Projet de volumes finis 1

Modélisation du trafic routier

novembre 2020

CROGUENNEC Guillaume

DUPONT Ronan

Table des matières

I) Introduction

Le trafic routier peut être modélisé de nombreuses manières, que ce soit de manière discrète ou continue. Cela peut permettre de comprendre, par exemple, les phénomènes d'apparition et de résorption des bouchons. Et grâce à cela on peut améliorer les outils de prédiction de la circulation.
Mais le fait qu'il y ait des discontinuités dans les flux de circulations rend la modélisation de ce problème particulière.

II) Modélisation

Dans le modèle de trafic routier choisi, l'inconnue est la fonction u qui représente la densité de véhicules en fonction de l'abscisse x et du temps t.
Le problème du transport est modélisé par le système hyperbolique d'équations suivant :

$$\begin{cases} u_t + (f(u))_x = 0, \ \forall x \in R, \forall t > 0 \\ u(x,0) = u_0(x), \ \forall x \in R \end{cases} \tag{1}$$

La première équation, appelée équation de transport, est l'égalité entre la dérivée temporelle de la densité et la dérivée spatiale du flux, car lorsque en un point la quantité de "matière" (ici les véhicules) varie, c'est qu'elle s'est déplacé à côté. Autrement dit, c'est dû aux continuités spatiales et temporelles. Et cela modélise le transport des véhicules.
Dans ce modèle, on a le flux :

$$f(u) = v_{max} u \left(1 - \frac{u}{u_{max}}\right)$$

Avec u_{max} la densité maximale possible et v_{max} la vitesse maximale autorisée.
Le flux est un débit, c'est pour cela qu'il est homogène à une vitesse multipliée à une densité. Et le modèle est fait pour qu'il soit nul lorsque la densité est nulle ou maximale, et qu'il respecte les comportements des conducteurs supposées linéaires.

III) Modèle numérique

Pour résoudre numériquement ce système d'équations, on utilise le schéma numérique des volumes finis qui est conservatif :

$$\boxed{\mathrm{U}_i^{n+1} = U_i^n - \frac{\delta t}{\delta x}\left(F_{i+1/2}(U_i^n, U_{i+1}^n) - F_{i-1/2}(U_{i-1}^n, U_i^n)\right)} \tag{2}$$

Dans ce schéma, $F_{i-1/2}$ et $F_{i+1/2}$ sont les flux discrets à la coordonnée x_i. C'est-à-dire que le premier est le flux entre x_{i-1} et x_i alors que le second est le flux entre x_i et x_{i+1}. On a donc la nouvelle densité à un endroit qui vaut la densité précédente moins le flux sortant plus le flux entrant, qui sont rendu homogènes grâce aux pas temporel et spatial.
Pour calculer leurs valeurs, on va utiliser trois solveurs différents afin de les comparer :

- Le solveur de Godunov
- Le solveur de Lax
- Le sloveur de Murmen Roe

De plus, un pas de temps adaptatif est utilisé afin de pouvoir être plus précis quand il le faut sans augmenter la précision pour rien quand ce n'est pas nécessaire. On le définit par :

$$\boxed{\delta t = \delta x \frac{\alpha}{\max(|f'(Un))|}}$$

On remarque que ce pas de temps dépend du pas spatial pour respecter la CFL, et qu'il est adaptatif car il dépend aussi de la dérivée du flux. Enfin, il permet d'avoir des résultats plus ou moins précis car il dépend d'un alpha que l'on choisit.

1) Le solveur de Godunov

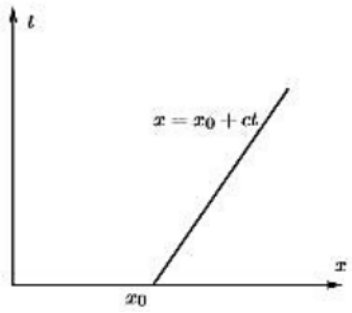
Le premier solveur utilisé est celui de Godunov :

$$F_{i+1/2}(u,v) = F\left(w^*(\frac{x}{t}=0;(u,v))\right)$$

Où on peut faire une disjonction de cinq cas de discontinuité pour obtenir la valeur de w^*. Dans tous les cas, on appelle U_g la valeur de u à gauche et U_d la valeur de u à droite.
Quand il y a une discontinuité cela peut être un choc, si $f'(U_g) > f'(U_d)$, ou une détente si $f'(U_g) < f'(U_d)$. En effet, si le ralentissement du flux est plus grand avant la discontinuité qu'après alors la discontinuité va juste se décaler car les voitures derrière seront "bloquées" par celles devant. Et au contraire si le ralentissement est moindre avant la discontinuité, alors les voitures vont pouvoir "accélérer" pour rattraper celles devant. Dans notre cas, $f'(u) = v_{max} - \frac{2.u.v_{max}}{u_{max}}$, donc simplement si $U_g < U_d$ c'est un choc, et si l'inégalité est dans l'autre sens c'est une détente.

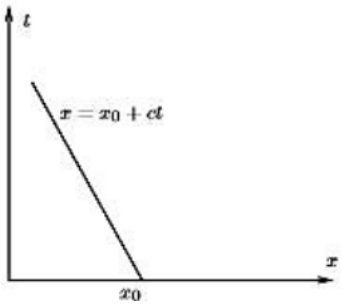
Dans le cas d'un choc, on a la pente de la caractéristique qui vaut $\boxed{c = \frac{F(U_d)-F(U_g)}{U_d-U_g}}$. Ce qui nous permet de distinguer deux cas :

— le 1er cas où $c > 0$



Dans ce cas $\boxed{w^* = U_g}$.

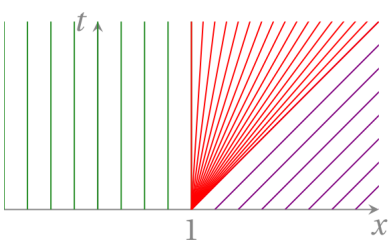
— le 2ème cas où $c < 0$



Dans ce cas $\boxed{w^* = U_d}$.

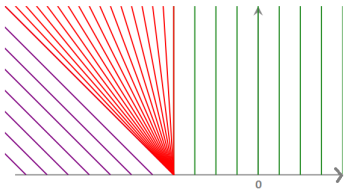
Dans le cas d'une détente, il y a un cône de détente entre $f'(U_g)$ et $f'(U_d)$. Et suivant la positivité de ce cône, on peut donc distinguer trois cas différents :

— le 3ème cas : le cône de détente est du côté négatif, donc $0 < f'(U_g) < f'(U_d)$



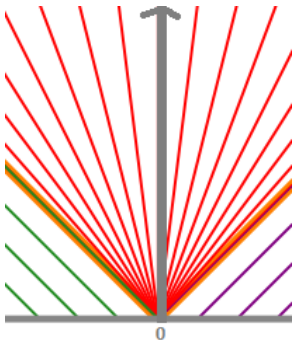
Dans ce cas $\boxed{w^* = U_g}$.

— le 4ème cas : le cône de détente est du côté positif, donc $f'(U_g) < f'(U_d) < 0$



Dans ce cas $\boxed{w^* = U_d}$.

— le 5ème cas : le cône de détente est à cheval entre les côtés négatif et positif, donc $f'(U_g) < 0 < f'(U_d)$



Dans ce cas, on a $w^* = f'^{-1}(0)$.

Or $f(u) = v_{max}u\left(1 - \frac{u}{u_{max}}\right)$ ie $f'(u) = v_{max} - 2u\frac{v_{max}}{u_{max}}$.

En posant $y = f'(u)$, on obtient $u = \frac{u_{max}}{2} - y\frac{u_{max}}{2.v_{max}}$ ie $f'^{-1}(u) = \frac{u_{max}}{2} - u\frac{u_{max}}{2.v_{max}}$

Donc $\boxed{w^* = \frac{u_{max}}{2}}$.

2) Le solveur de Lax

Ensuite on utilise le solveur de Lax :

$$F_{i+1/2}(u,v) = \frac{1}{2}[F(u) + F(v)] - \frac{1}{2}D(v-u) \; avec \; D = \frac{\delta x}{\delta t} \; si \; u \neq v$$

Il modélise le flux discret entre deux points comme la moyenne du flux en ces deux points moins la moyenne de la densité en ces deux points(rendue homogène grâce aux pas temporel et spatial). Cela donne un schéma conservatif d'ordre 1 en espace et en temps.

3) Le sloveur de Murman-Roe

Puis on utilise le solveur de Murman-Roe :

$$F_{i+1/2}(u,v) = \frac{1}{2}(F(u) + F(v)) - \frac{1}{2}|a(v-u)|(v-u) \; avec \; a(u,v) = \begin{cases} \frac{F(u)-F(v)}{u-v} \; si \; u \neq v \\ F'(u) \; si \; u = v \end{cases}$$

Il modélise le flux discret entre deux points comme la moyenne du flux en ces deux points moins la moyenne de la densité en ces points multipliée par la vitesse qui représente la dérivée du flux à cette interface. Mais le problème avec le schéma est qu'il n'est pas entropique. C'est-à-dire qu'à

ce problème il peut y avoir plusieurs solutions, mais une seule sera physique. Or ce schéma ne choisit pas la solution physique plutôt que les autres.

IV) Résultats

Comme expliqué précédemment, lors d'une discontinuité dans le trafic, cela peut créer un choc ou une détente. Donc pour les résultats on choisit de traiter respectivement un cas de choc et un cas de détente. On pose comme conditions initiales :

$$U(x,t=0)=U_0(x)=\begin{cases}0\ si\ x<1\\2\ si\ x>1\end{cases}\qquad U_0(x)=\begin{cases}\frac{1}{2}\ si\ x<1\\0\ si\ x>1\end{cases}$$

On rappelle que le flux vaut :

$$f(u)=v_{max}u\left(1-\frac{u}{u_{max}}\right)$$

Et pour simplifier les calculs on choisit $v_{max}=1$ et $u_{max}=1$.
Donc :

$$f(u)=u(1-u)$$

Et en dérivant :

$$f'(u)=1-2u$$

1) Solutions exactes

- 1) Cas d'un choc : $U(x,t=0)=U_0(x)=\begin{cases}0\ si\ x<1\\2\ si\ x>1\end{cases}$

L'équation de la caractéristique de pied (ξ, 0) est donc :

$$x(t)=\xi+f'(U_0(\xi))t=\xi+(1-2(U_0(\xi)))t=\begin{cases}\xi+t\ si\ x<1\\\xi-3t\ si\ x>1\end{cases}$$

On a donc des caractéristique qui se croisent et forment donc un choc, la pente de la caractéristique s vaut :

$$s'(t)=\frac{f(U_d)-f(U_g)}{U_d-Ug}=\frac{-2-0}{2-0}=-1$$

Or $s(0)=1$ donc $x=s(t)=1-t$

La solution est donc :

$$\boxed{\text{U(x,t)}=\begin{cases}0\ si\ x<1-t\\2\ si\ x>1-t\end{cases}}$$

- 2) Cas d'une détente : $U(x,t=0)=U_0(x)=\begin{cases}\frac{1}{2}\ si\ x<1\\0\ si\ x>1\end{cases}$

L'équation de la caractéristique de pied (ξ, 0) est donc :

$$x(t)=\xi+f'(U_0(\xi))t=\xi+(1-2(U_0(\xi)))t=\begin{cases}\xi\ si\ x<1\\\xi+t\ si\ x>1\end{cases}$$

La solution est donc :

$$\boxed{\text{U(x,t)}=\begin{cases}\frac{1}{2}\ si\ x<1\\\frac{1+t-x}{2t}\ si\ 1<x<1+t\\0\ si\ x>1+t\end{cases}}$$

2) Cas Test

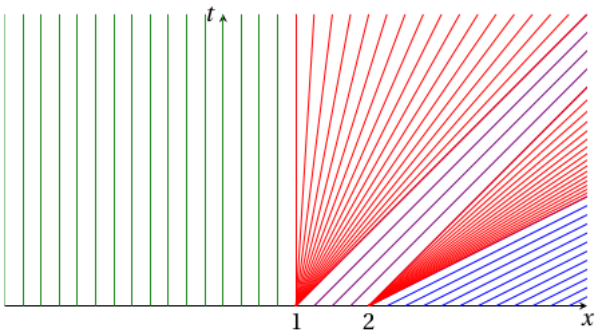
Tout d'abord, pour vérifier que le schéma numérique des volumes finis est suffisamment précis, on compare ses résultats à des résultats connus dans le cas du système d'équation de Burgers (en choisissant arbitrairement le solveur de Murman-Roe). Il s'agit toujours d'une première équation de transport, mais avec $f(u) = \frac{u^2}{2}$.

2).1 Double détente avec l'équation de Burgers

Pour le cas de la détente, on utilise comme deuxième équation (ou condition initiale) :

$$U(x, t = 0) = U_0(x) = \begin{cases} 0 \ si \ x < 1 \\ 1 \ si \ 1 < x < 2 \\ 2 \ si \ x > 2 \end{cases}$$

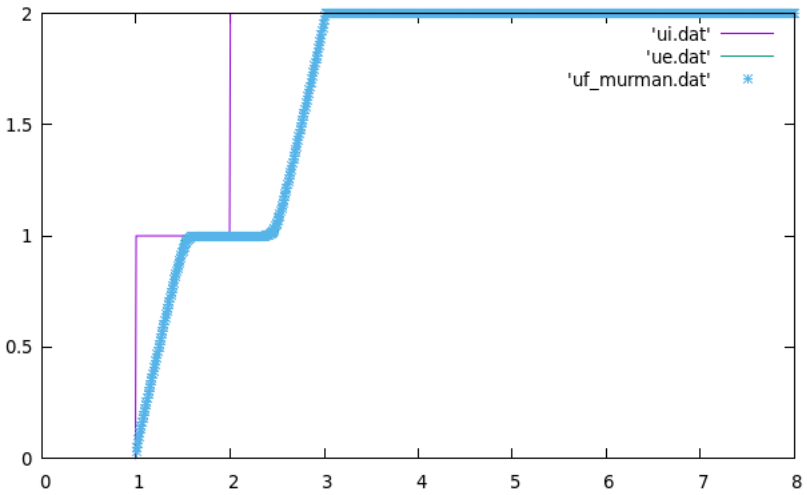
Ce qui donne les caractéristiques suivantes :



Et donc la solution exacte suivante :

$$U(x,t) = \begin{cases} 0 & si \ \ x < 1 \\ \frac{x-1}{t} & si \ \ 1 < x < 1 + t \\ 1 & si \ \ 1 + t < x < 2 + t \\ \frac{x-2}{t} & si \ \ 2 + t < x < 2 + 2t \\ 2 & si \ \ x > 2 + 2t \end{cases}$$

En essayant ce cas avec le solveur de Murman-Roe et 1000 points de discrétisation spatiale on obtient :



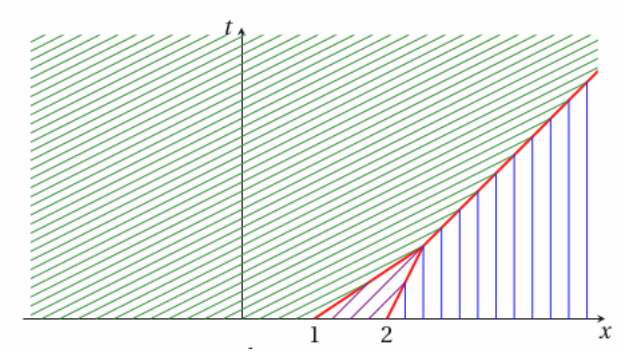
Le résultat obtenu est donc bien confondu avec la solution exacte.

2).2 Double choc avec l'équation de Burger

Ensuite, on essaye aussi le cas d'un double choc pour vérifier l'efficacité du schéma numérique choisi. Dans ce cas, on utilise comme deuxième équation (ou condition initiale) :

$$U(x,t=0) = U_0(x) = \begin{cases} 2 \ si \ x < 1 \\ 1 \ si \ 1 < x < 2 \\ 0 \ si \ x > 2 \end{cases}$$

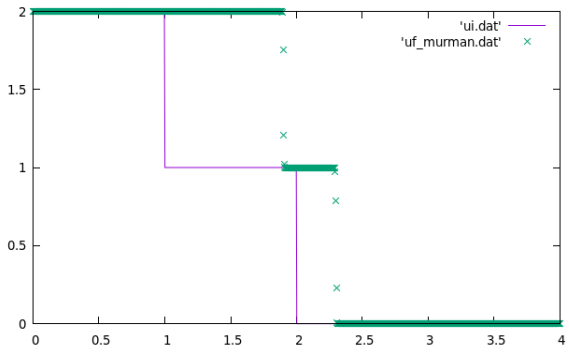
Cela donne les caractéristiques suivantes :



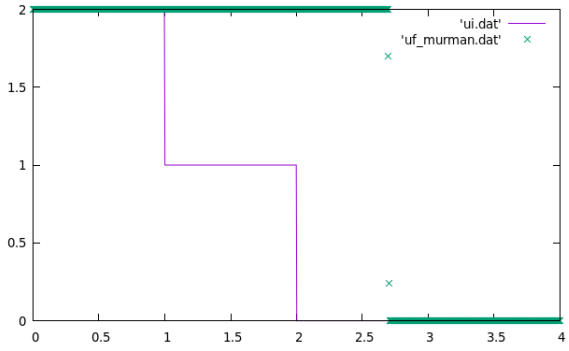
Et le solution exacte est :

$$\underline{Si \ t < 1 :} \ \ U(x,t) = \begin{cases} 2 \ si \ x < 3/2t + 1 \\ 1 \ si \ 3/2t + 1 < x < 1/2t + 2 \\ 0 \ si \ x > 1/2t \end{cases} \quad \underline{Si \ t > 1 :} \ \ U(x,t) = \begin{cases} 2 \ si \ x < t + 3/2 \\ 0 \ si \ x > t + 3/2 \end{cases}$$

En essayant ce cas avec le solveur de Murman-Roe et 1000 points de discrétisation spatiale on obtient : $Tf < 1$:



Pour pour $Tf > 1$



On remarque qu'au début les deux chocs se décalent, puis celui de gauche rattrape celui de droite et il en résulte une seule discontinuité qui se déplace. On conclue donc que le schéma numérique des volumes finis est suffisamment précis.

3) Tracés : cas du choc

Ensuite, on revient sur le cas du trafic routier avec le flux précédent afin de vérifier que tous les solveurs sont des schémas numérique qui conviennent à notre résolution numérique.
Pour le cas du choc, on utilise donc cette condition initiale en prenant $Tf = 0.5s$:

$$U_0(x) = \begin{cases} 0 \ si \ x < 1 \\ 2 \ si \ x > 1 \end{cases} \qquad \boxed{\text{U(x,t)}= \begin{cases} 0 \ si \ x < 1 - t \\ 2 \ si \ x > 1 - t \end{cases}}$$

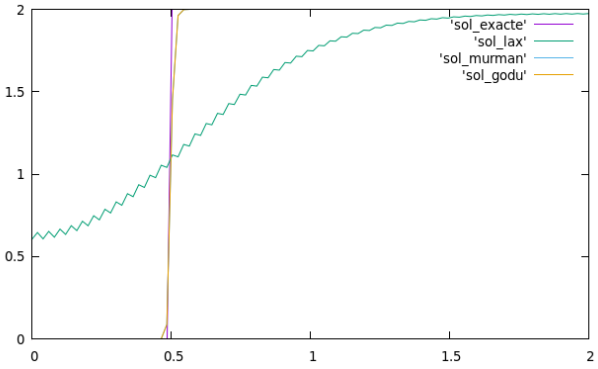


FIGURE 1 – $\alpha = 0.05$ et $n = 100$

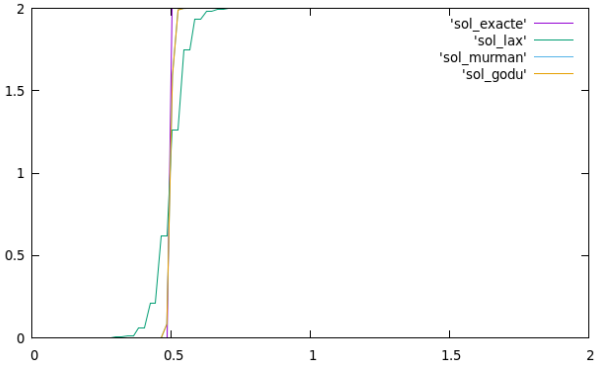


FIGURE 2 – $\alpha = 0.5$ et $n = 100$

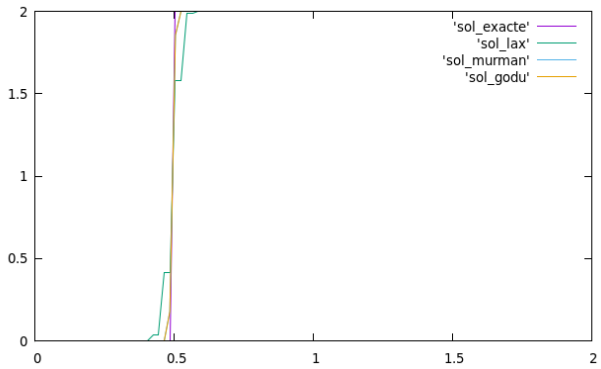


FIGURE 3 – $\alpha = 0.95$ et $n = 100$

On effectue les 3 simulations ci-dessus avec différents α pour un $Tf = 0.5s$ et $n = 100$. Et on remarque que le transport s'effectue vers l'arrière : l'échelon initial était en 1 et se retrouve en 0.5. Ce déplacement "en arrière" est dû à la concavité de la fonction. Et cela a pour conséquence que la pente de la caractéristique est négative.
On remarque que les solutions de Godunov et Murman sont sensiblement les mêmes, elles sont plus précises que la solution de Lax. En effet, on remarque que pour des α petits, le sloveur de Lax est très imprécis. Par exemple, sur la simulation ci-dessous, on prend un $\alpha = 0.2$ et on voit que le modèle est non viable. Il est viable à partir de $\alpha = 0.5$.

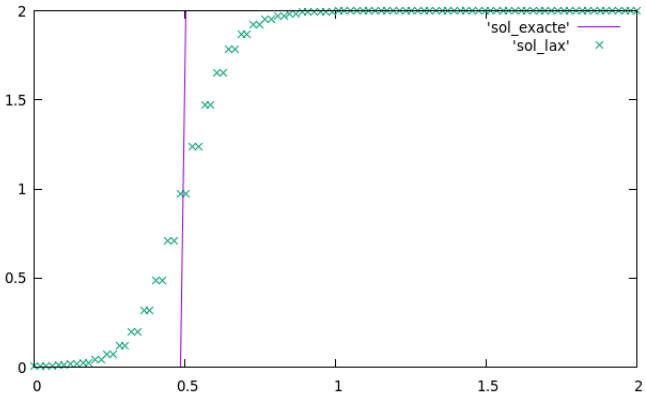


FIGURE 4 – Lax avec $\alpha = 0.2$ et $n = 100$

Grâce à toutes ces simulations, on remarque visuellement que la solution est plus précise quand α est proche de 1 (sans le dépasser sinon la CFL ne sera plus respectée).

4) Tracés : cas de la détente

Pour le cas de la détente, on utilise ces conditions initiales avec 1s comme temps final :

$$U_0(x) = \begin{cases} 0.5 \ si \ x < 1 \\ 0 \ si \ x > 1 \end{cases} \qquad \boxed{\text{U(x,t)}= \begin{cases} \frac{1}{2} \ si \ x < 1 \\ \frac{1+t-x}{2t} \ si \ 1 < x < 1+t \\ 0 \ si \ x > 1+t \end{cases}}$$

Cela correspond à un échelon, ce qui nous permet d'observer comme convenu : une droite plus ou moins inclinée (selon Tf) qui rejoint les deux segments en $y = 0.5$ et $y = 0$.
On choisit ce cas plutôt que certains autres car le solveur de Murman-Roe disfonctionne pour certains cas de détente. En effet n'étant pas entropique il peut ne pas renvoyer la bonne solution.

4).1 Cas $\alpha = 0.05$

On commence par tracer pour différents pas spatiaux le cas où $\alpha = 0.05$.

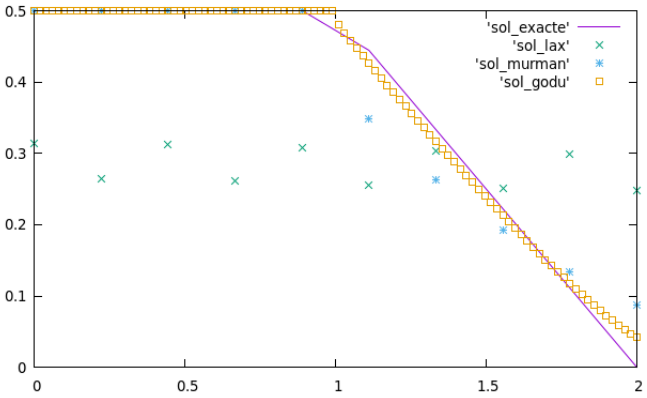


FIGURE 5 – n=10

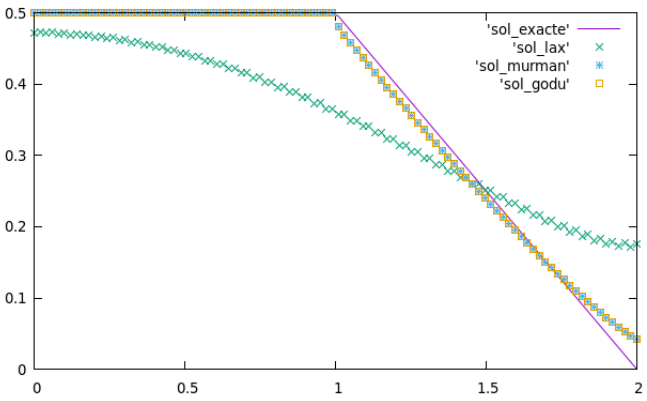


FIGURE 6 – n=100

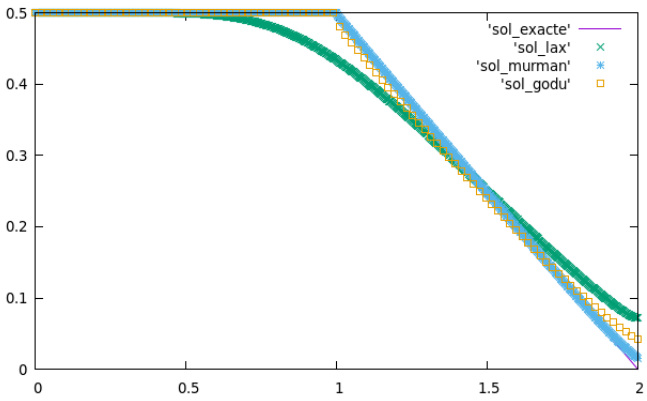


FIGURE 7 – n=1000

Pour $\alpha = 0.05$, on remarque que le Solveur de Lax est vraiment imprécis. Les deux autres solveurs restent assez précis, ils renvoient la même solution.

4).2 Cas $\alpha = 0.5$

Puis on fait de même pour $\alpha = 0.5$.

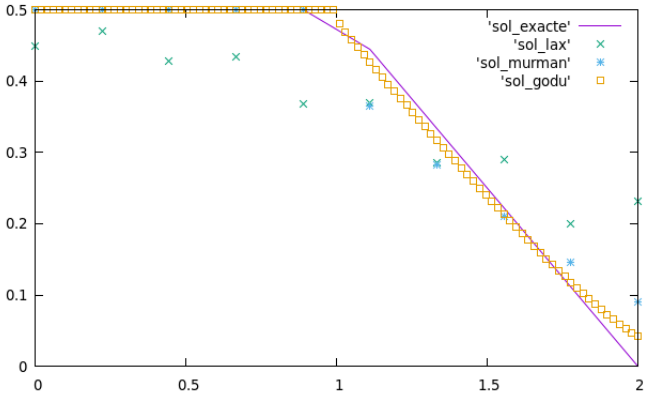


FIGURE 8 – n=10

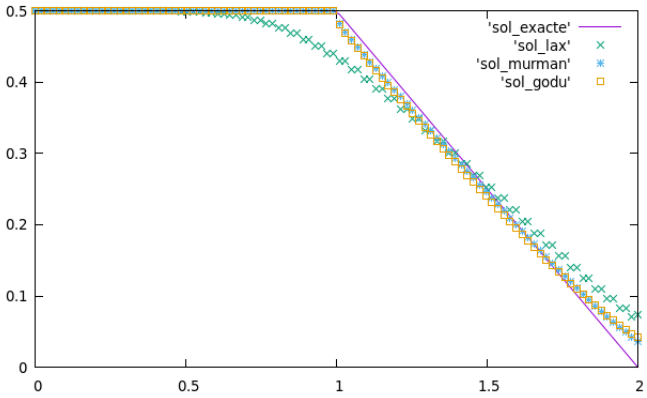


FIGURE 9 – n=100

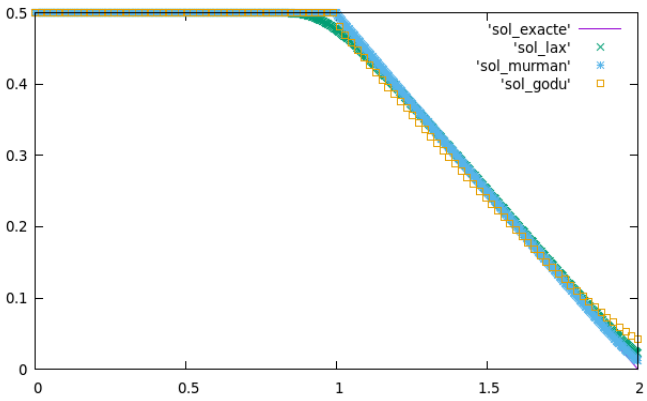


FIGURE 10 – n=1000

Pour $\alpha = 0.5$, on remarque que le Solveur de Lax est toujours imprécis mais bien mieux qu'avec $\alpha = 0.05$. Les deux autres solveurs restent assez précis, ils renvoient toujours la même solution.

4).3 Cas $\alpha = 0.95$

Enfin on réitère encore cela mais avec $\alpha = 0.95$.

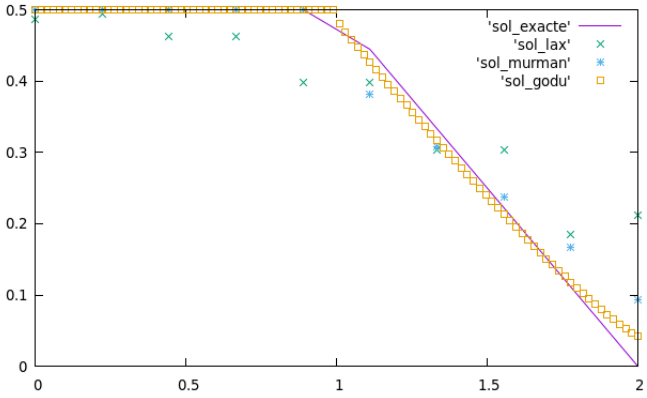


FIGURE 11 – n=10

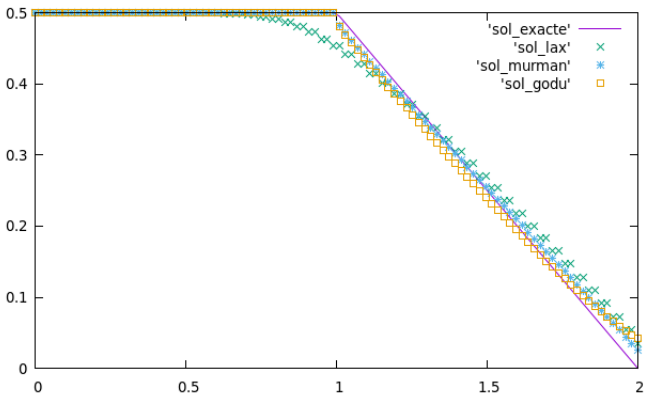


FIGURE 12 – n=100

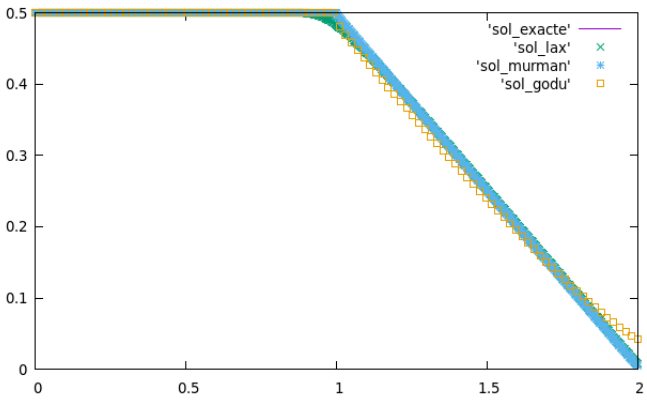


FIGURE 13 – n=1000

Pour $\alpha = 0.95$, on remarque que le Solveur de Lax devient assez précis mais il est toujours moins bon que les deux autres solveurs.

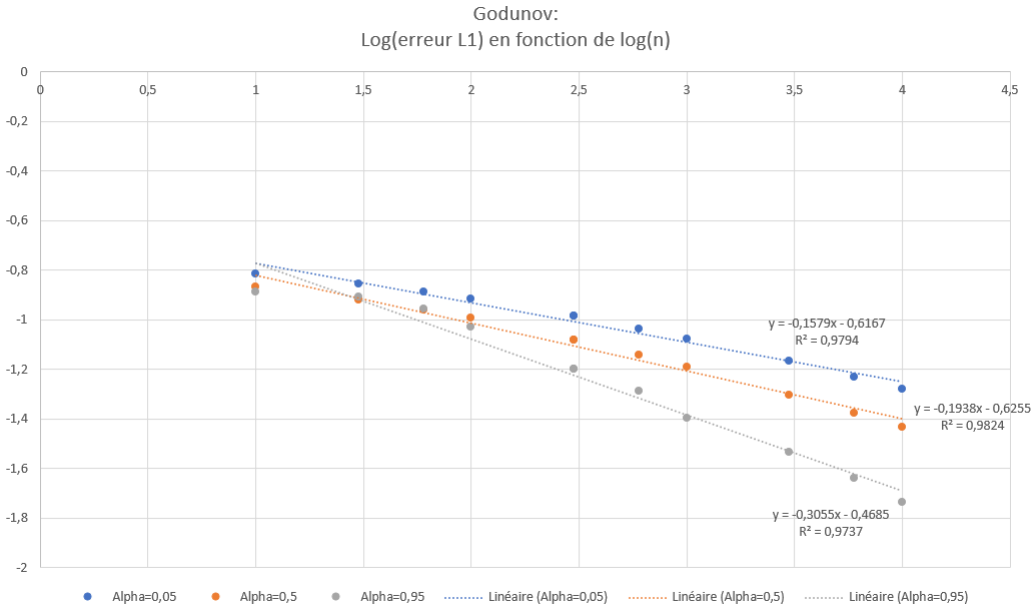
5) Erreur L^1

Afin de connaître la précision des méthodes numériques, il est important de calculer leur erreur L^1 :

$$Erreur = \sum_{i=1}^{n} |U(i) - U_{exact}(i)|$$

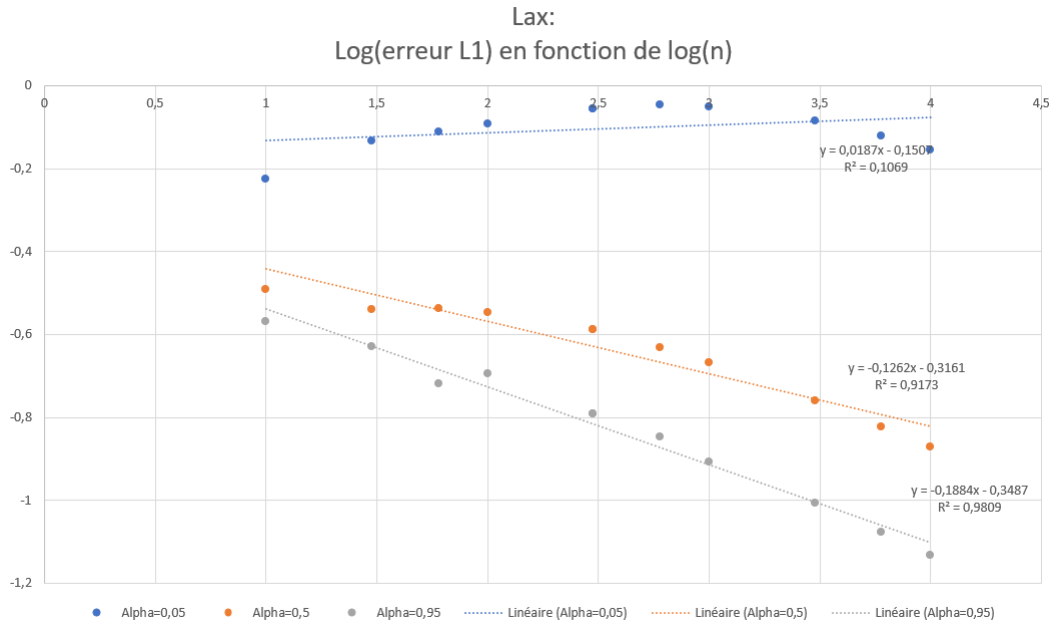
On trace donc pour chaque schéma numérique, et pour différents *alpha*, l'erreur en fonction du nombre de points de subdivision du domaine.

5).1 Godunov



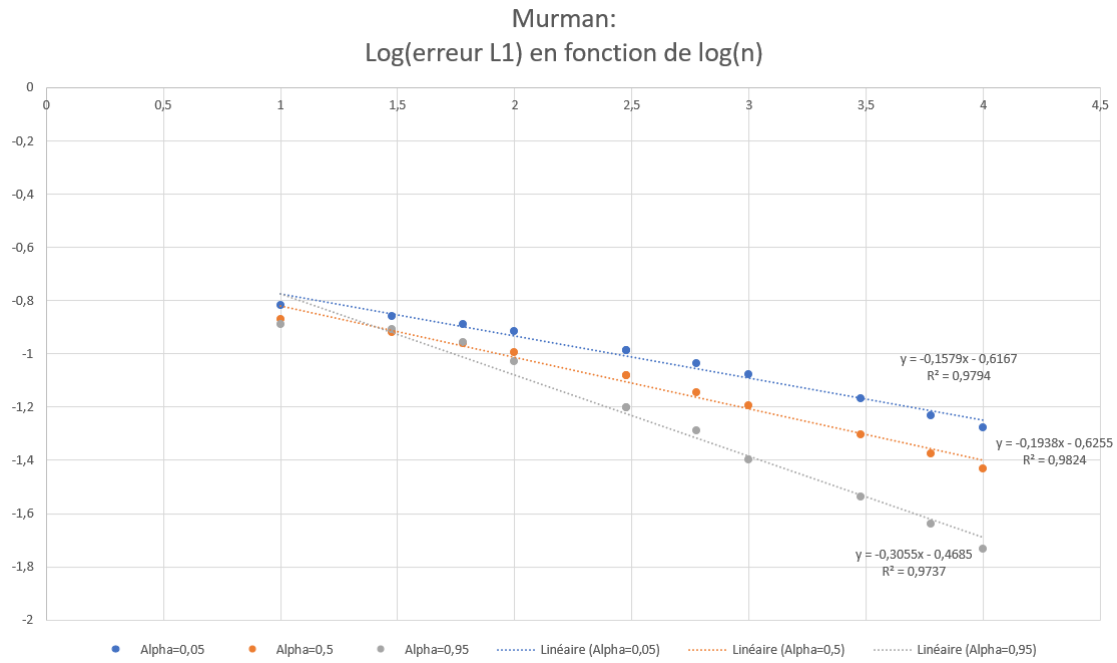
Pour le solveur de Godunov, on remarque qu'il y a convergence quelque soit le α car on a un $R^2 > 0.98$.
Et bien sûr, le plus précis est celui avec $\alpha = 95$.

5).2 Lax



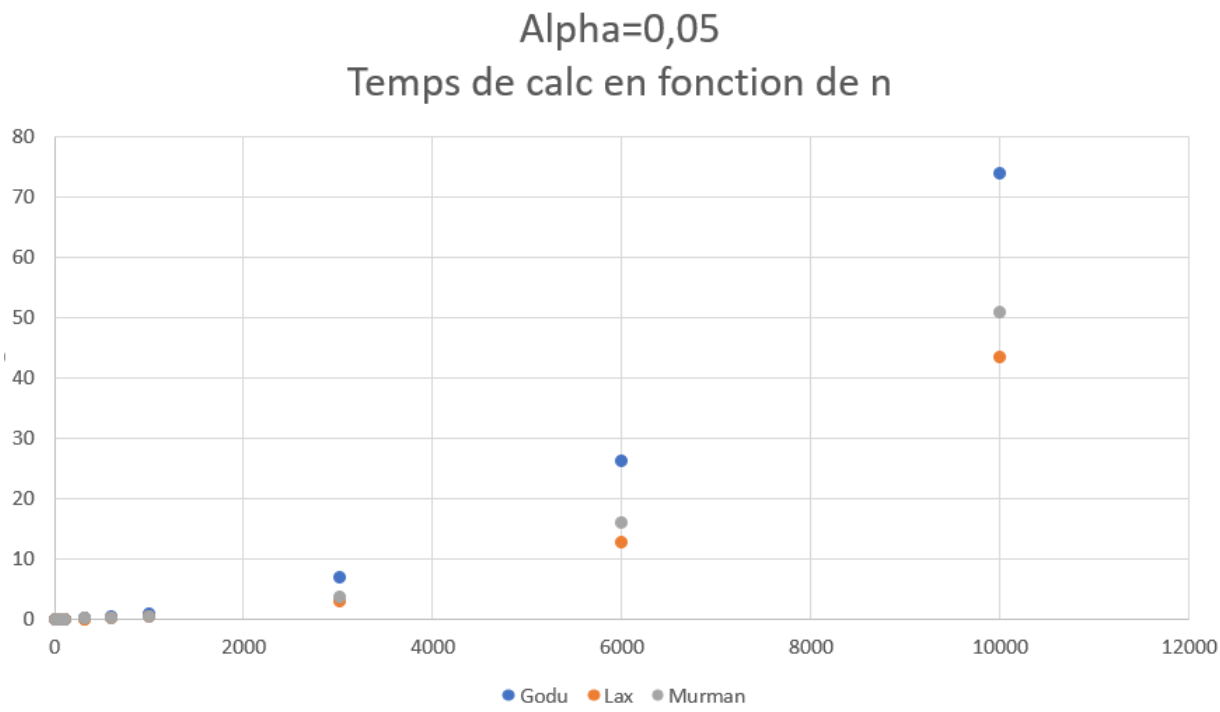
Pour le solveur de Lax, on remarque qu'il y a convergence pour les cas $\alpha = 0.95$ et $\alpha = 0.5$. Mais le coefficient de régression linéaire pour $\alpha = 0.05$ est positif donc il n'y a pas convergence vers la solution exacte.
Ce solveur est moins précis que ceux de Godunov et Murman donc son coefficient de régression linéaire est supérieur aux deux autres.

5).3 Murman-Roe

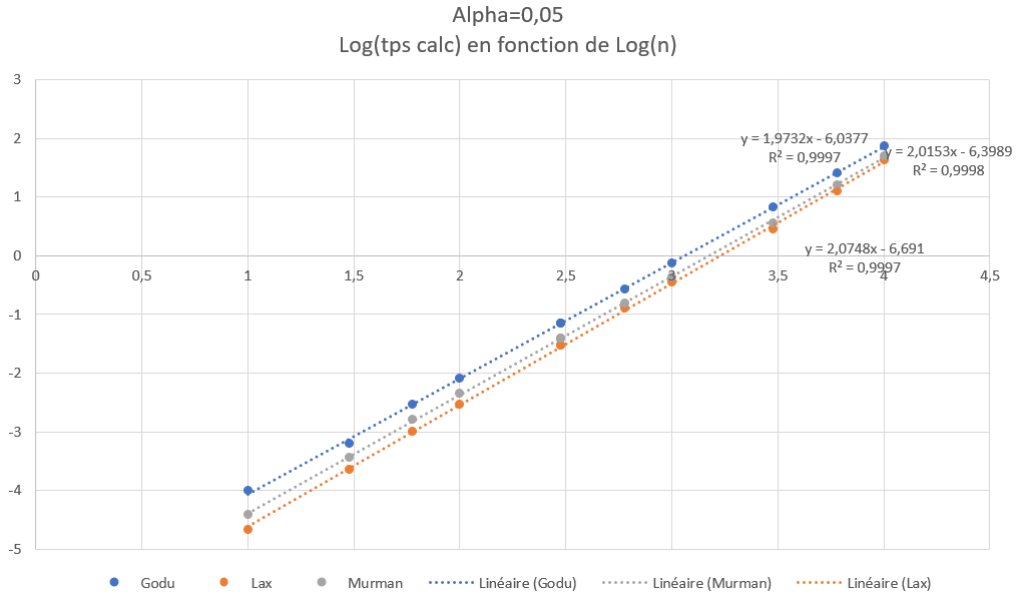


Quant au solveur de Murman, il renvoie exactement la même erreur que le solveur de Godunov. Mais bien sûr, seulement quand on évite les cas de détente où il ne renvoie pas la solution physique.

6) Complexité temporelle



En traçant le temps de calcul en fonction de n, on obtient la courbe ci-dessus. On peut donc le tracer de manière logarithmique pour obtenir la complexité temporelle.



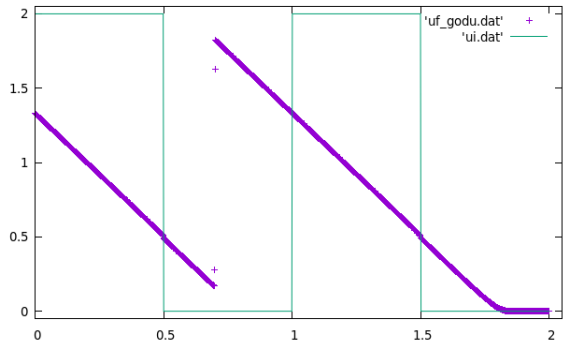
On remarque que le solveur le plus rapide est celui de Lax. Et le plus lent est celui de Godunov car il appel beaucoup de IF qui sont coûteux en temps de calculs. Tous ces solveurs ont la même complexité temporelle en o(n^2). En effet, dans nos programmes on a bien deux boulces imbriqués qui justifient la complexité temporelle.

7) Cas réel

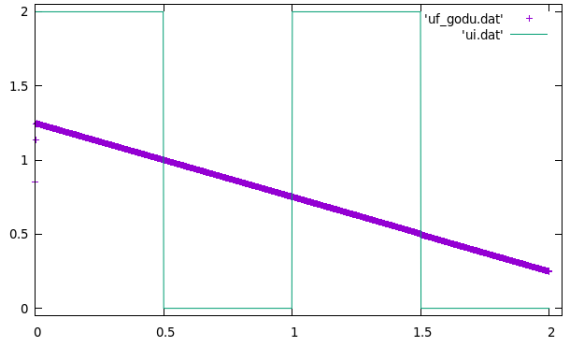
Le but de ces programmes est de simuler un cas réel. Donc on choisit la situation où sur un tronçon de taille L=2, il y a deux feux rouges à $x = 0.5$ et $x = 1.5$.
On modélise ça par les conditions initiales suivantes [1] :

$$U(x, t = 0) = U_0(x) = \begin{cases} 0 \ si \ x \in [0.5, 1] \bigcup [1.5, 2] \\ 2 \ si \ x \in [0, 0.5] \bigcup [1, 1.5] \end{cases}$$

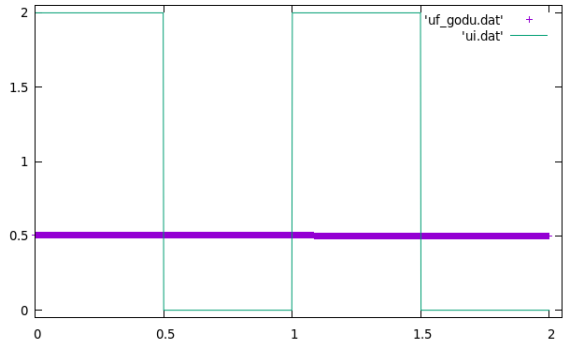
A $Tf = 0.3$ on obtient ce résultat (en violet, avec la fonction initiale en vert) :



A $Tf = 1$ on obtient ce résultat (en violet, avec la fonction initiale en vert) :



On remarque que la densité des voitures commence à s'équilibrer quand les feux passent au vert. C'est-à-dire que l'on a une détente à chaque feu. Donc si on fait tendre Tf vers l'infini, on devrait obtenir une densité moyenne à 0.5, due au mélange de chocs et de détentes. On remarque également que le choc se déplace vers la gauche en raison de la pente de sa caractéristique, mais cela à la densité et la vitesse maximale qui sont un peu spéciales dans notre situation (les deux ont toujours 1 pour valeur).



Pour $Tf = 100$, on a bien quasiment une densité constante à 0.5.

1. Elles ressemblent à un trombone

V) Conclusion

Le modèle repose sur le principe de conservation et considère que les comportements des véhicules sont linéaires et que le flux ne dépend que de la densité. Mais malgré ces approximations, on obtient des résultats totalement cohérents dans tous les cas testés.
Quant aux schémas, celui de Lax-Friedrichs est le moins précis mais le plus rapide. Donc il peut être utilisé dans des cas où on veut rapidement une solution qui n'a pas besoin d'être précise. Le schéma de Murman-Roe est très rapide et très précis, mais il présente des problèmes dans certaines cas de détentes, et est donc régulièrement inadapté. Enfin le schéma de Godunov est aussi précis que celui de Murman-Roe, et ne présente aucun dysfonctionnement, mais il est le plus coûteux en calculs. Ainsi, c'est ce schéma numérique qu'il faut choisir si on recherche une précision optimale.

Merci de nous avoir lu

VI) Annexe

1) Godunov

```
Program godu
! Petit programme simple illustrant les capacites du FORTRAN 90

!-- Declarations
  implicit none
  integer                :: i,j,ii,jj,iter
  integer                :: n
  real(kind=kind(0.d0)) :: dt,dx,y,umax,vmax,w1,w2,fa,fb,M,t,alpha,maxi,Tf,L,nrm
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un,Uold,Uexact,Ue

n=100
L=2
dx=L/(n-1)
!dx=dx*L
umax=1
vmax=1
Tf=0.3
t=0
alpha=0.99

allocate(Un(n))
allocate(Uold(n))
allocate(Uexact(n))
allocate(Ue(n))
! ------------- Corps du programme -----------------

! Initialisation des listes

DO i=1,n/2
        Un(i)=0
ENDDO
DO i=n/2+1,n
        Un(i)=2
ENDDO

! Ecriture de la u_0
OPEN(UNIT=48,FILE='ui.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Un(i)
ENDDO

! Ecriture de u_exacte
DO i=1,n
        IF ((i-1)*dx<1-Tf) THEN
                Uexact(i)=0
        ELSE
                Uexact(i)=2
        END IF
```

```
ENDDO

OPEN(UNIT=48,FILE='ue.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Uexact(i)
ENDDO


! Corps du programme
DO WHILE (t<Tf)
  Uold(1:n)=Un(1:n)
  call delta_t(Uold,alpha,dx,umax,vmax,maxi,y,n,dt)
  t=t+dt
  call calc_w(Uold(1),Uold(2),umax,vmax,w1)
  call calc_w(Uold(2),Uold(1),umax,vmax,w2)
  call f(w1,umax,vmax,fa)
  call f(w2,umax,vmax,fb)
      Un(1)=Uold(1)-dt*(fb-fa)/dx
  DO i=2,n-1
    call calc_w(Uold(i),Uold(i-1),umax,vmax,w1)
    call calc_w(Uold(i+1),Uold(i),umax,vmax,w2)
    call f(w1,umax,vmax,fa)
    call f(w2,umax,vmax,fb)
    Un(i)=Uold(i)-dt*(fb-fa)/dx
  ENDDO
        call calc_w(Uold(n),Uold(n-1),umax,vmax,w1)
  call calc_w(Uold(n-1),Uold(n),umax,vmax,w2)
  call f(w1,umax,vmax,fa)
  call f(w2,umax,vmax,fb)
  Un(n)=Uold(n)-dt*(fb-fa)/dx
ENDDO

!Calcul d'erreur:
Ue=abs(Un-Uexact)
CALL norm(Ue,n,nrm)

print*," "
print*,"Godu"
print*,"n=",n, " alpha=",alpha
print*,"Erreur: ",nrm
print*," "

! Ecriture de xk dans un fichier sous forme de matrice de temperature
OPEN(UNIT=48,FILE='uf_godu.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Un(i)
ENDDO
CLOSE(48)
!=========================================================
! SUBROUTINES
```

```
CONTAINS

SUBROUTINE f(u,umax,vmax,y)
  real(kind=kind(0.d0)) :: u,umax,vmax,y
  y=u*vmax*(1-u/umax)
  RETURN
END SUBROUTINE f

SUBROUTINE f1(u,umax,vmax,y)
  real(kind=kind(0.d0)) :: u,umax,vmax,y
  y=vmax-2*u*vmax/umax
  RETURN
END SUBROUTINE f1

SUBROUTINE calc_sigma(ud,ug,umax,vmax,sigma)
  real(kind=kind(0.d0)) :: ud,ug,f_ud,f_ug,umax,vmax,sigma
  CALL f(ud,umax,vmax,f_ud)
  CALL f(ug,umax,vmax,f_ug)
  sigma=(f_ud-f_ug)/(ud-ug)
  RETURN
END SUBROUTINE calc_sigma


SUBROUTINE calc_w(ud,ug,umax,vmax,w)
  real(kind=kind(0.d0)) :: ud,ug,sigma,f1_ud,f1_ug,umax,vmax,w
  CALL calc_sigma(ud,ug,umax,vmax,sigma)
  CALL f1(ud,umax,vmax,f1_ud)
  CALL f1(ug,umax,vmax,f1_ug)

  IF (f1_ug>f1_ud) THEN !Cas choc
    IF (sigma>0) THEN
      w=ug
    END IF
    IF (sigma<0) THEN
      w=ud
    END IF
  END IF

  IF (f1_ug<f1_ud) THEN !Cas detente
    IF (f1_ug>0) THEN !Cone a droite
      w=ug
    END IF
    IF ( (f1_ug<0).AND.(f1_ud<0) ) THEN !Cone a gauche
      w=ud
    END IF
    IF ( (f1_ug<0).AND.(f1_ud>0) ) THEN !Cone central
      w=umax/2
    END IF
  END IF
```

```
  IF (f1_ug.eq.f1_ud) THEN
      w=ud
  ENDIF

  RETURN
END SUBROUTINE calc_w

SUBROUTINE delta_t(Un,alpha,dx,umax,vmax,maxi,y,n,dt)
  integer                 :: n
  real(kind=kind(0.d0)) :: alpha,dx,umax,vmax,maxi,y,dt
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un

  CALL f1 (Un(1),umax,vmax,y)
  maxi=abs(y)
  DO i=2,n
      CALL f1(Un(i),umax,vmax,y)
      IF (abs(y)>maxi) THEN
          maxi=abs(y)
      ENDIF
  ENDDO
  dt=dx*alpha/maxi
  RETURN
END SUBROUTINE delta_t

SUBROUTINE norm(Un,n,nrm)
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un
  real(kind=kind(0.d0)) :: nrm
  integer                 :: n
  nrm=0
  do i=1,n
    nrm=nrm+Un(i)**2
  enddo
  nrm=sqrt(nrm)
  RETURN
END SUBROUTINE norm

!=========================================================
! Fin du programme
End program godu
```

2) Lax

```
Program laxeu
! Petit programme simple illustrant les capacites du FORTRAN 90

!-- Declarations
  implicit none
  integer                 :: i,j,ii,jj,iter
  integer                 :: n
  real(kind=kind(0.d0)) :: dt,dx,y,umax,vmax,w1,w2,fa,fb,M,t,alpha,maxi,Tf,L,fc,fun,fde,nrm
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un,Uold,Uexact,Ue
```

```
n=100
L=2
dx=L/(n-1)
!dx=dx*L
umax=1
vmax=1
Tf=0.3
t=0
alpha=0.99

allocate(Un(n))
allocate(Uold(n))
allocate(Uexact(n))
allocate(Ue(n))
! ------------- Corps du programme ------------------

! Initialisation des listes

DO i=1,n/2
        Un(i)=0
ENDDO
DO i=n/2+1,n
        Un(i)=2
ENDDO

! Ecriture de la u_0
OPEN(UNIT=48,FILE='ui.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Un(i)
ENDDO

! Ecriture de u_exacte
DO i=1,n
        IF ((i-1)*dx<1-Tf) THEN
                Uexact(i)=0
        ELSE
                Uexact(i)=2
        END IF
ENDDO

OPEN(UNIT=48,FILE='ue.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Uexact(i)
ENDDO

! Corps du programme
DO WHILE (t<Tf)
!do j=1,15
!  CALL Dt(Un,alpha,dx,umax,vmax,dt)
    Uold(1:n)=Un(1:n)
    call delta_t(Uold,alpha,dx,umax,vmax,maxi,y,n,dt)
```

```
    t=t+dt
    call f(Uold(2),umax,vmax,fa)
    call f(Uold(1),umax,vmax,fb)
    call f(Uold(2),umax,vmax,fc)
        fun=(fa+fb)/2-dx*(Uold(1)-Uold(2))/(dt*2)
        fde=(fb+fc)/2-dx*(Uold(2)-Uold(1))/(dt*2)
    Un(1)=Uold(1)-dt*(fde-fun)/dx
    DO i=2,n-1
        call f(Uold(i-1),umax,vmax,fa)
        call f(Uold(i),umax,vmax,fb)
        call f(Uold(i+1),umax,vmax,fc)
                fun=(fa+fb)/2-dx*(Uold(i)-Uold(i-1))/(dt*2)
                fde=(fb+fc)/2-dx*(Uold(i+1)-Uold(i))/(dt*2)
        Un(i)=Uold(i)-dt*(fde-fun)/dx
    ENDDO
    call f(Uold(n-1),umax,vmax,fa)
    call f(Uold(n),umax,vmax,fb)
    call f(Uold(n-1),umax,vmax,fc)
        fun=(fa+fb)/2-dx*(Uold(n)-Uold(n-1))/(dt*2)
        fde=(fb+fc)/2-dx*(Uold(n-1)-Uold(n))/(dt*2)
    Un(n)=Uold(n)-dt*(fde-fun)/dx
ENDDO

!Calcul d'erreur:
Ue=abs(Un-Uexact)
CALL norm(Ue,n,nrm)

print*,"Lax"
print*,"n=",n, " alpha=",alpha
print*,"Erreur : ",nrm
print*," "

! Ecriture de xk dans un fichier sous forme de matrice de temperature
OPEN(UNIT=48,FILE='uf_lax.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Un(i)
ENDDO
CLOSE(48)
!============================================================
! SUBROUTINES



CONTAINS

SUBROUTINE f(u,umax,vmax,y)
  real(kind=kind(0.d0)) :: u,umax,vmax,y
  y=u*vmax*(1-u/umax)
  RETURN
END SUBROUTINE f
```

```
SUBROUTINE f1(u,umax,vmax,y)
  real(kind=kind(0.d0)) :: u,umax,vmax,y
  y=vmax-2*u*vmax/umax
  RETURN
END SUBROUTINE f1

SUBROUTINE calc_sigma(ud,ug,umax,vmax,sigma)
  real(kind=kind(0.d0)) :: ud,ug,f_ud,f_ug,umax,vmax,sigma
  CALL f(ud,umax,vmax,f_ud)
  CALL f(ug,umax,vmax,f_ug)
  sigma=(f_ud-f_ug)/(ud-ug)
  RETURN
END SUBROUTINE calc_sigma


SUBROUTINE delta_t(Un,alpha,dx,umax,vmax,maxi,y,n,dt)
  integer                :: n
  real(kind=kind(0.d0)) :: alpha,dx,umax,vmax,maxi,y,dt
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un

  CALL f1 (Un(1),umax,vmax,y)
  maxi=abs(y)
  DO i=2,n
      CALL f1(Un(i),umax,vmax,y)
      IF (abs(y)>maxi) THEN
          maxi=abs(y)
      ENDIF
  ENDDO
  dt=dx*alpha/maxi
  RETURN
END SUBROUTINE delta_t

SUBROUTINE norm(Un,n,nrm)
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un
  real(kind=kind(0.d0)) :: nrm
  integer                :: n
  nrm=0
  do i=1,n
    nrm=nrm+Un(i)**2
  enddo
  nrm=sqrt(nrm)
  RETURN
END SUBROUTINE norm

!=============================================================
! Fin du programme
End program laxeu
```

3) Murman

```
Program murman
! Petit programme simple illustrant les capacites du FORTRAN 90
```

```
!-- Declarations
  implicit none
  integer                 :: i,j,ii,jj,iter
  integer                 :: n
  real(kind=kind(0.d0)) :: dt,dx,y,umax,vmax,w1,w2,fa,fb,M,t,alpha,maxi,Tf,L,fc,fun,fde,nrm
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un,Uold,Uexact,Ue

n=100
L=2
dx=L/(n-1)
!dx=dx*L
umax=1
vmax=1
Tf=0.3
t=0
alpha=0.99

allocate(Un(n))
allocate(Uold(n))
allocate(Uexact(n))
allocate(Ue(n))
! ------------- Corps du programme -----------------

! Initialisation des listes

DO i=1,n/2
        Un(i)=0
ENDDO
DO i=n/2+1,n
        Un(i)=2
ENDDO

! Ecriture de la u_0
OPEN(UNIT=48,FILE='ui.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Un(i)
ENDDO

! Ecriture de u_exacte
DO i=1,n
        IF ((i-1)*dx<1-Tf) THEN
                Uexact(i)=0
        ELSE
                Uexact(i)=2
        END IF
ENDDO

OPEN(UNIT=48,FILE='ue.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Uexact(i)
```

```
ENDDO

! Corps du programme
DO WHILE (t<Tf)
!do j=1,15
!  CALL Dt(Un,alpha,dx,umax,vmax,dt)
    Uold(1:n)=Un(1:n)
    call delta_t(Uold,alpha,dx,umax,vmax,maxi,y,n,dt)
    t=t+dt
    call f(Uold(2),umax,vmax,fa)
    call f(Uold(1),umax,vmax,fb)
    call f(Uold(2),umax,vmax,fc)
    if (Uold(1)/=Uold(2)) then
            fun=(fa+fb)/2-abs((fb-fa)/(Uold(1)-Uold(2)))*(Uold(1)-Uold(2))/2
            fde=(fb+fc)/2-abs((fc-fb)/(Uold(2)-Uold(1)))*(Uold(2)-Uold(1))/2
    endif
    if (Uold(1)==Uold(2)) then
            fun=(fa+fb)/2
            fde=(fb+fc)/2
    endif
    Un(1)=Uold(1)-dt*(fde-fun)/dx
    DO i=2,n-1
        call f(Uold(i-1),umax,vmax,fa)
        call f(Uold(i),umax,vmax,fb)
        call f(Uold(i+1),umax,vmax,fc)
        if (Uold(i-1)/=Uold(i)) then
                    fun=(fa+fb)/2-abs((fb-fa)/(Uold(i)-Uold(i-1)))*(Uold(i)-Uold(i-1))/2
        endif
        if (Uold(i-1)==Uold(i)) then
                    fun=(fa+fb)/2
        endif
        if (Uold(i)/=Uold(i+1)) then
                    fde=(fb+fc)/2-abs((fc-fb)/(Uold(i+1)-Uold(i)))*(Uold(i+1)-Uold(i))/2
        endif
        if (Uold(i)==Uold(i+1)) then
                    fde=(fb+fc)/2
        endif
        Un(i)=Uold(i)-dt*(fde-fun)/dx
    ENDDO
    call f(Uold(n-1),umax,vmax,fa)
    call f(Uold(n),umax,vmax,fb)
    call f(Uold(n-1),umax,vmax,fc)
    if (Uold(n-1)/=Uold(n)) then
            fun=(fa+fb)/2-abs((fb-fa)/(Uold(n)-Uold(n-1)))*(Uold(n)-Uold(n-1))/2
            fde=(fb+fc)/2-abs((fc-fb)/(Uold(n-1)-Uold(n)))*(Uold(n-1)-Uold(n))/2
    endif
    if (Uold(n-1)==Uold(n)) then
            fun=(fa+fb)/2
            fde=(fb+fc)/2
    endif
    Un(n)=Uold(n)-dt*(fde-fun)/dx
```

```
ENDDO


!Calcul d'erreur:
Ue=abs(Un-Uexact)
CALL norm(Ue,n,nrm)
print*,"Murman"
print*,"n=",n, " alpha=",alpha
print*,'Erreur: ',nrm
print*," "

! Ecriture de xk dans un fichier sous forme de matrice de temperature
OPEN(UNIT=48,FILE='uf_murman.dat')
DO i=1,n
        WRITE(48,*) (i-1)*dx,Un(i)
ENDDO
CLOSE(48)
!=========================================================
! SUBROUTINES



CONTAINS

SUBROUTINE f(u,umax,vmax,y)
  real(kind=kind(0.d0)) :: u,umax,vmax,y
  y=u*vmax*(1-u/umax)
  RETURN
END SUBROUTINE f

SUBROUTINE f1(u,umax,vmax,y)
  real(kind=kind(0.d0)) :: u,umax,vmax,y
  y=vmax-2*u*vmax/umax
  RETURN
END SUBROUTINE f1

SUBROUTINE calc_sigma(ud,ug,umax,vmax,sigma)
  real(kind=kind(0.d0)) :: ud,ug,f_ud,f_ug,umax,vmax,sigma
  CALL f(ud,umax,vmax,f_ud)
  CALL f(ug,umax,vmax,f_ug)
  sigma=(f_ud-f_ug)/(ud-ug)
  RETURN
END SUBROUTINE calc_sigma


SUBROUTINE delta_t(Un,alpha,dx,umax,vmax,maxi,y,n,dt)
  integer                :: n
  real(kind=kind(0.d0)) :: alpha,dx,umax,vmax,maxi,y,dt
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un

  CALL f1 (Un(1),umax,vmax,y)
```

```
  maxi=abs(y)
  DO i=2,n
      CALL f1(Un(i),umax,vmax,y)
      IF (abs(y)>maxi) THEN
          maxi=abs(y)
      ENDIF
  ENDDO
  dt=dx*alpha/maxi
  RETURN
END SUBROUTINE delta_t

SUBROUTINE norm(Un,n,nrm)
  real(kind=kind(0.d0)), dimension(:),allocatable :: Un
  real(kind=kind(0.d0)) :: nrm
  integer               :: n
  nrm=0
  do i=1,n
    nrm=nrm+Un(i)**2
  enddo
  nrm=sqrt(nrm)
  RETURN
END SUBROUTINE norm
!=========================================================
! Fin du programme
End program murman
```