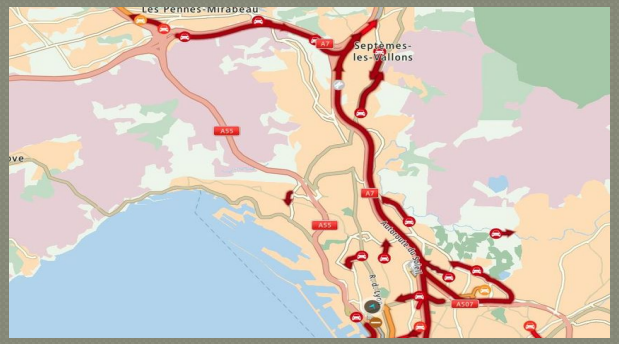
Thème 2018-2019 :
Transport

Scei:18005

Etude de la capacité et de la congestion sur le réseau MARIUS: Autoroute A7 en France



Objectifs:

Le premier objectif est d'établir le diagramme fondamental (tracé du débit en fonction de la concentration) .

Le deuxième objectif est de mettre en place une simulation
numérique adaptée au tronçon étudié à partir d'un modèle basé sur la mécanique des fluides afin de tester l'approche physique.

Plan:

I. Choix d’un modèle

II. Etude théorique

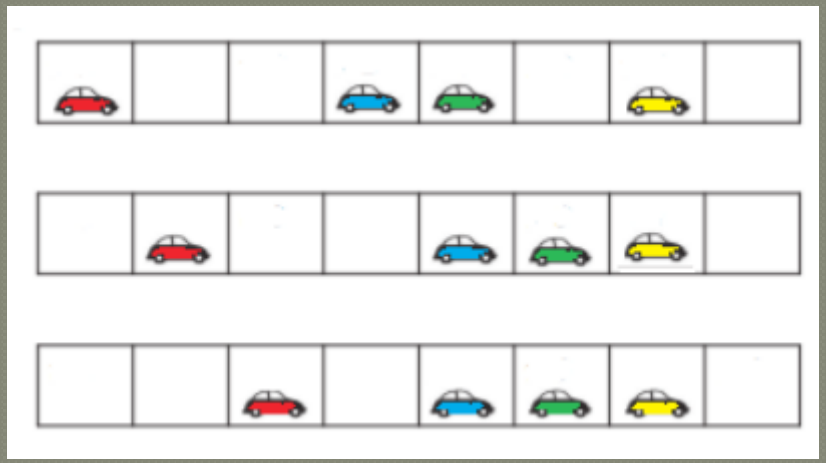
III. Elaboration de la simulation du trafic par la mécanique des fluides

III.1 Résolution graphique

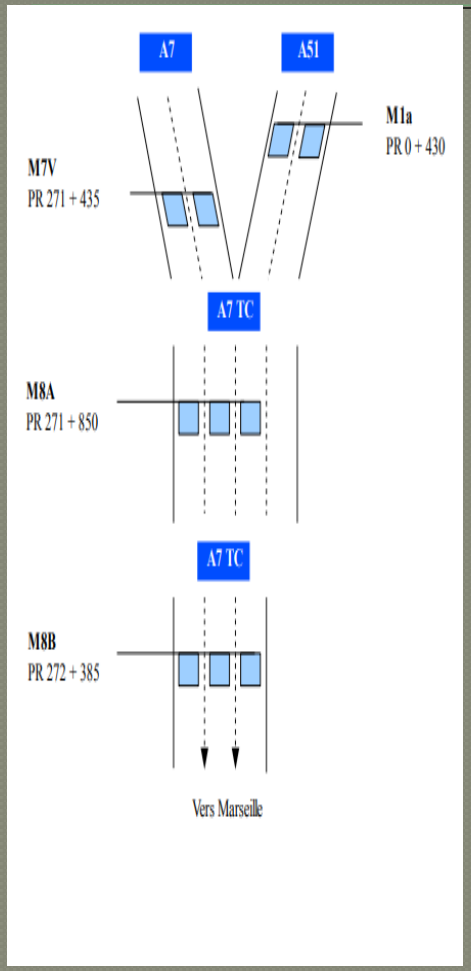
III.2 Simulation numérique

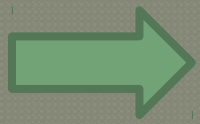
I. Choix d’un modèle

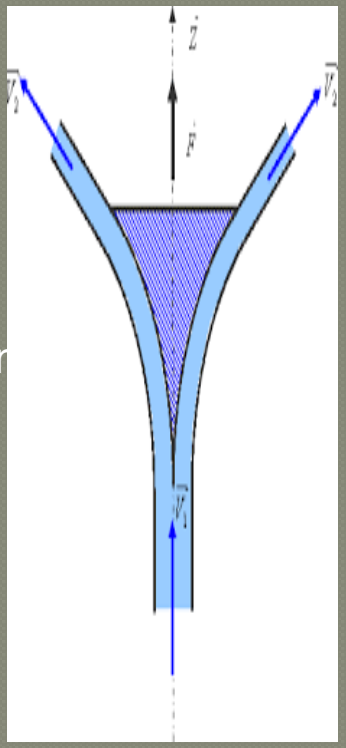
e seule voie d’autoroute → étude unidimensionnelle
s de contacts entre les voitures
squ’une voiture avance ,celle qui est derrière prend sa place



Analogies entre physique et mathématique :







On choisie alors de modéliser Le trafic routier par le principe D'écoulement d'un fluide Incompressible.
Régi par les équation du modèles macroscopique

$$\frac{\partial q(x,t)}{\partial x}+\frac{\partial c(x,t)}{\partial t}=0$$

.

$$Q(x,t)=C(x,t)\times V(x,t)$$

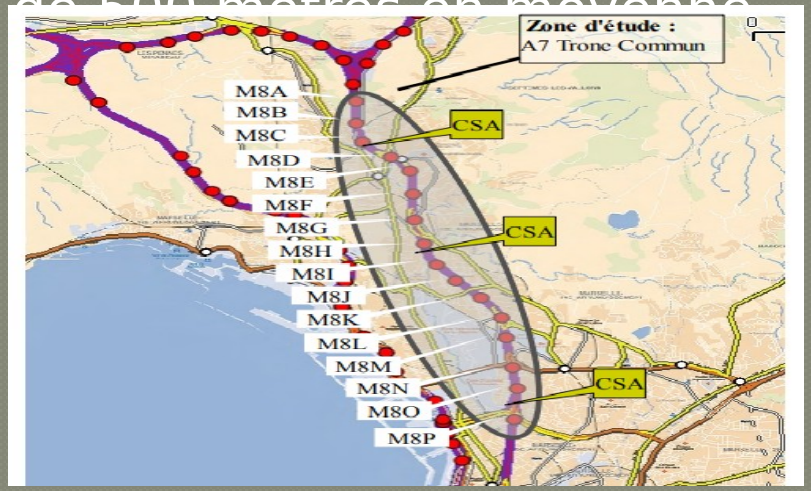
.

Ou Q,C, et V désignent respectivement débit ,concentration et vitesse moyenne des véhicules à (x, t) donnés.

Mise en application :

Zone d’étude :

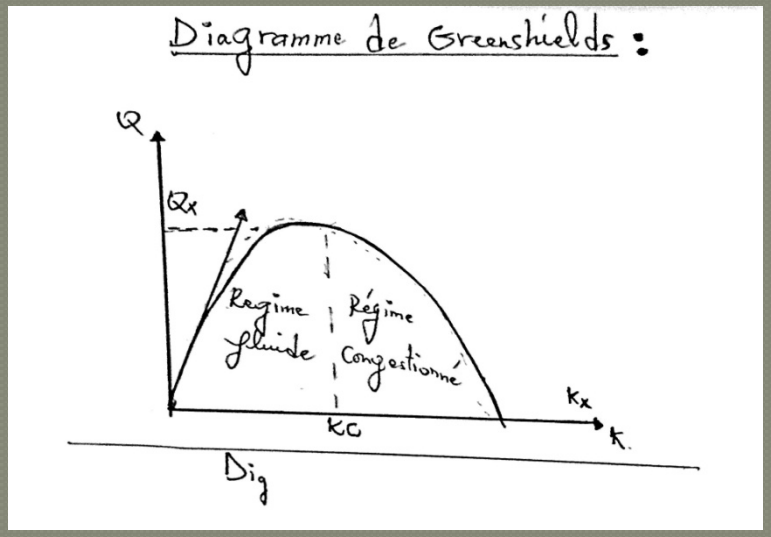
La cartographie ci-après présente la zone d'étude, l'emplacement des stations SIREDO et des contrôles sanctions automatiques. Le sens étudié : A7TC : Septièmes => Marseille Centre. La longueur de l'axe étudié : 8,5 kilomètres entre le PR 271+850 et le PR 280+90. Ce réseau comporte globalement 3 voies, sauf au niveau de la dernière station au sud qui en comporte deux. Les stations sont distantes de 500 mètres en moyenne.



Etude théorique:

Le diagramme de Greenshields suppose tout simplement que la décroissance de la vitesse. En remarquant que le débit est égal au produit de la vitesse et la concentration

On obtient un diagramme fondamental en débit de forme parabolique ,comme on peut le voir sur la figure ci-dessous ;



L’allure obtenue correspond à une parabole puisque Q=C.V d’équation

$$q = v \times c = c(1 - \frac{c}{c\_\max})v\_\max$$

II.2 Estimation d’ état de congestion:

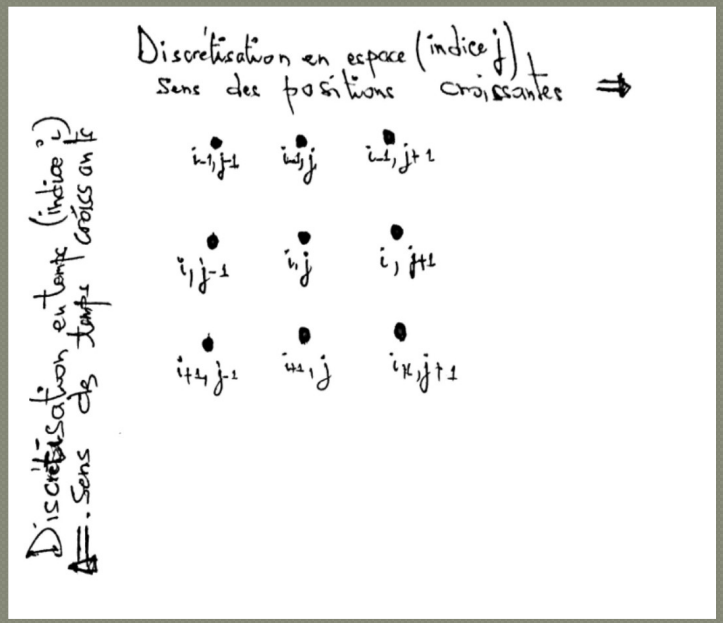
- Au niveau d’une station de mesure, la situation est dite congestionnée lorsque les vitesses prises par les véhicules restent inférieures à 40 km /h et la situation est dite fluide lorsque les vitesses restent supérieures à 80 km /h.

III. élaboration d’une simulation du trafic par la mécanique des fluides:

- En considérant une portion d’autoroute dx pendant une durée dt et en supposant qu’il n’y a ni perte, ni création de véhicule, il est possible de montrer que q(t,x) et c(t,x) verifient l’equation aux dérivées partielles suivante : ∂q(t,x) ∂x + ∂c(t,x) ∂t = 0. Pour comprendre comment évoluent la concentration, la vitesse moyenne ou le débit de v éhicules au cours du temps le long de l’autoroute, il convient donc de résoudre cette équation aux derivées partielles a partir de la situation initiale.

Discrétisation:

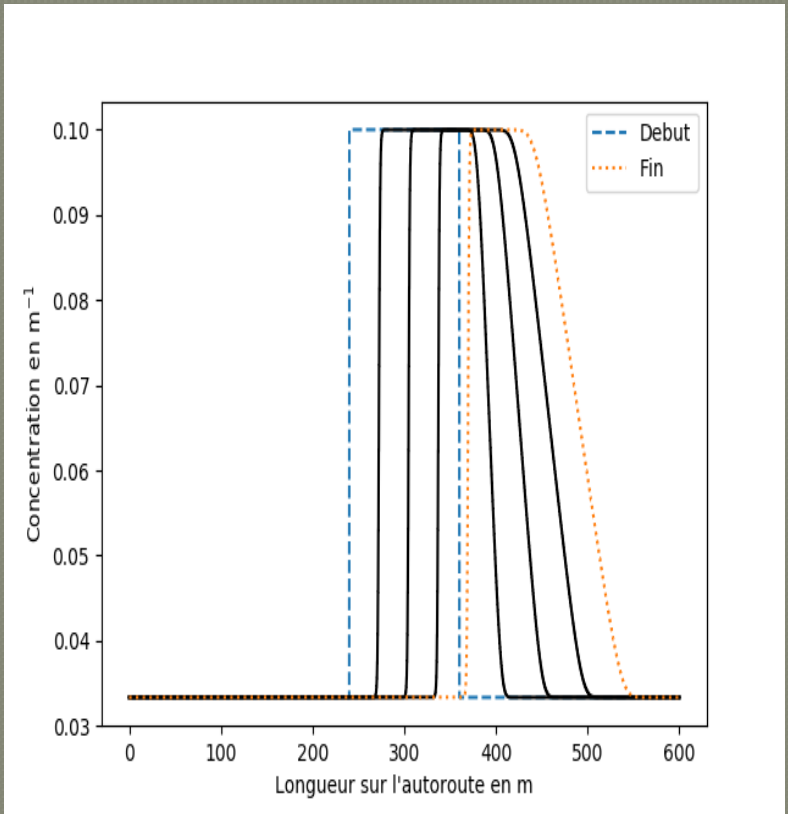
- S’il est possible de résoudre analytiquement le modèle de LWR sur des cas simples, cette résolution devient vite très difficile quand on s’intéresse à des cas plus complexes. En effet, si le calcul des ondes de choc est relativement facile, celui des éventails est en revanche bien plus difficile dans des cas complexes, comme il est noté dans [Bourrel et Henn 2002b].

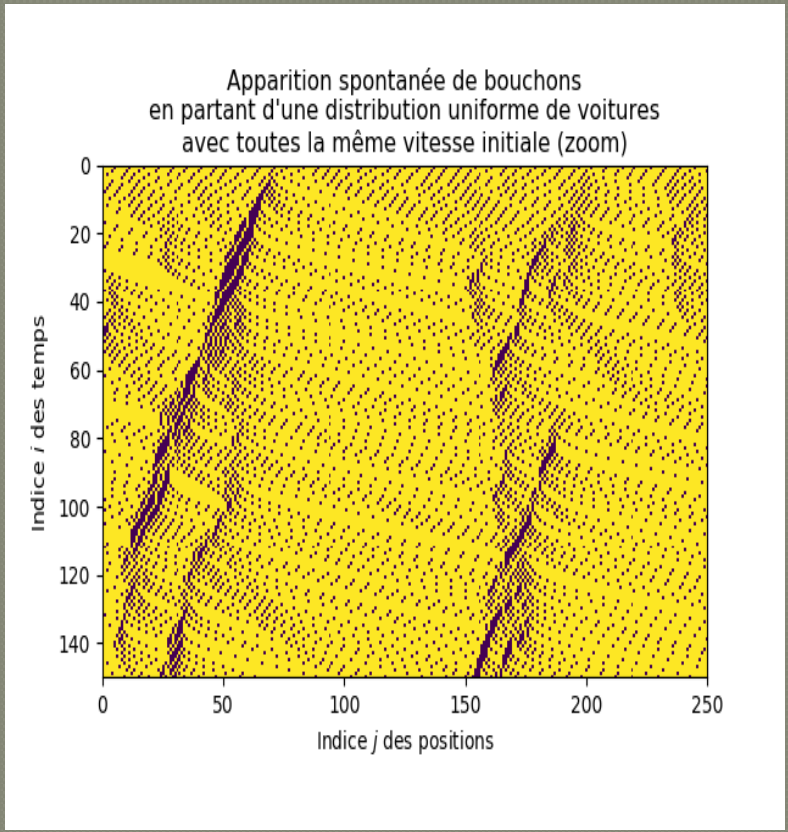


Conclusion:

- Nous avons ici passé en revue le principale modèle d’écoulement : le modèle macroscopique dans lequel le flot de véhicules est assimilé à un flux continu.
- Cette présentation de modèle d’écoulement nous a permis de voir que quel que soit le type de modèle, ils reposent tous sur des hypothèses simplificatrices, et qu’aucun des modèles existants ne peut prétendre représenter de façon parfaite l’écoulement .

Résolution graphique:





```
from numpy import *              # Imports présupposés par le sujet
from matplotlib.pyplot import *   # (mais mauvaise bonne idée en pratique)
from random import random as rand # Pour la fonction rand() de l'énoncé

dx = 7.5                # Pas en espace (en m)
dt = 1.2                # Pas en temps  (en s)
vm = 130                  # Vitesse maximale (en km/h)
La = 8500                 # Longueur de l'autoroute (en m)
N  = int(La // dx + 1)   # Nombre de cellules dans la simulation
p  = 160                 # Nombre de pas de temps
vcell = int(ceil(vm/3.6 * dt / dx))  # Vitesse en nombre de cellules par seconde
lance_simulation = True  # Veut-on lancer la simulation ?


def maj(Route,Vitesses,p,v_max,i):
    Vitesses_suivantes = array(Vitesses[i]) # Copie de l'état actuel
    N = len(Vitesses_suivantes)
    for j in range(N):
        if Route[i][j] == 1: # Si il y a une voiture dans la case
            # Étape 1: Accélération
            if Vitesses_suivantes[j] < v_max:
                Vitesses_suivantes[j] = Vitesses_suivantes[j] + 1
            # Étape 2: Décélération
            dn = distance(Route,i,j)
            if Vitesses_suivantes[j] > dn - 1:
                Vitesses_suivantes[j] = dn - 1
            # Étape 3: Facteur aléatoire
            if rand() < p and Vitesses_suivantes[j] > 0:
                Vitesses_suivantes[j] = Vitesses_suivantes[j] - 1
    return Vitesses_suivantes
```

```
def distance(Route,i,j):
    N = len(Route[i])
    d = 1 # On est au moins à une distance 1 de la prochaine case
    while Route[i][(j+d)%N] != 1: # Tant qu'on ne rencontre pas d'autre voiture,
        d = d+1                  # on continue à regarder devant soi
    return d

def deplacement(Vitesses,Route,Vitesses_suivantes,i):
    N = len(Vitesses_suivantes)
    for j in range(N):
        if Route[i][j] == 1: # Si il y a une voiture,
            # on prédit sa nouvelle position
            prochain = (j + Vitesses_suivantes[j])%N
            # et on met à jour
            Route[i+1][prochain] = 1
            Vitesses[i+1][prochain] = Vitesses_suivantes[j]
    return Route, Vitesses
```

```
if lance_simulation:
    Route = zeros((p,N),dtype=int)     # Initialisation du tableau des routes successives
    Vitesses = zeros((p,N),dtype=int)  # Pareil pour les vitesses
    proba_presence = 0.15              # "Proba" de trouver une voiture
    intervalle = int(1/proba_presence) # Intervalle entre deux voitures successives
    proba_attention= 0.3               # Proba que le conducteur freine
    for j in range(N): # Initialisation des voitures
        # On démarre avec les voitures bien espacées à la vitesse maximale
        if j%intervalle == 0:
            Route[0][j] = 1
            Vitesses[0][j] = vcell
    for i in range(p-1): # On passe chaque temps en revue
        v_next = maj(Route,Vitesses,proba_attention,vcell,i)
        Route,Vitesses = deplacement(Vitesses,Route,v_next,i)
    imshow(1-Route)
    xlim(0,250)
    ylim(150,0)
    xlabel('Indice $j$ des positions')
    ylabel('Indice $i$ des temps')
    title("Apparition spontanée de bouchons\nen partant d'une distribution uniforme de
voitures\navec toutes la même vitesse initiale (zoom)")
    savefig('bouchons.png')
```

```
from numpy import *              # Imports présupposés par le sujet
from matplotlib.pyplot import *   # (mais mauvaise bonne idée en pratique)
from random import random as rand # Pour la fonction rand() de l'énoncé

dx = 7.5                # Pas en espace (en m)
dt = 1.2                # Pas en temps  (en s)
vm = 130                  # Vitesse maximale (en km/h)
La = 8500                 # Longueur de l'autoroute (en m)
N  = int(La // dx + 1)   # Nombre de cellules dans la simulation
p  = 160                 # Nombre de pas de temps
vcell = int(ceil(vm/3.6 * dt / dx))  # Vitesse en nombre de cellules par seconde
lance_simulation = True  # Veut-on lancer la simulation ?


def maj(Route,Vitesses,p,v_max,i):
    Vitesses_suivantes = array(Vitesses[i]) # Copie de l'état actuel
    N = len(Vitesses_suivantes)
    for j in range(N):
        if Route[i][j] == 1: # Si il y a une voiture dans la case
            # Étape 1: Accélération
            if Vitesses_suivantes[j] < v_max:
                Vitesses_suivantes[j] = Vitesses_suivantes[j] + 1
            # Étape 2: Décélération
            dn = distance(Route,i,j)
            if Vitesses_suivantes[j] > dn - 1:
                Vitesses_suivantes[j] = dn - 1
            # Étape 3: Facteur aléatoire
            if rand() < p and Vitesses_suivantes[j] > 0:
                Vitesses_suivantes[j] = Vitesses_suivantes[j] - 1
    return Vitesses_suivantes
```

```
def distance(Route,i,j):
    N = len(Route[i])
    d = 1 # On est au moins à une distance 1 de la prochaine case
    while Route[i][(j+d)%N] != 1: # Tant qu'on ne rencontre pas d'autre voiture,
        d = d+1                  # on continue à regarder devant soi
    return d

def deplacement(Vitesses,Route,Vitesses_suivantes,i):
    N = len(Vitesses_suivantes)
    for j in range(N):
        if Route[i][j] == 1: # Si il y a une voiture,
            # on prédit sa nouvelle position
            prochain = (j + Vitesses_suivantes[j])%N
            # et on met à jour
            Route[i+1][prochain] = 1
            Vitesses[i+1][prochain] = Vitesses_suivantes[j]
    return Route, Vitesses
```

```
if lance_simulation:
    Route = zeros((p,N),dtype=int)     # Initialisation du tableau des routes successives
    Vitesses = zeros((p,N),dtype=int)  # Pareil pour les vitesses
    proba_presence = 0.15              # "Proba" de trouver une voiture
    intervalle = int(1/proba_presence) # Intervalle entre deux voitures successives
    proba_attention= 0.3               # Proba que le conducteur freine
    for j in range(N): # Initialisation des voitures
        # On démarre avec les voitures bien espacées à la vitesse maximale
        if j%intervalle == 0:
            Route[0][j] = 1
            Vitesses[0][j] = vcell
    for i in range(p-1): # On passe chaque temps en revue
        v_next = maj(Route,Vitesses,proba_attention,vcell,i)
        Route,Vitesses = deplacement(Vitesses,Route,v_next,i)
    imshow(1-Route)
    xlim(0,250)
    ylim(150,0)
    xlabel('Indice $j$ des positions')
    ylabel('Indice $i$ des temps')
    title("Apparition spontanée de bouchons\nen partant d'une distribution uniforme de
voitures\navec toutes la même vitesse initiale (zoom)")
    savefig('bouchons.png')
```