Modélisation

I. **Fonction**

Exemple 1:

def Fonction1(x,t):

——return 4*x**2+6*t

Fonction1(5,2) renvoie 112

Exemple 2:

def Fonction2(x,t):

——return 100,12

Fonction2(5,2) renvoie la liste $100, 12]$

Fonction2(5,2)[0] renvoie 100

Fonction2(5,2)[1] renvoie 12

II. **Faire une somme**

s=0 # on initialise la somme

for i in range(..): # on ajoute les termes dans la boucle for

——s=s+.....

Exemple: $\sum_{i=0}^{i=N} i^3$

s=0

for i in range(N+1):

——s=s+i**3

III. **Euler à 1D**

La méthode d'Euler avec des tableaux à 1D (aussi appelés vecteurs) s'applique aux fonctions qui ne dépendent que d'une variable.

Exemple : un point matériel M de masse m se déplace sur l'axe Oz subit son poids et la force de frottements de norme mkv^2. A l'instant initial M se trouve en $z = h$ sans vitesse initiale. Déterminer z(t) et v(t) entre les instants $t = 0$ et $t = t1$

Principe : on applique la RFD pour trouver la relation de récurrence qui donne la position et la vitesse à l'instant t+tau connaissant la position et la vitesse à l'instant t. Le pas de temps dépend de t1 et du nombre d'itérations N que l'on fait.

$ma = +mg - mkv^2$ soit $a = g - kv^2$

On utilise les DL à l'ordre 1 : on note tau le pas de temps pour la simulation

$v(t + tau) = v(t) + \frac{dv}{dt}.tau = v(t) + a(t).tau$ et $z(t + tau) = z(t) + \frac{dz}{dt}.tau = z(t) + v(t).tau$

d'où les relations de récurrence $v_{i+1} = v_i + a * tau$ et $z_{i+1} = z_i + v_i * tau$

code 1 : on utilise des listes que l'on complète

```
lt=[0] # on entre les conditions initiales
lz=[h]
lv=[0]
tau=t1/N
for i in range(0,N): # on complète les listes en ajoutant n éléments avec les relations de récurrence
    lt.append(lt[i]+tau)
    a=g-k*v**2
    lv.append(lv[i]+a*tau)
    lz.append(lz[i]+lv[i]*tau)
```

code 2 : on crée des tableaux 1D avec N+1 éléments tous égaux à zéros et on modifie les termes des tableaux avec les relations de récurrence

```
lt=np.zeros((N+1))
lz=np.zeros((N+1))
lv=np.zeros((N+1))
tau=t1/N
lt[0] ,lz[0],lv[0]=0,h,0 # on modifie le premier terme des tableaux avec les CI
for i in range(0,N) : # les tableaux contiennent N+1 éléments, on a déjà modifié l'élément i=0, on modifie les éléments pour i=1 jusqu'à N+1
    lt[i+1]=lt[i]+tau
    a=g-k*v**2
    lv[i+1]=lv[i]+a*tau
    lz[i+1]=lz[i]+lv[i]*tau
```

code pour tracer les courbes:

```
plt.plot(lt,lz) # courbe z(t) plt.show()
plt.plot(lt,lv) # courbe z(t) plt.show()
```

Remarque : l'avantage de la liste (code 1) que l'on complète c'est que l'on peut transformer la boucle for en boucle while et donner une condition d'arrêt pour compléter les listes. Par exemple sur la situation précédente, on peut demander de tracer les courbes z(t) et v(t) jusqu'à ce que le point M touche le sol soit jusqu'à ce que $z = 0$.

La boucle for transformée en while dans le code 1 devient :

```
i=0 # on initialise le compteur
while lz[i] > 0:
    lt.append(lt[i]+tau)
    a=g-k*v**2
    lv.append(lv[i]+a*tau)
    lz.append(lz[i]+lv[i]*tau)
    i=i+1 # on incrémente le compteur
```

IV. **Euler à 2D**

La méthode d'Euler avec des tableaux à 2D (matrice) s'applique aux fonctions qui dépendent de deux variables.

Exemple : la résolution de l'équation de diffusion thermique $\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2}$

Principe : on définit un pas de temps dt et un pas d'espace dx et une matrice température telle que $T[i,j] = T(x = i.dx, t = j.dt)$ (température sur la ligne i et la colonne j).

Expression de $\frac{\partial^2 T}{\partial x^2}$ grâce à des DL à l'ordre 2 en dx de $T(x+dx,t)$ et $T(x-dx,t)$:

$T(x+dx,t) = T(x,t) + dx\frac{\partial T}{\partial x} + \frac{dx^2}{2}\frac{\partial^2 T}{\partial x^2}$

$T(x-dx,t) = T(x,t) - dx\frac{\partial T}{\partial x} + \frac{dx^2}{2}\frac{\partial^2 T}{\partial x^2}$

$T(x+dx,t) + T(x-dx,t) = 2T(x,t) + dx^2\frac{\partial^2 T}{\partial x^2}$

d'où $\frac{\partial^2 T}{\partial x^2} = \frac{T(x+dx,t) + T(x-dx,t) - 2T(x,t)}{dx^2}$

Expression de $\frac{\partial T}{\partial t}$ grâce à un DL à l'ordre 1 en dt de $T(x,t+dt)$:

$T(x,t+dt) = T(x,t) + dt\frac{\partial T}{\partial t}$ d'où $\frac{\partial T}{\partial t} = \frac{T(x,t+dt) - T(x,t)}{dt}$.

On remplace dans l'équation de diffusion: $\frac{\partial T}{\partial t} = D\frac{\partial^2 T}{\partial x^2}$ soit $\frac{T(x,t+dt) - T(x,t)}{dt} = \frac{D}{dx^2}(T(x+dx,t) + T(x-dx,t) - 2T(x,t))$.

On en déduit $T(x,t+dt) = T(x,t) + r(T(x+dx,t) + T(x-dx,t) - 2T(x,t))$ avec $r = \frac{Ddt}{dx^2}$.

Ou encore $T[i,j+1] = T[i,j] + r(T[i+1,j] + T[i-1,j] - 2T[i,j])$

Code:

r=dt*D/dx**2

T=np.zeros((Nx+1,Nt+1)) # tableau des températures avec Nx+1 lignes et Nt+1 colonnes avec des 0 partout

T[0,:]=.... # $T[0,j] = T(x = 0, j.dt)$ représente la condition aux limites en $x = 0$ (1ère ligne du tableau)

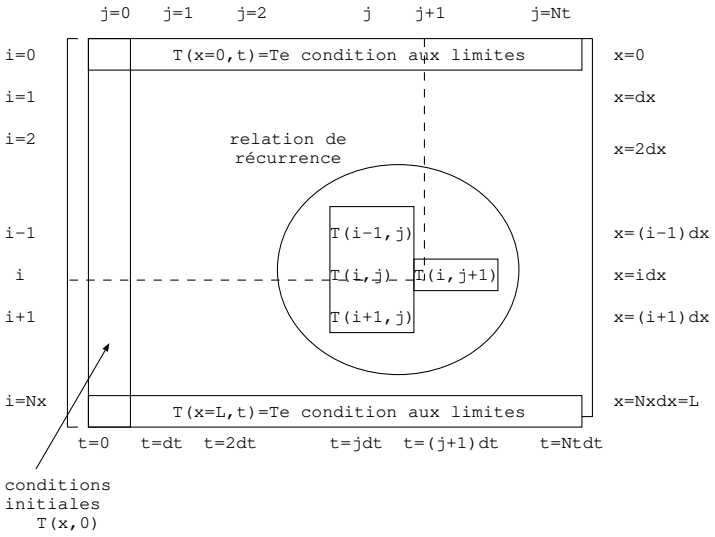
T[Nx,:]=.... # $T[Nx,j] = T(x = Nx.dx, j.dt)$ représente la condition aux limites en $x = Nx.dx$ (dernière ligne du tableau)

T[:,0]=.... # $T[i,0] = T(x = i.dx, t = 0)$ représente la condition initiale (1ère colonne du tableau)

for j in range(0,Nt-1):

——for i in range(1,Nx-1): # on applique la relation de récurrence pour compléter la colonne j+1 connaissant la colonne j

————-T[i,j+1]=T[i,j]+r*(T[i-1,j]+T[i+1,j]-2*T[i,j])



Pour tracer une courbe $T(x_i,t)$ en x_i fixé:

temps=np.linspace(0,Nt*dt,Nt+1)

plt.plot(temps,T[i*dt,:])

Pour tracer une courbe $T(x,t_j)$ à l'instant t_j fixé:

x=np.linspace(0,Nx*dx,Nx+1)

plt.plot(x,T[:,j.dt])