CORRECTION SUJET INFORMATIQUE CCP PSI – 2017

Partie I : Exploitation des données de mesure :

Q1 : Requête SQL qui renvoie les données de comptage :

SELECT (id_comptage, date, voie, q_exp, v_exp)
FROM COMPTAGES **JOIN** STATIONS **ON** COMPTAGES.id_station=STATIONS.id_station
WHERE nom='M8B' ;

Q2 : Débit correspondant à la somme des débits de chaque voie

SELECT date, SUM(q_exp) **FROM** COMPTAGES_M8B
GROUP BY date;

Q3 : Fonction qui permet d'afficher le nuage de points du diagramme fondamental

```
def trace(q_exp, v_exp):
    nb_mesures = len(q_exp)
    c_exp = [0] * nb_mesures
    for i in range(nb_mesures):
        c_exp[i] = q_exp[i] / v_exp[i]
    plt.plot(c_exp, q_exp, 'o')
    plt.xlabel('Concentration (véhicule/km)')
    plt.ylabel('Débit (véhicule/h)')
    plt.show()
```

Partie II : Estimation de l'état de congestion

Q4 : Recherche de la médiane

Il s'agit d'un tri par insertion, le choix des lignes manquantes est la proposition 2.

```
def congestion(v_exp):
    nbmesures=len(v_exp)
    for i in range(nbmesures):
        v=v_exp[i]
        j=i
        while 0<j and v<v_exp[j-1]:

            v_exp[j]=v_exp[j-1]      # ligne manquante
            j=j-1                    # ligne manquante

        v_exp[j]=v
    return v_exp[nbmesures//2]
```

La complexité dans le pire des cas quand la liste est rangée en ordre décroissant : $O(n^2)$

La complexité dans le meilleur des cas quand la liste est rangée en ordre croissant : $O(n)$

Ce n'est pas l'algorithme le plus rapide, mais étant donné que la mesure de la congestion se fait sur un intervalle de temps réduit, la taille de la liste n'est pas très grande, donc le critère de rapidité n'est pas critique.

Q5 : Conclusion sur la valeur de la médiane

Avec une valeur médiane de vitesse de 30 km/h, cela indique que largement plus de la moitié des mesures sont inférieures à 40 km/h. La situation est donc congestionnée plus de 50% du temps.

Partie III : Élaboration d'une première simulation du trafic routier par la mécanique des fluides

Q6 : Dimensions de C, le tableau de valeurs contenant les concentrations :

Le nombre de points pour la discrétisation en espace est (La//dx + 1)

Le nombre d'instants pour la discrétisation en temps est (Temps//dt + 1)

C est un tableau de dimensions (Temps//dt + 1) lignes par (La//dx + 1) colonnes

Q7 : F onction qui calcule les débits aux différentes positions à un instant

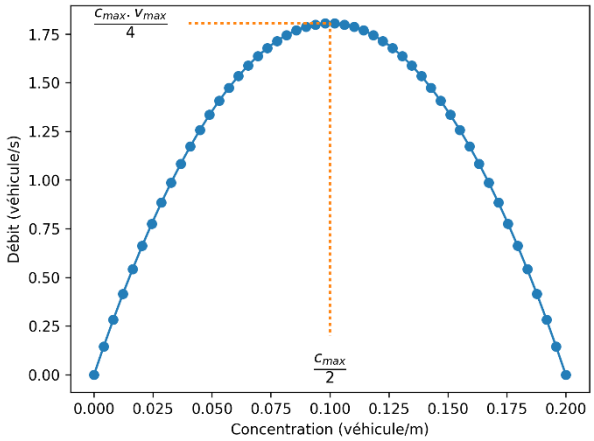
```
def debit(v_max, c_max, C_ligne):
    debit = []
    for concentration in C_ligne:
        vitesse =  v_max * (1 - concentration / c_max)
        debit.append(concentration * vitesse)
    return debit
```

Q8 : 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 :

$$Q\left(C_{i,j}\right)=v_{max}\times\left(C_{i,j}-\frac{C_{i,j}^{2}}{c_{max}}\right) \quad \text{avec} \quad c_{max}>C_{i,j}$$



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.

Q9 : 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

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

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

Q10 : 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 2 : $C_{i+1,j} = C_{i,j} - \frac{Q_{j+1} - Q_j}{dx} \times dt$

Q11 : 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
```

Q12 : Visualisation des schémas d'intégration :

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

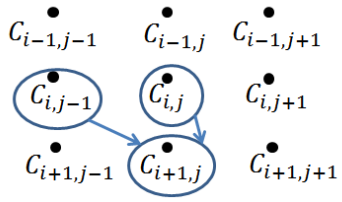
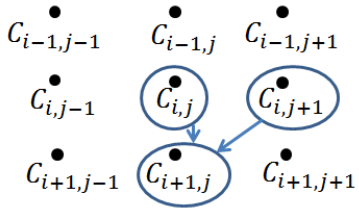


Schéma Euler « avant » pour la discrétisation : $C_{i+1,j} = C_{i,j} - \frac{Q_{j+1} - Q_j}{dx} \times dt$



Q13 : 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.

Q14 : 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)
```

Q15 : 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.

Q16 : Justification du choix de la valeur du pas d'espace
La taille d'une cellule doit être supérieure à celle d'un véhicule (entre 4 et 5m) et inférieure à celle de deux véhicules.
Détermination de v_max :

vitesse_maxi * pas_t / pas_x = 130*1000/3600 * 1.2/7.5 = 5.77 cellules /pas de temps

Donc **v_max = 6** cellules / pas de temps

Partie IV : Deuxième simulation du trafic routier : simulation de Nagel et Schreckenberg (NaSch)

Q17 : Fonction qui calcule le vecteur vitesse_suivante

```
def maj(Route, Vitesse, p, vmax, i):
    Vitesses_suivantes = list(Vitesse[i]) # copie des vitesse
    n = len(Route[0])
    for x in range(n):
        if Route[i][x] == 1:
            if Vitesses_suivantes[x] < vmax:
                Vitesses_suivantes[x] += 1 # accélération
            d = distance(Route, i, x)
            if Vitesses_suivantes[x] >= d :
                Vitesses_suivantes[x] = d - 1  # décélération
            if (Vitesses_suivantes[x] > 0) and (random() < p):
                Vitesses_suivantes[x] -= 1    # facteur aléatoire
    return Vitesses_suivantes
```

Q18 : Fonction qui permet de déterminer les valeurs de Vitesses[i + 1, :] et de Route[i + 1, :].

```
def deplacement(Vitesse, Route, Vitesses_suivantes, i):
    n = len(Route[0])
    for x in range(n):
        if Route[i][x] == 1:
            x_suivant = (x + Vitesses_suivantes[x]) % n
            Route[i + 1][x_suivant] = 1
            Vitesse[i + 1][x_suivant] = Vitesses_suivantes[x]
    return Route, Vitesse
```

Q19 : Les zones plus foncées représentent la forte concentration des véhicules, les zones « reculent » au cours du temps ce qui représente le fait que les voitures qui arrivent sur l'embouteillage freinent en amont. En sortie d'embouteillage, les voitures accélèrent (légèrement). La distance entre les points est faible.
Paramètres : modification du comportement des conducteurs (météo, nuit, visibilité, travaux...), gestion des différentes voies, prise en compte de la diversité des voitures (camions, caravanes, voitures rapides...)