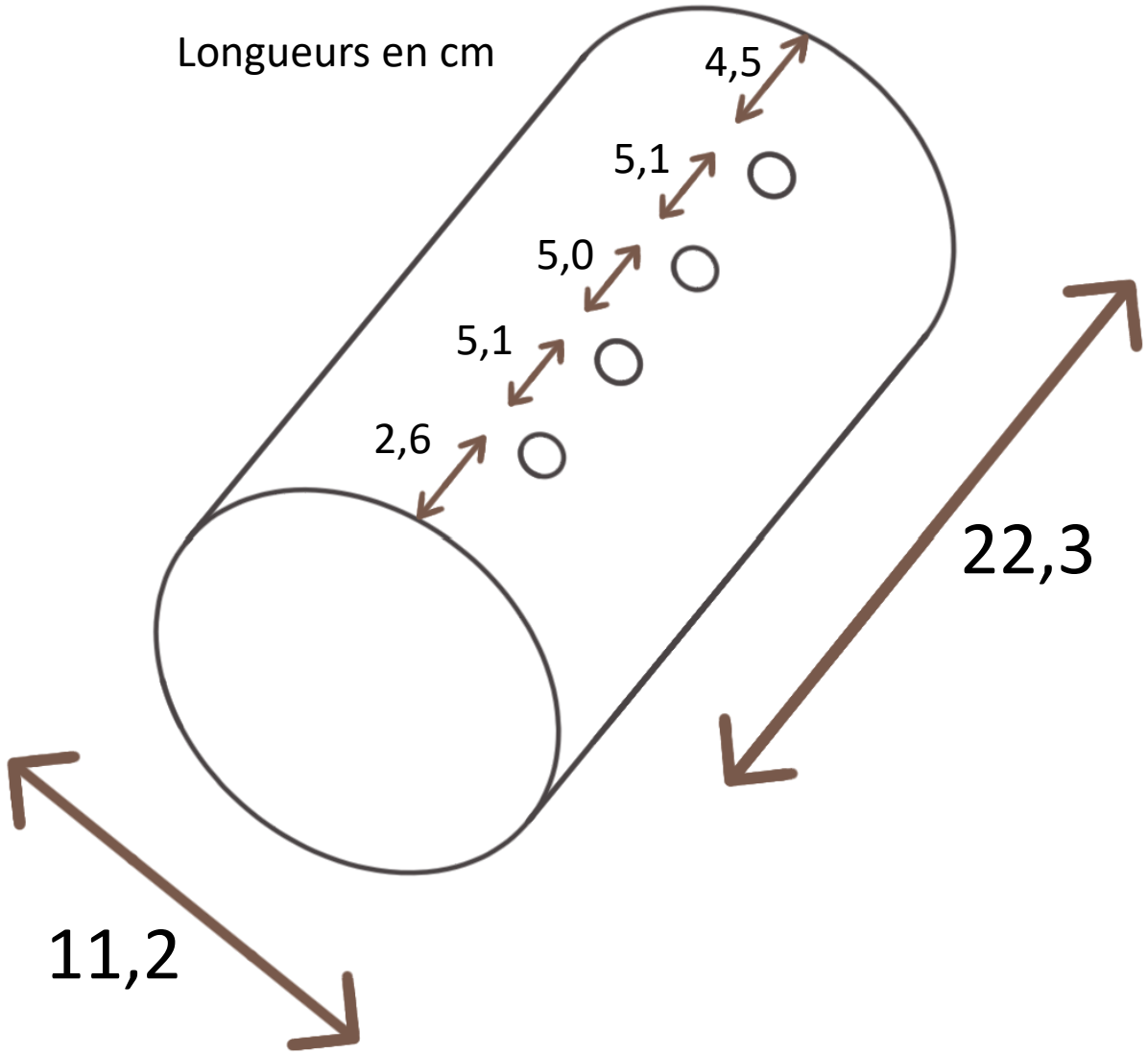
Janvier :	180°	Juillet :	177°
Février :	181°	Aout :	176°
Mars :	180°	Septembre :	180°
Avril :	183°	Octobre :	180°
Mai :	181°	Novembre :	180°
Juin :	178°	Décembre :	180°

Orientation optimale : 180° (Plein sud)

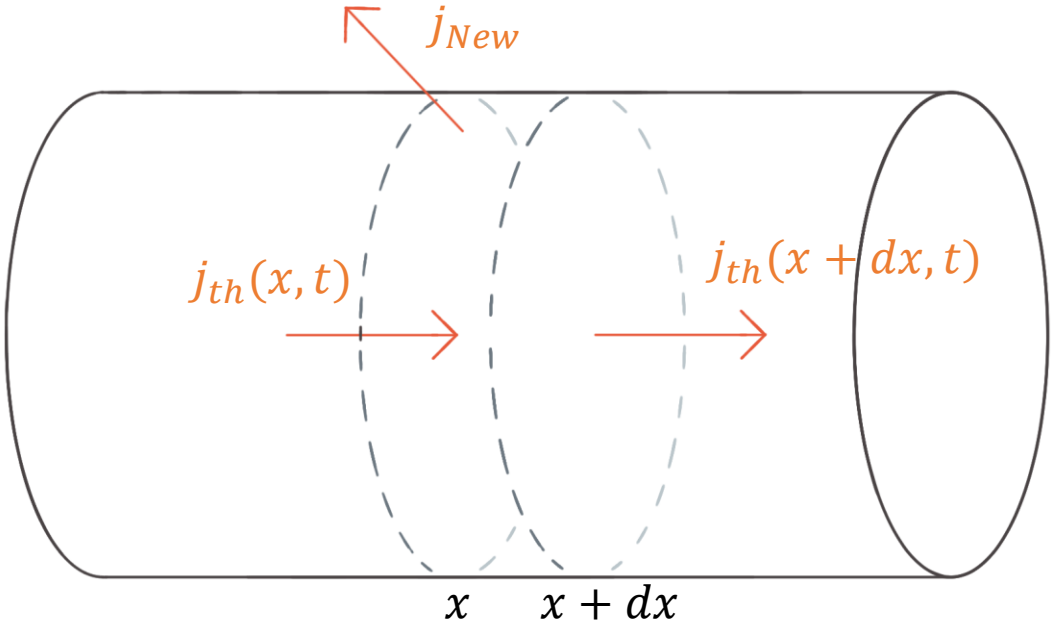
Bloc de béton

Composition pour 10 L :

- Ciment : 3 kg
- Eau : 1,5 L
- Sable : 6,8 kg
- Gravillons : 11,5 kg



Bilan d'énergie thermique dans le bloc



$$dU = \delta Q$$

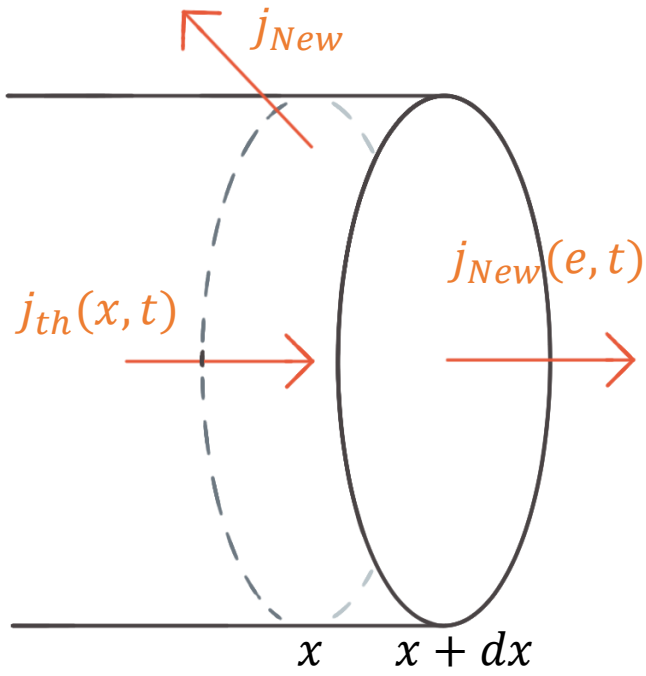
$$C\ dT = (\phi_{th}(x,t) - \phi_{th}(x+dx,t) + \phi_{new}(lat,t))dt$$

$$\rho\ c\ S\ dTdx = -\frac{\partial j_{th}}{\partial x}\ S\ dx\ dt + j_{new}(lat,t)\ dt\ S$$

$$\rho\ c\ dTdx\ \pi\ r^2 = -\lambda\ \frac{\partial^2 T}{\partial x^2}\pi\ r^2 dx\ dt + h_{cc}(T_{ext} - T)\ dt\ dx\ 2\pi\ r$$

$$\rho\ c\ \frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2} + h_{cc}(T_{ext} - T)\ \frac{2}{r}$$

Bilan d'énergie thermique à l'extrémité du bloc



$$dU = \delta Q$$

$$C\ dT = (\phi_{th}(x,t) - \phi_{new}(e,t) + \phi_{new}(lat,t))dt$$

$$\rho\ c\ S\ dTdx = j_{th}(x,t)dt\ S - j_{new}(e,t)\ dt\ S + j_{new}(lat,t)$$

$$\rho\ c\ dTdx\ \pi\ r^2 = -\lambda\ \frac{\partial T}{\partial x}\pi\ r^2 dt + h_{cc}(T_{ext} - T)\ dt\ \pi\ r^2 + h_{cc}(T_{ext} - T)\ dt\ dx\ 2\pi\ r$$

$$\rho\ c\ \frac{\partial T}{\partial t} = -\frac{\lambda}{dx}\frac{\partial T}{\partial x} + \frac{h_{cc}(T_{ext} - T)}{dx} + h_{cc}(T_{ext} - T)\ \frac{2}{r}$$

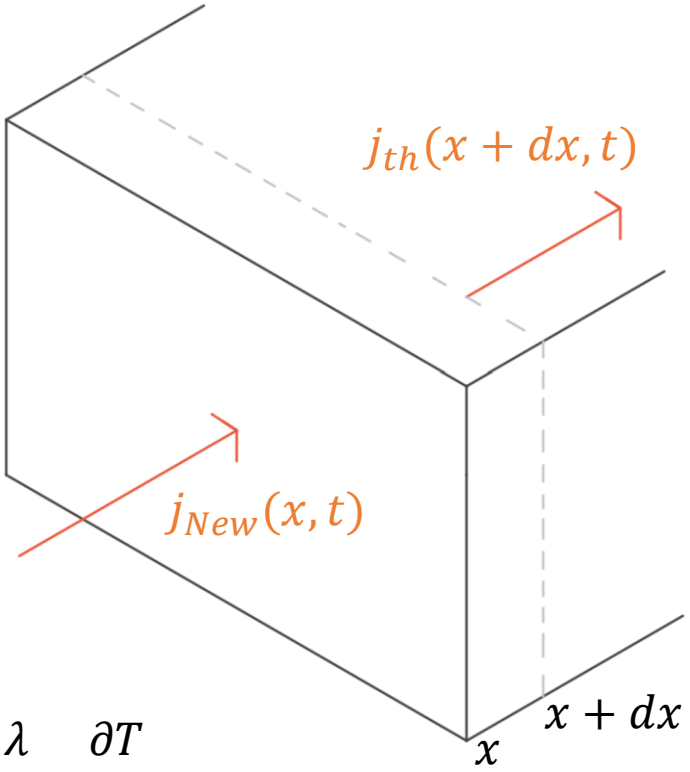
Bilan d'énergie thermique à l'extrémité du mur extérieur

$$dU = \delta Q$$

$$C\ dT = (\phi_{new}(0,t) - \phi_{th}(dx,t))dt$$

$$\rho\ c\ S\ dTdx = j_{new}(0,t)\ dt\ S - j_{th}(dx,t)dt\ S$$

$$\rho\ c\ dTdx = h_{cc}(T_{ext} - T)\ dt + \lambda\ \frac{\partial T}{\partial x}\ dt$$



$$\frac{\partial T}{\partial t} = \frac{h_{cc}(T_{ext} - T)}{\rho\ c\ dx} + \frac{\lambda}{\rho\ c\ dx}\frac{\partial T}{\partial x}$$

$$\frac{T_x^{t+1} - T_x^t}{n_t} = \frac{h_{cc}(T_{ext} - T_x^t)}{\rho\ c\ n_x} + \frac{\lambda}{\rho c}\frac{T_{x+1}^t - T_x^t}{n_x^2}$$

$$T_x^{t+1} = T_x^t\left(1 - \frac{\lambda n_t}{\rho c n_x^2} - \frac{h_{cc} n_t}{\rho c n_x}\right) + T_{x+1}^t\frac{\lambda n_t}{\rho c n_x^2} + T_{ext}\frac{h_{cc} n_t}{\rho c n_x}$$

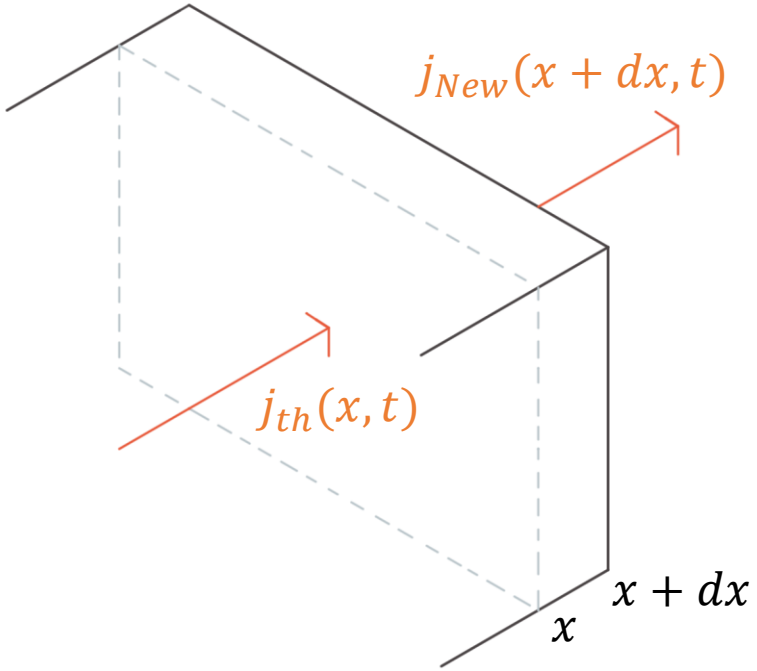
Bilan d'énergie thermique à l'extrémité du mur intérieur

$$dU = \delta Q$$

$$C\,dT = (\phi_{th}(x,t) - \phi_{new}(e,t))dt$$

$$\rho\, c\, S\, dTdx = j_{th}(x,t)\, dt\, S - j_{new}(e,t)\, dt\, S$$

$$\rho\, c\, dTdx = -\lambda \frac{\partial T}{\partial x}\, dt + h_{cc}(T_{int} - T)dt$$

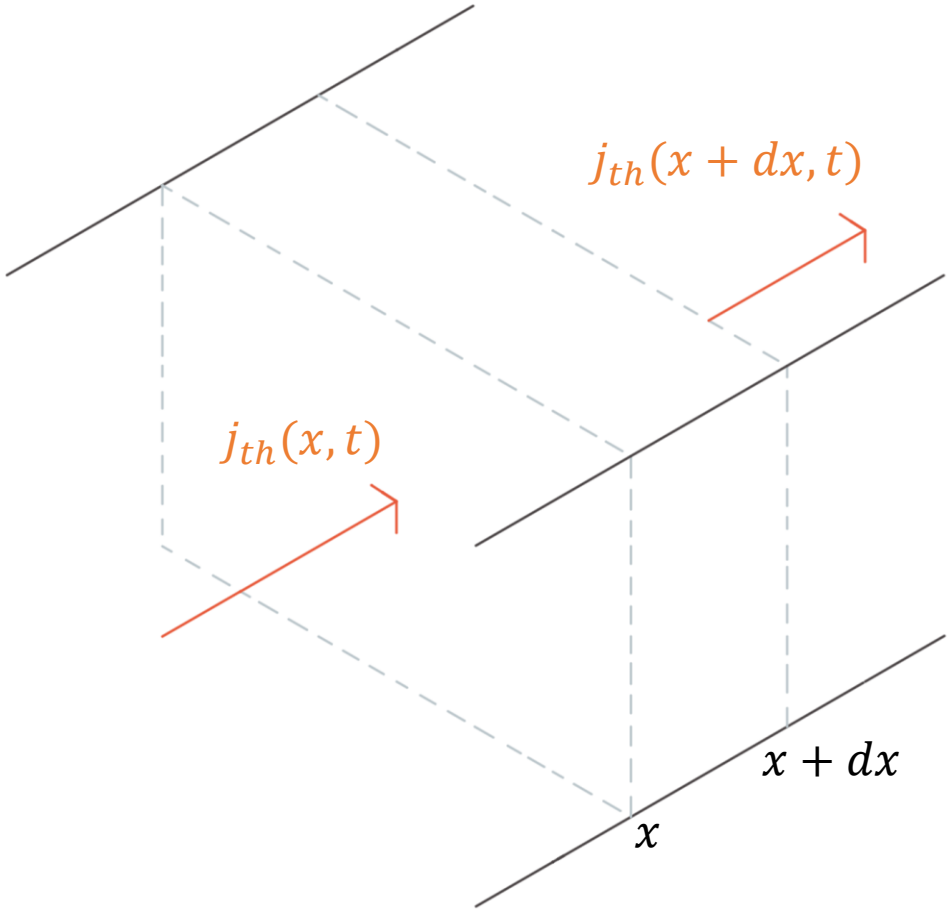


$$\frac{\partial T}{\partial t} = -\frac{\lambda}{\rho\, c\, dx}\frac{\partial T}{\partial x} + \frac{h_{cc}(T_{int} - T)}{\rho\, c\, dx}$$

$$\frac{T_x^{t+1} - T_x^t}{n_t} = \frac{\lambda}{\rho c}\frac{T_{x-1}^t - T_x^t}{n_x^2} + \frac{h_{cc}(T_{int} - T_x^t)}{\rho\, c\, n_x}$$

$$T_x^{t+1} = T_x^t\left(1 - \frac{\lambda n_t}{\rho c n_x^2} - \frac{h_{cc} n_t}{\rho c n_x}\right) + T_{x-1}^t \frac{\lambda n_t}{\rho c n_x^2} + T_{int}\frac{h_{cc} n_t}{\rho c n_x}$$

Bilan d'énergie thermique dans le mur



$$dU = \delta Q$$

$$C\ dT = (\phi_{th}(x,t) - \phi_{th}(x+dx,t))dt$$

$$\rho\ c\ S\ dT dx = -\frac{\partial j_{th}}{\partial x}\ S\ dx\ dt$$

$$\rho\ c\ dT dx\ \pi\ r^2 = -\lambda\ \frac{\partial^2 T}{\partial x^2}\pi\ r^2 dx\ dt$$

$$\rho\ c\ \frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2}$$

Bilan d'énergie thermique de l'intérieur

$$dU = \delta Q$$

$$C\ dT = \phi_{new}(e,t)\ dt$$

$$\rho\ c\ V\ dT = j_{new}(e,t)\ dt\ S$$

$$\rho\ c\ V\ dT = h_{cc}(T_{mur} - T_{int})\ S\ dt$$

$$\frac{dT}{dt} = \frac{h_{cc}S(T_{mur} - T_{int})}{\rho\ c\ V}$$

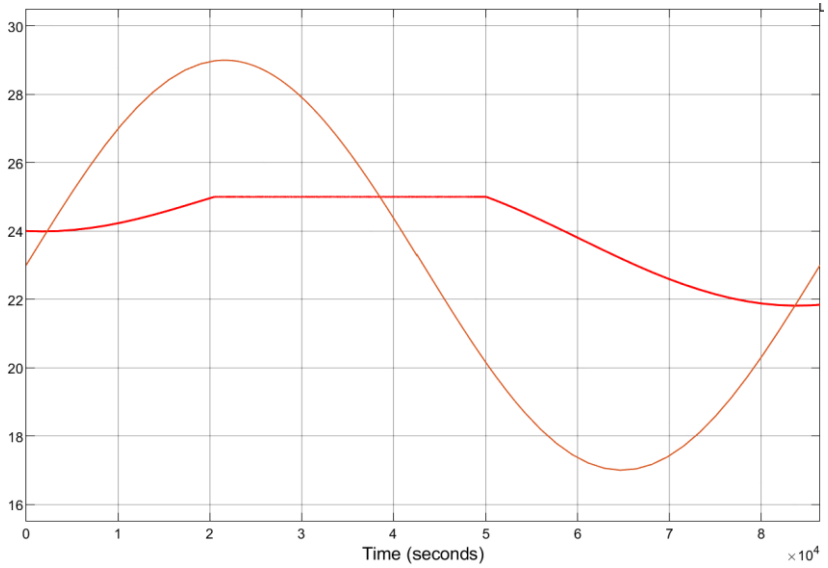
$$\frac{T_{int}^{t+1} - T_{int}^{t}}{n_t} = \frac{h_{cc}S(T_{mur} - T_{int})}{\rho\ c\ V}$$

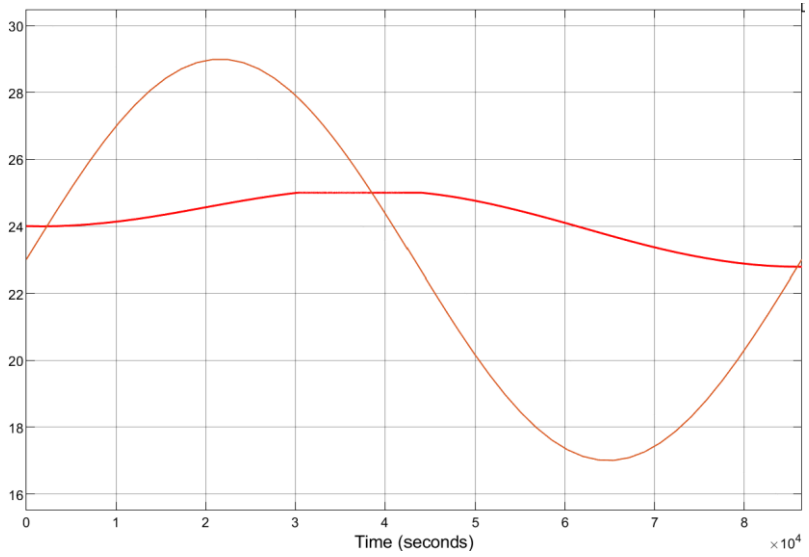
$$T_{int}^{t+1} = T_{int}^{t}\left(1 - \frac{h_{cc}n_t\ S}{\rho cV}\right) + T_{mur}\frac{h_{cc}n_t\ S}{\rho cV}$$

Modélisation Matlab Simulink

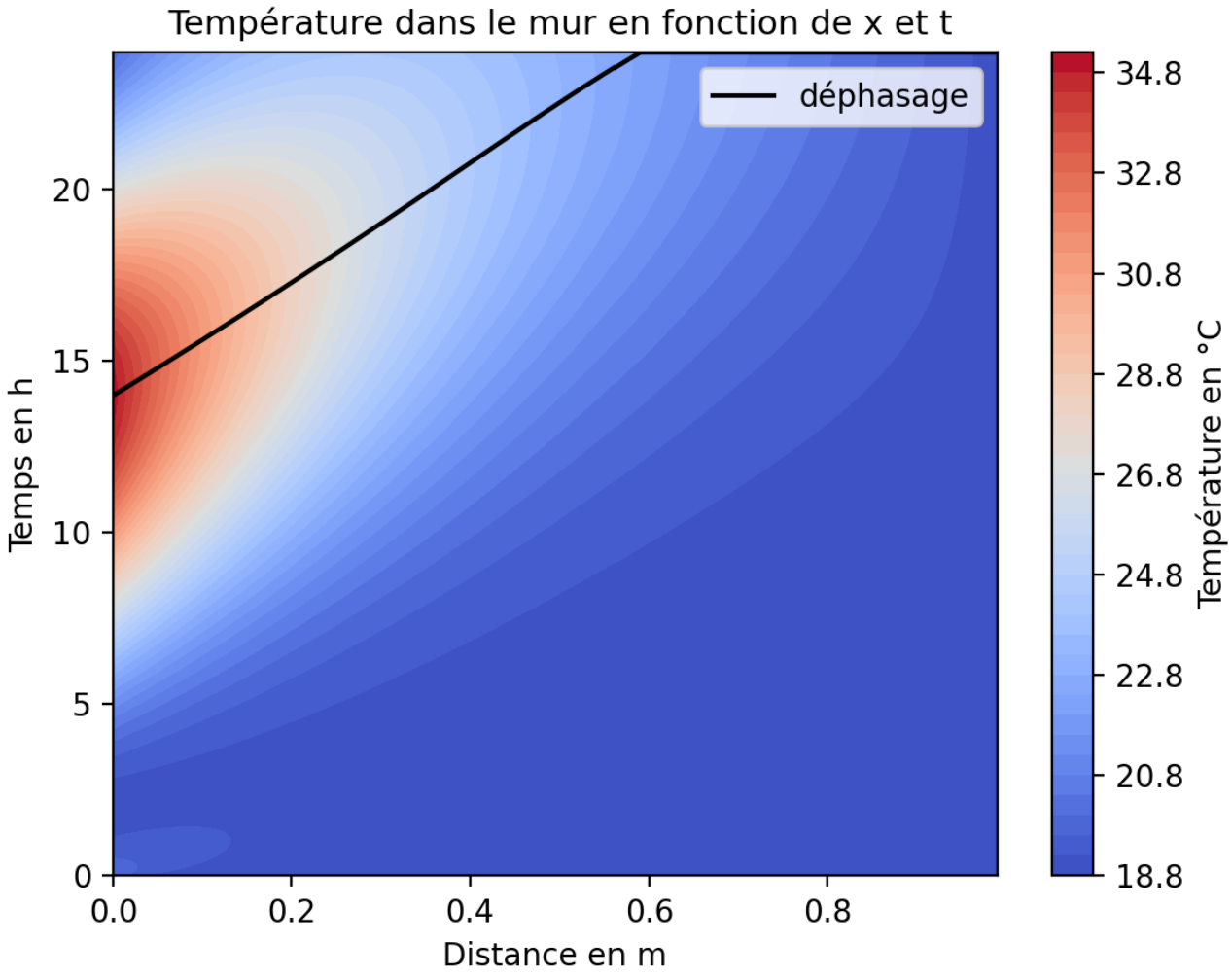
Pièce :

- Longueur : 10m
- Largeur : 8m
- Hauteur : 2,5m
- Epaisseur : 0,5m
- S = 170m²
- Ssol = 60m²

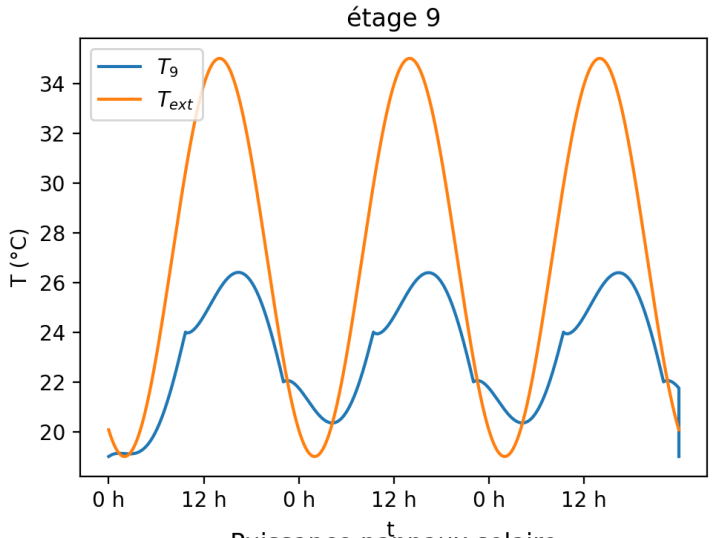


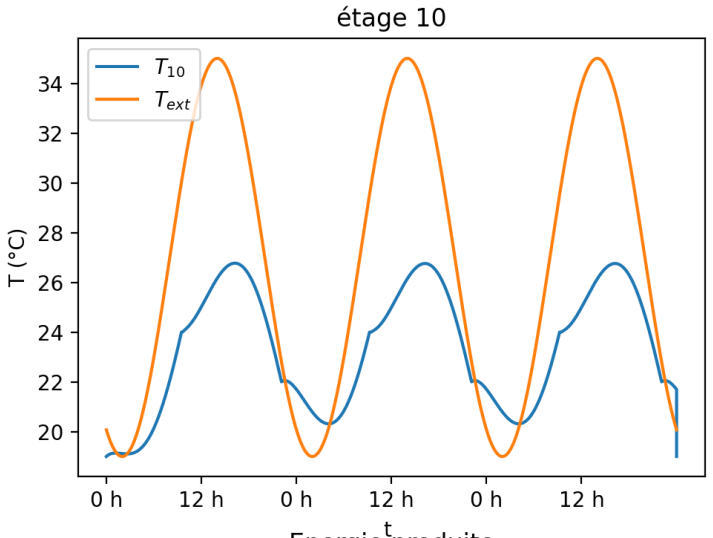


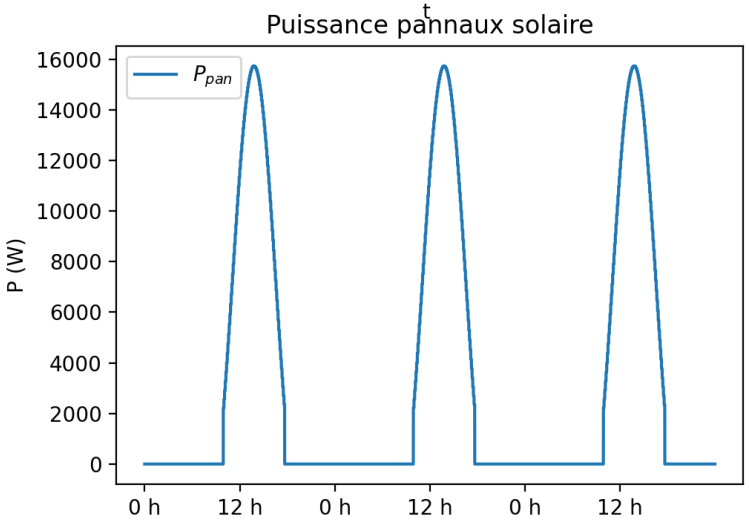
Exemple de déphasage

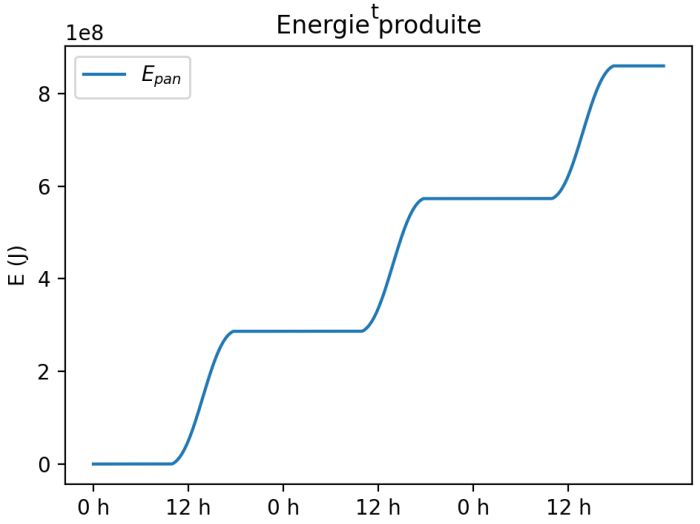


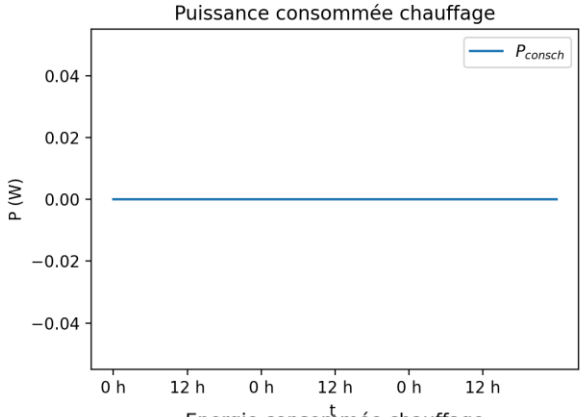
Résultats du programme non utilisé

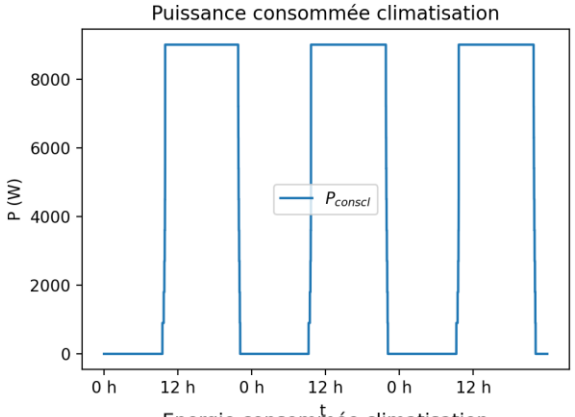


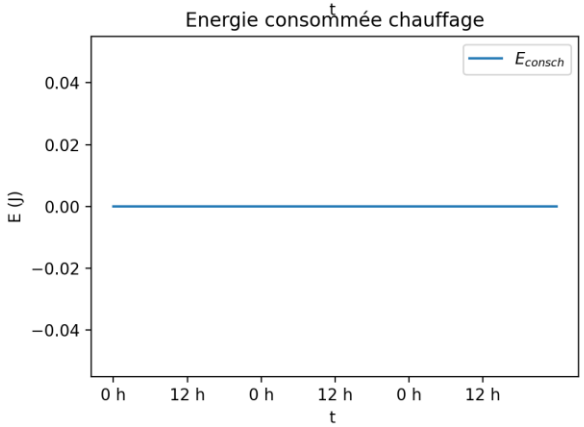


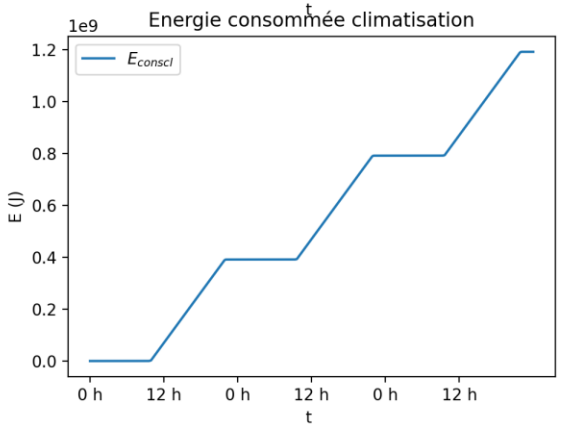


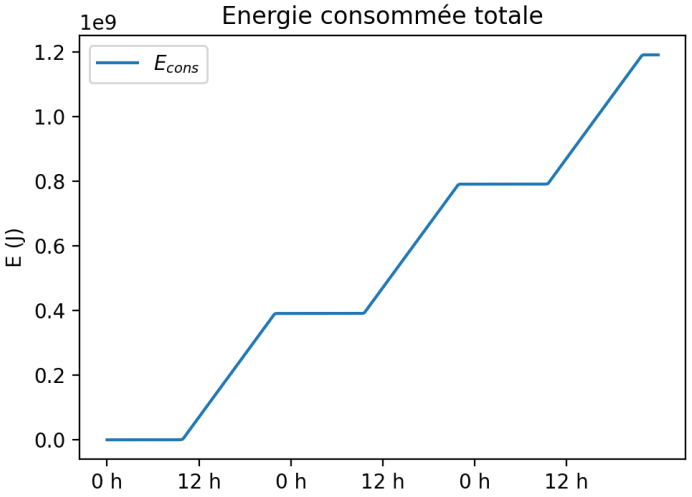


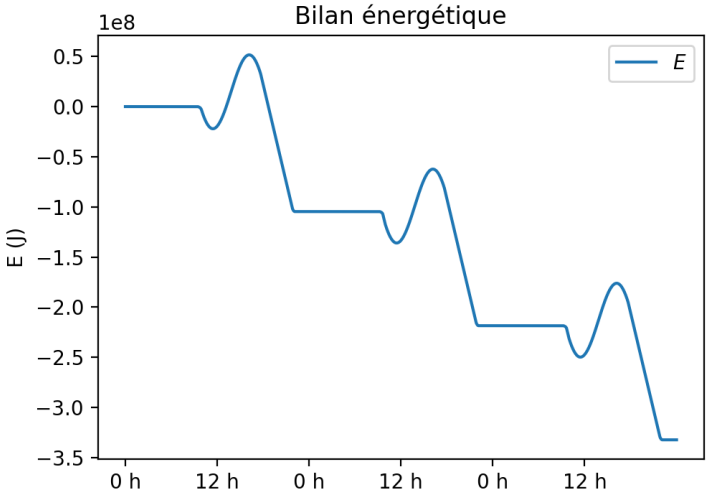

Puissance consommée chauffage
P_{consch}
P (W)
0.04
0.02
0.00
−0.02
−0.04
0 h
12 h
0 h
12 h
0 h
12 h
t


Puissance consommée climatisation
P_{conscl}
P (W)
8000
6000
4000
2000
0
0 h
12 h
0 h
12 h
0 h
12 h
t


Energie consommée chauffage
E_{consch}
E (J)
0.04
0.02
0.00
−0.02
−0.04
0 h
12 h
0 h
12 h
0 h
12 h
t


Energie consommée climatisation
1e9
E_{conscl}
E (J)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 h
12 h
0 h
12 h
0 h
12 h
t


Energie consommée totale
1e9
E_{cons}
E (J)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0 h
12 h
0 h
12 h
0 h
12 h


Bilan énergétique
1e8
E
E (J)
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.5
−3.0
−3.5
0 h
12 h
0 h
12 h
0 h
12 h

Programmes Python

Panneaux solaires

```python
## Importations

import numpy as np
import math as m
from matplotlib import pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D

def progress_bar(progress, total):   # Affiche le pourcentage de calcul
    pourcentage = 100 * progress / float(total)
    barre = '' * int(pourcentage/2) + '-'*int(50 - pourcentage/2)   # Alt 219
#    print(f'|{barre}| {round(pourcentage,3)}%\r' , end='')
    print(f' {round(pourcentage,3)}%\r' , end='')

## Définition des constantes

jour = 86400   # temps d'un jour en seconde
nbjour = 1
temps = [t for t in range(nbjour*jour)]
dt = 1
Span = 50   # Surface passeau solaire m²
Rendpan = 0.15   # Efficacité panneau solaire

## Récupération des données

fichierHiver=open("D:/TIPE 5-2/5minHiver.csv","r")   # Fichier avec temps et positions en degrés de l'hiver

def Recup(nom_fichier):   # Récupère les données temps, azimut, élevation du soleil
    fichier=open(nom_fichier,"r")   # Fichier avec temps et positions en degrés d'été
    tempssol1=[]
    azimut1=[]
    elevation1=[]

    fichier.readline()  # Enlève les 5 premières lignes
    fichier.readline()
    fichier.readline()
    fichier.readline()
    fichier.readline()

    for ligne in fichier :
        ligne=ligne.strip()   # Séparation des valeurs du fichier csv
        A=ligne.split(";")
        B=A[0].split(":")
        if not B == ['']:
            tempssol1.append(int(B[0])*3600+int(B[1])*60+int(B[2])) # en seconde
            elevation1.append(float(A[1]))
            azimut1.append(float(A[2]))

    fichier.close()
    return tempssol1,azimut1,elevation1   # Valeurs par rapport au soleil

```

```
## Configuration

def Config(nom_fichier):   # Définit les valeurs par rapport au temps

    tempssol1,azimut1,elevation1 = Recup(nom_fichier)
    tempssol,azimut,elevation = [],[],[]
    t = 0
    indice = 0

    while t <= len(temps):
        if t >= tempssol1[-1] or t < tempssol1[0]:   # Tant qu'il n'y a pas de soleil
            elevation.append(0)
            azimut.append(0)
        else:
            if t == tempssol1[indice]:
                indice += 1
            elevation.append(elevation1[indice])
            azimut.append(azimut1[indice])
        tempssol.append(t)
        t += 1
    return tempssol,azimut,elevation

## Résolution

def produit_scalaire(u, v):
    ux, uy, uz = u
    vx, vy, vz = v
    return ux*vx + uy*vy + uz*vz

def sph_cart(r, phi, theta):   # Passage des coordonnées en cartésien : er = ..ex + ..ey + ..ez
    return (r*np.sin(theta)*np.cos(phi), r*np.sin(theta)*np.sin(phi), r*np.cos(theta))

def Phi(t,ori,inc,azimut,elevation):  # Puissance surfacique recue par le panneau
    if t>=2*86400 :   # Pour rester sur une journée
        t=t-2*86400
    if t>=86400 :
        t=t-86400

    azi=azimut[t]*2*np.pi/360   # En radian
    ele=elevation[t]*2*np.pi/360   # En radian
    phimax = 1000   # Phi maximal (Plein soleil) en W/m²
    upan = sph_cart(1, ori, np.pi/2-inc)   # Vecteur normal au panneau
    usol = sph_cart(1, azi, np.pi/2-ele)   # Vecteur dirigé par les rayons du soleil

    ps = produit_scalaire(upan, usol)

    if ele <= 5*2*np.pi/360:    # Si l'élevation du soleil est inférieure à 5 degrés
        return 0
    else:
        if ps <= 0:
            return 0
        else:
            return phimax*ps
```

```python

def maxEpan(ori,inc,azimut,elevation):   # Renvoie l'énergie totale recue

    Ppan=[]   # Puissance générée par le panneau
    Epan=[]   # Energie générée par le panneau

    res = 5*60   # resolution de 5 min en seconde
    for k in range(0,nbjour*jour, res):   #Parcourt journée avec pas de 5min
        Ppan.append( Rendpan*Span*Phi(k,ori,inc,azimut,elevation) )
        if k-res < 0:
            Epan.append(Ppan[-1]*res)
        else:
            Epan.append(Epan[-1] + Ppan[-1]*res)

    return Epan[-1]   # Energie finale

## Récupérer les orientations et inclinaions optimales

ori_debut = 0   #En degres
ori_fin = 360   #En degres
ori_resolution = 361   #nb points

inc_debut = 0   #En degres
inc_fin = 90   #En degres
inc_resolution = 91   #nb points

nom_fichier = "C:/Paul/Prepa/TIPE/Epan(ori, inc)"   # Fichier sur lequel on écrit les données

def Calcul_surface(mois):   # Ecrit les valeurs dans les fichiers
    _,azimut,elevation = Config("D:/TIPE 5-2/"+mois+".csv")   # _ car on ne prend pas le temps

    Lori = np.linspace(ori_debut,ori_fin,ori_resolution)*2*np.pi/360   # En radian
    Linc = np.linspace(inc_debut,inc_fin,inc_resolution)*2*np.pi/360   # En radian
    LincM, LoriM, LEpanM = [0]*len(Lori)*len(Linc), [0]*len(Linc)*len(Lori), [0]*len(Lori)*len(Linc)

    progress_bar(0,len(Lori)*len(Linc))   # Pourcentage de calcul

    for i,ori in enumerate(Lori):
        for k,inc in enumerate(Linc):

            progress_bar(k+i*len(Linc),len(Lori)*len(Linc))

            m = maxEpan(ori,inc,azimut,elevation)

            LEpanM[k + i*len(Linc)] = m
            LoriM[k + i*len(Linc)] = ori
            LincM[k + i*len(Linc)] = inc

    with open(nom_fichier + mois + ".txt", "w") as memoire:   # Sauvegarde et écrit dans le fichier
        for k in range(len(LEpanM)):
            memoire.write(f"{LEpanM[k]};{LoriM[k]};{LincM[k]}\n")

```

```
## Tracer les graphiques

def Tracer_Surface():   # Trace les graphes en 2D

    X = np.linspace(ori_debut,ori_fin,ori_resolution)
    Y = np.linspace(inc_debut,inc_fin,inc_resolution)
    X, Y = np.meshgrid(Y, X)

    Lmois = ['Janvier','Fevrier','Mars','Avril','Mai','Juin','Juillet','Aout','Septembre','Octobre','Novembre','Decembre']
    fig, ((ax1, ax2, ax3, ax4, ax5, ax6), (ax7, ax8, ax9, ax10, ax11, ax12)) = plt.subplots(2, 6, figsize =(12,9))
    Laxe = [ax1,ax2,ax3,ax4,ax5,ax6,ax7,ax8,ax9,ax10,ax11,ax12]

    def RecupDeterm(mois):
        Z = []
        with open(nom_fichier + mois + ".txt", "r") as Donnees:
            m=0
            oriM, incM = 0, 0
            for ligne in Donnees:
                ligne = ligne.strip().split(";")
                Z.append(float(ligne[0])/1e6)
                if Z[-1] >= m:   #Récupère la config idéale
                    m = Z[-1]
                    oriM = float(ligne[1])
                    incM = float(ligne[2])
        return Z,oriM,incM,m

    for i,mois in enumerate(Lmois):
        Z,oriM,incM,m = RecupDeterm(mois)

        print(f'Config opti {mois} Orientation : {oriM*180/np.pi} Inclinaison : {incM*180/np.pi}')

        Z = np.array(Z).reshape(Y.shape)

        if mois == 'Janvier' or mois == 'Juillet':
            Laxe[i].set_ylabel ('Inclinaison')

        if i >= 6:
            Laxe[i].set_xlabel('Orientation')

        Laxe[i].set_title(mois)
        Laxe[i].contourf(Y, X, Z, 50,  cmap = cm.coolwarm)
        Laxe[i].axvline(x = oriM*180/np.pi, color="k", ls = "--")  # Barre verticale
        Laxe[i].axhline(y =incM*180/np.pi, color="k", ls = "--")  # Barre horizontale
        Laxe[i].plot(oriM*180/np.pi, incM*180/np.pi, "w.",markersize=5)

        if mois == 'Juin' or mois == 'Decembre':
            surf=Laxe[i].contourf(Y, X, Z, 50,  cmap = cm.coolwarm)
            fig.colorbar(surf,label = "Energie en MJ")

    plt.tight_layout()
    plt.show()
```

Simulation béton

```
## Importations

import numpy as np
import matplotlib.pyplot as plt

##Données numériques

L = 0.223          # Longueur du bloc en m
r = 0.056          # Rayon du cylindre en m
g = 2401           # Masse volumique du béton en kg/m3
lam = 4            # conductivité thermique du béton en W/K/m
c = 1100           # capacité thermique massique en J/K/kg
h = 10             # Coefficient de conducto-convection en W/m²/K
temps = 3600*6.5   # Duree totale evolution en secondes
nx = 100           # Nombre de troncons
nt = 100000        # Nombre intervalles de temps
Deltax = L/nx      # Longueur tronçon
Deltat = temps/nt # Intervalle de temps
Text = 21          # Température initiale du bloc (Température extérieure)
Textr = 190        # Température de l'extremité

x = np.linspace(0.0,L,nx)
position1 = int(2.6*nx/22.3)
position2 = int(7.7*nx/22.3)
position3 = int(12.7*nx/22.3)
position4 = int(17.8*nx/22.3)

T = [Textr]+(nx-1)*[Text] # Initialisation des températures
Tx1 = []
Tx2 = []
Tx3 = []
Tx4 = []

# Construction du tableau vierge des accroissements de température
accroissT = np.zeros(nx)

## Résolution

for n in range(nt): # boucle évolution du temps

    for m in range(1,nx-1): # boucle de calcul de l'accroissement de température pour chaque abscisse
        accroissT[m] = (T[m-1]+T[m+1]-2*T[m])*lam*Deltat/(g*c*Deltax**2)+(Text-T[m])*2*h*Deltat/(g*c*r)

    for m in range(1,nx-1): # boucle de calcul de T à l'instant suivant
        T[m] += accroissT[m]

    T[-1] = T[-1] + (T[nx-2]-T[-1])*lam*Deltat/(g*c*Deltax**2)+(Text-T[-1])*2*h*Deltat/(g*c*r)+ (Text-
T[-1])*h*Deltat/(g*c*Deltax)
    Tx1.append(T[position1])
    Tx2.append(T[position2])
    Tx3.append(T[position3])
    Tx4.append(T[position4])

nom_fichier = 'D:/TIPE 5-2/BetonTPython'
```

```
with open(nom_fichier+'.csv','w') as f:
    f.write('temps en s;T1;T2;T3;T4\n')
    for i in range(nt):
        f.write(f'{i*Deltat};{Tx1[i]};{Tx2[i]};{Tx3[i]};{Tx4[i]}\n')

## Températures réelles

tempsr = []
T1r = []
T2r = []
T3r = []
T4r = []

fichier = 'D:/TIPE 5-2/BetonTreel.csv'
with open(fichier,'r') as f:
    f.readline()
    for ligne in f:
        ligne = ligne.replace(',','.').split(';')
        tempsr.append(float(ligne[0])*3600)
        if float(ligne[1]) < 0 :     # Pour ajuster l'erreur du capteur
            T1r.append(float(ligne[1])+175.7)
        else :
            T1r.append(float(ligne[1]))
        T2r.append(float(ligne[2]))
        T3r.append(float(ligne[3]))
        T4r.append(float(ligne[4]))

## Affichage

print('Valeur de T1 au bout de', temps/3600,'heures :', round(Tx1[-1],2),'°C')
print('Valeur de T2 au bout de', temps/3600,'heures :', round(Tx2[-1],2),'°C')
print('Valeur de T3 au bout de', temps/3600,'heures :', round(Tx3[-1],2),'°C')
print('Valeur de T4 au bout de', temps/3600,'heures :', round(Tx4[-1],2),'°C')

plt.plot(np.linspace(0, temps, nt), Tx1,'r', label = "T1 sim")
plt.plot(np.linspace(0, temps, nt), Tx2,'r', label = 'T2 sim')
plt.plot(np.linspace(0, temps, nt), Tx3,'r', label = 'T3 sim')
plt.plot(np.linspace(0, temps, nt), Tx4,'r', label = 'T4 sim')

plt.plot(tempsr, T1r,'b', label = 'T1 réel')
plt.plot(tempsr, T2r,'b', label = 'T2 réel')
plt.plot(tempsr, T3r,'b', label = 'T3 réel')
plt.plot(tempsr, T4r,'b', label = 'T4 réel')

plt.grid()
plt.xlabel(r'$t(s)$', fontsize=12)
plt.ylabel(r'$T(°C)$', fontsize=12,rotation=0)
plt.title(u" Evolution de la température des positions trouées du bloc de béton", size =11)
plt.legend()
plt.show()

```

Essai simulation pièce

```
## Importations

import math as m
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

## Définition des constantes

h = 2 # hauteur
l = 8 # largeur
L = 8 # longueur
S = h*(2*l+2*L) # surface de contact mur-extérieur
e = 2 # épaisseur mur

Vbet = S*e # volume de mur
Vair = h*l*L # volume d'air piece

cbet = 900 # capacité thermique massique béton en J/K/kg
cair = 1100 # capacité thermique massique air en J/K/kg

mvbet = 2400 # masse volumique béton en kg/m3
mvair = 1.292 # masse volumique air en kg/m3

lambet = 1 # conductivité thermique béton W/K/m

hextver = 16.66 #Coefficient vertical extérieur en W/m²/K
hexthor = 20 #Coefficient horizontal extérieur en W/m²/K
hintver = 9.1 #Coefficient vertical intérieur en W/m²/K
hinthordesc = 5.88 #Coefficient horizontal intérieur (plus chaud en haut qu'en bas) en W/m²/K
hinthormont = 11.11 #Coefficient horizontal intérieur (plus chaud en bas qu'en haut) en W/m²/K

Cp = cair*Vair*mvair # capacité thermique de la piece

## Temps

jour = 86400 # temps d'un jour en s
nbjour = 3
temps = [t for t in range(nbjour*jour)]
dt = 1

nt = 1000       # Nombre intervalles de temps
Deltat = len(temps)/nt # Intervalle de temps

## Positions

nx = 100          # Nombre de troncons
Deltax = e/nx     # Longueur tronçon

x = np.linspace(0.0,e,nx)

```

```
## Extérieur

def Text(t):
    Textmax = 35
    Textmin = 19
    return Textmin+(Textmax-Textmin)/2+(Textmax-Textmin)/2*m.cos(2*m.pi/jour*(t-jour*14/24))

## Températures

T = []
for _ in range(nx+2): # Parcours toutes les positions plus int et ext
    T.append([])
    for i in range(nbjour*jour): # Parcours le temps
        T[-1].append(22)

## Evolution des températures

for t in range(0,nbjour*jour-1):

    T[0][t] = Text(t)

    T[1][t+1] = T[1][t]*(1-lambet*Deltat/(mvbet*cbet*Deltax**2)-hextver*Deltat/(mvbet*cbet*Deltax)) + T[2]
[t]*lambet*Deltat/(mvbet*cbet*Deltax**2) + T[0][t]*hextver*Deltat/(mvbet*cbet*Deltax)

    for i in range(2,nx):
        T[i][t+1] = T[i][t] + (T[i+1][t]-2*T[i][t]+T[i-1][t])*lambet*Deltat/(mvbet*cbet*Deltax**2)

    T[-2][t+1] = T[-2][t]*(1-lambet*Deltat/(mvbet*cbet*Deltax**2)-hextver*Deltat/(mvbet*cbet*Deltax)) + T[-3]
[t]*lambet*Deltat/(mvbet*cbet*Deltax**2) + T[-1][t]*hextver*Deltat/(mvbet*cbet*Deltax)

    T[-1][t+1] = T[-1][t+1]*(1-hintver*Deltat*S/(mvair*cair*Vair)) + T[-2][t+1]*hintver*Deltat*S/(mvair*cair*Vair)

## Graphique

plt.plot(temps,T[-1],label='$T_{int}$')
plt.plot(temps,T[0],label='$T_{ext}$')
plt.xlabel('t ')
plt.ylabel ('T (°C)')
plt.legend()

plt.show()
```

Programme non utilisé prêt à être utiliser pour bilan

```
import math as m
from matplotlib import pyplot as plt
import numpy as np

## Dimentions batiment

Et = 10 # nb etages
S = 500 # surface sol par étage
H = 33.5 # hauteur toit
h = 3.35 # hauteur etage
V = S*h # Volume air étage
L = 100 # Longueur batiment
l = 50 # largeur batiment
Sl = h*l # Surface largeur
SL = h*L # Surface longueur
SL2 = SL*2
Sl2 = Sl*2
Stote = Sl2 + SL2 # Surface extérieur étage

## Temps

jour = 86400 # temps d'un jour en s
nbjour = 3
temps = [t for t in range(nbjour*jour)]
dt = 1

## Panneaux solaires

Span = 500 # Surface passeau solaire m²
Rendpan = 0.15 # Efficacité panneau solaire
alb = 0.8 # Albedo
phimax = 1000 # Aport du soleil maximal W/m²

## Températures

LT = []
for _ in range(11):
    LT.append([])
    for i in range(nbjour*jour):
        LT[-1].append(19)

## Extérieur

def Text(t):
    Textmax = 35
    Textmin = 19
    return Textmin+(Textmax-Textmin)/2+(Textmax-Textmin)/2*m.cos(2*m.pi/jour*(t-jour*14/24))

## Fichier
fichier=open("D:/TIPE 5-2/Janvier.csv","r")
```

```
tempssol1=[]
azimut1=[]
elevation1=[]

fichier.readline()
fichier.readline()
fichier.readline()
fichier.readline()
fichier.readline()

for ligne in fichier :
    ligne=ligne.strip()
    A=ligne.split(";")
    B=A[0].split(":")
    tempssol1.append(int(B[0])*3600+int(B[1])*60+int(B[2])) # en seconde
    elevation1.append(float(A[1]))
    azimut1.append(float(A[2]))
fichier.close()

tempssol=[]
azimut=[]
elevation=[]

## Configuration

i = 0
indice = 0

while i <= len(temps):

    if i >= tempssol1[-1] or i < tempssol1[0]:
        elevation.append(0)
        azimut.append(0)
    else:
        if i == tempssol1[indice]:
            indice += 1
        elevation.append(elevation1[indice])
        azimut.append(azimut1[indice])
    tempssol.append(i)
    i += 1

## Soleil

def produit_scalaire(u, v):

    ux, uy, uz = u
    vx, vy, vz = v
    return ux*vx + uy*vy + uz*vz

def sph_cart(r, phi, theta): # er = ..ex + ..ey + ..ez
    return (r*np.sin(theta)*np.cos(phi), r*np.sin(theta)*np.sin(phi), r*np.cos(theta))

```

```
106 def Phi(t,ori,inc):
107 
108     if t>=2*86400 :
109         t=t-2*86400
110     if t>=86400 :
111         t=t-86400
112 
113     azi=azimut[t]*2*np.pi/360
114     ele=elevation[t]*2*np.pi/360
115     phimax = 900 #W/m²
116 
117 
118     upan = sph_cart(1, ori, inc)   #  r, teta, phi
119 
120     usol = sph_cart(1, azi, ele)
121 
122     ps = produit_scalaire(upan, usol)
123 
124 
125 
126     if ele <= 5*2*np.pi/360:    #Avant Lever Soleil
127         return 0
128     else:
129         if ps <= 0:
130             return 0
131         else:
132             return phimax*ps
133 
134 alb = 0.60 # albedo
135 eff = 0.000
136 
137 ## Dimention thermiques
138 
139 # Capacités themiques volumiques en J/m3/K
140 
141 Cair = 1256 # air
142 Cpoly = 21000 # polystyrène
143 Cbet = 2500000 # béton
144 Cldv = 99000 # laine de verre
145 
146 # Capacités thermques en J/K
147 
148 LCapa = [Cair*V]*11 # on pren pa le C0
149 
150 # conductivité thermique en W/m/K
```

```python
cbet = 0.9 # béton
ccar = 1.3 # carrelage
cpoly = 0.036 # polystyrène
cpar = 0.25 # parquet
ctui = 1.2 # tuiles
cpla = 0.35 # plâtre
cldv = 0.035 # laine de verre
cver = 1 # verre
cgaz = 0.0177 # argon

# Conducto-Convection

hextver = 16.66 #Coefficient vertical extérieur en W/m²/K
hexthor = 20 #Coefficient horizontal extérieur en W/m²/K

hintver = 9.1 #Coefficient vertical intérieur en W/m²/K
hinthordesc = 5.88 #Coefficient horizontal intérieur (plus chaud en haut qu'en bas) en W/m²/K
hinthormont = 11.11 #Coefficient horizontal intérieur (plus chaud en bas qu'en haut) en W/m²/K

# Epaisseur des murs extérieurs en m

eext = 0.2 # béton
eextisol = 0.3 # polystyrène
eextint = 0.05 # plâtre

# Epaisseur d'une fenêtre
efen = 0.005 # verre
egaz = 0.016 # argon

# Epaisseur entre les étages en m

eetage = 0.2 # béton
eetageisol = 0.02 # paquet

# Résistances thermiques

def Rcond(e,S,c) :
    return e/(S*c)

def Rconv(h,S) :
    return 1/(h*S)

def Rtotcond(Rfentot,Rcond,) :
    return  (Rfentot*Rcond)/(Rfentot*Rcond)

LR = [2*Rconv(hinthordesc,S) + Rcond(eetage,S,cbet) + Rcond(eetageisol,S,cpar)]*11
# LR[k] resistance entre k et k + 1
```

```
LRe = [Rconv(hextver,Stote) + Rconv(hintver,Stote) + Rcond(eext,Stote,cbet) + Rcond(eextisol,Stote,cpoly)]*11
LRe[10] = LRe[10] * (Rconv(hexthor,S) + Rconv(hinthordesc,S) + Rcond(eext,S,cbet) + Rcond(eextisol,S,cpoly)) /
(LRe[10] + Rconv(hexthor,S) + Rconv(hinthordesc,S) + Rcond(eext,S,cbet) + Rcond(eextisol,S,cpoly))

# Conductance thermique en W/K

LG = [1/LR[i] for i in range(11)]

LGe = [1/LRe[i] for i in range(11)]

## Evolution de température

# Puissances en W

Ph = 80 #P Puissance humain
Pch = 900 # puissance chauffage
Pcl = -900 # Puissance climatisation
Pconsch=[0]*nbjour*jour
Pconscl=[0]*nbjour*jour

Lchauff = []
Lclim = []
for _ in range(11):
    Lchauff.append(False)
    Lclim.append(False)

for t in range(1,nbjour*jour-1):

# ETAGE 1

    if LT[1][t-1] > 20:
        Lchauff[1] = False
    if LT[1][t-1] < 22 :
        Lclim[1] = False

    if LT[1][t-1] < 18 or Lchauff[1]  :
        Lchauff[1] = True
        LT[1][t] = LT[1][t-1] + (1/LCapa[1]) * (-LGe[1] * (LT[1][t-1]-Text(t-1)) - LG[1] * (LT[1][t-1]-LT[2]
[t-1]) + Pch) * dt
        Pconsch[t] = Pconsch[t] + Pch * dt

    if LT[1][t-1] > 24 or Lclim[1] :
        Lclim[1] = True
        LT[1][t] = LT[1][t-1] + (1/LCapa[1]) * (-LGe[1] * (LT[1][t-1]-Text(t-1)) - LG[1] * (LT[1][t-1]-LT[2]
[t-1]) + Pcl) * dt
        Pconscl[t] = Pconscl[t] - Pcl * dt

    if not Lchauff[1] and not Lclim[1]:
        LT[1][t] = LT[1][t-1] + (1/LCapa[1]) * (-LGe[1] * (LT[1][t-1]-Text(t-1)) - LG[1] * (LT[1][t-1]-LT[2]
[t-1])) * dt

```

```
# Etage 2-9

    for i in range(2,10):

        if LT[i][t-1] > 20:
            Lchauff[i] = False
        if LT[i][t-1] < 22 :
            Lclim[i] = False

        if LT[i][t-1] < 18 or Lchauff[i]  :
            Lchauff[i] = True
            LT[i][t] = LT[i][t-1] + (1/LCapa[i])*(-LGe[i]*(LT[i][t-1]-Text(t-1))-LG[i]*(LT[i][t-1]-LT[i-1][t-1])-
LG[i]*(LT[i][t-1]-LT[i+1][t-1]) + Pch)*dt
            Pconsch[t] = Pconsch[t] + Pch * dt

        if LT[i][t-1] > 24 or Lclim[i] :
            Lclim[i] = True
            LT[i][t] = LT[i][t-1] + (1/LCapa[i])*(-LGe[i]*(LT[i][t-1]-Text(t-1))-LG[i]*(LT[i][t-1]-LT[i-1][t-1])-
LG[i]*(LT[i][t-1]-LT[i+1][t-1]) + Pcl)*dt
            Pconscl[t] = Pconscl[t] - Pcl * dt

        if not Lchauff[i] and not Lclim[i]:
            LT[i][t] = LT[i][t-1] + (1/LCapa[i])*(-LGe[i]*(LT[i][t-1]-Text(t-1))-LG[i]*(LT[i][t-1]-LT[i-1][t-1])-
LG[i]*(LT[i][t-1]-LT[i+1][t-1]))*dt

# Etage 10

    if LT[10][t-1] > 20:
        Lchauff[10] = False
    if LT[10][t-1] < 22 :
        Lclim[10] = False

    if LT[10][t-1] < 18 or Lchauff[10]  :
        Lchauff[10] = True
        LT[10][t] = LT[10][t-1] + (1/LCapa[10])*(-LGe[10]*(LT[10][t-1]-Text(t-1))-LG[9]*(LT[10][t-1]-LT[9][t-1])
+ Pch)*dt
        Pconsch[t] = Pconsch[t] + Pch * dt

    if LT[10][t-1] > 24 or Lclim[10] :
        Lclim[10] = True
        LT[10][t] = LT[10][t-1] + (1/LCapa[10])*(-LGe[10]*(LT[10][t-1]-Text(t-1))-LG[9]*(LT[10][t-1]-LT[9][t-1])
+ Pcl)*dt
        Pconscl[t] = Pconscl[t] - Pcl * dt

    if not Lchauff[10] and not Lclim[10]:
        LT[10][t] = LT[10][t-1] + (1/LCapa[10])*(-LGe[10]*(LT[10][t-1]-Text(t-1))-LG[9]*(LT[10][t-1]-LT[9]
[t-1]))*dt


## Energie produite

# Puissance instantanée

```

```
293
294 Ppan=[0]*nbjour*jour
295 for k in range(1,nbjour*jour-1):
296     Ppan[k] = Rendpan*alb*Span*Phi(k,np.pi,np.pi/2)
297
298 # Energie produite
299
300 Epan=[0]*nbjour*jour
301 for k in range(1,nbjour*jour-1):
302     Epan[k+1] = Epan[k] + Ppan[k+1]
303
304 print('Energie produite :',round(Epan[-1]),'J =',round(Epan[-1]/(3600*1000)),'kWh\nSoit :',round(Epan[-1]/
    (3600*1000*Span)),'kWh par m² par jour et',round(Epan[-1]/(3600*1000*Span))*365,'kWh par m² par an')
305
306 ## Energie consommée
307
308 Econsch = [0]*nbjour*jour
309 for k in range(1,nbjour*jour-1):
310     Econsch[k+1] = Econsch[k] + Pconsch[k+1]
311
312 Econscl = [0]*nbjour*jour
313 for k in range(1,nbjour*jour-1):
314     Econscl[k+1] = Econscl[k] + Pconscl[k+1]
315
316 Econs = [0]*nbjour*jour
317 for k in range(1,nbjour*jour-1):
318     Econs[k+1] = Econscl[k+1] + Econsch[k+1]
319
320 print('Energie consommée :',round(Econs[-1]),'J =',round(Econs[-1]/(3600*1000)),'kWh')
321
322 ## Bilan énergétique
323
324 Ebil = [0]*nbjour*jour
325 for k in range(1,nbjour*jour-1):
326     Ebil[k+1] = Epan[k+1] - Econs[k+1]
327
328 print('Bilan énergétique :',round(Ebil[-1]),'J =',round(Ebil[-1]/(3600*1000)),'kWh')
329
330 ## Graphique
331
332 ord2=[Text(k) for k in temps]
333
334 fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize =(12,9))
335 fig, ((ax5, ax6), (ax7, ax8)) = plt.subplots(2, 2, figsize =(12,9))
336 fig, ((ax9, ax10), (ax11, ax12)) = plt.subplots(2, 2, figsize =(12,9))
337 fig, ((ax13, ax14), (ax15, ax16)) = plt.subplots(2, 2, figsize =(12,9))
338 fig, ((ax17, ax18), (ax19, ax20)) = plt.subplots(2, 2, figsize =(12,9))
339
```

```
339
340 ax1.plot(temps,LT[1],label='$T_{1}$')
341 ax1.plot(temps,ord2,label='$T_{ext}$')
342 ax1.legend()
343 ax1.set_xlabel('t ')
344 ax1.set_ylabel ('T (°C)')
345 ax1.set_xticks([k*jour/2 for k in range(2*nbjour)],['0 h','12 h']*nbjour)
346 ax1.set_title('étage 1')
347
348 ax2.plot(temps,LT[2],label='$T_{2}$')
349 ax2.plot(temps,ord2,label='$T_{ext}$')
350 ax2.legend()
351 ax2.set_xlabel('t ')
352 ax2.set_ylabel ('T (°C)')
353 ax2.set_xticks([k*jour/2 for k in range(2*nbjour)],['0 h','12 h']*nbjour)
354 ax2.set_title('étage 2')
355
356 ax3.plot(temps,LT[3],label='$T_{3}$')
357 ax3.plot(temps,ord2,label='$T_{ext}$')
358 ax3.legend()
359 ax3.set_xlabel('t ')
360 ax3.set_ylabel ('T (°C)')
361 ax3.set_xticks([k*jour/2 for k in range(2*nbjour)],['0 h','12 h']*nbjour)
362 ax3.set_title('étage 3')
363
364 ax4.plot(temps,LT[4],label='$T_{4}$')
365 ax4.plot(temps,ord2,label='$T_{ext}$')
366 ax4.legend()
367 ax4.set_xlabel('t ')
368 ax4.set_ylabel ('T (°C)')
369 ax4.set_xticks([k*jour/2 for k in range(2*nbjour)],['0 h','12 h']*nbjour)
370 ax4.set_title('étage 4')
371
372 ax5.plot(temps,LT[5],label='$T_{5}$')
373 ax5.plot(temps,ord2,label='$T_{ext}$')
374 ax5.legend()
375 ax5.set_xlabel('t ')
376 ax5.set_ylabel ('T (°C)')
377 ax5.set_xticks([k*jour/2 for k in range(2*nbjour)],['0 h','12 h']*nbjour)
378 ax5.set_title('étage 5')
379
380 ax6.plot(temps,LT[6],label='$T_{6}$')
381 ax6.plot(temps,ord2,label='$T_{ext}$')
382 ax6.legend()
383 ax6.set_xlabel('t ')
384 ax6.set_ylabel ('T (°C)')
385 ax6.set_xticks([k*jour/2 for k in range(2*nbjour)],['0 h','12 h']*nbjour)
386 ax6.set_title('étage 6')
387
```