Modélisation mathématique du trafic routier

Abderrahmane EDDAHBI

Numéro du dossier : 32038

La congestion

Nécessité de gérer au mieux

Le pétrole à 200 $ le baril

Réduire: consommation, émission polluants

Rationaliser le système de transports et l'usage de la voiture

Intolérance au risque, demandes sécuritaires

Comment modéliser le trafic

Modèle microscopique

$\boldsymbol{x_i(t)}$: La position du véhicule i au temps t

$\boldsymbol{u_i(t) = \dot{x}_i(t)}$: La vitesse instantanée du véhicule i au temps t

$\boldsymbol{a_i(t) = \ddot{x}_i(t)}$: L'accélération du véhicule i au temps t

Modèle de Chandler

$$M\frac{d}{dt}u_i(t)=\gamma[u_{i-1}(t-\tau)-u_i(t-\tau)]$$ **(1)**

Loi de poursuite

$(\boldsymbol{u_i})_{\boldsymbol{i}}$ est l'ensemble des vitesses des N véhicules considérés,
M est la masse supposée identique entre tous ces véhicules,
$\boldsymbol{\gamma}$ est un coefficient réel donnée
$\tau \geq 0$ est un temps de retard (identique entre tous les conducteurs).

On introduit $(x_i)_i$ l'ensemble des positions des véhicules

$$x_i(t) = \int_0^t u_i(s)ds$$

$$\forall t \geq \tau\,, \forall i \in \{2,\cdots,N\}, \qquad \ddot{x}_i(t) = \alpha[\,\dot{x}_{i-1}(t-\tau) - \dot{x}_i(t-\tau)\,]$$

(2)

$$\alpha = \frac{\gamma}{M}$$

Résultat préliminaire

En un point x et pour une valeur h du pas de discrétisation tels que u soit trois fois dérivable sur l'intervalle [x-2h, x], la formule de Taylor-Young conduit à la relation :

$$\boldsymbol{u'(x) \approx \frac{u(x)-u(x-h)}{h}}$$

$$\boldsymbol{u''(x) \approx \frac{u(x)-2u(x-h)+u(x-2h)}{h^2}}$$

Résolution numérique

schéma numérique sur les positions

- On introduit une discrétisation régulière du temps Δt tel que $0 < \Delta t < \tau$
- Pour simplifier les calculs on choisit Δt tel que $\exists k \in IN$, $\tau = k\Delta t$
- Soit U_i^n une solution numérique approchée de la solution exacte de l'EDO (1)
- $$\boxed{U_i^n \approx u_i(n\Delta t)}$$

$$Soient: \quad X_i^n \approx x_i(n\Delta t) \;,\; U_i^n \approx \dot{x}_i(n\Delta t) \;,\; A_i^n \approx \ddot{x}_i(n\Delta t)$$

$$u_i(t-\tau) = u_i(n\Delta t - k\Delta t) = u_i\big((n-k)\Delta t\big) = U_i^{n-k}$$

(2) $\Rightarrow A_i^n = \alpha[U_{i-1}^{n-k} - U_i^{n-k}]$: **(3)**

On prend $h=\Delta t$ et $t=n\Delta t$

Le résultat préliminaire nous donne :

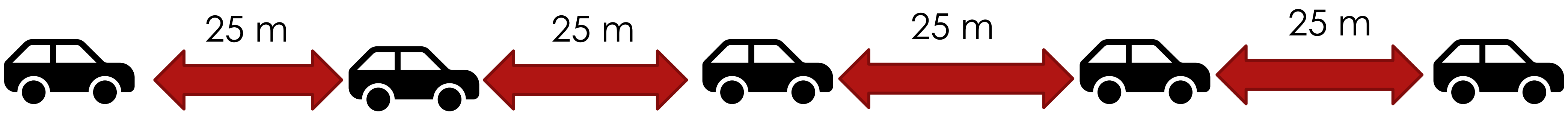
(4) : $$\begin{cases} A_i^n = \dfrac{X_i^n - 2X_i^{n-1} + X_i^{n-2}}{\Delta t^2} \\ U_i^n = \dfrac{X_i^n - X_i^{n-1}}{\Delta t} \end{cases}$$

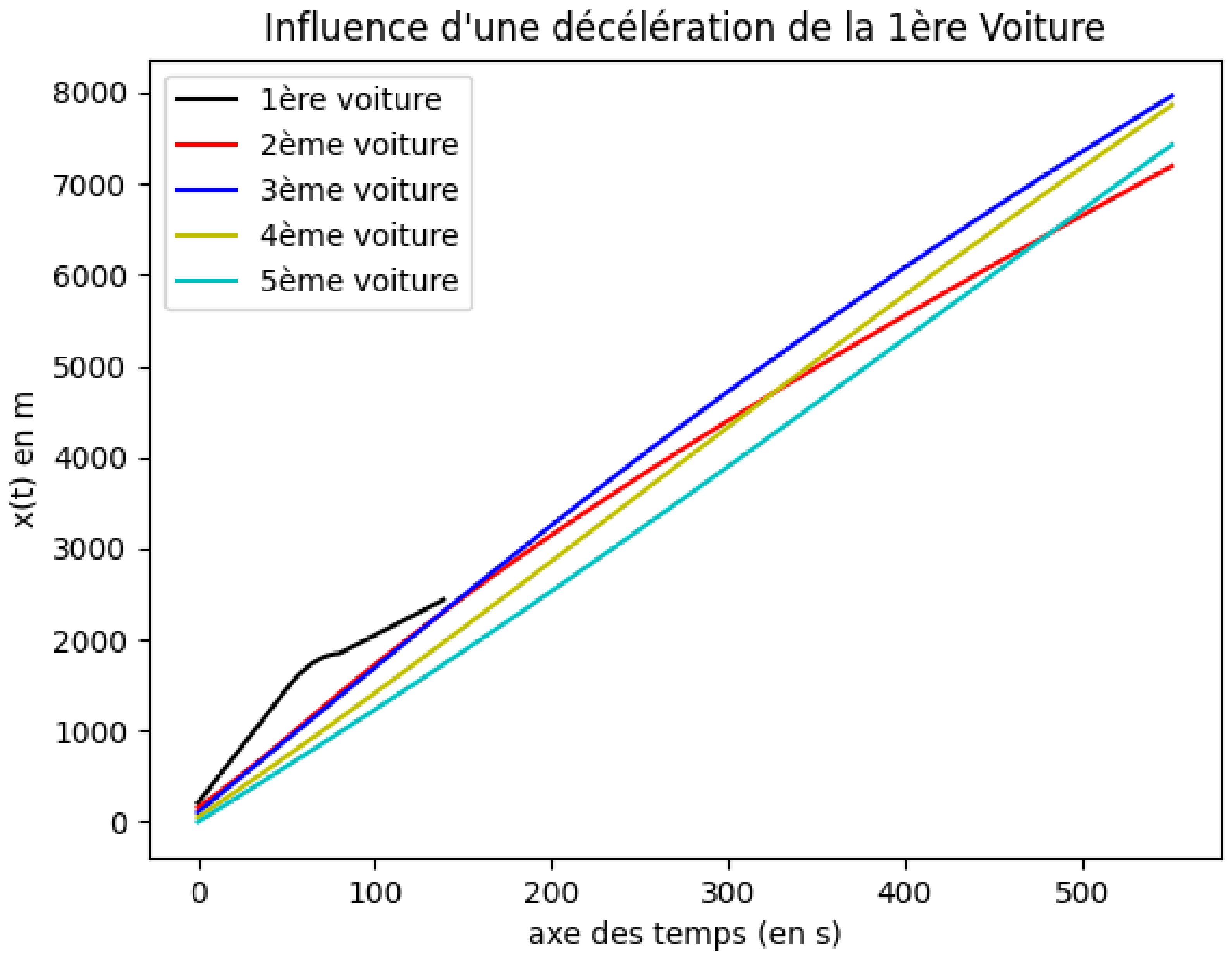
On obtient en insérant (4) dans (3)

$$X_i^{n+1} = \alpha\Delta t\,[X_{i-1}^{n-k+1} - X_i^{n-k+1} + X_i^{n-k} - X_{i-1}^{n-k}] + 2\,X_i^{n} - X_i^{n-1}$$

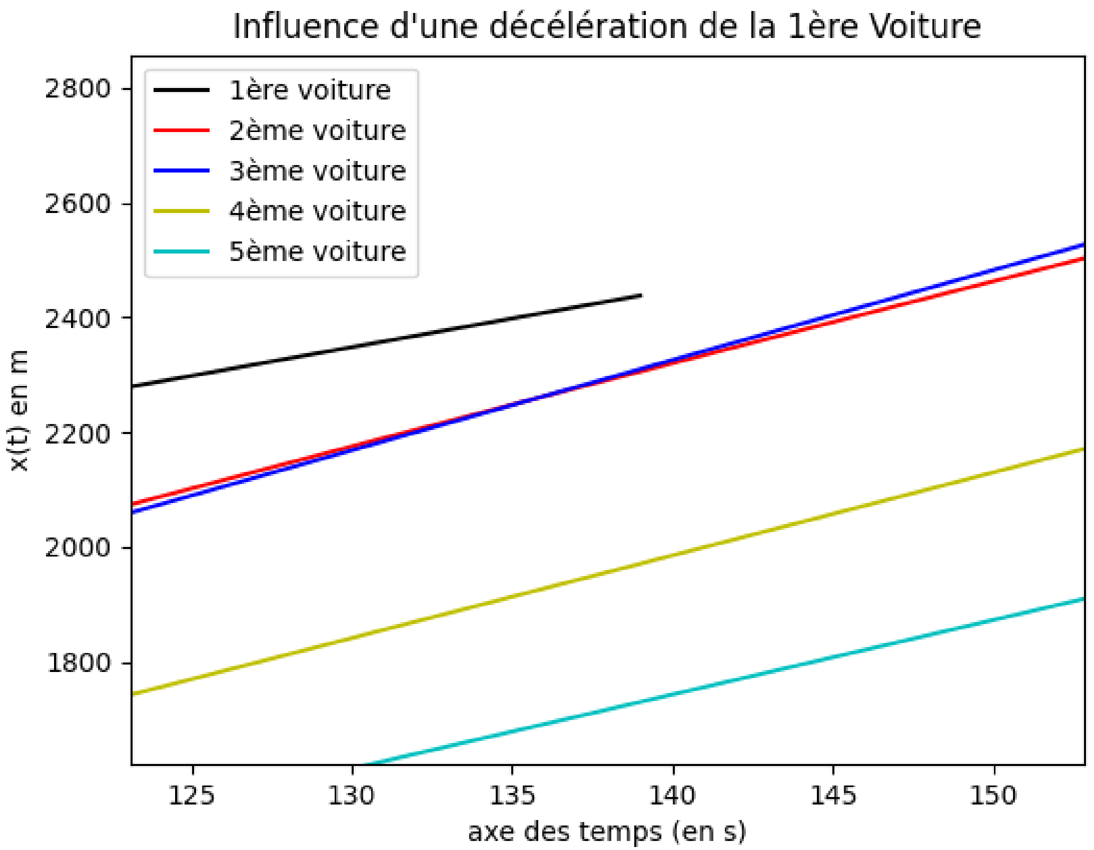
Pour simplifier on prend k=1 càd $\tau = \Delta t$

Distance de sécurité = 25m

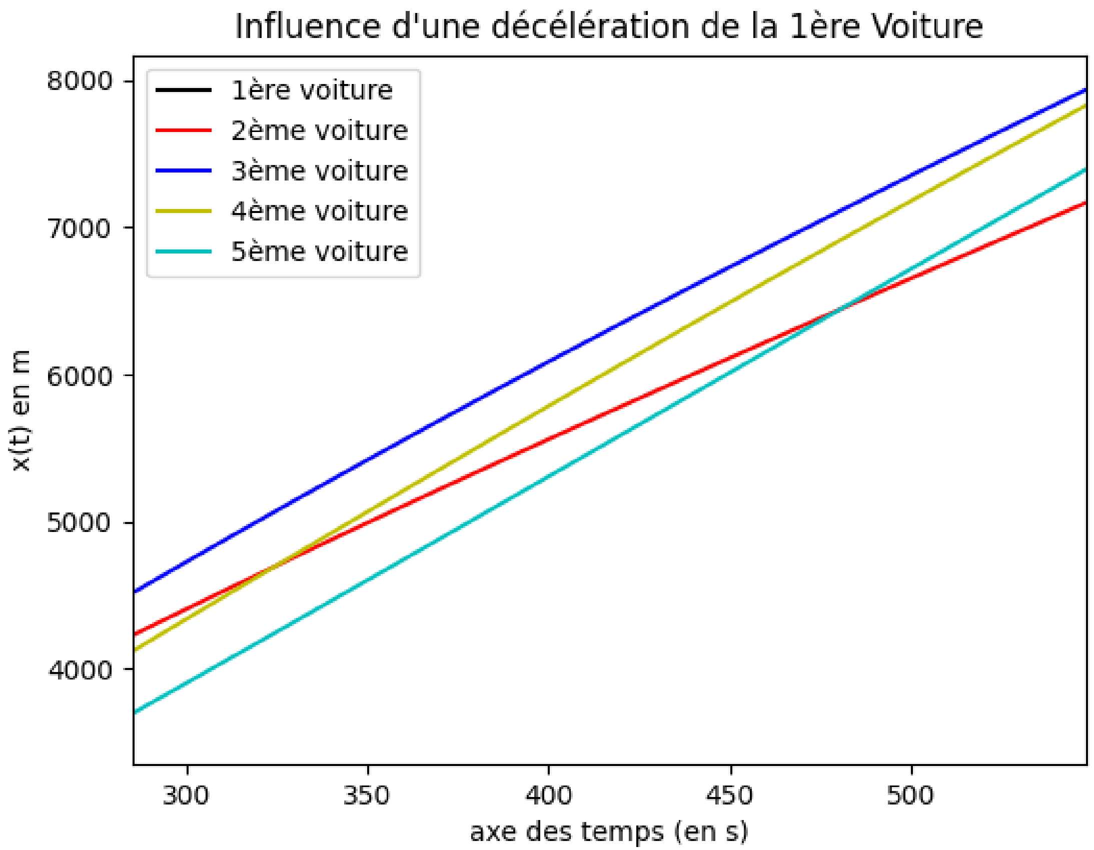




Distance de sécurité = 25m

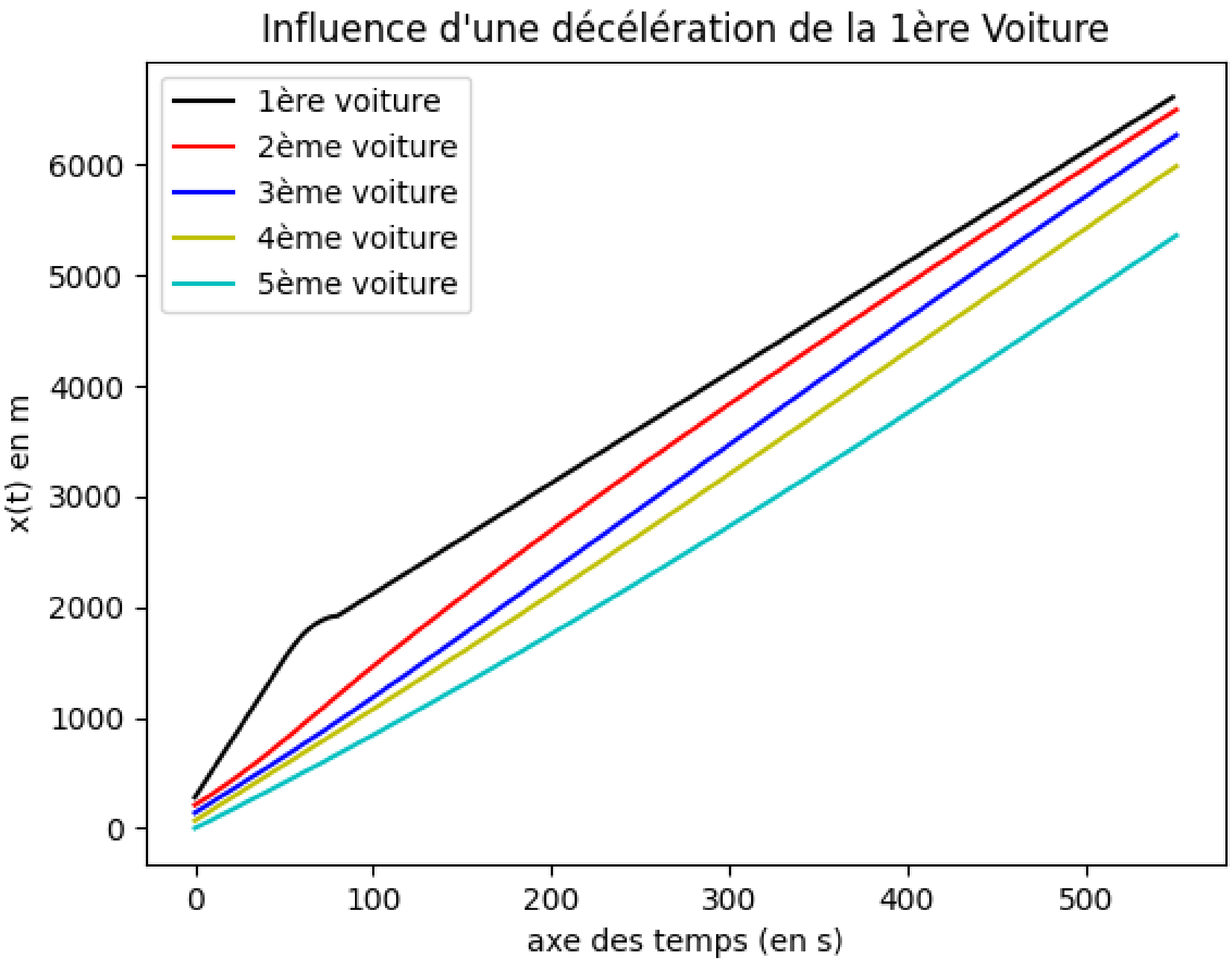


Influence d'une décélération de la 1ère Voiture
1ère voiture
2ème voiture
3ème voiture
4ème voiture
5ème voiture
x(t) en m
axe des temps (en s)
2800
2600
2400
2200
2000
1800
125
130
135
140
145
150



Influence d'une décélération de la 1ère Voiture
1ère voiture
2ème voiture
3ème voiture
4ème voiture
5ème voiture
x(t) en m
axe des temps (en s)
8000
7000
6000
5000
4000
300
350
400
450
500

Distance de sécurité = 70m



Importance de la distance de sécurité entre deux voitures pour éviter les accidents

Modèle macroscopique

$q(x, t)$: le flux de véhicules en un point et un instant donné

$\rho(x, t)$: La concentration (appelée également densité spatiale) instantanée correspondant au nombre de véhicules par unité de longueur se trouvant sur une section voisine du point d’abscisse x, au temps t

V : la vitesse moyenne

Deux processus en compétition

Besoin de sécurité: distance de sécurité qui augmente avec la vitesse
Désir d'aller vite : dont la réalisation est limitée par le besoin de sécurité

Résultats

-Régime fluide: concentration faible, vitesse élevée, débit croît avec la concentration
-Régime congestionné: concentration élevée, vitesse faible, débit décroît avec la concentration

Modèle de Lighthill-Whitham-Richards

Hypothèses

- une route unidimensionnelle et de longueur infinie
- Il n'y a pas d'intersections le long le de la route. Ainsi, le nombre de voitures sur la route est constant
- Il n'y a pas de dépassement possible.
- $v(\rho) = v_m(1 - \frac{\rho}{\rho_m})$

Loi de conservation :

$$\forall (t,x) \in D \subseteqq [0,+\infty[\times IR \qquad \frac{\partial}{\partial t}\rho(x,t) + \frac{\partial}{\partial x} q\big(\rho(x,t)\big) = 0$$

$$q = \rho v$$

$$\frac{\partial \rho}{\partial t} + v_m\left(1 - \frac{2\rho}{\rho_m}\right)\frac{\partial \rho}{\partial x} = 0 \quad : \textbf{(5)}$$

Démonstration en annexe

Enfin, on associe cette équation à la condition initiale suivante :

$$\rho(x,0) = h(x)$$

Méthode des caractéristiques

Pour chaque point (x,t) on le relie à $(x_0, 0)$ par une courbe le long de laquelle ρ est constante - une caractéristique tel que $\rho(x, t) = \rho(x_0, 0) = h(x_0)$

On obtient : $x(t) = q'(h(x_0))\, t + x_0$

Et $\rho(x, t) = h(x - q'(h(x_0))\, t)$

Démonstration en annexe

Applications

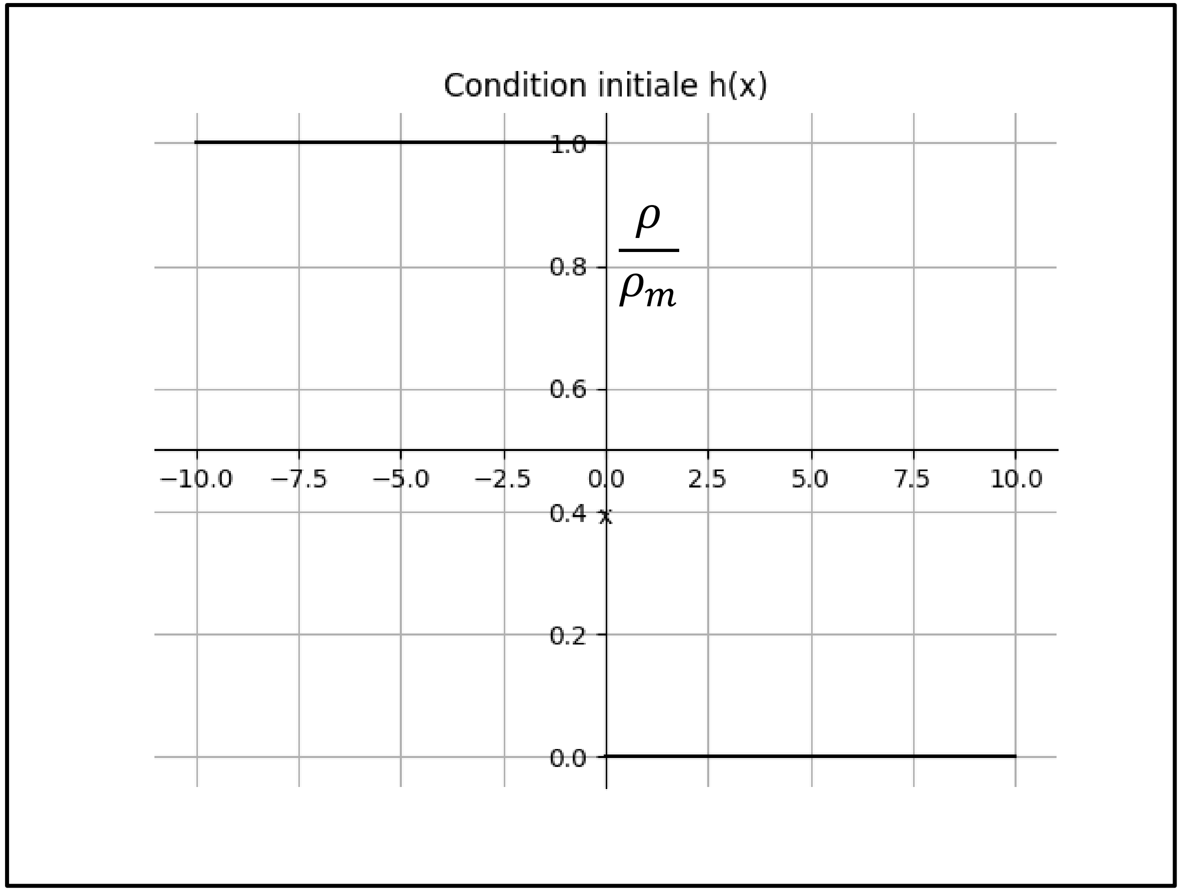
Feu tricolore

$$h(x) = \begin{cases} \rho_m & si\ x \leq 0 \\ 0 & si\ x > 0 \end{cases}$$

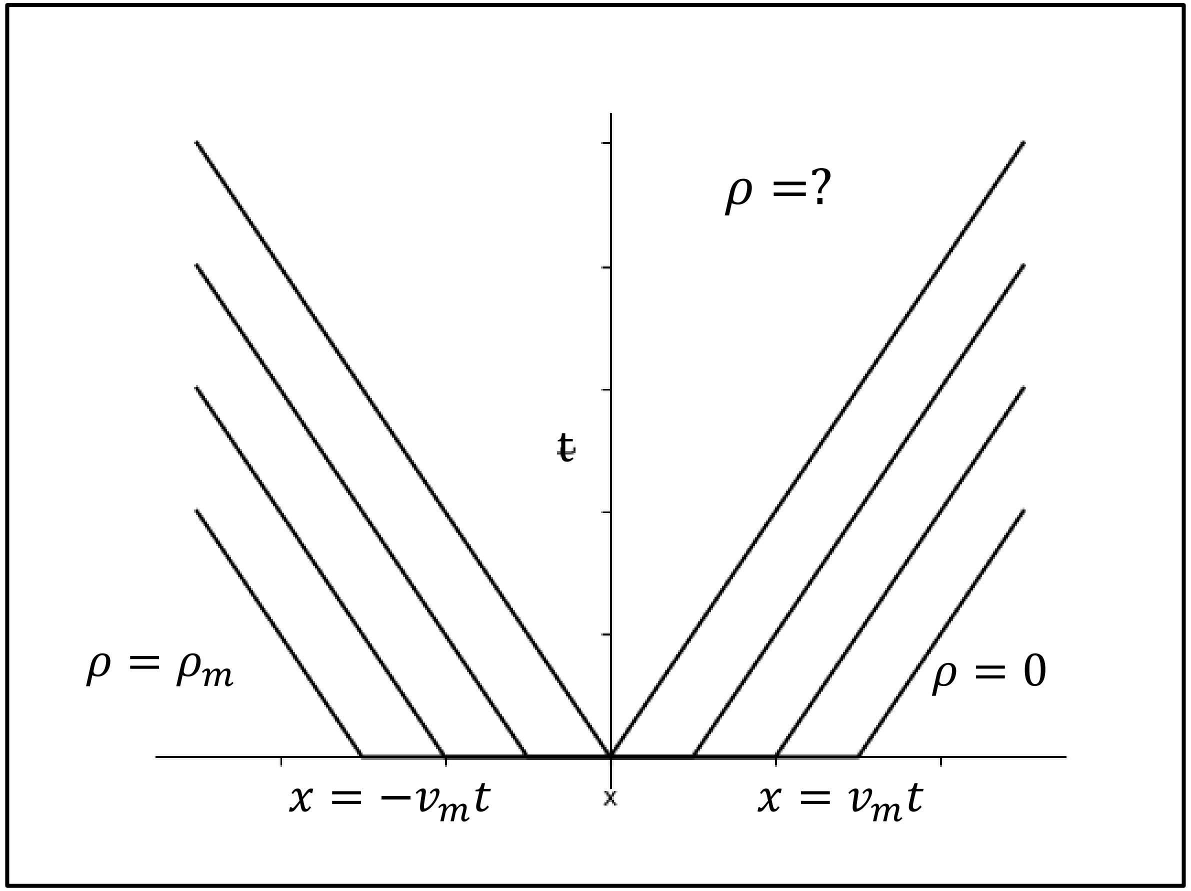
$$q'(\rho) = v_m\left(1 - \frac{2\rho}{\rho_m}\right)$$

$$q'(h(x_0)) = \begin{cases} -v_m & si\ x_0 \leq 0 \\ v_m & si\ x_0 > 0 \end{cases}$$

On cherche à déterminer ce qui se passe dans la région délimitée par les droites d'équation $\boldsymbol{x = v_m t}$ et $\boldsymbol{x = -v_m t}$ où ne passe aucune caractéristique



Condition initiale h(x)
1.0
0.8
0.6
0.4
0.2
0.0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
x
$\frac{\rho}{\rho_m}$

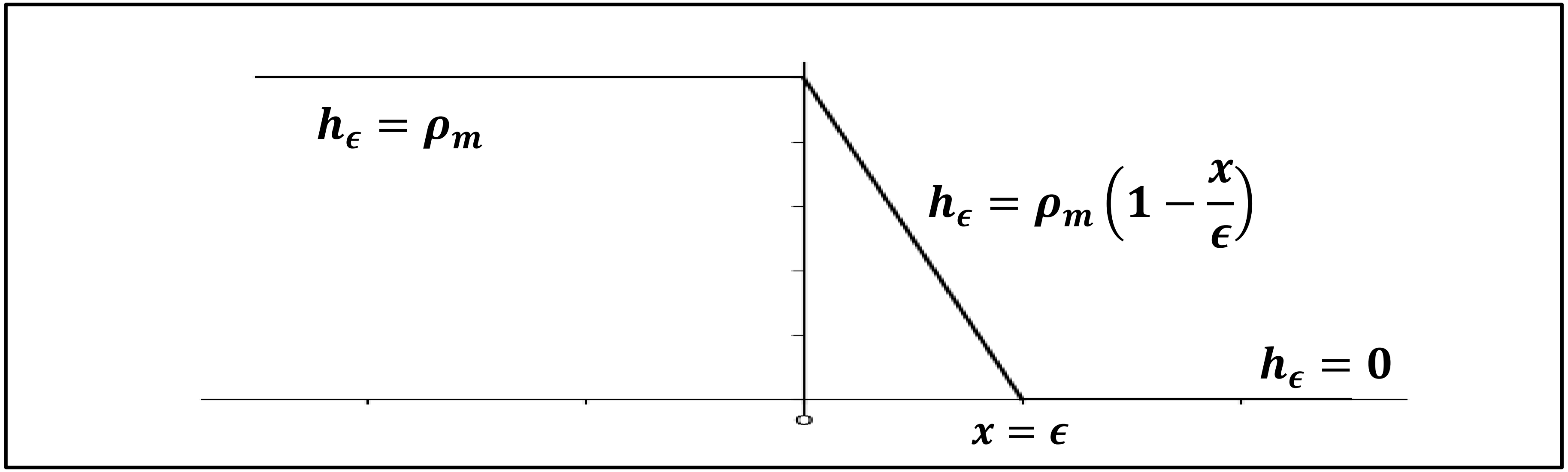


$\rho =?$
t
$\rho = \rho_m$
$\rho = 0$
$x = -v_m t$
x
$x = v_m t$

Problème de discontinuité

$$h_\epsilon(x) = \begin{cases} \rho_m & si\ x \leq 0 \\ \rho_m\left(1 - \frac{x}{\epsilon}\right) & si\ 0 < x < \epsilon \\ 0 & si\ x \geq \epsilon \end{cases}$$

h_ϵ est continue



L'équation des caractéristiques dans la région comprise entre $-\boldsymbol{v_m t}$ et $\boldsymbol{v_m t}$ est :

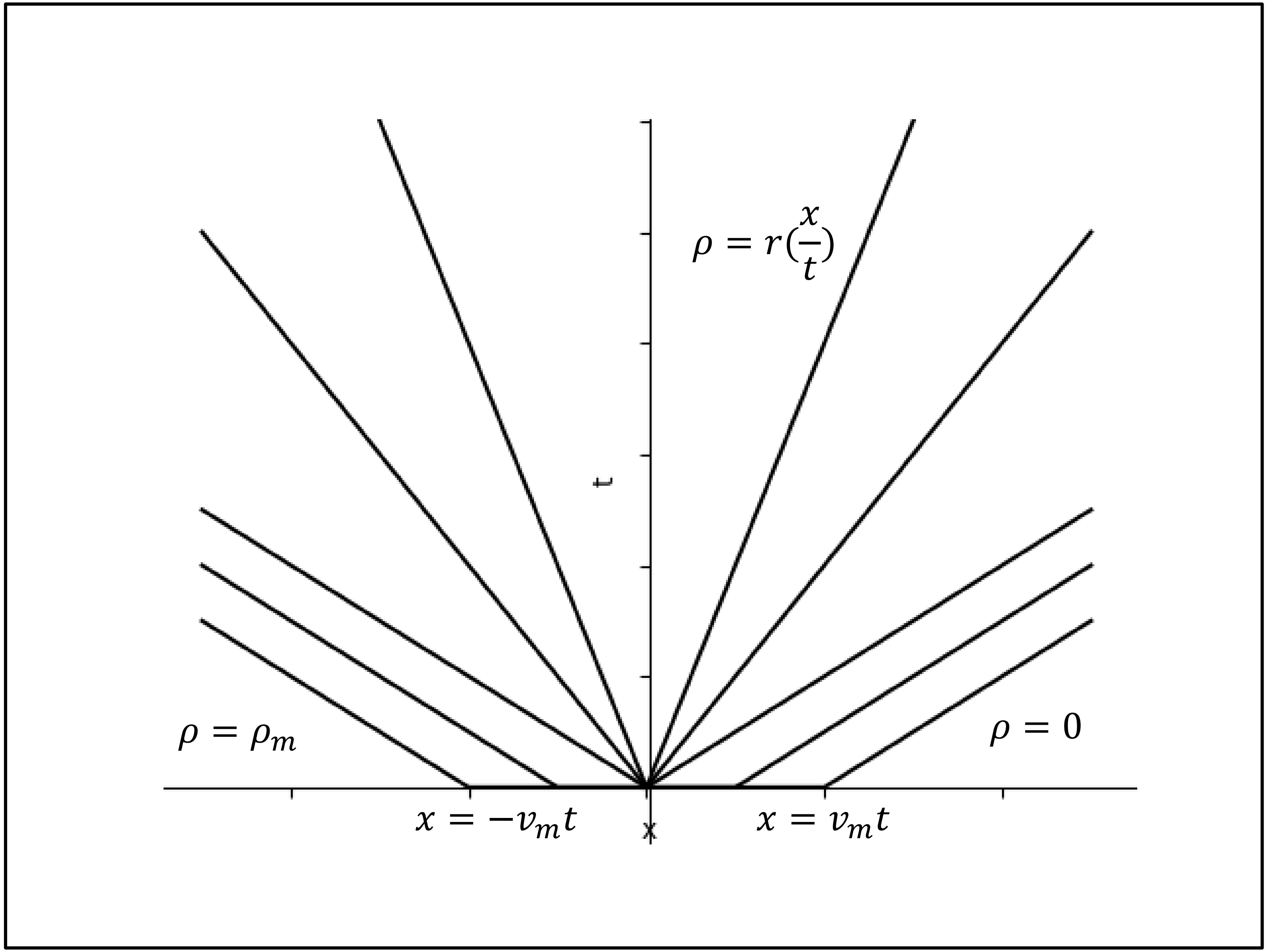
$$x(t) = -v_m\left(1 - \frac{2x_0}{\epsilon}\right)t + x_0$$

Démonstration en annexe

D'où $$\rho_\epsilon = \rho_m\left(1 - \frac{x+v_m t}{2v_m t+\epsilon}\right)$$

On fait tendre ϵ vers 0 :

$$\rho(x,t) = \begin{cases} \rho_m & si \quad x \le -v_m t \\ \frac{\rho_m}{2}\left(1 - \frac{x}{v_m t}\right) & si \quad -v_m t < x < v_m t \\ 0 & si \quad x \ge v_m t \end{cases}$$



Onde de raréfaction

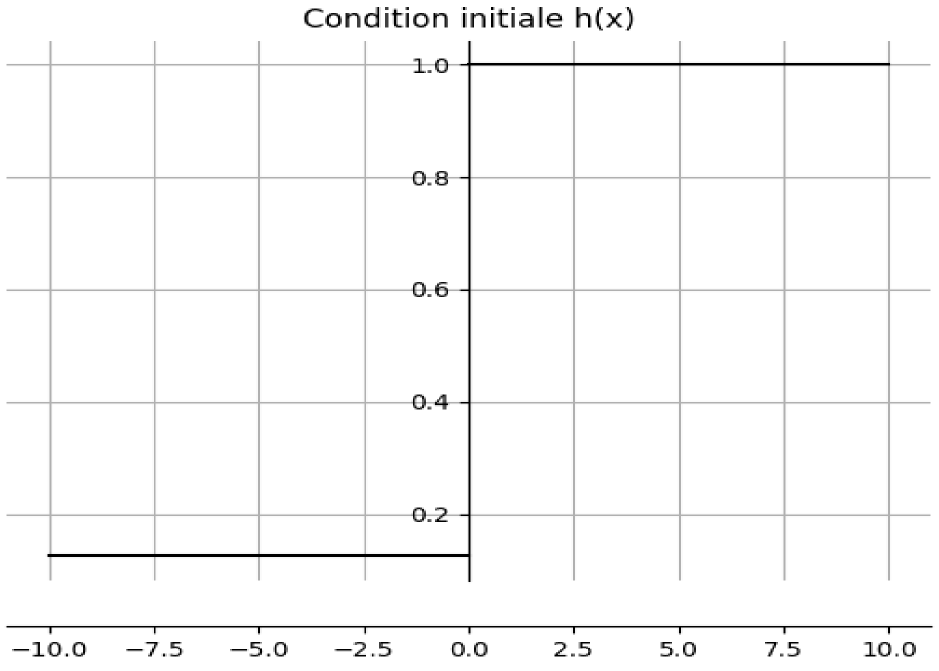
Embouteillage

$$h(x) = \begin{cases} \frac{1}{8}\rho_m & si\ x \leq 0 \\ \rho_m & si\ x > 0 \end{cases}$$

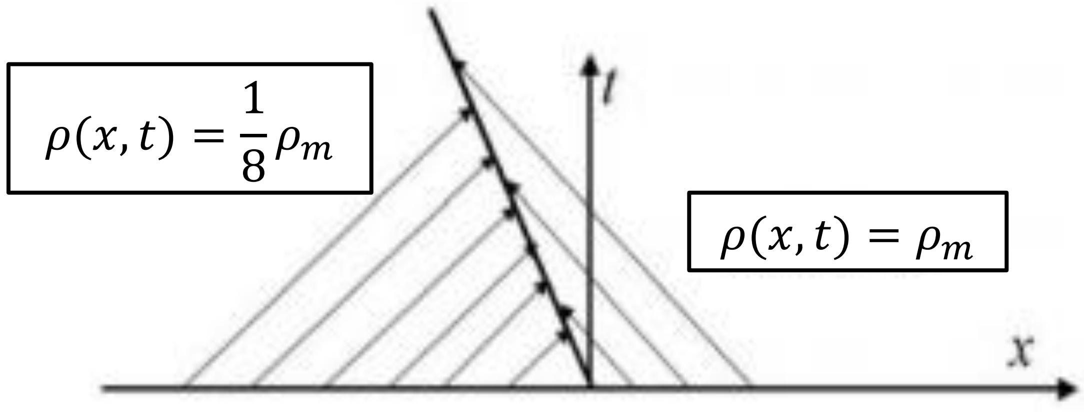
$$q'(h(x_0)) = \begin{cases} \frac{3}{4}v_m & si\ x_0 \leq 0 \\ -v_m & si\ x_0 > 0 \end{cases}$$

$$x(t) = \begin{cases} \frac{3}{4}v_m t + x_0 & si\ x_0 \leq 0 \\ -v_m t + x_0 & si\ x_0 > 0 \end{cases}$$

$$\rho(x,t) = \begin{cases} \frac{1}{8}\rho_m & si\ x \leq -\frac{1}{8}v_m t \\ \rho_m & si\ x > -\frac{1}{8}v_m t \end{cases}$$



Condition initiale h(x)



Onde de choc pour un embouteillage

Conclusion

Merci pour votre attention !

Annexes:

Loi de conservation

$$q = \rho v_m \left(1 - \frac{\rho}{\rho_m}\right) = v_m(\rho - \frac{\rho^2}{\rho_m})$$

$$\frac{\partial q}{\partial x} = \frac{\partial q}{\partial \rho}\frac{\partial \rho}{\partial x} = v_m \left(1 - \frac{2\rho}{\rho_m}\right) \frac{\partial \rho}{\partial x}$$

$$\frac{\partial \rho}{\partial t} + v_m \left(1 - \frac{2\rho}{\rho_m}\right) \frac{\partial \rho}{\partial x} = 0$$: **(5)**

Méthode des caractéristiques

Cherchons x(t):

On a $\forall t \geq 0, \rho(x(t), t) = h(x_0)$

On dérive : $\frac{d}{dt}\rho(x(t),t) = \frac{\partial}{\partial x}\rho(x(t),t)x'(t) + \frac{\partial}{\partial t}\rho(x(t),t) = 0$: (6)

(5) $\Rightarrow$ $\frac{\partial}{\partial t}\rho(x(t),t) + q'(h(x_0))\frac{\partial}{\partial x}\rho(x(t),t) = 0$: (7)

(7)-(6) : $$\frac{\partial}{\partial t}\rho(x(t),t)\,[x'(t) - q'(h(x_0))] = 0$$

Donc : $$x'(t) = q'(h(x_0))$$

D'où : $$\mathrm{x}(t) = q'(h(x_0))\,t + x_0$$

On a finalement : $$\rho(x,t) = h(x - q'(h(x_0))\,t)$$

Feu tricolore

$$q'(\rho) = v_m\left(1 - \frac{2\rho}{\rho_m}\right)$$

$$q'(h(x_0)) = v_m\left(1 - \frac{2\rho_m}{\rho_m}\left(1 - \frac{x_0}{\epsilon}\right)\right) = v_m\left(1 - 2 + \frac{2x_0}{\epsilon}\right) = -v_m\left(1 - \frac{2x_0}{\epsilon}\right)$$

Et comme : $\mathrm{x}(t) = q'(h(x_0))\, t + x_0$

Alors : $x(t) = -v_m\left(1 - \frac{2x_0}{\epsilon}\right) t + x_0$

On obtient $x_0 = \frac{x + v_m t}{\frac{2v_m t}{\epsilon} + 1}$

Et comme : $$\rho(x,t) = h(x - q'(h(x_0))\,t)$$

Alors : $$\rho = h\left(x + v_m\left(1 - \frac{2x_0}{\epsilon}\right)t\right) \quad \text{et} \quad h(x) = \rho_m\left(1 - \frac{x}{\epsilon}\right)$$

Finalement : $$\rho_\epsilon = \rho_m\left(1 - \frac{x + v_m t}{2v_m t + \epsilon}\right)$$

Démonstration de la formule obtenue par Taylor Young

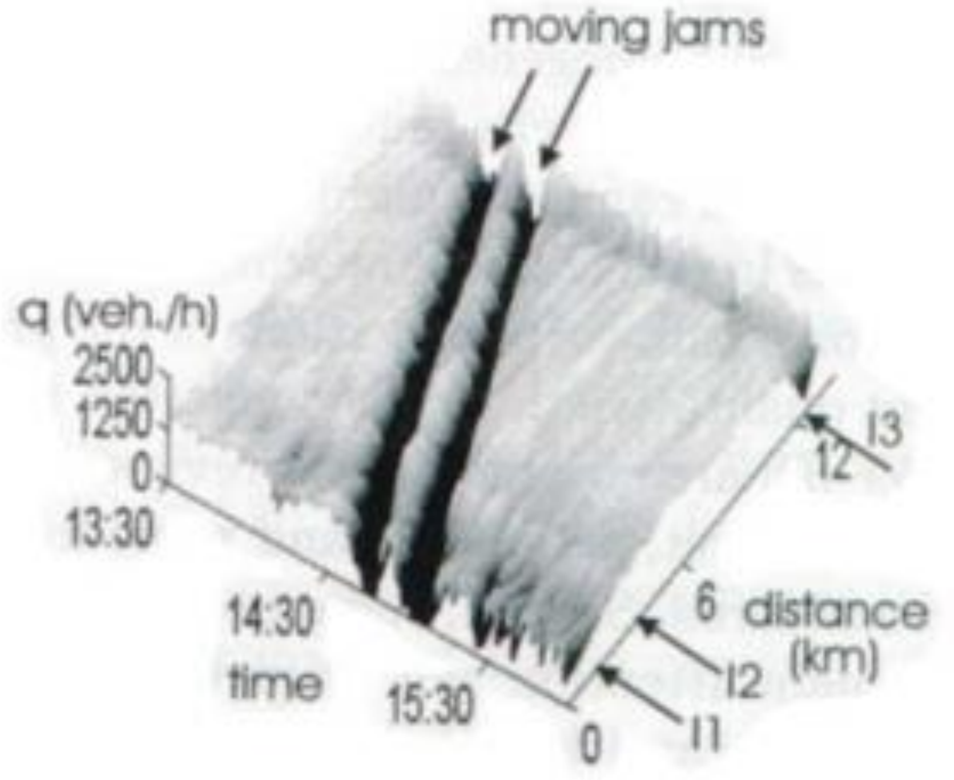
$$u(x+h) = u(x) + \sum_{n=1}^{3} \frac{h^n}{n!} u^{(n)}(x) + h^3 \varepsilon_1(x,h)$$

où la fonction $\varepsilon_i(x,h)$ converge vers 0 avec h.

$$\frac{u(x+h)-u(x)}{h} = u'(x) + \frac{h}{2}u''(x) + \frac{h^2}{3!}u^{(3)}(x) + h^2 \varepsilon_1(x,h)$$

$$u''(x) = \frac{\frac{u(x+2h)-u(x+h)}{h} - \frac{u(x+h)-u(x)}{h}}{h} + \varepsilon_2(x,h)$$

$$\frac{u(x+2h)-2u(x+h)+u(x)}{h^2} = u''(x) + \frac{h^2}{6}u^{(3)}(x) + h^2 \varepsilon_3(x,h)$$



Propagation de deux congestions sur l'autoroute A5-Nord en Allemagne, d'après [Kerner, 1999]

Code Python

```
import matplotlib.pyplot as plt
import numpy as np


def Resol_1_(f,t,a,p,x0,x1):
    # f est la fonction de mouvement de la voiture qui precede
    # t est x le pas de discrétisation en temps
    # a est alpha de l'équation
    # x0 et x1 sont les valeurs initiales de la position de la voiture
    T=[0,t]
    L=[x0,x1]
    for i in range(p):
        L.append(a*t*(f(T[i+1])-L[i+1]+L[i]-f(T[i]))+2*L[i+1]-L[i])
        T.append((i+2)*t)
    return T, L

def Resol_2_(M,t,a,p,x0,x1):
    T=[0,t]
    L=[x0,x1]
    for i in range(p):
        L.append(a*t*(M[i+1]-L[i+1]+L[i]-M[i])+2*L[i+1]-L[i])
        T.append((i+2)*t)
    return T, L
```

```python
# Fonction de déceleration:
def g(t):
    if t>=0 and t<=50:
        return 25*t+208
    elif t>50 and t<=80 :
        return -0.4*((t-50)**2)+25*(t-50)+1250+208
    elif t>80 :
        return   10*(t-80)+1640+208

# Simulation :
N=[i for i in range(140)]

Y2=Resol_1_(g,1,5*(10**-3),550,156,170)
P=Y2[0]
Y1=[g(x) for x in N]
Y3=Resol_2_(Y2[1],1,5*(10**-3),550,104,120)
Y4=Resol_2_(Y3[1],1,5*(10**-3),550,52,65)
Y5=Resol_2_(Y4[1],1,5*(10**-3),550,0,12)

plt.plot(N,Y1,"k",label='1ère voiture')
plt.plot(Y2[0],Y2[1],"r",label='2ème voiture')
plt.plot(Y3[0],Y3[1],"b",label='3ème voiture')
plt.plot(Y4[0],Y4[1],"y",label='4ème voiture')
plt.plot(Y5[0],Y5[1],"c",label='5ème voiture')
```

```
plt.legend()
plt.title("Influence d'une décélération de la 1ère Voiture")
plt.xlabel("axe des temps (en s)")
plt.ylabel("x(t) en m")
plt.show()


def h(t):
    if t>=0 and t<=50:
        return 25*t+208
    elif t>50 :
         return -0.05*((t-50)**2)+25*(t-50)+1250+208


def s(t):
    if t>=0 and t<=50:
        return 25*t+280
    elif t>50 and t<=80 :
        return -0.4*((t-50)**2)+25*(t-50)+1250+280
    elif t>80 :
        return   10*(t-80)+1640+280

# 2ème Simulation :
M=[i for i in range(550)]

```

```
Y6=Resol_1_(g,1,5*(10**-3),550,210,220)
P=Y6[0]
Y7=[s(x) for x in M]
Y8=Resol_2_(Y6[1],1,5*(10**-3),550,140,150)
Y9=Resol_2_(Y8[1],1,5*(10**-3),550,70,80)
Y10=Resol_2_(Y9[1],1,5*(10**-3),550,0,8)

plt.plot(N,Y7,"k",label='1ère voiture')
plt.plot(Y6[0],Y6[1],"r",label='2ème voiture')
plt.plot(Y8[0],Y8[1],"b",label='3ème voiture')
plt.plot(Y9[0],Y9[1],"y",label='4ème voiture')
plt.plot(Y10[0],Y10[1],"c",label='5ème voiture')
plt.legend()
plt.title("Influence d'une décélération de la 1ère Voiture")
plt.xlabel("axe des temps (en s)")
plt.ylabel("x(t) en m")
plt.show()
```

```python
# Fonctions utilisées sur le modèle de LWR :
# Feu tricolore :
def f_2(x):
    if x<=0 :
        return 1
    else :
        return 0


def f_3(x,v,x0):
    if x<=-x0 :
        return (-x-x0)/v
    elif x>=x0 :
        return (x-x0)/v
    elif x<x0 and x>-x0 :
        return 0


def f_4(x,e,gho):
    if x<=0 :
        return gho
    elif x<e and x>0 :
        return gho*(1-(x/e))
    else :
        return 0

```

```python
def f_5(f,L,v,x0):
    M=[]
    for c in L :
        M.append(f(float(c),v,x0))
    return M

def f_6(f,L):
    M=[]
    for c in L :
        M.append(f(float(c)))
    return M

# Tracage des fonctions :
L1=np.linspace(-50,50,2001)
L2=np.linspace(-10,0,11)
L3=np.linspace(0.0000001,10,11)
M2=f_6(f_2,L2)
M3=f_6(f_2,L3)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
ax.spines['bottom'].set_position('center')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
```

```
ax.xaxis.set_ticks_position('bottom')
ax.yaxis.set_ticks_position('left')
plt.plot(L2,M2,'k')
plt.plot(L3,M3,'k')
plt.title("Condition initiale h(x)")
plt.xlabel("x")
plt.grid()
plt.show()

M4=f_5(f_3,L1,2,0)
M5=f_5(f_3,L1,2,10)
M6=f_5(f_3,L1,2,20)
M7=f_5(f_3,L1,2,30)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.yaxis.set_ticks_position('left')
ax.set_xticklabels([])
ax.set_yticklabels([])
```

```
plt.plot(L1,M4,'k')
plt.plot(L1,M5,'k')
plt.plot(L1,M6,'k')
plt.plot(L1,M7,'k')
plt.xlabel("x")
plt.ylabel("t")
plt.show()


L8=np.linspace(-50,0,100)
L9=np.linspace(0.00001,20,100)
L10=np.linspace(20.00001,50,100)
M8=f_5(f_4,L8,20,10)
M9=f_5(f_4,L9,20,10)
M10=f_5(f_4,L10,20,10)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.yaxis.set_ticks_position('left')
ax.set_xticklabels([])
```

```
ax.set_xticklabels([])
ax.set_yticklabels([])
plt.plot(L8,M8,'k')
plt.plot(L9,M9,'k')
plt.plot(L10,M10,'k')
plt.xlabel('0')
plt.show()


M11=f_5(f_3,L1,2,0)
M12=f_5(f_3,L1,2,10)
M13=f_5(f_3,L1,2,20)
M14=f_5(f_3,L1,1,0)
M15=f_5(f_3,L1,0.5,0)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
```

```
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.yaxis.set_ticks_position('left')
ax.set_xticklabels([])
ax.set_yticklabels([])
plt.plot(L1,M11,'k')
plt.plot(L1,M12,'k')
plt.plot(L1,M13,'k')
plt.plot(L1,M14,'k')
plt.plot(L1,M15,'k')
plt.xlabel("x")
plt.ylabel("t")
plt.show()
```

```
# Embouteillage :
def f_7(x):
    if x<0:
        return 1/8
    else :
        return 1

L11=np.linspace(-50,50,2001)
L12=np.linspace(-10,-0.00000001,11)
L13=np.linspace(0.0000001,10,11)
M16=f_6(f_7,L12)
M17=f_6(f_7,L13)
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
ax.spines['left'].set_position('center')
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.spines['bottom'].set_position('zero')
ax.yaxis.set_ticks_position('left')
plt.plot(L12,M16,'k')
plt.plot(L13,M17,'k')
plt.title("Condition initiale h(x)")
plt.xlabel("x")
plt.grid()
plt.show()
```