

El Houssaini Youness

**EDP2 (P. Helluy) :**

**Systmes Hyperboliques**

Master 2 CSMI  
Academic year 2020-2021 - Semester 1

# Table of contents

<b>1 TP1</b>	<b>4</b>
1.1 Résolution numérique de l'équation de transport . . . . .	4
1.1.1 La solution du problème . . . . .	4
1.1.2 Cas de $c < 0$ . . . . .	5
1.1.3 La méthode de Godunov . . . . .	6
1.1.4 Tracer le logarithme de l'erreur en norme $L^1$ . . . . .	7
1.1.5 La condition de CFL . . . . .	8
1.2 Résolution numérique de l'équation de Burgers . . . . .	11
1.2.1 La solution de Lax . . . . .	11
1.2.2 le problème de Riemann pour l'équation de Burgers . . . . .	14
1.2.3 La méthode de Godunov . . . . .	15
1.2.4 La correction MUSCL de van Leer . . . . .	19
<b>2 TP2</b>	<b>21</b>
2.1 Résolution numérique de l'équation de Saint-Venant . . . . .	21
2.1.1 La solution du problème de Riemann pour le modèle de Saint-Venant	21
2.1.2 Programmer le schéma de Godunov . . . . .	26
2.1.3 La condition aux limites d'un bassin . . . . .	27
2.1.4 Évolution de la surface d'eau au cours du temps . . . . .	27
2.1.5 Le schéma de Rusanov . . . . .	34
2.1.6 Décrire le schéma VFRoe et le flux numérique . . . . .	38
2.1.7 Programmer le schéma VFRoe . . . . .	41
2.1.8 Programmer la correction entropique qui utilise le flux de Rusanov aux points soniques . . . . .	42

2.1.9	Comparer avec le schéma Godunov . . . . .	42
2.1.10	Le schéma de Roe . . . . .	42
2.1.11	La correction MUSCL de van Leer . . . . .	42
<b>3</b>	<b>TP3</b>	<b>43</b>
3.1	Rèsolution numérique de l'équation de Saint-Venant en 2D . . . . .	43
3.1.1	Réécrire le système de Saint-Venant 2D . . . . .	43
3.1.2	Le système de Saint-Venant 2D est hyperbolique? . . . . .	44
3.1.3	Le flux numérique $f(w_L, w_R, n)$ de Rusanov en 2D . . . . .	45
3.1.4	Décrire comment utiliser le solveur de Riemann 1D pour le calcul en 2D	45
3.1.5	Conditions aux limites: miroir, valeurs imposées, zone sèche . . . . .	46
3.1.6	La résolution du sytème de Saint-Venant sur un maillage volume fini régulier . . . . .	47
3.1.7	Tester sur le cas 2D avec un fond plat . . . . .	47
3.1.8	La correction MUSCL en 2D . . . . .	47
3.1.9	Facultatif . . . . .	47

# Chapter 1

## TP1

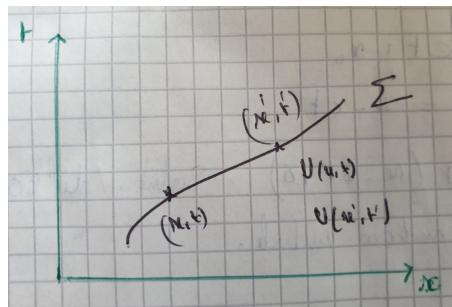
### 1.1 Résolution numérique de l'équation de transport:

Soit  $c > 0$  et le problème de transport suivant:

$$\begin{cases} \partial_t u + c \partial_x u = 0, & x \in ]0, L[, t \in ]0, T[, \\ u(0, t) = e^{-t}, & t \in ]0, T[, \\ u(x, 0) = 0, & x \in ]0, L[, \end{cases} \quad (1.1)$$

#### 1.1.1 La solution du problème:

On utilise la méthode des caractéristiques :



Notons  $\Sigma$  la courbe du plan paramétrée par  $t \mapsto \begin{pmatrix} x(t) \\ t \end{pmatrix}$  telle que  $u$  soit constante le long de  $\Sigma$ .

La fonction  $t \mapsto u(x(t), t)$  est donc constante, donc si tout est régulier sa dérivée exacte en  $t$  est nulle :

$$\frac{du(x(t), t)}{dt} = 0$$

or la différentielle de  $u$  en  $(x(t), t)$  dans la direction  $(dx, dt)$  est:

$$du(x(t), t) = \langle \nabla u, (dx, dt) \rangle = \partial_x u(x(t), t) dx + \partial_t u(x(t), t) dt$$

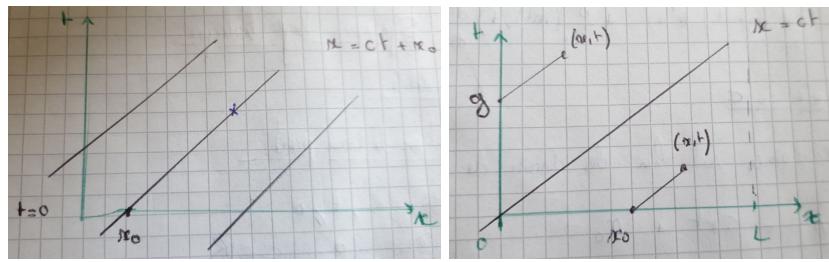
En divisant cette relation par  $dt$ , on obtient :

$$\partial_x u(x(t), t) x'(t) + \partial_t u(x(t), t) = 0$$

Par identification au système (1.1),  $x'(t) = c$ .

Les courbes caractéristiques  $\Sigma$  sont donc les droites  $x(t) = ct + x_0$

On retient  $x(t) = ct + x_0$  et  $x_0 = x(t) - ct$ .



- si  $x \geq ct$  :

Dans ce cas,

$$u(x, t) = u(x_0, 0) = u(x - ct, 0) = 0$$

d'après la deuxième condition initiale.

- si  $x < ct$  :

La courbe caractéristique coupe l'axe des ordonnées, donc

$$u(x, t) = u\left(0, -\frac{x_0}{c}\right) = e^{\frac{x_0}{c}} = e^{\frac{x}{c} - t}$$

d'après la première condition initiale.

On a donc au final :

$$u(x, t) = \begin{cases} 0 & \text{si } x \geq ct \\ e^{\frac{x}{c} - t} & \text{si } x < ct \end{cases} \quad (1.2)$$

### 1.1.2 Cas de $c < 0$ :

Si  $c < 0$ , On suppose que (1.1) a une solution:

Les droites caractéristiques sont inclinées dans l'autre sens et coupe les deux axes.

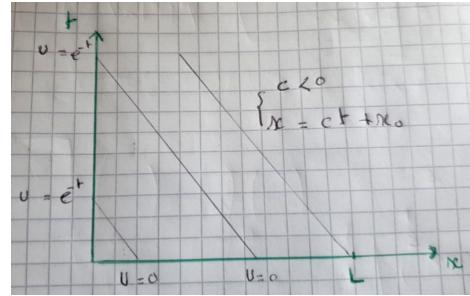
On aurait d'une part:

$$u(x, t) = u(ct + x_0, t) = u(x_0, 0)$$

et d'autre part:

$$u(x, t) = u(ct + x_0, t) = u\left(0, -\frac{x_0}{c}\right)$$

Ceci est présenté dans la figure suivante:



Donc d'après les conditions initiales on a:

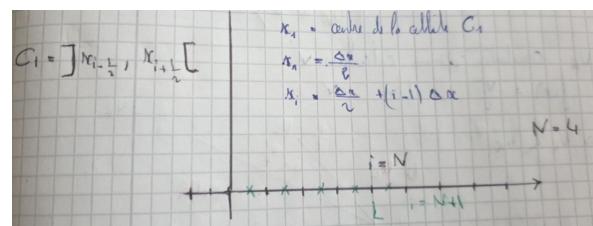
$$0 = e^{\frac{x_0}{c}}$$

ce qui est absurde

Donc si  $c < 0$ , le système n'a pas de solution.

### 1.1.3 La méthode de Godunov:

le schéma de Godunov est basé sur la discritisation suivante:



avec:

$$u_i^n \sim u(x_i, t_n)$$

Donc (1.1)  $\Rightarrow$ :

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{f(u_i^n, u_{i+1}^n) - f(u_{i-1}^n, u_i^n)}{\Delta x} = 0$$

Avec:  $f(u_L, u_R)$  est le flux numérique qui repose sur la résolution du problème de Riemann suivant:

$$\begin{cases} \partial_t v + \partial_x F(v) = 0, & x \in ]0, L[, t \in ]0, T[, \\ v(x, 0) = \begin{cases} u_L & \text{si } x < 0 \\ u_R & \text{si } x > 0 \end{cases} & \end{cases} \quad (1.3)$$

et

Avec dans notre cas (Transport)  $F(u) = cu = u$  (on prend  $c = 1$ ).

Alors la solution est basée sur le même raisonnement qu'en question 1:

$$v(x, t) = \begin{cases} u_R & \text{si } x \geq ct \\ u_L & \text{si } x < ct \end{cases} \quad (1.4)$$

On note

$$v(x, t) = R(u_L, u_R, \frac{x}{t})$$

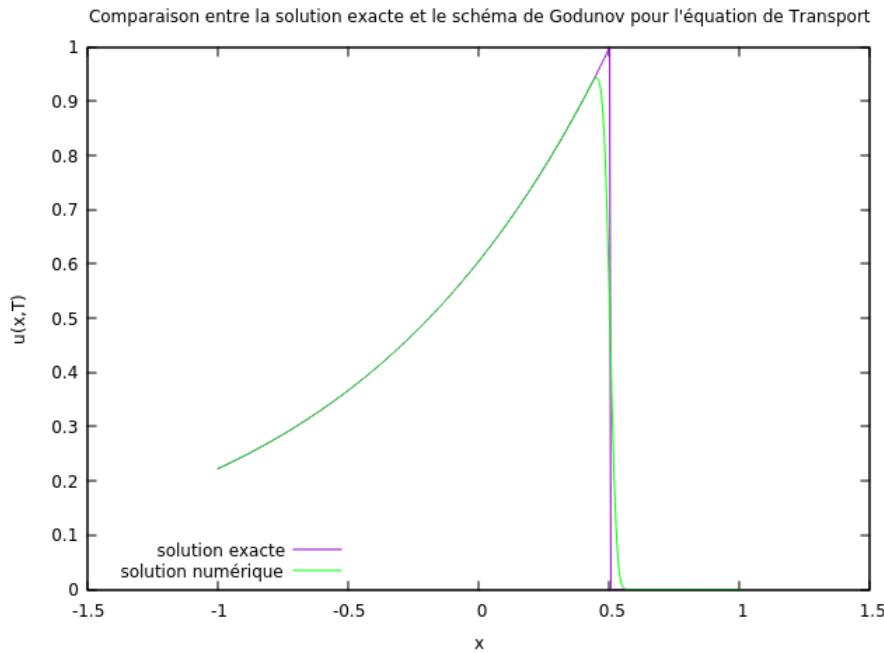
Et on peut en déduire le flux numérique par la formule suivante:

$$f(a, b) = F(R(a, b, 0))$$

#### **1.1.4 Tracer le logarithme de l'erreur en norme $L^1$ :**

Pour la validation du code on choisit une CFL = 0.8 et on visualise la solution et le taux de convergence pour une valeur de T = 0.5.

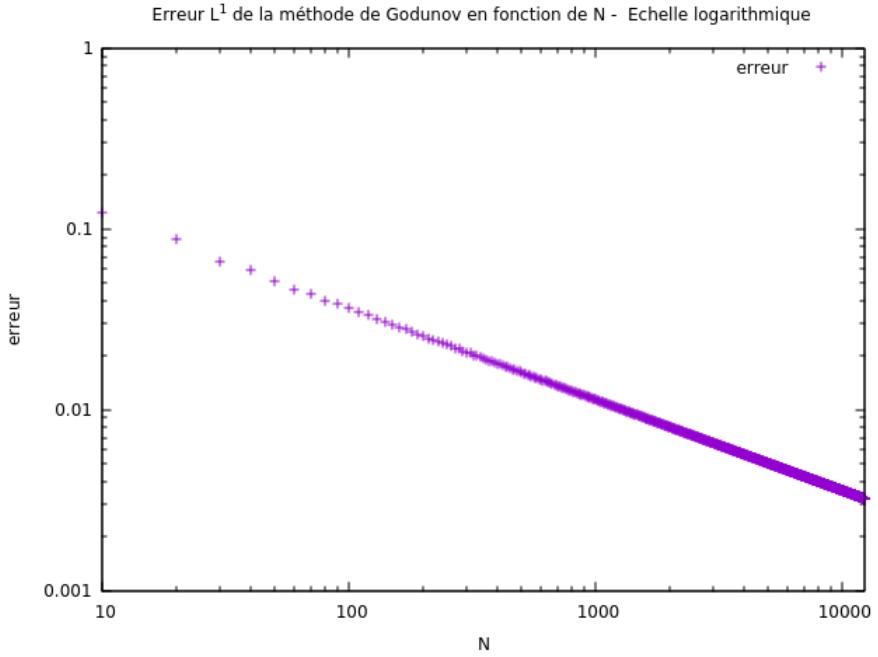
- Solution:



- Erreur:

Voici les résultats obtenus pour différents valeurs de  $N$  :

Plus N est grand donc le pas  $\Delta x$  est petit l'erreur en norme  $L^1$  est petite. Ceci est vrai pour plusieurs valeurs de T.



En partant de la relation théorique de l'erreur en norme  $L^1$ , on a:

$$\text{erreur}(\Delta x) \sim c \times \Delta x^\beta$$

On peut se permettre de faire l'étude de convergence en fonction de N (nombre de points) car  $\Delta x = \frac{x_{\max} - x_{\min}}{N}$  en effet:

$$\text{erreur}(N) \sim c \times N^{-\beta}$$

et en échelle logarithmique:

$$\ln(\text{erreur}(N)) \sim \ln(c) - \beta \times \ln(N)$$

La pente du graphe en dessus est donc le taux de convergence pour ce cas de transport. en effet:

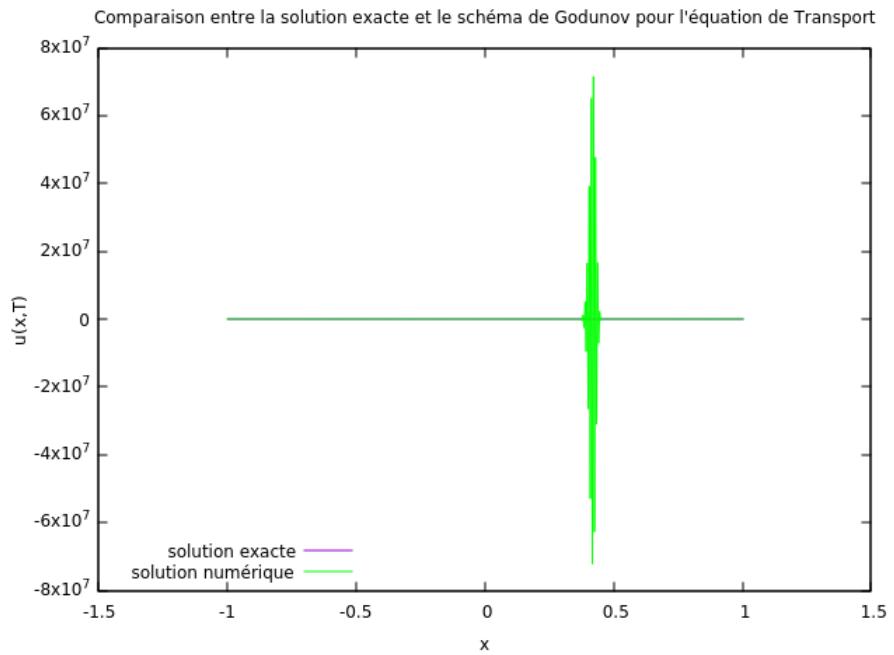
$$\beta = \frac{0.1 - 0.01}{2 \times (0.1 - 0.01)} = \frac{1}{2}$$

### 1.1.5 La condition de CFL:

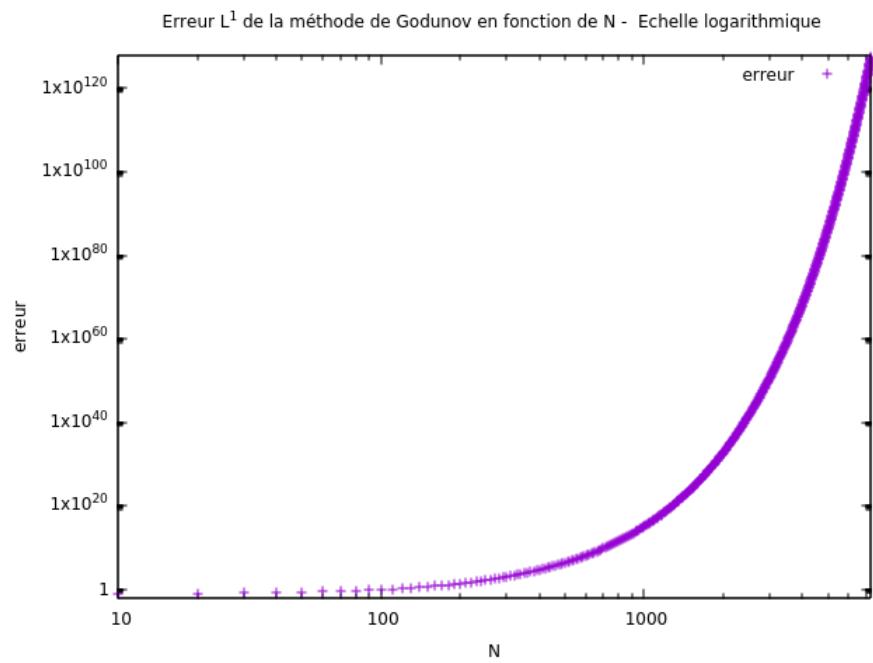
On prend une  $\text{CFL} = 1.1$  telle que la condition de stabilité n'est pas vérifiée.

Et on a pour  $c = 1$ :

$$\frac{\Delta t}{\Delta x} = CFL = 1.1 > 1$$



Et l'erreur en norme L1 en fonction de N:



On observe numériquement que le schéma devient instable lorsque la condition de CFL n'est pas satisfaite.

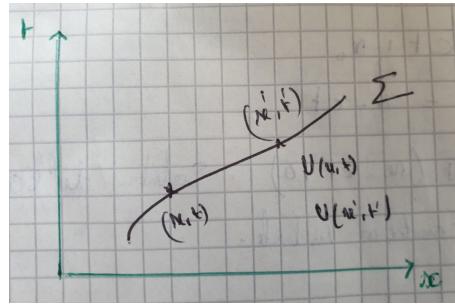
## 1.2 Résolution numérique de l'équation de Burgers:

Soit maintenant  $F(u) = \frac{u^2}{2}$  et on a problème suivant dit de Burgers à résoudre:

$$\left\{ \begin{array}{l} \partial_t u + \partial_x F(u) = 0, \quad F(u) = \frac{u^2}{2} \quad x \in ]-1, 2[, \\ u(0, t) = 1, \quad t \in ]0, T[, \\ u(x, 0) = \begin{cases} 1 & \text{si } x < 0 \\ 1-x & \text{si } 0 \leq x \leq 1, \\ 0 & \text{si } x > 1 \end{cases} \end{array} \right. \quad (1.5)$$

### 1.2.1 La solution de Lax:

On utilise la méthode des caractéristiques :



Notons  $\Sigma$  la courbe du plan paramétrée par  $t \mapsto \begin{pmatrix} x(t) \\ t \end{pmatrix}$  telle que  $u$  soit constante le long de  $\Sigma$ .

La fonction  $t \mapsto u(x(t), t)$  est donc constante, donc si tout est régulier sa dérivée exacte en  $t$  est nulle :

$$\frac{du(x(t), t)}{dt} = 0$$

or la différentielle de  $u$  en  $(x(t), t)$  dans la direction  $(dx, dt)$  est:

$$du(x(t), t) = \langle \nabla u, (dx, dt) \rangle = \partial_x u(x(t), t) dx + \partial_t u(x(t), t) dt$$

En divisant cette relation par  $dt$ , on obtient :

$$\partial_x u(x(t), t) x'(t) + \partial_t u(x(t), t) = 0$$

Par identification au système (1.5), qui peut être réécrit:

$$\partial_t u + F'(u) \partial_x u = 0 \quad \text{avec } F'(u) = u$$

On a:

$$x'(t) = F'(u) = u(x(t), t)$$

Comme  $u(x(t), t)$  est constante par définition de la caractéristique.

Donc:

$$x'(t) = F'(u) = u(x(t), t) = cst$$

Les courbes caractéristiques  $\Sigma$  sont donc des droites mais elles ne sont pas forcément parallèles.

En effet selon les conditions initiales:

$$\left\{ \begin{array}{l} u(0, t) = 1, \quad t \in ]0, T[, \\ u(x, 0) = \begin{cases} 1 & \text{si } x < 0 \\ 1 - x & \text{si } 0 \leq x \leq 1, \\ 0 & \text{si } x > 1 \end{cases} \end{array} \right. \quad (1.6)$$

Pour calculer des solutions, on a introduit la notion des solutions faibles. Une des solutions faibles est donnée par la vitesse de discontinuité dite vitesse de choc.

$$\sigma = x'(t)$$

La valeur de  $\sigma$  doit vérifier le critère de Lax. En effet, un choc de vitesse  $\sigma$  séparant les états  $u_L$  (à gauche) et les états  $u_R$  (à droite) est admissible au sens de Lax si et seulement si:

$$F'(u_L) > \sigma > F'(u_R)$$

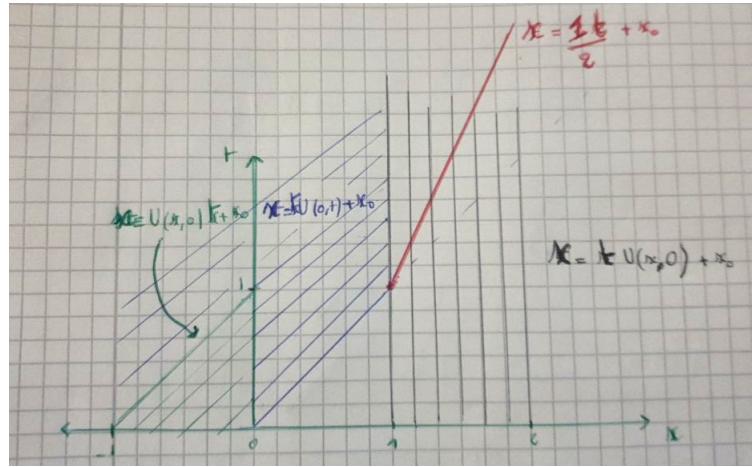
Dans notre cas comme visualisé dans la figure en dessous, on pose une courbe de discontinuité (en rouge) telle que:

$$x(t) = \frac{t}{2} + \frac{1}{2}$$

On retient

$$x'(t) = u(x(t), t)$$

et on trace les droites caractéristiques.



en effet, on distingue donc deux cas:

- cas de choc ou ( $t \geq 1$ ) :

Justement pour ce cas on a introduit la droite de discontinuité qui sépare deux sous cas.

$$u(x, t) = \begin{cases} u_L = 1 & \text{si } x < \frac{t+1}{2} \\ u_R = 0 & \text{sinon} \end{cases} \quad (1.7)$$

On note  $u_L$  à gauche du choc et  $u_R$  à droite du choc.

On a  $u_L = 1$ ,  $u_R = 0$  et  $F'(u) = u$ .

Alors selon la relation de Rankine Hugoniot:

$$\begin{aligned} F(u_R) - F(u_L) &= \sigma(u_R - u_L) \\ \implies \sigma &= x'(t) = \frac{u_L + u_R}{2} = \frac{1}{2} \end{aligned}$$

Qui vérifie le critère de Lax:

$$F'(u_L) = u_L = 1 > \sigma = \frac{1}{2} > F'(u_R) = u_R = 0$$

- cas sans choc ou ( $t < 1$ ):

On a dans ce cas trois sous cas définis selon les conditions initiales.

–  $x < t$  :

les caractéristiques vérifient:

$$x = u(x, t)t + x_0$$

et coupent soit l'axe des abscisses en  $x_0$ , soit l'axe des ordonnées en  $t'$  donc:

$$u(x, t) = u(x_0, 0) = u(0, t') = 1$$

–  $x > 1$  :

les caractéristiques vérifient:

$$x = u(x, t)t + x_0$$

et coupent l'axe des abscisses en  $x_0$ , donc:

$$u(x, t) = u(x_0, 0) = 0$$

–  $1 \geq x \geq t$  :

les caractéristiques vérifient:

$$x = u(x, t)t + x_0$$

alors:

$$u(x, t) = u(x_0, 0) = 1 - x_0$$

Calculons  $1 - x_0$  en remplaçant  $u(x, t)$  par  $x_0$  dans  $x = u(x, t)t + x_0$  :

On a donc:

$$x = (1 - x_0)t + x_0$$

$\implies$

$$x = (1 - x_0)t - (1 - x_0) + 1$$

$\implies$

$$x = (1 - x_0)(t - 1) + 1$$

alors:

$$u(x, t) = u(x_0, 0) = 1 - x_0 = \frac{x - 1}{t - 1}$$

On a donc la solution de Lax:

- $t < 1$  sans choc :

$$u(x, t) = \begin{cases} \frac{x - 1}{t - 1} & \text{si } t \geq x \geq 1 \\ 1 & \text{si } x < t \\ 0 & \text{sinon} \end{cases} \quad (1.8)$$

- $t \geq 1$  avec choc :

$$u(x, t) = \begin{cases} 1 & \text{si } x < \frac{t+1}{2} \\ 0 & \text{sinon} \end{cases} \quad (1.9)$$

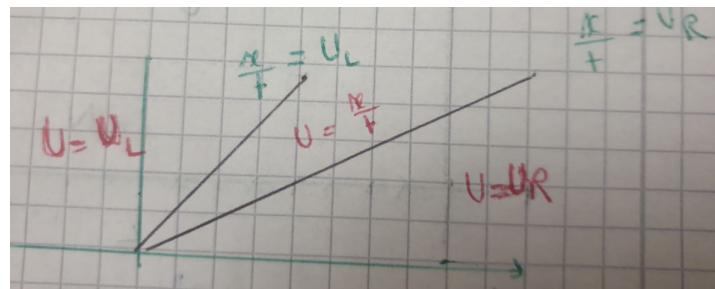
### 1.2.2 le problème de Riemann pour l'équation de Burgers:

Soit le problème de Riemann pour l'équation de Burgers (1.5):

$$\begin{cases} \partial_t v + \partial_x F(v) = 0, \quad F(v) = \frac{v^2}{2} \\ v(x, 0) = \begin{cases} u_L & \text{si } x < 0 \\ u_R & \text{si } x > 0 \end{cases} \end{cases} \quad (1.10)$$

On distingue deux cas possible:

- si  $u_L < u_R$  sans choc :

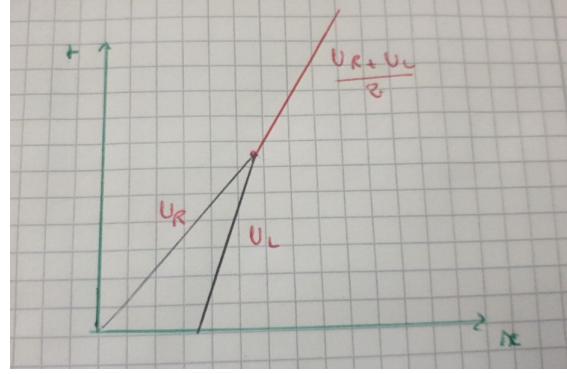


$$v(x, t) = \begin{cases} u_L & \text{si } \frac{x}{t} < u_L \\ u_R & \text{si } \frac{x}{t} > u_R \\ \frac{x}{t} & \text{sinon} \end{cases} \quad (1.11)$$

- si  $u_L < u_R$  avec choc :

Dans ce cas introduit la vitesse de choc donnée par Rankine Hugoniot:

$$\sigma = x'(t) = \frac{u_L + u_R}{2}$$



$$u(x, t) = \begin{cases} u_L & \text{si } \frac{x}{t} < \frac{u_L + u_R}{2} \\ u_R & \text{sinon} \end{cases} \quad (1.12)$$

### 1.2.3 La méthode de Godunov:

La méthode de Godunov reste la même sauf que pour le cas de Burgers on adopte les changements:

- le flux physique:

$$F(u) = \frac{u^2}{2}$$

- le flux numérique:

On utilise la solution de Riemann explicitée en dessus en remplaçant  $\frac{x}{t}$  par 0. Et on note  $R(u_L, u_R, 0)$ :

- si  $u_L < u_R$  sans choc :

$$R(u_L, u_R, 0) = \begin{cases} u_L & \text{si } 0 < u_L \\ u_R & \text{si } 0 > u_R \\ \frac{x}{t} & \text{sinon} \end{cases} \quad (1.13)$$

- si  $u_L < u_R$  avec choc :

$$R(u_L, u_R, 0) = \begin{cases} u_L & \text{si } 0 < \frac{u_L + u_R}{2} \\ u_R & \text{sinon} \end{cases} \quad (1.14)$$

avec le flux numérque devient:

$$F(R(u_L, u_R, 0)) = \frac{R(u_L, u_R, 0)^2}{2}$$

- La solution exacte:

On implémente la solution trouvé dans la question 1 de la partie Burgers:

- $t < 1$  sans choc :

$$u(x, t) = \begin{cases} \frac{x-1}{t-1} & \text{si } t \geq x \geq 1 \\ 1 & \text{si } x < t \\ 0 & \text{sinon} \end{cases} \quad (1.15)$$

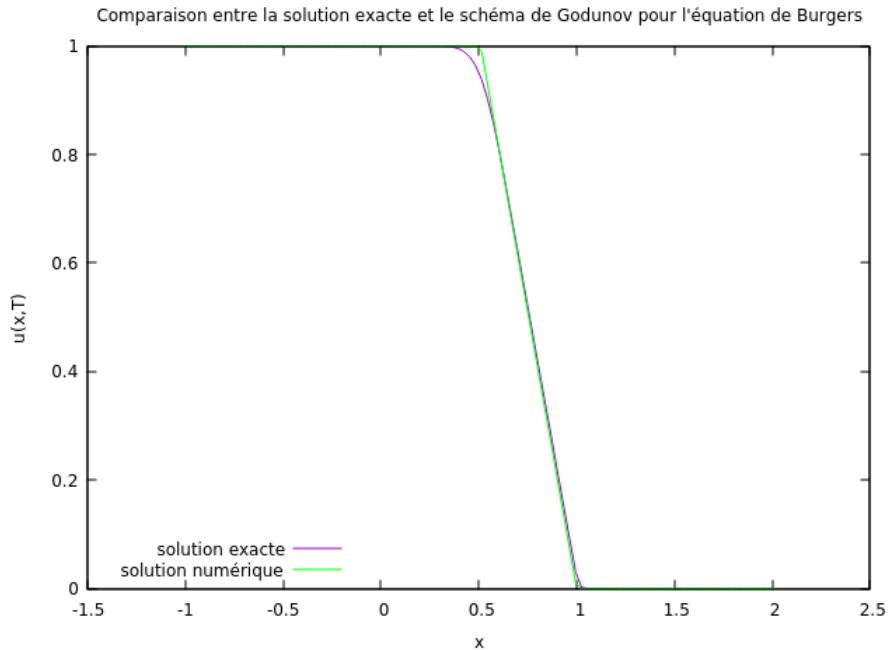
- $t \geq 1$  avec choc :

$$u(x, t) = \begin{cases} 1 & \text{si } x < \frac{t+1}{2} \\ 0 & \text{sinon} \end{cases} \quad (1.16)$$

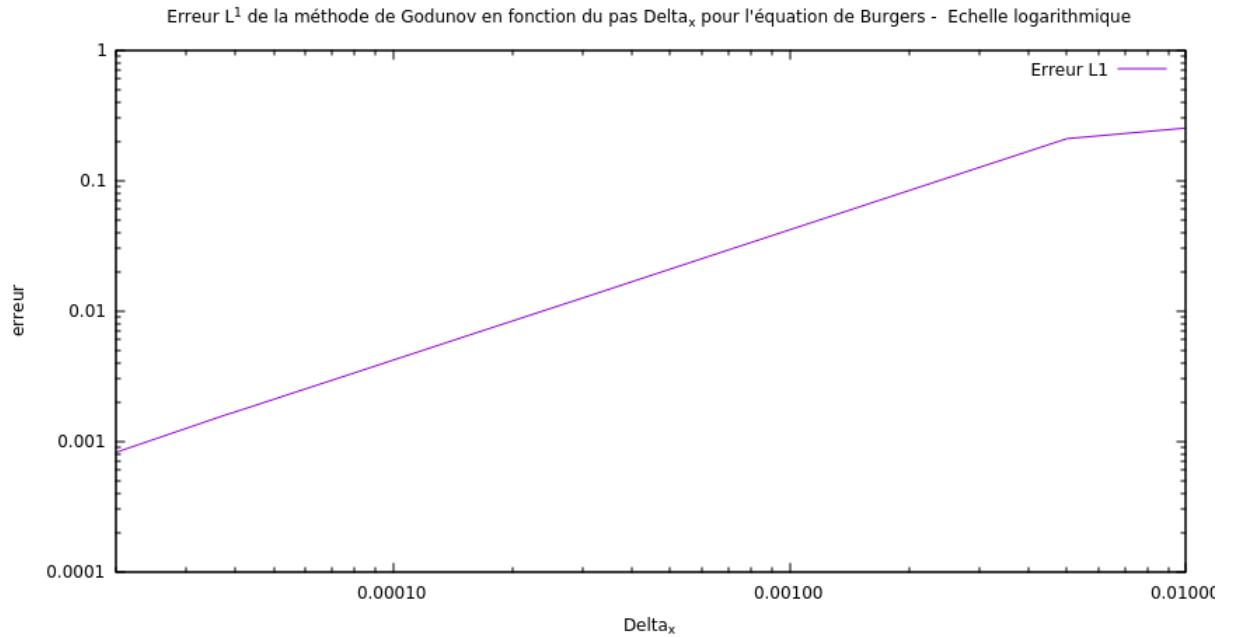
Pour une  $CFL = 0.5$  et une discréétisation de  $N=100$  on a les résultats visualisant la solution et l'erreur  $L^1$  en fonction du pas  $\Delta x$  en échelle logarithmique.

- Pour  $T = 0.5$ :

- Solution:

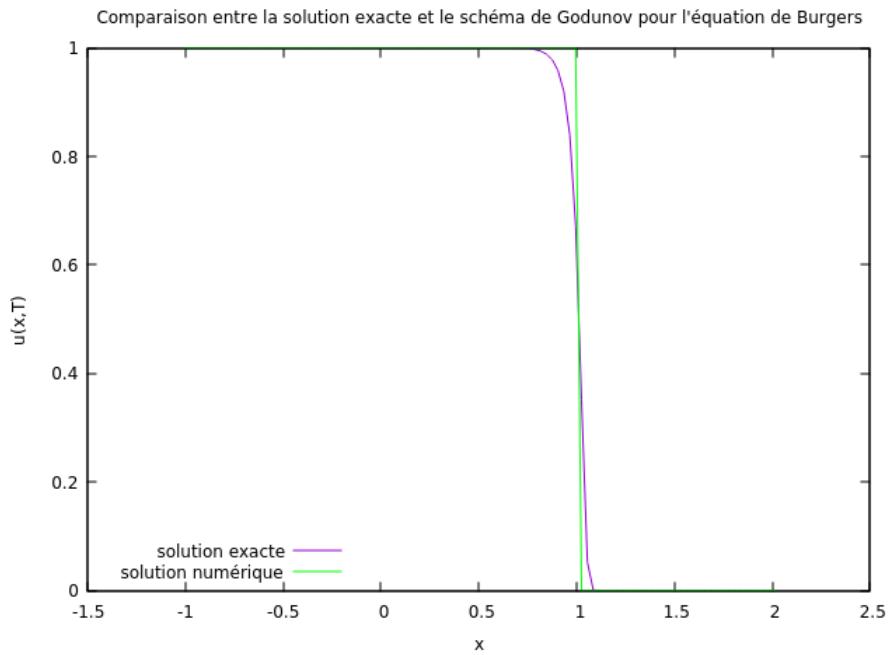


- Erreur  $L^1$ :



- Pour  $T = 1$ :

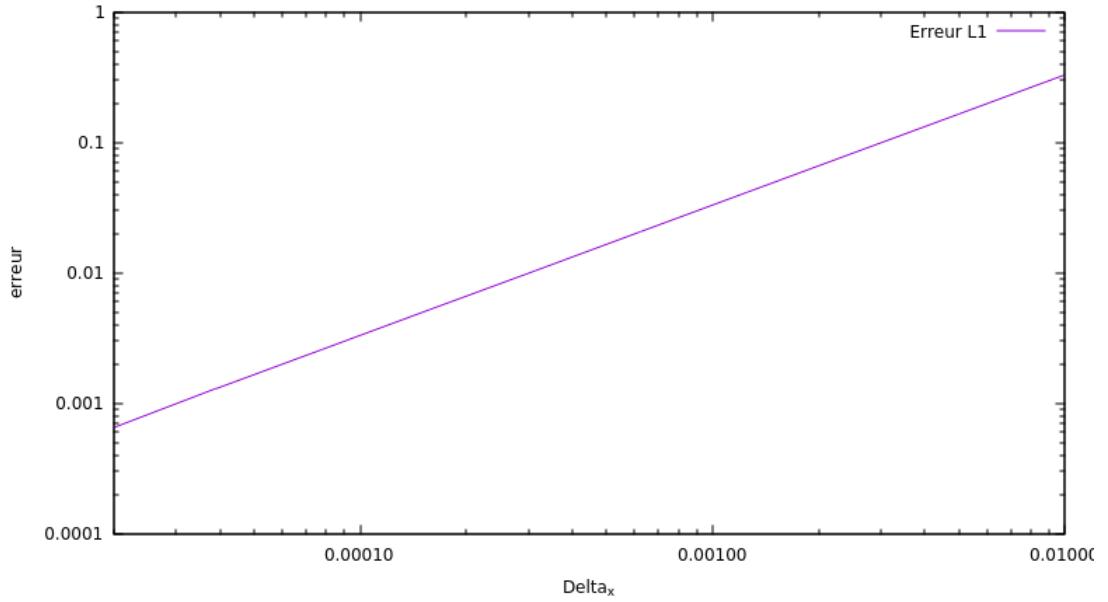
- Solution:



- Erreur  $L^1$ :

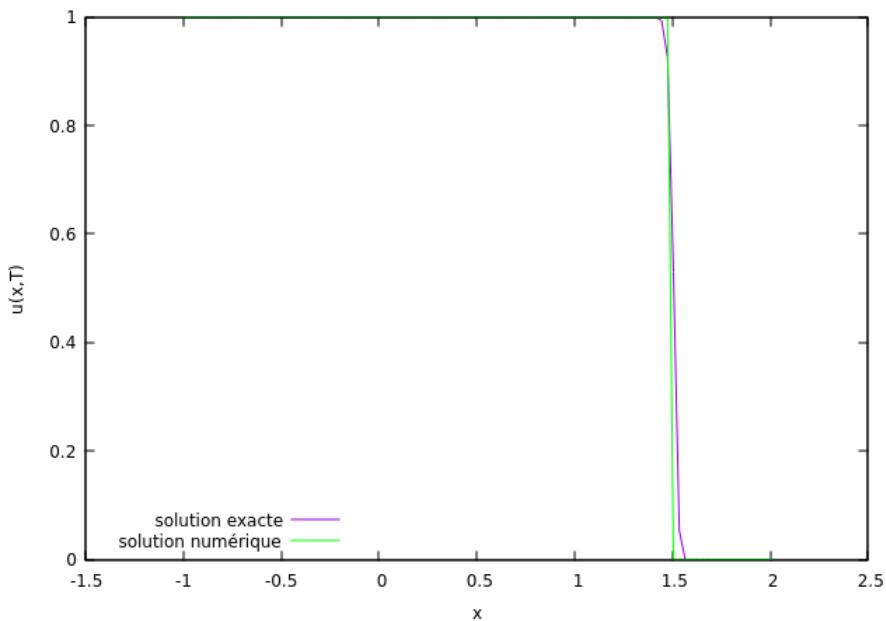
- Pour  $T = 2$ :

Erreurs  $L^1$  de la méthode de Godunov en fonction du pas  $\Delta x$  pour l'équation de Burgers - Echelle logarithmique

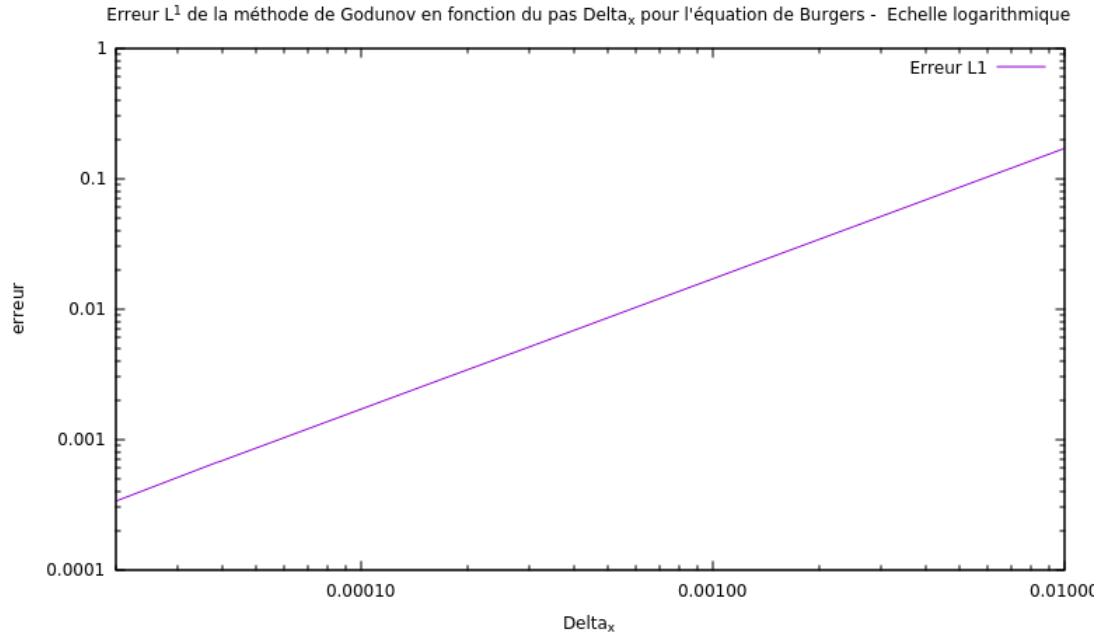


– Solution:

Comparaison entre la solution exacte et le schéma de Godunov pour l'équation de Burgers



– Erreur  $L^1$ :



En partant de la relation théorique de l'erreur en norme  $L^1$ , on a:

$$\text{erreur}(\Delta x) \sim c \times \Delta x^\beta$$

et en échelle logarithmique:

$$\ln(\text{erreur}(\Delta x)) \sim \ln(c) + \beta \times \ln(\Delta x)$$

La pente des graphes des erreurs en dessus est donc le taux de convergance pour ce cas de Burgers. En effet:

$$\beta = \frac{0.00035 - 0.0001}{0.000035 - 0.00001} = 1$$

#### **1.2.4 la correction MUSCL de van Leer:**

Le schéma de Muscl en 1D tient en compte l'erreur en  $\Delta x$  et calcule le flux en  $i + \frac{1}{2}$ . Pour ce, on calcule la pente  $s_i^n$  grâce au min des modules des pentes (décentré à gauche, à droite, centré) et puis on en déduit  $r_i^n$  telle que:

$$r_i^n = -f'(w_i^n)s_i^n$$

en effet:

```
double minmod(double a, double b, double c)
{
```

```

    double res = 0.0;
    if ((a > 0) && (b > 0) && (c > 0))
    {
        res = a < b ? (a < c ? a : c) : (b < c ? b : c);
    }
    else if ((a < 0) && (b < 0) && (c < 0))
    {
        res = a > b ? (a > c ? a : c) : (b > c ? b : c);
    }
    return res;
}

```

et donc:

```

// calcul de pentes
for (int i = 1; i < gd->N + 1; i++)
{
    // pour MUSCL
    double a = (gd->un[i] - gd->un[i - 1]) / gd->dx;
    double b = (gd->un[i + 1] - gd->un[i]) / gd->dx;
    double c = (gd->un[i + 1] - gd->un[i - 1]) / 2.0 / gd->dx;

    si[i] = minmod(a,b,c);
    ri[i] = -gd->un[i] * si[i]; // Burgers
}

```

# Chapter 2

## TP2

### 2.1 Résolution numérique de l'équation de Saint-Venant:

On désire calculer numériquement au moyen du solveur de Riemann exact fourni la solution du problème de Riemann pour le modèle de Saint-Venant suivant:

$$\left\{ \begin{array}{l} \partial_t w + \partial_x F(w) = 0, \quad w = (h, hu), \text{ et } F(w) = (hu, hu^2 + g\frac{h^2}{2}) \quad \text{avec } g = 9.81 \text{ m/s}^2, \\ u(x, 0) = 0, \\ h(x, 0) = \begin{cases} h_L = 2 & \text{si } x < 0 \\ h_R = 1 & \text{si } x > 0 \end{cases} \end{array} \right. \quad (2.1)$$

#### 2.1.1 La solution du problème de Riemann pour le modèle de Saint-Venant:

On applique les conditions de Rankine Hugoniot pour les deux choc.

On note:  $[\alpha] = \alpha_R - \alpha_L$

$$\left\{ \begin{array}{l} \sigma[h] = [hu] \\ \sigma[hu] = [hu^2 + g\frac{h^2}{2}] \end{array} \right. \quad (2.2)$$

on pose  $J = h(\sigma - u)$

$\implies$

$$\left\{ \begin{array}{l} [J] = 0 \\ [-Ju + g\frac{h^2}{2}] = 0 \end{array} \right.$$

- pour 1-choc:

$$J = -\sqrt[3]{g} \sqrt[2]{h_L h_R} \sqrt[2]{\frac{h_L + h_R}{2}}$$

- pour 2-choc:

$$J = +\sqrt[3]{g} \sqrt[2]{h_L h_R} \sqrt[2]{\frac{h_L + h_R}{2}}$$

On se donne un état gauche  $(h_L, u_L)$  et on cherche un état  $(h^*, u^*)$  reliée à l'état gauche par 1-choc.

$$\left\{ \begin{array}{l} u^* = u_L - (h^* - h_L) Z(h_L, h^*) \end{array} \right.$$

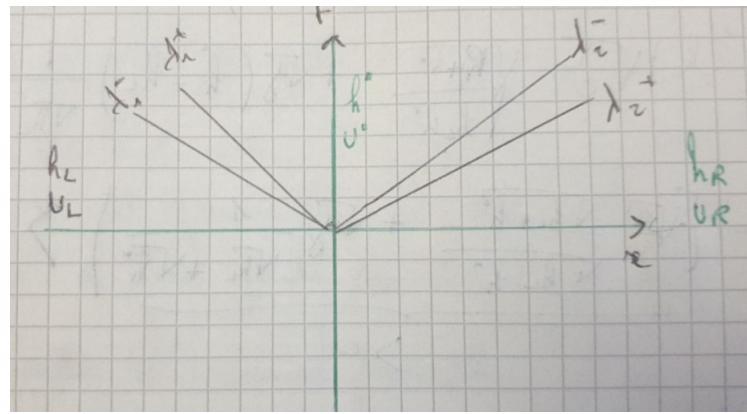
et, on se donne un état droite  $(h_R, u_R)$  et on cherche un état  $(h^*, u^*)$  reliée à l'état droite par 2-choc.

$$\left\{ \begin{array}{l} u^* = u_R - (h^* - h_R) Z(h_R, h^*) \end{array} \right.$$

avec  $Z$  de classe  $C^1$  tel que:

$$Z(a, b) = \begin{cases} \frac{2\sqrt[3]{g}}{\sqrt[2]{a} + \sqrt[3]{b}} & \text{si } b < a \\ \sqrt[3]{g} \sqrt[2]{\frac{a+b}{ab}} & \text{si } b > a \end{cases}$$

Résoudre Riemann, c'est trouver un état central  $(h^*, u^*)$ :



on distingue deux cas pour chaque i-onde:

- $\lambda_i^- < \lambda_i^+$ : Cas onde simple (i-détente).
- $\lambda_i^- = \lambda_i^+$ : Cas de i-choc.

et la vitesse du choc est:

$$\sigma = u + \frac{\lambda}{h}$$

La méthode se résume en 3 étapes:

- calcule de  $hs = h^*$  (Newton)

On déduit  $u^*$  selon les équation en dessus pour les cas 1-choc et 2-choc.

- $\lambda_{1,2}^{-,+}$  (1-choc et 2-choc)

Dans cette partie en utilise les invariants de Riemann. L'invariant de Riemann  $R$  est recherché sous la forme:

$$u + fct(h)$$

avec la condition de  $w^- > R(w)$  i-invariant de Riemann ssi:

$$\nabla R \cdot r_i = 0$$

$r_i$  est le  $i$ ème vecteur propre qui définis par convention aussi la  $i$ ème choc.

- On test  $\frac{x}{t}$  avec les vitesses d'ondes.

Voici le code commenté de la résolution du problème fourni ci-dessous:

```
void riem_stvenant(double *wL, double *wR, double xi, double *w) {

    double g = 9.81;

    double hL = wL[0];
    double uL = wL[1]/wL[0];

    double hR = wR[0];
    double uR = wR[1]/wR[0];

    double hs = 1e-6;
    int itermax = 10;

    //on applique newton pour r'\esoudre f'(hs) = 0
    for (int i = 0; i < itermax; ++i){
        double f = uL - (hs - hL) * Z(hs, hL) - uR - (hs - hR) * Z(hs, hR);
        double df = -(hs - hL) * dZ(hs, hL) - Z(hs, hL) - (hs - hR) * dZ(hs, hR);
        double dhs = -f / df;
        hs += dhs;

        // printf("i=%d f=%e df=%e hs=%e dhs=%e \n", i, f, df, hs, dhs);
    }

    double us = uL - (hs - hL) * Z(hs, hL);

    double v1m, v1p, v2m, v2p;

    // 1 - onde
    if (hs < hL){ // d tente
```

```

v1m = uL - sqrt(g*hL);
v1p = us - sqrt(g*hs);

} else { // choc

    double a = sqrt(hs) / (sqrt(hs) + sqrt(hL));
    double u = a * us + (1 - a) * uL;
    double h = (hs + hL) / 2;

    v1m = u - sqrt(g*h); //relation
    v1p = v1m;

}

// 2 - onde
if (hs < hR){ // d tente

    v2m = us + sqrt(g*hs);
    v2p = uR + sqrt(g*hR);

} else { // choc

    double a = sqrt(hs) / (sqrt(hs) + sqrt(hR));
    double u = a * us + (1 - a) * uR;
    double h = (hs + hR) / 2;

    v2m = u + sqrt(g*h);
    v2p = v2m;

}

// printf("v = %f %f %f %f\nhs=%f us=%f\n", v1m, v1p, v2m, v2p, hs, us);

if (xi < v1m){

    w[0] = wL[0];
    w[1] = wL[1];

} else if (xi < v1p) {

    double u = (uL + 2*xi + 2*sqrt(g * hL)) / 3;
    double h = (u - xi) * (u - xi) / g;

    w[0] = h;
    w[1] = h * u;

} else if (xi < v2m) {

    w[0] = hs;
    w[1] = hs * us;

} else if (xi < v2p) {

```

```

    double u = (uR + 2*xi - 2*sqrt(g*hR)) / 3;
    double h = (u - xi) * (u - xi) / g;

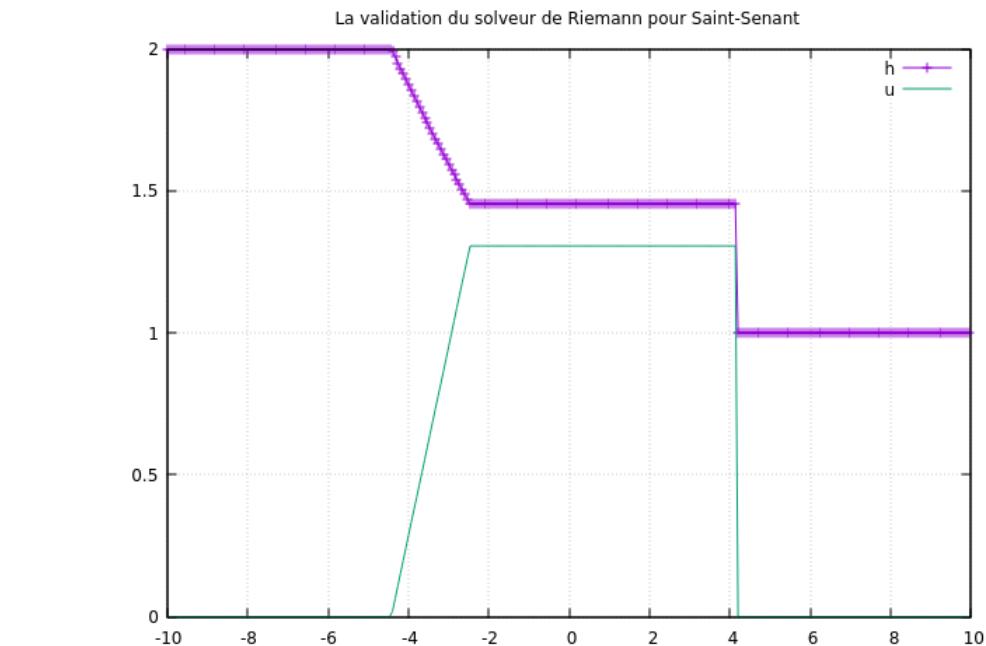
    w[0] = h;
    w[1] = h * u;

} else {

    w[0] = wR[0];
    w[1] = wR[1];

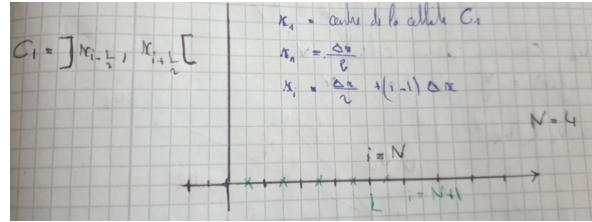
}
}

```



### 2.1.2 Programmer le schéma de Godunov:

le schéma de Godunov est basé sur la discritisation suivante:



avec:

$$w_i = (h, hu)_i = u_i^n \sim u(x_i, t_n)$$

On rappel que le schéma adopté est:

$$\frac{w_i^{n+1} - w_i^n}{\Delta t} + \frac{f(w_i^n, w_{i+1}^n) - f(w_{i-1}^n, w_i^n)}{\Delta x} = 0$$

Avec:  $f(w_L, w_R)$  est le flux numérique qui repose sur la résolution du problème de Riemann suivant:

$$\begin{cases} \partial_t w + \partial_x F(w) = 0, \quad w = (h, hu), \text{ et } F(w) = (hu, hu^2 + g \frac{h^2}{2}) \quad \text{avec } g = 9.81 m/s^2, \\ u(x, 0) = 0, \\ h(x, 0) = \begin{cases} h_L = 2 & \text{si } x < 0 \\ h_R = 1 & \text{si } x > 0 \end{cases} \end{cases} \quad (2.3)$$

Alors la solution est basée sur le même raisonnement qu'en question 1:

$$v(x, t) = \begin{cases} u_R & \text{si } x \geq ct \\ u_L & \text{si } x < ct \end{cases} \quad (2.4)$$

On note

$$v(x, t) = R(w_L, u_R, \frac{x}{t})$$

Et on peut en déduire le flux numérique par la formule suivante:

$$f(a, b) = F(R(a, b, 0))$$

Le pas  $\Delta t$  est basé sur la formule:

$$\Delta t = CFL \times \frac{\Delta x}{\lambda_{max}}$$

avec  $\lambda_{max}$  est la plus grande des valeurs propres:

$$\lambda_{max} = |u| + \sqrt[3]{gh}$$

l'implémentation est disponible sur le lien Github suivant:

[https://github.com/youness-elh/TP\\_EDP2\\_CSMI2/tree/main/TP2](https://github.com/youness-elh/TP_EDP2_CSMI2/tree/main/TP2)

### **2.1.3 La condition aux limites d'un bassin:**

On considère un bassin  $[10, 10]$  fermé. Le bassin est séparé en deux parties égales grâce à une paroi située en  $x = 0$ . À l'instant  $t = 0$ , la partie gauche (resp. droite) est remplie avec de l'eau immobile d'une hauteur  $h_L$  (resp.  $h_R$ ). À l'instant  $t = 0$ , on retire la paroi. La condition aux limites qu'on a imposé en  $x = 10$  et  $x = 0$ :

```
// valeurs aux bords
int i;
double wb[2];
double *wm;

i = 0; // bord gauche
wm = svf->wn + (i+1)*m;
// etat miroir
wb[0] = wm[0];
wb[1] = -wm[1];
wm = svf->wn + i*m;
for(int iv = 0; iv < m; ++iv) wm[iv] = wb[iv];

i = n + 1; // bord droite
wm = svf->wn + (i-1) * m;
// etat miroir
wb[0] = wm[0];
wb[1] = -wm[1];
wm = svf->wn + i * m;
for(int iv=0; iv<m; ++iv) wm[iv] = wb[iv];
```

Comme c'est indiqué sur la partie condition aux limites, on a un effet miroir gardant la même hauteur mais inversant le sens du mouvement de l'eau et donc la vitesse  $u$ . Ceci est fait en multipliant par un signe '-' la vitesse au bords gauche et droite.

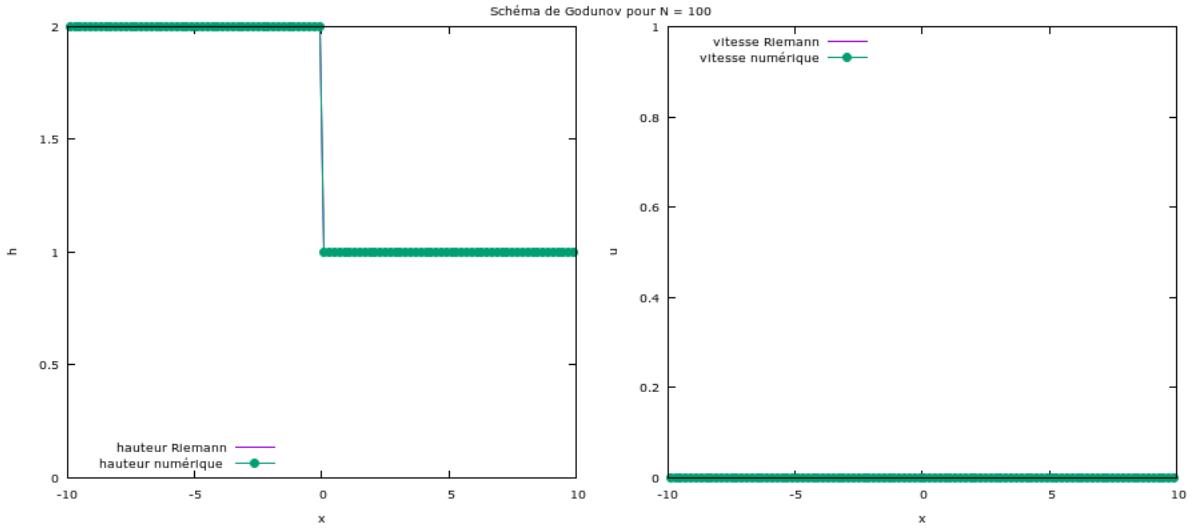
Dans le cas où le bassin est infini, cette condition doit être enlevée.

### **2.1.4 Évolution de la surface d'eau au cours du temps:**

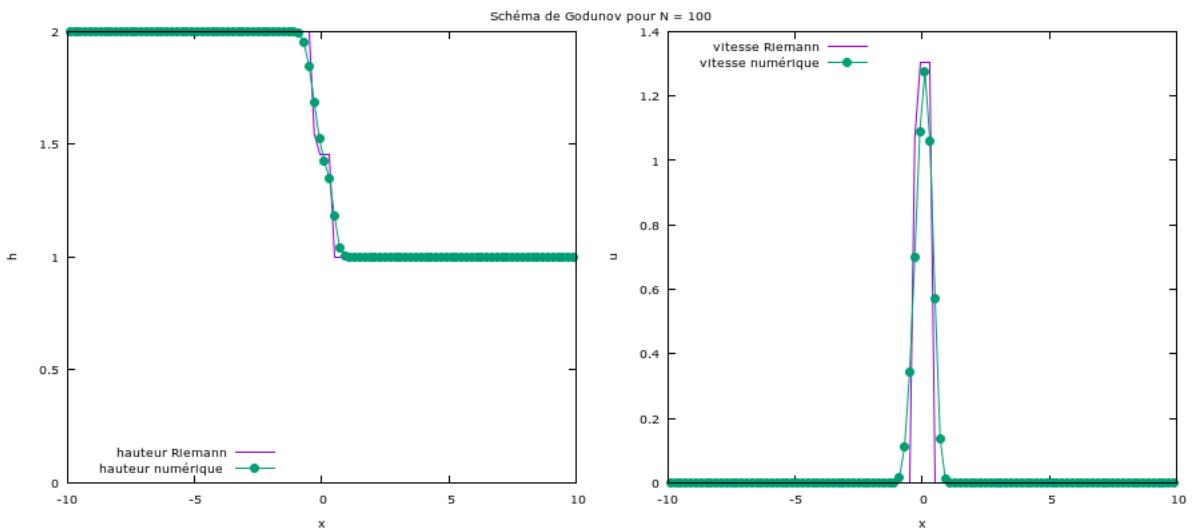
On lance le code pour une  $CFL = 0.5$  et on évalue pour plusieurs valeurs de temps maximal (i.e 0,0.1,0.5,1,2,3) et pour plusieurs valeurs de discritisation (i.e 100, 1000).

- Pour  $N = 100$

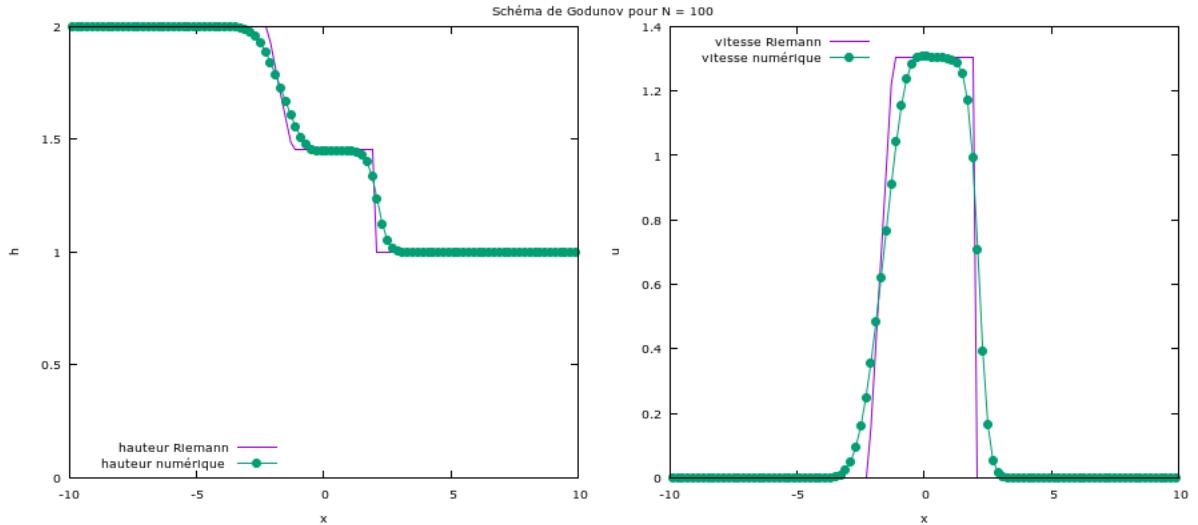
– Pour  $T = 0$ :



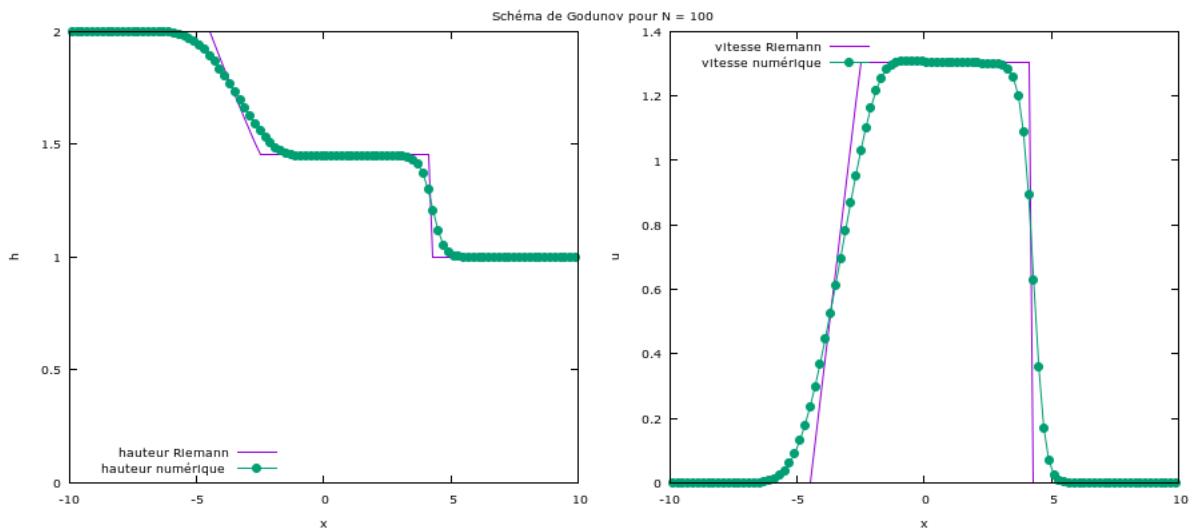
– Pour  $T = 0.1$ :



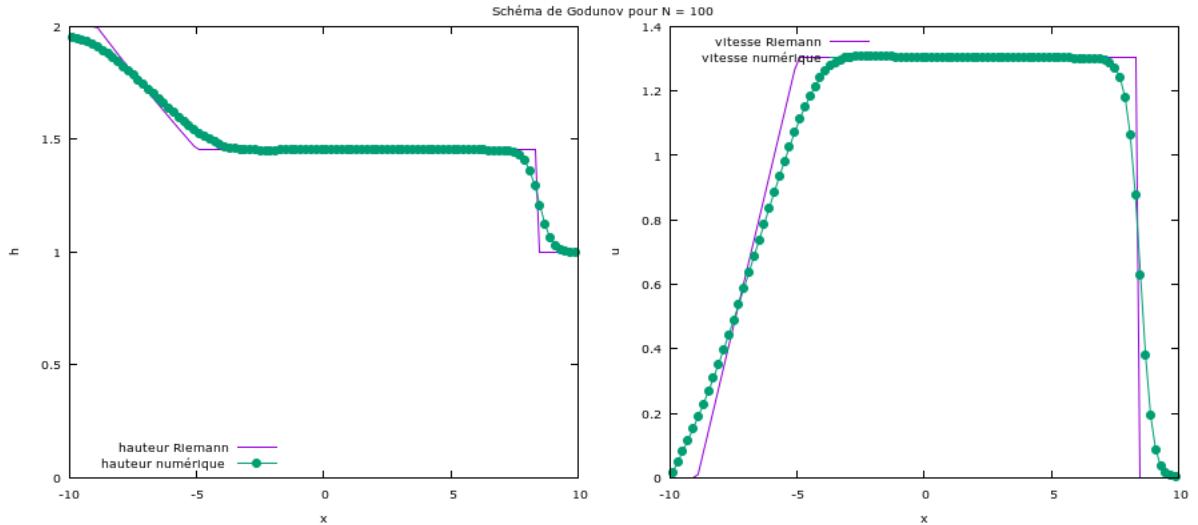
– Pour  $T = 0.5$ :



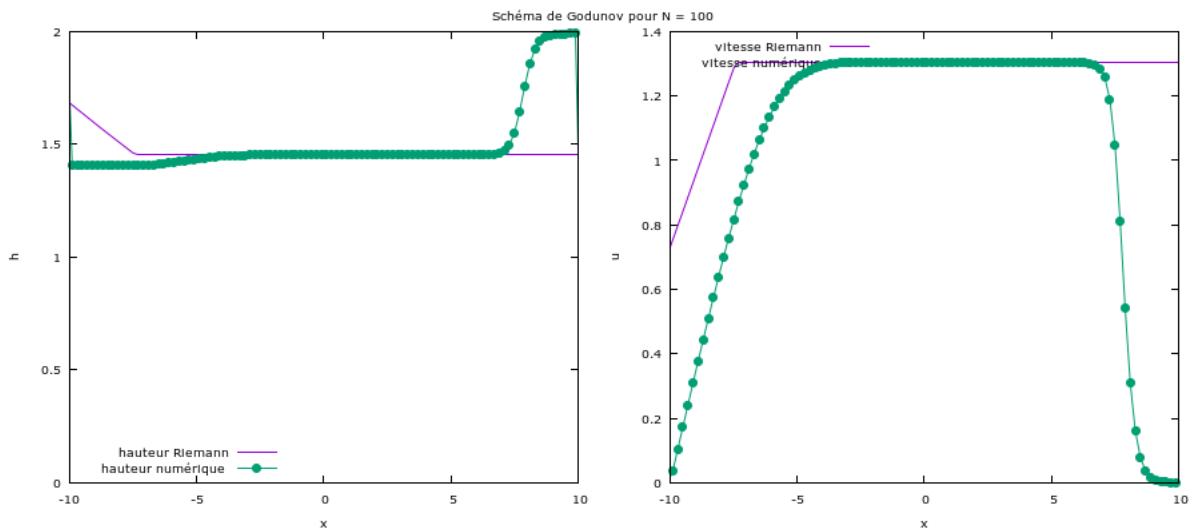
– Pour  $T = 1$ :



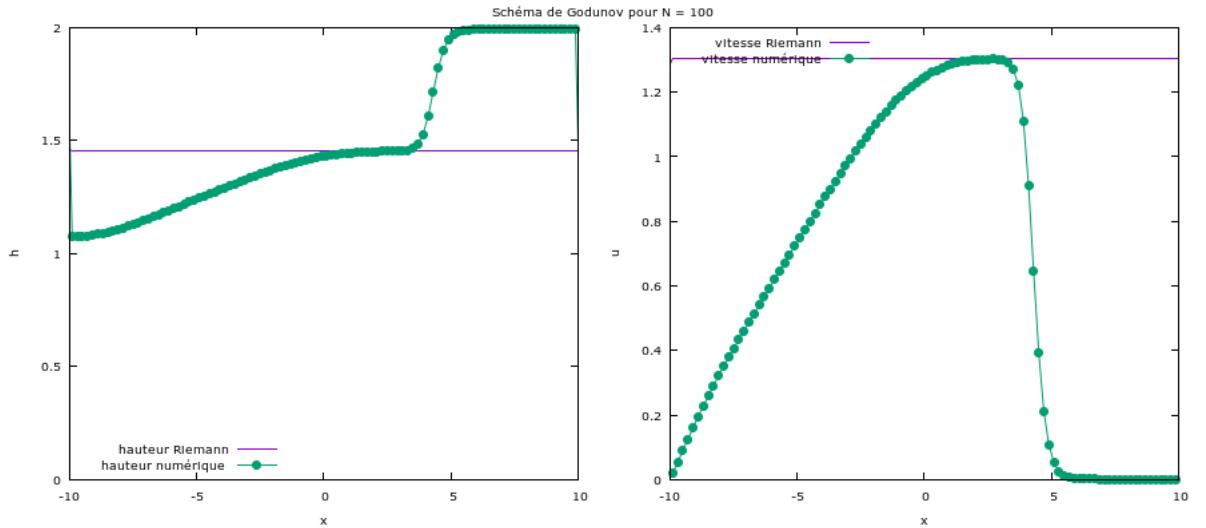
– Pour  $T = 2$ :



– Pour  $T = 3$ :

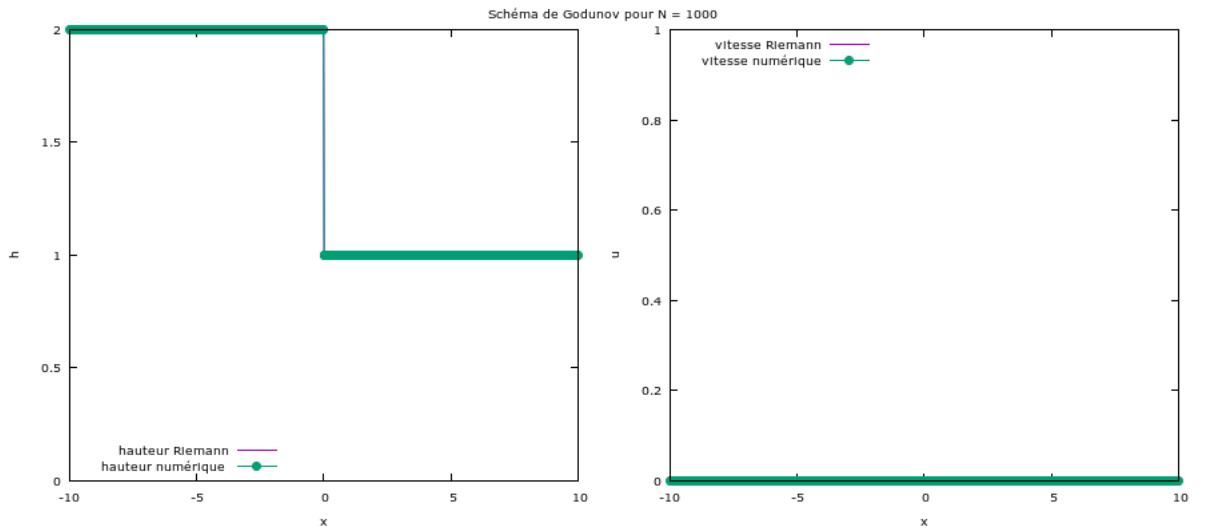


– Pour  $T = 4$ :

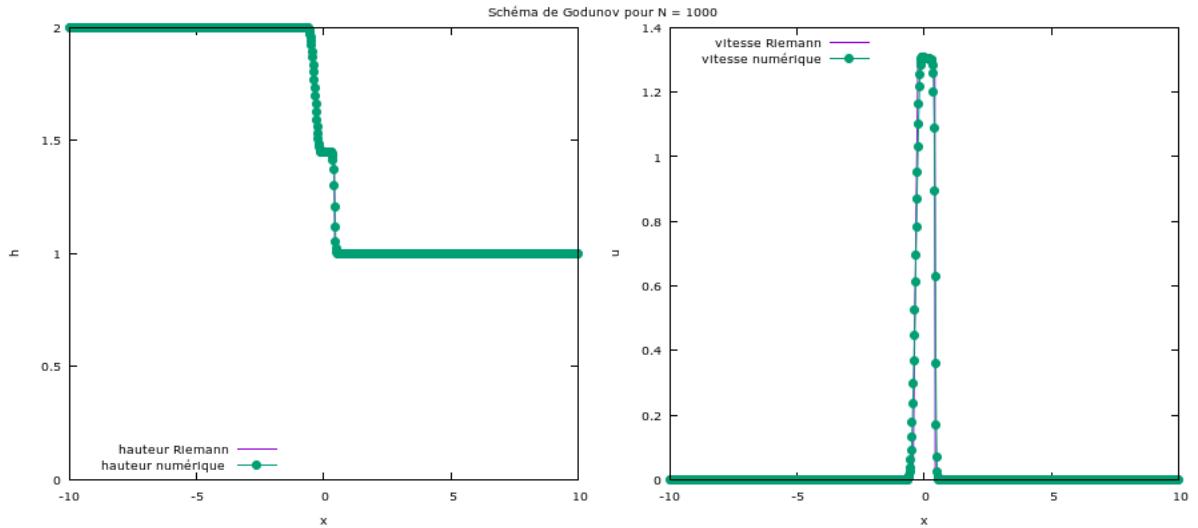


- Pour  $N = 1000$

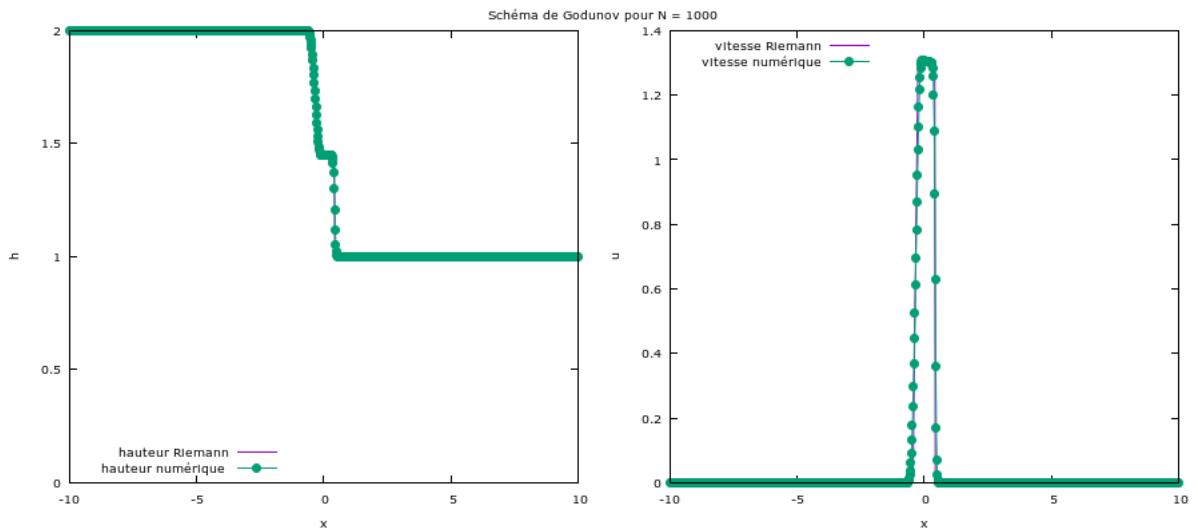
- Pour  $T = 0$ :



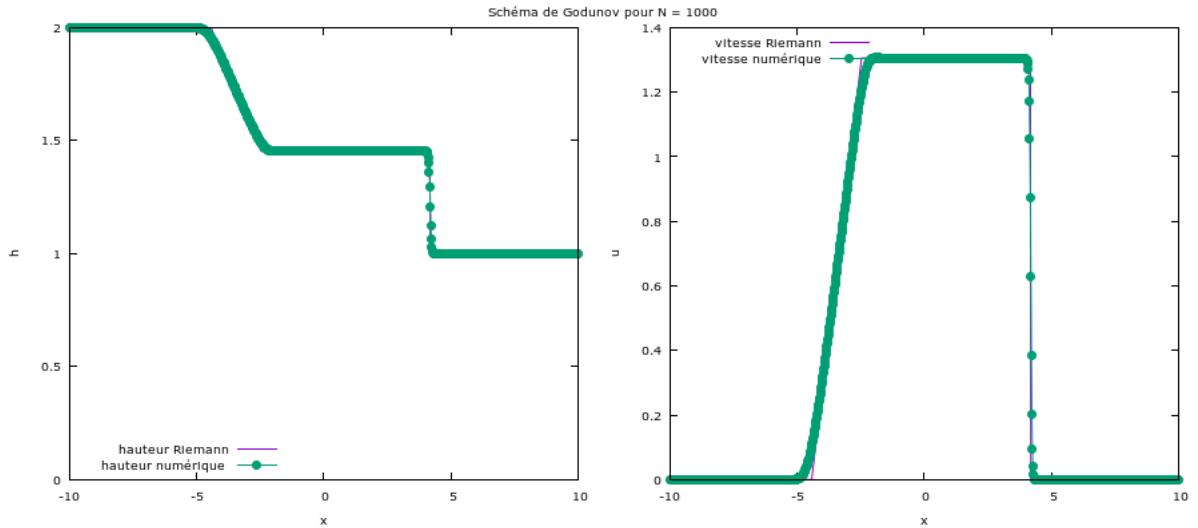
- Pour  $T = 0.1$ :



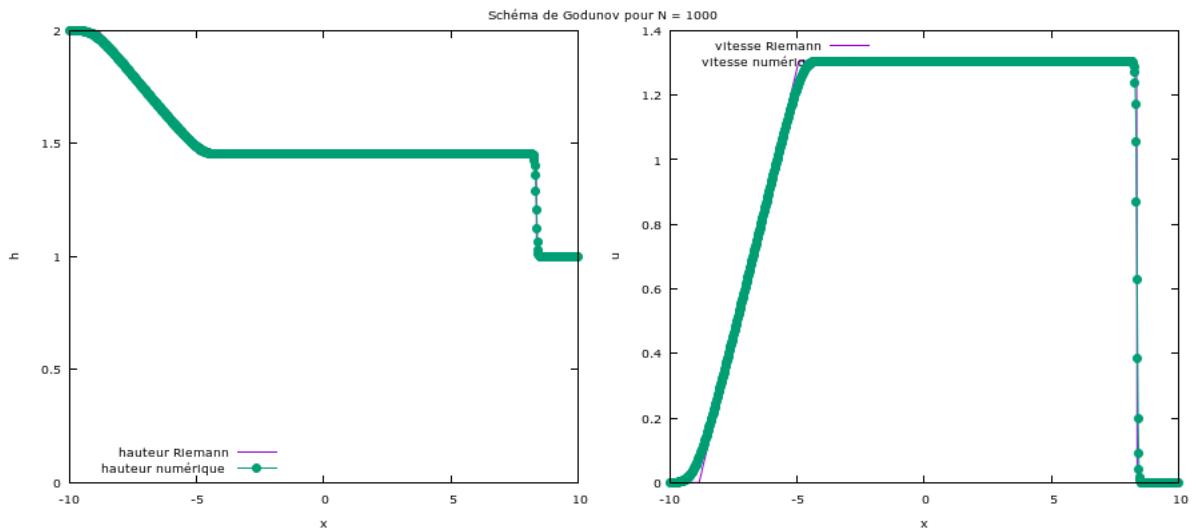
– Pour  $T = 0.5$ :



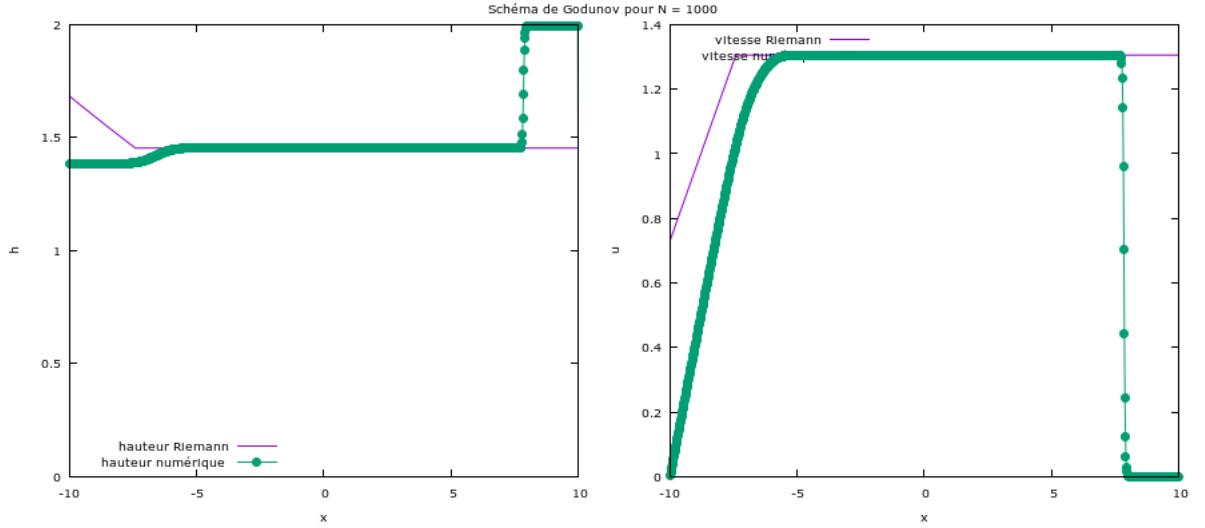
– Pour  $T = 1$ :



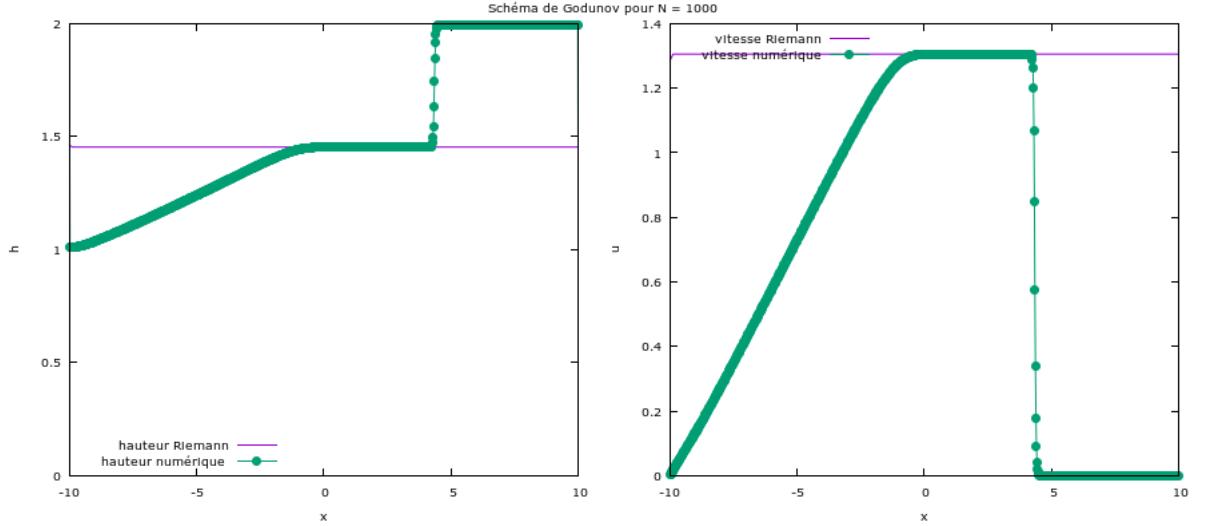
– Pour  $T = 2$ :



– Pour  $T = 3$ :



– Pour  $T = 4$ :



La solution numérique est proche de la solution exacte du problème de Riemann. Encore plus pour un grand nombre de discritisation  $N$ . Cependant, à partir d'un instant  $T \sim 3$ , il est plus claire que l'approximation Godunov n'est plus proche de la solution exacte Riemann. Ceci est dû au ratio  $\frac{x}{t}$  qui devient plus petit quand  $t_{\max}$  est grand et donc  $< \lambda(w_L)$  engendrant une solution constante  $= w_L$ .

### **2.1.5 Le schéma de Rusanov:**

Le schéma Rusanov n'est pas basé sur le solveur Riemann quant à lui, contrairement à Godunov. La différence est donc dans le calcul du flux numérique qui est dans le cas Rusanov:

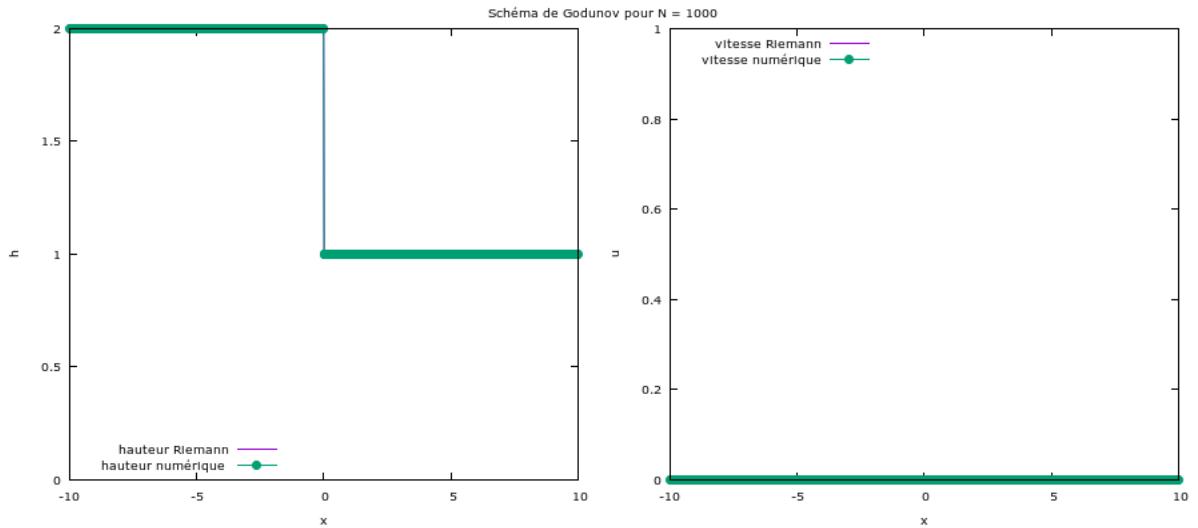
$$f(a, b) = \frac{F(a) + F(b)}{2} - \frac{\lambda(a, b)(b - a)}{2}$$

avec  $F$  est le flux physique et  $\lambda(a, b)$  se calcule par la formule:

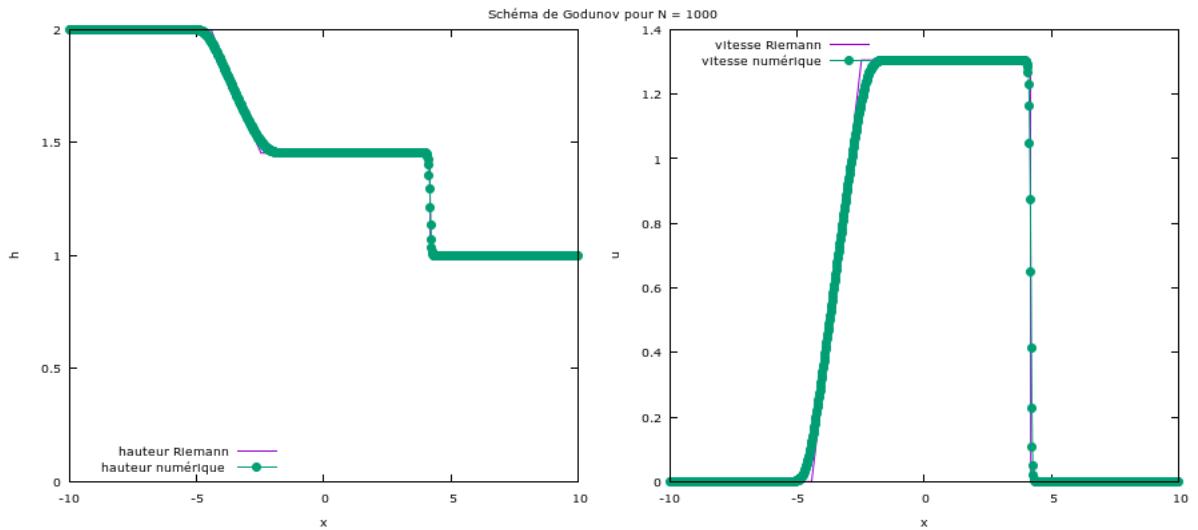
$$\lambda(a, b) = \max(|F'(a)|, |F'(b)|)$$

On test pour une CFL = 0.5 et un nombre de discritisation N = 1000.

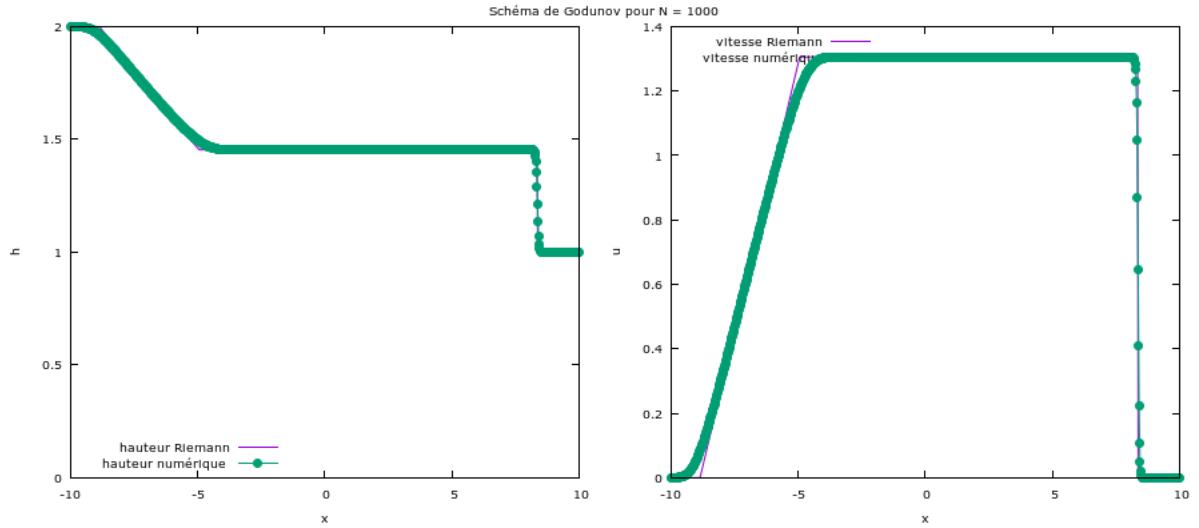
- Pour T = 0:



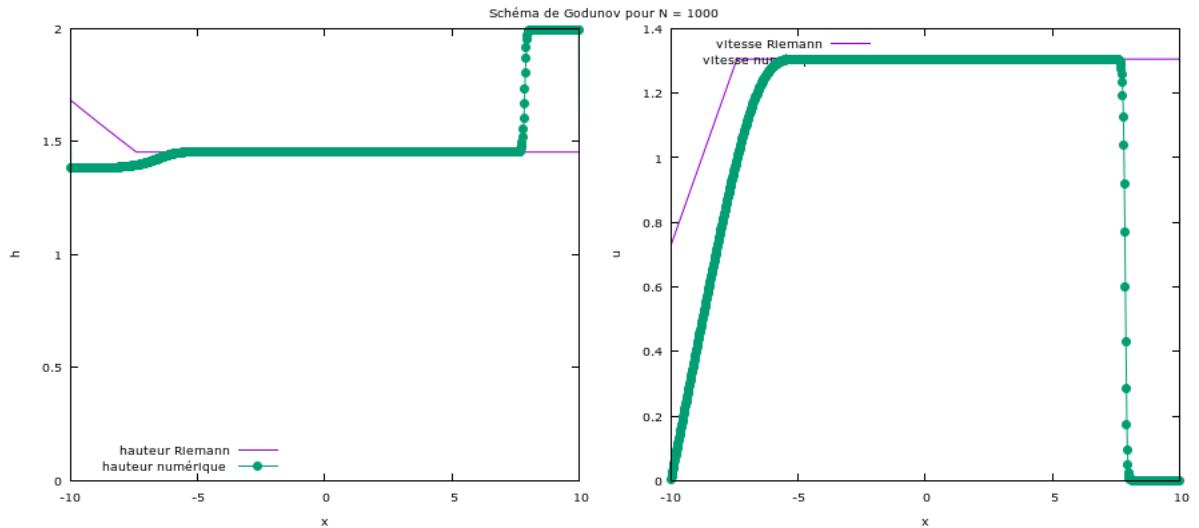
- Pour T = 1:



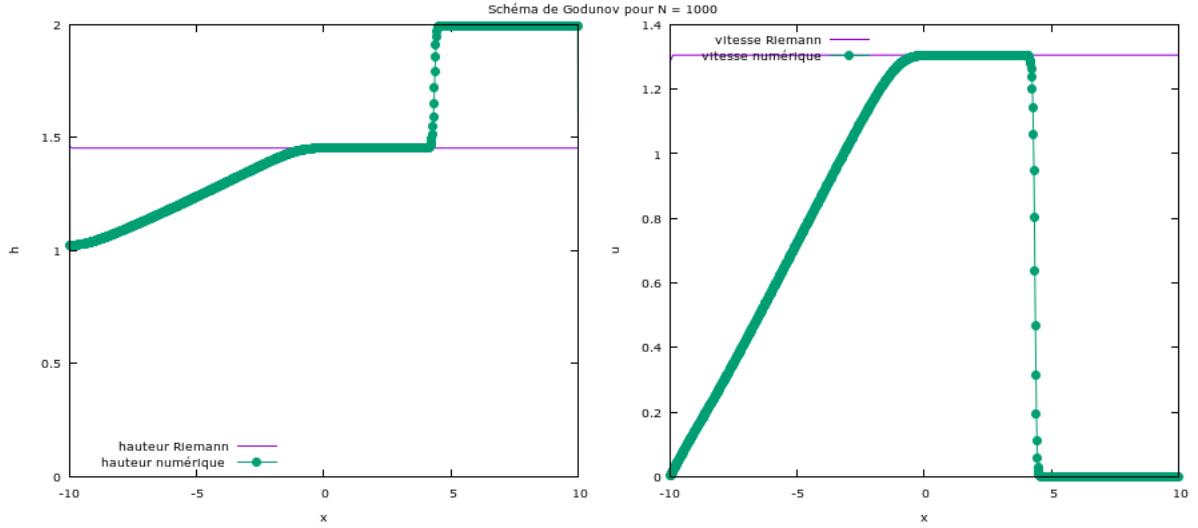
- Pour  $T = 2$ :



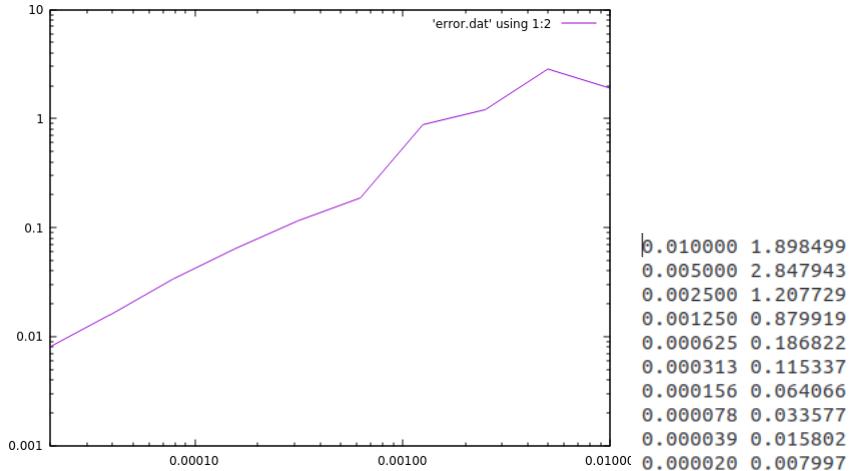
- Pour  $T = 3$ :



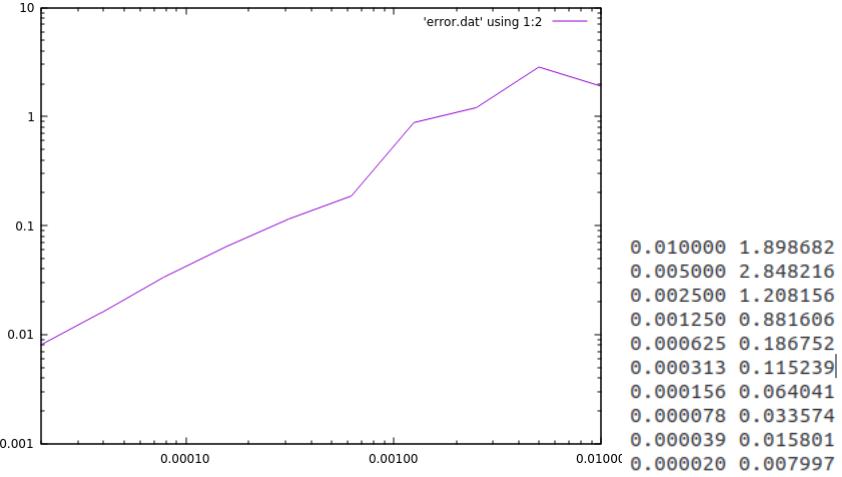
- Pour  $T = 4$ :



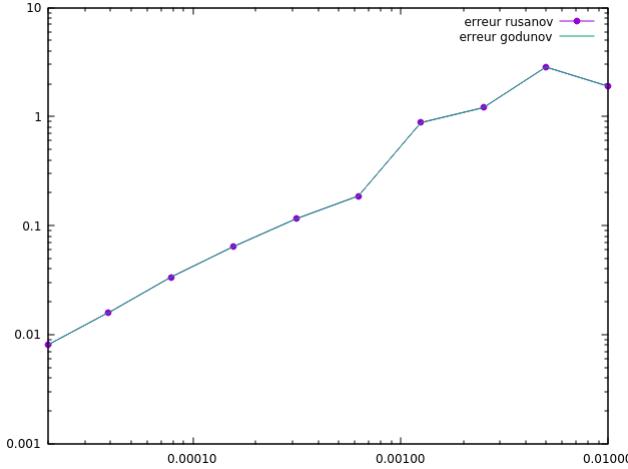
La méthode Rusanov est bien plus rapide (un solveur de Riemann de moins), mais est elle précise? les graphes suivant sont des graphes d'erreur  $L^1$  de l'approximation Rusanov en premier temps puis Godunov de la hauteur en fonction du pas de discritisation  $\Delta x$  pour un instant  $T = 0.5$ .



Et la figure suivante représente l'erreur L1 de la hauteur de Godunov, aussi pour un instant  $T = 0.5$  et un  $\Delta x$  variant entre 0.01 et  $1e - 5$ .



On trace les deux erreurs dans le même graphe.



L'erreur Rusanov est légèrement plus grande que celle produite par Godunov, aussi la valeur d'erreur atteinte en  $\Delta x = 1e - 5$  et plus petite pour Godunov donc on peut donc conclure que la méthode Rusanov est moins précise que Godunov.

### 2.1.6 Décrire le schéma VFRoe et le flux numérique:

L'idée du schéma de VFRoe est de remplacer le solveur de Riemann exact par un solveur de Riemann approché, plus rapide et plus simple.

On rappel que le schéma adopté est:

$$\frac{w_i^{n+1} - w_i^n}{\Delta t} + \frac{f(w_i^n, w_{i+1}^n) - f(w_{i-1}^n, w_i^n)}{\Delta x} = 0$$

Avec:  $f(w_L, w_R)$  est le flux numérique qui repose sur la résolution du problème de VFRoe  $\tilde{R}$ .

$$f(w_L, w_R) = F(\tilde{R}(w_L, w_R, 0))$$

On a:

$$w_t + F(w)_x = 0 \quad (*)$$

avec  $w = (h, hu)$

En posant  $y = \begin{pmatrix} h \\ u \end{pmatrix}$   $(*) \iff$

$$y_t + B(y)y_x = 0 \quad \text{avec } B(y) = \begin{pmatrix} u & h \\ g & u \end{pmatrix}$$

avec  $y \begin{pmatrix} h \\ u \end{pmatrix}$  variables primitives.

On va linéariser les équations autour d'un état moyen.

$$\tilde{y} = \frac{y_L + y_R}{2}$$

On remplace la résolution du problème Riemann par:

$$\begin{cases} y_t + B(\tilde{y})y_x = 0, \\ y(x, 0) = \begin{cases} y_L & \text{si } x < 0 \\ y_R & \text{si } x > 0 \end{cases} \end{cases} \quad (2.5)$$

et on utilise la formule:

$$y(0, t) = \tilde{y} - \text{sign}(B(\tilde{y})) \times \tilde{y}$$

On détermine le signe de  $B(\tilde{y})$  en utilisant le polynôme d'interpolation  $Q$  de la fonction  $g$  sur les valeurs propres de  $B$ .

On pose:

$$\lambda_1 = \tilde{u} - \sqrt{g\tilde{h}} \text{ et } \lambda_2 = \tilde{u} + \sqrt{g\tilde{h}} \text{ les valeurs propres de } B(\tilde{y}).$$

$$B = PDP^{-1}$$

$$g(B) := P \begin{pmatrix} g(\lambda_1) & 0 \\ 0 & g(\lambda_2) \end{pmatrix} P^{-1}$$

$$Q(\lambda_1) = g(\lambda_1), \quad Q(\lambda_2) = g(\lambda_2), \quad \text{et } dQ = 1$$

$$Q(B) = g(B)$$

Soit

$$B(\tilde{y}) = \begin{pmatrix} \tilde{u} & \tilde{h} \\ g & \tilde{u} \end{pmatrix}$$

soit :  $\tilde{c} = \sqrt{g\tilde{h}}$  Et on distingue 3 cas pour calculer le signe de  $B(\tilde{y})$ :

- cas  $\lambda_1$  et  $\lambda_2$  sont  $> 0$ :

$$\text{sign}(B(\tilde{y})) = I$$

alors:

$$y(0, t) = y_L$$

Et donc:

$$\tilde{R}(w_L, w_R, 0) = w_L$$

- cas  $\lambda_1$  et  $\lambda_2$  sont  $< 0$ :

$$\text{sign}(B(\tilde{y})) = -I$$

alors:

$$y(0, t) = y_R$$

Et donc:

$$\tilde{R}(w_L, w_R, 0) = w_R$$

- cas  $\lambda_1 < 0$  et  $\lambda_2 > 0$ : On définit  $Q$ : si  $\lambda = \lambda_1$  alors  $Q(\lambda) = -1$

si  $\lambda = \lambda_2$  alors  $Q(\lambda) = 1$

et donc:

$$Q(\lambda) = -1 \frac{\lambda - \lambda_2}{\lambda_1 - \lambda_2} + 1 \frac{\lambda - \lambda_1}{\lambda_1 - \lambda_2} = \frac{\lambda - \tilde{u}}{\tilde{c}}$$

alors:

$$\begin{aligned} \text{sign}(B(\tilde{y})) &= Q(B(\tilde{y})) = \frac{B - \tilde{u}I}{\tilde{c}} \\ &= \frac{1}{\tilde{c}} \begin{pmatrix} 0 & \tilde{h} \\ g & 0 \end{pmatrix} \end{aligned}$$

donc:

$$y(0, t) = \tilde{y} - \frac{1}{\tilde{c}} \begin{pmatrix} 0 & \tilde{h} \\ g & 0 \end{pmatrix} \times \tilde{y}$$

On a donc les variables primitives:

$$\begin{aligned} h^* &= \frac{h_L + h_R}{2} - \frac{1}{2\sqrt{g\tilde{h}}} \tilde{h}(u_R - u_L) \\ u^* &= \frac{u_L + u_R}{2} - \frac{1}{2\sqrt{g\tilde{h}}} g(h_R - h_L) \end{aligned}$$

Et donc:

$$\tilde{R}(w_L, w_R, 0) = w(y(0, t))$$

Et on déduit le flux numérique pour chacun des cas expliqué en dessus.

$$f(w_L, w_R) = F(\tilde{R}(w_L, w_R, 0))$$

### 2.1.7 Programmer le schéma VF Roe:

Le code qui mis en oeuvre le flux numérique calculé en dessus est le suivant:

```
void vfroe_stvenant(double *wL, double *wR, double xi, double *w) {
    double g = 9.81;

    double hL = wL[0];
    double uL = wL[1]/wL[0];

    double hR = wR[0];
    double uR = wR[1]/wR[0];

    double uc = (uR + uL)/2;
    double hc = (hL + hR)/2;

    double cc = sqrt(g*hc);

    double lbd1 = uc - cc;
    double lbd2 = uc + cc;

    double h, u;
    if (lbd1 < 0 && lbd2 < 0){
        h = hR;
        u = uR;
    }
    else if (lbd1 > 0 && lbd2 > 0) {
        h = hL;
        u = uL;
    } else {
        h = hc*(1 - (uR - uL)/cc);
        u = uc - g*(hR - hL)/2/cc;
    }
    w[0] = h;
    w[1] = h*u;
}
```

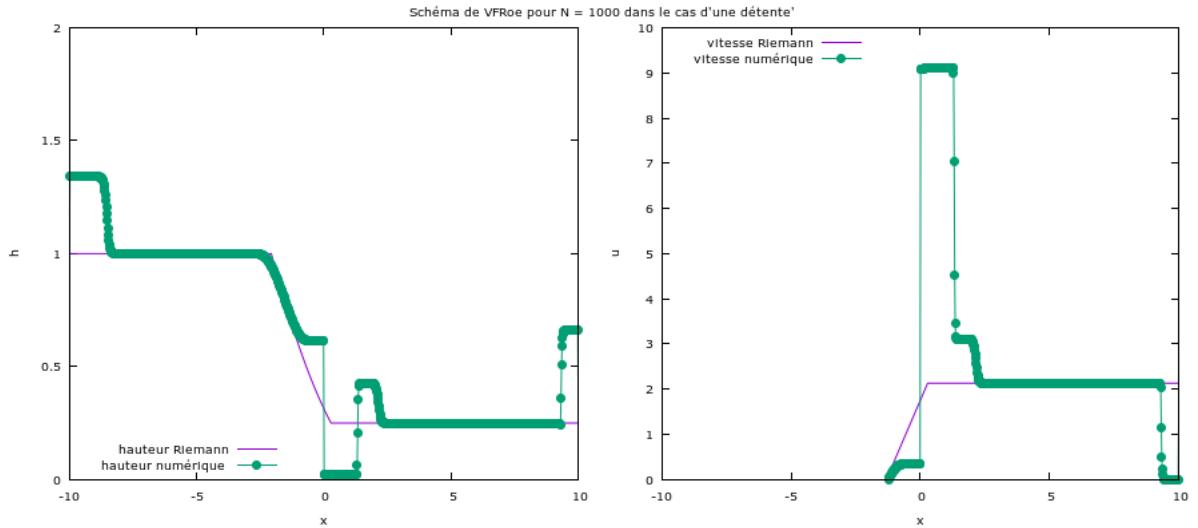
Pour vérifier que ce schéma ne donne pas toujours la bonne solution on choisit un problème de Riemann associé à une onde de détente qui traverse une valeur propre nulle. Dans cette détente,  $u + 2\sqrt{gh}$  est constant et londe contient un point  $u - 2\sqrt{gh} = 0$ .

En effet on prend les conditions aux limites suivants:

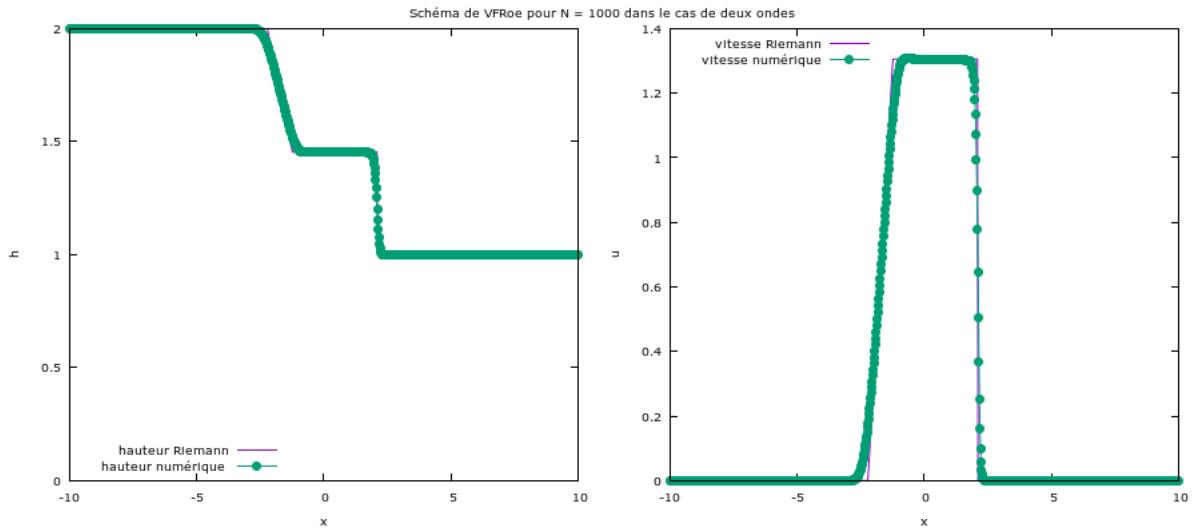
```
double hL = 1.;
double uL = -1.;

double hR = 1./4;
double uR = uL + 2*sqrt(9.81) * (sqrt(hL)-sqrt(hR));
```

Et on a donc une fausse approximation de la partie de discontinuité:



alors que pour un cas de deux ondes ca fonctionne correctement.



### 2.1.8 Programmer la correction entropique qui utilise le flux de Rusanov aux points soniques:

### 2.1.9 Comparer avec le schéma Godunov:

### 2.1.10 Le schéma de Roe:

### 2.1.11 la correction MUSCL de van Leer:

# Chapter 3

## TP3

On considère une carte géographique modélisée par une fonction  $A$  définie sur le carré  $[0, L] \times [0, L]$ . La quantité  $A(x, y)$  désigne la latitude du point  $(x, y)$ . De l'eau peut se déverser sur cette topographie. La surface de l'eau se trouve à la latitude  $A(x, y) + h(x, y, t)$ . La hauteur d'eau  $h$  est petite devant  $L$ , le champ de vitesse est quasi-horizontal et constant suivant  $z$ . La vitesse est entièrement déterminée par ses composantes suivant  $x$  et  $y$   $u(x, y, t)$  et  $v(x, y, t)$ . Par convention, on convient que la vitesse est nulle sur une zone sèche, c'est à dire aux points  $(x, y, t)$  où  $h(x, y, t) = 0$ .

### 3.1 Résolution numérique de l'équation de Saint-Venant en 2D:

Dans ce cadre, le modèle de Saint-Venant (ou shallow water) secrète:

$$\begin{cases} \partial_t h + \partial_x hu + \partial_y hv = 0, \\ \partial_t hu + \partial_x (hu^2 + g\frac{h^2}{2}) + \partial_y huv = -gh\partial_x A, \\ \partial_t hv + \partial_x huv + \partial_y (hv^2 + g\frac{h^2}{2}) = -gh\partial_y A, \end{cases} \quad (3.1)$$

#### 3.1.1 Réécrire le système de Saint-Venant 2D:

On pose:  $w = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$

On a donc:

$$F_1(w) = \begin{pmatrix} w_2 \\ \frac{w_2^2}{w_1} + g\frac{w_1^2}{2} \\ \frac{w_1 w_2 w_3}{w_1} \end{pmatrix} = \begin{pmatrix} hu \\ hu^2 + g\frac{h^2}{2} \\ huv \end{pmatrix} \text{ et: } F_2(w) = \begin{pmatrix} w_3 \\ \frac{w_2 w_3}{w_1} \\ \frac{w_3^2}{w_1} + g\frac{w_1^2}{2} \end{pmatrix} = \begin{pmatrix} hv \\ huv \\ hv^2 + g\frac{h^2}{2} \end{pmatrix}$$

$$\text{avec } S(w) = \begin{pmatrix} 0 \\ -gw_1\partial_x A \\ -gw_1\partial_x A \end{pmatrix}$$

Afin de réécrire le système de Saint-Venant 2D:

$$\partial_t w + \partial_x F_1(w) + \partial_y F_2(w) = S(w)$$

### 3.1.2 Le système de Saint-Venant 2D est hyperbolique?:

pour montrer que le système est dit hyperbolique, on montre que  $\partial_w F(w)n$  est diagonalisable avec des valeurs propres réelles pour tout  $w$  tel que  $w_1 \neq 0$  et pour tout  $n$ . En particulier, puisque le système est symétrique par rotation on prend  $n = (1, 0)$  alors il suffit de montrer que,  $\partial_w F_1(w)$  est diagonalisable avec des valeurs propres réelles pour tout  $w$ .

En effet:

$$F'_1(w) = \begin{pmatrix} 0 & 1 & 0 \\ -\frac{w_2^2}{w_1^2} + gw_1 & 2\frac{w_2}{w_1} & 0 \\ -\frac{w_2 w_3}{w_1^2} & \frac{w_3}{w_1} & \frac{w_2}{w_1} \end{pmatrix}$$

On calcule le polynôme caractéristique:

$$\det(F'_1(w) - \lambda I) = \begin{vmatrix} -\lambda & 1 & 0 \\ -\frac{w_2^2}{w_1^2} + gw_1 & 2\frac{w_2}{w_1} - \lambda & 0 \\ -\frac{w_2 w_3}{w_1^2} & \frac{w_3}{w_1} & \frac{w_2}{w_1} - \lambda \end{vmatrix}$$

$\implies$

$$\det(F'_1(w) - \lambda I) = \left(\frac{w_2}{w_1} - \lambda\right)\left(-\lambda\left(2\frac{w_2}{w_1} - \lambda\right) + \frac{w_2^2}{w_1^2} - gw_1\right)$$

$\implies$

$$\det(F'_1(w) - \lambda I) = \left(\frac{w_2}{w_1} - \lambda\right)\left(\left(\lambda - \frac{w_2}{w_1}\right)^2 - gw_1\right)$$

$\implies$

$$\det(F'_1(w) - \lambda I) = \left(\frac{w_2}{w_1} - \lambda\right)\left(\lambda - \frac{w_2}{w_1} + \sqrt{gw_1}\right)\left(\lambda - \frac{w_2}{w_1} - \sqrt{gw_1}\right)$$

en outre  $\det(F'_1(w) - \lambda I) \implies$

$$\begin{cases} \lambda_1 = \frac{w_2}{w_1} \\ \lambda_2 = \frac{w_2}{w_1} + \sqrt{gw_1} \\ \lambda_3 = \frac{w_2}{w_1} - \sqrt{gw_1} \end{cases}$$

et donc pour toute direction:

$$\begin{cases} \lambda_1 = \frac{w_2}{w_1} \cdot n \\ \lambda_2 = \frac{w_2}{w_1} \cdot n + \sqrt{gw_1} \\ \lambda_3 = \frac{w_2}{w_1} \cdot n - \sqrt{gw_1} \end{cases}$$

On remarque que les valeurs propres sont bien réelles pour tout w et de multiplicité 3 donc le système est bien hyperbolique.

### 3.1.3 Le flux numérique $f(w_L, w_R, n)$ de Rusanov en 2D:

On souhaite réaliser une approximation numérique du système de Saint-Venant au moyen de la méthode des volumes finis. Pour cela on explicite d'abord le flux numérique en l'occurrence celui de rusanov:

$$f(w_L, w_R, n) = \frac{1}{2} (f(w_L, n) + f(w_R, n)) - \frac{\lambda(w_L, w_R)}{2} (w_R - w_L)$$

avec le flux physique dans une direction n:

$$f(w, n) = f(w) \cdot n = F_1(w) \cdot n_1 + F_2(w) \cdot n_2$$

où les  $\lambda_k$  sont les valeurs propres de  $f'(w, n)$ :

$$\lambda(w_L, w_R) = \max_k \max(|\lambda_k(w_L, n)|, |\lambda_k(w_R, n)|)$$

Pour optimiser les calculs on fixe  $\lambda$ :

$$\lambda(w_L, w_R) = \max\left(\left|\frac{w_2}{w_1} \cdot n + \sqrt{gw_1}\right|_L, \left|\frac{w_2}{w_1} \cdot n - \sqrt{gw_1}\right|_R\right)$$

### 3.1.4 Décrire comment utiliser le solveur de Riemann 1D pour le calcul en 2D:

On fait un changement de variable décrit ci-dessous qui ramène le problème 2D en un problème 1D pour le résoudre et en déduire w.

```
void flux_riem_2d(double *wL, double *wR, double *vnorm, double *flux)
{
    double qnL = wL[1] * vnorm[0] + wL[2] * vnorm[1];
    double qnR = wR[1] * vnorm[0] + wR[2] * vnorm[1];
```

```

double qtL = -wL [1] * vnorm [1] + wL [2] * vnorm [0];
double qtR = -wR [1] * vnorm [1] + wR [2] * vnorm [0];

double vL [2] = {wL [0], qnL};
double vR [2] = {wR [0], qnR};

double v [2];
double xi = 0;

riem_stvenant (vL, vR, xi, v);

double un = v [1] / v [0];
double ut;

ut = (un > 0) ? qtL / wL [0] : qtR / wR [0];

double qn = v [1];
double qt = ut * v [0];

double w [3];
w [0] = v [0];
w [1] = qn * vnorm [0] - qt * vnorm [1];
w [2] = qn * vnorm [1] + qt * vnorm [0];

// puis calcul du flux physique (appliqu w)
fluxphy (w, vnorm, flux);
}

```

### 3.1.5 Conditions aux limites: miroir, valeurs imposées, zone sèche:

- CL miroir: Sur les bords les vitesses sont inversées de sens:

$$w_m = \begin{pmatrix} h \\ -hu \\ -hv \end{pmatrix}$$

- CL imposées:

On substitue les valeurs imposées:

$$w_i = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}$$

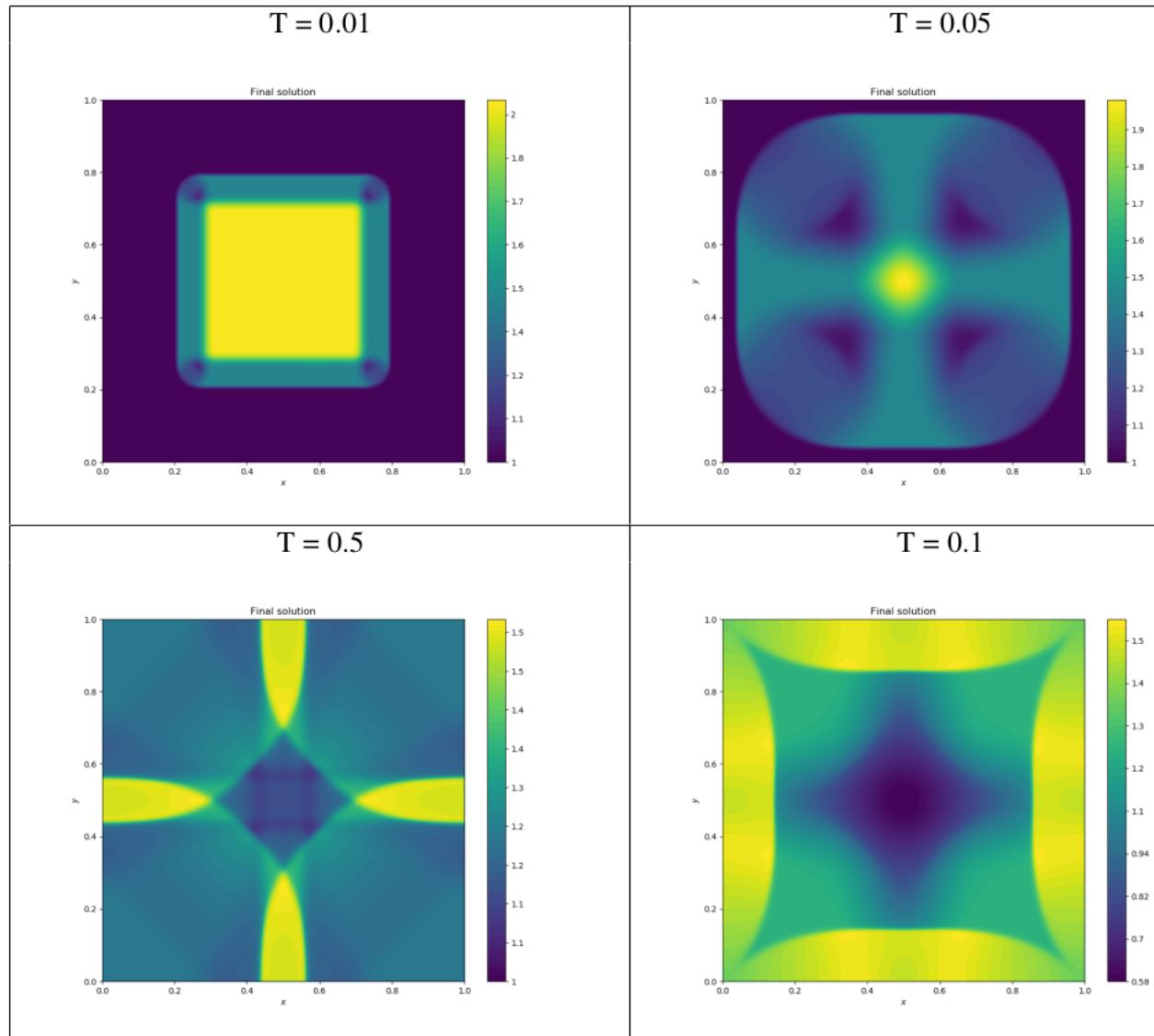
- CL zone sèche: Par convention, on convient que la vitesse est nulle sur une zone sèche, c'est à dire aux points  $(x, y, t)$  où  $h(x, y, t) = 0$ . donc:

$$w_s = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

### **3.1.6 La résolution du système de Saint-Venant sur un maillage volume fini régulier:**

### **3.1.7 Tester sur le cas 2D avec un fond plat:**

pour différentes valeurs de T et avec un schéma de Rusanov on obtient:



### **3.1.8 La correction MUSCL en 2D:**

### **3.1.9 Facultatif:**