TP1 – SIMULATION NUMERIQUE

1. METHODE D'EULER

```
#Q1
def Euler(F,a,b,y0,h):
  """Donne une approximation de y(b), où y est une solution de y'=F(t,y) telle que y(a)=y0, en utilisant la méthode d'Euler. """
  y=y0    #si on veut tracer on garde la liste des y  →y=[ y0 ]
  n=int((b-a)/h+1)
  t=a
  for k in range(n):
    y+=h*F(t,y)                                  #y.append( y[k]+h* F( t , y[k] ))
    t+=h
  return y,t
```

Q2.

$$A = \begin{pmatrix} 0 & 1 \\ -\omega_0^2 & -2m\omega_0 \end{pmatrix} \quad et \quad B = \begin{pmatrix} 0 \\ K.\omega_0^2 \end{pmatrix}$$

```
#Q3
#plusieurs tactique
import numpy as np
pas = Tmax / (N - 1)
N = int (Tmax / pas ) + 1
T = np . linspace(0 , Tmax , N)
T = np . arange (0 , Tmax + pas , pas )

def Tableau ( tmax , h ) :
  return ( np . arange (0 , tmax + h , h ) )

T = Tableau (Tmax , pas )

#Q4
def f1(ti, yi) :
  # Cette fonction ne dépend pas directement de ti car elle est linéaire en yi
  return( np.array([yi[1], pow(omega0, 2) * (K - yi[0]) - 2 * m * omega0 * yi[1]]) )
```

Q5

$$Y_{i+1} = Y_i + h \cdot \frac{\mathrm{d}Y(t)}{\mathrm{d}t} + o(h) \approx Y_i + h \cdot F(t, Y(t))$$

```
#Q6
def EulerExplicite(Yini, h, Tmax, F) :
  '''renvoie le tableau de la solution de l'équation dY/dt = F(t, Y).'''
  T = Tableau(Tmax, h)
  SY = np.zeros([len(T), 2]) # tableau de type array à coefficients nuls
  SY[0] = Yini # première ligne du tableau renseignée
  for i in range(len(T) - 1) : # on s'arrête à l'avant dernier terme
    SY[i + 1] = SY[i] + h * F(T[i], SY[i])
  return( SY )
```

Q7. L'erreur diminue

Q8. Boucle for qui est effectue en temps constant (ne dépend pas du pas de discrétisation)

complexité linéaire en *O(Tmax/h)*

Si le pas est divisé par 10 le temps de calcul devrait être multiplié par 10

Q9. Un double est stocké sur 64 bits = 8 octets

Mémoire = 8 octets * 3 listes * Nvaleurs =24000 octets

2. CONGESTION DE L'AUTOROUTE A7

Q1 : Dimensions de C, le tableau de valeurs contenant les concentrations :

Le nombre de points pour la discrétisation en espace est (La//dx)

Le nombre d'instants pour la discrétisation en temps est (Temps//dt)

C est un tableau de dimensions (Temps//dt) lignes par (La//dx) colonnes

Q2 : F onction qui calcule les débits aux différentes positions à un instant

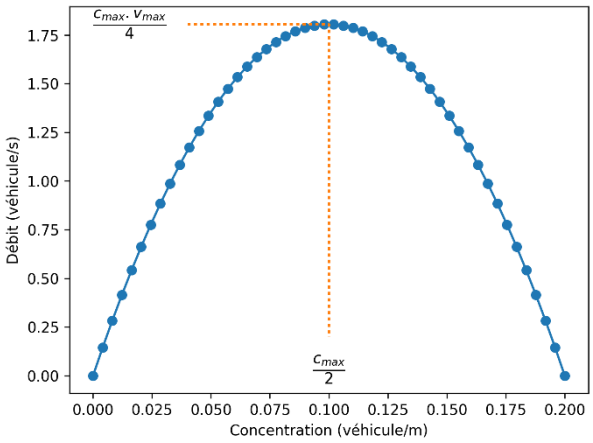
```
def debit(v_max, c_max, C_ligne):
    debit = []
    for concentration in C_ligne:
        vitesse =  v_max * (1 - concentration / c_max)
        debit.append(concentration * vitesse)
    return debit
```

Q3 : Arguments d'entrée de diagramme(v_max,c_max,C_ligne)

v_max	Flottant	m/s	Vitesse maximale
c_max	Flottant	véhicule/m	Concentration maximale
C_ligne	Liste de Flottant	véhicule/m	Concentration à un endroit donné

La relation entre le débit et la concentration est de la forme :

$$Q(C_{i,j}) = v\_max \times (C_{i,j} - \frac{C_{i,j}^2}{c\_max}) \text{ avec } c\_max > C_{i,j}$$



Si l'on trace le diagramme q(t,x) en fonction de c(t,x) on obtiendra une parabole dont l'allure est indépendante du temps. Cependant, la répartition des points sur cette parabole sera fonction des concentrations réelles et donc du temps.

Q4 : Fonction qui permet d'initialiser la première ligne du tableau C.

`C_depart(La,c1,c2,d1,d2,dx)` les arguments d'entrée sont :

La	Flottant	mètre	Vitesse maximale
C1,c2	Flottant	véhicule/m	Concentration maximale
d1,d2	Flottant	mètre	Distance de consigne
dx	Flottant	mètre	Pas de discrétisation

Variable retournée :

C0	Liste de Flottant	véhicule/m	Concentration à tous les endroits

```
def C_depart(La, dx, c1, c2, d1, d2):
    C0 = []
    for l in np.arange(0, La, dx):
        if l < d1 or l > d2:
            C0.append(c1)
        else:
            C0.append(c2)
    return C0
```

Pour $j \in [0, d1//dx] \cup]d2//dx, La//dx]$ alors $C_{0,j} = c1$

Pour $j \in]d1//dx, d2//dx]$ alors $C_{0,j} = c2$

Q5 : Relation de récurrence correspondant au schéma d'Euler « avant »:

$$(1) : \frac{Q_{j+1} - Q_j}{dx} + \frac{C_{i+1,j} - C_{i,j}}{dt} = 0$$

Donc la réponse est : $C_{i+1,j} = C_{i,j} - \frac{Q_{j+1} - Q_j}{dx} \times dt$

Q6 : Fonction qui effectue la résolution complète de C :

```
def resolution(C , dt , dx , c_max , v_max):
    for i in range(1,len(C)):
        Q = debit(v_max , c_max ,C[i-1])
        Q.append(Q[0]) # ajoute le véhicule de droite du dernier
        for j in range(len(C[i])):
            C[i][j] = C[i-1][j] - dt * (Q[j+1] - Q[j]) / dx
```

Q7 : Visualisation des schémas d'intégration :

Schéma Euler « arrière » pour la discrétisation : $C_{i+1,j} = C_{i,j} - \frac{Q_j - Q_{j-1}}{dx} \times dt$

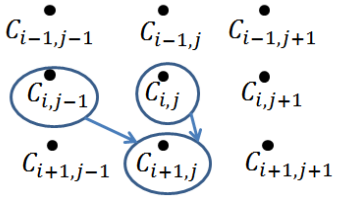
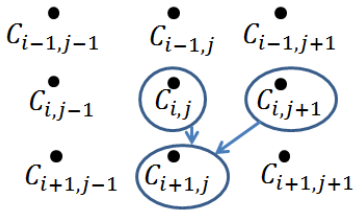


Schéma Euler « avant » pour la discrétisation : $C_{i+1,j} = C_{i,j} - \frac{Q_{j+1} - Q_j}{dx} \times dt$



Q8 : Choix du schéma d'Euler

On choisira la méthode du schéma d'Euler « avant » quand la concentration sera forte, et le schéma d'Euler « arrière » pour des concentrations faibles.

Q9 : Schéma de Lax-Friedriechs :

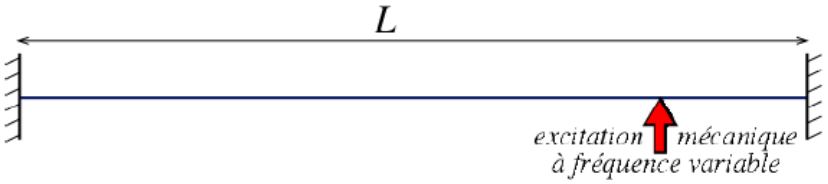
$$C_{i+1,j} = \frac{C_{i,j+1} + C_{i,j-1}}{2} - \frac{Q_{j+1} - Q_{j-1}}{2 \times dx} \times dt$$

```
def resolution_lax(C , dt , dx , c_max , v_max):
    for i in range(1, len(C)):
        Q = debit(v_max , c_max , C[i-1])
        n = len(C[i])
        for j in range(n):
            C_moy = (C[i - 1][j - 1] + C[i - 1][(j + 1) % n]) / 2
            C[i][j] = C_moy - dt * (Q[(j + 1) % n] - Q[j - 1]) / (2 * dx)
```

Q10 : Modification pour prendre en compte le nouveau diagramme fondamental.

Dans la fonction ***resolution*** il faut d'abord appeler la fonction **regression** pour calculer les coefficients a_i. Ensuite, pour chaque ligne de C, le calcul de Q se fera avec cette nouvelle modélisation.

3. CHUTE EXCITATION D'UNE CORDE DE PIANO



Ce mouvement vibratoire est décrit par l'équation suivante :

$$\frac{\sigma}{\rho}.\frac{d^2y(x)}{dx^2} + \omega^2 y(x) = 0$$

Q1.Mettre cette équation sous forme matricielle suivante $([M] + \omega^2[I])(y)$ *en utilisant le schéma des différences finies centrées (dérivée partielle d'ordre 2) avec un pas d'espace* $h = \frac{L}{4}$

Q2.Expliquer la résolution

$$\frac{\partial^2 y}{\partial x^2} = \frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + O(h)$$

Ce qui donne l'équation :

$$\frac{\sigma}{\rho}.\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + \omega^2\, y_i = 0$$

Si h=L/4 on a 5 points avec $y_0 = 0 = y_4$

pour i=1 : $\frac{\sigma}{\rho h^2}(y_2 + y_0 - 2y_1) + (\omega^2.y_1) = 0 \rightarrow \frac{\sigma}{\rho h^2}(y_2 - 2y_1) = -\omega^2.y_1$

pour i=2 : $\frac{\sigma}{\rho h^2}(y_3 + y_1 - 2y_2) + (\omega^2.y_2) = 0 \rightarrow \frac{\sigma}{\rho h^2}(y_3 + y_1 - 2y_2) = -\omega^2.y_2$

pour i=3 : $\frac{\sigma}{\rho h^2}(y_4 + y_2 - 2y_3) + (\omega^2.y_3) = 0 \rightarrow \frac{\sigma}{\rho h^2}(y_2 - 2y_3) = -\omega^2.y_3$

Donc $M = \frac{\sigma}{\rho h^2}.\begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}$

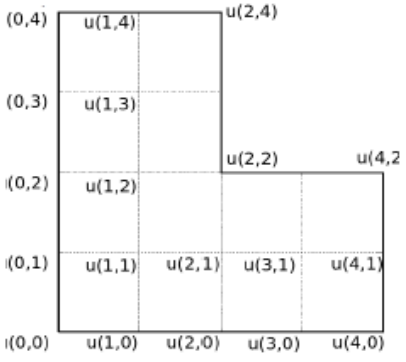
Pour résoudre on cherche les valeurs propres de M (diagonalisation)

4. CONDUCTION STATIONNAIRE 2D

On s'intéresse à la conduction dans un domaine en forme de L (afin de comprendre la conduction du son dans une des parties du piano)

Les lois de la physique nous ont permis d'obtenir l'équation différentielle suivante :

$$\frac{\partial^2 u(x,y)}{\partial x^2} + \frac{\partial^2 u(x,y)}{\partial y^2} = 0$$



Les conditions aux limites sont les suivantes :

- sur la ligne $x = 0 : u(0,y) = y(1-y)$
- sur la ligne $y = 0 : u(x,0) = x(1-x)$
- sur le reste de la frontière : $u = 0$.

La discrétisation avec un pas de ¼ nous donne

Q3.Discrétiser l'équation de Laplace par les différences finies centrées en utilisant un pas constant h = 0.25 en x et en y.

La différence centrée revient à écrire la dérivée partielle à l'ordre 2 :

$$\frac{\partial u(x,y)}{\partial x} = \frac{u_{i-1,j)} - 2u_{(i,j)} + u_{(i+1,j)}}{h^2} + O(h)$$

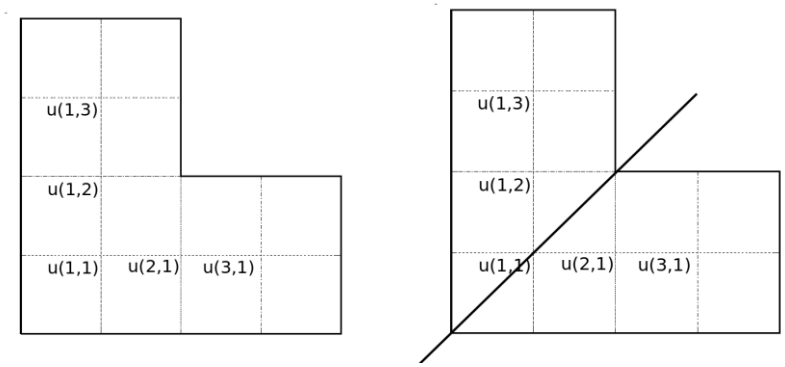
$$\frac{\partial u(x,y)}{\partial y} = \frac{u_{(i,j-1)} - 2u_{(i,j)} + u_{(i,j+1)}}{h^2} + O(h)$$

En remplaçant dans l'équation de Laplace on obtient :

$$\boldsymbol{u_{(i-1,j)} + u_{(i+1,j)} + u_{(i,j-1)} + u_{(i,j+1)} - 4u_{(i,j)} = 0}$$

Q4.En utilisant les conditions aux limites et la symétrie du problème, expliciter le système linéaire (3 × 3) qui régit ce problème. On se ramènera à un problème du type Ax=B

Les conditions aux limites étant connues sur toute la frontière, il nous reste à déterminer :



En utilisant les conditions de symétrie, le domaine, les équations et les conditions aux limites sont échangées en changeant x en y et y en x, ce qui correspond à une symétrie pour la première bissectrice

$$u(x,y) = u(y,x)$$

$$u_{(i,j)} = u_{(j,i)}$$

$$u_{(1,3)} = u_{(3,1)} \quad et \quad u_{(1,2)} = u_{(2,1)}$$

On a donc 3 inconnues dans notre système : $u_{(2,1)}$; $u_{(3,1)}$ et $u_{(1,1)}$

Pour i=j=1, l'équation devient :

$$\boldsymbol{u_{(0,1)} + u_{(2,1)} + u_{(1,0)} + u_{(1,2)} - 4u_{(1,1)} = 0}$$

$$\boldsymbol{\frac{3}{16} + u_{(2,1)} + \frac{3}{16} + u_{(1,2)} - 4u_{(1,1)} = 0}$$

$$\boldsymbol{4u_{(1,1)} - 2\,u_{(2,1)} = \frac{3}{8}}$$

Pour i=2, j=1, l'équation devient :

$$\boldsymbol{u_{(1,1)} + u_{(3,1)} + u_{(2,0)} + u_{(2,2)} - 4u_{(2,1)} = 0}$$

$$\boldsymbol{u_{(1,1)} + u_{(3,1)} + \frac{1}{4} + 0 - 4u_{(2,1)} = 0}$$

$$\boldsymbol{-u_{(1,1)} + 4\,u_{(2,1)} - u_{(3,1)} = \frac{1}{4}}$$

Pour i=3, j=1, l'équation devient :

$$\boldsymbol{u_{(2,1)} + u_{(4,1)} + u_{(3,0)} + u_{(3,2)} - 4u_{(3,1)} = 0}$$

$$\boldsymbol{u_{(2,1)} + 0 + \frac{3}{16} + 0 - 4u_{(3,1)} = 0}$$

$$\boldsymbol{-u_{(2,1)} + 4\,u_{(3,1)} = \frac{3}{16}}$$

On obtient donc la matrice : $\begin{bmatrix} 4 & -2 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{bmatrix} \begin{bmatrix} u_{(1,1)} \\ u_{(2,1)} \\ u_{(3,1)} \end{bmatrix} = \begin{bmatrix} \frac{3}{8} \\ \frac{1}{4} \\ \frac{3}{16} \end{bmatrix}$

Q5. Considérerons la décomposition de A qui amène à la résolution de 2 systèmes linéaires, qui sont moins coûteux à résoudre que le système initial. Décrire ces 2 phases de résolution.

On supprime une des équations, par exemple avec L2*4+L3 et L2*4+L1

On obtient alors

$$\begin{cases} 4\,u_{1,1} - 2\,u_{2,1} = \frac{3}{8} \\ -4.\,u_{1,1} + 15\,u_{2,1} = \frac{19}{16} \end{cases} \qquad \begin{cases} -4\,u_{3,1} + 14\,u_{2,1} = \frac{1}{16} \\ -u_{2,1} + 4\,u_{3,1} = \frac{3}{16} \end{cases}$$