

Constantes globales

T (notée Temps dans l'énoncé...) = durée de la simulation

L_a = longueur de l'autoroute

L_0 = unité de longueur (grande devant la longueur des véhicules)

$C_{\max} = \frac{1}{L_0}$ = concentration maximale (il y a au plus un véhicule par unité de longueur)

Grandeurs étudiées

$q(t, x)$	débit de véhicules	véhicules par seconde
$v(t, x)$	vitesse moyenne du trafic	mètres par seconde
$c(t, x)$	concentration du trafic	véhicules par mètre

Ces grandeurs sont reliées par la relation suivante.

$$\forall 0 \leq t \leq T, \quad \forall 0 \leq x \leq L_a, \quad q(t, x) = c(t, x) \cdot v(t, x) \quad (1)$$

Équation du trafic

$$\forall 0 \leq t \leq T, \quad \forall 0 \leq x \leq L_a, \quad \frac{\partial q}{\partial x}(t, x) + \frac{\partial c}{\partial t}(t, x) = 0 \quad (2)$$

III.1 Discrétisation

Q6.

Pas d'espace : dx en mètres

Pas de temps : dt en secondes

On décompose la durée de l'étude en n intervalles de temps :

$$T = n \times dt$$

et chaque instant $t_i = i \cdot dt$ est représenté par une ligne du tableau C . Comme t_i varie de $t_0 = 0$ à $t_n = T$ (inclus), il y a $(n + 1)$ instants et donc le tableau C compte $(n + 1)$ lignes.

De même, on décompose la longueur de l'autoroute en p sous-secteurs :

$$L_a = p \times dx.$$

Chaque abscisse $x_j = j \cdot dx$ est représentée par une colonne de C et x_j varie de $x_0 = 0$ à $x_p = L_a$ (inclus), il y a donc $(p + 1)$ colonnes dans le tableau C .

♣ **Attention!** Pour une raison qui m'échappe, l'autre corrigé appelle n , le nombre de colonnes et p , le nombre de lignes. Pourquoi pas ? Mais il est assez étrange de ne pas se conformer à la tradition de l'algèbre linéaire — sans parler des probables erreurs d'indice engendrées par une telle convention.

III.2 Modèle de diagramme fondamental

On utilise ici le modèle de Greenshield.

$$\forall 0 \leq t \leq T, \quad \forall 0 \leq x \leq L_a, \quad v(t, x) = v_{\max} \cdot \left(1 - \frac{c(t, x)}{c_{\max}}\right). \quad (3)$$

Q7. En combinant (1) et (3), on obtient

$$\forall 0 \leq t \leq T, \quad \forall 0 \leq x \leq L_a, \quad q(t, x) = c(t, x) \cdot v_{\max} \cdot \left(1 - \frac{c(t, x)}{c_{\max}}\right). \quad (4)$$

L'annexe suggère de travailler avec des tableaux numpy, ne nous en privons pas !

```
def debit(v_max, c_max, C_ligne):
    Q = C_ligne * v_max * (1 - C_ligne/c_max)
    return Q
```

Variante pour ceux qui ne sont pas encore à l'aise avec numpy.

```
def debit(v_max, c_max, C_ligne):
    Q = []
    for c in C_ligne:
        q = c * v_max * (1 - c/c_max)
        Q.append(q)
    return np.array(Q)
```

Autre variante pour les nostalgiques de la programmation à l'ancienne (on pose ici $r = p + 1$).

```
def debit(v_max, c_max, C_ligne):
    r = len(C_ligne)
    Q = np.zeros(r)
    for j in range(r):
        c = C_ligne[j]
        Q[j] = c * v_max * (1 - c/c_max)
    return Q
```

Q8. Le diagramme fondamental exprime le débit Q en fonction de la concentration C . D'après l'équation (4), un instant t_i étant donné, on exprime les débits $[Q(t_i, x_j)]_{0 \leq j \leq p}$ en fonction des concentrations $[C(t_i, x_j)]_{0 \leq j \leq p}$ au moyen d'un polynôme de degré 2 dont le coefficient dominant est négatif. La courbe est donc une parabole tournée vers le bas.

Les arguments de la fonction diagramme sont ceux qui sont nécessaires pour le calcul des $Q(t_i, x_j)$, c'est-à-dire v_max (flottant, en mètres par seconde); c_max (flottant, en véhicules par mètre) et C_ligne (tableau unidimensionnel de flottants, en véhicules par mètre).

III.3 Résolution de l'équation

La question suivante est assez vague, la réponse que je donne est assez différente de celle proposée dans l'autre corrigé que je vous fournis. Il me semble que ces deux réponses sont également acceptables (seul le point de vue change).

Q9. Pour définir la concentration initiale, il faut connaître les paramètres c_1 et c_2 (concentrations), d_1 et d_2 (distances), ainsi que le pas d'espace dx utilisé pour discrétiser le problème. Tous ces arguments sont des flottants.

- ♣ Un code possible serait alors de la forme suivante.

```
def C_depart(c1, c2, d1, d2, dx):
    p = int(L_a/dx) # L_a est une variable globale
    p1, p2 = ...
    C_ligne = c1 * ones(p+1)
    C_ligne[p1:p2] *= c2/c1
    return C_ligne
```

La valeur retournée est un tableau unidimensionnel de flottants, de même taille que chaque ligne du tableau C .

Pour initialiser la première ligne du tableau C , on devra exécuter l'instruction suivante.

```
C[0] = C_depart(c1, c2, d1, d2, dx)
```

- ♣ Dans un premier temps, on affecte la concentration c_1 à tous les éléments du tableau C_ligne .

Dans un second temps, on affecte la concentration $c_1 \times (c_2/c_1 = c_2$ aux éléments du tableau dont l'indice varie de p_1 (inclus) à p_2 (exclu), c'est-à-dire aux abscisses

$$p_1 \cdot dx \leq x < p_2 \cdot dx.$$

Il convient que les deux abscisses $p_1 \cdot dx$ et $p_2 \cdot dx$ soient aussi proches que possibles des distances d_1 et d_2 . On choisit donc

```
p1, p2 = int(d1/dx)+1, int(d2/dx)+1
```

de telle sorte que

$$p_1 \cdot dx - dx < d_1 \leq p_1 \cdot dx \quad \text{et} \quad p_2 \cdot dx - dx < d_2 \leq p_2 \cdot dx.$$

- ♣ Les indices p_1 et p_2 doivent être des entiers, c'est pourquoi on les convertit en entiers avec la fonction `int`.

Q10. Avec le schéma d'Euler "avant", l'équation (2) devient

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

c'est-à-dire la propriété 2 de l'énoncé :

$$C_{i+1,j} = C_{i,j} - \frac{dt}{dx} \cdot (Q_{j+1} - Q_j). \quad (5)$$

L'indice i varie de 0 à n (inclus), on applique donc la relation de récurrence (5) pour $0 \leq i < n$.

Pour chaque valeur de i , l'indice j varie de 0 à p (inclus). L'énoncé impose une condition aux limites, qui se traduit par le fait que, si $j = p$, alors $Q_{j+1} = Q_0$.

Q11. L'énoncé impose une autre forme d'initialisation que celle que j'ai choisie. Trop tard, je ne change pas ma réponse à **Q9**. (Sérieusement, qui va lire de bout en bout un énoncé aussi long avant de se lancer dans l'étude des questions ?)

On commence par récupérer la taille du tableau C .

On itère les calculs pour $0 \leq i < n$ (cf **Q10.**) en calculant d'abord la liste $[Q_{i,j}]_{0 \leq j \leq p}$. On applique ensuite le schéma d'Euler (5) pour déduire les concentrations $[C_{i+1,j}]_{0 \leq j \leq p}$ des concentrations $[C_{i,j}]_{0 \leq j \leq p}$ et des débits $[Q_j]_{0 \leq j \leq p}$ tout en tenant compte de la condition aux limites.

```
def resolution(C, dt, dx, c_max, v_max):
    s = C.shape
    n, p = s[0]-1, s[1]-1
    for i in range(n):
        Q = debit(v_max, c_max, C[i])
        for j in range(p+1):
            if (j==p):
                q_suivante = Q[0]
            else:
                q_suivante = Q[j+1]
            C[i+1, j] = C[i, j] - dt/dx*(q_suivante-Q[j])
```

III.4 Étude des solutions, modification du schéma

Q12. Il s'agit toujours de traduire l'équation (2) et de calculer le tableau des concentrations $(C_{i+1,j})_{0 \leq j \leq p}$ en fonction des débits $(Q_j)_{0 \leq j \leq p}$.

On conserve le schéma temporel

$$\frac{\partial c}{\partial t} = \frac{C_{i+1,j} - C_{i,j}}{dt}.$$

On compare le schéma spatial "avant"

$$\frac{\partial q}{\partial x} = \frac{Q_{j+1} - Q_j}{dx}$$

au schéma spatial "arrière"

$$\frac{\partial q}{\partial x} = \frac{Q_j - Q_{j-1}}{dx}.$$

Avec le schéma "avant", la relation de récurrence était :

$$\forall 0 \leq i < n, \forall 0 \leq j \leq p, \quad C_{i+1,j} = C_{i,j} - \frac{dt}{dx} \cdot (Q_{j+1} - Q_j)$$

avec la convention $Q_{p+1} = Q_0$.

Avec le schéma "arrière", la relation de récurrence devient :

$$\forall 0 \leq i < n, \forall 0 \leq j \leq p, \quad C_{i+1,j} = C_{i,j} - \frac{dt}{dx} \cdot (Q_j - Q_{j-1}).$$

Par défaut, Python interprète Q_{-1} comme Q_p (= le dernier élément du tableau), il n'y a donc plus de précaution particulière à prendre dans ce cas.

Q13. Je ne vois pas en quoi la discussion présentée dans l'énoncé et reprise par le corrigé résulte du schéma de **12**.

Q14. L'équation (2) devient cette fois

$$\forall 0 < j < p, \forall 0 \leq i < n, \quad C_{i+1,j} = \frac{C_{i,j-1} + C_{i,j+1}}{2} - \frac{dt}{2 dx} \cdot (q_{j+1} - q_{j-1}). \quad (6)$$

Pour $j = 0$, on doit remplacer $C_{i,j-1}$ par $C_{i,p}$ et q_{j-1} par q_p : ce n'est pas une difficulté, Python effectue cette opération automatiquement (puisque $L[-1]$ désigne aussi le dernier élément de la liste L).

Pour $j = p$, on doit remplacer $C_{i,j+1}$ par $C_{i,0}$ et q_{j+1} par q_0 . Cette fois, Python ne se charge pas tout seul de cette substitution.

III.5 Amélioration

Q15. Pour calculer le tableau Q , il faut remplacer l’appel à la fonction `debit` (définie en **Q7**) par le code suivant (ou par un code analogue).

```
a0, a1, a2, a3 = regression(q_exp, c_exp)
Q = a0 + a1*C[i] + a2*C[i]**2 + a3*C[i]**3
```
