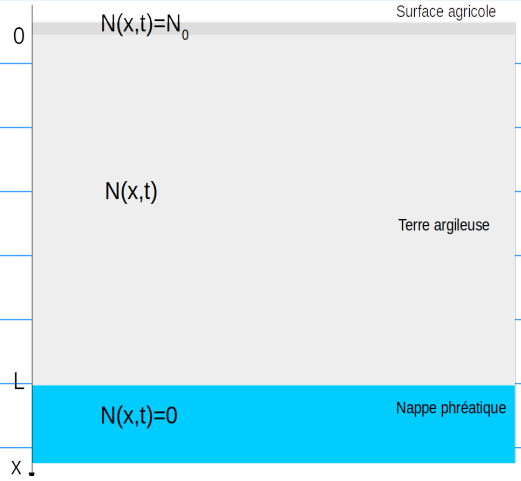
TP CV5 - corrigé

Modélisation du problème.



$N(x,t)$: champ de concentrat° du pesticide.

C.L. : $\forall t,\ N(x=0,t) = N_0$

$N(x=L,t) = 0$

C.I. : à $t=0$ $\begin{cases} N(x=0,t=0) = N_0 \\ N(x\neq 0, t=0) = 0 \end{cases}$

$N(x,t)$ obéit à l'équation de diffusion :

$$\forall t \geq 0,\ \forall\ 0 \leq x \leq L,\quad \frac{\partial N}{\partial t} = D\frac{\partial^2 N}{\partial x^2} \qquad (1)$$

Adimensionnement

On pose $N = N_0 \times N^*$

$x = L \times x^*$ avec $\tau = \frac{L^2}{D}$

$t = \tau \times t^*$

+ C.L. $\forall t^*,\ N^*(x^*=0,t^*) = 1$

$N^*(x^*=1,t^*) = 0$

C.I $N^*(x^*=0,t^*=0) = 1$

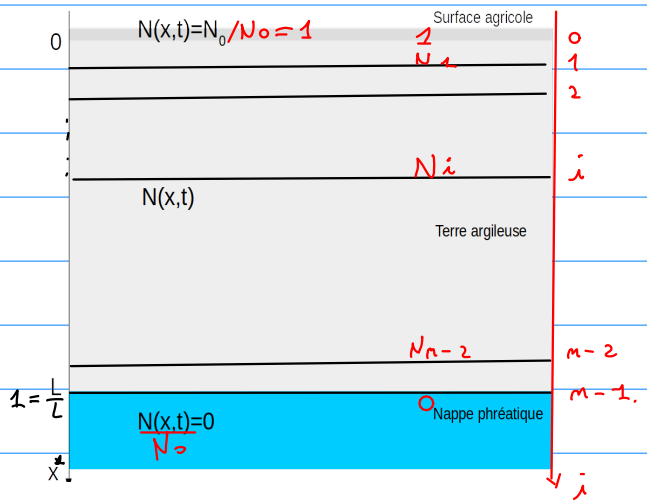
$N^*(x^*\neq 0, t^*=0) = 0$

Alors, $$\forall t^* \geq 0,\ \forall\ 0 \leq x^* \leq 1,\quad \frac{\partial N^*}{\partial t^*} = \frac{\partial^2 N^*}{\partial x^{*2}} \qquad (2)$$

Rem : on omet la notation * par la suite.

Discrétisation.

* Discrétisat° en temps : $t_j = j \times k$, k : pas de temps
* Discrétisat° en espace : $x_i = i \times h$, h : pas d'espace avec $n.h = 1$



Alors, la concentrat° $N(x,t)$ est représentée par le vecteur :

$$N^j = \begin{pmatrix} 1 \\ \vdots \\ N_i^j \\ \vdots \\ 0 \end{pmatrix}$$

où N_i^j est une approximat° de $N(x_i, t_j)$

avec $x_i = ih$

$t_j = jk$.

Schéma numérique

On adopte le schéma de Cranck-Nicholson.

$\forall i,\ \dfrac{\partial N_i^j}{\partial t_j} \simeq \dfrac{N_i^{j+1} - N_i^j}{t_{j+1} - t_i} = \dfrac{N_i^{j+1} - N_i^j}{k}$

$\forall j,\ \dfrac{\partial^2 N_i^j}{\partial^2 x_i} \simeq \dfrac{1}{2}\left[\dfrac{N_{i+1}^{j+1} - 2N_i^{j+1} + N_{i-1}^{j+1}}{h^2} \quad \dfrac{N_{i+1}^{j} - 2N_i^{j} + N_{i-1}^{j}}{h^2} \right]$ (Valeur moyenne de l'approx de la dérivée seconde aux instants t_j et t_{j+1})

Alors, l'équation de diffusion adimensionnée (2) devient :

$$\frac{N_i^{j+1} - N_i^j}{k} = \frac{N_{i+1}^{j+1} - 2N_i^{j+1} + N_{i-1}^{j+1} + N_{i+1}^{j} - 2N_i^{j} + N_{i-1}^{j}}{2h^2}$$

$$\Leftrightarrow N_i^{j+1} - N_i^j = aN_{i+1}^{j+1} - 2aN_i^{j+1} + aN_{i-1}^{j+1} + aN_{i+1}^{j} - 2aN_i^{j} + aN_{i-1}^{j} \quad \text{avec } \boxed{a = \frac{k}{2h^2}}$$

$$\Leftrightarrow \boxed{-aN_{i+1}^{j+1} + (2a+1)N_i^{j+1} - aN_{i-1}^{j+1} = aN_{i+1}^{j} - (1-2a)N_i^{j} + aN_{i-1}^{j}} \quad (3)$$

Cette relation est valable $\forall\ 1 \le i \le n-2$.

Pour $i=0$, $N_0^{j+1} = N_0^j$ (conditions aux limites) (4)

Pour $i=n-1$, $N_{n-1}^{j+1} = N_{n-1}^j$

(3) et (4) s'écrivent matriciellement sur le vecteur N^j :

$\boxed{AN^{j+1} = BN^j}$ avec $A = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ -a & (2a+1) & -a & \cdots \\ 0 & -a & (2a+1) & -a \\ \cdots & -a & (2a+1) & -a \\ 0 & \cdots & 0 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ a & (1-2a) & a & \cdots \\ 0 & a & (1-2a) & a \\ \cdots & a & (1-2a) & a \\ 0 & \cdots & 0 & 1 \end{pmatrix}$

A et B sont tridiagonales.

Résoudre le problème revient à résoudre le système d'équations algébriques linéaires

$$AX = Y \quad \text{avec} \quad X = N^{j+1}$$
$$Y = BN^j$$

Chaque itération permet de calculer N^{j+1} à $t_{j+1} = (j+1)k$ connaissant N^j à t_j.

Implémentation en Python

On travaille avec des tableaux de type ndarray.

(1) Calcul et renvoie des matrices de diffusion A et B.

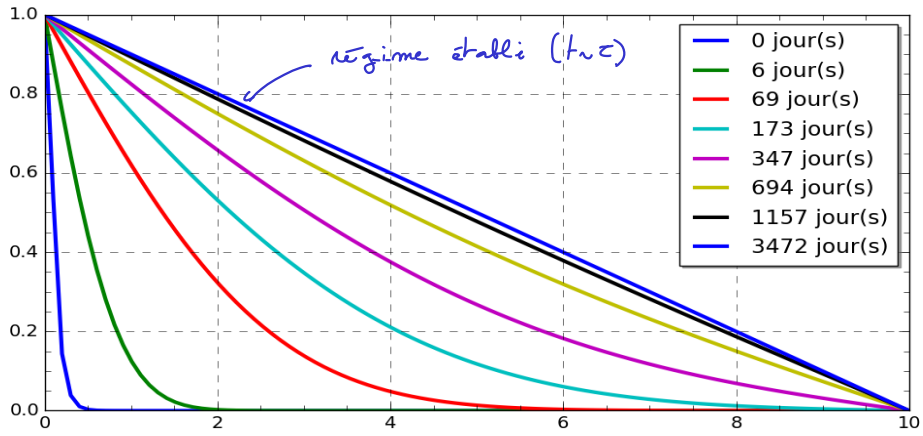
```
def diff_matrix(h,k):
    """
    h: float, space step
    k: float, time step
    A,B: diffusion matrix
    """
    n=int(1/h)  # n doit être entier
    a=k/2*n**2
    A=np.zeros((n,n));B=np.zeros((n,n))
    for i in range(1,n-1):
        A[i,i-1]=-a;A[i,i]=1+2*a;A[i,i+1]=-a
        B[i,i-1]=a;B[i,i]=1-2*a;B[i,i+1]=a
    A[0,0]=1;A[n-1,n-1]=1
    B[0,0]=1;B[n-1,n-1]=1
    return A,B
```

(2) Calcul de BY en exploitant le caractère trigonal de B : fct° de complexité $O(n)$ au lieu de $O(n^2)$ pour le calcul de BY ds le cas général.

```
def prod(A,X):
    """
    Compute Y=AX, assuming that A is tridiagonal
    A= 2d array of size nxn
    X : 1d array of size n
    Y : 1d array of size n, Y=AX
    """
    Y=np.zeros(n)
    Y[0]=X[0];Y[n-1]=X[n-1]
    for i in range(1,n-1):
        Y[i]=A[i,i-1]*X[i-1]+A[i,i]*X[i]+A[i,i+1]*X[i+1]
    return Y
```

(3) Calcul et tracé du champ de concentration à différents instants.

```
"""Calcul et graphe du champ de concentration
a differents instants"""
plt.figure()
A,B=diff_matrix(h,k)
x=np.linspace(0,1,n)
N=np.zeros(n);N[0]=1
p=int(1/k)
for l in range(p+1):
    Y=prod(B,N)
    N=solve(A,Y)
    t=l*k*tau
    if l in [0,int(p/500),int(p/50),int(p/20),int(p/10),int(p/5),int(p/3),p]:
        t=l*k*tau
        plt.plot(x*L,N,label='{} jour(s)'.format(int(t/jour)))
```



Le pesticide diffuse de plus en plus profondément dans le sol.
En régime établi le champ de concentration est affine.

(4) Seuil d'alerte.

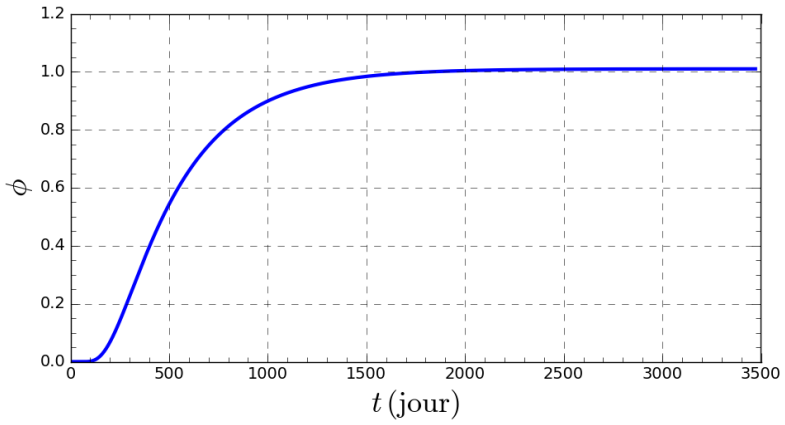
```
def alert(h,k,p,d):
    """
    Determine l'instant a partir duquel la concentration
    N(d) a la profondeur d vaut p*N[0] avec 0<p<1.
    """
    t=0
    A,B=diff_matrix(h,k)
    N=np.zeros(n);N[0]=1
    while N[d]<=p*N[0]:
        Y=prod(B,N)
        N=solve(A,Y)
        t=t+k
    return t,N


"""Alerte : N(H=2)>N(0)*10%"""
H=2
t,N=alert(h,k,0.1,int(H/L*n));plt.plot(N)
print(t*tau/jour)
```

→ 25,7 jours

(5) Evolut° du flux de pesticide

```
def flux(h,k,i,d):
    """s
    Calcul le flux au cours du temps a la profondeur
    i!=0 sur une duree d
    """
    t=np.arange(0,d*k,k)
    J=np.zeros(d)
    A,B=diff_matrix(h,k)
    N=np.zeros(n);N[0]=1
    for l in range(d):
        Y=prod(B,N)
        N=solve(A,Y)
        J[l]=-(N[i]-N[i-1])/h
    return t,J
```

```
"""Flux dans la nappe phreatique"""
plt.figure()
t,J=flux(h,k,n-1,int(1/k))
plt.plot(t*tau/jour,J)
plt.xlabel(r'$t\,(\rm{jour})$')
plt.ylabel(r'$\phi$')
plt.savefig('evolution_flux.png')
```



Le flux de pesticides dans la nappe phréatique ↗ avec le temps
Il est maximal en régime établi.