
SeaTech École d'ingénieurs

Modélisation et Calculs Fluides et Structures




# Compte Rendu de TP 3: Calculs Numérique Avancés

## Résolution de l'équation du transport en 1D

**Enseignant:** Mr.Golay

**Etudiant:** Dupont Ronan

**Année Universitaire 2019-2020**

# I) Introduction

Le but de ce TP est de résoudre l'équation de transport. Pour se faire, nous n'allons pas écrire de programme mais simplement utiliser un programme écrit par Gloria Faccanoni. Nous étudierons différents caractéristiques de certains shémas numériques comme: gauche, droite, centré ainsi que d'autres schémas plus performants comme celui de Després-Lagoutière.

# II) Modèle physique

Durant ce TP, nous résolverons l'équation du transport qui est régi par le système suivant:

$$\begin{cases} \frac{du}{dt}(x,t) - c\frac{du}{dt}(x,t) = 0 \\ u(x,0) = g(x) \end{cases}$$

Si l'on considère un polluant de densité $u(x,t)$, après avoir "passé $u$ dans cette équation", on aura un polluant qui aura été transporté (translaté) d'une quantité $ct$ de manière conservatrice: on aura toujours la même densité.

On peut vérifier cela facilement en exploitant la solution de cette équation (que l'on obtient rapidement par la méthode des caractéristiques).
On a: $\forall (x,t) \in \mathbb{R}$ x $\mathbb{R}^+$, $u(x,t) = g(x-ct)$.
On voit bien le transport. On en déduit donc si $c > 0$, la quantité sera translatée vers la droite et si $c > 0$, elle sera translatée vers la gauche.

# III) Modèle numérique

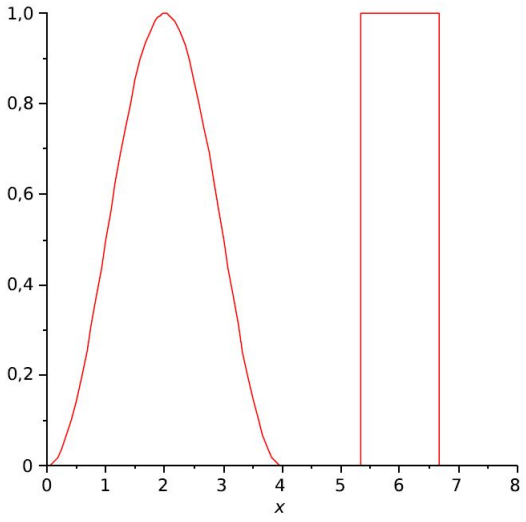
Pour résoudre l'équation suivante:

$$\begin{cases} \frac{du}{dt}(x,t) - c\frac{du}{dt}(x,t) = 0 \\ u(x,0) = g(x) \end{cases}$$

Avec:

$$g(x) = \begin{cases} \frac{1}{2} + \frac{1}{2}\sin\left(\frac{4\pi}{L}x - \frac{\pi}{2}\right) & \text{si } x \in ]0; \frac{L}{2}[, \\ 0 & \text{si } x \in ]\frac{L}{2}; \frac{2L}{3}[ \cup ]\frac{5L}{6}; L[, \\ 1 & \text{sinon.} \end{cases}$$

Où $g$ est représenté en $t = 0$ par la courbe ci-dessous avec L=8.



Nous allons dans un premier temps discrétiser l'intervalle I=[0,L] en n sous-intervalles de taille $h = \frac{L}{n-1}$. Ensuite, nous appliquerons des schémas numériques sur les $x_i$ avec $x_i = ih$ et $i \in$ [0,L].

## 1) Schémas numériques de base

Nous utiliserons 9 schémas numériques différents. On pose dans un premier temps $\alpha = c\frac{\Delta t}{\Delta x}$ et on obtient par différentes méthodes comme la méthode des différences finies ($Cf$ les 3 premiers schémas) les schémas suivants:

Décentré gauche:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_j^n - u_{j-1}^n}{\Delta x} = 0 \quad ie \quad u_j^{n+1} = u_j^n - \alpha(u_j^n - u_{j-1}^n) \tag{1}$$

Décentré droite:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_{j+1}^n - u_j^n}{\Delta x} = 0 \quad ie \quad u_j^{n+1} = u_j^n - \alpha(u_{j+1}^n - u_j^n) \tag{2}$$

Décentré centré:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0 \quad ie \quad u_j^{n+1} = u_j^n - \alpha\frac{u_{j+1}^n - u_{j-1}^n}{2} \tag{3}$$

Upwind:

$$u_j^{n+1} = u_j^n - (\frac{\alpha + |\alpha|}{2}(u_j^n - u_{j-1}^n) + \frac{\alpha - |\alpha|}{2}(u_{j+1}^n - u_j^n)) \tag{4}$$

Lax-Friedrichs:

$$u_j^{n+1} = \frac{1-\alpha}{2}u_{j+1}^n + \frac{1+\alpha}{2}u_{j-1}^n \tag{5}$$

Lax-Wendroff:

$$u_j^{n+1} = u_j^n - \alpha\frac{u_{j+1}^n - u_{j-1}^n}{2} + \alpha^2\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{2} \tag{6}$$

Beam-Warming (c>0):

$$u_j^{n+1} = u_j^n - \alpha((u_j^n - u_{j-1}^n) + \frac{1-\alpha}{2}(u_j^n - 2u_{j-1}^n + u_{j-2}^n)) \tag{7}$$

Fromm (c>0):

$$u_j^{n+1} = \frac{\alpha(\alpha-1)}{4}u_{j-2}^n + \frac{\alpha(5-\alpha)}{4}u_{j-1}^n + \frac{(1-\alpha)(\alpha+4)}{4}u_j^n + \frac{\alpha(\alpha-1)}{4}u_{j+1}^n \tag{8}$$

Anti-diffusif de Després-Lagoutière (c>0):

$$u_j^{n+1} = u_j^n - \alpha(g(u_{j-1}^n, u_j^n, u_{j+1}^n) - g(u_{j-2}^n, u_{j-1}^n, u_j^n))$$

$$o\grave{u}\ g(L,C,R) = \begin{cases} A, & si\ R \le A, \\ B, & si\ R \ge B, \\ R, & sinon, \end{cases} \quad et \quad \begin{cases} A = \max(L,C) + \frac{C - \max(L,C)}{\alpha} \\ B = \min(L,C) + \frac{C - \min(L,C)}{\alpha} \end{cases} \tag{9}$$

Par la suite, nous étudierons les schémas en fonction d'une CFL (condition de stabilité). De manière générale, si la cfl est inférieure à 1, le schéma est stable et si elle est supérieur, il est instable.

### 2) Adaptation des schéma numériques pour c<0

Certains schémas numériques ont été configurés pour répondre à un c particulier: c>0. Nous allons donc deviner ces schémas pour qu'ils fonctionnent aussi pour le cas où c<0.

Nous allons donc chercher à adapter les schémas 7, 8, 9.
Pour se faire il faut savoir que le transport se passe dans le sens inverse que si $c > 0$. On va donc poser $\alpha = -\alpha$. La résolution se faisant dans le sens inverse, on peut noter que tous les $u^n_{j-1}$ deviennent des $u^n_{j+1}$ et inversement.
Après simplification, nous obtenons les schémas numériques suivants:

Beam-Warming (c<0):

$$u^{n+1}_j = u^n_j + \alpha((u^n_j - u^n_{j+1}) + \frac{1-\alpha}{2}(u^n_j - 2u^n_{j+1} + u^n_{j+2})) \tag{10}$$

Fromm (c<0):

$$u^{n+1}_j = \frac{\alpha(\alpha+1)}{4}u^n_{j+2} + \frac{-\alpha(5+\alpha)}{4}u^n_{j+1} + \frac{(1+\alpha)(4-\alpha)}{4}u^n_j + \frac{\alpha(\alpha+1)}{4}u^n_{j-1} \tag{11}$$

Anti-diffusif de Després-Lagoutière (c>0):

$$u^{n+1}_j = u^n_j + \alpha(g(u^n_{j+1}, u^n_j, u^n_{j-1}) - g(u^n_{j+2}, u^n_{j+1}, u^n_j))$$
$$o\grave{u}\ g(L,C,R) = \begin{cases} A, & si\ R \le A, \\ B, & si\ R \ge B,\ et \\ R, & sinon, \end{cases} \begin{cases} A = \max(L,C) - \frac{C - \max(L,C)}{\alpha} \\ B = \min(L,C) - \frac{C - \min(L,C)}{\alpha} \end{cases} \tag{12}$$

### 3) Résolution en utilisant les schémas

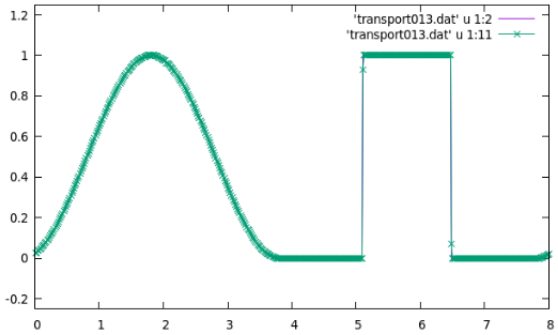
Afin d'utiliser les schémas numériques, nous n'allons pas utiliser de systèmes matriciel ou autre pour résoudre l'équation. En effet, ce serait une mauvaise manie car tous les schémas sont explicites, nous avons simplement à manipuler des listes représentant les $u^n_j$, $u^n_{j-1}$, $u^n_{j-2}$, $u^n_{j+1}$, $u^n_{j+2}$. En utilisant l'écriture vectorielle de Fortran90, on arrive très rapidement aux résolutions.

## IV) Résultats

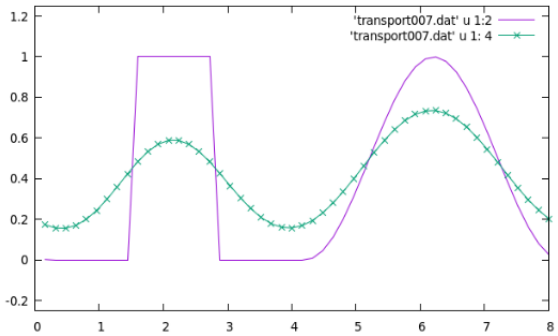
Pour chacun des schémas numériques, nous étudierons :

- La stabilité: un schéma est stable si $\exists\ c > 0 \mid \forall\ \delta t, n$ assez petits: $||u^n||_\infty \le ||u^0||_\infty$, $\forall n \le n\delta t \le T$. C'est à dire qu'on peut majorer la norme de n'importe quelle valeur du schéma par une constante multiplié par la norme de la valeur initiale.
  Par exemple, un schéma qui divergera ne pourra pas se faire majorer en norme.

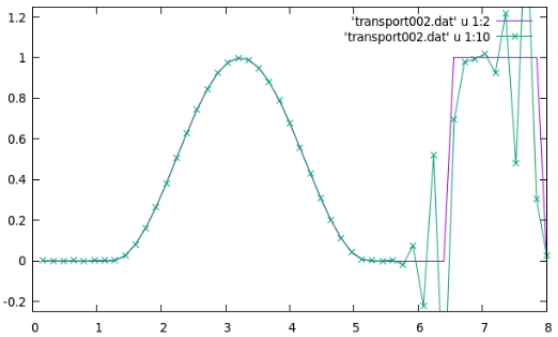
- La convergence: Si la courbe suit ou non la courbe réelle. Ci-dessous un schéma stable: il épouse bien la courbe réelle.




- La diffusion : Si la courbe "s'écrase": elle diminue en amplitude. Ci-dessous un schéma diffusif: la courbe simulée perd en amplitude.




- La dispersion: Si la courbe oscille par rapport à la courbe réelle. Ci-dessous un schéma dispersif: il oscille beaucoup.




## 1) Courbes de résultats

Afin d'obtenir des résultats, nous avons décidé de tracer sur un même graphiques les schémas de 4 à 9. Nous avons volontairement exclu les schémas 1 à 3 qui sont très souvent divergent ce qui fausserait la plupart des résultats. Nous les étudierons donc au cas par cas. Voici les courbes que nous avons obtenues:

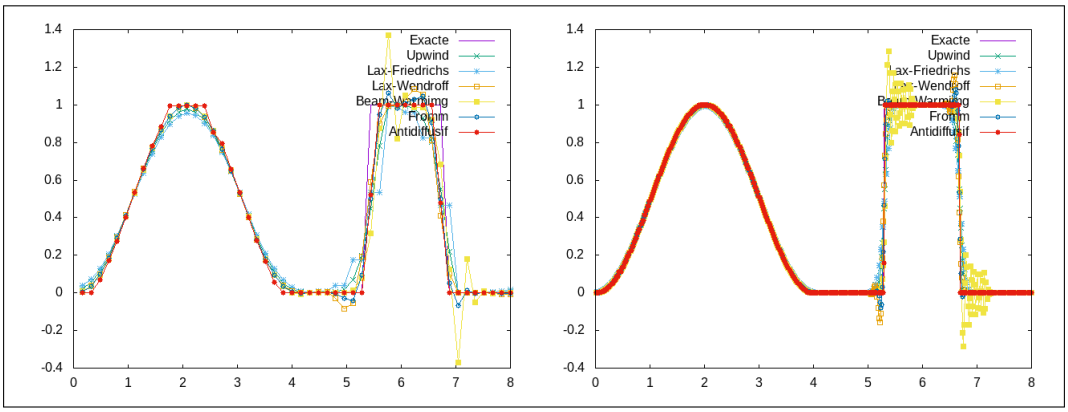



Figure 1: c=1, cfl=0.99, N=50,500

On peut voir que pour la cf=0.99 et c=1, les schémas 4 à 9 convergent. Certains font apparaître le phénomène de diffusion comme ont le voit nettement sur le schéma jaune de Beam-Warming sur le triangle.

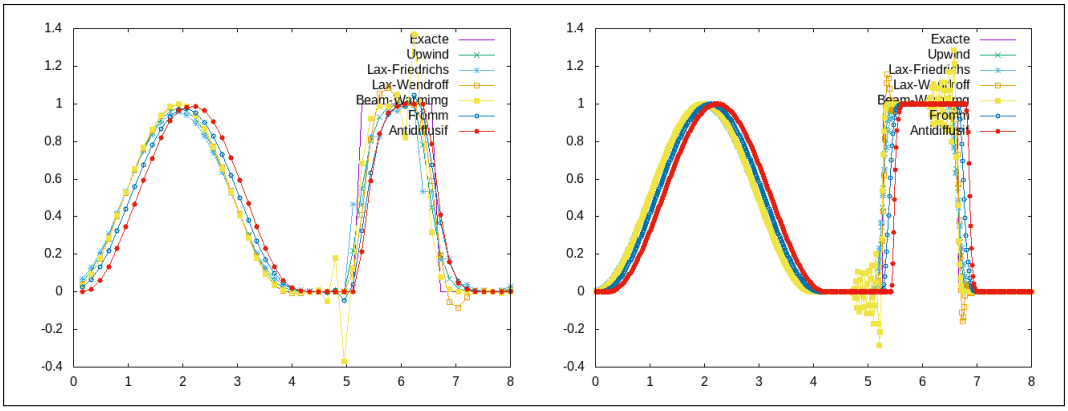



Figure 2: c=-1, cfl=0.99, N=50,500

On peut voir que pour la cf=0.99 et c=-1, les schémas 4 à 9 convergent. Certains font apparaître le phénomène de diffusion comme ont le voit nettement sur le schéma jaune de Beam-Warming sur le triangle. Cependant, ces courbes simulés font apparaître un décalage en x par rapport à la courbe réelle.

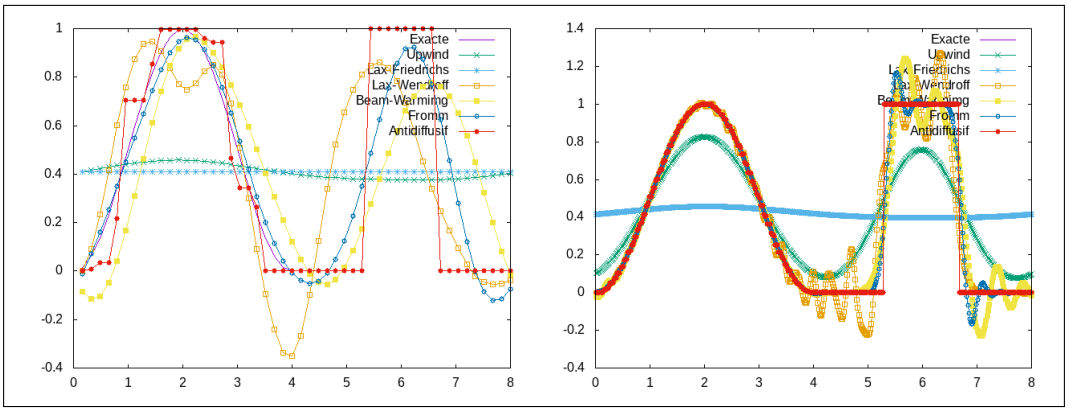



Figure 3: c=1, cfl=0.1, N=50,500

On peut voir que pour la cf=0.1 et c=1, les schémas 4 à 9 ne convergent pas tous. Par exemple, les schémas de Lax-Friedrichs et d'Upwind subissent une très forte diffusion qui fait tendre le résultat du schéma de Lax-Friedrichs vers une constante. Sinon les autres schémas convergent mais sont très dispersifs.

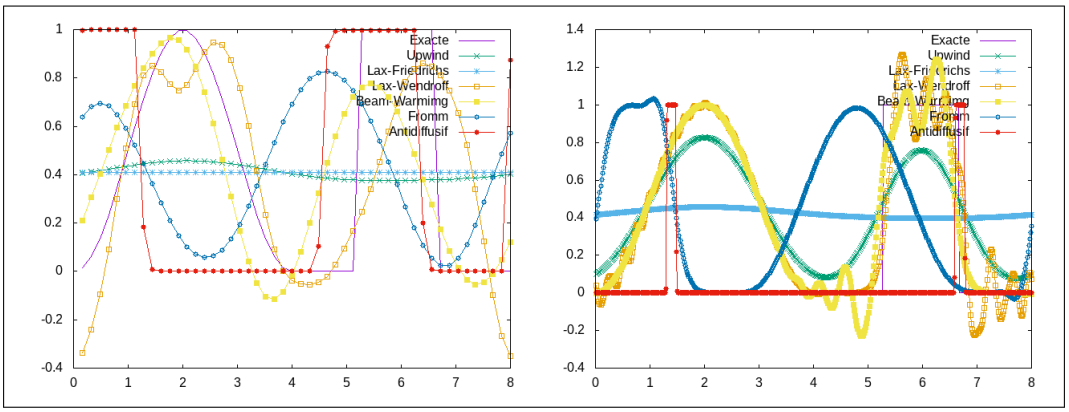



Figure 4: c=-1, cfl=0.1, N=50,500

On peut voir que pour la cf=0.1 et c=-1, tous les schémas semblent être décalés selon x par rapport à la solution exacte. Comme précédemment a de même les schémas de Lax-Friedrichs et d'Upwind qui subissent une très forte diffusion. Et les autres schémas sont très dispersifs sur l'approximation du rectangle.

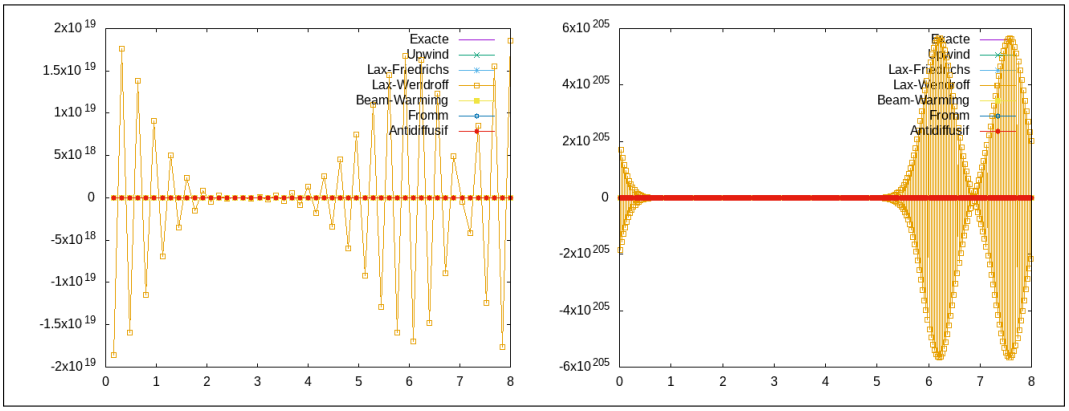



Figure 5: c=1, cfl=1.1, N=50,500

On peut voir que pour la cf=1.1 et c=-1, le schéma de Lax-Wendroff diverge. Si l'on enlève ce schéma du plot, on remarque que d'autres schémas divergent mais eux à d'autres vitesse. On ne peut pas conclure grand chose quant à la convergence des autres schémas. Nous verrons ceci plus précisément en les traçant cas par cas dans les tableaux de la prochaine partie.

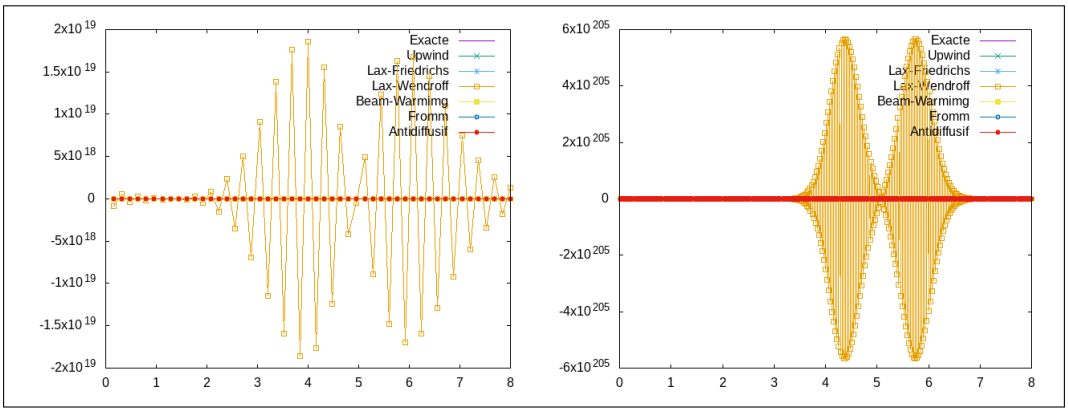



Figure 6: c=-1, cfl=1.1, N=50,500

On peut voir que pour la cf=1.1 et c=-1, le schéma de Lax-Wendroff diverge. Si l'on enlève ce schéma du plot, on remarque que d'autres schémas divergent mais eux à d'autres vitesse. On ne peut pas conclure grand chose quant à la convergence des autres schémas. Nous verrons ceci plus précisément en les traçant cas par cas dans les tableaux de la prochaine partie.

## 2) Schéma décentré gauche

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Non | Assez conséquent | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma peut être utilisé uniquement pour $c > 0$ avec la condition cfl=0.99.

## 3) Schéma décentré droite

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion=s'écrase | Stabilité |
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Non |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Non | Assez conséquent | Non |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma peut être utilisé uniquement pour $c < 0$ avec la condition cfl=0.99.

### 4) Schéma centré

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Oui | Non | Non | Non |
| cf1=0.1 | Oui | Non | Non | Non |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma n'est pas adapté à ce problème car il diverge dans tous les cas.

### 5) Schéma upwind

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Non | Assez conséquent | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Non | Assez conséquent | Non |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma est adapté quelque soit le $c$ mais il nécessite une condition cfl très proche 1 (de manière inférieur).

### 6) Schéma de Lax-Friedrichs

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Non |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Oui | Non | S'écrase vers y=0.4 | Non |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma est adapté quelque soit le $c$ mais il nécessite une condition cfl très proche 1 (de manière inférieur).

## 7) Schéma de Lax-Wendroff

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | mauvaise approximation" | Oui | Assez conséquente | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | mauvaise approximation | Oui | Assez conséquente | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma est adapté quelque soit le $c$ et quelque soit la condition de cfl tant qu'elle est inférieure à 1. Cependant les approximations sont pas très bonnes si l'on s'éloigne de la cfl=1.

## 8) Schéma de Beam-Warmimg

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Oui | Assez conséquente | Oui |
| cf1=1.1 | Très mauvaise approximation | Oui | Non | Oui |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Aucune dispersion | Oui | Non | Oui |
| cf1=1. | Aucune dispersion | Oui | Non | Oui |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | pas très bien sur le "rectangle" | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Oui | Assez conséquente | Oui |
| cf1=1.1 | Très mauvaise approximation | Oui | Non | Oui |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Aucune dispersion | Oui | Non | Oui |
| cf1=1. | Aucune dispersion | Oui | Non | Oui |

Ce schéma converge dans tous les cas mais son approximation n'est pas très précise si l'on a un n faible. Il est donc très adapté à condition d'avoir un n assez grand.

## 9) Schéma de Fromm

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | Faible dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Oui | légère | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Aucune dispersion | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Faible dispersion | Oui | Non | Oui |
| cf1=0.1 | pas très bien sur le "rectangle" | Oui | légère | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Aucune dispersion | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma semble très adapté quelque soit les cas tant que la condition cfl est respecté.

## 10) Schéma anti-diffusif de Després-Lagoutière

| c=1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
|---|---|---|---|---|
| cf1=0.99 | Faible dispersion | Oui | Non | Oui |
| cf1=0.1 | Mauvaise approximation sur le sin | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Aucune dispersion | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=50 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Faible dispersion | Oui | Non | Oui |
| cf1=0.1 | Mauvaise approximation sur le sin | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |
| c=-1, Nx=500 | Dispersion | Convergence | Diffusion | Stabilité |
| cf1=0.99 | Aucune dispersion | Oui | Non | Oui |
| cf1=0.1 | Aucune dispersion | Oui | Non | Oui |
| cf1=1.1 | Oui | Non | Non | Non |

Ce schéma semble très adapté quelque soit les cas tant que la condition cfl est respecté. En effet, les approximations sont très propres.

# V) Conclusion

Dans ce TP, nous avons pu étudier différents schémas numériques permettant de résoudre l'équation du transport en 1D. On a pu étudier différents paramètres comme la diffusion, dispersion, la stabilité ainsi que la convergence.
On a pu voir que les schémas de base (1-3) issu des différences finies sont pas très puissant tandis que les 3 deniers (7-9) sont beaucoup plus poussés. Le schéma anti-diffusion permet d'avoir des approximations quasi-parfaite sans aucune diffusion.
Cependant, quelque soit le schéma numérique, si la condition cfl est supérieur à 1, on aura très rarement un modèle qui converge.

# Annexe

```
!
! Resolution du
! probleme de Cauchy
! d_t u(x,t) + c d_x u(x,t) = 0
! u(x,t=0)=g(x)
! pour t>0 et x in [0;L] avec conditions au bord periodiques
!
! Creer une directory data dans laquelle seront sauvegarde les sorties
!
! --- Pour compiler && executer
! gfortran transport.f90 -o transport.o && ./transport.o
!
! --- Pour voir les ''films'' avec gnuplot
! cd data
! gnuplot plot_centree.gnu
! gnuplot plot_decentreegauche.gnu
! gnuplot plot_decentreedroite.gnu
! gnuplot plot_upwind.gnu
! gnuplot plot_laxfriedrichs.gnu
! gnuplot plot_laxwendroff.gnu
! gnuplot plot_beamwarming.gnu
! gnuplot plot_fromm.gnu
! gnuplot plot_antidiffusif.gnu
!


program transport

        implicit none
        double precision, parameter :: pi=2.0*acos(0.0)

        integer,                        parameter :: Nx    = 500          ! nombre de points
        double precision,       parameter :: L     = 8.0              ! domaine
        double precision,       parameter :: c     = 1.               ! vitesse du transport
        double precision,       parameter :: Tmax  = 24.0                ! borne en temps
        integer,                        parameter :: Smax  = 40                   ! nombre de sauvegardes
        double precision,       parameter :: cfl   = 0.99             ! coefficient CFL

        integer                                                 :: Nt, j, sauv
        double precision                                :: dx, dt, t
        double precision, dimension(Nx) :: x
        double precision, dimension(Nx) :: exacte
        double precision, dimension(Nx) :: centree
        double precision, dimension(Nx) :: decentreegauche
        double precision, dimension(Nx) :: decentreedroite
        double precision, dimension(Nx) :: upwind
        double precision, dimension(Nx) :: laxfriedrichs
        double precision, dimension(Nx) :: laxwendroff
        double precision, dimension(Nx) :: beamwarming
        double precision, dimension(Nx) :: fromm
        double precision, dimension(Nx) :: antidiffusif

        Nt = 0                                  ! numero de la sauvegarde
        dx = L/Nx                               ! pas d'espace
        dt = cfl*abs(c)*dx              ! pas de temps
        t  = 0.0                                ! instant

        x                               = (/(j*dx, j=1,Nx)/)    ! position
        exacte                  = (/(0., j=1,Nx)/)              ! solution exacte
        centree                 = (/(0., j=1,Nx)/)              ! solution avec schema centre
        decentreegauche = (/(0., j=1,Nx)/)              ! solution avec schema decentre a gauche
        decentreedroite = (/(0., j=1,Nx)/)              ! solution avec schema decentre a droite
        upwind                  = (/(0., j=1,Nx)/)              ! solution avec schema upwind
        laxfriedrichs   = (/(0., j=1,Nx)/)              ! solution avec schema de Lax-Friedrichs
        laxwendroff     = (/(0., j=1,Nx)/)              ! solution avec schema de Lax-Wendroff
        beamwarming     = (/(0., j=1,Nx)/)              ! solution avec schema de Beam-Warming
        fromm                   = (/(0., j=1,Nx)/)              ! solution avec schema de Fromm
        antidiffusif    = (/(0., j=1,Nx)/)              ! solution avec schema AntiDiffusif

        ! Initialisations
        call Calcul_Exacte(x,t,exacte)
        centree                 = exacte
        decentreegauche = exacte
        decentreedroite = exacte
        upwind                  = exacte
        laxfriedrichs   = exacte
        laxwendroff             = exacte
        beamwarming             = exacte
        fromm                   = exacte
        antidiffusif    = exacte

    ! Sauvegarde de la CI
    sauv = 0
```

Figure 7: Code du programme

```
    call sauvegarde(sauv,x,exacte,centree, decentreegauche,decentreedroite,upwind,laxfriedrichs, laxwendroff
        ➥, beamwarming,&
                    fromm, antidiffusif)


! ***********************************************************************
! MARCHE EN TEMPS
! ***********************************************************************
        do while (t<Tmax)

                Nt = Nt+1
                t  = t+dt

        ! calcul de la solution exacte
                call Calcul_Exacte(x,t,exacte)
                ! calcul des solutions approchees
                call Calcul_Centree(centree)
                call Calcul_Decentree_Gauche(decentreegauche)
                call Calcul_Decentree_Droite(decentreedroite)
                call Calcul_Upwind(upwind)
                call Calcul_Lax_Friedrichs(laxfriedrichs)
                call Calcul_Lax_Wendroff(laxwendroff)
                call Calcul_Beam_Warming(beamwarming)
                call Calcul_Fromm(fromm)
                call Calcul_Anti_Diffusif(antidiffusif)
                ! sauvegarde dans un fichier de toutes les solutions
                if ( floor(t*Smax/Tmax)>floor((t-dt)*Smax/Tmax) ) then
                        sauv=sauv+1
                        call sauvegarde(sauv,x,exacte,centree, decentreegauche,decentreedroite,upwind,
                            ➥laxfriedrichs, laxwendroff, beamwarming,&
                                                        fromm, antidiffusif)
        end if
        end do

        call scriptgnuplot(sauv)



! ***********************************************************************
! SOUS-PROGRAMMES
! ***********************************************************************
        contains


        subroutine sauvegarde(sauv,x,exacte,centree, decentreegauche,decentreedroite,&
                              upwind,laxfriedrichs, laxwendroff, beamwarming, fromm, antidiffusif)
            integer                                         :: j
            integer,                                           intent(in)     :: sauv
            double precision, dimension(Nx), intent(in) :: x, exacte,centree, decentreegauche,
                ➥decentreedroite,upwind,&
                                                               laxfriedrichs, laxwendroff, beamwarming, fromm,
                                                                   ➥antidiffusif
                character(len=30)                                          :: file_name ! nome du fichier a
                    ➥sauvegarder

            write(file_name,'("./data/transport",i3.3,".dat")')  sauv
            open(unit=11,file=file_name)
            do j=1,Nx
                write(11,*) x(j), &
                        exacte(j), &
                        centree(j), &
                        decentreegauche(j), &
                        decentreedroite(j), &
                        upwind(j), &
                        laxfriedrichs(j), &
                        laxwendroff(j), &
                        beamwarming(j), &
                        fromm(j), &
                        antidiffusif(j)
            end do
            close(11)
            print "(A,I3,A)", "␣=====␣Sauvegarde␣num.", sauv, "␣====="
        end subroutine


                subroutine Calcul_Exacte(x,t,exacte)
                        double precision, dimension(Nx), intent(in)      :: x
                        double precision,                                   intent(in)      :: t
                        double precision, dimension(Nx), intent(out)     :: exacte
                        double precision, dimension(Nx)                                             :: xi
                        xi = x-c*t - floor((x-c*t)/L)*L
                        where (xi<L*0.5)
                                exacte = (1.+sin(4.*pi*xi/L-0.5*pi))*0.5
                        elsewhere (xi<L*0.66 .or. xi>L*0.833)
                                exacte = 0.0
```

Figure 8: Code du programme

```
                elsewhere
                        exacte = 1.0
                end where
        end subroutine Calcul_Exacte


        subroutine Calcul_Centree(centree)
                double precision, dimension(Nx), intent(inout)  :: centree
                integer
                    ➥          :: j
                double precision, dimension(Nx)                                 :: centree_P,
                    ➥centree_M
                double precision                                                ::
                    ➥alpha
                alpha = c*dt/dx
    ! conditions periodiques
                centree_P(1:Nx-1) = centree(2:Nx)
                centree_P(Nx)     = centree(1)
                centree_M(2:Nx)   = centree(1:Nx-1)
                centree_M(1)      = centree(Nx)
    ! schema
                centree = centree-alpha*0.5*(centree_P-centree_M)
        end subroutine Calcul_Centree


        subroutine Calcul_Decentree_Gauche(decentreegauche)
                double precision, dimension(Nx), intent(inout)  :: decentreegauche
                integer
                    ➥          :: j
                double precision, dimension(Nx)                                 :: decentreegauche_P
                    ➥, decentreegauche_M
                double precision                                                ::
                    ➥alpha
                alpha = c*dt/dx
    ! conditions periodiques
                decentreegauche_M(2:Nx)   = decentreegauche(1:Nx-1)
                decentreegauche_M(1)      = decentreegauche(Nx)
    ! schema
                decentreegauche = decentreegauche-alpha*(decentreegauche-decentreegauche_M)
        end subroutine Calcul_Decentree_Gauche


        subroutine Calcul_Decentree_Droite(decentreedroite)
                double precision, dimension(Nx), intent(inout)  :: decentreedroite
                integer
                    ➥          :: j
                double precision, dimension(Nx)                                 :: decentreedroite_P
                    ➥, decentreedroite_M
                double precision                                                ::
                    ➥alpha
                alpha = c*dt/dx
    ! conditions periodiques
                decentreedroite_P(1:Nx-1) = decentreedroite(2:Nx)
                decentreedroite_P(Nx)     = decentreedroite(1)
    ! schema
                decentreedroite = decentreedroite-alpha*(decentreedroite_P-decentreedroite)
        end subroutine Calcul_Decentree_Droite


        subroutine Calcul_Upwind(upwind)
                double precision, dimension(Nx), intent(inout)  :: upwind
                double precision, dimension(Nx)                                 :: upwind_P,
                    ➥upwind_M
    ! conditions periodiques
                upwind_P(1:Nx-1) = upwind(2:Nx)
                upwind_P(Nx)     = upwind(1)
                upwind_M(2:Nx)   = upwind(1:Nx-1)
                upwind_M(1)      = upwind(Nx)
    ! schema
                upwind = upwind-(dt/dx)*(c+abs(c))*0.5*(upwind-upwind_M)-(dt/dx)*(c-abs(c))*0.5*(
                    ➥upwind_P-upwind)
        end subroutine Calcul_Upwind


        subroutine Calcul_Lax_Friedrichs(laxfriedrichs)
                double precision, dimension(Nx), intent(inout)  :: laxfriedrichs
                double precision, dimension(Nx)                                 :: laxfriedrichs_P,
                    ➥laxfriedrichs_M
                double precision                                                ::
                    ➥alpha
                alpha = c*dt/dx
    ! conditions periodiques
                laxfriedrichs_P(1:Nx-1) = laxfriedrichs(2:Nx)
                laxfriedrichs_P(Nx)     = laxfriedrichs(1)
                laxfriedrichs_M(2:Nx)   = laxfriedrichs(1:Nx-1)
                laxfriedrichs_M(1)      = laxfriedrichs(Nx)
```

Figure 9: Code du programme

```
! schema
            laxfriedrichs = laxfriedrichs_P + laxfriedrichs_M - alpha*(laxfriedrichs_P-
                ➥laxfriedrichs_M)
            laxfriedrichs = 0.5*laxfriedrichs
    end subroutine Calcul_Lax_Friedrichs


    subroutine Calcul_Lax_Wendroff(laxwendroff)
            double precision, dimension(Nx), intent(inout)  :: laxwendroff
            double precision, dimension(Nx)                                 :: laxwendroff_P,
                ➥laxwendroff_M
            double precision                                                ::
                ➥alpha
            alpha = c*dt/dx
! conditions periodiques
            laxwendroff_P(1:Nx-1) = laxwendroff(2:Nx)
            laxwendroff_P(Nx)     = laxwendroff(1)
            laxwendroff_M(2:Nx)   = laxwendroff(1:Nx-1)
            laxwendroff_M(1)      = laxwendroff(Nx)
! schema
            laxwendroff = laxwendroff &
                                - 0.5*alpha*(laxwendroff_P - laxwendroff_M) &
                                + 0.5*alpha**2*(laxwendroff_P-2*laxwendroff+laxwendroff_M)
    end subroutine Calcul_Lax_Wendroff


    subroutine Calcul_Beam_Warming(beamwarming)
            double precision, dimension(Nx), intent(inout)  :: beamwarming
            double precision, dimension(Nx)                                 :: beamwarming_P,
                ➥beamwarming_PP, beamwarming_M, beamwarming_MM, g_P, g_M
            double precision                                                ::
                ➥alpha
            alpha = c*dt/dx
! conditions periodiques
            beamwarming_P(1:Nx-1)  = beamwarming(2:Nx)
            beamwarming_P(Nx)      = beamwarming(1)
            beamwarming_PP(1:Nx-1) = beamwarming_P(2:Nx)
            beamwarming_PP(Nx)     = beamwarming_P(1)
            beamwarming_M(2:Nx)    = beamwarming(1:Nx-1)
            beamwarming_M(1)       = beamwarming(Nx)
            beamwarming_MM(2:Nx)   = beamwarming_M(1:Nx-1)
            beamwarming_MM(1)      = beamwarming_M(Nx)
! schema
if (c>0) then
                    beamwarming=beamwarming-dt/dx*(beamwarming-beamwarming_M+0.5*(1.-alpha)*(
                        ➥beamwarming-2.*beamwarming_M+beamwarming_MM))
            else
                    ! a implementer
                    beamwarming = 0.0*beamwarming
            end if
    end subroutine Calcul_Beam_Warming


    subroutine Calcul_Fromm(fromm)
            double precision, dimension(Nx), intent(inout)  :: fromm
            double precision, dimension(Nx)                                 :: fromm_P, fromm_PP
                ➥, fromm_M, fromm_MM
            double precision                                                ::
                ➥alpha
            alpha = c*dt/dx
! conditions periodiques
            fromm_P(1:Nx-1)  = fromm(2:Nx)
            fromm_P(Nx)      = fromm(1)
            fromm_PP(1:Nx-1) = fromm_P(2:Nx)
            fromm_PP(Nx)     = fromm_P(1)
            fromm_M(2:Nx)    = fromm(1:Nx-1)
            fromm_M(1)       = fromm(Nx)
            fromm_MM(2:Nx)   = fromm_M(1:Nx-1)
            fromm_MM(1)      = fromm_M(Nx)
! schema
if (c>0) then
                    fromm = alpha*(alpha-1.)*0.25*fromm_MM &
                              + alpha*(5.-alpha)*0.25*fromm_M &
                              + (1.-alpha)*(alpha+4.)*0.25*fromm &
                              + alpha*(alpha-1.)*0.25*fromm_P
            else
                    ! a implementer
                    fromm = 0.0*fromm
            end if
    end subroutine Calcul_Fromm


    subroutine Calcul_Anti_Diffusif(antidiffusif)
            double precision, dimension(Nx), intent(inout)  :: antidiffusif
            double precision, dimension(Nx)                                 :: antidiffusif_P,
                ➥antidiffusif_PP, antidiffusif_M, antidiffusif_MM
```

Figure 10: Code du programme

```
                double precision, dimension(Nx)                                 :: A_P, A_M, B_P,
                    ➥B_M, g_P, g_M
                double precision                                                ::
                    ➥alpha
                alpha = c*dt/dx
    ! conditions periodiques
                antidiffusif_P(1:Nx-1)  = antidiffusif(2:Nx)
                antidiffusif_P(Nx)      = antidiffusif(1)
                !antidiffusif_PP(1:Nx-1) = antidiffusif_P(2:Nx)
                !antidiffusif_PP(Nx)     = antidiffusif_P(1)
                antidiffusif_M(2:Nx)    = antidiffusif(1:Nx-1)
                antidiffusif_M(1)       = antidiffusif(Nx)
                antidiffusif_MM(2:Nx)   = antidiffusif_M(1:Nx-1)
                antidiffusif_MM(1)      = antidiffusif_M(Nx)
    ! schema
    if (c>0) then
                    A_P = max(antidiffusif_M ,antidiffusif  )+(antidiffusif  -max(antidiffusif_M
                        ➥ ,antidiffusif  ))/alpha
                    A_M = max(antidiffusif_MM,antidiffusif_M)+(antidiffusif_M-max(
                        ➥antidiffusif_MM,antidiffusif_M))/alpha
                    B_P = min(antidiffusif_M ,antidiffusif  )+(antidiffusif  -min(antidiffusif_M
                        ➥ ,antidiffusif  ))/alpha
                    B_M = min(antidiffusif_MM,antidiffusif_M)+(antidiffusif_M-min(
                        ➥antidiffusif_MM,antidiffusif_M))/alpha
                    where(antidiffusif_P<=A_P)
                            g_P=A_P
                    elsewhere(antidiffusif_P>B_P)
                            g_P=B_P
                    elsewhere
                            g_P=antidiffusif_P
                    end where
                    where(antidiffusif<=A_M)
                            g_M=A_M
                    elsewhere(antidiffusif>B_M)
                            g_M=B_M
                    elsewhere
                            g_M=antidiffusif
                    end where
                    antidiffusif = antidiffusif - alpha*(g_P-g_M)
                else
                    ! a implementer
                    antidiffusif = 0.0*antidiffusif
                end if
            end subroutine Calcul_Anti_Diffusif


    !*****************************************************************
    ! Ecriture des fichier "plot_XXX.gnu" de commandes gnuplot
    subroutine scriptgnuplot(sauv)
            integer, intent(in) :: sauv
            integer                         :: i, j, h
            character(len=30)       :: file_name

            open(unit=13,file="./data/plot_centree.gnu")
            open(unit=14,file="./data/plot_decentreegauche.gnu")
            open(unit=15,file="./data/plot_decentreedroite.gnu")
            open(unit=16,file="./data/plot_upwind.gnu")
            open(unit=17,file="./data/plot_laxfriedrichs.gnu")
            open(unit=18,file="./data/plot_laxwendroff.gnu")
            open(unit=19,file="./data/plot_beamwarming.gnu")
            open(unit=20,file="./data/plot_fromm.gnu")
            open(unit=21,file="./data/plot_antidiffusif.gnu")

            do i=13,21
                    write(i,'("set␣yrange[-0.25:1.25];")')
                    h=i-10
                    write(i,'(a58,i2,a6)') "plot␣'transport000.dat'␣u␣1:2␣w␣l,␣'transport000.dat'␣u␣1:",
                        ➥h,"␣w␣lp;"
                    write(i,'("pause␣-1;")')
                    do j=1,sauv
                            h=i-10
                            write(i,'(a15,i3.3,a27,i3.3,a10,i2,a6)') "plot␣'transport",j,".dat'␣u␣1:2␣w␣
                                ➥l,␣'transport",j,".dat'␣u␣1:",h,"␣w␣lp;"
                            write(i,'("pause␣0.25;")')
                    end do
                    write(i,'("pause␣-1;")')
                    close(i)
            end do

            open(unit=30,file="./data/plot_comparaison_finale.gnu")
                    write(i,'("set␣yrange[-0.25:1.25];")')
                    write(30,'(a18,i3.3,a6)') "fichier='transport",sauv,".dat';"
                    write(30,*) "plot␣fichier␣u␣1:␣2␣w␣l␣ti␣'Exacte'\"
                    !write(30,*) "    , fichier u 1: 3 w lp ti 'Centre'\"
                    !write(30,*) "    , fichier u 1: 4 w lp ti 'Decentre a gauche'\"
                    !write(30,*) "    , fichier u 1: 5 w lp ti 'Decentre a droite'\"
```

Figure 11: Code du programme