TD Modélisation 4 : Phénomènes de diffusion

Préambule python :

[1]:
```
import matplotlib.pyplot as plt
from math import sqrt, cos, sin, exp, pi
import numpy as np
from random import randint
```

1 Transfert thermique dans un mur

D'après les épreuves de modélisation CCINP 2015 et 2023 (débuts). ## Étude analytique du régime stationnaire On étudie les transferts thermiques dans le mur d'une maison (voir figure ci-dessous), considéré comme un isolant de section S et d'épaisseur e. La température à l'intérieur de la maison est notée T_{int}, et celle à l'extérieur T_{ext}.

Le mur a une épaisseur $L = e = 40cm$. Les propriétés physiques du mur sont constantes :

- conductivité thermique $\lambda = 1,65 W.m^{-1}.K^{-1}$
- capacité thermique massique à pression constante $c_p = 1000 J.kg^{-1}.K^{-1}$
- masse volumique $\rho = 2150 kg.m^{-3}$.

On note $T(x,t)$ la température dans le mur en fonction du temps. Dans le mur (pour $0 \leq x \leq e$), la température suit la loi de diffusion thermique :

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

Dans la suite, on se place en régime stationnaire.

Question 1 : A quelle condition peut-on supposer que la température ne dépend pas des coordonnées y et z ?

Correction 1 : Lorsque les dimensions transversales (selon y et z) du mur sont grandes devant l'épaisseur e du mur, on peut négliger les effets de bords, il y a invariance par translation selon y et z, et ainsi on se ramène à un problème unidimensionnel selon l'axe des x : $T(x,y,z) = T(x)$.

Question 2 : Réécrire l'équation $\rho c_p \frac{\partial T}{\partial t} = \lambda \frac{\partial^2 T}{\partial x^2}$ en régime stationnaire. Justifier que la température dans l'isolant est de la forme $T(x) = Ax + B$.

Correction 2 : En régime stationnaire $\frac{\partial T}{\partial t} = 0 \rightarrow \frac{\partial^2 T}{\partial x^2} = 0$ et donc $T(x) = Ax + B$.

Question 3 : En déduire l'expression de la température $T(x)$ en fonction des données du problème.

Correction 3 : En utilisant que $T(0) = T_{int}$ et $T(e) = T_{ext}$, on en déduit que $A = \frac{T_{ext}-T_{int}}{e}$ et $B = T_{int}$ et donc finalement :

$$T(x) = (T_{ext} - T_{int})\frac{x}{e} + T_{int}$$

On rappelle les expressions :

- de la loi de Fourier : $\vec{j}_{th}(x) = -\lambda \frac{\mathrm{d}T}{\mathrm{d}x}\vec{e_x}$
- du flux thermique : $\Phi = \iint \vec{j}_{th}(x).\mathrm{d}\vec{S}$
- de la résistance thermique : $R_{th} = \frac{T_{int}-T_{ext}}{\Phi}$

Question 4 : Proposer une interprétation physique de la résistance thermique. Montrer que $R_{th} = \frac{e}{\lambda S}$.

Correction 4 : Interprétation physique : citer (et expliquer rapidement) l'analogie électrique/thermique).

Avec la loi $T(x)$, on obtient :

$$\vec{j}_{th}(x) = -\lambda \frac{\mathrm{d}T}{\mathrm{d}x}\vec{e_x} = -\lambda \frac{T_{ext} - T_{int}}{e}\vec{e_x}$$

On a alors :

$$\Phi = \iint \vec{j}_{th}(x).\mathrm{d}\vec{S} = -\lambda \frac{T_{ext} - T_{int}}{e}\vec{e_x}.\iint \mathrm{d}\vec{S}$$

Avec $\mathrm{d}\vec{S} = \mathrm{d}S\vec{ex}$, on obtient :

$$\Phi = -\lambda \frac{T_{ext} - T_{int}}{e}\iint \mathrm{d}S = -\lambda \frac{T_{ext} - T_{int}}{e}S$$

Finalement :

$$R_{th} = \frac{T_{int} - T_{ext}}{\Phi} = \frac{e}{\lambda S}$$

Question 5 : Déterminer l'épaisseur e du mur isolant permettant d'obtenir, pour une surface $S = 2,0m^2$, une résistance thermique $R_{th} = 0,12K.W^{-1}$.

Correction 5 : $e = 40cm$

1.1 Étude numérique du régime transitoire : version 2023

On cherche à résoudre numériquement l'équation aux dérivées partielles :

$$\frac{\partial T}{\partial t} = k_{th}\frac{\partial^2 T}{\partial x^2}$$

avec $k_{th} = \frac{\lambda}{\rho c_P}$ la constante de diffusivité thermique.

Pour ce faire, on discrétise le temps et l'espace de la façon suivante :

- le temps est discrétisé en N_t dates comprises entre 0 et t_{max}, la durée de la simulation. Ces dates sont séparées du pas temporel $\mathrm{d}t = t_{max}/(N_t - 1)$ et chacune d'entre elle s'écrit $t_n = n\mathrm{d}t$ avec $n \in [\![0, N_t - 1]\!]$. Ainsi, $t_0 = 0$ et $t_{N_t-1} = t_{max}$.

- l'espace est discrétisé en N_x valeurs comprises entre 0 et $e = L$, séparées du pas spatial $\mathrm{d}x$. On note $i \in [\![0, N_x - 1]\!]$ l'indice repérant le numéro de l'un des N_x points et $x_i \in [0, L]$ la position du point correspondant. Ainsi, $x_0 = 0$ et $t_{N_x-1} = L$.

Question 6 : Donner l'expression de $\mathrm{d}x$ en fonction de L et de N_x.

Correction 6 : $\mathrm{d}x = \frac{L}{N_x-1}$

La paramétrisation précédente permet de noter $T_i^n = T(x_i, t_n)$ la température échantillonnée.

L'énoncé de 2023 donne les formules de discrétisation pour $\frac{\partial T}{\partial x}$ et $\frac{\partial T}{\partial t}$, puis demande d'en déduire celle pour $\frac{\partial^2 T}{\partial x^2}$. Réalisons nous-même ce travail, dans l'esprit de l'énoncé de 2015.

Question 7a : A l'aide d'un développement limité de la fonction $x \to T(x, t)$, donner une expression de $T(x + \mathrm{d}x, t)$ à l'ordre 3 ($o(\mathrm{d}x^3)$) en fonction de T et de ses dérivées partielles par rapport à x évaluées en (x, t). De même, donner une expression de $T(x - \mathrm{d}, t)$ à l'ordre 3.

Correction 7a : $T(x + \mathrm{d}x, t) = T(x, t) + \mathrm{d}x\frac{\partial T}{\partial x}(x, t) + \frac{(\mathrm{d}x)^2}{2}\frac{\partial^2 T}{\partial x^2}(x, t) + \frac{(\mathrm{d}x)^3}{6}\frac{\partial^3 T}{\partial x^3}(x, t) + o(\mathrm{d}x^3)$
$T(x - \mathrm{d}x, t) = T(x, t) - \mathrm{d}x\frac{\partial T}{\partial x}(x, t) + \frac{(\mathrm{d}x)^2}{2}\frac{\partial^2 T}{\partial x^2}(x, t) - \frac{(\mathrm{d}x)^3}{6}\frac{\partial^3 T}{\partial x^3}(x, t) + o(\mathrm{d}x^3)$

Question 7b : En déduire une expression approchée à l'ordre 1 ($o(\mathrm{d}x)$) de $\frac{\partial^2 T}{\partial x^2}(x, t)$ (dérivée partielle spatiale seconde de T évaluée au point x à l'instant t) en fonction de $T(x+\mathrm{d}x, t)$,$T(x-\mathrm{d}x, t)$, $T(x, t)$ et $\mathrm{d}x$.

Correction 7b : Grâce aux deux expressions précédentes qu'on somme, on obtient : $\frac{\partial^2 T}{\partial x^2}(x, t) = \frac{T(x+\mathrm{d}x,t)+T(x-\mathrm{d}x,t)-2T(x,t)}{(\mathrm{d}x)^2} + o(\mathrm{d}x)$

Question 7c : Déduire de la question précédente une expression approchée de $\frac{\partial^2 T}{\partial x^2}\Big|_{x_i,t_n}$ (dérivée partielle spatiale seconde de T évaluée en x_i à l'instant t_n) en fonction de T_i^n, T_{i+1}^n, T_{i-1}^n et $\mathrm{d}x$.

Correction 7c : On remplace dans l'expression précédente $T(x_i, t_n)$ par T_i^n, alors $T(x_i + \mathrm{d}x, t) = T(x_{i+1}, t) = T_{i+1}^n$ et on obtient : $\frac{\partial^2 T}{\partial x^2}\Big|_{x_i,t_n} \simeq \frac{T_{i+1}^n+T_{i-1}^n-2T_i^n}{(\mathrm{d}x)^2}$

La dérivée partielle temporelle de l'équation de diffusion est maintenant approchée grâce à un développement limité.

Question 7d : A l'aide d'un développement limité de la fonction $t \to T(x, t)$, donner une expression de $T(x, t + \mathrm{d}t)$ à l'ordre 1 ($o(\mathrm{d}t)$) en fonction de T et de sa dérivée partielle par rapport à t évaluées en (x, t).

Correction 7d : $T(x, t + \mathrm{d}t) = T(x, t) + \mathrm{d}t\frac{\partial T}{\partial t}(x, t) + o(\mathrm{d}t)$

Question 7e : En déduire une valeur approchée de $\frac{\partial T}{\partial t}(x, t)$ (dérivée partielle par rapport au temps de T évaluée au point x à l'instant t) à l'ordre 0 ($o(1)$) en fonction de $T(x, t + \mathrm{d}t)$, $T(x, t)$ et $\mathrm{d}t$.

Correction 7e : $\frac{\partial T}{\partial t}(x, t) = \frac{T(x,t+\mathrm{d}t)-T(x,t)}{\mathrm{d}t} + o(1)$

Question 7f : Donner une expression de $\frac{\partial T}{\partial t}\Big|_{x_i,t_n}$ (dérivée partielle par rapport au temps de T évaluée en x_i à l'instant t_n) en fonction de T_i^n, T_i^{n+1} et $\mathrm{d}t$, avec $T_i^{n+1} = T(x_i, t_n + \mathrm{d}t)$.

Correction 7f : $\frac{\partial T}{\partial t}\Big|_{x_i,t_n} \simeq \frac{T_i^{n+1}-T_i^n}{\mathrm{d}t}$

L'équation de diffusion est valable en chaque point d'abscisse x_i et à chaque instant t_n.

Question 7g : Écrire la forme approchée de cette équation au point i et à l'instant n en approchant $\frac{\partial^2 T}{\partial x^2}\Big|_{x_i,t_n}$ avec la formule obtenue à la question 6, et en approchant $\frac{\partial T}{\partial t}\Big|_{x_i,t_n}$ avec la formule obtenue à la question précédente.

Correction 7g : $\frac{T_i^{n+1}-T_i^n}{\mathrm{d}t} \simeq k_{th}\frac{T_{i+1}^n+T_{i-1}^n-2T_i^n}{(\mathrm{d}x)^2}$

Question 8 : Montrer que l'équation obtenue à la question précédente peut s'écrire sous la forme :

$$T_i^{n+1} = K(T_{i-1}^n + T_{i+1}^n) + (1-2K)T_i^n$$

en précisant l'expression du paramètre K en fonction de $\mathrm{d}x$, $\mathrm{d}t$ et k_{th}.

Correction 8 : $K = k_{th}\frac{\mathrm{d}t}{(\mathrm{d}x)^2}$

Cette dernière équation est appelée schéma numérique explicite. Si on connaît la température en tous les points x_1, x_2, ..., x_{N+1}, x_N à l'instant t_n, on peut calculer grâce à elle la température en tous les points à l'instant ultérieur t_{n+1}.

On suppose tout d'abord que le mur est à la température extérieur : $T(x, t<0) = T_{ext} = 5,0°C$.

A la date $t = 0$, on impose les conditions aux limites :

$$\begin{cases} T(0, t \geq 0) = T_{int} = 20°C \\ T(L, t \geq 0) = T_{ext} = 5,0°C \end{cases}$$

Le code python ci-dessous implémente ce schéma numérique pour calculer les échantillons de température. Dans les questions qui suivent, on cherche à compléter les passages manquants du code, repérés par des "@@@".

[ ]:
```
import numpy as np

# Caractéristique du mur
rho = 2150                    # masse volumique [kg/m^3]
Lambda = 1.65                 # conductivité thermique [W/m/K]
cp = 1e3                      # capacité thermqieu massique [J/K/kg]
kth = Lambda / (rho * cp)     # diffusivité thermique [m^2/s]
L = 4e-1                      # épaisseur du mur [m]

# Conditions aux limites
Tint = 20                     # température intérieur [°C]
Text = 5.0                    # température extérieur [°C]

# Paramètres de la simulation
tmax = 72000                  # durée de la simulation [s] (ici 20h)
Nt = 3456                     # discrétisation temporelle (Nt dates)
Nx = 60                       # discrétisation spatiale (Nx points)

# Discrétisation spatio-temporelle
dt = tmax / (Nt - 1)          # pas temporel [s]
dx = @@@                      # pas spatial [m]                    # Q9
```

```
# Constante du schéma numérique
K = @@@                                                          # Q10

# Initialisation de la température
T = Text * np.ones((Nx, Nt))  # tableau des températures T(x,t) [°C]
@@@                                                              # Q11

# Calcul de la température
for n in range(0, Nt-1):      # boucle temporelle
    for i in range(1, @@@):   # boucle spatiale                  # Q12
        T[i, n + 1] = @@@                                        # Q13
```

Question 9 : A l'aide de la question 6, donner l'instruction permettant de définir le pas spatial `dx` en fonction des variables `L` et `Nx`.

Question 10 : A l'aide de la question 8, donner l'instruction permettant de définir la constante `K` du schéma numérique en fonction des variables `kth`, `dx` et `dt`.

Question 11 : Dans la partie `Initialisation des températures`, la première ligne permet d'initialiser une matrice des températures de N_x lignes et N_t colonnes à une valeur initiale identique T_{ext}. Proposer un code à la ligne suivante pour assurer la condition aux limites $T(x = 0, t) = T_{int}$.

Question 12/13 : On rappelle que la dernière valeur fournie par l'instruction `range(1, Nx)` est `Nx -1`, l'intervalle de la fonction `range` étant semi-ouvert. En examinant le schéma numérique, compléter la boucle spatiale. Implémenter ensuite le schéma numérique à l'intérieur des deux boucles.

Correction 9/10/11/12 :

[48]:
```
import numpy as np

# Caractéristique du mur
rho = 2150                    # masse volumique [kg/m^3]
Lambda = 1.65                 # conductivité thermique [W/m/K]
cp = 1e3                      # capacité thermqieu massique [J/K/kg]
kth = Lambda / (rho * cp)     # diffusivité thermique [m^2/s]
L = 4e-1                      # épaisseur du mur [m]

# Conditions aux limites
Tint = 20                     # température intérieur [°C]
Text = 5.0                    # température extérieur [°C]

# Paramètres de la simulation
tmax = 72000                  # durée de la simulation [s] (ici 20h)
Nt = 5000                     # discrétisation temporelle (Nt dates)
Nx = 60                       # discrétisation spatiale (Nx points)

# Discrétisation spatio-temporelle
dt = tmax / (Nt - 1)          # pas temporel [s]
```

```
dx = L / (Nx - 1)             # pas spatial [m]                          # Q9

# Constante du schéma numérique
K = kth * dt / dx**2                                                     # Q10

# Initialisation de la température
T = Text * np.ones((Nx, Nt))  # tableau des températures T(x,t) [°C]
for n in range(Nt):                                                      # Q11
    T[0, n] = Tint
# Ou plus efficace : T[0, :] = Tint

# Calcul de la température
for n in range(0, Nt - 1):       # boucle temporelle
    for i in range(1, Nx - 1):   # boucle spatiale                       # Q12
        T[i, n + 1] = K * (T[i-1, n] + T[i+1, n]) + (1 - 2*K) * T[i, n]# Q13
```

Une partie non mentionnée du code a permis de tracer la température dans le mur aux dates t_1, t_2 et t_3 (on a limité ici Nx à 5).

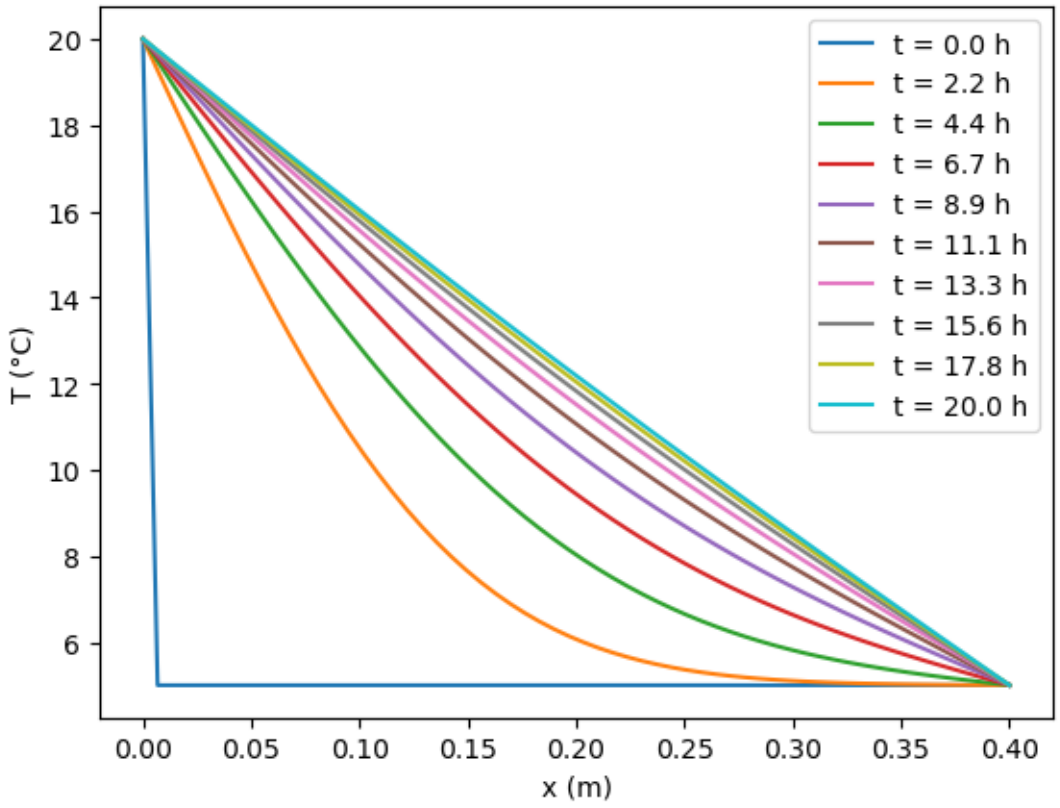
Question 14 : A l'aide de ce graphe, classer les dates t_1, t_2 et t_3 de façon croissante.

Question 15 : Compte tenu de la question 2, justifier que le régime permanent est pratiquement atteint.

Correction 14/15 : $t_3 < t_1 < t_2$ car plus le temps "avance", plus on se rapproche du régime permanent pourlequel la température est une fonction affine du temps (graphe : droite allant de T_{int} à T_{ext}). D'aillleurs, au temps t_2, on remarque que le graphe est quasiment une droite : le régime transitoire est quasiment fini, le régime permanent est pratiquement atteint.

La suite de l'énoncé traite d'autres domaines (plus de diffusion). On peut cependant ici tester la "bonne" exécution de notre code :

[57]:
```
X = np.linspace(0, L, Nx) # Création des Nx points d'abscisse

plt.figure()
for i in range(10): # création de 10 graphes régulièrement espacés dans le temps
    plt.plot(X,T[:, i*(Nt-1)//9],label ='t = '+str(round(i*(Nt-1)*dt/
 ↪(9*3600),1))+' h')
plt.xlabel('x (m)')
plt.ylabel('T (°C)')
plt.legend()
plt.show()
```



On trouve bien un graphe similaire (mais bien plus précis !) à celui proposé à la question 14.

1.2 Étude numérique du régime transitoire : version 2015

Dans la version précédente (2023), la condition initiale considérait le mur à la température extérieure. Cela revient au problème d'une maison dans laquelle le chauffage a été longtemps éteint : après un régime transitoire, toute la maison (murs compris) sont donc à la température extérieur $T_{ext} = 5,0°C$. On allume alors le chauffage à $t = 0$: on considère que l'air est immédiatement à la température intérieure désirée $T_{int} = 20°C$, alors que les murs sont pour l'instant encore à la température extérieure.

Dans cette autre version, la température à l'intérieur de la maison est constante dans le temps et égale à $T_{int} = 20°C$. Aux temps négatifs ($t < 0$, la température extérieure est égale à $T^1_{ext} = 10°C$. A $t = 0$, elle chute brusquement à $T^2_{ext} = -10°C$ et elle reste égale à cette valeur aux temps positifs ($t > 0$). On souhaite étudier l'évolution du profil de température dans le mur au cours du temps.

Cela revient à considérer le problème du passage jour/nuit, pendant lequel la température extérieure chute. On considère ici que ce changement de température extérieur est immédiat.

Question 16 : On considère deux instants :

- pour un instant particulier négatif $t_1 < 0$,
- pour un instant particulier positif $t_2 > 0$, très longtemps après la variation de température

extérieure, quand le régime permanent est de nouveau établi dans le mur.

Quelle est la nature des profils $T(x)$ obtenus (en régime permanent) à ces deux instants ? Tracer à la main les deux profils sur un même graphique.

Correction 16 : En régime stationnaire, le profil suit celui d'une fonction affine. En rouge pour $t < 0$, et en bleu ciel pour $t \to +\infty$ et dans d'autres couleurs pour les cas intérmédiaires.

Question 17 : Sur le même graphique, tracer à la main qualitativement les profils intermédiaires à différents instants entre la variation brutale de la température extérieure ($t = 0$) et l'instant t_2 où le régime est de nouveau permanent.

Correction 17 : Juste après le changement brutal, la température n'a significativement varié qu'au voisinage de e (tracé en bleu foncé). Après un temps suffisamment long le changement de température se fait ressentir dans toute l'épaisseur du mur (tracé vert). On peut utiliser le temps caractéristique d'évolution $L \simeq \sqrt{k_{th}\tau}$ de la température sur une longueur L, avec $k_{th} = \frac{\lambda}{\rho c_p}$ coefficient de diffusion thermique :

- $\tau = 10s$ donne $L \simeq 3mm$
- $\tau = 500s$ donne $L \simeq 2cm$

Pour résoudre numériquement le nouveau régime transitoire, les conditions initiale et aux limites s'écrivent :

$$\begin{cases} T(0,t) = T_{int} & \forall t > 0 \\ T(e,t) = T_{ext}^2 & \forall t > 0 \\ T(x,0) = a.x + b & \forall x \in [0,e] \end{cases}$$

Question 18 : Exprimer a et b en fonction de T_{int}, T_{ext}^1 et e.

Correction 18 : Voir question 3 : $a = \frac{T_{ext}^1 - T_{int}}{e}$ et $b = T_{int}$.

Question 19 : Compléter le code ci-dessous pour effectuer la simulation numérique donnant la matrice des températures.

[24]:
```
# Conditions aux limites
Tint = 20                    # température intérieur [°C]
Text1 = 10                   # température extérieur pour t < 0 [°C]
Text2 = -10                  # température extérieur pour t >= 0 [°C]

# Initialisation de la température
T = np.zeros((Nx, Nt))                    # tableau des températures T(x,t)
 ↪[°C] initialisé à 0
a, b = (Text1 - Tint) / L, Tint           # respect de la condition initiale
 ↪en t = 0
X = np.linspace(0, L, Nx)
T[:, 0] = a * X +b
T[0, :] = Tint                            # respect de la condition à la
 ↪limite en x = 0
T[Nx - 1, :] = Text2                      # respect de la condition à la
 ↪limite en x = e
```

```
# Calcul de la température
for n in range(0, Nt-1):      # boucle temporelle
    for i in range(1, Nx-1):  # boucle spatiale
        T[i, n + 1] =  K * (T[i-1, n] + T[i+1, n]) + (1 - 2*K) * T[i, n]
```

On souhaite arrêter le calcul lorsque la température ne varie presque plus dans le temps. Dans ce but, on évaluera la norme 2 de $T^n - T^{n-1}$ à chaque itération. On donne la définition de la norme 2 d'un vecteur V :

$$||V||_2 = \sqrt{\sum_{i=1}^{n} V_i^2}$$

avec $V = \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_{n-1} \\ V_n \end{pmatrix}$.

Question 20: Ecrire une fonction `calc_norme` qui calcule la norme 2 d'un vecteur.

Correction 20:

[25]:
```
def calc_norme(V):
    return sqrt(sum([vi**2 for vi in V]))

# Ou, sans tricher :
def calc_norme(V):
    s = 0
    for elem in V:
        s += elem**2
    return sqrt(s)
```

Test :

[26]:
```
B=[1,2,3]
print(calc_norme(B)**2)
```

```
14.0
```

Question 21: Modifier le code de la question 17 pour interrompre la simulation numérique lorsque la la norme 2 du vecteur $T_n - T_{n-1}$ deviendra inférieure à 5.10^{-3} (on considère alors que la régime permanent est atteint) ou lorsque le nombre d'itérations atteindra la valeur `Nt` (prévoir les deux cas). Utiliser, pour cela, la fonction `calc_norme` définie à la question précédente.

Correction 21:

[58]:
```
# Conditions aux limites
Tint = 20                    # température intérieur [°C]
Text1 = 10                   # température extérieur pour t < 0 [°C]
Text2 = -10                  # température extérieur pour t >= 0 [°C]
```

```
# Initialisation de la température
T = np.zeros((Nx, Nt))                          # tableau des températures T(x,t) [°C] initialisé à 0
a, b = (Text1 - Tint) / L, Tint                 # respect de la condition initiale en t = 0
X = np.linspace(0, L, Nx)
T[:, 0] = a * X +b
T[0, :] = Tint                                  # respect de la condition à la limite en x = 0
T[Nx - 1, :] = Text2                            # respect de la condition à la limite en x = e

## Calcul de la température

# Première itération temporelle, afin d'avoir deux vecteurs de température successifs :
n = 0
for i in range(1, Nx-1):  # boucle spatiale
        T[i, n + 1] =  K * (T[i-1, n] + T[i+1, n]) + (1 - 2*K) * T[i, n]

while n < Nt-2 and calc_norme(T[:, n+1] - T[:, n]) > 5e-3:     # boucle temporelle
    n += 1
    for i in range(1, Nx-1):  # boucle spatiale
        T[i, n + 1] =  K * (T[i-1, n] + T[i+1, n]) + (1 - 2*K) * T[i, n]

print("Le nombre d'itération temporel est de  : "+str(n+2)+"/"+str(Nt))
```

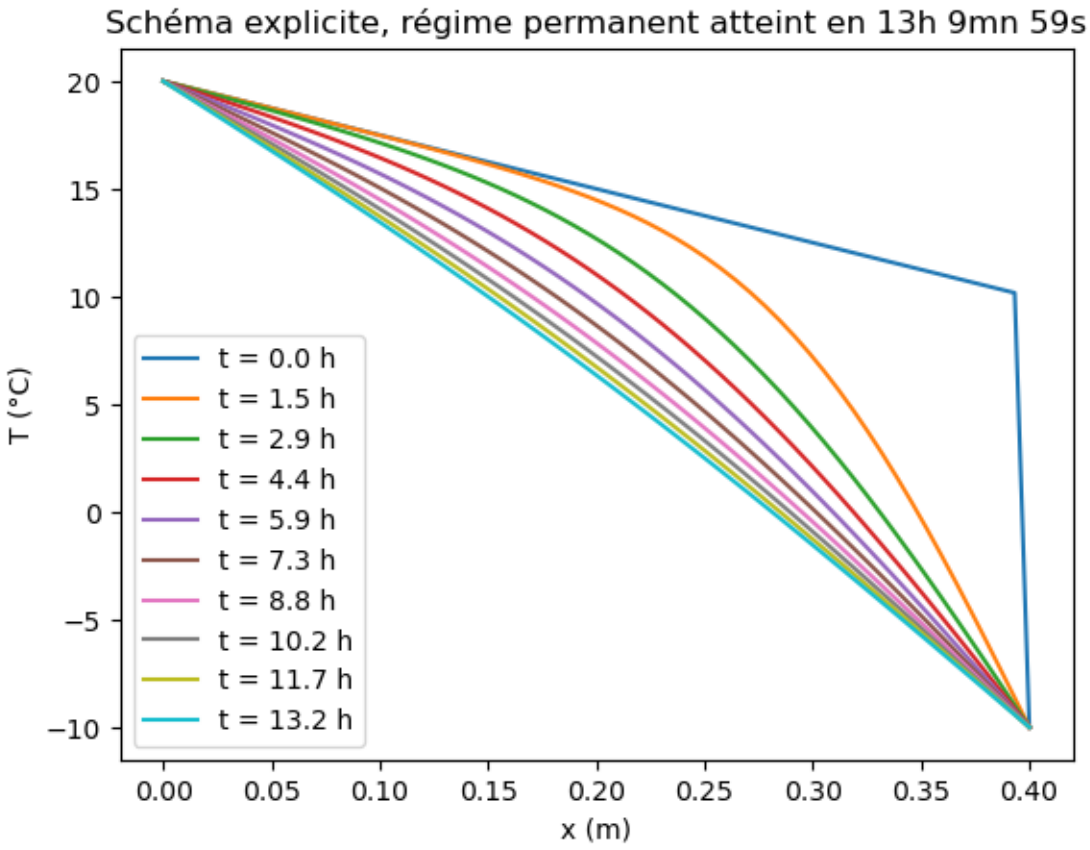
```
Le nombre d'itération temporel est de  : 3293/5000
```

Question 22 : Écrire un script permettant de tracer sur un même graphique le profil de température en fonction de x pour 10 temps régulièrement espacés. Faire afficher le temps en heures au bout duquel le régime permanent est établi.

Correction 22:

[63]:
```
X = np.linspace(0, L, Nx) # Création des Nx points d'abscisse

temps=n*dt
h=temps//3600
m,s=(temps%3600)//60,(temps%3600)%60

plt.figure()
plt.title("Schéma explicite, régime permanent atteint en {0}h {1}mn {2}s".format(int(h),int(m),int(s)))
for i in range(10): # création de 10 graphes régulièrement espacés dans le temps
    plt.plot(X,T[:, i*(n-1)//9],label ='t = '+str(round(i*(n-1)*dt/(9*3600),1))+' h')
```

```
plt.xlabel('x (m)')
plt.ylabel('T (°C)')
plt.legend()
plt.show()
```



Ces questions sont une version adaptée (simplifiée) du début de l'énoncé CCINP 2015. La suite de l'énoncé poursuit sur le même problème, avec des méthodes plus complexes…

1.3 Transfert thermique dans un oeuf

D'après Oral Centrale Physique 2.

1.4 Théorie et solution numérique

On s'intéresse à la cuisson d'un œuf de poule. Celui-ci est constitué d'une coquille calcaire contenant principalement deux zones distinctes : l'une périphérique appelée « blanc » (albumen) et l'autre centrale appelée « jaune » (vitellus). Ces deux zones sont fluides avant cuisson.

On modélise un œuf par une sphère de rayon R, avec une répartition à symétrie sphérique du blanc

et du jaune. L'équation de diffusion s'écrit :

$$\frac{\partial T(r,t)}{\partial t} = D\Delta T(r,t)$$

avec D diffusivité et Δ laplacien.

Question 1 : Montrer que, compte-tenu de la symétrique sphérique du problème, l'équation de diffusion s'écrit :

$$\frac{\partial T(r,t)}{\partial t} = D\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial T(r,t)}{\partial r}\right)$$

Aide : on rapelle que, en sphérique :

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

Correction 1 : D'après la symétrie du problème (invariance par rotation selon θ et φ), la température $T(r,\theta,\varphi,t)$ ne dépend que de t et r : $T(r,\theta,\varphi,t) = T(r,t)$. On a donc :

$$\Delta T(r,t) = \frac{\partial^2 T}{\partial r^2} + \frac{2}{r}\frac{\partial T}{\partial r} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial T}{\partial r}\right)$$

L'équation de diffusion $D\Delta T = \frac{\partial T}{\partial t}$ avec D coefficient de diffusion s'écrit ainsi comme demandé.

On souhaite maintenant modéliser numériquement l'évolution au cours du temps du profil de température au sein de l'œuf. On utilise pour cela la méthode d'Euler pour discrétiser l'équation précédente.

Le principe adopté est le suivant : le profil de température initial T (r, t = 0) au sein de l'œuf est connu. On pose comme variable intermédiaire : $\Theta(r,t) = \frac{\partial T(r,t)}{\partial r}$.

Question 2 : Ecrire le système d'équation différentiel d'ordre 1 faisant intervenir les fonctions $T(r,t)$ et $\Theta(r,t)$.

Correction 2 :

$$\begin{cases} \Theta(r,t) = \frac{\partial T(r,t)}{\partial r} \\ \frac{\partial T(r,t)}{\partial t} = D\frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\Theta(r,t)\right) \end{cases}$$

On note τ_e et l_e les périodes d'échantillonnage (ou « pas ») temporelle et spatiale de la résolution numérique.

Question 3 : Montrer, à l'aide d'un D.L. à l'ordre 1 en l_e de $T(r-l_e,t)$, qu'on a $\Theta(r,t) = \frac{T(r,t)-T(r-l_e,t)}{l_e} + o(1)$

Correction 3 : $T(r-l_e,t) = T(r,t) - l_e\frac{\partial T(r,t)}{\partial r} + o(l_e)$ et ainsi $T(r,t) - T(r-l_e,t) = l_e\frac{\partial T(r,t)}{\partial r} + o(l_e) = l_e\Theta(r,t) + o(l_e)$ et on obtient ainsi le résultat demandé.

Question 4 : Montrer, à l'aide d'un D.L. à l'ordre 1 en l_e de $f(r+l_e,t) = (r+l_e)^2\Theta(r+l_e,t)$, que $\frac{\partial r^2\Theta(r,t)}{\partial r} = \frac{(r+l_e)^2\Theta(r+l_e,t) - r^2\Theta(r,t)}{l_e} + o(1)$

Correction 4 : $f(r+l_e,t) = (r+l_e)^2\Theta(r+l_e,t) = f(r,t) + l_e\frac{\partial f(r,t)}{\partial r} + o(l_e) = r^2\Theta(r,t) + l_e\frac{\partial r^2\Theta(r,t)}{\partial r} + o(l_e)$ et donc $(r+l_e)^2\Theta(r+l_e,t) - r^2\Theta(r,t) = l_e\frac{\partial r^2\Theta(r,t)}{\partial r} + o(l_e)$, et ainsi on trouve le résultat demandé.

On note T_i^k la température $T(r_k, t_i)$, évaluée au point d'abscisse $r_k = k \times l_e$ à l'instant $t_i = i \times \tau_e$, et $\Theta_i^k = \Theta(r_k, t_i)$.

Question 5 : En déduire alors, via la méthode d'Euler, le schéma numérique suivant :

$$\begin{cases} \Theta_i^k = \frac{T_i^k - T_i^{k-1}}{l_e} \\ T_{i+1}^k = T_i^k + \frac{D\tau_e}{(kl_e)^2} \frac{((k+1)l_e)^2 \Theta_i^{k+1} - (kl_e)^2 \Theta_i^k}{l_e} \end{cases}$$

Correction 5 : Il "suffit" de ré-écire les deux équations obtenus à la question 2, en se servant des deux D.L. précédemment établis.

Dans la suite, on modifie légerement ce schéma par celui-ci, équivalent du point de vue mathématique, et évitant de diviser par 0 :

$$\begin{cases} \Theta_i^k = \frac{T_i^k - T_i^{k-1}}{l_e} \\ T_{i+1}^k = T_i^k + \frac{D\tau_e}{((k+1)l_e)^2} \frac{((k+1)l_e)^2 \Theta_i^{k+1} - (kl_e)^2 \Theta_i^k}{l_e} \end{cases}$$

On cherche à tracer des graphiques successifs sur l'évolution temporelle de la température en fonction de l'espace. Pour cela, on considérera deux listes `Temp` et `Theta` stockant les valeurs de la température et de Θ au niveau des N+1 points d'espace.

Question 6 : Ecrire un script python définissant le rayon de l'oeuf $R = 2\ cm$, et permettant, à partir de N = 100, de calculer le pas d'espace `le`, la liste des positions de l'espace `r` (contenant les N+1 points entre 0 et R), et créant la liste `Temp`, pour l'instant initial. A l'instant initial, on a tout juste plongé l'oeuf dans l'eau bouillante : la température initiale sera prise égale à 20°C, sauf à la surface de l'oeuf.

Correction 6 :

[8]:
```
N = 100
R = 2e-2
r = np.linspace(0,R,N+1)
le = R/N
Temp = N*[293] + [373]
```

Question 7 : Ecrire un scirpt python définissant les autres données du problème : $D = 1,4.10^{-7}\ m^2.s^{-1}$ le coefficient de diffusion, $n+1 = 10001$ le nombre de points temporels à considérer, et τ_e le pas temporel, sachant que la durée totale de l'expérience sera $T = 15\ min$

Correction 7 :

[9]:
```
D = 1.4e-7
n = 10000
T = 15 * 60
taue = T / n
```

Question 8 : Ecrire le script mettant à jour la liste `Theta` pour le temps i, et la liste `Temp` pour le temps $i+1$, sachant qu'on déjà les listes `Theta` pour le temps $i-1$ et `Temp` pour le temps i. La température extérieure sera en permanence celle de l'eau bouillante, et on admet qu'on a à tout instant `Theta[0]=0`.

Correction 8 :

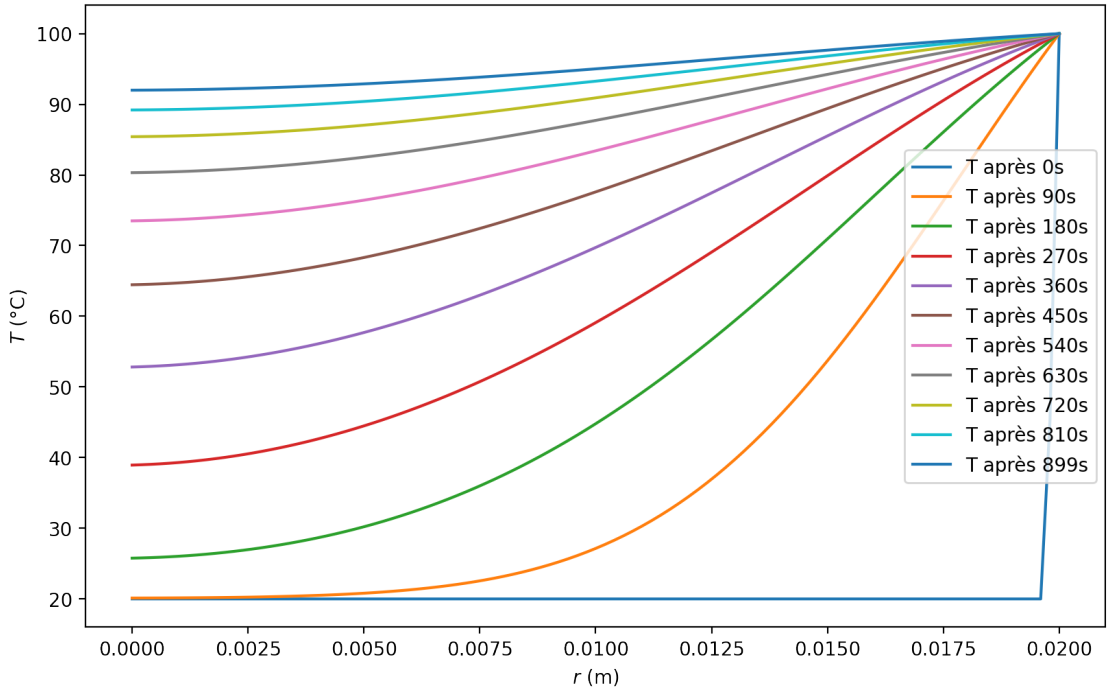
[10]:
```
Theta=[0] # theta au centre

for k in range(N) :
    Theta.append((Temp[k+1]-Temp[k])/le) # calcul de theta en k+1 à l'instant i

    Temp[k] += D*taue/
 ↪((k+1)*le)**2*(((k+1)*le)**2*Theta[k+1]-(k*le)**2*Theta[k])/le # calcul de␣
 ↪temp en k à l'instant i+1

Temp[N] = 373 #la dernière valeur reste inchangée (100°C)
```

Question 9 : Insérer ce script dans une boucle permettant d'effectuer tous les pas de temps. On ajoutera une commande permettant de tracer, sur un même graphique, 10 courbes représentant le profil de température à des temps réparties régulièrement entre le début et la fin de la cuisson.

Correction 9 :

[13]:
```
N = 100
R = 2e-2
r = np.linspace(0,R,N+1)
le = R/N
Temp = N*[293] + [373]
D = 1.4e-7
n = 10000
T = 15 * 60
taue = T / n

ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
for i in range(n): # on calcul pour chaque pas de temps

    Theta=[0] # theta au centre

    for k in range(N) :
        Theta.append((Temp[k+1]-Temp[k])/le) # calcul de theta en k+1 à␣
 ↪l'instant i
        Temp[k] += D*taue/
 ↪((k+1)*le)**2*(((k+1)*le)**2*Theta[k+1]-(k*le)**2*Theta[k])/le # calcul de␣
 ↪temp en k à l'instant i+1

    Temp[N] = 373 #la dernière valeur reste inchangée (100°C)

    # 9 premiers plot (T en degrés, pour une meilleure lisibilité)
    if i*10%n == 0:
        plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")
```

```
# Dernier graphe
plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
  ↪"+str(int(i*taue))+"s")

plt.xlabel('$r \ \mathrm{(m)}$')
plt.ylabel('$T \ \mathrm{(°C)}$')
plt.legend()
plt.show()
```



On donne quelques informations sur la cuisson d'un oeuf :

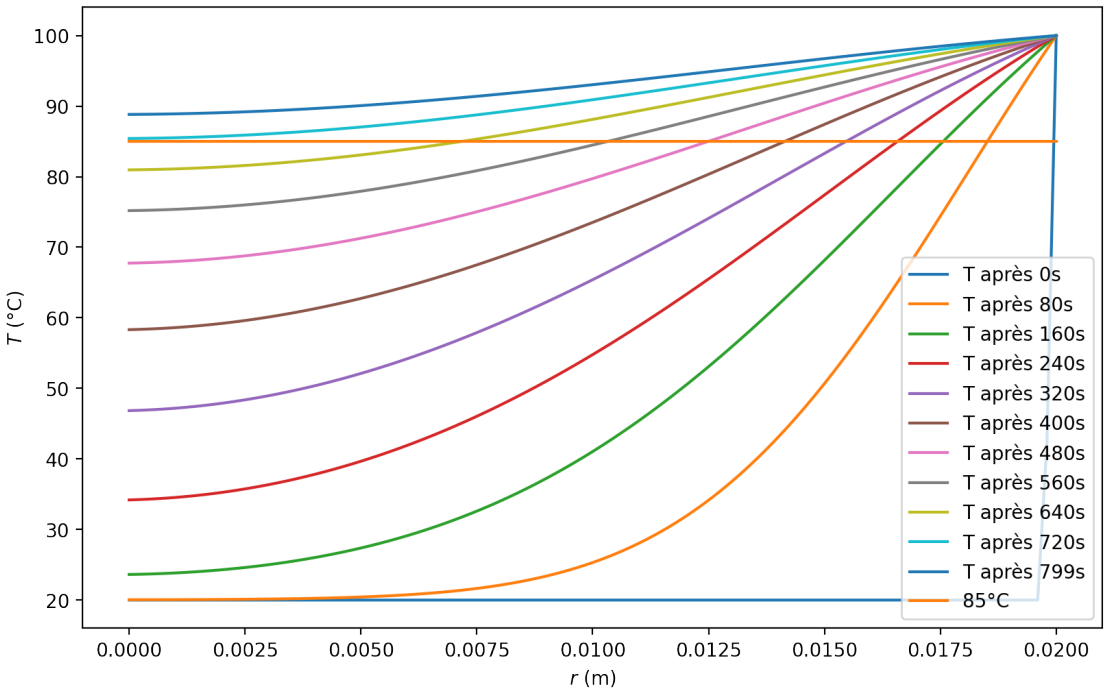
- Debut de coagulation de l'ovalbumine (protéines des blancs) à 62°C, et formation d'un gel très ferme à 70°C.
- Début de l'épaississement à 65°C par coagulation des protéines du jaune (ovovitelline), perte de fluidité à 70°C, coagulation totale à 85°C.

Question 10 : Estimer le temps pour obtenir un oeuf dur.

Correction 10 : D'après le graphique précédent, la température semble atteindre les 85°C à coeur aux alantours des 800s. Vérifions le :

[12]:
```
T = 800
taue = T / n
Temp = N*[293] + [373]
```

```
ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
for i in range(n): # on calcul pour chaque pas de temps

    Theta=[0] # theta au centre

    for k in range(N) :
        Theta.append((Temp[k+1]-Temp[k])/le) # calcul de theta en k+1 à␣
 ↪l'instant i
        Temp[k] += D*taue/
 ↪((k+1)*le)**2*(((k+1)*le)**2*Theta[k+1]-(k*le)**2*Theta[k])/le # calcul de␣
 ↪temp en k à l'instant i+1

    Temp[N] = 373 #la dernière valeur reste inchangée (100°C)

    # 9 premiers plot (T en degrés, pour une meilleure lisibilité)
    if i*10%n == 0:
        plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

# Dernier graphe
plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

plt.plot(r,[85 for k in range(N+1)],label = "85°C ")

plt.xlabel('$r \ \mathrm{(m)}$')
plt.ylabel('$T \ \mathrm{(°C)}$')
plt.legend()
plt.show()
```



De façon plus précise, 720s, soit 12 min semblent suffirent. Usuellement, on parle plutôt de 9 min pour un oeuf dur…

1.5 Pour aller plus loin : jaune et blanc

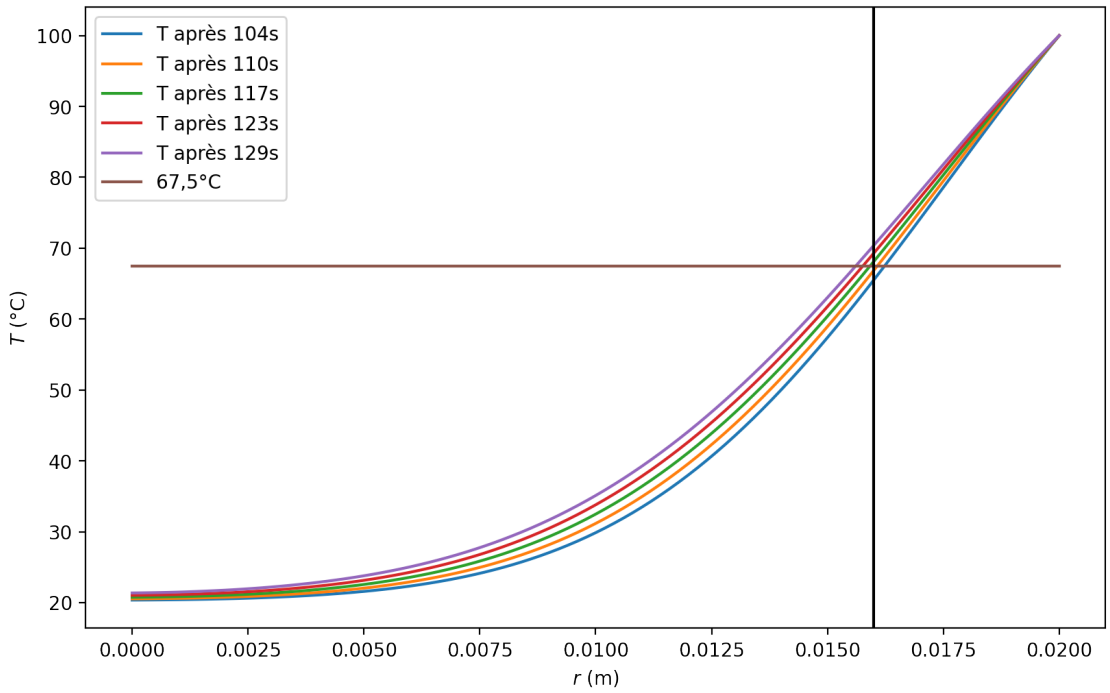
On considère que la répartition du jaune dans un oeuf vérifie $\alpha \simeq 0,8$.

Question 11 : Par tatonnement, en modifiant la valeur de `T`, déterminer la durée necessaire pour que l'oeuf soit "à la coque".

Correction 11 : Pour un oeuf à la coque, il faut que la majeure partie du jaune se trouve en dessous des 65°C, et la majeure partie du blanc au dessus de 70°C (sachant que le jaune se trouve jusqu'à 1,6 cm du centre). Sur le graphe précédent, on remarque que pour T = 160s, l'oeuf est déjà trop cuit. Regardons avec plus de précision :

[14]:
```
T = 130
taue = T / n
Temp = N*[293] + [373]

ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
for i in range(n): # on calcul pour chaque pas de temps

    Theta=[0] # theta au centre

    for k in range(N) :
```

```
        Theta.append((Temp[k+1]-Temp[k])/le) # calcul de theta en k+1 à␣
 ↪l'instant i
        Temp[k] += D*taue/
 ↪((k+1)*le)**2*(((k+1)*le)**2*Theta[k+1]-(k*le)**2*Theta[k])/le # calcul de␣
 ↪temp en k à l'instant i+1

    Temp[N] = 373 #la dernière valeur reste inchangée (100°C)

    if i*20%n == 0 and i*taue > 100:
        plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

# Dernier graphe
plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

# Ligne horizontale  "à la bonne température"
plt.plot(r,[67.5 for k in range(N+1)],label = "67,5°C ")

# Ligne verticale de séparation blanc/jaune
plt.axvline(x=1.6e-2,color='black')


plt.xlabel('$r \ \mathrm{(m)}$')
plt.ylabel('$T \ \mathrm{(°C)}$')
plt.legend()
plt.show()
```



Le temps nécessaire est donc compris entre 110 et 117s (alors qu'on parle plutôt de 3 min…).

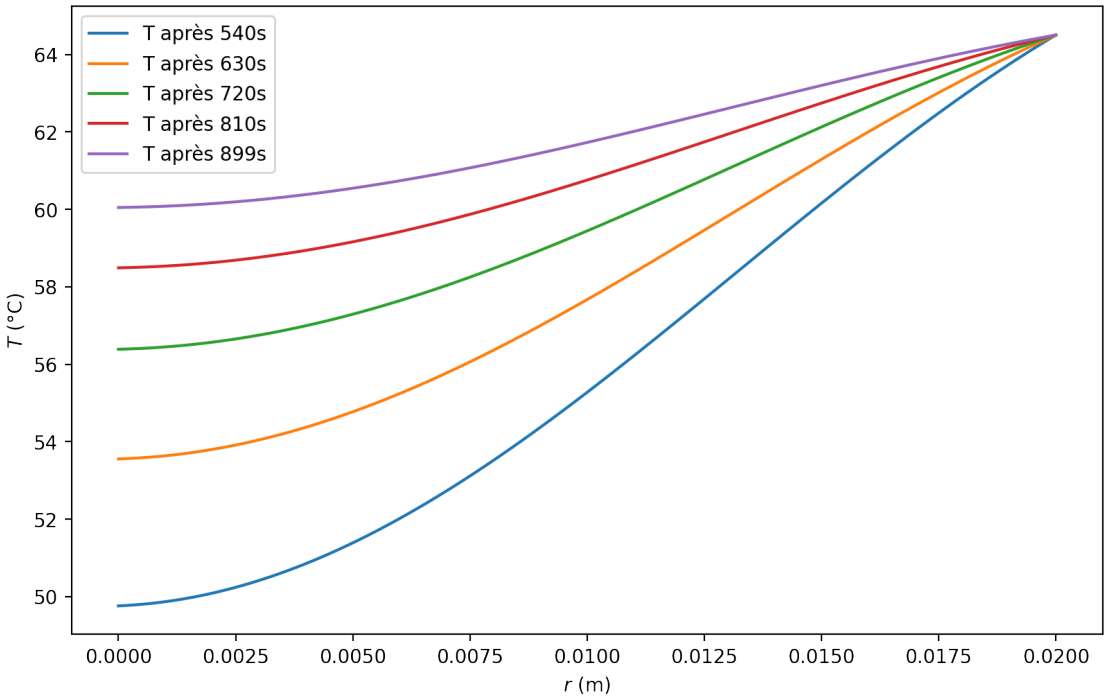
L'oeuf dit "parfait", servi dans certains restaurants, s'obtient à l'aide d'une cuisson à 64,5 °C.

Question 12 : Déterminer l'ordre de grandeur de la durée nécessaire pour que l'oeuf soit "parfait".

Correction 12 : On impose ici une température extérieur de cuisson de 64,5 °C, et on estime le temps nécessaire pour qu'elle soit atteinte partout, même au centre de l'oeuf :

[15]:
```
T = 15 * 60
taue = T / n
Temp = N*[293] + [273+64.5]

ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
for i in range(n): # on calcul pour chaque pas de temps

    Theta=[0] # theta au centre

    for k in range(N) :
        Theta.append((Temp[k+1]-Temp[k])/le) # calcul de theta en k+1 à␣
↪l'instant i
        Temp[k] += D*taue/
↪((k+1)*le)**2*(((k+1)*le)**2*Theta[k+1]-(k*le)**2*Theta[k])/le # calcul de␣
↪temp en k à l'instant i+1
```

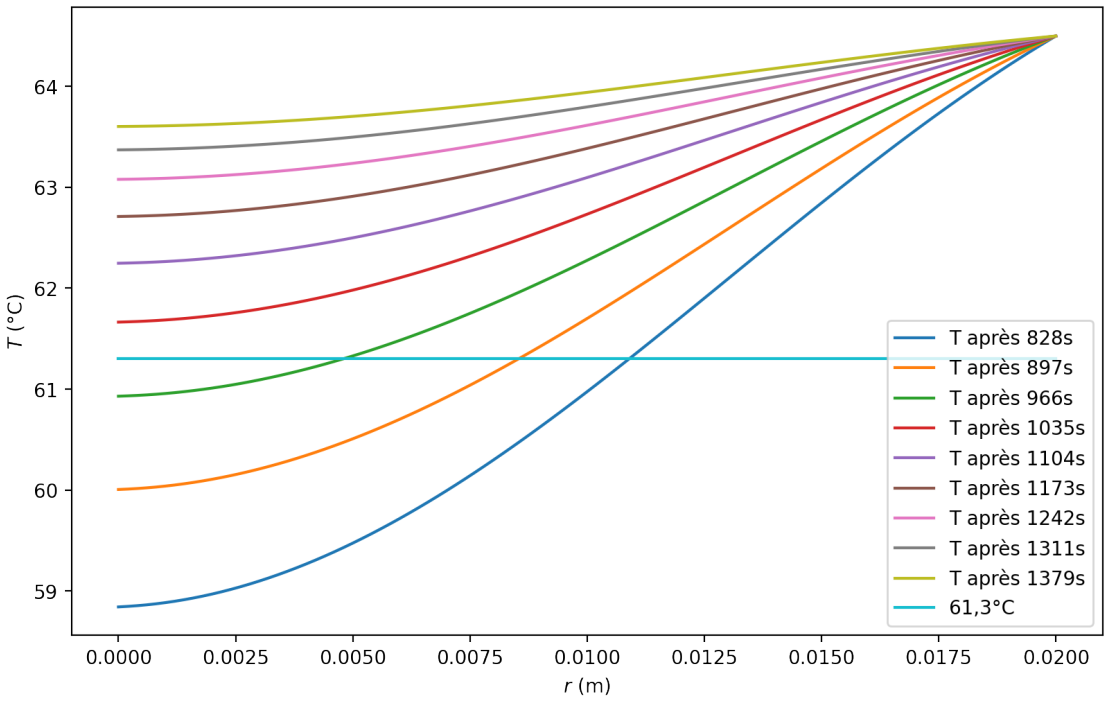
```
    Temp[N] = 273+64.5 #la dernière valeur reste inchangée (100°C)

    # 9 premiers plot (T en degrés, pour une meilleure lisibilité)
    if i*10%n == 0 and i*taue > 500:
        plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

# Dernier graphe
plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

plt.xlabel('$r \ \mathrm{(m)}$')
plt.ylabel('$T \ \mathrm{(°C)}$')
plt.legend()
plt.show()
```



Bien evidemment, cela n'est pas possible avant un temps infini. Estimons le temps necessaire pour qua la température au centre soit d'au moins 95% de celle à atteindre, soit 61,3°C :

[16]:
```
T = 23 * 60
taue = T / n
Temp = N*[293] + [273+64.5]

ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
```

```
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
for i in range(n): # on calcul pour chaque pas de temps

    Theta=[0] # theta au centre

    for k in range(N) :
        Theta.append((Temp[k+1]-Temp[k])/le) # calcul de theta en k+1 à␣
 ↪l'instant i
        Temp[k] += D*taue/
 ↪((k+1)*le)**2*(((k+1)*le)**2*Theta[k+1]-(k*le)**2*Theta[k])/le # calcul de␣
 ↪temp en k à l'instant i+1

    Temp[N] = 273+64.5 #la dernière valeur reste inchangée (100°C)

    # 9 premiers plot (T en degrés, pour une meilleure lisibilité)
    if i*20%n == 0 and i*taue > 800:
        plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

# Dernier graphe
plt.plot(r,[Temp[p]-273 for p in range(N+1)],label = "T après␣
 ↪"+str(int(i*taue))+"s")

# Ligne horizontale  "à 95% de la bonne température"
plt.plot(r,[61.3 for k in range(N+1)],label = "61,3°C ")

plt.xlabel('$r \ \mathrm{(m)}$')
plt.ylabel('$T \ \mathrm{(°C)}$')
plt.legend()
plt.show()
```



17min semblent donc être suffisantes !

En réalité le blanc et le jaune n'ont pas tout à fait la même conductivité thermique, elle est même dans un rapport double. On peut trouver que pour le blanc $D_b = 1,7 \times 10^{-7} m^2.s^{-1}$ *et que pour le jaune* $D_j = 1,0 \times 10^{-7} m^2.s^{-1}$.

Question 13 : Modifier le programme pour tenir compte de cette différence, et estimer à nouveau le temps de cuisson d'un oeuf.

2 Troisième partie : Diffusion de particules

2.1 Modèle de la marche aléatoire

Pour modéliser la diffusion de particules, on utilise un modèle discret : une molécule se trouve à l'instant t dans une cellule, et peut, à l'instant $t + \tau$, se trouver dans une cellule voisine.

Dans un premier temps, on se place dans le cas unidimensionnel : la particule se déplace sur selon l'axe x, et ne peut se trouver, par discrétisation, qu'aux positionx $x_i, i \in \mathbb{Z}$. On repère chacune de ces positions par l'entier relatif i. On choisit comme origine O, la position initiale de la particule.

Une particule située en i à l'instant t, a une chance sur deux (probabilité $0,5$) de se retrouver en $i-1$ à l'instant $t+\tau$, et l'autre chance sur deux (probabilité $0,5$) de se retrouver en $i+1$ à l'instant $t + \tau$: c'est ce qu'on nomme une marche aléatoire, c'est-à-dire qu'on effectue des pas dans une direction aléatoire !

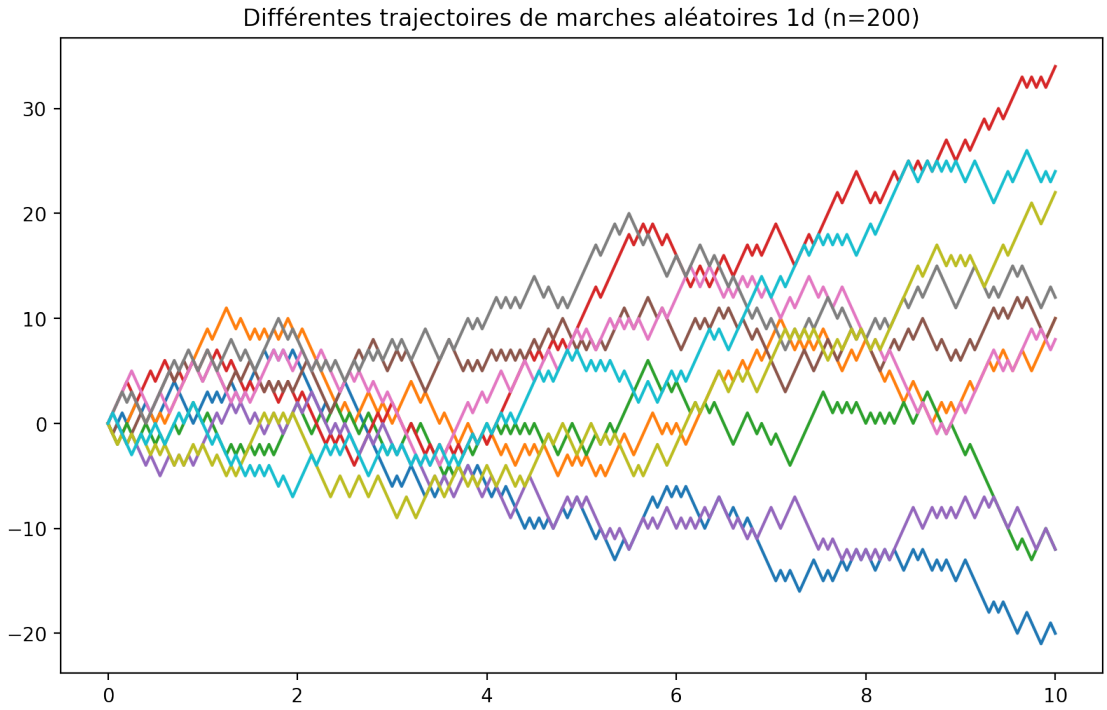
Question 1 : Ecrire une fonction `ma1d(tf,n)` qui retourne la liste T des temps à chaque pas et celle des positions L d'une particule, jusqu'à l'instant t_f. La particule est initialement (à $t_i = 0$) en

O, et a effectué une marche aléatoire de $n+1$ pas. On utilisera la fonction `randint(a,b)`, générant un entier aléatoirement compris entre a et b (compris).

[17]:
```
def ma1d(tf,n):
    tau = tf / n
    ti = 0 # temps initial
    i = 0 # position initiale
    L, T = [i], [ti]
    for k in range(n):
        r = randint(0,1) # tirage aléatoire
        d = 2*(r-0.5) #  déplacement (soit -1, soit 1)
        i += d # nouvelle position
        ti += tau
        L.append(i)
        T.append(ti)
    return T, L
```

Traçons alors différentes trajectoires :

[18]:
```
n = 200
tf = 10

ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.title("Différentes trajectoires de marches aléatoires 1d (n=200)")

for i in range(10):
    T, L = ma1d(tf,n)
    plt.plot(T,L)

plt.show()
```



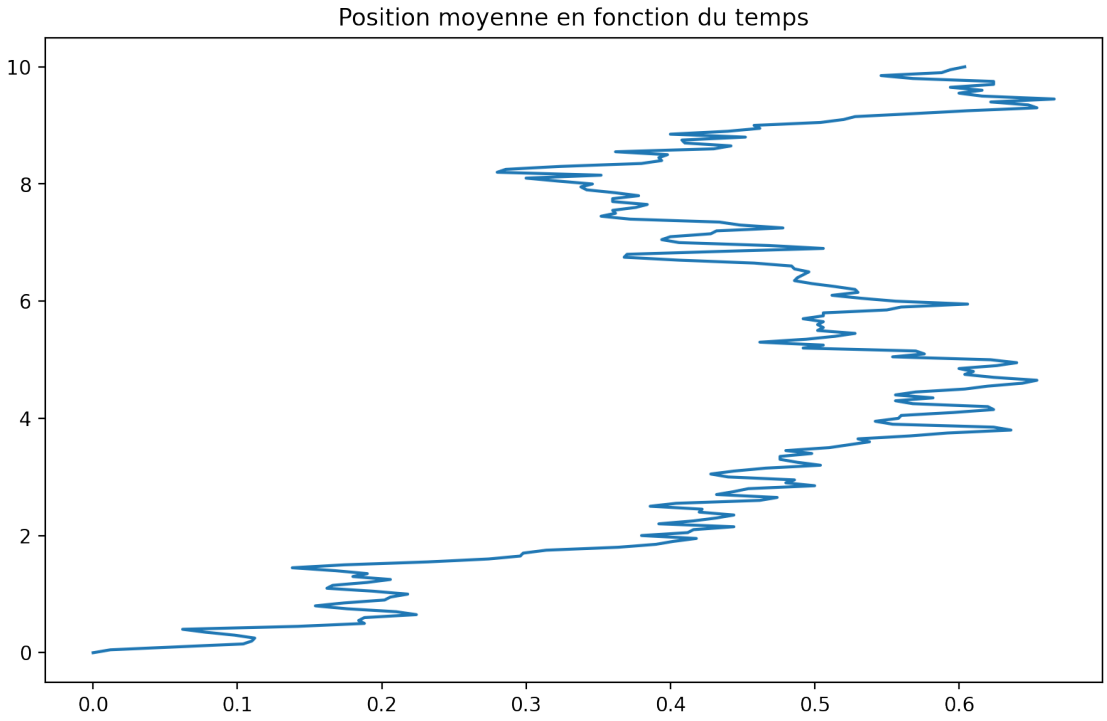
Faisons quelques statistiques sur ces marches : nous allons nous interesser à la position moyenne atteinte après un certain temps. Pour cela, nous fera des moyennes sur N marches différentes.

Question 2 : Ecrire la fonction `M1dpos(tf,n,N)` qui retourne la liste des $n+1$ temps T et la liste des $n+1$ positions moyennes M à chaque pas, en moyennant sur N marches alétoires.

[19]:
```
def M1dpos(tf,n,N):
    M=[0 for i in range(n+1)]
    for j in range(N) :
        T, L = ma1d(tf,n) # on réalise une marche
        for i in range(1,n+1):
            M[i] += L[i]/N # on ajoute la psotion à chaque instant dans la␣
↪liste M, divisé par N pour faire la moyenne
    return T, M
```

Traçons ces positions moyennes en fonction du temps :

[20]:
```
N = 100000
T, M = M1dpos(tf,n,1000)
ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.title("Position moyenne en fonction du temps")
plt.plot(M,T)
plt.show()
```



Diantre ! La position moyenne est vraiment très proche de l'origine, quelquesoit le temps considéré…alors qu'on observait précédemment des trajectoires qui s'éloignaient de cette même origine !

Question 3 : Expliquer ce phénomène !

Correction : A chaque pas, on a la même probabilité d'aller à droite ou à gauche, il est donc normal *qu'en moyenne*, on reste au milieu, donc proche de l'origine. Autrement dit, si on observe une marche (soit une succession de pas) qui, au bout d'un temps T, se retrouve en i, il existe, avec la même probabilité, la marche symétrique qui se retrouve en $-i$ au même instant T : en moyenne, à chaque instant T, on se trouve au milieu, à l'origine.

Pour tenir compte de la symétrie de ces marches, nous allons calculer, à la place de la position moyenne, la distance moyenne à l'origine

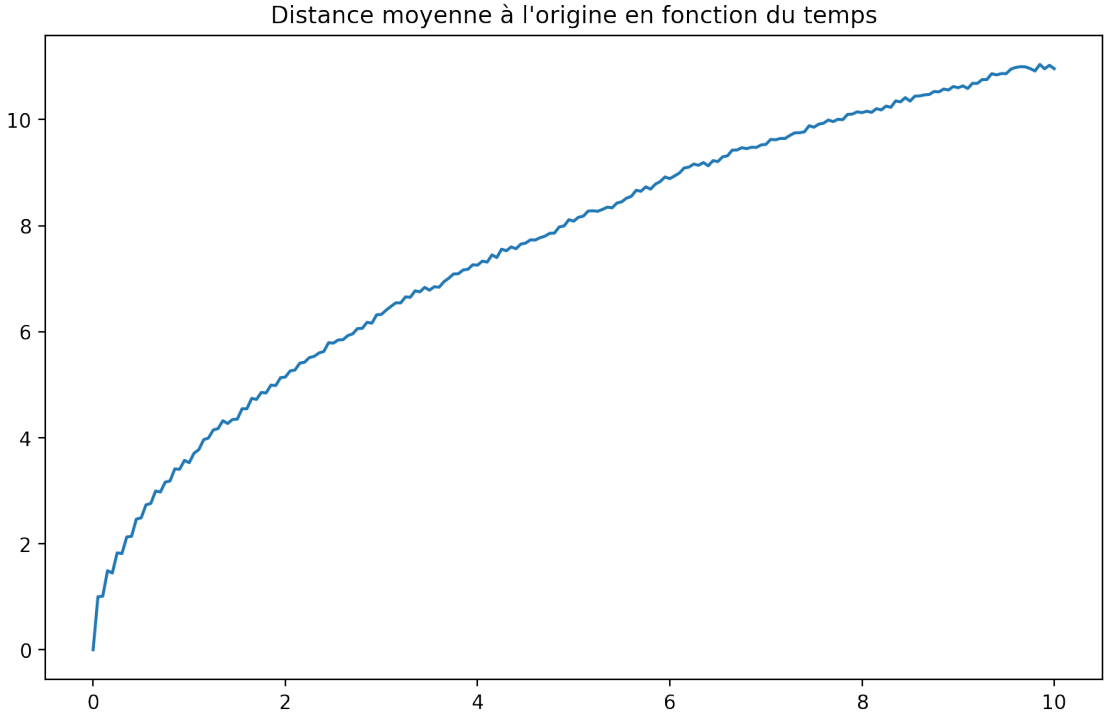
Question 4 : Ecrire une fonction `M1ddist(tf,n,N)` qui retourne la liste des $n+1$ temps T et la liste des $n+1$ distances moyennes à l'origine D à chaque pas, en moyennant sur N marches alétoires.

[21]:
```
def M1ddist(tf,n,N):
    D=[0 for i in range(n+1)]
    for j in range(N) :
        T, L = ma1d(tf,n) # on réalise une marche
        for i in range(1,n+1):
            D[i] += abs(L[i])/N # on ajoute la distance à l'origine à chaque␣
↪instant dans la liste M, divisé par N pour faire la moyenne
```

```
    return T, D
```

Voyons cela :

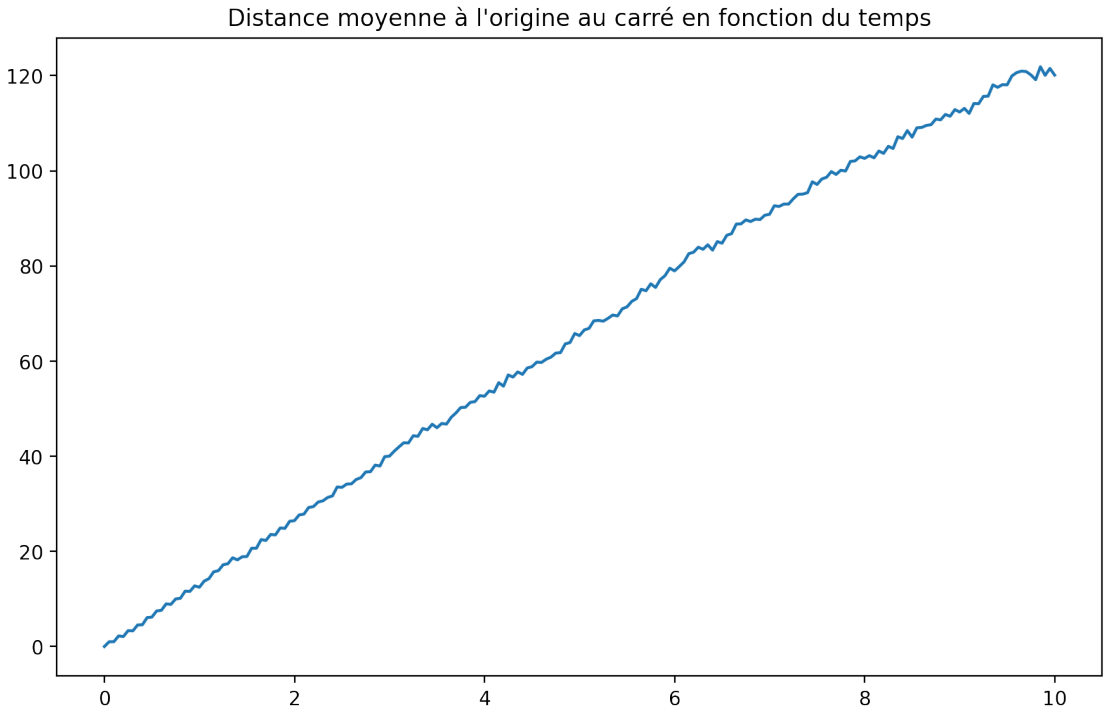
[22]:
```
T, D = M1ddist(tf,n,1000)
ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.title("Distance moyenne à l'origine en fonction du temps")
plt.plot(T,D)
plt.show()
```



Question 5 : Commenter le graphe précédemment obtenu : mathématiquement, comment semble évoluer la distance moyenne à l'origine en fonction du temps ? A quel phénomène cela fait-il penser ?

Correction : On semble avoir $D \propto \sqrt{T}$. Vérifions :

[23]:
```
D2 = [elem**2 for elem in D]
ratio = 1.2 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.title("Distance moyenne à l'origine au carré en fonction du temps")
plt.plot(T,D2)
plt.show()
```



La proportionnalité entre la distance moyenne à l'origine au carré et le temps semble donc être vérifiée. Cette relation est typique des **phénomènes de diffusion**. Vérifions cela théoriquement…

Question 6 : On note $p(x_i, t)$ la probabilité de se trouver en x_i à l'instant t. Exprimer $p(x_i, t+\tau)$ en fonction de $p(x_{i-1}, t)$ et $p(x_{i+1}, t)$.

Pour se trouver en x_i à l'instant t, il fallait soit se trouver en x_{i-1} à l'instant précédent (en t), soit en x_{i+1}. Cependant, dans chacune de ces positions, il faut faire "le bon pas" pour arriver en x_i : ce "bon pas" a pour probabilité 1/2. On a donc :

$$p(x_i, t+\tau) = \frac{1}{2}p(x_{i-1}, t) + \frac{1}{2}p(x_{i+1}, t)$$

Question 7 : On note $\mathrm{d}x = x_{i+1} - x_i$ la longueur d'un pas, et $\mathrm{d}t = \tau$ la durée d'un pas, et on pose $x = x_i$. Ré-écrire l'équation précédente à l'aide de x, t, $\mathrm{d}t$ et $\mathrm{d}x$. Utiliser alors des développements de Taylor pour montrer que $p(x,t)$ obéit à une équation de diffusion, et donner l'expression du coefficient de diffusion.

On a :

$$p(x, t+\mathrm{d}t) = \frac{1}{2}p(x-\mathrm{d}x, t) + \frac{1}{2}p(x+\mathrm{d}x, t)$$

On utilise alors des développements de Taylor à l'ordre 2 en spatial :

$$p(x-\mathrm{d}x, t) = p(x,t) - \mathrm{d}x\frac{\partial p}{\partial x}(x,t) + \frac{\mathrm{d}x^2}{2}\frac{\partial^2 p}{\partial x^2}(x,t) + \mathcal{O}(\mathrm{d}x^3)$$

$$p(x+\mathrm{d}x, t) = p(x,t) + \mathrm{d}x\frac{\partial p}{\partial x}(x,t) + \frac{\mathrm{d}x^2}{2}\frac{\partial^2 p}{\partial x^2}(x,t) + \mathcal{O}(\mathrm{d}x^3)$$

Et un développement à l'ordre 1 en temporel :

$$p(x, t + \mathrm{d}) = p(x, t) + \mathrm{d}t\frac{\partial p}{\partial t}(x, t) + \mathcal{O}(\mathrm{d}t^2)$$

Finalement, on obtient l'équation de diffusion suivante :

$$\frac{\partial p}{\partial t}(x, t) = D_{diff}\frac{\partial^2 p}{\partial x^2}(x, t) \text{ avec } D_{diff} = \frac{\mathrm{d}x^2}{2\mathrm{d}t^2}$$

On a donc vu que pour lorsque la longueur et la durée des pas tendent vers 0, on retrouve un espace "continu", et le mouvement de cette marche suit une loi de diffusion. Vérifions d'autres caractéristiques de cette loi de diffusion.

Retournons au cas discret : nous allons déterminer ce qu'on nomme la distribution des positions en fonction du temps : il s'agit de construire un histogramme représenter le nombre de marches qui, après un temps t, arrive à la position i.

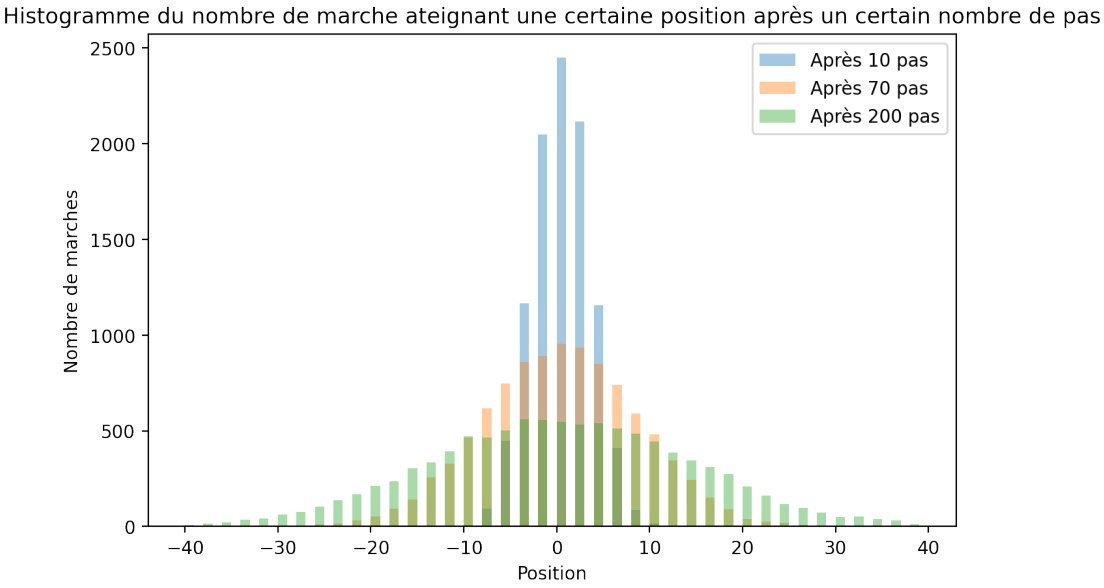
Question 8 : Ecrire une fonction `Dist1d(tf,n,N)` qui retourne la liste T des temps, et un tableau Tab (utiliser `np.array`) dans lequel chaque colonne représente les différentes positions en fonction du temps pour une marche (jusqu'au temps t_f). Chaque nouvelle colonne correspond à une nouvelle marche. Il s'agit donc d'un tableau de N colonnes et $n + 1$ lignes.

[24]:
```
def Dist1d(tf,n,N):
    Tab = np.zeros((n+1,N))

    for j in range(N):
        T, L = ma1d(tf,n) # une marche, qu'on va stocker dans ja jième colonne␣
 ↪du tableau
        Tab[:,j] = L

    return T, Tab
```

Traçons alors l'histogramme :

[26]:
```
n, tf, N = 200, 10, 10000
T, Tab = Dist1d(tf,n,N)

ratio = 1 # ratio de taille entre fig et texte (légende et axes), par défaut 1
plt.figure(figsize=(8*ratio,5*ratio),dpi = 200)
plt.title("Histogramme du nombre de marche atteignant une certaine position␣
 ↪après un certain nombre de pas")
plt.hist(Tab[10,:],bins=[i for i in range(-int(n/5),int(n/5),1)],alpha = 0.4,␣
 ↪label ="Après 10 pas")
plt.hist(Tab[70,:],bins=[i for i in range(-int(n/5),int(n/5),1)],alpha = 0.4,␣
 ↪label ="Après 70 pas")
plt.hist(Tab[200,:],bins=[i for i in range(-int(n/5),int(n/5),1)],alpha = 0.4,␣
 ↪label ="Après 200 pas")
plt.xlabel("Position")
plt.ylabel("Nombre de marches")
plt.legend()
```

```
plt.show()
```



On remarque que la distribution des position "s'étale" avec le temps. On pourrait montrer que cet étalement, caractérisé par la largeur typique, évolue comme la racine carré du temps (encore une signature du processus de diffusion.

Montrons cette propriété… en théorie continue !

Question 9 : Vérifier que la loi de propabilité gaussienne suivante :

$$p(x,t) = \frac{1}{\sigma(t)\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma(t)^2}}$$

vérifie l'équation de diffusion, avec $\sigma(t)$ l'écart-type de cette loi de probabilité. En déduire l'expression de $\sigma(t)$, et commenter.

A suivre :

- fit sur graphe précédent pour sigma(t) ?
- en 2d ?
- vérifier loi de la distance au carré en fonction du temps (voir T&D PC)