OB01-TP

Projet 3 & 4 : Équation de la chaleur à 1 et 2 dimensions

SEGHIR-DELAITRE Célibond

16/10/23

1 Équation de la chaleur à 1 dimension

1.1 Le shema explicite

1.1.1 L'equation et les fonctions

Dans cette première partie, nous souhaitons résoudre cette équation :

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

De plus nous avons ces condtions limites :

A $t = 0$, $T(0, x) = \sin(\pi x)$ avec $0 \leq x \leq 1$,

Pour $t \geq 0$, $T(t, 0) = T(t, 1) = 0$.

On sait d'ailleurs que la solution analytique peux s'écrire $T(t, x) = e^{-\frac{\pi^2}{2}t} \sin(\pi x$

On sait grâce au cours que la formule pour passer d'un instant t à $t + \Delta t$ est :

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

En fesant un petit schema, comme cela :

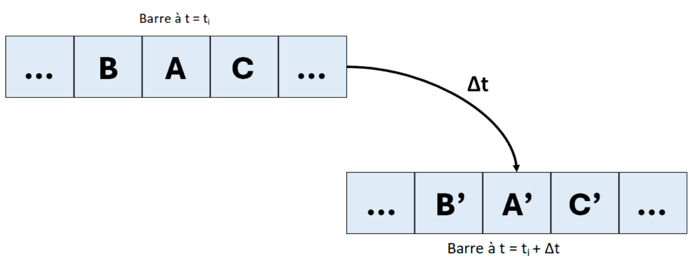

Figure 1: Schema des élements de la barre à t et t+Δt

Ce qui simplifie l'expression en :

$$A' = A + \frac{\Delta t}{\Delta x^2}(C + B - 2A)$$

On n'as plus qu'à parcourir la liste de points pour pour passer d'un état à un autres.

Mais attentions, on doit inclure les conditions limites, ce qui ferait une fonction qui pourrait se résumer à :

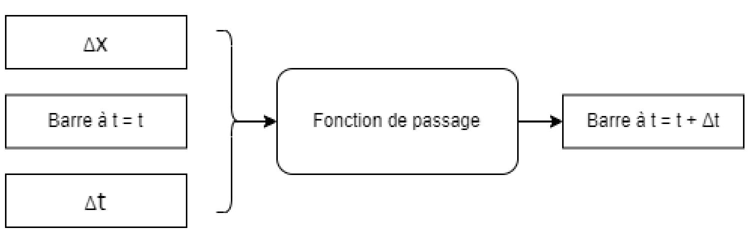

Figure 2: Logigrame fonction

Et pour ce qui est l'interrieur de la foncyion cela devrait ressemble à ça :

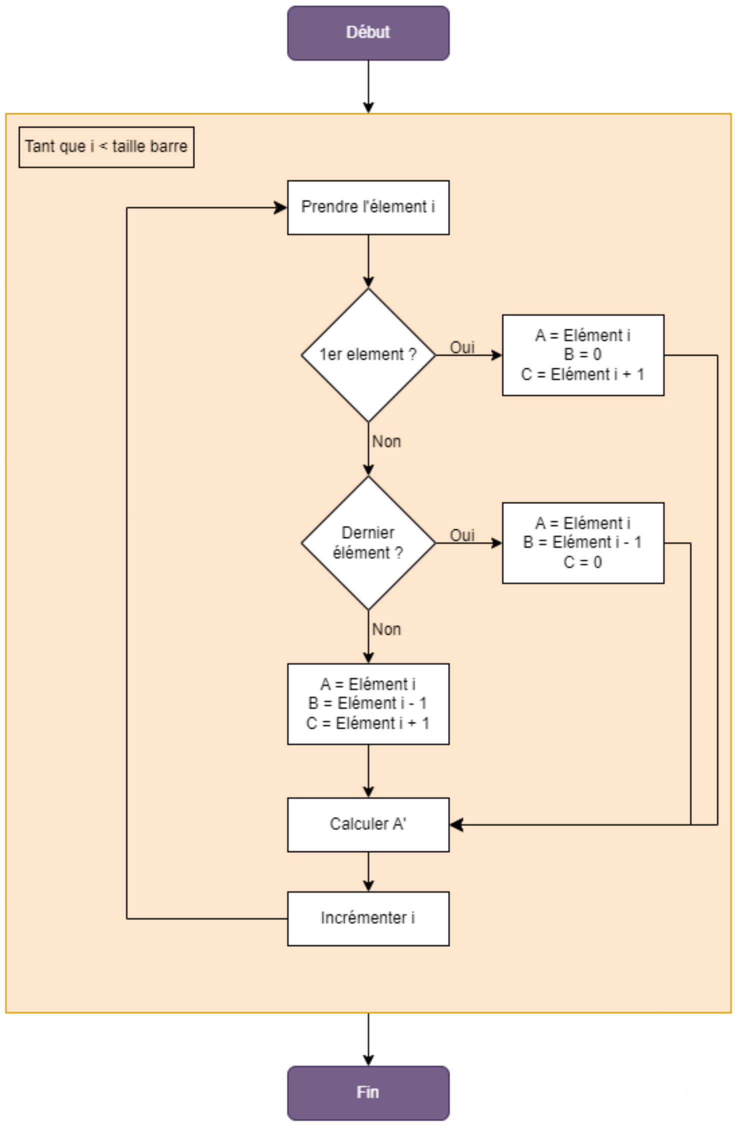

Figure 3: Logigrame intérieur de la fonction

1.1.2 Le code

En codant cela donne cette fonction :

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import *


def function_T(T_precedente, delta_t, delta_x):
    Tn_list = [] #On g n re une liste
    for i in range(len(T_precedente)): #On parcours la barre    l' intant t
        A = T_precedente[i] #On defnit A
        if i == 0: #On verifie les contions limites
            B = 0
            C = T_precedente[i + 1]
            Tn_list.append(A + (delta_t*(C + B - 2*A))/delta_x**2) #On calcule la temrature de catte postion    l'instant t + delta t
        elif i == len(T_precedente) - 1:
            C = 0
            B = T_precedente[i - 1]
            Tn_list.append(A + (delta_t*(C + B - 2*A))/delta_x**2)
        else:
            B = T_precedente[i - 1]
            C = T_precedente[i + 1]
            Tn_list.append(A + (delta_t*(C + B - 2*A))/delta_x**2)
    return Tn_list   #On renvoie la barre    l'instant t + delta t
```

Listing 1: Fonction 1d de t à t + Δt

Ici, on peux voir que j'utilise des listes, mais on aurait pu utiliser des vecteurs avec $np.array()$, en fesant un vector de zeros de la même taille que T$_{p}recedente, avec$np.zeros$_{l}ike$.

On a plus qu'à définir les paramètres tels que Δx, Δt et d'afficher la courbe avec ce code :

```python
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import *


def Ex_1_1():
    T = lambda t, x : np.exp(-t*(np.pi)**2)*np.sin(np.pi*x) #On defnitit la fonction analyte comme  tat  une fonction    deux variables, le temps et la position

    dx = 1e-2 #On defnit un delta x
    dt = 5e-5 #On d fnit un delta t
    n = 1500 #On defnit le nombre d'iteration que l'on veux faire

```

```
    x = np.arange(0, 1+dx, dx) #On cree une liste de x, qui represente les points de la barre avec delta x, on ajoute le +dx  la borne max pour avoir le 1
    T0_list = [T(0, i) for i in x] #On cree la condition initale
    T_list = [T0_list] #On cree une matrcie (ici, une liste de liste) pour stocker toutes les valeures de T  chaque instant
    for i in range(n): #On parcours tous les instant
        T_list.append(function_T(T_list[-1], dt, dx)) #On obtient la barre grace  la barre  l'intant pr c dents

    m = 10 #On utilise une variable pour voir
    plt.plot(x, T_list[m], label='Explicite', color='b') #On trace le resultat de la m thode explicite en bleu
    plt.plot(x, T(m*dt, x), label='Analytique', color='r') #On trace la fonction alaytique
     plt.ylim((0, 1.01)) #On fixe l'axe des ordon es
    plt.xlabel('Position') #On l gende l'axe des absisses et des ordonn es
    plt.ylabel('Temp rature')
    plt.title(f"Temp rature par rapport  x  t = {m}dt") #On met un titre
    plt.legend(loc='best')
    plt.show() #On affiche
```

Listing 2: Fonction 1d de t

On peux avoir ici une représentation des courbes avec m = 0, m = 100, m = 500, m = 1500 :

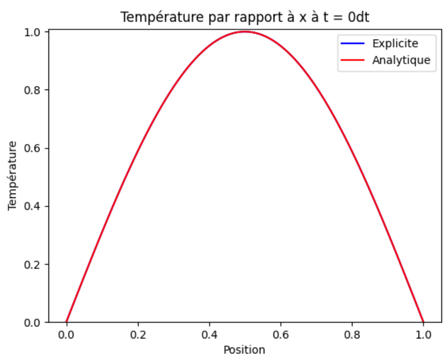

Figure 4: m = 0

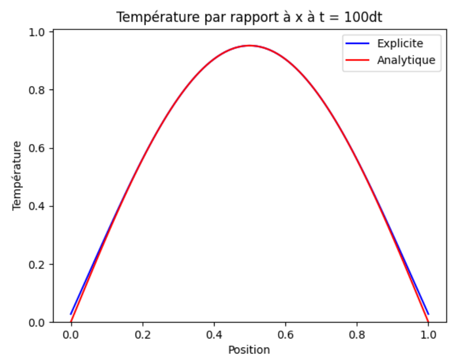

Figure 5: m = 100

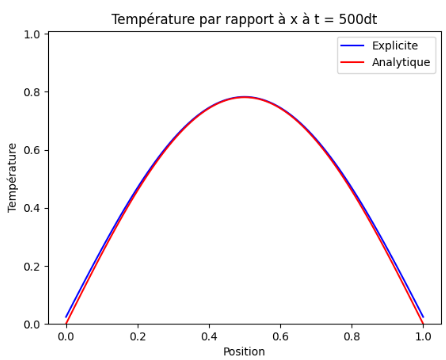

Figure 6: m = 500

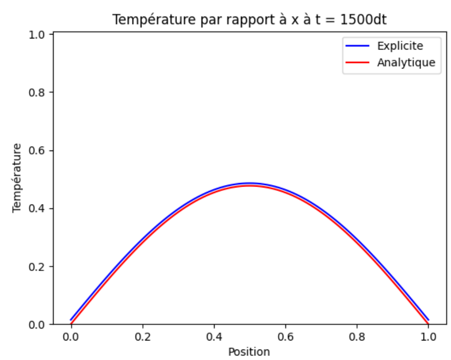

Figure 7: m = 1500

On peux voir que la méthode explicite réprésente plutôt bien la réalité, en même temps, on a choisi un $\Delta x = 1.10^{-2}$ et $\Delta t = 5.10^{-5}$. Ce qui permet d'avoir des resultat très proche.

1.1.3 Les limites

Cepeant s'il ont fait varier Δt en passat par 5.10^{-5}, 1.10^{-4}, 9.10^{-4} et 3.10^{-3}.On a fixé le m à 10 ce qui fait qu'on obtient ces courbes :

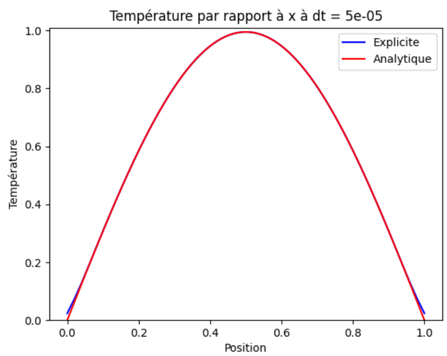

Figure 8: $\Delta t = 5.10^{-5}$

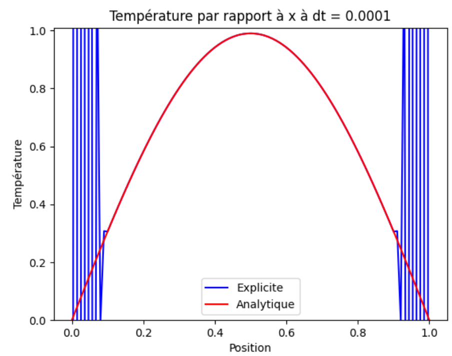

Figure 9: $\Delta t = 1.10^{-4}$

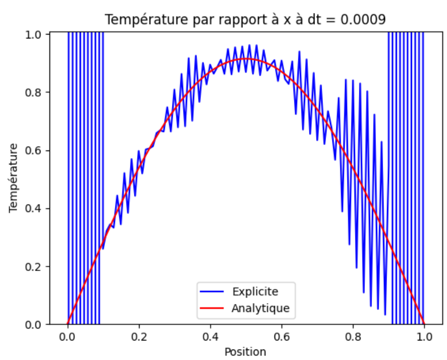

Figure 10: $\Delta t = 9.10^{-4}$

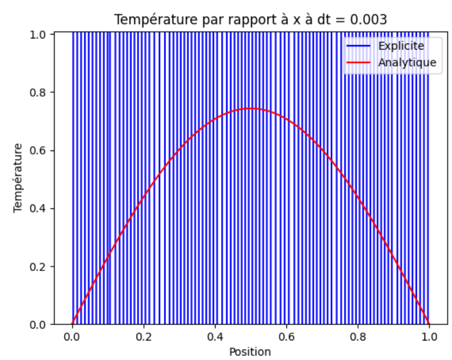

Figure 11: $\Delta t = 3.10^{-3}$

On peux voir que le delta T est un parètre très important à choisir. Et que s'il est trop grand, le modèle explixite divage completement de la prévision analytique.

On fesant varier les Δx en passant par 1.10^{-2}, 7.10^{-3}, 3.10^{-3} et 1.10^{-3}. On peux obtenir ces courbes :

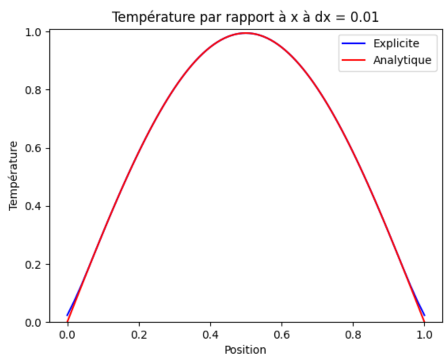

Figure 12: $\Delta x = 1.10^{-2}$

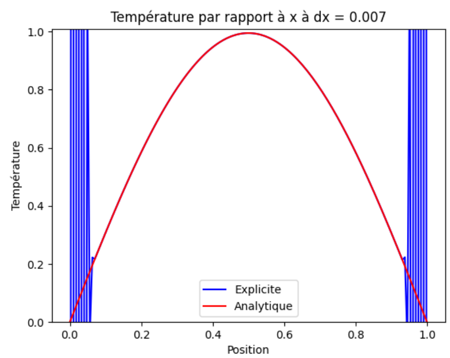

Figure 13: $\Delta x = 7.10^{-3}$

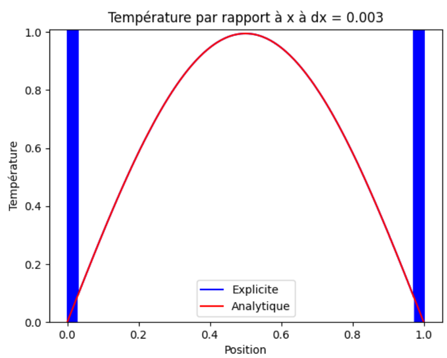

Figure 14: $\Delta x = 3.10^{-3}$

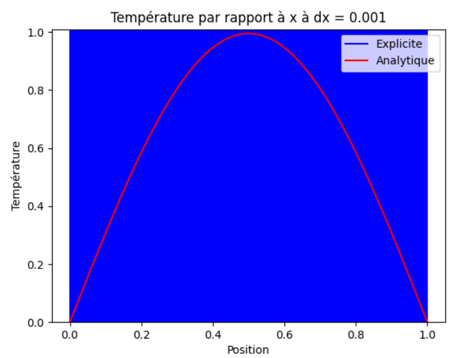

Figure 15: $\Delta x = 1.10^{-3}$

On peux voir que Δx est aussi très important dans la stabilité de ce modèle.

De plus d'après le cours on sait que c'est le rapport $\frac{\Delta x^2}{\Delta t}$ qui doit être supérieur à 2, qui vient de cette formule :

$$\Delta t \leq \frac{\Delta x^2}{2}$$

1.2 Le shema implicite

Pour le shema implicite, ce n'est que du calcul matriciel. Donc une fois qu'on a crée et posée la matrice, on a plus qu'à faire les caluls.

On peux rappeller l'équation :

$$[M]T_{n+1} = T_n$$

$$T_{n+1} = [M]^{-1}T_n$$

Avec $T_n = \begin{pmatrix} T_n^0 \\ T_n^1 \\ \vdots \\ T_n^{I-1} \\ T_n^I \end{pmatrix}$ et $[M] = \begin{pmatrix} 1 & 0 & 0 & \dots & \dots & \dots & 0 \\ -r & 1+2r & -r & \dots & \dots & \dots & 0 \\ 0 & -r & 1+2r & -r & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & -r & 1+2r & -r & 0 \\ \dots & \dots & \dots & \dots & -r & 1+2r & -r \\ \dots & \dots & \dots & \dots & \dots & 0 & 1 \end{pmatrix}$.

Ce qui donne ce code là :

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import *


def Ex_1_3():
    T = lambda t, x : np.exp(-t*(np.pi)**2)*np.sin(np.pi*x)
    n = 100
    etape = 10 #ici m est remplac  par etapes

    nT = n
    nX = n

    dx = 1e-2 #On prends les delta du cours
    dt = 1e-4

    r = dt/(dx**2) #On calcule r
    x_int = np.linspace(0, 1, nX)

    Tn = np.vstack(T(0, x_int)) #On cree le vecteur Tn en vertical
    T_list = [Tn] #On cree la matrice pour toutes les  tapes
    A = [] #On prepare la matrice M, ici nom e A

    for x in range(nX):
```

```
        if x == 0:
            ligne = [1 if i == 0 else 0 for i in range(nT)] #SI on est sur la 1 re  ligne on met une 1 puis que des zeros
        elif x == nX-1:
            ligne = [1 if i == nT-1 else 0 for i in range (nT)]#Si on est sur la derni re ligne on met que des zeros puis un 1.
        else:
            ligne = [] #Sinon on reproduit le shema des lignes de la matrice n.
            for t in range(nT):
                if t == x-1 or t == x + 1:
                    ligne.append(-r)
                elif t == x:
                    ligne.append(1 + 2*r)
                else:
                    ligne.append(0)

        A.append(ligne)

    for i in range(etape): #On cree la matrice avec tous les instants
        T1 = inv(A)@T_list[-1]
        T_list.append(T1)

    #Puis on l'affiche
    plt.plot(x_int, T_list[etape], label='Implicite',  color='b')
    plt.plot(x_int, T(etape*dt, x_int), label='Analytique',  color='r')
    plt.ylim((0, 1.01))
    plt.xlabel('Position')
    plt.ylabel('Temp rature')
    plt.title(f"Temp rature par rapport   x   dt = {etape}dt")
    plt.legend(loc='best')
    plt.show()
```

Listing 3: Fonction 1d de t

On peux optenir ces courbes ci :

On peux voir que la méthode implicite ce superpose parfaitement avec la courbe. Cela vient du fait qu'il n'ait de critère de stabilité avec cette méthode. Qu'importe les Δx ou Δt choisis.

1.3 Pour aller plus loin...

On peux déssiner la temérature en fonction de temps et de la position, grâce à curlf.

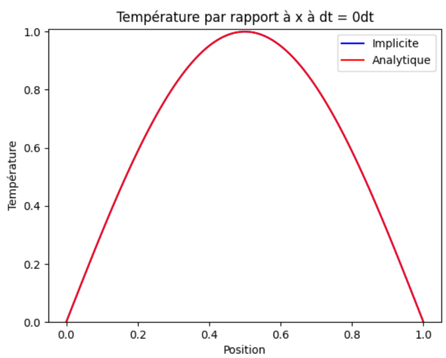

Figure 16: m = 0

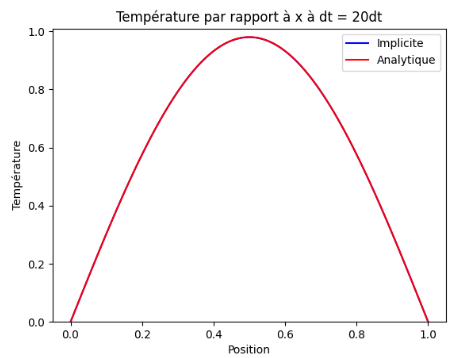

Figure 17: m = 20

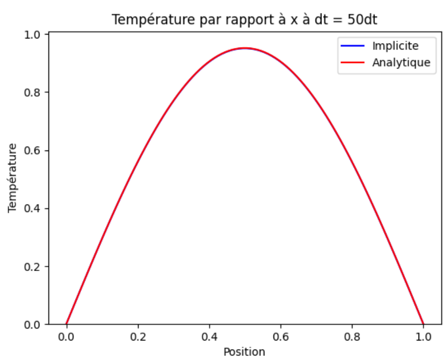

Figure 18: m = 50

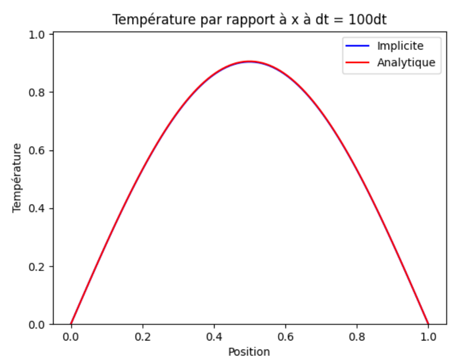

Figure 19: m = 100

2 Équation de la chaleur à 2 dimensions

2.1 D uniforme

Dans la deuxième partie nous souhaitons résoudre l'équation de la chaleur dans un domaine rectangulaire de dimensions (Lx, Ly) qui est donnée par :

$$\frac{\partial T}{\partial t} = D\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

où D représente le coefficient de diffusivité thermique.

De plus nous devons inclure les conditions initiales et limites qui sont :

-Condition initiale : $T(t = 0, x, y) = 0$

-Conditions limites : $T(t, x = 0, Ly/2 - a < y < Ly/2 + a) = 10, \quad T(t, x = Lx, y) = 0, \quad T(t, x, y = 0) = 0, \quad T(t, x, y = Ly) = 0$

Avec a étant une valeure choisie.

Pour cela, on puex utiliser la formule donnée dans le cours qui permet de passer de t à t est :

$$\frac{T_{i,j}^{n+1} - T_{i,j}^{n}}{\Delta t} = \kappa\left(\frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta x^2} + \frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{\Delta z^2}\right)$$

$$T_{i,j}^{n+1} = T_{i,j}^{n} + \Delta t\kappa\left(\frac{T_{i,j+1}^{n} - 2T_{i,j}^{n} + T_{i,j-1}^{n}}{\Delta x^2} + \frac{T_{i+1,j}^{n} - 2T_{i,j}^{n} + T_{i-1,j}^{n}}{\Delta z^2}\right)$$

Ici $\kappa = D$

En utilisant un le même schema que en 2D mais en l'adaptant en 3 comme cela :

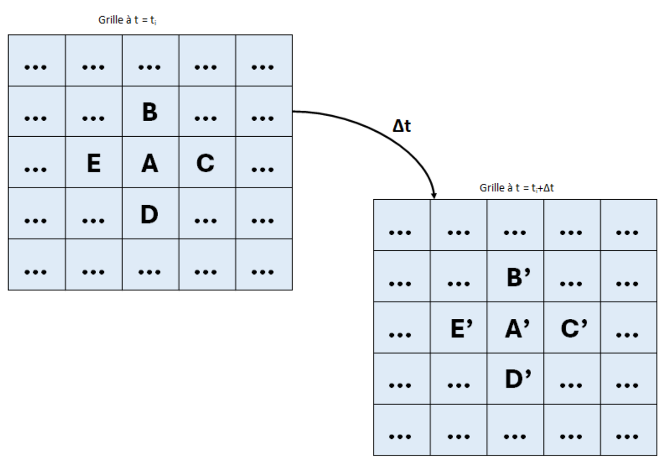

Figure 20: Schema Grille 3D

Ce qui donne cette fontion :

$$A' = A + \Delta t\kappa\left(\frac{C - 2A + E}{\Delta x^2} + \frac{D - 2A + B}{\Delta z^2}\right)$$

Globalement la fonction pour passer de la grille à l'intant t à la grille à l'intant $t + \Delta t$ devrait ressembler à cela :

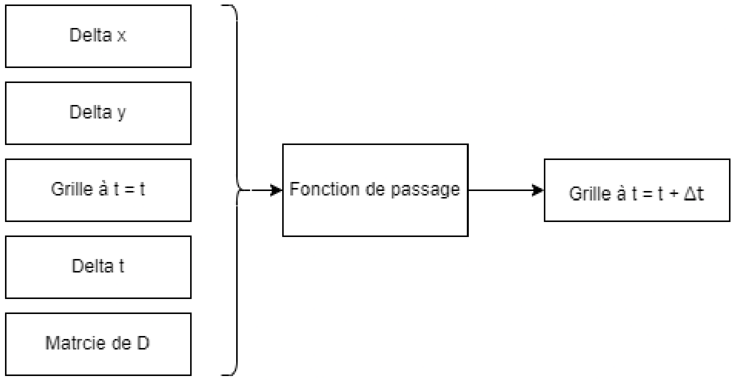

Figure 21: Schema Fonction Grille

Et l'interrieur de la fonction cela ressemble à cela :

Le code entier ressemble à cela :

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


Lx = 20
Ly = 20

dt = 1e-2
dx = 1
dy = 1
a = 7

Max_T = 200
T_global = np.zeros((Max_T, Lx, Ly))


Matrice_coeffs = np.ones((Lx, Ly))*1




Matrice_temp_0 = np.zeros((Lx, Ly))

def afficher_grille_temperature(temperature, m):
    plt.figure(figsize=(6, 4))
    #plt.imshow(temperature, cmap='hot', interpolation='nearest', vmin=0, vmax=1e-1)
    plt.contourf(temperature, cmap='hot', levels=50)
    plt.colorbar(label='Temp rature')
    plt.title(f' volution  de la temp rature  t = {m}dt')
    plt.xlabel('X')
    plt.ylabel('Y')
```

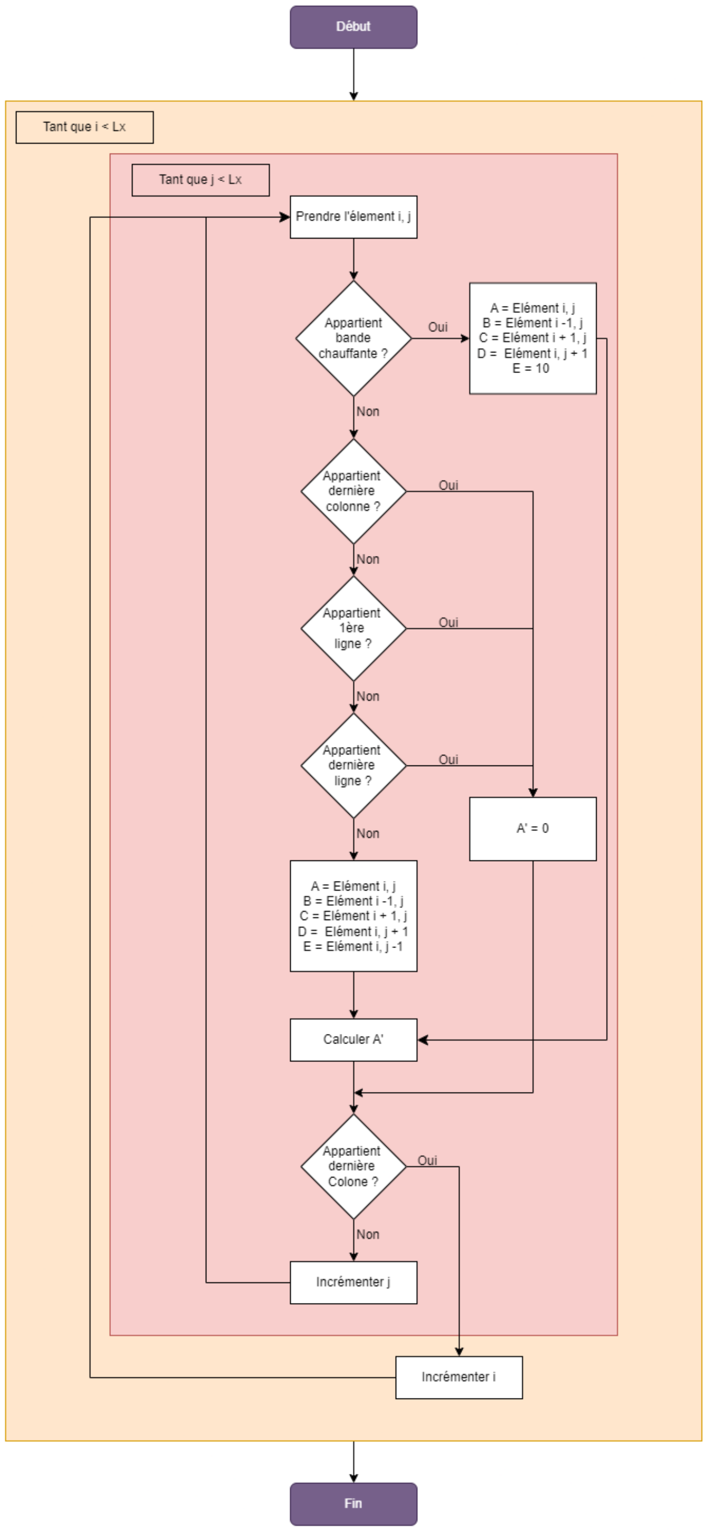

Figure 22: Logigrame fonction de passage

```python

    plt.show()

def prochain_instant(Mat_temp, Mat_coef, dt):
    Mat_temp_resultat = np.zeros_like(Mat_temp)
    for i in range(Ly):
        for j in range(Lx):
            A = Mat_temp[i][j]
            B = 0
            C = 0
            D = 0
            E = 0
            if j == 0 and i > Ly/2 - a and i < Ly/2 + a:
                E = 10
                B = Mat_temp[i - 1][j]
                c = Mat_temp[i][j + 1]
                D = Mat_temp[i + 1][j]
                Mat_temp_resultat[i][j] = Mat_coef[i][j]*dt*((C-2*A + E)/(dx**2)+(D - 2*A + B)/(dy**2)) + A
            elif j == Lx - 1:
                Mat_temp_resultat[i][j]
            elif i ==  0:
                Mat_temp_resultat[i][j]
            elif i == Ly - 1:
                Mat_temp_resultat[i][j]
            else:
                B = Mat_temp[i - 1][j]
                C = Mat_temp[i][j + 1]
                D = Mat_temp[i + 1][j]
                E = Mat_temp[i][j - 1]

                Mat_temp_resultat[i][j] = Mat_coef[i][j]*dt*((C-2*A + E)/(dx**2)+(D - 2*A + B)/(dy**2)) + A

    return Mat_temp_resultat

t = 0
T_global[0] = Matrice_temp_0


while t < Max_T - 1:
    T_global[t + 1] = prochain_instant(T_global[t], Matrice_coeffs, dt)
    t += 1

afficher_grille_temperature(T_global[1], 1)
afficher_grille_temperature(T_global[15], 15)
afficher_grille_temperature(T_global[60], 60)
afficher_grille_temperature(T_global[150], 150)
```

Listing 4: Fonction 2D de t

En utilisant un affiche discret des température avec $plt.imshow()$. C'est une fonction qui permet d'afficher des fonction 2D avec des valeures discretes. En prennant un L_x et L_x de 10, une diffusion de 1 pour toute la matrice, on obtient ces courbes là :

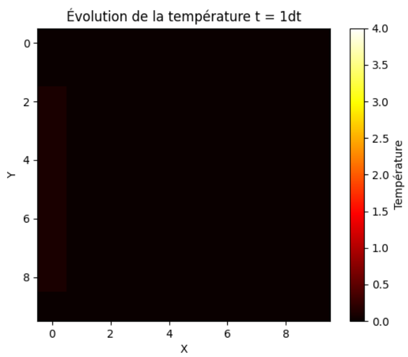

Figure 23: Imshow m = 1

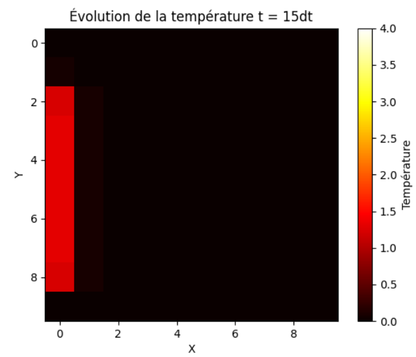

Figure 24: Imshow m = 15

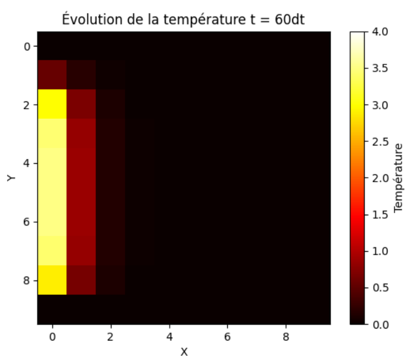

Figure 25: Imshow m = 60

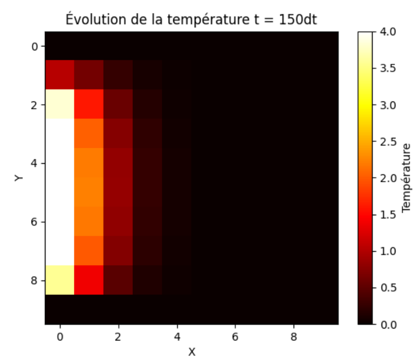

Figure 26: Imshow m = 150

Cependant, si on soufhaite un dégradé de couleur, on puex choisir la fonction $plt.contourf$, qui nous donne le gradiant des températures avec un pas réglable avec le paramètre $levels =$. En choisiant la même configuration mais en réduisant Lx à 5 pour centrer sur la diffusion, on obtient ces courbes :

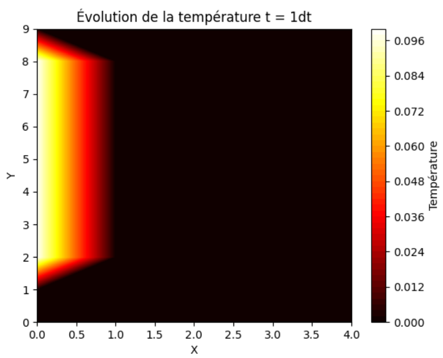

Figure 27: Contourf m = 1

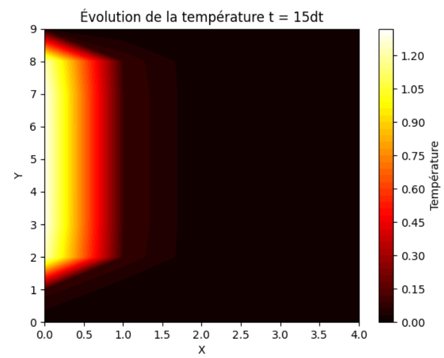

Figure 28: Contourf m = 15

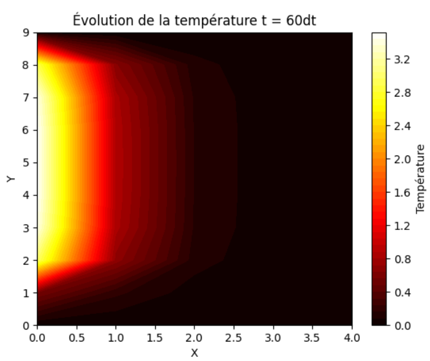

Figure 29: Contourf m = 60

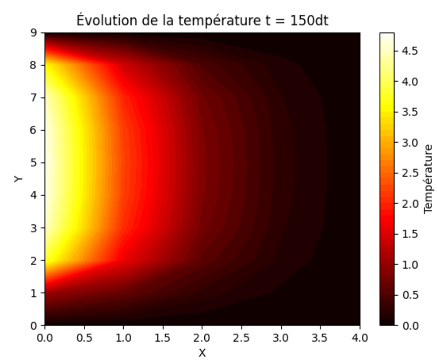

Figure 30: Coutourf m = 150

2.2 D non-uniforme

Pour une changer le coefficient de diffusion à certains endroit de la matrice il faut juste modifier localement la matrcie de coefficients avec les propriétes de *np.array*(). On peux d'ailleurs affciher la matrice de diffusion avec *plt.imshow* et en affichant le matrice de température avec *plt.contourf*:

Ce qui nous donne ce code ci :

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


Lx = 10
Ly = 10

dt = 1e-2
dx = 1
dy = 1
a = 4

Max_T = 200
T_global = np.zeros((Max_T, Ly, Lx))


Matrice_coeffs = np.ones((Ly, Lx))*2


Matrice_coeffs[2:4, 1:3] = 10
Matrice_coeffs[8:10, 1:3] = 0.1

Matrice_temp_0 = np.zeros((Ly, Lx))

def afficher_grille_temperature(temperature, m):
    #plt.imshow(temperature, cmap='hot', interpolation='nearest', vmin=0, vmax=4)
    plt.contourf(temperature, cmap='hot', levels=50)
    plt.colorbar(label='Temp rature')
    plt.title(f' volution  de la temp rature t = {m}dt')
    plt.xlabel('X')
    plt.ylabel('Y')

    plt.show()

def prochain_instant(Mat_temp, Mat_coef, dt):
    Mat_temp_resultat = np.zeros_like(Mat_temp)
    for i in range(Ly):
        for j in range(Lx):
            A = Mat_temp[i][j]
            B = 0
            C = 0
```

```
            D = 0
            E = 0
            if j == 0 and i > Ly/2 - a and i < Ly/2 + a:
                E = 10
                B = Mat_temp[i - 1][j]
                c = Mat_temp[i][j + 1]
                D = Mat_temp[i + 1][j]
                Mat_temp_resultat[i][j] = Mat_coef[i][j]*dt*((C-2*A + E)/(dx**2)+(D - 2*A + B)/(dy**2)) + A
            elif j == Lx - 1:
                Mat_temp_resultat[i][j]
            elif i ==  0:
                Mat_temp_resultat[i][j]
            elif i == Ly - 1:
                Mat_temp_resultat[i][j]
            else:
                B = Mat_temp[i - 1][j]
                C = Mat_temp[i][j + 1]
                D = Mat_temp[i + 1][j]
                E = Mat_temp[i][j - 1]

                Mat_temp_resultat[i][j] = Mat_coef[i][j]*dt*((C-2*A + E)/(dx**2)+(D - 2*A + B)/(dy**2)) + A

    return Mat_temp_resultat

t = 0
T_global[0] = Matrice_temp_0


while t < Max_T - 1:
    T_global[t + 1] = prochain_instant(T_global[t], Matrice_coeffs, dt)
    t += 1



afficher_grille_temperature(T_global[1], 1)
afficher_grille_temperature(T_global[15], 15)
afficher_grille_temperature(T_global[60], 60)
afficher_grille_temperature(T_global[150], 150)

plt.imshow(np.log(Matrice_coeffs), cmap='viridis')
plt.colorbar(label='D')
plt.title(f'Matrice de coefficient de diffusion')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```

Listing 5: Fonction 2D de t

Ce qui donne ces graphiques là :

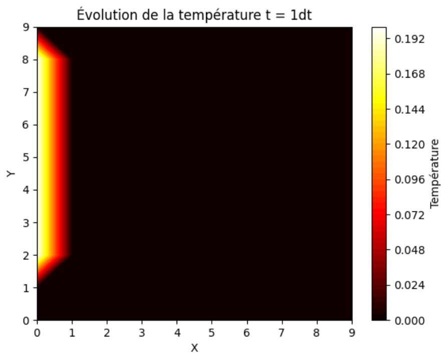

Figure 31: Contourf m = 1, D variable

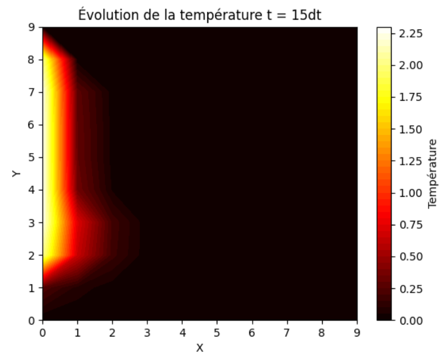

Figure 32: Contourf m = 15, D variable

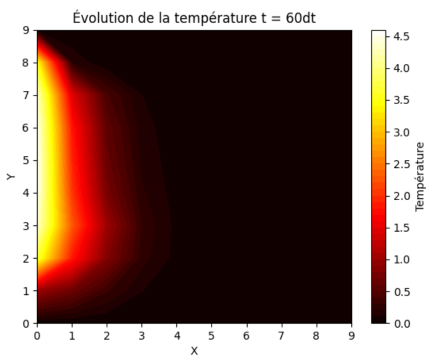

Figure 33: Contourf m = 60, D variable

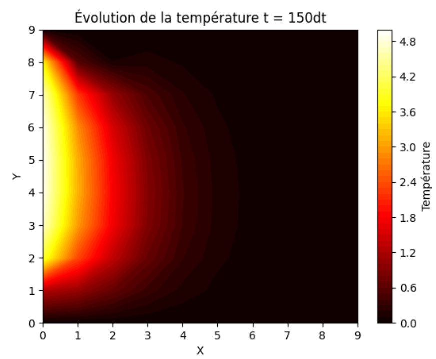

Figure 34: Coutourf m = 150, D variable

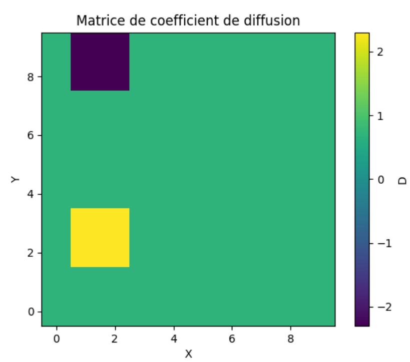

Figure 35: Matrice des D