Technische Universität München
Fakultät für Mathematik

Simulations invariantes par transformation Galiléenne pour des lois de conservations comportant une composante densité-vitesse sur un maillage en mouvement

Mémoire de Master de Ezra Rozier

Themensteller : Herr Prof Dr. Oliver Junge

Betreuer : Herr Prof Dr. Oliver Junge

Abgabedatum : 15.10.2020

Par la suivante je déclare que ce mémoire est entièrement le résultat de mon propre travailsauf quand cela est indiqué. Je n'ai utilisé que les ressources listées en référence.

Ezra Rozier	08.10.2020 Date

Abstract

Ce mémoire de Master traite d'une implémentation parallèle en Julia d'un schéma des volumes finis sur un maillage en mouvement pour résoudre un système hyperbolique de lois de conservation. L'objet principal de ce mémoire est de clarifier le travail de V.Springel en [1] ainsi aue de donner une courte implémentation en Julia du code AREPO.

Table des matières

A Introduction

Pour calculer la dynamique de formation de structures en astrophysique, il est devenu essentiel d'utiliser des méthodes numériques. Grâce à la précision accrue de ces calculs, et en les comparant avec ce qui se passe réellement, nous sommes capables de prédire les caractéristiques précises de composants physiques inconnus. Par exemple, nous sommes en mesure d'être plus précis dans l'évaluation des caractéristiques des halos de matière noire.
Comme on peut le voir, même si nous nous intéressons aux halos de matière noire qui peuvent être représentés comme des particules sans collision, les seuls effets que nous sommes encore capables de détecter résident dans la vitesse et la position des particules classiques, il nous faut donc calculer des simulations cosmologiques hydrodynamiques. Dans ce domaine, il existe plusieurs approches possibles, les deux principales sont SPH (smoothed particles hydrodynamics) qui est une méthode lagrangienne, et l'hydrodynamique à base de maillage qui est une méthode eulérienne, à laquelle on peut ajouter de l'AMR (adaptive mesh refinement). Ces deux méthodes (maillage mobile et raffinement du maillage en affinant le nombre de cellules) sont appelées stratégies de raffinement, pour la méthode du maillage mobile, on parle de r-raffinement et pour la méthode AMR de h-raffinement. Dans ce qui suit, j'appellerai le raffinement du maillage mobile r-refinement et le raffinement (ou déraffinement) h-raffinement.
Ces deux méthodes ont leurs avantages et leurs inconvénients. Nous pouvons notamment mentionner ce qui suit : SPH, en tant qu'approche lagrangienne est invariante par transformation galiléenne, mais elle a du mal à calculer les solutions de choc. D'autre part, l'approche eulérienne a une précision spatiale d'ordre élevé mais n'est pas invariante par transformation galiléenne. Pour faire face à cela, et essayer de combiner les avantages des deux méthodes, je vais, dans ce travail, présenter les travaux de V.Springel, 2010 [1]. Il a mélangé les approches eulérienne et lagrangienne sur un code ALE (arbitrary Lagrangien-Eulérien) basé sur l'approche SPH combinée avec AMR.
Comme je l'ai déjà écrit, une telle méthode doit être applicable à la fois à la matière classique et à la matière noire. Comme la partie gravitationnelle de la simulation sera prise en compte par une simulation à N corps, je me concentrerai uniquement sur la dynamique des particules classiques et ne m'occuperai pas des questions gravitationnelles. Dans la section B, je présenterai les aspects mathématiques d'un tel travail et quels sont les domaines d'intérêt. Dans la section C, je présenterai mon implémentation d'un tel code parallèle et comment

il s'adapte à Julia. Dans la section D, je présenterai mes résultats pour un problème de test spécifique.

B Aspects mathématiques

La méthode de Springer [1] consiste en une méthode des volumes finis appliquée à un système de lois de conservation et peut être généralisée à tout système de lois de conservation hyperbolique pour le mouvement des particules (tant que l'une d'entre elles a une composante de vitesse et de densité), mais je me concentrerai sur l'équation d'Euler pour ce qui suit. Pour ce faire, je commencerai par discuter de la forme de l'équation d'Euler et de sa résolution en volume fini, puis j'aborderai la méthode et présenterai tous ses aspects de calcul.
Ce schéma de volume fini sera appliqué sur un maillage mobile de Voronoï, je discuterai donc également du choix de la vitesse de chaque point de génération de maillage. Nous verrons plus tard que le mouvement de la maille ne sera pas complètement lagrangien.

B - 1 L'équation d'Euler

L'équation d'Euler est un système de lois de conservation masse-moment-énergie et peut être présenté comme suit :

$$\frac{\partial U}{\partial t} + \nabla \cdot F(U) = 0 \tag{1}$$

Avec $U = U(\boldsymbol{x}, t)$. Quant aux lois de conservation auxquelles on s'intéresse, on a :

$$U = \begin{pmatrix} \rho \\ \rho \boldsymbol{v} \\ \rho e \end{pmatrix} = \begin{pmatrix} \rho \\ \rho \boldsymbol{v} \\ \rho u + \frac{1}{2}\rho \boldsymbol{v}^2 \end{pmatrix} \tag{2}$$

$$F(U) = \begin{pmatrix} \rho \boldsymbol{v} \\ \rho \boldsymbol{v}\boldsymbol{v}^T + P \\ (\rho e + P)\boldsymbol{v} \end{pmatrix} \tag{3}$$

Avec $P = (\gamma - 1)\rho u$.
Nous allons utiliser une stratégie des volumes finis en intégrant le fluide dans un volume V_i de masse m_i, de moment $\boldsymbol{p}_i$ et d'énergie E_i :

$$Q_i = \int_{V_i} U dV = \begin{pmatrix} m_i \\ \boldsymbol{p}_i \\ E_i \end{pmatrix} \tag{4}$$

Ainsi

$$\frac{dQ_i}{dt} = -\int_{\partial V_i} [\underbrace{F(U)}_{\text{(1) \& loi de Green}} - \underbrace{U\boldsymbol{w}^T}_{\text{variation du volume}}] d\boldsymbol{n} \tag{5}$$

avec $\boldsymbol{n}$ étant le vecteur unitaire normal a la surface de la maille, et $\boldsymbol{w}$ la vitesse de chaque point à la surface.
Sachant que l'on travaille dans un maillage de Voronoï, alors en 2D (resp. 3D) ∂V_i est une réunion de segments (resp. polygones convexes) $\boldsymbol{A}_{ij}$. En notant :

$$F_{ij} = -\frac{1}{A_{ij}} \int_{Aij} [F(U) - U\boldsymbol{w}^T] d\boldsymbol{A}_{ij} \tag{6}$$

$$\frac{dQ_i}{dt} = -\sum_{i \neq j} A_{ij} F_{ij} \tag{7}$$

Ainsi le schéma est le suivant :

$$Q_i^{(n+1)} = Q_i^n - \Delta t \sum_j A_{ij} \hat{F}_{ij}^{(n+\frac{1}{2})} \tag{8}$$

$\hat{F}_{ij}^{(n+\frac{1}{2})}$ est ici le flux moyen à travers la frontière de ij prédit un demi pas de temps dans le futur. Par la suite, l'objectif sera de trouver une stratégie pour calculer $\hat{F}_{ij}^{(n+\frac{1}{2})}$. Pour cela, j'essaierai de mettre en place une bonne approximmation moyenne.

B - 2 La méthode

Avant de commencer à être plus précis sur les aspects mathématiques ou informatiques, je vais brièvement présenter les différentes étapes de la méthode pour donner un cadre à ce qui suit. C'est l'idée centrale de la méthode, nous verrons plus tard que l'on peut apporter de petites variations à cette méthode :
Supposons que du pas de temps précédent on a :

- $\boldsymbol{r}_i$ la position de chaque point générant le maillage
- $Q_i = (m_i, \boldsymbol{p}_i, E_i)^T$ les variables conservatives de la maille i.

L'algorithme
(i) Créer un mesh de Voronoï avec les r_i, cela donne :
- $\boldsymbol{s}_i$ centre de masse de la maille i
- V_i volume de la maille i
- A_{ij} et $\boldsymbol{f}_{ij}$ respectivement l'aire et le centre des interfaces interfaces entre les mailles i et j.

(ii) Calculer $\boldsymbol{W}_i = (\rho_i, \boldsymbol{v}_i, P_i)^T$ grâce aux Q_i.
(iii) Évaluer le gradient de la densité, des composantes de la vitesse et de la pression dans chaque maille et utiliser un limiteur de pente pour éviter l'introduction d'extrema et de dépassements.
(iv) Calculer $\boldsymbol{w}_i$ la vitesse des points générant le maillage pour chaque $\boldsymbol{r}_i$.
(v) Utiliser le critère de Courant pour calculer Δt_i.
(vi) Pour chaque interface du maillage par extrapolation linéaire en temps et en espace, déterminer l'état du fluide de part et d'autre du centre de l'interface puis calculer $\hat{F}_{ij}^{(n+\frac{1}{2})}$ en résolvant un problème de Riemann 1D.
(vii) Utiliser (8) pour calculer Q_i^{n+1}.
(viii) Décaler les $\boldsymbol{r}_i$ avec $\boldsymbol{w}_i$

Dans cette section, je suivrai le cadre de la méthode, excepté pour la géométrie du maillage et de sa construction qui seront abordés dans la partie suivante.

B - 3 Evaluation des gradients

(iii) Évaluer le gradient de la densité, des composantes de la vitesse et de la pression dans chaque maille et utiliser un limiteur de pente pour éviter l'introduction d'extrema et de dépassements.

Nous évaluerons le gradient en partant de :

$$\{\int_{\partial V} \phi d\boldsymbol{n} = \int_V \nabla\phi dV\} \Rightarrow \{\langle\nabla\phi\rangle_i \simeq -\frac{1}{V_i}\sum_j \phi(f_{ij})\boldsymbol{A}_{ij}\} \tag{9}$$

⚠Ici $\boldsymbol{A}_{ij}$ est le vecteur normal à l'interface entre i et j, de norme égale à l'aire et pointant de i vers j.
N.B. : on pourrait calculer facilement $\phi(f_{ij}) = \frac{1}{2}(\phi_i + \phi_j)$ mais par la géométrie du maillage, il n'est pas très coûteux dêtre plus précis.
Supposons que dans le voisinage de $\boldsymbol{r}_i$ on a une bonne approximation au point $\boldsymbol{r}$ of $\phi : \phi(\boldsymbol{r}) = \phi_i + \boldsymbol{b}\cdot(\boldsymbol{r} - \boldsymbol{r}_i)$, $\boldsymbol{b}$ est le gradient local.
Alors :

$$V_i\langle\nabla\phi\rangle = \int_{\partial V_i} \phi d\boldsymbol{n} = \sum_{i\neq j}\int_{A_{ij}} [\phi_i + \boldsymbol{b}\cdot(\boldsymbol{r} - \boldsymbol{r}_i)]\frac{\boldsymbol{r}_j - \boldsymbol{r}_i}{r_{ij}} dA \tag{10}$$

⚠$\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$
Soit

$$\boldsymbol{c}_{ij} = \frac{1}{A_{ij}}\int_{A_{ij}} \boldsymbol{r} dA - \frac{\boldsymbol{r}_i + \boldsymbol{r}_j}{2} \tag{11}$$

le vecteur pointant du milieu du segment $[\boldsymbol{r}_i; \boldsymbol{r}_j]$ au centre de la face.
Sachant que $\phi_j = \phi_i + \boldsymbol{b}\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)$:

$$V_i\langle\nabla\phi\rangle_i = -\sum_{i\neq j}[\frac{\phi_i + \phi_j}{2} + \boldsymbol{b}\cdot\boldsymbol{c}_{ij}]\frac{\boldsymbol{r}_{ij}}{r_{ij}}A_{ij} \tag{12}$$

Avec la relation suivante : $(\boldsymbol{b}\cdot\boldsymbol{c}_{ij})\boldsymbol{r}_{ij} = \underbrace{(\boldsymbol{b}\cdot\boldsymbol{r}_{ij})}_{=\phi_i - \phi_j}\boldsymbol{c}_{ij} + \boldsymbol{b}\times(\boldsymbol{r}_{ij}\times\boldsymbol{c}_{ij})$.

Et

$$\boldsymbol{b}\times\sum_{i\neq j}\frac{\boldsymbol{r}_{ij}\times\boldsymbol{c}_{ij}}{r_{ij}}A_{ij} = \boldsymbol{b}\times\sum_{i\neq j}\frac{\boldsymbol{r}_{ij}}{r_{ij}}\times\int_{A_{ij}}[\boldsymbol{r} - \frac{\boldsymbol{r}_i + \boldsymbol{r}_j}{2}]dA \tag{13}$$

Il apparaît que $\frac{\boldsymbol{r}_{ij}}{r_{ij}} = \boldsymbol{n}$, $-\int_{\partial V} \boldsymbol{r} \times d\boldsymbol{n} = \int_V \nabla \times \boldsymbol{r} dV$ ainsi $\int_{\partial V} \boldsymbol{r}_i \times d\boldsymbol{n} = \overrightarrow{0}$ car $\boldsymbol{r}_i$ est constant ici et V_i est un volume fermé, aussi $\boldsymbol{r}_{ij} \times \frac{\boldsymbol{r}_i + \boldsymbol{r}_j}{2} = \boldsymbol{r}_{ij} \times \boldsymbol{r}_i = \boldsymbol{r}_i \times \boldsymbol{r}_j$. De plus le terme $\sum_{i \neq j} \int_{A_{ij}} d\boldsymbol{n} \times \boldsymbol{r}$ disparaît parce que ∂V est une surface fermée. Finalement on a l'égalité (13) gale à $\overrightarrow{0}$.
Et :

$$\langle \nabla \phi \rangle_i = \frac{1}{V_i} \sum_{i \neq j} Aij[(\phi_j - \phi_i) \frac{\boldsymbol{c}_{ij}}{r_{ij}} - \frac{\phi_i + \phi_j}{2} \cdot \frac{\boldsymbol{r}_{ij}}{r_{ij}}] \tag{14}$$

Maintenant pour tous $\boldsymbol{r} \in V_i$ on pose $\rho(\boldsymbol{r}) = \rho_i + \langle \nabla \rho \rangle_i \cdot (\boldsymbol{r} - \boldsymbol{s}_i)$, il en va de même pour la vitesse et la densité.
N.B. : J'aurais pu utiliser $\boldsymbol{r}_i$ et non $\boldsymbol{s}_i$, j'ai choisi $\boldsymbol{s}_i$ car on s'intéresse plus au mouvement de la masse quáu mouvement de la maille. Cela montre l'importance de garder $\boldsymbol{r}_i \simeq \boldsymbol{s}_i$.

B - 4 Appliquer un limiteur de pente au gradient

> **(iii)** Évaluer le gradient de la densité, des composantes de la vitesse et de la pression dans chaque maille et **utiliser un limiteur de pente pour éviter l'introduction d'extrema et de dépassements**.

Pour éviter l'apparition de nouveaux extrema et dépassements, j'applique un limiteur de pente qui garde la valeur de la variable intensive à l'interface $\boldsymbol{f}_{ij}$ entre celles des deux centres de masse $\boldsymbol{s}_i$ et $\boldsymbol{s}_j$.
Je remplace $\langle \nabla \phi \rangle_i$ par $\langle \nabla \phi \rangle'_i = \alpha_i \langle \nabla \phi \rangle_i$. Avec

$$\alpha_i = \min_{j \text{ voisin de } i} (1, \psi_{ij})$$

$$\psi_{ij} = \begin{cases} \frac{\phi_i^{\max} - \phi_i}{\Delta \phi_{ij}} & \text{si } \Delta \phi_{ij} > 0 \\ \frac{\phi_i - \phi_i^{\min}}{\Delta \phi_{ij}} & \text{si } \Delta \phi_{ij} < 0 \\ 1 & \text{sinon} \end{cases}$$

avec $\Delta \phi_{ij} = \langle \nabla \phi \rangle_i \cdot (\boldsymbol{f}_{ij} - \boldsymbol{s}_i)$ estimation de la variation entre le centre de la face et le centre de masse de la maille.
on a aussi $\phi_i^{\max} = \max_{j \text{ voisin de } i \text{ incluant } i} (\phi_j)$ et $\phi_i^{\min} = \min_{j \text{ voisin de } i \text{ incluant } i} (\phi_j)$.
Les oscillations ne sont pas techniquement évitées car ce n'est pas TVD.

B - 5 Fixer les $\boldsymbol{w}_i$

> **(iv)** Calculer $\boldsymbol{w}_i$ la vitesse des points générant le maillage pour chaque $\boldsymbol{r}_i$.

Pour un schéma presque purement lagrangien on fixerait $\boldsymbol{w}_i = \boldsymbol{v}_i$ (ce sera le plus souvent le cas) ce qui est le plus arrangeant pour une équation d'advection mais je vais aussi explorer deux variations dans la valeur de $\boldsymbol{w}_i$ pour conserver une forme de régularité au maillage.

B - 5.1 Garder les mailles rondes

Pour conserver $\boldsymbol{r}_i \simeq \boldsymbol{s}_i$ une possibilité est d'avoir une maille ronde (une maille est ronde ssi $2R_i = \text{diam}(V_i)$ avec $R_i = \sqrt{\frac{V_i}{\pi}}$ en 2D et $R_i = \sqrt[3]{\frac{V_i}{4\pi}}$ en 3D).
Si la valeur $d_i = |\boldsymbol{r}_i - \boldsymbol{s}_i| > \eta R_i$ nous ajouterons une composante proportionnelle à la vitesse locale du son ($c_i = \sqrt{\frac{\gamma P_i}{\rho_i}}$). Pour adoucir de facteur, il y aura un virage :

$$\boldsymbol{w}_i' = \boldsymbol{w}_i + \chi \begin{cases} 0 & \text{pour } d_i < 0.9\eta R_i \\ c_i \frac{\boldsymbol{s}_i - \boldsymbol{r}_i}{d_i} \cdot \frac{d_i - 0.9\eta R_i}{0.2\eta R_i} & \text{pour } 0.9\eta R_i \leq d_i < 1.1\eta R_i \\ c_i \frac{\boldsymbol{s}_i - \boldsymbol{r}_i}{d_i} & \text{pour } d_i \geq 1.1\eta R_i \end{cases}$$

Pour ce faire, j'introduis deux nouveaux facteurs : η and χ. Généralement j'utiliserai $\eta = 0.25$ and $\chi = 1$.

B - 5.2 Garder masse-volume constant

Pour des raisons de calcul, il peut être intéressant de maintenir constant un facteur dépendant de la masse et du volume de chaque cellule. En effet, la masse est généralement un bon indicateur de la zone la plus active, ce qui permet de diminuer le volume lorsque la masse augmente, ce qui peut être un moyen peu coûteux et efficace d'avoir une adaptation du mouvement des mailles.
Dans ce qui suit, je vais présenter une façon de rendre la valeur $K_i := \frac{m_i}{\tilde{m}} + \frac{V_i}{\tilde{V}}$ constante dans l'espace. Avec $\tilde{m}$ and $\tilde{V}$ constantes définies apriori comme valeurs maximales de m_i et V_i.
Soit $n(\boldsymbol{x})$ la densité de point générant le maillage en $\boldsymbol{x}$ et $n_0(\boldsymbol{x})$ la densité idéale (au sens où : $\forall i\ K_i = \tilde{K}$). À n on associe les positions $\boldsymbol{x}_i$ et à n_0 les positions $\boldsymbol{q}_i$ et on note : $\boldsymbol{x}_i = \boldsymbol{q}_i + \epsilon \boldsymbol{d}_i$ with $\epsilon = 1$ pour cette situation. On suppose que $\boldsymbol{d}$ s'exprime comme le gradient d'un champ scalaire : $\boldsymbol{d} = -\nabla\Psi$.
On va faire varier ϵ dans le temps et étudier les variations de densité n.

Avec n_ϵ la densité de points générant le maillage $\boldsymbol{x}_i^\epsilon = \boldsymbol{q}_i + \epsilon \boldsymbol{d}_i$ au premier ordre :

$$n_\epsilon(\boldsymbol{x} + \epsilon \boldsymbol{d}) = \epsilon n(\boldsymbol{x} + \epsilon \boldsymbol{d}) + (1 - \epsilon) n_0(\boldsymbol{x})$$

On fixe la vitesse d'un point : $\boldsymbol{v} = \frac{d\boldsymbol{x}^\epsilon}{d\epsilon} = \boldsymbol{d}$.
Alors l'équation de continuité de Lagrange donne : $\frac{dn}{d\epsilon} + n\nabla \cdot \boldsymbol{v} = 0$. Ce qui donne en $\epsilon = 0$ ce aui suit :

$$\nabla^2 \Psi = \frac{n(\boldsymbol{x}+\boldsymbol{d})}{n_0(\boldsymbol{x})} - 1 \simeq \frac{n(\boldsymbol{x})}{n_0(\boldsymbol{x})} - 1.$$

Calculons n_0, On a pour le maillage idéal $m_i = \frac{\rho(\boldsymbol{q}_i)}{n_0(\boldsymbol{q}_i)}$ et $V_i = \frac{1}{n_0(\boldsymbol{q}_i)}$, alors :

$$n_0(\boldsymbol{x}) = \frac{1}{\tilde{K}}\left(\frac{\rho(\boldsymbol{x})}{m_i} + \frac{1}{V_i}\right)$$

Enfin, on trouve le déplacement $\boldsymbol{d} = -\nabla\Psi$ avec $\nabla^2\Psi = \frac{\tilde{K} n(\boldsymbol{x})}{\frac{\rho(\boldsymbol{x})}{\tilde{m}} + \frac{1}{\tilde{V}}} - 1$.
Pour des conditions aux bords périodiques, l'existence d'une solution à l'équation de Poisson dans un espace périodique infinie impose : $\tilde{K} = \frac{V_{\text{tot}}}{\sum_i (\frac{\rho_i}{\tilde{m}} + \frac{1}{\tilde{V}})^{-1}}$.
En effet on impose $\int_{V_{\text{tot}}} \nabla^2 \Psi = 0$ ainsi :

$$\int_{V_{\text{tot}}} \frac{\tilde{K} n(\boldsymbol{x})}{\frac{\rho(\boldsymbol{x})}{\tilde{m}} + \frac{1}{\tilde{V}}} = V_{\text{tot}}$$

Alors :

$$\tilde{K} \sum_i \left(\frac{\rho_i}{\tilde{m}} + \frac{1}{\tilde{V}}\right)^{-1} \underbrace{\int_{V_i} n(\boldsymbol{x})}_{=1} = V_{\text{tot}}$$

avec V_i la maille déterminée par $\boldsymbol{x}_i$.
Finalement :

$$\tilde{K} = \frac{V_{\text{tot}}}{\sum_i (\frac{\rho_i}{\tilde{m}} + \frac{1}{\tilde{V}})^{-1}}$$

Et on corrige la vitesse des points générant le maillage par : $\boldsymbol{w}_i' = \boldsymbol{w}_i + \kappa \frac{\boldsymbol{d}_i}{\Delta t_i}$ generally with $\kappa = 0.5$.

B - 6 Calculer les pas de temps

(v) Utiliser le critère de Courant pour calculer Δt_i.

On peut soit utiliser un pas de temps global (identique pour toutes les mailles) ou alors un pas de temps individuel (sur-mesure pour chaque maille).

B - 6.1 Pas de temps global

On va utiliser ici un critère de Courant global modifié simplement par le mouvement de la maille. Dans ce cas :

$$\Delta t_i = C_{\text{CFL}} \frac{R_i}{c_i + |\boldsymbol{v}_i \text{-} \boldsymbol{w}_i|} \tag{15}$$

$$\delta t = \min(\Delta t_i) \tag{16}$$

B - 6.2 Pas de temps individuel

I semble plus intéressant d'utiliser un pas de temps individuel : on peut calculer plus souvent l'état d'une maille proche du mouvement que celui d'une maille qui en est éloignée.Pour cela on va construire une hiérarchie : supposons le pas de temps Δt_i calculé et qu'on a choisi une constante Δt comme base, on peut associer $k_i = \lfloor \log_2(\frac{\Delta t}{\Delta t_i}) \rfloor + 1$ à chaque maille. Cette hiérarchie sera mise en place en k_i décroissant. Le pas de temps effectif de chaque maille sera $\frac{\Delta t}{2^{k_i}}$.

Évaluer Δt_i

L'évaluation de Δt_i doit prendre en compte deux choses : le critère de Courant pour la maille i et le temps minimum avant lequel une onde venant d'une autre maille arrive à la maille i surtout quand elle est plus rapide que la vitesse locale du son c_i.

Pour cette seconde condition on introduit une vitesse du "signal" entre les mailles i et j :

$$v_{ij}^{\text{sig}} = c_i + c_j - \boldsymbol{v}_{ij} \cdot \frac{\boldsymbol{r}_{ij}}{r_{ij}} \tag{17}$$

⚠$\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j$ et $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$.

Cette vitesse est toujours plus grande que la vitesse d'une onde se propageant de i vers j (ou de j vers i) et assure que i (ou j) sera de nouveau active avant que l'onde arrive dessus.

Maintenant pour calculer Δt_i on procède comme suit :

$$\Delta t_i = C_{\text{CFL}} \min(\tau_i, \frac{R_i}{c_i + |\boldsymbol{w}_i - \boldsymbol{v}_i|}) \tag{18}$$

$$\tau_i = \min_{j \neq i}(\frac{r_{ij}}{v_{ij}^{\text{sig}}}) \tag{19}$$

Je discuterai dan la section C d'une méthode arborescente pour calculer efficacement chaque pas de temps.

L'algorthme implémenté avec un pas de temps individuel sera changé dans la dernière partie.

Changer l'algorithme

Supposons qu'on a calculé tous les k_i. Supposons aussi que nous transitons entre les états t et $t + \frac{\Delta t}{2^{k_{\max}}}$ avec :
$k_{\max} = \max(k_i),\ \exists! k \in \mathbb{N},\ t + \frac{\Delta t}{2^{k_{\max}}} = m\frac{\Delta t}{2^k},\ m \in \mathbb{N},\ 2 \wedge m = 1.$
N.B. : - la transition sera toujours faite avec le pas de temps $\frac{\Delta t}{2^{k_{\max}}}$
- t est toujours multiple de $\frac{\Delta t}{2^k}$ pour un certain k car toutes les transitions entre états seront faites avec un pas de temps de la forme $\frac{\Delta t}{2^{k_{\max}}}$
- Nous verrons plus tard en **C - 4** pourquoi on a toujours $k \leq k_{\max}$.

Appelons "active" chaque maille i telle que $k_i \geq k$.
Pour toute interface $i \leftrightarrow j$, on l'appelle "active" si $\max(k_i, k_j) \geq k$.

Alors l'algorithme seprésente comme suit (les changements sont écrits en rouge) :

(i) Créer le maillage de Voronoï et fixer une organisation arborescent à vos données
(ii) Calculer $\boldsymbol{W}_i$ **de chaque maille active**
(iii) Evaluer le gradient **de chaque maille active**
(iv) Calculer $\boldsymbol{w}_i$
(v) Par l'organisation arborescente, **calculer chaque pas de temps des mailles actives, ce sera fait tel que** $k_i^{\text{new}} \geq k$ **puis** $k_{\max}$ **pour trouver le prochain pas de temps où il y a des mailles actives, mettre à jour** t **par** $t + \frac{\Delta t}{2^{k_{\max}}}$ **et calculer** k **pour déterminer les mailles actives**
(vi) Calculer le flux **à travers les interfaces actives, pour une interface** $i \leftrightarrow j$ **le flux sera calculer pour un pas de temps :** $\frac{\Delta t}{2^{\max(k_i, k_j)}}$
(vii) Calculer $Q_i^{(n+1)}$ **pour chaque cellule voisine d'une interface active**
(viii) Décaler les $\boldsymbol{r}_i$ avec les $\boldsymbol{w}_i$.

B - 7 Calcul du flux

(vi) Pour chaque interface du maillage par extrapolation linéaire en temps et en espace, déterminer l'état du fluide de part et d'autre du centre de l'interface puis calculer $\hat{F}_{ij}^{(n+\frac{1}{2})}$ en résolvant un problème de Riemann 1D.

B - 7.1 Vitesse de l'interface

L'interface entre deux mailles de vitesse $\overrightarrow{w_i}$ et $\overrightarrow{w_j}$ des points générant r_i et r_j possède unbe vitesse que je caractériserai par la vitesse de son centre : soit $\overrightarrow{OM}(t) = \frac{\overrightarrow{Or_i}+\overrightarrow{Or_j}}{2}$, $\overrightarrow{x} = \frac{\overrightarrow{Mr_i}}{|Mr_i|}(t=0)$ et $\overrightarrow{y} = \frac{\overrightarrow{Mf_{ij}}}{|Mf_{ij}|}(t=0)$ et travaillons dans le repère orthonormal $(M, \overrightarrow{x}, \overrightarrow{y})$ (s'il n'est pas en sens direct faisons $i \leftrightarrow j$). Supposons que le point M est immobile (pour cela, il suffit de translater toutes les vitesses de $-\frac{\overrightarrow{w}_i+\overrightarrow{w}_j}{2}$), donc $\overrightarrow{w}_i = -\overrightarrow{w}_j = v\overrightarrow{x} + \omega|Mr_i|\overrightarrow{y}$.

Dans le temps, l'angle entre la médiatrice et $\overrightarrow{x}$ est égal à : $\alpha(\delta t) = \omega\delta t$ aet donc le déplacement de f_{ij} sera : $\overrightarrow{d} = \alpha(\delta t)|Mf_{ij}|(t=0)\overrightarrow{y}$ (si on ne prend pas en compte la composante selon $\overrightarrow{x}$ de $\overrightarrow{w}_i$, on fait cette approximation car la divergence entre r_i et r_j change la forme de la face, mais ces changements dépendent des autres voisins, ainsi au premier ordre, au sens où on ne prend en compte que les deux cellules en interface, on peut négliger v).
Et ainsi la vitesse est

$$\overrightarrow{w} = \frac{\overrightarrow{w_i} + \overrightarrow{w_j}}{2} + (\overrightarrow{w_i} - \overrightarrow{w_j}) \cdot (\overrightarrow{Of_{ij}} - \frac{\overrightarrow{Or_i} + \overrightarrow{Or_j}}{2}) \cdot \frac{\overrightarrow{r_j r_i}}{|\overrightarrow{r_j r_i}|^2} \tag{20}$$

B - 7.2 Calcul du flux

Nous allons maintenant calculer le flux. Pour cela nous calculerons l'état $(\rho, \boldsymbol{v}, P)^T$ au centre de l'interface en tant que solution à un problème de Riemann 1D. Ainsi d'abord, je translate toutes les vitesses de $\boldsymbol{w}$, puis je prédis l'état plus loin en espace et dans le temps de $\frac{\delta t}{2}$ et tourne le référentiel de manière à ce que $(r_i; f_{ij})$ soit colinéaire avec l'axe des abscisses.

Soit $\boldsymbol{W} = (\rho, \boldsymbol{v}, P)^T = \boldsymbol{W}_i$ ou $\boldsymbol{W}_j$.

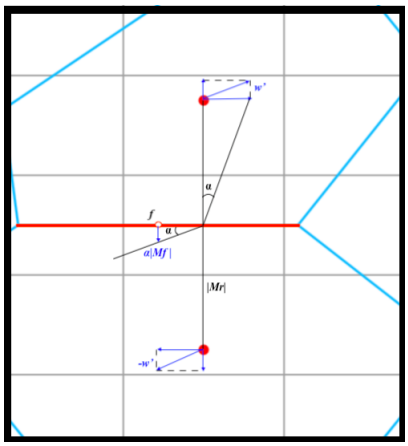


FIGURE 1 – Géométrie de la face

$$\boldsymbol{W}' = \boldsymbol{W} + \begin{pmatrix} 0 \\ \boldsymbol{w} \\ 0 \end{pmatrix} \tag{21}$$

$$\boldsymbol{W}'' = \boldsymbol{W}' + \frac{\partial \boldsymbol{W}'}{\partial \boldsymbol{r}} \cdot (\boldsymbol{f} - \boldsymbol{s}) + \frac{\partial \boldsymbol{W}'}{\partial t} \cdot \frac{\delta t}{2} \tag{22}$$

Pour évaluer $\frac{\partial \boldsymbol{W}'}{\partial t}$ j'utilise $\frac{\partial \boldsymbol{W}'}{\partial t} + A(\boldsymbol{W}')\frac{\partial \boldsymbol{W}'}{\partial \boldsymbol{r}} = 0$ avec $A(\boldsymbol{W}')$ étant la jacobienne en $\boldsymbol{W}'$:

$$A(\boldsymbol{W}) = \begin{pmatrix} \boldsymbol{v} & \rho & 0 \\ 0 & \boldsymbol{v} & \frac{1}{\rho} \\ 0 & \gamma P & \boldsymbol{v} \end{pmatrix} \tag{23}$$

Finalement il ne reste que la rotation de la vitesse pour que $(r_i; f_{ij})$ coïncide avec l'axe des abscisses :

$$\boldsymbol{W}''' = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \Lambda & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \boldsymbol{W}'' \tag{24}$$

Puis utiliser le solveur de Riemann R :

$$\boldsymbol{W}_F = R(\boldsymbol{W}_i''', \boldsymbol{W}_j''') \tag{25}$$

Maintenant pour calculer le flux on revient dans le référentiel du laboratoire et on prend en compte le mouvement de la face :

$$\boldsymbol{W}_{\text{lab}} = \begin{pmatrix} \rho \\ \boldsymbol{v}_{\text{lab}} \\ P \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \Lambda^{-1} & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \boldsymbol{W}_F + \begin{pmatrix} 0 \\ \boldsymbol{w} \\ 0 \end{pmatrix} \tag{26}$$

$$\hat{F} = F(U) - U\boldsymbol{w}^T = \begin{pmatrix} \rho(\boldsymbol{v}_{\text{lab}} - \boldsymbol{w}) \\ \rho \boldsymbol{v}_{\text{lab}}(\boldsymbol{v}_{\text{lab}} - \boldsymbol{w})^T + P_{\text{lab}} \\ \rho \boldsymbol{e}_{\text{lab}}(\boldsymbol{v}_{\text{lab}} - \boldsymbol{w}) + P\boldsymbol{v}_{\text{lab}} \end{pmatrix} \tag{27}$$

Avec $\boldsymbol{e}_{\text{lab}} = \frac{\boldsymbol{v}_{\text{lab}}^2}{2} + \frac{P}{(\gamma-1)\rho}$.

C Aspects d'implémentation

Dans cette section, je vais m'intéresser à tous les aspects de la mise en œuvre que j'ai pris en compte. Je présenterai donc dans une première partie la mise en œuvre d'une structure de données arborescente quad/oct pour une construction de maillage de Voronoï. Puis dans une deuxième partie, je mettrai en évidence les principaux éléments de la parallélisation de mon algorithme et je mettrai l'accent sur l'utilisation de Julia pour ce travail. Enfin, je reviendrai sur plusieurs aspects secondaires du calcul.

C - 1 La construction de Voronoï

Nous travaillerons ici dans l'espace $\Omega \subset \mathbb{R}^d$ borné convexe avec $d \in \{2, 3\}$.
Un maillage de Voronoï est dini comme suit : pour $(r_i)_{i\in I},\ r_i \in \Omega\ ,\ I \subset \mathbb{N}\ ,\ I$ fini, de points générant un maillage, les mailles seront comme suit,
$\Omega_i = \{r \in \Omega \mid i \in \arg\max_{j\in I} |r - r_j|\}$.

N.B. : On a :
Pour tous i, Ω_i fermé convexe avec $\forall r \in \overset{\circ}{\Omega}_i\ |\arg\max_{j\in I}|r - r_j|| = 1$ et $\forall r \in \partial\Omega_i\ |\arg\max_{j\in I}|r - r_j|| \geq 2$.

On peut facilement montrer pour $d = 2$:

$\forall i, j \in I\ \Omega_i \cap \Omega_j$ est vide, un point ou un segment

$\forall J \subset I,\ |J| \geq 3,\ \bigcap_{j\in J} \Omega_j$ est vide ou un point

et pour $d = 3$:

$\forall i, j \in I\ \Omega_i \cap \Omega_j$ est vide, un point, un segment ou un plan borné convexe.

$\forall J \subset I,\ |J| = 3,\ \bigcap_{j\in J} \Omega_j$ est vide, un point ou un segment

$\forall J \subset I,\ |J| \geq 4,\ \bigcap_{j\in J} \Omega_j$ est vide ou un point

Ainsi, $\{\overset{\circ}{\Omega}_i\}_{i\in I}$ est une triangulation de Ω.
On appellera interface $i \leftrightarrow j$ le $\Omega_i \cap \Omega_j$ quand c'est un segment pour $d = 2$ et un plan pour $d = 3$ ($\Omega_i \cap \Omega_j$ est inclus dans la médiatrice entre r_i et r_j.
Pour calculer un maillage de Voronoï, nous devons savoir que c'est le dual d'un maillage de Delaunay. Un maillage de Delaunay pour $(r_i)_{i\in I}$ est comme suit : c'est un maillage en triangle dont les sommets sont les r_i et le triangle $(r_i r_j r_k)$ est dans le maillage s'il n'y a pas d'autre point générateur à l'intérieur de son cercle circonscrit.

Ainsi les sommets d'un maillage de Voronoï sont tous les centres des cercles circonscrits et et les arêtes joignent les centres des cercles circonscrits à ses trois sommets les plus proches pour $d = 2$ et ses quatre plus proches pour $d = 3$.
La description précise de comment construire un maillage de Voronoï se basant sur les propriétés d'un maillage de Delaunay sont données en [1], je n'en parlerai pas ici.
Pour le construire informatiquement j'ai suivi [1] dans un carré de taille 0.95×0.95. J'ai choisi cette taille car la librairie GeometricalPredicates [10] en Julia est définie seulement sur un carré de taille 1×1 et j'avais besoin d'une marge pour y mettre les mailles fantôme représentant les conditions au bord périodiques/symétriques.
Enfin, j'ai stocké les points générant le maillage dans un oct-/quadtree [13] ce qui donne la possibilité d'accéder à une maille en $\mathcal{O}\log(n)$ ce qui améliore grandement la complexité de l'algorithme. Dans ces cellules, j'ai stocké toutes les informations dont j'allais avoir besoin au cours de la boucle.
J'ai construit deux arbres similaires au milieu de mon code pour le calcul de la vitesse $\boldsymbol{w}_i$ (TreePM code) et du pas de temps.

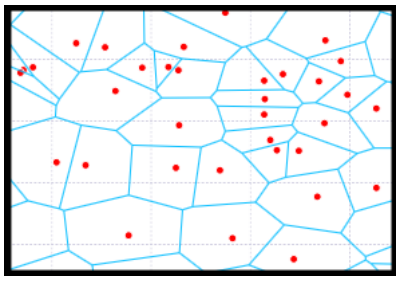


FIGURE 2 – Maillage de Voronoï

Pour la construction de Voronoï j'ai commencé avec [12] et adapté cela à la structure de mes données (restreindre la construction à un carré, adapter le tracé de graphes à Gadfly...).

C - 2 Parallélisation

Je présenterai le processus de parallélisation en trois parties : tout d'abord, j'expliquerai comment j'ai décomposé mon domaine pour paralléliser le calcul et comment cela modifie l'algorithme, puis je présenterai la norme classique

de parallélisation, MPI, et enfin j'expliquerai pourquoi j'ai préféré utiliser la méthode de calcul parallèle intégrée de Julia.

C - 2.1 Decomposition de domaine

Comme premier test, j'ai simplement brutalement décomposé mon domaine en quatre (ou neuf) surfaces égales.
Maintenant la question est : Quel algorithme ai-je utilisé pour calculer les mailles fantôme ? Mon algorithme est basé sur les propriétés du maillage de Delaunay. En effet, pour un processeur donné, je veux trouver tous les voisins dans le maillage globale de Voronoï des mailles stockées dans ce processeur : ces voisins extérieurs sont appelés mailles fantômes. Nous verrons cela en **C - 3.3**.
L'aspect le plus important de la parallélisation est d'avoir le moins de communication de données possible entre les processeurs, car elle est coûteuse. Toutes les mailles extérieures prises en compte dans le calcul d'un processeur donné seront importées sous forme de mailles fantôme. Ensuite, à la fin de chaque boucle, j'exporte les données de chaque processeur de calcul vers le processeur principal.

C - 2.2 La norme MPI

MPI (pour Message Passing Interface) est une norme conçue pour le calcul parallèle. Au départ, il a été conçu pour prendre en charge le calcul parallèle sur des entités multidimensionnelles (en tant que cluster d'ordinateurs hétérogènes) dans des codes C/C++ et Fortran, mais il est maintenant également implémenté dans d'autres langages comme Python ou Julia. Dans Julia, il est considéré comme la norme externe pour le calcul parallèle car il existe une norme de parallélisation interne. Il existe cependant une bibliothèque MPI [11] qui est par exemple utilisée comme norme pour l'implémentation initiale de l'AREPO [1].

C - 2.3 La norme de parallélisation de Julia

Julia a mis en place sa propre norme de calcul distribué. En effet, dans Julia vous pouvez paralléliser l'exécution des canaux : avec cette structure, j'ai pu paralléliser mon code avec une implémentation facile. L'accès aux données était également très pratique lorsque je devais établir une communication entre les processeurs.

C - 3 Autres aspects

Dans cette section, je reviendrai sur plusieurs autres aspects de l'algorithme, en particulier la stratégie de raffinement du maillage ainsi que la résolution de Riemann et la recherche sur les mailles fantôme.

C - 3.1 Adaptive mesh refinement

Dans le code original de l'AREPO, la stratégie de raffinement du maillage se déroule comme suit : si la masse ou le volume d'une cellule dépasse un seuil supérieur donné, le maillage sera raffiné en ajoutant un point de génération presque exactement à la même position que le précédent. Si ces propriétés descendent en dessous d'un seuil inférieur, le maillage sera redéfini en supprimant simplement le point de génération.
Cette stratégie peut être améliorée assez facilement, par exemple en triant les cellules et en raffinant (resp. déraffinant) le plus lourd (resp. le plus léger) 5% ou/et le plus grand (resp. le plus petit) 5%, sinon en modifiant le critère de raffinement. On peut utiliser par exemple un critère de mouvement avec le gradient des valeurs ou un critère qui prend en compte l'endroit où l'erreur est la plus créée. Pour l'approximation des erreurs, voir l'estimation des erreurs a priori pour les lois de conservation non linéaires calculées avec les volumes finis [2] ainsi que l'estimation des erreurs a posteriori (qui demande beaucoup plus d'efforts) [3].
Un autre moyen d'améliorer la condition de raffinement est d'utiliser un détecteur de choc ([4] section 8) mais cela demande plus de travail a priori. Dans cette stratégie, vous recherchez les grandes variations de vos données et vous les affinez dans ces régions.

C - 3.2 Le solveur de Riemann

Un autre aspect de la discussion concerne le choix du solveur de Riemann. En effet, dans cet exemple pour l'équation d'Euler en 2D/3D, il suffit de calculer une matrice jacobienne à 5×5 pour avoir un solveur de Riemann exact. Mais lorsque l'on dispose d'un système plus large de lois de conservation hyperboliques, il devient crucial d'utiliser un solveur de Riemann non jacobien. En effet, il est possible de calculer une bonne approximation de la solution d'un problème de Riemann, uniquement avec des évaluations de la fonction de flux.
Comme nous travaillons avec des lois de conservation hyperboliques, tous les

solveurs non exacts par la suite reposent sur une approximation de la solution pour le flux. Avoir un solveur non exact et non jacobien tel que le flux HLL (Harten-Lax-van Leer) permet d'obtenir de la vitesse sans perdre trop de précision (c'est par exemple ce qui est actuellement implémenté dans la version rapide d'AREPO sur Github [9] avec HLLC pour l'équation d'Euler classique et HLLD pour MHD). Cependant, ces solveurs HLL sont toujours construits pour un problème spécifique (HLLC, HLLD...) car ils s'appuient précisément sur le calcul du système aux valeurs propres. L'amélioration des grands systèmes de lois de conservation pour les HLL a été explorée dans [6]. Pour un problème avec des variables plus conservatrices, comme dans [8], le solveur de Krylov-Riemann [5] ou le solveur MUSTA [7] sont intéressants car ils ne reposent que sur l'évaluation de la fonction et non sur le calcul d'un système aux valeurs propres (à l'exception d'une estimation globale de la plus grande vitesse de propagation). De plus, ils résolvent tous deux mieux les discontinuités que le flux de Lax-Wendorff [5].

C - 3.3 Recherche de fantôme

Pour la recherche de fantômes, il y a deux algorithmes qui peuvent être utilisés [1] :

Premier algorithme

Prendre un point générant r dans un processeur local et faire :

- Soit h une longueur et s deux fois le diamètre du plus grand cercle circonscrit adjacent.

Si $h < s : h = \alpha h,\ (\alpha > 1)$ recommencer.

Sinon si il n'y a pas de point en dehors du processeur local et à l'intérieur du cercle centrée en r : l'algorithme est terminé.

Sinon (si il y a un point en dehors du processeur local et à l'intérieur du cercle centrée en r) : ajouter le(s) point(s) au maillage de Voronoï mettre s à jour et recommencer la recherche.

Second algorithme

Supposons que l'on commence avec un maillage de Voronoï.

Prendre chaque triangle de Delaunay qui a au moins un sommet à l'intérieurdu processeur local et trouver tous les points générateurs des autres processeurs à l'intérieur des cercles circonscrits associés.

Répéter cette opération tant qu'il y a encore des points qui s'ajoutent.

Le premier algorithme donne au moins tous les voisins des mailles à l'intérieur d'un processeur, le second donne exactement tous les voisins. J'ai utilisé le second algorithme.

Finalement, mon algorithme sera vu en page suivante.

Initialisation : points générateurs, variables conservatives et intensives

I/ Décomposer le domaine

II/ Entrer dans le premier calcul parallèle :

(i) Trouver les mailles fantôme

(ii) Construire l'arbre

(iii) Contruire le maillage de Voronoï local

III/ Mettre à jour toutes les données dépendant de la géométrie pour les mailles fantôme (volume, vitesse du son, radius...)

IV/ Construire l'arbre pour le calcul des pas de temps (qui concatène les arbres stockés dans les processeur travaillant), le stocké dans le processeur principal et calculer les pas de temps en parallèle.

V/ Trouver le premier temps δt auquel il y aura des mailles actives

VI/ Entrer dans le second calcul parallèle :

(i) Calculer les gradients et utiliser le limiteur de pente

(ii) Calculer la vitesse des points générateurs

VII/ Entrer dans la boucle de calcul : **while** $t < t_{\mathbf{final}}$

1 - Construire l'arbre pour le calcul des pas de temps (qui concatène les arbres stockés dans les processeur travaillant), le stocké dans le processeur principal et calculer les pas de temps en parallèle pour les mailles actives.

2 - Mettre à jour les gradients et les vitesses des mailles fantôme

3 - Calculer les flux pour les mailles actives

4 - Calculer le prochain temps où il y a des mailles actives , $t+\frac{\Delta t}{2^k_{\max}}$, et trouver m t.q. $t + \frac{\Delta t}{2^{k_{\max}}} = m\frac{\Delta t}{2^k}$

5 - Décaler tous les points générant le maillage de $\delta t \boldsymbol{w}_i = \frac{\Delta t}{2^k_{\max}} \boldsymbol{w}_i$ et mettre à jour les variables conservatives de toutes les mailles

6 - Décomposer le domaine

7 - Entrer dans un calcul parallèle :

(i) Trouver les mailles fantôme

(ii) Construire l'arbre

(iii) Construire le maillage de Voronoï local

(iv) Raffiner et déraffiner le maillage

(v) Mettre à jour les variables intensives des mailles actives

8 - Mettre à jour les variables intensives et les données dépendant de la géométrie des mailles fantôme

9 - Entrer dans un calcul parallèle :

(i) Calculer les gradients et utiliser le limiteur de pente

(ii) Calculer la vitesse des points générateurs

C - 4 Calcul du pas de temps

Pour ce calcul, j'ai utilisé un algorithme d'arbre avec un oct-/quadtree : j'ai construit un arbre de la même forme que l'arbre stockant les données de mon problème à chaque nœud. Si c'est une feuille qui stocke une cellule, je stocke la vitesse du son, la norme de la vitesse de la particule et les données de la cellule. Si c'est une feuille qui stocke la cellule vide (la cellule vide est une cellule que j'ai définie), je stocke 0 pour la vitesse du son, 0 pour la vitesse et la cellule vide. Comme pour un nœud interne, je stocke la valeur maximale des vitesses du son stockées dans ses quatre (ou huit) enfants, la valeur maximale de la norme de vitesse des particules stockées dans ses quatre (ou huit) enfants et la cellule vide. J'appelle cet arbre T.

Pour calculer le pas de temps d'une maille en particulier j'ai fait comme suit : j'ai construit un algorithme récursif qui prend en entrée un nœuds N, une cellule C de point générateur $\boldsymbol{r}_i$ et un pas de temps δt, je l'appelle $f(C, N, \delta t)$. Un nœuds dans un oct-/quadtree est délimité par huit ou quatre sommets, grâce à ceux-ci je calcule la distance d entre le point générateur de C et le nœuds N.
Si N est une cellule $C' \neq C$ avec point générateur $\boldsymbol{r}_j$, je calcule $\boldsymbol{v}_{ij}^{\text{sig}}$, je retourne $\min(\frac{|\boldsymbol{r}_i - \boldsymbol{r}_j|}{\boldsymbol{v}_{ij}^{\text{sig}}}, \delta t)$.
Si N est un nœuds stockant la cellule vide je retourne δt.
Si N est la cellule C je retourne δt.
Sinon, N est un nœuds interne, j'appelle c la vitesse du son stockée et v la vitesse stockée

Si $\frac{d}{c+v} \geq \delta t$ je retourne δt

Sinon, j'écris N_l, $l \in \{1, 2, 3, 4\}$ ou $\{1, 2, 3, 4, 5, 6, 7, 8\}$, je retourne $\min_l(f(C, N_l, \delta t))$.

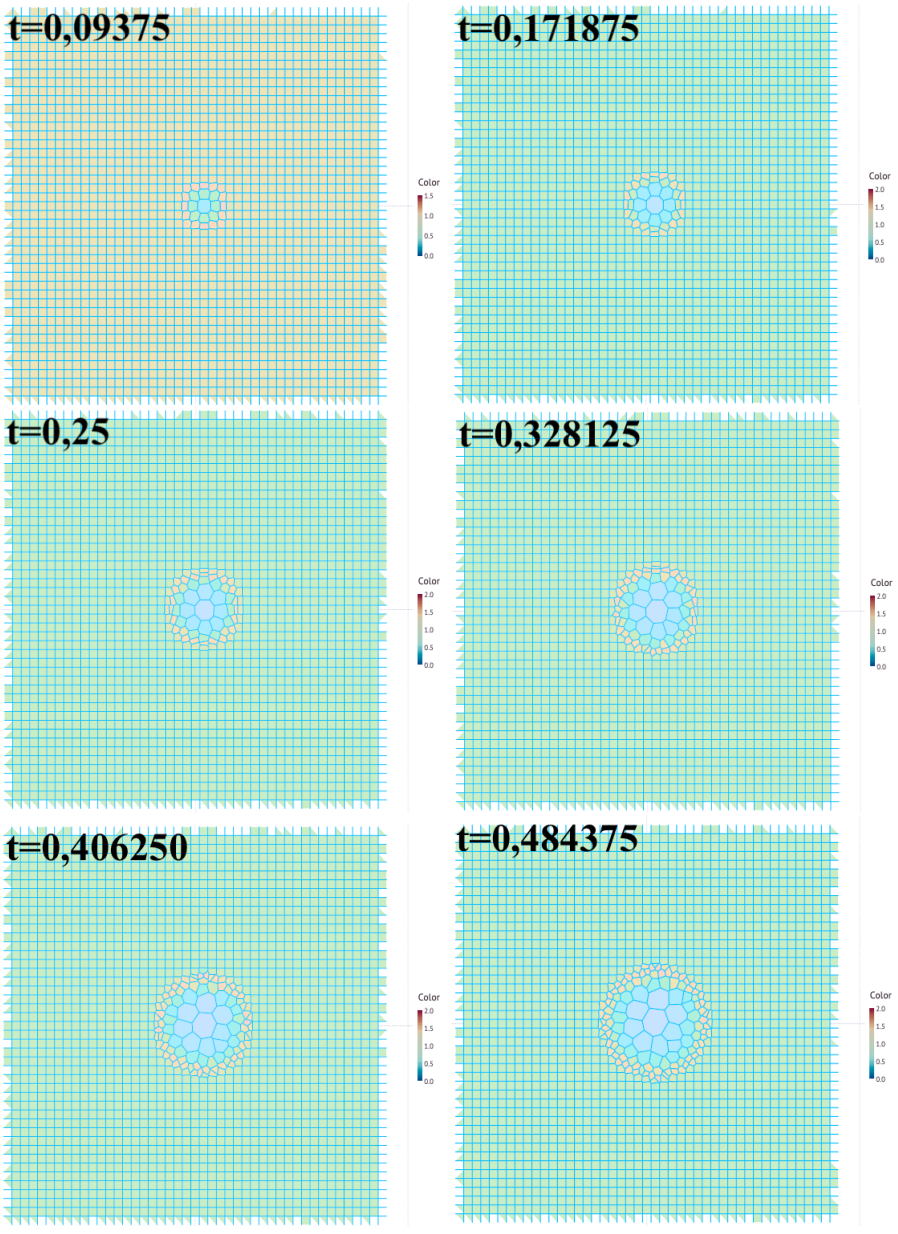
Pour chaque cellule C je définirai son pas de temps $\Delta t_i = f(C, T, \frac{R_i}{c_i + |\boldsymbol{v}_i - \boldsymbol{w}_i|})$ et je calculerai $k_i = \min(k, \log_2(\lfloor \frac{\Delta t}{\Delta t_i} \rfloor + 1))$ avec k comme en **B - 6.2** et je pose finalement $\Delta t_i = \frac{\Delta t}{2^{k_i}}$.
Cela assure $k_{\max} \leq k$ (tant qu'il n'y a pas de déraffinement).

D Test numérique

Le code est trouvable dans [14].
J'ai fait un test : l'explosion ponctuelle. En tant que première expérience je dois vérifier que mon code réponds comme en [1] pour le calcul le plus simple.
N.B. : Je n'ai pas pu éviter quatre erreur de négativité :

- La pression (resp. la masse) négative du centre de masse des cellules après un pas de temps lorsque je calcule les variables intensives à l'étape **VII/ 7-(v)** (resp. lorsque je calcule les variables conservatrices à l'étape **VII/ 5-**). Pour éviter que ces erreurs ne fassent planter l'algorithme (si elles ne sont pas significatives avec une valeur absolue $|P| < 10^{-4}$). J'ai brutalement remis la pression à zéro et recalculé les variables conservatrices correspondantes. Quant à la masse, j'ai utilisé un limiteur de flux pour éviter que la masse ne devienne négative.

- Les erreurs de pression et de masse lorsque je prévois la valeur des variables intensives plus loin dans le temps et l'espace au centre des faces (étape **VII/ 3-**). Ceci est dû au fait que pour certains gradients, même limités, la valeur au centre de l'interface est nulle pour la pression et la masse (ceci est dû à la construction du limiteur de pente), et elle peut devenir négative avec la prédiction temporelle. On peut voir sur la figure 3 qu'après une première partie symétrique de l'explosion ponctuelle, entre t=0,25 et t=0,328125 la simulation devient non symétrique (figure 4). Cette non-symétrie est probablement due à une faille dans mon code mais elle pourrait aussi être due à la capacité des CPU que j'ai utilisés pour la parallélisation. En effet, des erreurs de calcul pourraient se produire dans le solveur de Riemann, ce qui provoquerait cette perte de symétrie.


t=0,09375
t=0,171875
t=0,25
t=0,328125
t=0,406250
t=0,484375
Color
2.0
1.5
1.0
0.5
0.0

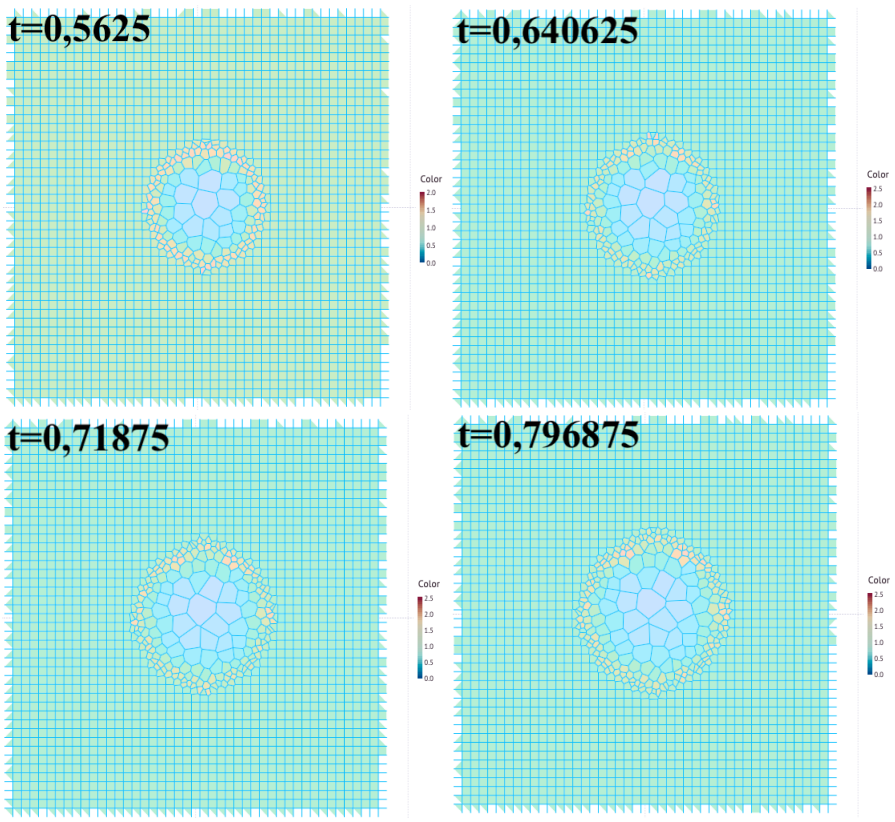


FIGURE 3 – Test d'explosion ponctuelle, $P = 1, \rho = 1$

E Conclusion

Dans ce mémoire, j'ai essayé de clarifier autant que possible les principaux aspects mathématiques de la méthode AREPO pour résoudre l'équation d'Euler. Pour ce faire, j'ai essayé de souligner les étapes qui mènent d'une méthode de base des volumes finis à maillage mobile lagrangien à quelque chose de plus complexe en termes de théorie mathématique, mais avec de forts gains en temps et en précision pour le calcul. Cette complexité est particulièrement présente dans l'utilisation de la géométrie du maillage pour calculer plus facilement le gradient et le problème de Riemann ainsi que dans l'introduction de l'échelonnement individuel du temps, ce qui représente un énorme gain de temps. En outre, les techniques d'ajustement des vitesses des points de génération du maillage, en particulier la résolution de l'équation de Poisson, font partie de cet objectif d'amélioration de la précision grâce à un travail mathématique apriori.
En fin de compte, l'implémentation que j'ai produite dans Julia est capable de donner des résultats, mais je pense qu'ils peuvent être grandement améliorés. En particulier, j'ai dû faire face à l'inévitable négativité des variables non négatives survenant systématiquement à un moment donné de mes calculs ainsi qu'à la non préservation de la symétrie, qui traduit des défauts sous-jacents dans mon code.
Pourtant, je pense avoir réussi à donner ici une application simple et claire à Julia ainsi que tout le potentiel pour généraliser le code à un plus grand nombre de lois de conservation.

F Bibliographie

[1] Springel,V. : E pur si muove : Galilean-invariant cosmological hydrodynamical simulations on a moving mesh, *Monthly Notices of the Royal Astronomical Society*, Volume 401, Issue 2,Pages 791–851, January 2010

[2] Chainet-Hillairet,C. : Finite volume schemes for a nonlinear hyperbolic equation. Convergence towards the entropy solution and error estimate, *ESAIM : M2AN* **33**, Pages 129-156, 1999

[3] Kröner,D.,Ohlberger,M. : A Posteriori Error estimates for upwind finite volume schemes for nonlinear conservation laws in multi dimensions, *Mathematics of Computation* **69**, Pages 25-39, 2000

[4] Vijayan,P.,Kallinderis,Y. : A 3D finite-volume scheme for the Euler equations on adaptive tetrahedral grids, *Journal of Computational Physics* 113(2), 249-267, 1994

[5] Torrilhon,M. : Krylov : Riemann solver for large hyperbolic systems of conservation laws, *SIAM Journal on Scientific Computing* **34(4)**, Pages 2072–A2091, 2012

[6] Schmidtmann,B.,Torrilhon,M. : A Hybrid Riemann Solver for Large Hyperbolic Systems of Conservation Laws, *SIAM Journal of Scientific Computing* **39(6)**, A2911–A2934, 2017

[7] Toro,E.,F. : MUSTA : A multi-stage numerical flux, *Applied Numerical Mathematics* **56**, pp. 1464– 1479, 2006

[8] Torrilhon,M. : Two-dimensional bulk microflow simulations based on regularized Grad's 13- moment equations, *Multiscale Modelling Simulation* **5**, pp. 695–728, 2006

[9] https ://github.com/astrowq/arepo

[10] https ://github.com/JuliaGeometry/GeometricalPredicates.jl

[11] https ://github.com/JuliaParallel/MPI.jl

[12] https ://github.com/JuliaGeometry/VoronoiDelaunay.jl

[13] https ://github.com/rdeits/RegionTrees.jl

[14] https ://github.com/EzRo7511/ArepoJulia