UCLouvain

École polytechnique de Louvain

# Simulation en temps réel de films visqueux fins

Auteur: **Antoine Quiriny**
Promoteurs: **Vincent Legat, Jean-François Remacle**
Lecteurs: **Philippe Chatelain, Jonathan Lambrechts**
Année académique 2020–2021
Master [120] : ingénieur civil mécanicien

# Résumé

L'étude des films visqueux fins est le sujet de nombreuses recherches scientifiques. Leurs équations sur surfaces planes et courbes ont depuis longtemps été établies et différentes méthodes permettent leur résolution. Mais ce n'est que très récemment, grâce au développement de la puissance de calcul des ordinateurs, qu'il est désormais envisageable de réaliser leur simulation en temps réel sur de simples *laptops* voire des *smartphones*.

Dans ce travail, nous développons les équations du film en se basant sur une série de perturbation. La méthode des volumes finis sur une grille cartésienne ainsi que la méthode de flux de gradient et les pas fractionnaires permettent d'obtenir un modèle explicite stable et local. Lorsqu'il est implémenté sur GPU, ce modèle révèle sa puissance grâce à son haut parallélisme. Cette efficacité permet des simulations interactives en temps réel à plus de 60 images par seconde.

Le développement de certains modèles s'écartant un peu de la physique réelle a permis d'obtenir des schémas bien plus stables tout en conservant les effets physiques qui caractérisent les films visqueux. De plus, l'interaction du fluide avec des obstacles et l'écoulement sur des surfaces non planes telles que des briques permettent des effets visuels supplémentaires. Une démonstration de ces simulations est disponible à cette adresse : https://www.youtube.com/watch?v=6n8uDTYJ684

# Remerciements

Je tiens à exprimer ma gratitude à toutes les personnes qui m'ont aidé lors de la rédaction de ce mémoire.

Je voudrais tout d'abord remercier mes deux promoteurs, les Professeurs Jean-François REMACLE et Vincent LEGAT, pour leur disponibilité, leur bienveillance et leur passion. Les nombreux échanges, hebdomadaires aux moments les plus cruciaux, et les multiples conseils et suggestions m'ont fortement stimulé et ont guidé ma réflexion tout au long de cette expérience.

J'adresse également mes remerciements à Jonathan Lambrechts d'avoir pris le temps de partager ses connaissances dans la programmation sur GPU.

Enfin un grand merci à mes parents et à Ophélie qui ont toujours été là pour moi. Leur soutien inconditionnel et leurs encouragements sans faille ont été d'une grande aide.

# Table des matières

# 1 Introduction

L'étude des films fins fait depuis longtemps l'objet de recherches scientifiques. Déjà en 1886, Osborne Reynolds publiait le premier article sur l'approximation de lubrification [1]. Cette approximation permet de réduire les équations de Navier-Stokes et la détection de l'interface libre en un modèle simplifié ayant comme variable la hauteur du film. Cette formulation se révèle pratique car, dans de nombreuses applications, nous sommes davantage intéressés par la hauteur du film que par la vitesse locale ou la pression de celui-ci. C'est le cas en optique lorsque des lentilles ou des cellules photovoltaïques sont recouvertes d'un film anti-réflection. Le champ d'application des films fins ne s'arrête pas là, on les retrouve en chimie dans le cas de couche anti-corrosion, en mécanique pour éviter la friction entre deux pièces, en électricité pour isoler un élément et même en biologie lorsqu'une larme se répand sur la cornée de l'oeil.

De nombreux articles traitent de la simulation numérique des films fins mais dans ce mémoire notre intérêt se porte sur leur faisabilité en temps réel. Nous nous sommes principalement basé sur le travail de Vantzos, Raz et Ben-Chen dans [2]. Les auteurs y présentent des résultats convaincants et atteignables par des appareils limités en puissance tels que des smartphones. Comme nous pouvons le voir à la figure 1, leur modèle permet de prendre en compte des obstacles et même l'aspect 3D comme l'écoulement sur des briques. Cet article est issu du journal *ACM Transaction on Graphics* et a été présenté à la conférence SIGGRAPH en 2019. Dans ce travail, la mesure de qualité des résultats obtenus ne se fait pas dans la métrique usuelle où nous cherchons à vérifier la convergence de nos équations et à valider nos résultats au moyen d'expériences mais bien visuellement. Dans cette optique, la notion de temps réel est également à relativiser. Nous désirons des simulations qui s'écoulent à une vitesse suffisante pour que visuellement cela semble être du temps réel. Pour que le lecteur puisse juger au mieux des résultats obtenus, une vidéo avec différentes simulations est disponible à cette adresse : https://youtu.be/6n8uDTYJ684

Les simulations actuelles de films fins ont besoin soit d'un pas de temps extrêmement petit pour être stable, soit d'utiliser des modèles implicites pour améliorer la stabilité. L'implémentation parallèle est l'une des clés des simulations en temps réel car elle permet de profiter pleinement de la puissance de calcul des GPU modernes qui surpassent de plus en plus les CPU. Or un modèle implicite nécessite de résoudre un système d'équation non-linéaire ce qui empêche une parallélisation efficace. Ce qui nous amène à nous poser la question centrale de ce travail : comment améliorer la stabilité de notre modèle tout en permettant une implémentation parallèle ?

Au fur et à mesure de nos recherches nous avons été confronté à d'autres questions :
Comment profiter de l'approximation de lubrification pour simplifier nos équations tout en gardant les effets qui nous intéressent : la gravité et la tension de surface ?
Une fois les équations établies, quelle méthode utiliser pour les résoudre ?
Comment améliorer le modèle et quelles sont les implications de ces modifications sur la physique ?
Comment optimiser l'implémentation de notre modèle sur CPU et sur GPU ?

Le film fin est traité dans nombre d'articles de la littérature scientifique. Oron et al. [3] donnent un bon aperçu des différents modèles mathématiques développés pour l'écoulement

FIGURE 1 – Écoulement de miel avec des obstacles hexagonaux et simulation de vin sur des briques

sur surface plane. Le développement d'équations pour l'écoulement de films sur des surfaces 3D a également été étudié avec un premier modèle en 1984 par Wang [4]. Un modèle de lubrification plus précis a été développé par Roy, Roberts et Simpson [5] sous la forme d'une équation aux dérivées partielles.

Pour résoudre numériquement les équations développées différentes méthodes ont été utilisées. [6], [5] et [7] utilisent des différences finies pour résoudre le problème. [8] ainsi que [9] ont quant à eux utilisé des éléments finis comme solutions. La méthode des volumes finis dans le cas des films fins a également été largement étudiée. Pour calculer le flux entre les différents volumes, il est alors possible d'utiliser la méthode de flux de gradient [10] ou bien de discrétiser l'expression du flux à l'aide de différences finies [5].

Dans ce travail, nous étudierons l'approximation de lubrification au moyen d'une série de perturbation et verrons comment l'ordre de grandeur des différents effets comme la gravité ou la tension de surface agissent sur les simulations. La technique des volumes finis et la méthode de flux de gradient pour le calcul du flux nous permettront de résoudre l'équation aux dérivées partielles développée. Les différents modèles obtenus au moyen de simplifications plus ou moins fortes des équations seront comparés et analysés. Enfin, nous nous attarderons sur l'implémentation du modèle sur CPU et GPU et la façon dont nous l'avons optimisée pour obtenir les simulations les plus efficaces possible.

# 2 De Navier-Stokes à l'équation classique du film

Dans cette partie nous allons tenter de passer des équations qui régissent l'écoulement d'un fluide newtonien incompressible à une équation pour le cas du film fin. Cette équation finale sera exprimée uniquement en fonction de $h$ la hauteur du film. Pour simplifier les expressions, nous nous limiterons au cas 1D plan qui pourra par la suite être généralisé. Le bilan de quantité de mouvement et l'équation de continuité pour un fluide incompressible donnent alors le système suivant :

$$\begin{cases} \rho\left(\partial_t u + u\partial_x u + w\partial_z u\right) = -\partial_x p + \mu(\partial_x^2 u + \partial_z^2 u) - \partial_x \phi \\ \rho\left(\partial_t w + u\partial_x w + w\partial_z w\right) = -\partial_z p + \mu(\partial_x^2 w + \partial_z^2 w) - \partial_z \phi \\ \partial_x u + \partial_z w = 0 \end{cases} \tag{1}$$

avec $x$ et $z$ les directions parralèles et perpendiculaires au support comme représenté à la figure 2. $u$ et $w$ sont respectivement la vitesse du fluide dans la direction $x$ et $z$, $p$ est la pression, $\rho$ la masse volumique du fluide et $\mu$ la viscosité dynamique. Remarquons que le potentiel gravitationnel $\phi$ est considéré de manière générale. Pour une surface verticale, la composante en z du gradient de $\phi$ est nulle.
Pour compléter ces équations, il faut considérer les deux conditions frontières de notre système. La première est à l'interface solide/liquide en $z = 0$ et la seconde est à l'interface liquide/air en $z = h(x,t)$. La condition classique que nous choisissons ici est de considérer qu'il n'y a ni pénétration ni glissement en $z = 0$. La condition à la seconde interface consiste en trois équations. La première est la condition limite cinématique, obtenue à partir de la conservation de la masse. La seconde correspond à l'équilibre des forces à l'interface dans la direction normale à cette dernière, nous y retrouvons le tenseur de contrainte du fluide mais également la tension de surface qui applique une force proportionnel au rayon de courbure de l'interface $\kappa$. La troisième impose quant à elle l'équilibre des forces dans la direction tangente à l'interface, dans notre cas cela implique uniquement le tenseur de contrainte du fluide. Cela s'exprime sous forme d'équation de la manière suivante :

$$\text{En } z = 0 : \quad w = 0, \quad u = 0 \tag{2}$$

$$\text{En } z = h(x,t) : \quad \begin{cases} w = \partial_t h + u\partial_x h \\ \begin{pmatrix} n_x & n_z \end{pmatrix} \begin{pmatrix} T_{xx} & T_{x,z} \\ T_{z,x} & T_{z,z} \end{pmatrix} \begin{pmatrix} n_x \\ n_z \end{pmatrix} = -\kappa\sigma \\ \begin{pmatrix} n_x & n_z \end{pmatrix} \begin{pmatrix} T_{xx} & T_{x,z} \\ T_{z,x} & T_{z,z} \end{pmatrix} \begin{pmatrix} t_x \\ t_z \end{pmatrix} = -\kappa\sigma \end{cases} \tag{3}$$

$T$ est le tenseur de contrainte du fluide, $\kappa$ le rayon de courbure de l'interface et $\sigma$ la constante de tension de surface. $n_x$ et $n_z$ sont respectivement les composantes en x et en z du vecteur unitaire normal sortant à l'interface et $t_x$ et $t_z$ les composantes tangentielles.

Cependant il faut faire attention au fait que sans glissement le film ne peut pas se propager sur une surface sèche. Il faut alors s'assurer qu'un léger film est présent sur tout le domaine lors de l'initialisation. Une autre façon de faire serait de permettre un glissement à l'interface sous la forme d'un modèle de Navier : la vitesse est proportionelle à la contrainte de cisaillement tel que $u - \beta\partial_z u = 0$ mais nous n'étudierons pas ce cas ici.

Pour arriver à simplifier nos équations nous considérons une approximation géométrique : la hauteur moyenne $h_0$ est petite par rapport à la longueur d'onde moyenne $\lambda$ des perturbations

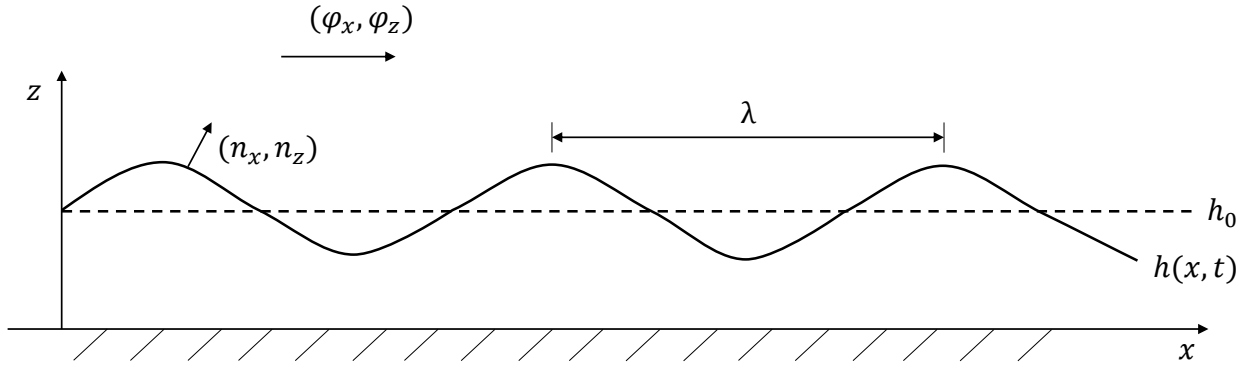



FIGURE 2 – Film fin

subies par le film. Ces deux grandeurs sont représentées à la figure 2. Définissons également le rapport de ces 2 valeurs $\epsilon$ tel que :

$$\epsilon := \frac{h_0}{\lambda} \ll 1$$

Cette hypothèse considère donc une surface libre relativement lisse et va nous permettre de développer nos équations comme une série de perturbation. Mais pour ce faire, nous devons dans un premier temps adimensionnaliser nos équations.

## 2.1 Adimensionnalisation des équations

Pour les variables spatiales, nous utilisons $h_0$ et $\lambda$ comme longueur caractéristique. Nous considérons la vitesse caractéristique $U_0$ dans la direction $x$ que nous définirons par la suite. Le temps caractéristique choisi correspond au temps pris par le film pour parcourir une distance $\lambda$ : $T_0 = \frac{\lambda}{U_0}$. Cela nous permet de définir les nombres adimensionnels suivants :

$$Z = \frac{z}{h_0} \qquad X = \frac{x}{\lambda} = \frac{\epsilon\, x}{h_0} \qquad U = \frac{u}{U_0} \qquad T = \frac{U_0 t}{\lambda} = \frac{\epsilon\, U_0 t}{h_0} \tag{4}$$

Il nous reste alors à adimensionnaliser la vitesse $w$, la pression $p$ et le potentiel de gravité $\phi$. Nous utilisons l'équation de continuité du système 1 $\partial_x u + \partial_z w$ pour obtenir la vitesse adimensionnelle dans la direction $z$. Nous considérons également un flux localement parallèle impliquant $\partial_x p \simeq \mu \partial_z^2 u$. Cela nous permet ainsi de définir la variable adimensionnelle correspondant à la pression. En insérant dans ces équations les variables adimensionnelles obtenues en 4, nous obtenons :

$$\begin{aligned} \partial_x u + \partial_z w &= 0 \\ \frac{U_0}{\lambda}\partial_X U + \frac{1}{h_0}\partial_Z w &= 0 \\ \partial_X U + \frac{1}{\epsilon\, U_0}\partial_Z w &= 0 \end{aligned}$$

$$\partial_x p = \mu \partial_z^2 u$$
$$\frac{\epsilon}{h_0}\partial_X p = \mu \frac{U_0}{h_0^2}\partial_Z^2 U$$
$$\frac{\epsilon\, h_0}{\mu U_0}\partial_X p = \partial_Z^2 U$$

Le potentiel de gravité joue le même rôle que la pression dans le système d'équation 1, ils ont donc les mêmes unités et peuvent être adimensionnalisés de la même façon. Cela nous donne les variables suivantes :

$$W = \frac{w}{\epsilon U_0} \qquad P = \frac{\epsilon\, h_0}{\mu U_0} p \qquad \Phi = \frac{\epsilon\, h_0}{\mu U_0}\phi \tag{5}$$

Maintenant que nous avons défini l'ensemble des variables adimensionnelles, nous pouvons les substituer dans le système d'équation 1. Nous décidons de laisser tomber les notations majuscules des variables adimensionnelles pour simplifier l'écriture.

$$\begin{cases} \epsilon\,\mathrm{Re}\left(\partial_t u + u\partial_x u + w\partial_z u\right) = & -\partial_x p + \partial_z^2 u + \epsilon^2\partial_x^2 u - \partial_x \phi \\ \epsilon^3\,\mathrm{Re}\left(\partial_t w + u\partial_x w + w\partial_z w\right) = & -\partial_z p + \epsilon^2\left(\partial_z^2 w + \epsilon^2\partial_x^2 w\right) - \partial_z \phi \\ \partial_x u + \partial_z w = & 0 \end{cases} \tag{6}$$

Nous pouvons également adimensionnaliser les conditions aux interfaces, ce qui donne :

$$\text{En } z = 0 : \qquad w = 0, \quad u = 0 \tag{7}$$

$$\text{En } z = h : \quad \begin{cases} w &= \partial_t h + u\partial_x h \\ 0 &= \left(\partial_z u + \epsilon^2\partial_x w\right)\left[1 - \epsilon^2\left(\partial_x h\right)^2\right] - 4\epsilon^2\left(\partial_x h\right)\left(\partial_x u\right) \\ \frac{C^{-1}\epsilon^3\partial_x^2 h}{\left[1+\epsilon^2(\partial_x h)^2\right]^{3/2}} &= -p + \frac{2\epsilon^2}{\left[1+\epsilon^2(\partial_x h)^2\right]}\left\{\partial_x u\left[\epsilon^2\left(\partial_x h\right)^2 - 1\right] - \partial_x h\left(\partial_z u + \epsilon^2\partial_x w\right)\right\} \end{cases} \tag{8}$$

Les développements de ces équations se trouvent en annexe. On voit apparaître dans ces expressions le nombre de Reynolds et le nombre Capillaire :

$$Re = \frac{U_0 h_0}{\nu}$$
$$C = \frac{U_0 \mu}{\sigma}$$

Le nombre de Reynolds représente le rapport des effets inertiels sur les effets visqueux tandis que le nombre Capillaire correspond aux effets visqueux divisés par les effets de tension de surface.
Avant de résoudre notre système 6, transformons l'équation de continuité de ce dernier dans une forme qui se révélera pratique par la suite. Nous pouvons l'intégrer selon $z$ sur la hauteur

du film et utiliser les conditions frontières :

$$\begin{aligned}
\partial_x u + \partial_z w &= 0 \\
\int_0^h \partial_x u dz + \int_0^h \partial_z w dz &= 0 \\
\int_0^h \partial_x u dz + w|_{z=h} - w|_{z=0} &= 0 \\
\overset{7,8}{\Longrightarrow} \quad \partial_t h + u\partial_x h + \int_0^h \partial_x u dz &= 0 \\
\overset{\text{Par partie}}{\Longrightarrow} \quad \partial_t h + \partial_x \left( \int_0^h u dz \right) &= 0
\end{aligned} \tag{9}$$

## 2.2 Série de perturbation et résolution

L'excellent cours *Mathematical Modelling of Physical Systems* [11] du Professeur Roland Keunings contient une très bonne introduction à la théorie des perturbations. L'idée principale est de considérer un problème complexe à résoudre comme étant un problème simple auquel nous ajoutons une perturbation. Par exemple, considérons le problème suivant :

$$x^5 + x = 1$$

Ce problème ne peut pas être résolu de manière exacte comme pour une équation quadratique. Par contre l'équation $x^5 = 1$ est très facile à résoudre et a comme solution exacte $x = 1$. Nous allons donc partir de ce problème simple et ajouter une perturbation pour arriver au problème initial :

$$x^5 + \epsilon x = 1 \tag{10}$$

Lorsque $\epsilon$ vaut 0 il s'agit de notre problème simple et lorsque $\epsilon$ vaut 1 cela correspond à l'équation que nous désirons résoudre.
La seconde étape est de considérer l'expression de la solution comme une somme de puissance en $\epsilon$ :

$$x(\epsilon) = \sum_{n=0}^{\infty} a_n \epsilon^n$$

Plus nous considérons de termes, plus la solution sera précise. Remarquons également que plus $\epsilon$ est petit et moins les termes avec des exposants élevés auront d'impact sur la solution.
Nous allons calculer les deux premiers termes de cette suite, les termes suivants peuvent être obtenus de la même manière. L'expression de x est alors $x(\epsilon) = a_0 + a_1\epsilon$ et si l'on substitue cette expression dans 10 cela donne :

$$\begin{aligned}
(a_0 + a_1\epsilon)^5 + \epsilon\,(a_0 + a_1\epsilon) &= 1 \\
(a_0 + 5a_1a_0\epsilon + ...) + \left(a_0\epsilon + a_1\epsilon^2\right) &= 1
\end{aligned}$$

Nous pouvons résoudre ce problème pour $a_0$ et $a_1$ en obtenant les coefficients du polynôme en $\epsilon$ par identification. Cela nous donne $a_0 = 1$ et $a_1 = -\frac{1}{5}$. Dans notre cas $\epsilon$ doit valoir 1 pour

correspondre au problème initial et donc la solution approchée vaut $x(\epsilon = 1) = a_0 + 1 \cdot a_1 = 0.8$. La solution en réalité vaut $\simeq 0.75488$, nous avons donc bien une solution approchée. Si nous avions calculé 4 termes au lieu de deux, la solution approchée aurait alors été de 0.754342 qui est encore plus proche de la vraie solution.

Nous pouvons résoudre notre problème de film fin en considérant les solutions des équations du système 6 comme une série de perturbation en puissance de $\epsilon$ :

$$\begin{aligned} u &= u_0 + \epsilon\, u_1 + \epsilon^2\, u_2 + \cdots \\ w &= w_0 + \epsilon\, w_1 + \epsilon^2\, w_2 + \cdots \\ p &= p_0 + \epsilon\, p_1 + \epsilon^2\, p_2 + \cdots \end{aligned}$$

Cette hypothèse va nous permettre de simplifier grandement les équations lorsque $\epsilon \to 0$ en ne considérant que les termes en $\mathcal{O}(\epsilon^0)$. Dans notre cas, nous considérons des valeurs d'$\epsilon$ très petites, il n'est donc pas nécessaire de conserver un grand nombre de termes. Cependant il reste le nombre de Reynolds $Re$ et le nombre capillaire $C$, il faut alors émettre une hypothèse sur leurs valeurs pour un $\epsilon$ tendant vers 0. Si l'on considère l'approximation de lubrification telle que $Re$ et $C = O(1)$ quand $\epsilon \to 0$, les effets inertiels $\epsilon Re$ sont bien un ordre inférieur aux effets visqueux mais la tension de surface est 3 ordres en dessous des effets visqueux. Pour garder les effets de la tension de surface on va considérer $\bar{C} = C\epsilon{-}3$, et la limite $Re$ et $\bar{C} = O(1)$ quand $\epsilon \to 0$. Ce choix n'est pas arbitraire, il permet d'être dans un régime dominé par les effets visqueux comme désiré. Nous pourrons par la suite changer ce choix pour modifier le comportement du fluide.

En considérant uniquement le terme en $\mathcal{O}(\epsilon^0)$, le système d'équations 6 et les conditions aux interfaces 7 et 8 donnent :

$$\begin{cases} \partial_z^2 u = \partial_x p + \partial_x \phi \\ 0 = \partial_z p + \partial_z \phi \\ \partial_t h + \partial_x \left( \int_0^h u dz \right) = 0 \end{cases} \tag{11}$$

En $z = 0$,

$$u = 0 \qquad\qquad w = 0$$

En $z = h$,

$$\begin{aligned} \partial_z u &= 0 \\ -p &= \bar{C}^{-1} \partial_x^2 h \end{aligned} \tag{12}$$

Nous pouvons observer que $p$ et $\phi$ agissent de la même façon dans les équations 11 . Nous allons donc introduire une pression augmentée :

$$\bar{p} = p + \phi$$

La seconde équation du système 11 nous indique que cette pression ne varie pas en fonction de $z$. En exprimant cette pression en $z = h$ grâce aux conditions frontières nous connaîtrons alors sa valeur en n'importe quel z. L'équation 12 permet d'écrire :

$$\bar{p} = \phi|_{z=h} - \bar{C}^{-1} \partial_x^2 h$$

Cette pression augmentée simplifie grandement la première équation de notre système 11. Nous pouvons l'intégrer 2 fois selon z et obtenir une expression pour $u$ : $u = \partial_x \bar{p} \frac{z^2}{2} + Az + B$ avec $A$ et $B$ des constantes d'intégration que nous pouvons obtenir grâce aux conditions frontières.

$$u = \partial_x \bar{p} \frac{z^2}{2} + Az + B$$
$$\partial_z u = \partial_x \bar{p} z + A$$

En $z = h$ : $\partial_z u = 0 \Longrightarrow A = -h\partial_x \bar{p}$

En $z = 0$ : $u = 0 \Longrightarrow B = 0$

La première équation du système 11 devient alors :

$$u = \partial_x \bar{p} \left( \frac{1}{2} z^2 - hz \right)$$

Nous pouvons alors substituer l'expression de $u$ dans l'équation 9 et calculer son intégrale de 0 à $h$ :

$$\partial_t h - \partial_x \left\{ \left( \frac{1}{3} h^3 \right) \partial_x \bar{p} \right\} = 0 \tag{13}$$

Avec $\bar{p} = \phi|_{z=h} - \bar{C}^{-1} \partial_x^2 h$. Nous avons là l'équation générale décrivant l'évolution de la hauteur du film.

## 2.3 Le nombre de Bond

Le nombre de Bond $\zeta$ correspond au rapport des effets de gravité sur les effet de tension de surface : $\zeta = \frac{\rho g L^2}{\sigma}$, où $L$ est la longueur caractéristique du problème. Ce nombre sera le principal paramètre de nos simulations et nous pouvons sauter quelques étapes pour directement observer son effet sur les simulations à la figure 3. Lorsque $\zeta$ vaut 2 (figure de gauche), le fluide est dominé par les effets de tensions de surface et résulte en un écoulement lent : du miel par exemple. Lorsque $\zeta$ prend des valeurs plus importantes (figure de droite), la gravité prend le dessus et l'écoulement est plus rapide.
L'étude de la tension de surface est un sujet très vaste permettant de modéliser notamment les films, les interfaces, les bulles. Nous ne pouvons que recommander le livre [12] qui couvre de manière globale la thématique et bien d'autres.

L'équation 13 peut être retransformée en équation dimensionnelle et nous donne :

$$\mu \partial_t h - \partial_x \left\{ \left( \frac{h^3}{3} \right) \partial_x \left[ \phi - \sigma \partial_x^2 h \right] \right\} = 0$$

Le potentiel de gravité est défini tel que $\phi = \rho g z$ avec z la hauteur, l'équation devient :

$$\mu \partial_t h - \rho g \partial_x \left( \frac{h^3}{3} \partial_x z \right) + \sigma \partial_x \left( \frac{h^3}{3} \partial_x^3 h \right) \tag{14}$$

Considérons la vitesse adimensionelle $U = \frac{u}{U_0} = u\frac{\mu}{\epsilon^3\sigma}$, le temps adimensionel devient alors $T = \frac{t}{T_0} = t\frac{\sigma\epsilon^3}{\mu\lambda}$. Cela nous permet d'écrire l'équation 14 sous forme adimensionnelle suivante :

$$\frac{\mu\sigma\epsilon^3 h_0}{\mu\lambda}\partial_t h - \frac{\rho g h_0^3}{\lambda}\partial_x\left(\frac{h^3}{3}\partial_x z\right) + \frac{\sigma h_0^4}{\lambda^4}\partial_x\left(\frac{h^3}{3}\partial_x^3 h\right) = 0$$
$$\sigma\epsilon^4\partial_t h - \rho g h_0^2\epsilon\partial_x\left(\frac{h^3}{3}\partial_x z\right) + \sigma\epsilon^4\partial_x\left(\frac{h^3}{3}\partial_x^3 h\right) = 0$$

où z a été adimensionnalisé avec $\lambda$.
Nous avons considéré $\bar{C} = C\epsilon^{-3}$ pour que la tension de surface soit de premier ordre. Nous considérons la même hypothèse pour le nombre de Bond $\zeta = \frac{\rho g h_0^2}{\sigma}$ tel que $\bar{\zeta} = \frac{\rho g h_0^2}{\sigma\epsilon^3} \to \mathcal{O}(1)$ quand $\epsilon \to 0$. L'équation devient alors :

$$\partial_t h - \bar{\zeta}\partial_x\left(\frac{h^3}{3}\partial_x z\right) + \partial_x\left(\frac{h^3}{3}\partial_x^3 h\right) = 0$$

Il est aussi possible de considérer $\bar{\zeta} = \frac{\zeta}{\epsilon^2}$ et $U_0 = \frac{\epsilon^2\sigma}{\mu}$, la tension de surface passe alors en $O(\epsilon)$ mais reste plus importante que les effets inertiels. L'équation devient alors :

$$\partial_t h - \bar{\zeta}\partial_x\left(\frac{h^3}{3}\partial_x z\right) + \epsilon\partial_x\left(\frac{h^3}{3}\partial_x^3 h\right) = 0 \tag{15}$$

Nous pourrions allez jusqu'à $\bar{\zeta} = \zeta$ et $U_0 = \frac{\sigma}{\mu}$. Cela donne comme équation :

$$\partial_t h - \bar{\zeta}\partial_x\left(\frac{h^3}{3}\partial_x z\right) + \epsilon^3\partial_x\left(\frac{h^3}{3}\partial_x^3 h\right) = 0 \tag{16}$$

La différence entre ces deux équations se trouve dans l'adimensionnalisation des variables. La vitesse dans la direction $x$ est plus grande de deux ordres pour l'équation 16 comparé à l'équation 15. C'est l'inverse pour le temps adimensionnel qui est deux ordre plus grand dans l'équation 15.
Pour que l'équation 16 soit complètement correcte il faudrait alors considérer les effets inertiels qui seraient de plus grande importance que la tension de surface ou alors faire l'hypothèse que $Re = \mathcal{O}(\epsilon^{-2})$ quand $\epsilon \to 0$.

## 2.4 Densité de masse et hauteur

Au lieu d'utiliser la variable h pour décrire l'évolution du film, nous pouvons introduire la distribution de masse adimensionnelle $u$ (nous n'utiliserons plus la vitesse par la suite, la notation u servira donc à désigner la distribution de masse). Nous relions cette variable à la hauteur en considérant que le volume total $\mathcal{X}_h$ du film se situant sur l'aire $U$ (voir figure 4) est égal à l'intégrale de la distribution de masse :

$$\int_{\mathcal{X}_h(U)} \mathrm{d}V = \epsilon\int_U u\ \mathrm{d}a$$

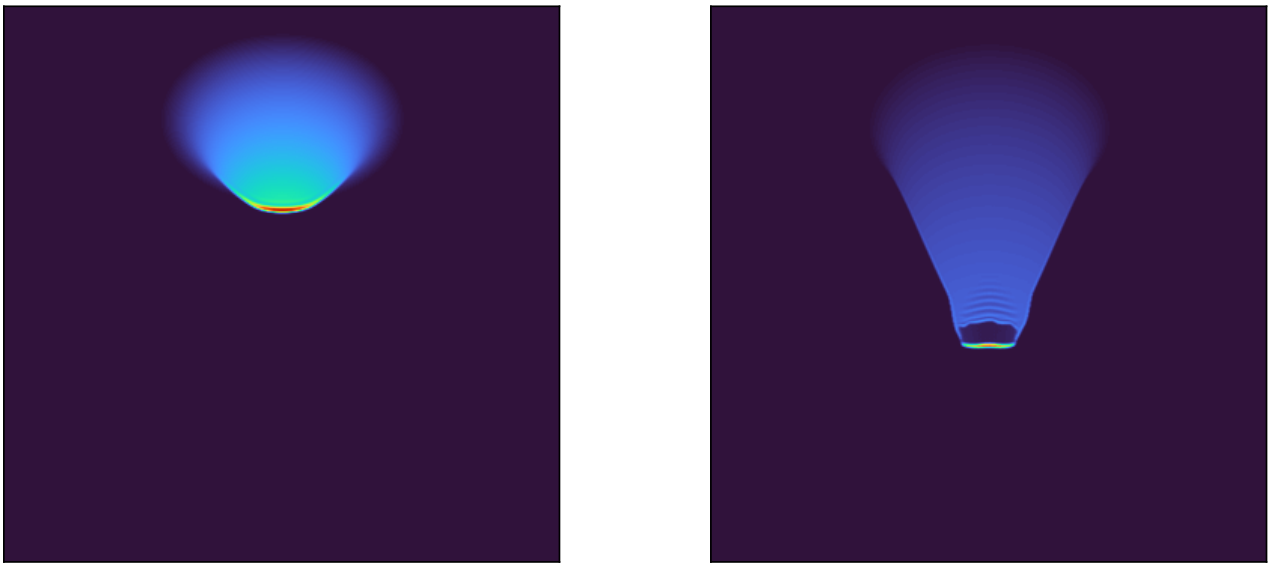


FIGURE 3 – Évolution d'une gaussienne pour $\zeta = 2$ et $\zeta = 10$

Pour le cas plan, il est assez évident que $u = h$, mais dans le cas d'une surface en 3D il faudra prendre en compte la courbure de celle-ci.
Définissons également $M(u) := \frac{1}{3}u^3$ et $f := \partial_x \bar{p}$, la mobilité et le flux (le flux réel est le produit de la mobilité et de $f$). L'équation dans sa forme la plus simple devient alors :

$$\partial_t u - \partial_x \{M(u) f\} = 0$$

Nous pouvons alors généraliser l'expression 15 en 3 dimensions et en fonction de $u$ :

$$\begin{aligned} \partial_t u - \nabla \cdot \left[ \frac{u^3}{3} \nabla \bar{p} \right] &= 0 \\ \partial_t u + \nabla \cdot \left[ M(u) \nabla \left( \zeta z - \epsilon \Delta u \right) \right] &= 0 \end{aligned} \tag{17}$$

Nous obtenons finalement l'équation décrivant l'évolution du film au moyen d'une seule variable $u$ représentant la quantité de matière au dessus de la surface solide. De plus comme nous l'avons montré dans cette section, il est possible d'adimensionnaliser de différentes façons notre équation. Ce choix a un impact sur le comportement du film et nous considérons pour la suite du développement du modèle la vitesse adimensionnelle $U_0 = \frac{\epsilon^2 \sigma}{\mu}$ et donc l'équation 17.

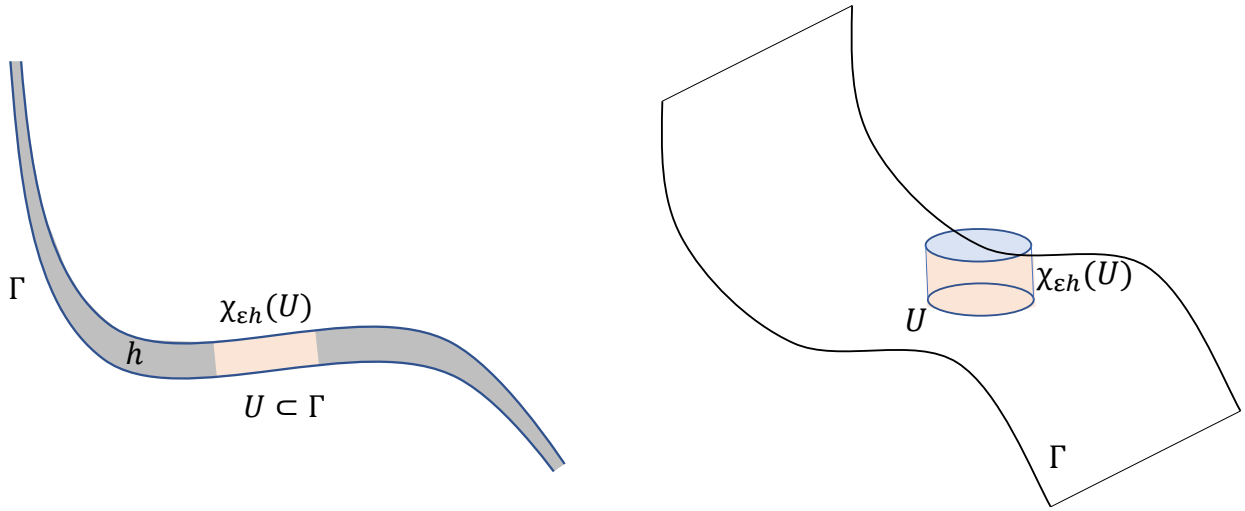



FIGURE 4 – Film en 2D et 3D

# 3 Modèle

## 3.1 Volumes finis et flux de gradient

L'équation 17 correspond à la conservation de $u$ la distribution de masse du film fin. Il est alors tentant d'utiliser la méthode des volumes finis pour la résoudre comme dans [13]. Cette méthode consiste à utiliser le théorème de la divergence pour passer de l'intégrale sur la surface de la divergence à l'intégrale sur les interfaces du flux. L'équation devient alors :

$$\begin{aligned} &\int_\Omega \partial_t u + \nabla \cdot (-M(u)\nabla(\zeta \boldsymbol{z} - \epsilon\Delta u)) = 0 \\ \iff &\int_\Omega \partial_t u + \int_{\partial\Omega} (-M(u)\nabla(\zeta \boldsymbol{z} - \epsilon\Delta u)) \cdot \boldsymbol{n} = 0 \end{aligned}$$

Posons $\hat{f} = (-M(u)\nabla(\zeta \boldsymbol{z} - \epsilon\Delta u)) \cdot \boldsymbol{n}$ le flux du fluide et considérons une grille cartésienne comme représenté à la figure 5.
Les valeurs u sont considérées comme étant au centre des cases tandis que le flux est à l'interface. Pour résoudre notre EDP il est donc nécessaire de calculer les différents flux $\hat{f}_i$ entre les cellules. Cela peut se faire en estimant les dérivées au moyen de différences finies comme dans [14] ou bien en considérant comme nous allons le faire un flux de gradient. Cela consiste à dire que le fluide se propage selon le gradient d'une fonction d'énergie. Cela s'exprime sous forme d'équation comme ceci :

$$\partial_t u + \partial_u E = 0 \tag{18}$$

Cette fonction d'énergie $E(u)$ doit alors prendre en compte les différents effets qui nous intéressent : la gravité et la tension de surface. Si l'énergie du film était uniquement due à la gravité, le fluide s'écoulerait dans la direction de plus grande pente de la surface sur laquelle il se trouve.
Cette façon de considérer le problème va nous permettre de transformer ce dernier en un problème de minimisation comme nous allons le voir maintenant dans le point discrétisation temporelle. Il s'agit de la discrétisation de notre équation 18.

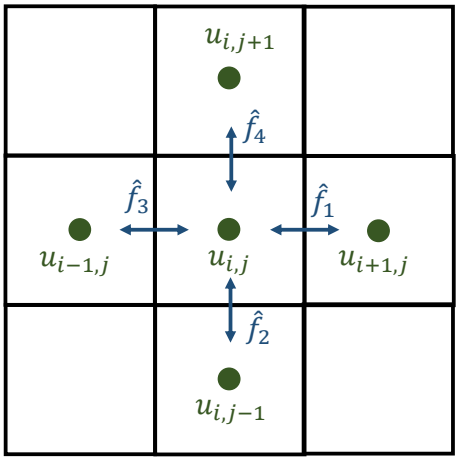



FIGURE 5 – Volumes finis sur grille cartésienne

## 3.2 Discrétisation temporelle

Pour discrétiser de manière temporelle l'équation 18, nous pouvons considérer sa discrétisation dite naturelle. Nous l'exprimons comme un problème de minimisation : Soit $u^k$ la distribution de masse à un temps $t^k$ et $u^{k+1}$ la distribution de masse à un temps $t^{k+1} = t^k + \tau$ :

$$u^{k+1} := \underset{u}{\operatorname{argmin}} \left\{ \frac{1}{2\tau} \operatorname{dist}^2 (u_k, u) + E[u] \right\} \tag{19}$$

où $\operatorname{dist}^2(\cdot, \cdot)$ est la distance au carré entre deux distributions de masse.
Pour prouver que ce problème correspond bien à un flux de gradient, considérons le cas où la distance choisie est la distance euclidienne classique $\operatorname{dist}^2 (u_k, u) = |u - u_k|^2$. En remplaçant cette expression de la distance dans 19 cela donne :

$$u^{k+1} := \underset{u}{\operatorname{argmin}} \left\{ \frac{1}{2\tau} |u - u_k|^2 + E(u) \right\}$$

Le minimum est atteint lorsque la dérivée par rapport à $u$ est nulle, ce qui nous donne comme équation :

$$\frac{u_{k+1} - u_k}{\tau} + \frac{\partial E(u_{k+1})}{\partial u} = 0$$

Cela correspond bien à la méthode d'Euler implicite appliquée à l'équation 18 et donc à un fluide se propageant selon le gradient d'une fonction énergie $E$.
Il nous reste alors 2 éléments à déterminer : que choisir comme distance et comme énergie ?
Pour l'énergie, nous considérerons uniquement la gravité et la tension de surface en concordance avec l'équation du film développée précédemment. Tandis que pour le carré de la distance entre 2 distributions de masse, nous utiliserons la dissipation visqueuse minimale requise pour passer de l'une à l'autre. Ce choix n'est pas anodin et nous pourrons le justifier par la suite en montrant que l'on obtient le même modèle via des différences finies.
Les deux sections suivantes permettent de déterminer l'énergie adimensionnelle du film ainsi que le carré de la distance adimensionnelle entre deux distributions de masse pour obtenir la discrétisation temporelle.

### 3.2.1 L'énergie du film

Nous allons utiliser l'expression de l'énergie adimensionnelle développée par Rumpf & Vantsoz [10] dans laquelle la gravité et la tension de surface sont prises en compte pour un film sur une surface courbe $\Gamma$ :

$$\begin{aligned}
\mathfrak{E}_{\text{gravité}} &= \int_\Gamma \zeta(zu + \frac{\epsilon}{2} \cos\theta u^2) \, \mathrm{d}a \quad + O(\epsilon^2) \\
\mathfrak{E}_{\text{surface}} &= \int_\Gamma -Hu - \frac{\epsilon}{2} T u^2 + \frac{\epsilon}{2} \left| \operatorname{grad}_\Gamma u \right|_\Gamma^2 \, \mathrm{d}a \quad + O(\epsilon^2)
\end{aligned}$$

En additionnant les deux nous obtenons notre expression de l'énergie :

$$E[u] = \int_\Gamma (\zeta z - H)u + \frac{\epsilon}{2}(\zeta \cos\theta - T)u^2 + \frac{\epsilon}{2} \left| \operatorname{grad}_\Gamma u \right|_\Gamma^2 \, \mathrm{d}a \quad + O(\epsilon^2)$$

Nous y retrouvons $u$ la distribution de masse, $\zeta$ le nombre de Bond, $z$ la hauteur, $\theta$ l'angle entre le vecteur normal à la surface et la verticale, $H$ est le rayon de courbure moyen et $T = H^2 - 2K$

avec $K$ le rayon de courbure gaussien. $|.|_\Gamma := \sqrt{\langle ., . \rangle_\Gamma}$ est la norme issue du produit interne $\langle ., . \rangle_\Gamma$ de la surface.
Nous pouvons également agir sur la stabilité de la solution directement dans l'expression de l'énergie. En effet, pour obtenir un film plus homogène et donc avec moins d'instabilité, nous pouvons introduire un terme anisotrope :

$$E[u] = \int_\Gamma (\zeta \boldsymbol{z} - H)u + \frac{\epsilon}{2}(\zeta \cos\theta - T)u^2 + \frac{\epsilon}{2} |\mathrm{grad}_\Gamma\, u|_\Gamma^2 + \frac{\eta}{2}u^2 \mathrm{d}a \tag{20}$$

Ce terme en $u^2$ associe une plus grande énergie aux zones de forte densité de fluide. Cette propriété a tendance à vouloir répartir équitablement le fluide sur la surface et donc stabiliser la solution.

Observons que dans le cas d'un plan vertical, les rayons de courbure sont nuls et que l'angle $\theta$ vaut 90°. Cela simplifie notre expression de l'énergie, qui devient alors :

$$E[u] = \int_\Gamma \zeta z u + \frac{\epsilon}{2} |\mathrm{grad}_\Gamma\, u|_\Gamma^2 + \frac{\eta}{2}u^2 \ \mathrm{d}a \quad + O(\epsilon^2) \tag{21}$$

Nous venons de développer l'expression de l'énergie exprimée en fonction de $u$. Il nous reste alors à obtenir la distance entre deux distributions et nous aurons tous les éléments nécessaires à la discrétisation temporelle.

### 3.2.2 Distance entre deux distributions

Pour que notre problème de minimisation soit complet, il nous faut déterminer une mesure de distance. Nous utiliserons pour ce faire, la dissipation visqueuse minimale pour passer d'une distribution de masse à l'autre. La dissipation visqueuse correspond à la variation d'une partie de l'énergie cinétique du fluide et prend la forme suivante :

$$g_u(v) = \mu \int_{\mathcal{X}_{\epsilon h}(\Gamma)} (\nabla v)^2 \ \mathrm{d}V = \frac{\mu}{2} \int_{\mathcal{X}_{\epsilon h}(\Gamma)} (\nabla v + \nabla v^T)^2 \ \mathrm{d}V$$

La seconde forme fait apparaître le tenseur de taux de déformation $D[v] = \frac{1}{2}\left(\nabla v + \nabla v^T\right)$. Pour la suite de notre développement nous désirons exprimer la dissipation visqueuse en fonction non pas de la vitesse mais du flux et ce pour une surface non-plane. Cela a été fait par Rumpf & Vantsoz dans [10] et ils obtiennent comme expression :

$$g_u(f) := \int_\Gamma \langle M(u)f, f\rangle_\Gamma \mathrm{d}a \tag{22}$$

Avec $M(u)$ la mobilité définie par :

$$M(u) := \frac{u^3}{3} + \frac{\epsilon}{6}u^4(H\ \mathrm{id}\ + S)$$

$S$ représente ici l'opérateur de surface et id est l'opérateur identité.
Nous considérons le carré de la distance entre 2 distributions de masse comme étant le minimum

de dissipation visqueuse nécessaire pour passer de l'une à l'autre. La distance au carré entre $u^k$ et $u^{k+1}$ devient donc :

$$\mathrm{dist}^2(u^k, u^{k+1}) = \tau \min_{f(t)} \left\{ \int_{t^k}^{t^{k+1}} g_{u(t)}(f(t))\mathrm{d}t \right\}$$

Le développement précis de la fonction d'énergie et de la distance se trouve en annexe. Le développement est long et correspond principalement à de la géométrie variationnelle qui n'a pas sa place dans le corps du texte. Nous l'avons tout de même détaillé pour le lecteur curieux. Maintenant que l'énergie et la distance sont toutes deux définies en fonction de $u$, nous pouvons les remplacer dans notre problème de minimisation originel 19.

### 3.2.3 Expression de la discrétisation temporelle

Les expressions de l'énergie et de la distance peuvent maintenant être substituées dans notre problème de minimisation 19. Cependant nous devons nous limiter dans le choix de $u$ et $f$ pour que ceux-ci soient compatibles avec l'équation de conservation de la masse. L'ensemble des paires $(u, f)$ au temps $t^{k+1}$ qui sont compatibles avec la distribution de masse $u^k$ au temps $t^k$ est désigné par $T_{t^k}^{t^{k+1}}\left[u^k\right]$.

$$u^{k+1} = \min_u \left\{ \frac{1}{2} \min_{(\tilde{u},\tilde{f}) \in T_{t^k}^{t^{k+1}}\left[u^k\right]} \int_{t^k}^{t^{k+1}} g_{\tilde{u}(t)}(\tilde{f}(t))\mathrm{d}t + E[u] \right\}. \tag{23}$$

$\tilde{u}$ et $\tilde{f}$ appartiennent à $T_{t^k}^{t^{k+1}}\left[u^k\right]$ si ils respectent l'équation suivante :

$$\frac{\tilde{u} - u^k}{\tau} + \mathrm{div}_\Gamma\left(M\left(u^k\right)\tilde{f}\right) = 0$$

Nous pouvons également utiliser un schéma de quadrature à un point pour obtenir la valeur de l'intégrale : $\int_{t^k}^{t^{k+1}} g_{u(t)}(f(t))\mathrm{d}t \simeq \tau g_{u(t)}(f(t))$. En intégrant ces 2 éléments dans l'équation 23, nous arrivons finalement à la discrétisation temporelle suivante :

$$\begin{array}{l} \min_{u,f} \left\{ \frac{\tau}{2} g_{u^k}(f) + E[u] \right\} \\ \text{Avec } u - u^k + \tau\, \mathrm{div}_\Gamma\left(M\left(u^k\right) f\right) = 0. \end{array} \tag{24}$$

En remplaçant les expressions de l'énergie $E[u]$ et de $g_{u^k}(f)$, cela nous donne :

$$\begin{array}{l} \min_{u,f} \left\{ \int_\Gamma \frac{\tau}{2} \langle M(u^k) f, f \rangle_\Gamma + (\zeta z - H)\, u + \frac{\epsilon}{2} \left(\zeta \cos\theta - T\right) u^2 + \frac{\epsilon}{2} \left| grad_\Gamma u \right|_\Gamma^2 + \frac{\eta}{2} u^2 da \right\} \\ \text{Avec } u - u^k + \tau\, \mathrm{div}_\Gamma\left(M\left(u^k\right) f\right) = 0. \end{array} \tag{25}$$

Nous venons donc d'obtenir la discrétisation temporelle de notre équation 18 pour le cas d'un film fin en considérant gravité et tension de surface. Celle-ci se présente sous la forme d'un problème d'optimisation sous contrainte. La discrétisation temporelle obtenue, nous pouvons nous intéresser à la discrétisation spatiale. Il s'agit de la dernière étape avant de pouvoir tout mettre ensemble et finalement avoir notre modèle.

## 3.3 Discrétisation spatiale

Pour la discrétisation spatiale, nous considérons une grille cartésienne et utilisons la méthode *Discrete Exterior Calculus* (DEC). La surface $\Gamma$ sur laquelle s'écoule le fluide est approximée par $N_C$ cellules $\{C_1, C_2, ..., C_{N_C}\}$ formant ensemble $\Gamma_C$. Chaque sommet du maillage $\Gamma_C$ se situe sur la surface $\Gamma$ et il existe une projection bijective $\Pi_\Gamma : \Gamma_C \to \Gamma$. La frontière $\partial C_i$ des cellules est composée de $N_E$ arrêtes $\{E_1, E_2, ..., E_{N_E}\}$. Les arrêtes sont orientées comme indiqué sur la figure 6 de telle sorte que $\pm E_j \in \partial C_{j\pm}$. Nous pouvons également définir l'opérateur matrice $\partial_C$ de taille $N_E \times N_C$ et tel que $(\partial_C)_{j,j\pm} = \pm 1$. Définissons également les opérateurs diagonaux $|C|$ et $|E|$ :

$$|C| := diag(|C_1|, |C_2|, ..., |C_{N_C}|)$$
$$|E| := diag(|E_1|, |E_2|, ..., |E_{N_E}|)$$

$|C_i|$ et $|E_i|$ étant respectivement l'aire de la cellule i et la longueur de l'arrête i. Dans notre cas, cela correspond pour $|C|$ à une matrice diagonale de $h^2$ et une matrice diagonale de $h$ pour $|E|$.

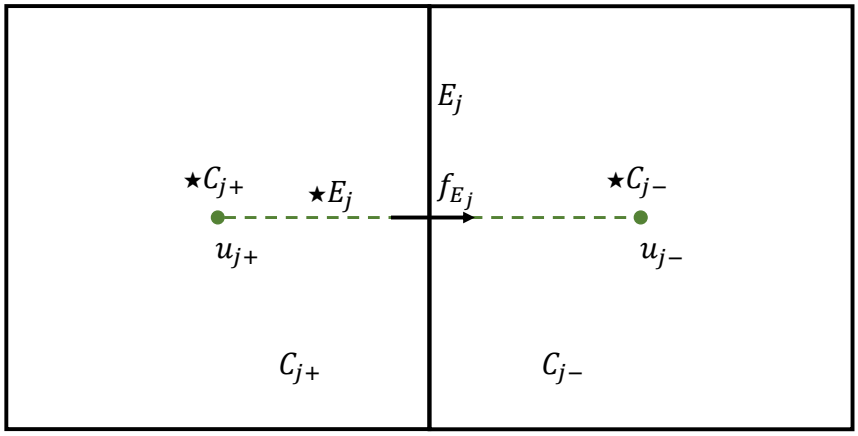



FIGURE 6 – Discrétisation spatiale avec *discrete exterior calculus*

Définissons également les *dual nodes* : chaque cellule comprend un noeuds dénoté $\star C_i \in C_i$ au centre de la case. Enfin les arrêtes *dual* $\star E_j$ connectent les *dual nodes* $\star C_{j-}$ et $\star C_{j+}$. Le même opérateur diagonal que pour les arrêtes classiques peut être défini :

$$|\star E| := diag(|\star E_1|, |\star E_2|, ..., |\star E_{N_E}|)$$

$|\star E_i|$ correspond à la distance entre $\star C_{j-}$ et $\star C_{j+}$ et vaut $h$ dans notre cas. $|\star E|$ est donc une matrice contenant des $h$ sur sa diagonale et des 0 partout ailleurs.

Les valeurs de $u$ dans les différentes cellules peuvent être représentées sous forme vectorielle de sorte que

$$u_C = \begin{bmatrix} u_{C_1} \\ u_{C_2} \\ ... \\ u_{C_{N_C}} \end{bmatrix}$$

Avec les $u_{C_i}$ définis comme la valeur moyenne de u sur $C_i$ :

$$u_{C_i} \simeq \left|\Pi_\Gamma(C_i)^{-1}\right| \int_{\Pi_\Gamma(C_i)} u da$$

Remarquons que $\partial_C u_C = (u_{C_{j+}} - u_{C_{j-}})$.
Nous pouvons représenter les quantités vectorielles $v$ sur le fibré tangent de la variété différentielle (*tangent bundle of the differentiable manifold*) comme les valeurs moyennes sur les arrêtes $E_j$ :

$$v_E = \begin{bmatrix} v_{E_1} \\ v_{E_2} \\ \dots \\ v_{E_{N_E}} \end{bmatrix}$$

$$v_{E_j} \simeq |\Pi_\Gamma(E_j)|^{-1} \int_{\Pi_\Gamma(E_j)} \langle v, \nu_j \rangle_\Gamma dl$$

Où $\nu_j$ est le vecteur conormal unitaire de l'arrête $E_j$ dans la direction de $C_{j+}$ et où $\langle ., . \rangle_\Gamma$ est le produit interne de $\Gamma$.
Le fibré tangent $T\Gamma$ de la variété différentielle $\Gamma$ est la réunion disjointe des espaces tangents de $\Gamma$. L'espace tangent de $\Gamma$ en $x$, noté $T\Gamma_x$, correspond à l'ensemble des vecteurs tangents à $\Gamma$ en ce point. Dans le fibré tangent $T\Gamma$ nous reprenons l'ensemble de ces vecteurs tangents sous la forme :

$$T\Gamma = \{(x, v) | x \in \Gamma, v \in T\Gamma_x\}$$

Les flux entre les volumes finis sont des quantités vectorielles que nous pouvons donc représenter sur le fibré tangent.

Nous pouvons maintenant introduire les opérateurs différentiels discrets gradient et divergence suivants :

$$\partial_{grad} = |\star E|^{-1} \partial_C \tag{26}$$

$$\partial_{div} = -|C|^{-1} \partial_C^T |E| \tag{27}$$

Il reste alors à définir les produits internes :

$$\langle u_C, \tilde{u}_C \rangle_C := u_C^T |C| \tilde{u}_C \simeq \int_\Gamma u \tilde{u} da \tag{28}$$

$$\langle v_E, \tilde{v}_E \rangle_E := v_E^T |\star E||E| \tilde{v}_E \simeq \int_\Gamma \langle v, \tilde{v} \rangle_\Gamma da \tag{29}$$

ainsi que les normes associées, respectivement $|u_C|_C$ et $|u_E|_E$.

En utilisant les opérateurs définis en 26 et 27 ainsi qu'en rappelant que les matrices $|C|$, $|E|$ et $|\star E|$ sont diagonales, on peut alors obtenir la propriété :

$$\begin{aligned} \partial_{div}^T |C| &= -|C|^{-1} \partial_C |E||C| \\ &= -|E| \partial_C \\ &= -|E||\star E||\star E|^{-1} \partial_C \\ &= -|E||\star E| \partial_{grad} \end{aligned}$$

De cette propriété nous pouvons déduire une analogie discrète à l'intégration par partie. En multipliant l'équation par $u_C$ et en prenant la transposée nous obtenons :

$$\begin{aligned}\left(\partial_{div}^T|C|u_C\right)^T &= -\left(|E|| \star E|\partial_{grad}u_C\right)^T \\ u_C^T|C|\partial_{div} &= -\left(\partial_{grad}u_C\right)^T | \star E||E| \\ \langle u_C, \partial_{div}v_E\rangle_C + \langle \partial_{grad}u_C, v_E\rangle_E &= 0\end{aligned} \tag{30}$$

### 3.3.1 Énergie discrète

Rappelons l'expression de l'énergie obtenue précédemment en 20 :

$$E[u] = \int_\Gamma (\zeta z - H)u + \frac{\epsilon}{2}(\zeta\cos\theta - T)u^2 + \frac{\epsilon}{2}\left|\mathrm{grad}_\Gamma\, u\right|_\Gamma^2 + \frac{\eta}{2}u^2\ \mathrm{d}a$$

Nous pouvons alors remplacer $\mathrm{grad}_\Gamma$ par son équivalent discret $\partial_{grad}$ et utiliser les produits internes définis en 28 et 29 pour approximer les intégrales et exprimer l'énergie sous la forme :

$$\boldsymbol{\mathcal{E}}_h\left(u_C\right) := \langle u_C, \zeta z_C - H_C\rangle_C + \frac{\epsilon}{2}\left\langle u_C, \left[\zeta\cos\theta_C - T_C\right]u_C\right\rangle_C + \frac{\epsilon}{2}\left|\partial_{\mathrm{grad}}u_C\right|_E^2 + \frac{\eta}{2}\left\langle u_C, u_C\right\rangle_C \tag{31}$$

La propriété 30 appliquée à la quantité vectorielle $\partial_{grad}u_C$ donne $\langle u_C, \partial_{div}\partial_{grad}u_C\rangle_C = |\partial_{grad}u_C|_E^2$. Nous pouvons la remplacer dans l'équation 31 et développer les produits internes selon 28 et 29, ce qui nous donne :

$$\boldsymbol{\mathcal{E}}_h\left(u_C\right) := u_C^T|C|\left(\zeta z_C - H_C\right) + \frac{\epsilon}{2}u_C^T|C|\left(\zeta\cos\theta_C - T_C\right)u_C + \frac{\epsilon}{2}\left(u_C^T|C|\partial_{div}\partial_{grad}u_C\right) + \frac{\eta}{2}u_C^T|C|u_C$$

Pour une grille cartésienne, $|C|$ est une matrice diagonale de $h^2$. L'énergie associée à une cellule $p$ vaut donc :

$$\mathcal{E}_h\left(u_p\right) = u_p h^2\left(\zeta z_p - H_p\right) + \frac{\epsilon h^2 u_p^2}{2}\left(\zeta\cos\theta_p - T_p\right) + \frac{\epsilon h^2}{2}\left(u_p\Delta u_p\right) + \frac{\eta h^2}{2}u_p^2 \tag{32}$$

où $\Delta$ est l'opérateur laplacien.
Nous venons d'obtenir l'expression discrète de l'énergie que nous pourrons remplacer dans le problème de minimisation sous contrainte 19.

### 3.3.2 Distance et mobilité discrète

Il nous reste alors à déterminer une expression discrète pour la distance et la mobilité $M(u)$ pour que notre problème de minimisation soit entièrement discrétisé. Pour cela, nous devons obtenir l'expression discrète de la fonction $g_u(f)$.

Grâce au produit interne 29 nous pouvons approximer l'équation 22 par :

$$\begin{aligned}g_u(f) &= \int_\Gamma \langle f_E, M(u)f_E\rangle_\Gamma \mathrm{d}a \\ &\simeq \langle f_E, \boldsymbol{M}_E(u)f_E\rangle_E\end{aligned} \tag{33}$$

où $\boldsymbol{M}_E(u)$ est une matrice de taille $N_E \times N_E$ qui contient les valeurs de la mobilité sur l'ensemble des arrêtes sur chacune de ses lignes.

La valeur de la mobilité pour l'arrête $j$ est nommée $M_E(u_C)_j$ et suivant [14] correspond à l'intégrale harmonique :

$$M_E\left(u_C\right)_j := \left(\frac{1}{u_{C_{j+}} - u_{C_{j-}}} \int_{u_{C_{j-}}}^{u_{C_{j+}}} M_{E_j}(t)^{-1} \mathrm{d}t\right)^{-1} \tag{34}$$

où $M_{E_j}(u) := \frac{1}{3}u^3 + \frac{\epsilon}{6}u^4\left(H_{E_j} + \kappa_{E_j}\right)$, $H_{E_j}$ le rayon de courbure moyen au centre de l'arrête et $\kappa_{E_j}$ la courbure normale dans la direction perpendiculaire à l'arrête.
En calculant l'intégrale et en ne gardant que les termes en $\mathcal{O}(\epsilon)$ ou plus grand, l'équation 34 devient :

$$\begin{aligned} M_E\left(u_C\right)_j &= \left(\frac{18}{u_{C_{j+}} - u_{C_{j-}}} \int_{u_{C_{j-}}}^{u_{C_{j+}}} \frac{1}{6t^3 + 3\epsilon t^4\left(H_{E_j} + \kappa_{E_j}\right)} \mathrm{d}t\right)^{-1} \\ &= \left(\frac{3}{2(u_{C_{j+}} - u_{C_{j-}})}\left(\frac{\epsilon\left(H_{E_j} + \kappa_{E_j}\right)u_{C_{j+}} - 1}{u_{C_{j+}}^2} - \frac{\epsilon\left(H_{E_j} + \kappa_{E_j}\right)u_{C_{j-}} - 1}{u_{C_{j-}}^2}\right)\right)^{-1} \\ &= \frac{\frac{2}{3}\left(u_{C_{j+}} - u_{C_{j-}}\right)u_{C_{j+}}^2 u_{C_{j-}}^2}{\epsilon\left(H_{E_j} + \kappa_{E_j}\right)\left(u_{C_{j+}}u_{C_{j-}}^2 - u_{C_{j+}}^2 u_{C_{j-}}\right) + \left(u_{C_{j+}}^2 - u_{C_{j-}}^2\right)} \end{aligned} \tag{35}$$

Dans le cas plan, $H_{E_j}$ et $\kappa_{E_j}$ sont tous les deux nuls ce qui nous donne $M_E(u_C)_j = \frac{2u_{C_{j+}}^2 u_{C_{j-}}^2}{3\left(u_{C_{j+}} + u_{C_{j-}}\right)}$.
Lorsque $u_{C_{j+}} = u_{C_{j-}} = u$ nous obtenons bien $M_E(u) = \frac{u^3}{3}$

## 3.4 Un modèle local

Dans les chapitres précédents nous avons développé et discrétisé les différents éléments nécessaires à notre modèle. Avant de tout rassembler, il reste un point à discuter : le caractère explicite ou implicite de nos équations.

La majorité des articles traitant de l'étude des films fins utilisent une méthode implicite. La principale raison à cela vient de la plus grande stabilité du schéma. L'utilisation d'une méthode implicite nécessite de résoudre un système non-linéaire à chaque pas de temps. La résolution de système non-linéaire n'est pas très adaptée à la parallélisation ce qui pose problème dans notre recherche de simulation en temps réel.
Pour pallier cet inconvénient, nous allons utiliser une méthode à pas fractionnaires.

### 3.4.1 Méthode à pas fractionnaires

[15] et [16] décrivent et analysent la méthode à pas fractionnaires qui consiste à considérer que le flux total $f$ est une combinaison linéaire de flux locaux :

$$\hat{f} = \sum_{k=1}^{K} \hat{f}^k$$

Définissons $T_{\hat{f}^i}$, l'opérateur qui met à jour les valeurs des densités de masses $u$ avec le flux $\hat{f}^i$. La méthode des pas fractionnaires nous permet d'approximer l'application du flux total aux

densités de masses par l'application successive des flux locaux :

$$T_{\hat{f}}u \simeq T_{\hat{f}^K} \cdot T_{\hat{f}^{K-1}} ... T_{\hat{f}^1} u$$

Les valeurs des flux locaux permettent de mettre à jour les densités de masse $u$ concernées et c'est avec les nouvelles valeurs de $u$ que nous calculerons les flux suivants. Cette méthode se rapproche de Gauss-Seidel : nous espérons une meilleure convergence en utilisant des solutions intermédiaires.

L'opérateur $T_{\hat{f}^i}$ correspond donc à la résolution de cette équation :

$$\int_\Omega \partial_t u + \int_{\partial\Omega} \hat{f}^i = 0$$

Dans notre cas nous considérons une quadrature de Gauss à un point. Cela nous arrange puisque $u$ est positionné au centre de la cellule et $\hat{f}^i$ au centre de l'interface comme représenté à la figure 5. L'équation devient :

$$\begin{aligned}
\int_\Omega \partial_t u + \int_{\partial\Omega} \hat{f}^i &= 0 \\
\int_\Omega \frac{u^{k+1} - u^k}{\tau} + \int_{\partial\Omega} \hat{f}^i &= 0 \\
\frac{u^{k+1} - u^k}{\tau} h^2 + \hat{f}^i h &= 0 \\
u^{k+1} &= u^k - \frac{\tau}{h} \hat{f}^i
\end{aligned}$$

Où $u^k$ est la valeur de $u$ au temps $k$ et $\tau$ est le pas de temps $\Delta t$.
Le calcul d'un seul pas de temps peut alors être défini par ces équations :

$$\begin{aligned}
u^{(0)} &= u^n \\
u^{(k)} &= u^{(k-1)} - \frac{\tau}{h} \hat{f}^{(k)} \\
u^{n+1} &= u^K
\end{aligned} \tag{36}$$

Nous pouvons alors utiliser l'équation 36 pour utiliser les valeurs $u^{(k)}$ dans le calcul de $\hat{f}^{(k)}$ et ainsi obtenir un modèle implicite. La méthode des pas fractionnaires nous a ainsi permis d'obtenir une méthode "implicite" sans devoir résoudre de système.

### 3.4.2 Modèle Final

Nous pouvons maintenant remplacer nos expressions discrètes de l'énergie et de la distance dans notre discrétisation temporelle 24. Nous désirons calculer le flux entre deux cellules $p$ et $q$ de notre grille cartésienne. L'expression de l'énergie discrétisée 32 donne l'énergie associée à une cellule. Nous devons donc additionner l'énergie associée aux cellules $p$ et $q$ ensemble :

$$\mathcal{E}_h(u_p, u_q) = \frac{h^2\epsilon}{2} \sum_{i=p,q} u_i \Delta u_i + h^2 \sum_{i=p,q} (\zeta z_i - H_i) u_i + \frac{\eta h^2}{2} \sum_{i=p,q} u_i^2 + \frac{\epsilon h^2}{2} \sum_{i=p,q} (\zeta \cos\theta_i - T_i) u_i^2$$

Dans cette expression, nous considérons l'opérateur classique laplacien discret à 5 points : $\Delta = \frac{1}{h^2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$ L'expression de la fonction de distance $g_u(f)$ 33 dans le cas d'une grille cartésienne et pour le flux entre les cases $p$ et $q$ peut alors être obtenue. Rappelons également que $\hat{f} = M(u)f$ où $\hat{f}$ est le flux réel.

$$\begin{aligned} g_{u_p,u_q}(f) &= h^2 f^2 M(u_p, u_q) \\ &= h^2 \frac{\hat{f}^2}{M(u_p, u_q)} \end{aligned}$$

où $M(u_p, u_q)$ correspond à $M_E(u_C)_j$ en 35 pour $u_{C_{j+}}, u_{C_{j-}} = u_p, u_q$.
Nous pouvons maintenant utiliser la méthode des pas fractionnaires. Nous allons utiliser les valeurs $\tilde{u}_p$ et $\tilde{u}_q$ les densités des cellules $p$ et $q$ après l'application du flux partiel grâce à 36 et ainsi obtenir un modèle "implicite". Cependant la mobilité restera explicite et nous utiliserons les valeurs de $u_p$ et $u_q$. Remarquons que les facteurs $h^2$ peuvent être mis en évidence et n'interviennent donc pas dans le problème de minimisation. Cela nous donne finalement :

$$\begin{aligned} \min_{\hat{f}\in\mathbb{R}} \Bigg\{ &\frac{\tau|\hat{f}|^2}{2M(u_p,u_q)} + \frac{\epsilon}{2}\tilde{u}_p\Delta\tilde{u}_p + \tilde{u}_q\Delta\tilde{u}_q + ((\zeta \boldsymbol{z}_p - H_p)\tilde{u}_p + (\zeta \boldsymbol{z}_q - H_q)\tilde{u}_q) + \frac{\eta}{2}\left(|\tilde{u}_p|^2 + |\tilde{u}_q|^2\right) \\ &+ \frac{\epsilon}{2}\left((\zeta\cos\theta_p - T_p)\tilde{u}_p^2 + (\zeta\cos\theta_q - T_q)\tilde{u}_q^2\right)\Bigg\} \end{aligned}$$

$$\begin{aligned} \tilde{u}_p &= u_p - \frac{\tau}{h}\hat{f}, \quad \tilde{u}_p \geq 0 \\ \tilde{u}_q &= u_q + \frac{\tau}{h}\hat{f}, \quad \tilde{u}_q \geq 0 \end{aligned}$$

En remplaçant les expressions de $\tilde{u}_p$ et $\tilde{u}_q$ on obtient alors une expression quadratique en $\hat{f}$ de type :

$$\min_{\hat{f}\in\mathbb{R}} \left\{ \frac{1}{2}\alpha\hat{f}^2 - \beta\hat{f} + \gamma \right\}$$

Or cette expression a un minimum unique en $\hat{f} = \frac{\beta}{\alpha}$. On trouve alors comme expression pour $\alpha$ et $\beta$ :

$$\begin{aligned} \alpha =& \frac{\tau}{M} + \frac{\tau^2}{h^2}\left(\frac{10\epsilon}{h^2} + \epsilon\zeta(\cos\theta_p + \cos\theta_q) - \epsilon(T_p + T_q)\right) \\ \beta =& \epsilon\frac{\tau}{h}(\Delta u_p - \Delta u_q) + \zeta\frac{\tau}{h}\left[(\boldsymbol{z}_p - H_p)u_p - (\boldsymbol{z}_q - H_q)u_q\right] + \epsilon\frac{\tau}{h}\left[(\zeta\cos\theta_p - T_p)u_p - (\zeta\cos\theta_q - T_q)u_q\right] \\ &+ \eta\frac{\tau}{h}(u_p - u_q) \end{aligned}$$

L'expression pour le flux est donc :

$$\begin{aligned} \hat{f} =& \frac{Mh}{K}\{\epsilon(\Delta u_p - \Delta u_q) + \zeta\left[(\boldsymbol{z}_p - H_p) - (\boldsymbol{z}_q - H_q)\right] \\ &+ \epsilon\left[(\zeta\cos\theta_p - T_p)u_p - (\zeta\cos\theta_q - T_q)u_q\right] + \eta(u_p - u_q)\} \end{aligned} \tag{37}$$

avec $K = h^2 + \tau M(\epsilon(\frac{10}{h^2} + \zeta(\cos\theta_p + \cos\theta_q) - T_p - T_q) + 2\eta)$
Dans le cas plan, cette expression se simplifie :

$$\hat{f} = \frac{Mh}{K}\left\{\epsilon(\Delta u_p - \Delta u_q) + \zeta\left[\boldsymbol{z}_p - \boldsymbol{z}_q\right] + \eta(u_p - u_q)\right\} \tag{38}$$

avec $K = h^2 + 2\tau M(\frac{5}{h^2}\epsilon + \eta)$

## 3.5 Différences finies et volume fini

Pour vérifier nos calculs, il est possible de calculer le flux de nos volumes finis directement en utilisant les différences finies pour approximer les dérivées au lieu d'utiliser un flux de gradient. L'expression du flux tiré de l'équation 17 du film fin en 2D à laquelle nous rajoutons un terme anisotropique est :

$$\begin{aligned} \partial_t u + \nabla \cdot \hat{f} &= 0 \\ \hat{f} &= M(u)\nabla\left(-\zeta z - \epsilon\Delta u + \eta u\right) \end{aligned} \tag{39}$$

Tout comme dans le modèle des auteurs nous allons considérer une méthode à pas fractionnaires, une mobilité complètement explicite et une quadrature d'intégration à un point telles que :

$$\begin{aligned} \tilde{u}_p &= u_p - \frac{\tau}{h}\hat{f} \\ \tilde{u}_q &= u_q + \frac{\tau}{h}\hat{f} \end{aligned}$$

où $\hat{f}$ est le flux entre la case $p$ et $q$.
Il ne reste plus qu'à déterminer comment nous allons approximer les dérivées. Nous choisissons les différences finies centrée d'ordre deux de sorte que l'équation 39 devienne :

$$\begin{aligned} \hat{f} &= -\frac{M(u_p,u_q)}{h}\left[W_q - W_p - \epsilon\left(\Delta u_q - \Delta u_p - \frac{10\tau}{h^3}\hat{f}\right) + \eta(u_q - u_p + 2\frac{\tau}{h}\hat{f})\right] \\ \hat{f}\left(1 + \frac{\tau M(u_p,u_q)}{h^2}(\frac{10\epsilon}{h^2} + 2\eta)\right) &= \frac{-M(u_p,u_q)}{h}\left[W_q - W_p - \epsilon(\Delta u_q - \Delta u_p) + \eta(u_q - u_p)\right] \\ \hat{f} &= \frac{M(u_p,u_q)h}{K}\left[W_q - W_p - \epsilon(\Delta u_q - \Delta u_p) + \eta(u_q - u_p)\right] \end{aligned}$$

Avec $K = h^2 + 2\tau M(\frac{5}{h^2}\epsilon + \eta)$, il s'agit bien là du modèle 38 développé en flux de gradient.

## 3.6 Discussion du modèle

Le modèle développé jusqu'à présent et obtenu en 37 correspond au modèle décrit dans [2] par O. Vantzos, S. Raz et M. Ben-Chen. L'objectif de ce modèle était d'obtenir des simulations en temps réel de films visqueux fins. Comme expliqué précédemment dans la section *Un modèle local*, l'idée est de conserver la possibilité de paralléliser sur GPU mais d'améliorer la stabilité en rendant le modèle "implicite" grâce à la méthode des pas fractionnaires.

**Modèle développé**
Malheureusement lorsque nous implémentons ce modèle sur une grille de 512 par 512 cellules, nous observons une stabilité pour un pas de temps de maximum $\tau = 9 \cdot 10^{-9}$ pour une initialisation gaussienne. Il s'agit là d'un résultat décevant comparé aux pas de temps de l'ordre de $10^{-4}$ pour des modèles implicites. L'écoulement d'une gaussienne sur une surface plane simulé à l'aide de ce modèle est représenté à la figure 7. L'écoulement est dans ce cas très lent et complètement dominé par les effets de tension de surface. La simulation jusqu'à $T = 0.5$ prend plus de 4h et ne permet pas du tout des simulations qui semble visuellement être en temps réel. Cette stabilité et ces résultats ne correspondent pas à ceux de l'article [2] puisqu'ils obtenaient

une solution stable pour un pas de temps de $\tau = 10^{-2}...10^{-3}$. Leurs résultats directement tirés de l'article sont affichés à la Figure 8.

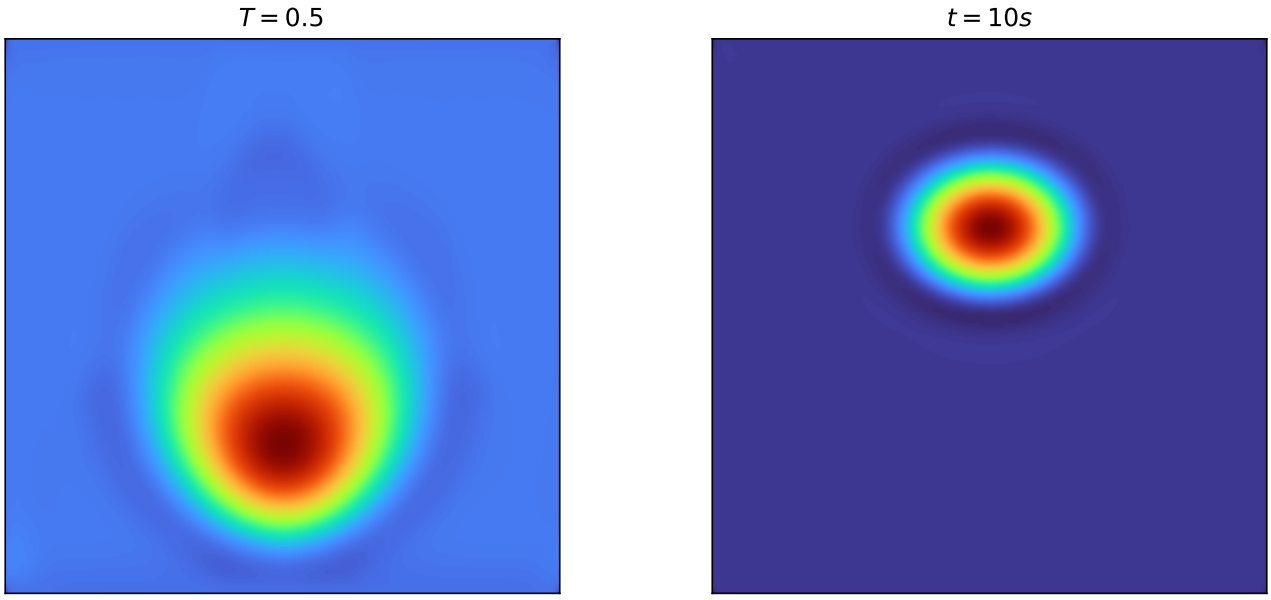



FIGURE 7 – Simulation d'une gaussienne pour le modèle développé 38

**Modèle implémenté**

Les auteurs de [2] ont donné accès à leur implémentation ce qui a permis de comprendre les différences entre les résultats obtenus à l'aide de leur modèle et ceux présentés dans l'article. Le modèle implémenté ne correspond pas à l'article, il s'agit de l'équation suivante :

$$\hat{f} = -\frac{M\left(u_p, u_q\right)}{\theta}\left\{W_q - W_p - \epsilon\left(\left(\Delta_{h^2} u\right)_q - \left(\Delta_{h^2} u\right)_p\right) + \eta\left(u_q - u_p\right)\right\} \tag{40}$$

Avec $\theta := 1 + 2\tau M\left(u_p, u_q\right)(5\epsilon + \eta)$, $W_q = G \cdot z/h$ et où $\Delta_{h^2}$ correspond à l'opérateur laplacien discret mais sans le facteur $\frac{1}{h^2}$

Lorsque nous implémentons ce modèle et en utilisant les mêmes paramètres que ceux présentés dans [2] nous obtenons des résultats semblables comme montré à la figure 9.

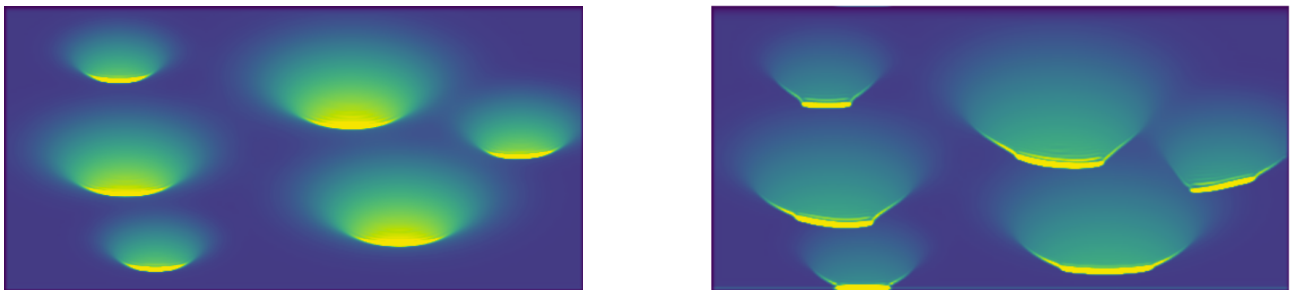


FIGURE 8 – Simulation d'une série de gaussiennes tirée de [2]

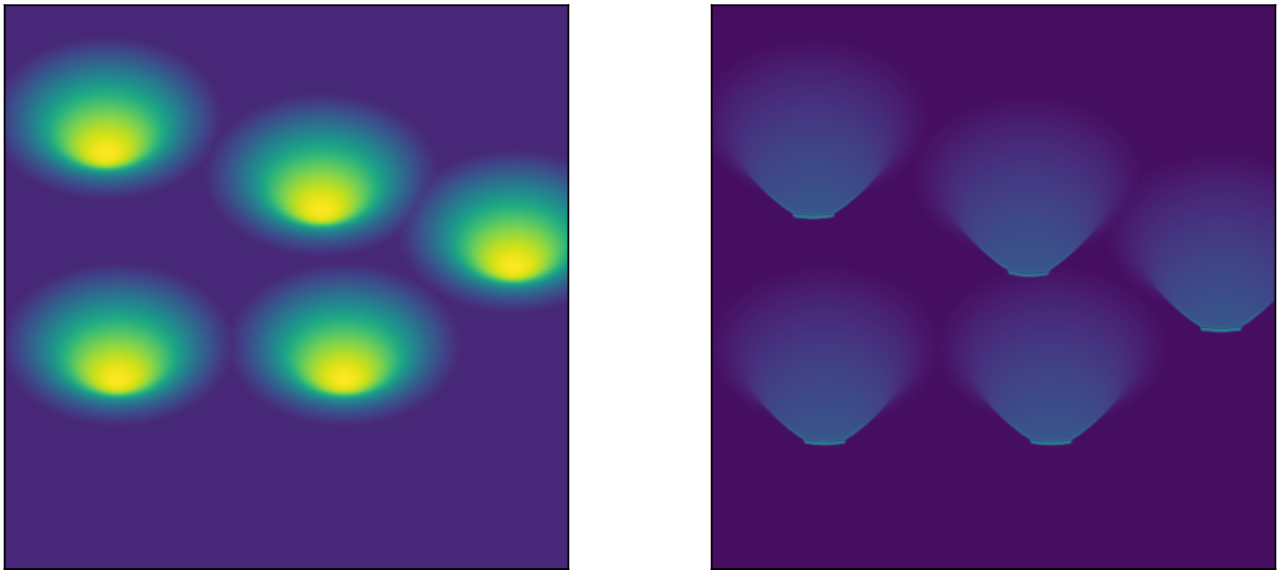


FIGURE 9 – Simulation d'une série de gaussiennes avec le modèle implémenté 40 pour $\zeta = 10$, $\epsilon = 10$ et $\eta = 2$

**Modèle avec viscosité adaptée à la grille**

Initialement, nous avons testé le modèle décrit dans l'article [2] avec les paramètres donnés dans celui-ci. Cela ne fonctionnait évidemment pas car ces paramètres étaient ceux du modèle implémenté. En cherchant à comprendre pourquoi un modèle fonctionnait et pas l'autre, nous avons commis une erreur. En discrétisant de manière erronée l'intégrale de l'opérateur laplacien, nous avons développé le modèle suivant pour le cas plan :

$$\hat{f} = \frac{Mh}{K} \left\{ \epsilon \left( \Delta u_p - \Delta u_q \right) + \zeta \left[ \boldsymbol{z}_p - \boldsymbol{z}_q \right] + \eta (u_p - u_q) \right\} \tag{41}$$

avec $K = h^2 + \tau M(10\epsilon + 2\eta)$ et $\Delta$ qui correspond à nouveau à l'opérateur laplacien discret sans le facteur $\frac{1}{h^2}$

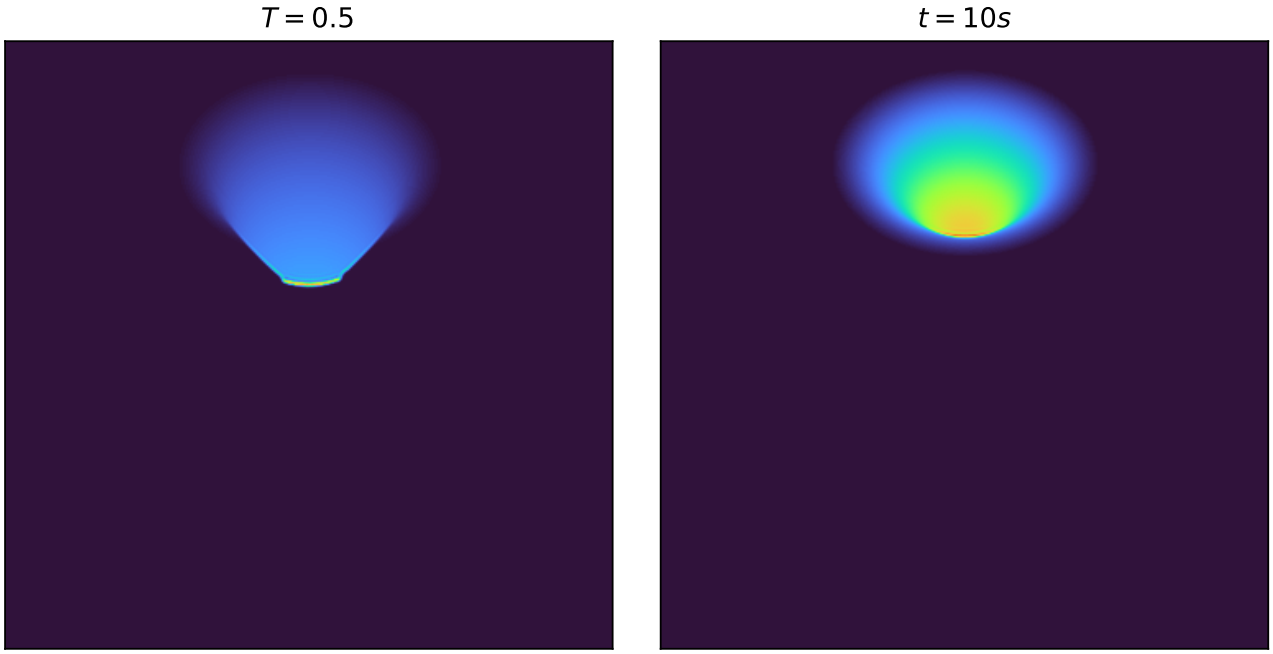



FIGURE 10 – Simulation d'une gaussienne pour le modèle avec viscosité adaptée à la grille 41 en t=10s et T=0.5

Les résultats obtenus pour l'écoulement d'une gaussienne sont montrés à la figure 10. Puisque le modèle est obtenu à partir d'une erreur mais que les résultats semblent valides, intéressons-nous à l'équation qui est réellement résolue :

$$\partial_t u + \nabla \cdot f = 0$$
$$f = \frac{u^3}{3} \nabla \left( -\zeta \boldsymbol{z} + \epsilon \Delta u + \eta u + \epsilon (h^2 - 1) \Delta u \right)$$

Nous observons un terme supplémentaire : $\epsilon(h^2 - 1)\Delta u$. Il s'agit là d'un terme diminuant la dissipation du film plus ou moins fort en fonction de la discrétisation de la grille. En pratique cela améliore la stabilité car pour une grille de 512 par 512 et la même initialisation gaussienne que précédemment la solution est stable pour un pas de temps maximum $\tau = 7 \cdot 10^{-4}$. Cet opérateur laplacien sans le facteur $\frac{1}{h^2}$ peut être vu comme un opérateur où la viscosité est adaptée à la grille. Cela ne correspond plus à la physique du film fin mais étant donné que la métrique dans laquelle nous jugeons de la qualité de nos simulations correspond au visuel, il s'agit là d'une solution intéressante. Cette stabilité accrue laisse sous-entendre que le terme de viscosité est le terme limitant.

Pour vérifier cette hypothèse, nous testons la stabilité de nos différents modèles pour la même initialisation mais pour des discrétisations spatiales différentes. Comme nous pouvons l'observer à la figure 11 la stabilité expérimentale est en $\mathcal{O}(n^4)$ pour le modèle de l'article et en $\mathcal{O}(n^2)$ pour le modèle 41. Ce résultat s'explique par le fait qu'il s'agit d'une équation faisant rentrer en jeu la dérivée 4e de $u$ normalement. Dans le cas du modèle 41, le facteur $h^2$ vient en quelque sorte contrer l'opérateur laplacien, ne reste alors qu'une dérivée seconde et donc une convergence en $\mathcal{O}(n^2)$. Cela confirme bien l'hypothèse que la stabilité est dominée par le terme visqueux.

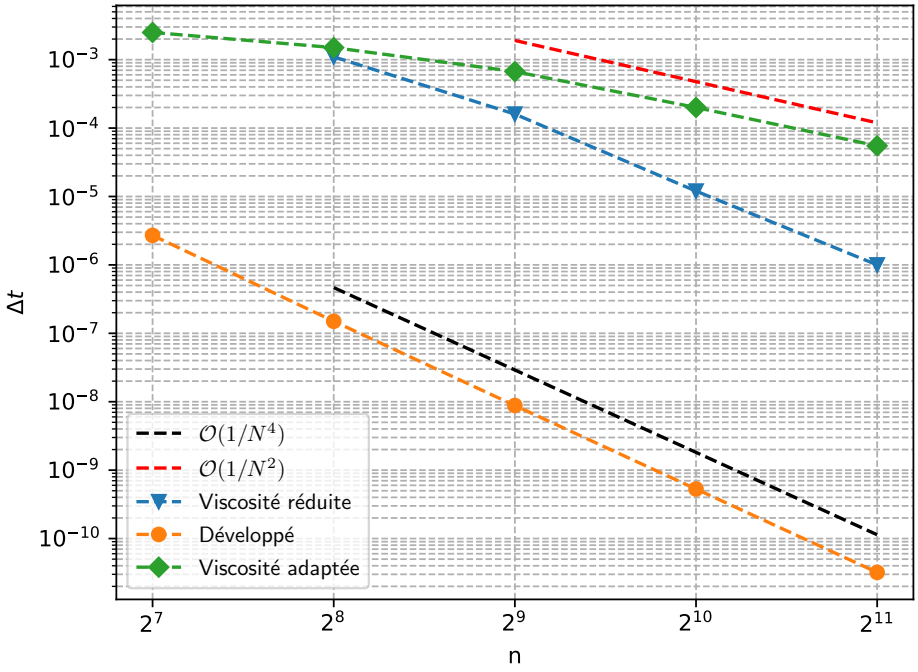



FIGURE 11 – Stabilité des modèles

Dans le graphe 11, la stabilité du modèle implémenté 40 n'est pas représentée car les paramètres pour $\epsilon$ et $\zeta$ ne coïncident pas avec les autres modèles. Les valeurs de $\epsilon$ utilisées varient de 5 à 20 et ne correspondent donc pas du tout à l'hypothèse du film fin. L'initialisation n'est également pas la même, le modèle implémenté 40 nécessite des valeurs de $u$ bien plus petites.

**Modèle à viscosité réduite**

Pour améliorer la stabilité de notre modèle sans pour autant qu'il soit erroné, nous pouvons nous intéresser à diminuer l'effet visqueux. En effet, dans le premier chapitre nous avons adimensionnalisé les équations et avons fait l'hypothèse que $\bar{\zeta} = \frac{zeta}{\epsilon^2} \to \mathcal{O}(1)$ lorsque $\epsilon \to 0$. Nous avions également vu que nous pouvions changer cette adimensionnalisation pour ne plus faire cette hypothèse sur $\zeta$ et donc diminuer l'effet de la tension de surface. Cela nous mène au modèle pour un écoulement sur surface plane suivant :

$$\hat{f} = \frac{Mh}{K}\left\{\epsilon^3\left(\Delta_h u_p - \Delta_h u_q\right) + \zeta\left[z_p - z_q\right] + \eta(u_p - u_q)\right\} \tag{42}$$

avec $K = h^2 + 2\tau M(\frac{5}{h^2}\epsilon^3 + \eta)$

Nous obtenons alors une meilleure stabilité avec un pas de temps maximum de $\tau = 1.2 \cdot 10^{-5}$. Les résultats obtenus pour l'écoulement d'une gaussienne se rapprochent des modèles 41 et 40 comme nous pouvons le voir à la figure 12.

Nous pouvons observer à la figure 11 que la stabilité expérimentale en fonction du $n$ de la discrétisation spatiale est bien de l'ordre de $\mathcal{O}\left(n^4\right)$ comme pour le modèle de l'article [2].

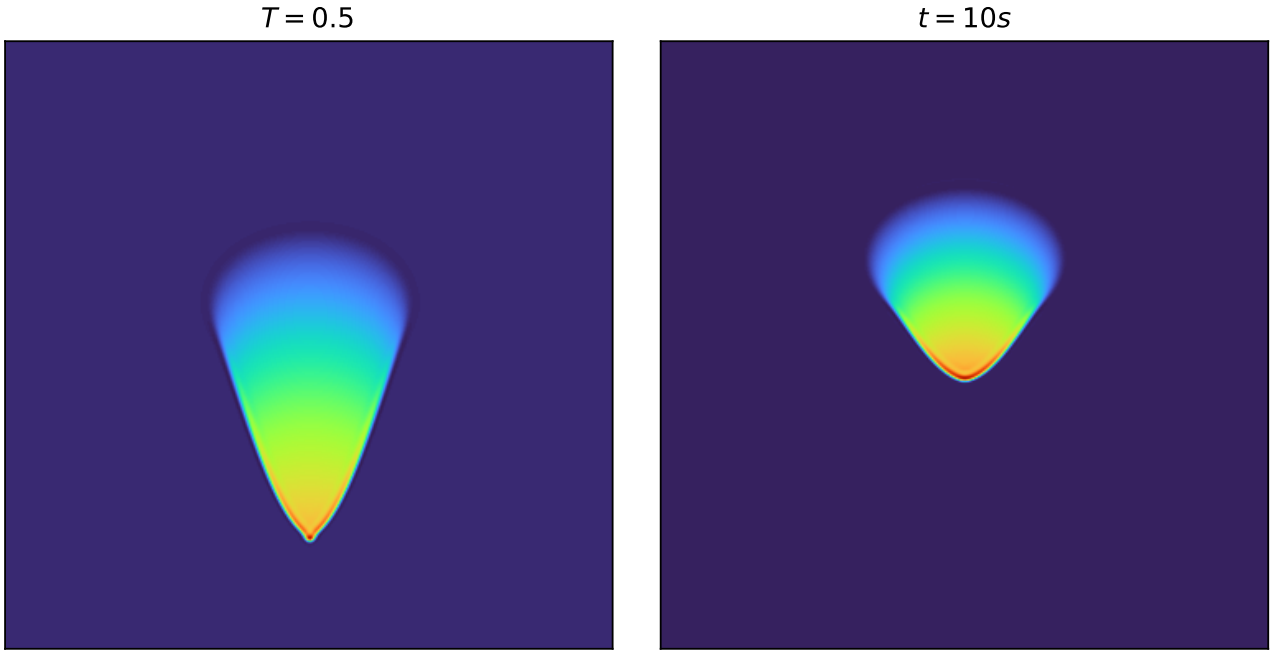



FIGURE 12 – Simulation d'une gaussienne pour le modèle à viscosité réduite 42 en t=10s et T=0.5

# 4 Implémentation

L'objectif recherché pour l'implémentation est de réaliser des simulations en temps réel permettant des interactions et avec un pas de temps suffisamment petit que pour obtenir des simulations de qualité. Dans cette perspective, notre regard s'est évidemment tourné vers la programmation parallèle et l'utilisation du GPU. Ce chapitre discute des différentes solutions apportées pour obtenir la simulation la plus rapide possible. Le code est disponible sur le github https://github.com/dazerki/TFE_Thin_Film

## 4.1 Une implémentation parallèle

Nos différents modèles finaux 37, 40, 41 et 42 sont très locaux et ne nécessitent de connaître la valeur de densité de masse que des quelques cases adjacentes comme représenté à la figure 13. Cette particularité nous permet d'espérer obtenir une implémentation hautement parallèle. Dans la programmation parallèle, il y a un risque de concurrence entre les différents *threads* : le résultat diffère en fonction de quel *thread* finit sa tâche en premier. Pour éviter ce problème, nous allons utiliser une division de la grille bien particulière évitant que le programme ne lise et n'écrive en même temps une valeur de $u$. Cette implémentation a également la particularité de considérer la méthode de pas fractionnaires, où les nouvelles valeurs calculées sont directement utilisées pour le calcul des flux suivants, à l'intérieur du même pas de temps.
La division de la grille est présentée à la Figure 13, à chaque case est associée une parité $\rho_{ij}$ où $i$ et $j$ sont les indices du tableau. La parité est calculée grâce à la formule :

$$\rho_{ij} = ((d_j + 1)i + (d_i + 1)j + \rho) \mod 4$$

Où $d_i$ et $d_j$ indiquent s'il s'agit du calcul du flux horizontal $(d_i, d_j) = (1, 0)$ ou bien vertical $(d_i, d_j) = (0, 1)$. Le paramètre $\rho$ varie entre 0 et 3 pour permettre de considérer le flux entre toutes les cases.
Le flux est calculé entre les cases qui ont une parité égale à 2 et 3 ; pour calculer ce flux les cases numérotées 1 et 0 sont nécessaires.

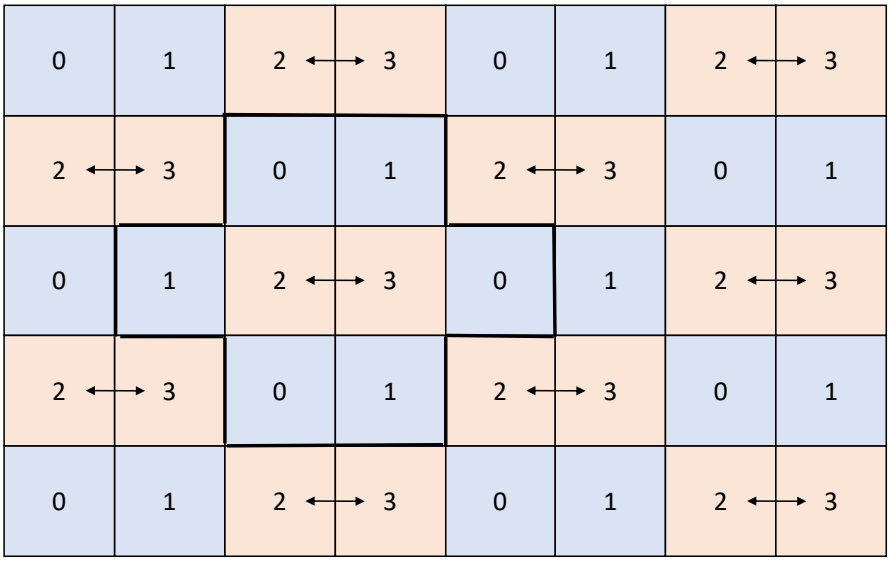



FIGURE 13 – Division de la grille pour l'implémentation parallèle

## 4.2 Algorithme

Maintenant que nous savons comment calculer les flux de manière parallèle sans que les *threads* n'interfèrent entre eux, nous pouvons écrire l'algorithme de notre programme. Celui-ci est composé d'une boucle sur les directions, à l'intérieur de laquelle on retrouve une boucle sur le paramètre $\rho$ permettant ainsi de calculer l'ensemble des flux. Enfin, nous avons notre boucle sur chaque case de notre maillage régulier et à l'intérieur le calcul du flux suivant l'équation 41 pour le cas du modèle de frottement :

**Algorithm 1** Gausse-Seidel

**for each** Direction $(d_i, d_j) \in (1,0),(0,1)$ **do**
  **for each** $\rho \in [0,1,2,3]$ **do**
    **for each** cellule aux coordonées (i,j) **do**
      $\rho_{ij} \leftarrow ((d_j+1)i + (d_i+1)j + \rho) \mod 4$
      **if** $\rho_{ij} = 3$ **then**
        $(i', j') \leftarrow (i - di, j - dj)$
        $\Delta_{ij} \leftarrow u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}$
        $\Delta_{i'j'} \leftarrow u_{i'+1,j'} + u_{i'-1,j'} + u_{i',j'+1} + u_{i',j'-1} - 4u_{i',j'}$
        $M \leftarrow M(u_{i,j}, u_{i',j'})$
        $K \leftarrow h^2 + 2\tau M(4\epsilon + \eta)$
        $W_{i,j} \leftarrow \zeta \boldsymbol{z}_{i,j}$
        $W_{i',j'} \leftarrow \zeta \boldsymbol{z}_{i',j'}$
        $f \leftarrow \frac{Mh}{K}\left(\epsilon(\Delta u_{i,j} - \Delta u_{i',j'}) + W_{i,j} - W_{i',j'} + \eta(u_{i,j} - u_{i',j'})\right)$
        $\delta u = max(-u_{i',j'}, min(\frac{\tau}{h} f, u_{i,j}))$
        $u_{i,j} \leftarrow u_{i,j} - \delta u$
        $u_{i',j'} \leftarrow u_{i',j'} + \delta u$
      **end if**
    **end for**
  **end for**
**end for**

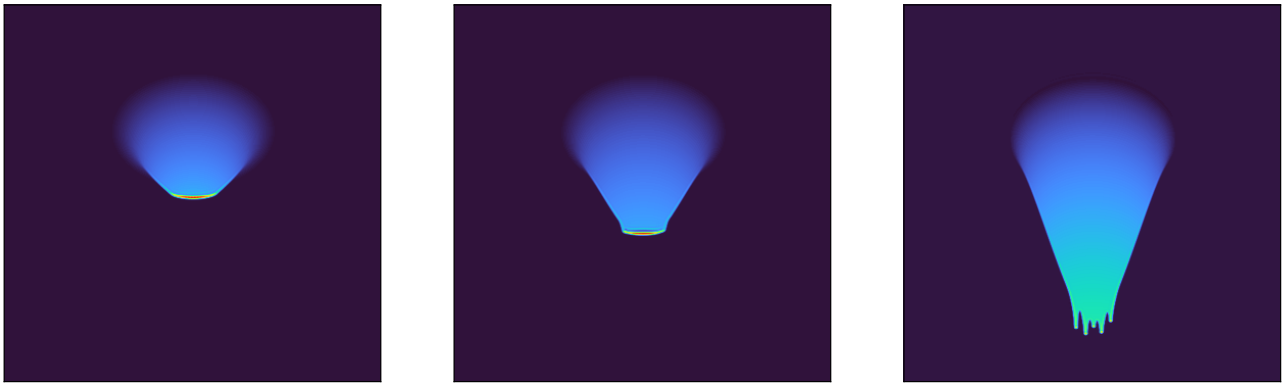
## 4.3 Initialisation et surface 3D

L'algorithme défini, il nous reste une seule étape avant d'obtenir des résultats : l'initialisation. L'initialisation peut être divisée en deux parties : l'initialisation des densités de masse du fluide d'une part et l'initialisation de la surface solide de l'autre.
Si l'on considère une vitesse nulle à l'interface solide liquide, un léger film doit être présent sur l'ensemble de la surface pour que le fluide puisse s'écouler. C'est le cas considéré par les auteurs de *Real-Time Viscous Thin Films* et cela a l'avantage de facilement simuler des obstacles en imposant à certaines zones la valeur de $u = 0$.
Pour ce qui est de la surface, nous avons besoin de connaître le rayon de courbure moyen $H$, $T = H^2 - 2K$ où K est le rayon de courbure Gaussien et $\theta$ l'angle entre la normale de la surface et le vecteur gravitationnel $\vec{g}$. Nous discrétisons ces fonctions continues en considérant la valeur au centre de la case. De plus nous considérons une paramétrisation de Monge de la surface : la surface peut être vue comme une grille cartésienne en 2D à laquelle nous avons assigné une hauteur $h$ pour chaque cellule. Cette paramétrisation nous permet de calculer $H$, $K$ et $\cos\theta$ de la sorte :

$$H = \frac{h_{xx}\left(1 + h_y^2\right) + h_{yy}\left(1 + h_x^2\right) - 2h_{xy}h_x h_y}{2\left(1 + h_x^2 + h_y^2\right)^{3/2}}$$

$$K = \frac{h_{xx}h_{yy} - (h_{xy})^2}{\left(1 + h_x^2 + h_y^2\right)^2}$$

$$\cos\theta = \frac{-h_y}{\sqrt{1 + h_x^2 + h_y^2}}$$



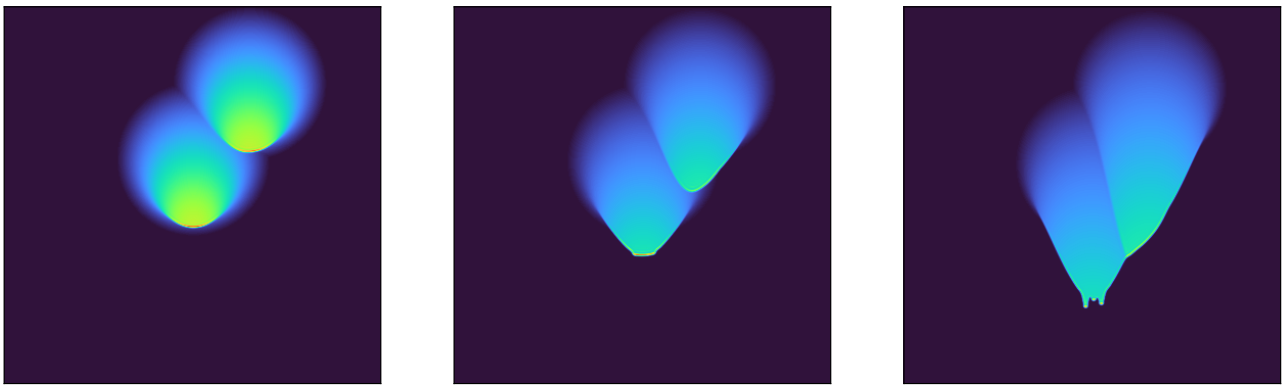
(a) $t = 0.5$ (b) $t = 1$ (c) $t = 3$

FIGURE 14 – Evolution d'une gaussienne avec $\zeta = 5$

où $h_x$, $h_y$ sont les dérivées de h par rapport à x ou y et $h_{xx}$, $h_{yy}$ sont les dérivées secondes de h par rapport à x et à y. Ces dérivées sont calculées au moyen de différences finies centrées d'ordre 2.

## 4.4 Résultats

Intéressons-nous maintenant aux résultats. A la figure 14 nous pouvons voir ce que donne notre simulation sur une grille de 512 par 512 pour une initialisation gaussienne et des valeurs de paramètres de $\epsilon = 0.01$, $\eta = 0$, $\tau = 0.001$ et $\zeta = 5$. Le fluide a un aspect visqueux, s'écoule lentement en formant un front de matière qui devient progressivement instable, jusqu'à ce que se forment des *fingering*. Il s'agit là d'un effet physique intéressant visuellement et que l'on retrouve dans de nombreux écoulements rampants.



(a) $t = 0.5$ (b) $t = 1$ (c) $t = 2$

FIGURE 15 – Interaction entre deux gouttes

L'interaction entre deux gouttes a également attiré notre attention. Le fluide tend à s'écouler là où la densité de masse est la plus importante. Sur la figure 15, nous voyons clairement cet effet. La gaussienne la plus basse coule de manière verticale, dans le sens du potentiel de gravité, tandis que la seconde se décale vers la gauche. Dans le modèle, cela est dû à la mobilité qui est

(a) $t = 0.5$ (b) $t = 2s$ (c) $t = 4s$

FIGURE 16 – Évolution d'une gaussienne avec 2 obstacles

(a) $t = 0.5$ (b) $t = 2s$ (c) $t = 3s$

FIGURE 17 – Evolution d'une gaussienne avec 2 obstacles

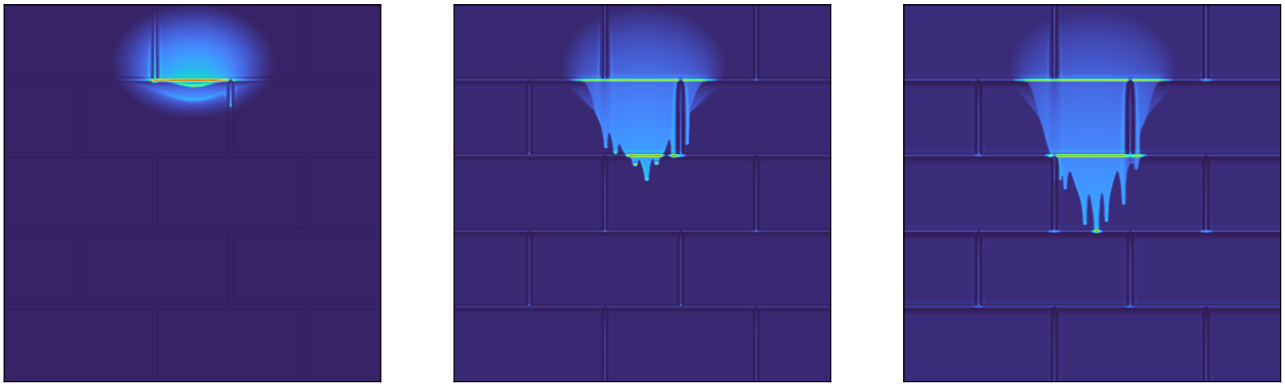
plus grande lorsque les valeurs de $u_p$ et $u_q$ sont plus grandes.

Considérer une vitesse nulle à l'interface comme nous l'avons fait en 2 a le grand avantage de pouvoir facilement donner l'impression d'obstacle. En effet, s'il n'y a pas de vitesse en $h = 0$ alors le fluide ne peut pas s'écouler là où il n'y a pas de matière. En imposant $u = 0$ sur certaines zones du domaine nous donnons l'impression d'un obstacle comme nous pouvons le voir aux figures 16 et 17.

Enfin, le modèle a été développé pour des surfaces 3D et fait intervenir les différents rayons de courbure de celles-ci. La figure 18 montre l'écoulement d'une gaussienne sur un mur de briques. Le fluide est attiré par les zones de creux dans le mur et est repoussé par les zones bombées.

## 4.5 Performance sur CPU

L'objectif étant d'obtenir des simulations en temps réel, la performance a pris une part importante de ce travail. Cette volonté se retrouve dans le modèle de par son aspect hautement parallélisable mais également dans l'implémentation qui se veut optimale. Avant de passer sur GPU nous avons d'abord implémenté l'algorithme 1 sur CPU. Tous les résultats qui suivent ont



(a) $t = 0.1$ (b) $t = 1$ (c) $t = 2$

FIGURE 18 – Evolution d'une gaussienne sur une surface 3D

été obtenus au moyen du processeur Intel Core I7-9750H. Avec ses 6 coeurs et une fréquence maximale de 4.5 GHz, ce CPU atteint 337.4 GFlop/s (*Floating operation per seconde*) en simple précision.

Une première version a été réalisée en C avec un seul coeur et optimisée à l'aide du profiler HPCToolkit . Ce dernier permet de récolter un grand nombre de données relevant de la performance de notre programme comme le nombre de cycle, Flop/s, instructions etc. Le nombre de cycle et le nombre de Flop/s sont 2 mesures importantes car elles permettent d'identifier quelles opérations prennent le plus de temps mais également celles qui sont les moins efficaces en terme de puissance de calcul. Cela a permis d'optimiser grandement le code et d'arriver à environ 9 GFlops pour un seul coeur dans le cas 3D. Le cas d'une surface plane est plus rapide à simuler car le programme doit aller chercher en mémoire moins de valeurs et également réaliser moins d'opérations. Le tableau 2 reprend les performances pour ces deux cas mais aussi lorsque le programme est parallélisé pour le calcul de 1000 itérations.

La figure 19 présente de manière visuelle la répartition des cycles passés pour chaque ligne de l'algorithme 1 ainsi que l'efficacité en terme d'opération par cycle. Cette efficacité est obtenue en divisant le nombre réel de Flop exécuté par le nombre de Flop que le processeur pourrait exécuter dans des conditions optimales. Dans le cas du CPU utilisé, le processeur peut effectuer 12 opérations par cycle et donc $\eta = 100 \cdot \frac{Flop}{12 n_{cycle}}$.
Une grande partie du temps est passée à calculer la parité de la cellule pour définir s'il est nécessaire de calculer le flux à cet endroit. Ceci s'explique par le fait que le programme doit faire ce calcul pour chaque case tandis que les étapes 6 à 16 ne sont réalisées que pour un quart de la grille, lorsque la parité vaut 3. Les instructions les plus chronophages sont les calculs de la variation de u, du flux et de K.
Si nous ignorons l'initialisation non représentée dans l'algorithme, 96.8 % du programme se trouve à l'intérieur de la boucle 3-17.

**Algorithm 1** Gausse-Seidel

1: **for each** Direction $(d_i, d_j) \in (1,0),(0,1)$ **do**
2: **for each** $\rho \in [0,1,2,3]$ **do**
3: **for each** cellule aux coordonées (i,j) **do** — 3.2%
4: $\rho_{ij} \leftarrow ((d_j+1)i + (d_i+1)j + \rho) \mod 4$ — 20.4%
5: **if** $\rho_{ij} = 3$ **then** — 4%
6: $(i', j') \leftarrow (i - di, j - dj)$ — 6%
7: $\Delta_{ij} \leftarrow u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}$ — 4.1%
8: $\Delta_{i'j'} \leftarrow u_{i'+1,j'} + u_{i'-1,j'} + u_{i',j'+1} + u_{i',j'-1} - 4u_{i',j'}$ — 4.1%
9: $M \leftarrow M(u_{i,j}, u_{i',j'})$ — 2.8%
10: $K \leftarrow h^2 + 2\tau M(4\epsilon + \eta)$ — 8.4%
11: $W_{i,j} \leftarrow \zeta \boldsymbol{z}_{i,j}$ — 3.5%
12: $W_{i',j'} \leftarrow \zeta \boldsymbol{z}_{i',j'}$ — 3.5%
13: $f \leftarrow \frac{Mh}{K}(\epsilon(\Delta u_{i,j} - \Delta u_{i',j'}) + W_{i,j} - W_{i',j'} + \eta(u_{i,j} - u_{i',j'}))$ — 11.8%
14: $\delta u = max(-u_{i',j'}, min(\frac{\tau}{h} f, u_{i,j}))$ — 13.4%
15: $u_{i,j} \leftarrow u_{i,j} - \delta u$ — 2.4%
16: $u_{i',j'} \leftarrow u_{i',j'} + \delta u$ — 2.4%
17: **end if**
18: **end for**
19: **end for**
20: **end for**

FIGURE 19 – Pourcentage de cycles passés dans chaque partie de l'algo

**Parallélisation :**

Dans l'algorithme 1, la boucle sur les pixels (lignes 3 à 17) est un boucle qui peut être parallélisée. Pour ce faire, nous utilisons OpenMP (OMP en abrégé) qui permet simplement de paralléliser la boucle à l'aide de l'instruction *#pragma omp parallel for*.

La figure 20 (gauche) représente le temps d'exécution nécessaire en fonction du nombre de *threads* utilisés par OpenMP. Nous pouvons observer que les temps suivent la loi d'Amdahl pour une proportion du code parralélisable de 96.8 %. La loi d'Amdahl prédit le temps d'exécution d'un programme en fonction du nombre de *threads* $s$, du temps d'exécution du programme mono-thread $T_{mono}$ et de $p$, la proportion parralélisable du programme :

$$T = T_{mono}\left((1-p) + \frac{p}{s}\right)$$

Notre programme suit bien cette loi lorsque nous modifions le nombre de *threads* utilisés par OpenMP. Cependant le code est plus lent lorsqu'on utilise OpenMP avec 1 seul *thread* par rapport à une implémentation sans OpenMP. Cela a pour effet de ne pas améliorer les performances de notre programme par presque 6 comme le prévoit la loi d'Amdahl mais par environ 3 comme nous pouvons le voir dans le tableau 1 dans la colonne "Coefficient". La figure 20 (droite) présente ce facteur d'amélioration par rapport à OpenMP 1 *thread* et par rapport à une implémentation sans OpenMP. Nous observons une évolution quasi linéaire pour les 2 cas et une pente moins proche de l'unité pour l'implémentation sans OpenMP. Nous avons également mesuré la variation de temps entre le cas où nous utilisons OpenMP avec 1 *thread* et le cas où nous ne l'utilisons pas, il s'agit de la dernière colonne du tableau 1. Cet indicateur correspond au coût nécessaire à la répartition des instructions aux différents *threads*. Nous observons que ce coût est proportionnel à la grille utilisée mais pas au temps nécessaire pour exécuter un passage dans la boucle parallélisée. Le temps nécessaire à openMP pour distribuer une itération de la boucle parmi les *threads* est alors de $\simeq 1.896 \cdot 10^{-8}$ s.

| Implémentation | OMP 6 *threads* | OMP 1 *thread* | Sans OMP | Coefficient | $\Delta t$ 1 *thread* |
|---|---|---|---|---|---|
| 3D, N=512 | 2.26 | 12.2 | 7.23 | 3.19 | 4.97 |
| 2D, N=512 | 2.1 | 10.8 | 5.72 | 2.72 | 5.08 |
| 3D, N=1024 | 9.73 | 51.1 | 31.6 | 3.25 | 19.4 |
| 2D, N=1024 | 8.01 | 41.9 | 22.7 | 2.83 | 19.2 |

TABLE 1 – Performance d'OpenMP sur CPU pour 1000 itérations

Ce problème d'*overhead* peut en partie être résolu par une implémentation GPU qui est bien plus optimisé pour du calcul parallèle.

La mesure des GFlop/s du programme est un bon moyen de connaître la proportion de puissance du CPU utilisée. Le tableau 2 permet d'observer que le cas 3D nécessite environ 3 fois plus de calcul que le cas 2D. Pourtant le temps d'exécution n'est pas triplé, ce qui indique que le programme est fort probablement limité en terme d'accès mémoire. Dans le cas parallèle et en 3D le programme utilise au maximum le processeur avec 27.9 GFlop/s.

| Implémentation | temps $[s]$ | GFlop | GFlop/s |
|---|---|---|---|
| Mono-thread 3D | 7.75 | 65.5 | 8.45 |
| Mono-thread 2D | 5.81 | 22.0 | 3.79 |
| Parallèle 3D | 2.35 | 65.5 | 27.9 |
| Parallèle 2D | 2.0 | 22.0 | 11.0 |

TABLE 2 – Performance sur CPU pour 1000 itérations

**Affichage :**
Dans des simulations en temps réel, l'affichage prend une nouvelle dimension. En effet, son exécution doit être efficace et rapide pour ne pas ralentir inutilement le programme. Si nous arrivons à obtenir un affichage dont le temps d'exécution est d'un ordre inférieur à celui de la simulation, il devient alors possible d'avoir un grand nombre d'images par seconde sans pour autant ralentir le rendu de la simulation. Trois outils ont été testés pour l'affichage :

- *matplotlibcpp* un wrapper en C autour de la librairie bien connue *matplotlib*. Son principal avantage est sa facilité de prise en main.
- *Basic OpenGL Viewer* (*BOV*), un wrapper autour de la libraire OpenGL et développé à l'UCLouvain.
- Une implémentation optimisée pour notre cas d'OpenGL.

Les temps nécessaires à l'affichage d'une grille de 512 par 512 cases sont les suivants :

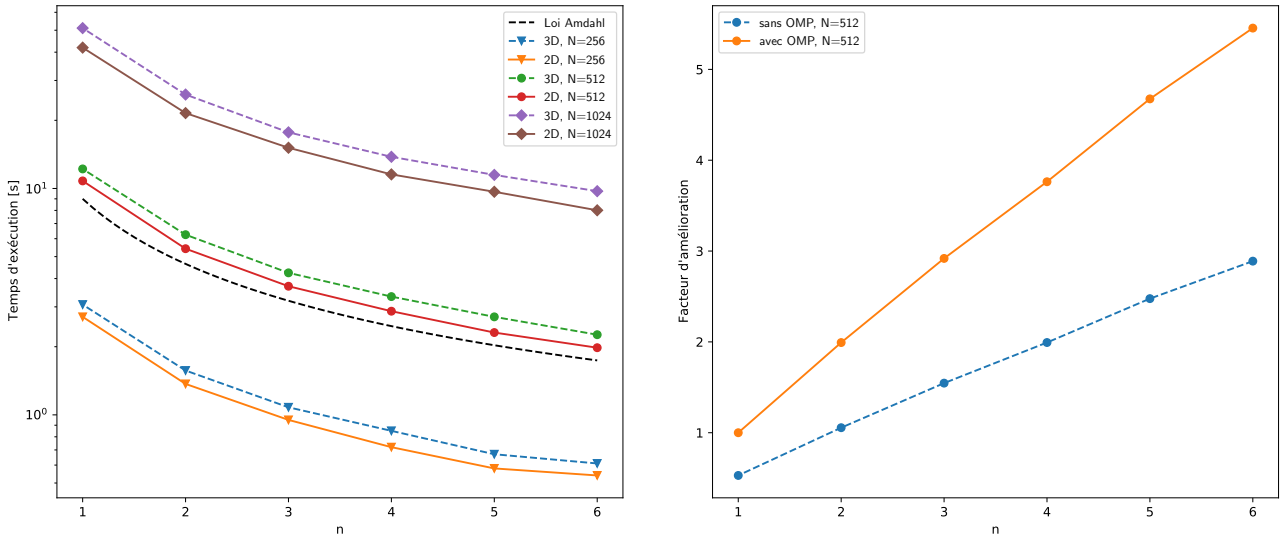



FIGURE 20 – (Gauche) Performance sur CPU en 2D/3D pour un nombre de *threads* variable. (Droite) Facteur d'amélioration par rapport à OpenMP 1 *thread* et par rapport à une implémentation sans OpenMP

| Implémentation | temps $[ms]$ |
|---|---|
| *matplotlibcpp* | 110 |
| *BOV* | 50 |
| OpenGl | 0.7 |

L'implémentation maison d'OpenGL est la plus rapide, ce qui peut surprendre car *BOV* et *matplotlib* utilisent également OpenGL. Pour *BOV*, nous pouvons expliquer cette différence de temps. Dans notre cas, la grille est fixe et seulement les couleurs associées à chaque cellule doivent être mises à jour. BOV est très rapide mais ne permet pas de mettre à jour uniquement les couleurs, il faut alors retracer la grille déjà existante ce qui prend davantage de temps
Ce temps d'affichage est satisfaisant puisque le temps d'exécution de la simulation pour un pas de temps est de environ $2ms$ sur CPU et environ $0.5ms$ sur GPU. De plus, avec 0.7ms par affichage, nous pouvons donc obtenir des simulations avec un taux de rafraîchissement de l'image de 60 Hz en ne dédiant que 4% du temps à l'affichage.

# 5 Implémentation GPU

L'implémentation sur CPU a permis de montrer le potentiel du modèle lorsqu'il est exécuté de manière parallèle. C'est donc tout naturellement que nous nous sommes tournés vers le GPU, optimisé pour les exécutions parallèles et avec une plus grande puissance de calcul. L'utilisation des processeurs graphiques pour calculer et simuler a débuté il y a une dizaine d'années avec notamment Tim Warburton [17]. Cependant comme nous le verrons par la suite, le GPU a son accès mémoire plus lent que celui du CPU. Pour pallier cet inconvénient nous avons mis au point différentes implémentations développées dans cette section, toutes réalisées en CUDA.

## 5.1 Architecture du GPU

Avant de nous intéresser à ces différentes implémentations, il est important de bien comprendre le fonctionnement d'un GPU. Un GPU est un ensemble de petites machines de type SIMD indépendantes appelées multi-processeurs et partageant une mémoire globale. SIMD pour *Single instruction, multiple data* est une catégorie d'architecture dans laquelle la machine exécute la même opération mais sur des données différentes simultanément.
Chaque multi-processeur est composé de plusieurs processeurs et de différents espaces de mémoire partagés entre les processeurs : la mémoire partagée, le cache des constantes et le cache des textures. La mémoire commune aux différents multi-processeurs aussi appelée mémoire de l'appareil est composée de la mémoire RAM du GPU et des caches L1 et L2. Ces différentes éléments sont représentés sur le schéma 21. Les caches sont des espaces de mémoire plus rapides que la mémoire RAM dans lesquels sont stockés des variables fréquemment utilisées pour éviter de devoir accéder à de la mémoire lente de manière répétitive. Le cache L1 est le plus rapide et se situe au même endroit que la mémoire partagée, le cache L2 est plus lent mais est de plus grande taille. Le GPU est également composé d'autres éléments utiles à son bon fonctionnement comme le planificateur de warp mais nous ne les développerons pas dans ce travail.

Toutes les simulations et tous les résultats obtenus ont été réalisés au moyen du GPU GeForce GTX 1650 de la marque Nvidia. En cadence de base, il permet de calculer 83 GFlop/s en double précision, 2.7 TFlop/s en simple précision et 5.4 TFlop/s en *half*. Nos calculs étant réalisés en simple précision c'est la valeur de 2.7 TFlop/s que nous retiendrons. Ce GPU est donc 8 fois plus puissant que le CPU que nous avons utilisé précédemment. A titre de comparaison, la carte graphique dernier cri GeForce RTX 3090 (qui a beaucoup fait parler d'elle à cause d'une rupture de stock dés le jour de sa sortie) a comme performance 556 GFlop/s en double précision et 35.6 TFlop/s en simple précision et *half*. Elle est donc 10 fois plus performante en simple précision que le GPU utilisé dans ce travail. Nous pouvons également comparer CPU et GPU sur d'autres critères tel que les Flop/€ ou les Flop/watt. Pour notre CPU nous obtenons 1.04 GFlop/€ et 3.55 GFlop/watt là où notre GPU est bien plus efficace avec 24.4 GFlop/€ et 40 GFlop/watt. La programmation sur GPU est donc très intéressante d'un point de vue économique et environnemental.

L'architecture de l'appareil est perçue différemment en fonction du langage utilisé. Nous avons dans notre cas utilisé CUDA qui est le langage développé par Nvidia pour profiter au mieux du GPU utilisé ici. La partie du code spécifique au langage est minime et il ne serait donc pas

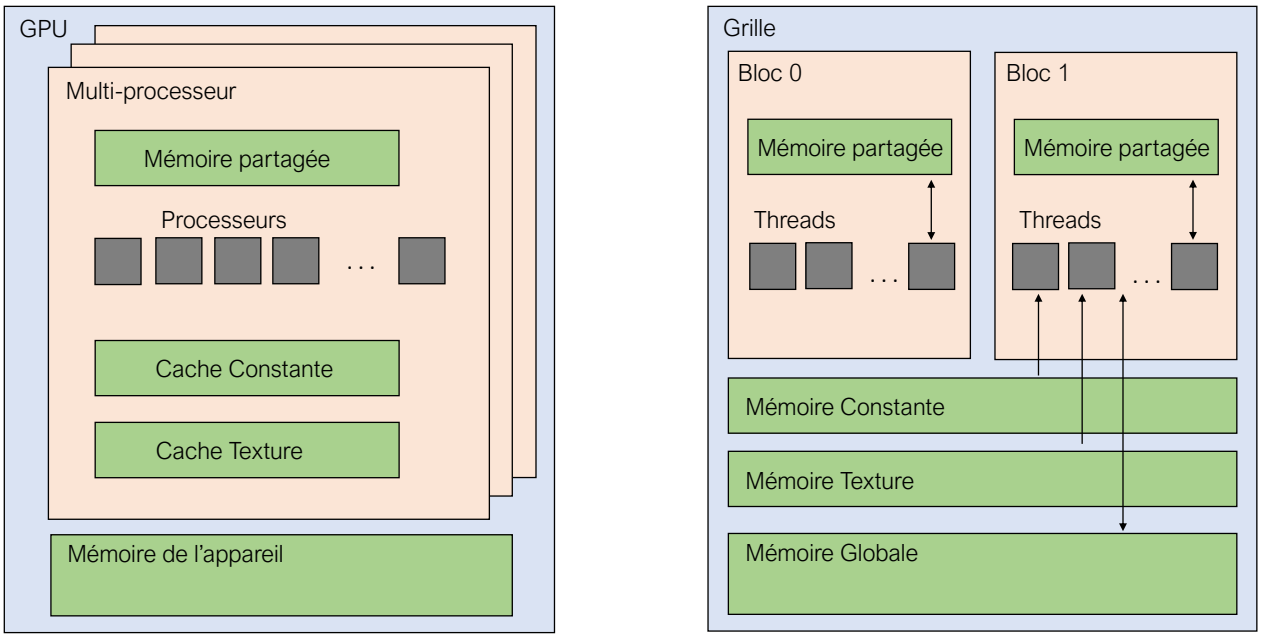



FIGURE 21 – (Gauche) Représentation schématique de l'architecture classique d'un GPU. (Droite) Perception du GPU dans le langage CUDA

très difficile de traduire le code en OpenCL par exemple pour être compatible avec des GPU d'autres marques.
Les fonctions qui s'exécutent sur le GPU s'appellent des kernels et sont écrites en CUDA. Ces fonctions sont lancées via un programme en C sur l'*host* qui est dans notre cas le CPU. Lorsque l'*host* lance l'exécution des kernels, ceux-ci sont exécutés en parallèle par une grille de *threads*. La taille de la grille correspond au nombre total de *threads* qui exécutent le kernel. Cette grille est divisée en blocs et plusieurs blocs sont exécutés sur un seul multiprocesseur du GPU.
En CUDA la mémoire est perçue de manière légèrement différente que dans l'architecture. Comme nous pouvons le voir à la figure 21, la mémoire partagée est uniquement accessible aux différents threads d'un même bloc tandis que les mémoires globales, des constantes et des textures sont partagées entre les blocs.

## 5.2 Gauss-Seidel

Dans un premier temps, nous avons implémenté l'algorithme 1 de la même façon que sur le CPU en traduisant simplement de C vers CUDA. Les différents tableaux sont alors stockés dans la mémoire globale du GPU et les valeurs de densité de masse sont transférées sur CPU uniquement pour l'affichage par OpenGL. Pour une simulation avec une surface solide en 3D, le modèle a besoin de l'équivalent de 8 tableaux de la taille du maillage : u, H, T et $\cos\theta$ au centre de la cellule, H et $\kappa$ aux interfaces entre les cellules. Il y a deux fois plus d'interfaces que de cellules. Nous avons deux kernels qui vont être exécutés à tour de rôle pour les flux selon l'axe x et y.

Nous obtenons de meilleures performances que sur CPU, le tableau 3 reprend le temps nécessaire pour les différents cas simulés. Le rapport entre le temps GPU et le temps CPU correspond

au facteur d'amélioration de notre implémentation GPU et est de 14.1 dans le cas 2D mais de seulement 4.8 dans le cas 3D.

| Implémentation | temps GPU $[s]$ | temps CPU $[s]$ | Facteur d'amélioration |
|---|---|---|---|
| 3D | 0.49 | 2.35 | 4.80 |
| 2D | 0.142 | 2.0 | 14.1 |
| 3D simplifié | 0.22 | 2.83 | 12.86 |

TABLE 3 – Performance sur GPU pour 1000 itérations

Pour mieux comprendre les raisons de cette différence si grande, nous allons utiliser le profiler de Nvidia *Nsight Compute*. Le graphe figure 22 sont les *rooflines* du GPU, autrement dit les limites inhérentes de l'ordinateur. L'axe horizontal représente l'intensité arithmétique, ce qui correspond au ratio entre travail (exprimé en Flop/s) et trafic de la mémoire (exprimé en byte/s). Nous obtenons alors une mesure en Flop/byte. L'axe vertical quant à lui correspond à la mesure classique de performance : les *Floating operation per second*, Flop/s.
Les lignes horizontales aux valeurs de 83 GFlop/s et 2667 GFlop/s sont respectivement les limites de puissance pic de calcul en double et simple précision. Nos calculs étant réalisés en simple précision pour être le plus rapide possible, la limite qui nous intéresse est donc celle à 2667 GFlop/s. La dernière limite présente sur ce graphe est celle de l'accès mémoire : il s'agit de la droite oblique. La pente de cette droite est déterminée par la vitesse de transfert de mémoire du GPU. Il n'est pas possible de se retrouver au dessus. Enfin, les deux points bleus situés en 0.78 Flop/byte, 75 GFlop/s et en 0.95 Flop/byte, 89.8 GFlop/s correspond aux performances de l'implémentation de Gauss-Seidel pour nos deux kernels. Le calcul du flux dans l'axe des y est donc plus performant que pour l'axe x. Nous n'arrivons pas complètement à expliquer cette différence, les deux kernels réalisent les mêmes calculs et les mêmes accès mémoire. La structure de la mémoire pourrait expliquer cette différence en fonction des accès évanescents à la mémoire.

Les valeurs atteintes sur le graphe 22 sont extrêmement proches de la limite de l'accès mémoire, le programme est donc borné par celle-ci. Dans le cas 2D nous observons que le programme est moins proche de cette limite, cela peut s'expliquer par le fait que le programme doit aller chercher moins de données dans la mémoire.

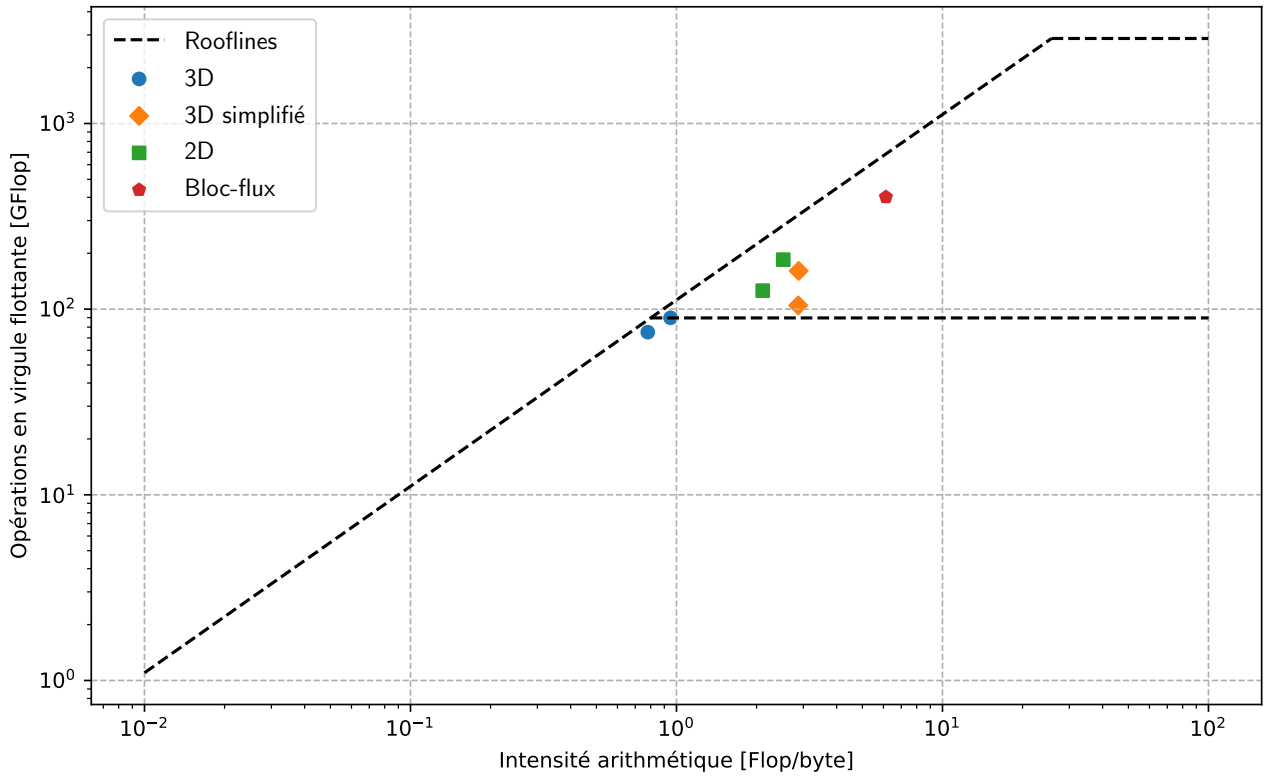



FIGURE 22 – *Rooflines* des différentes implémentations

La figure 23 représente l'architecture et l'accès à la mémoire du GPU utilisé dans le cas 3D. A gauche se trouve le kernel exécutant le code qui communique avec la mémoire. Plus la mémoire est lente/éloignée, plus elle se trouve sur la droite. Idéalement, il faudrait stocker toute la mémoire le plus proche possible du kernel pour obtenir les meilleurs performances. Malheureusement ce n'est pas toujours possible en fonction de l'algorithme utilisé. Dans notre cas, l'algorithme avec Gauss-Seidel doit connaître les valeurs mises à jour pour exécuter l'itération suivante. Or la mémoire partagée ne permet pas cela car elle n'est partagée que sur un bloc et non pas sur toute la grille.
Sur la figure 23 nous voyons que l'accès mémoire le plus durement sollicité est celui de la mémoire globale qui est également la mémoire la plus lente. Le taux de hit-cache L1 vaut 72% ce qui est très correct et s'explique par le fait que les threads côte à côte accèdent à de la mémoire proche l'une de l'autre.
Les performances sur GPU sont tout de même bien supérieures avec 0.49 s pour 1000 itérations en 3D. Il est possible d'améliorer ce résultat pour certains cas particuliers tout en gardant la même implémentation. En effet l'accès à la mémoire est quasi entièrement dû à la caractérisation de la surface 3D. Une autre observation peut être faite : dans le cas d'un écoulement sur des briques, la surface est plane la majorité du temps et donc les valeurs des différents rayons de courbure sont nulles. Le programme prend donc du temps pour aller chercher en mémoire en majorité des 0. Pour éviter cela nous pouvons à l'intérieur du kernel qui calcule le flux et en fonction de la position sur la grille assigner une valeur aux différents paramètres de courbure. On obtient le résultat présenté à la figure 24 et avec un temps de 0.22s pour 1000 itérations comme indiqué dans le tableau 3. Il s'agit là d'une bonne alternative pour des géométries simples.

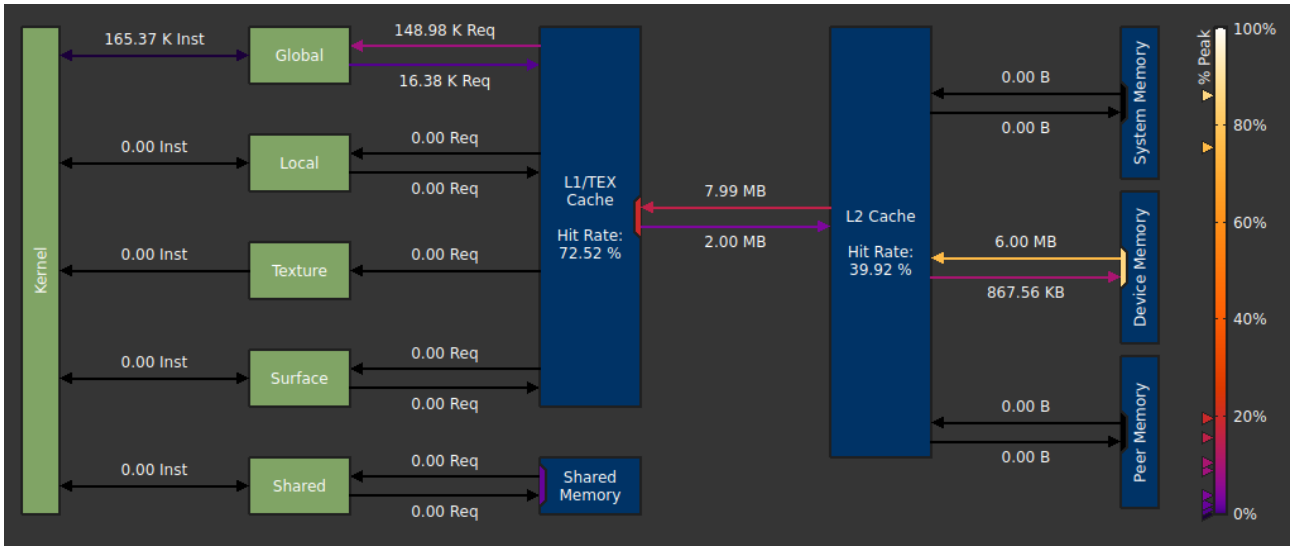



FIGURE 23 – Accès mémoire de l'implémentation Gauss-Seidel

## 5.3 Bloc-flux

Pour éviter de trop accéder à la mémoire sans devoir considérer de cas particulier, nous pouvons changer d'implémentation et donc changer légèrement l'algorithme initial. La première chose qui a été tentée, c'est de considérer des blocs de cases (de 4 fois 4 cases par exemple comme représenté à la figure 25) pour lesquels l'ensemble des flux sont calculés par un seul thread. Les flux sont stockés en mémoire et puis ajoutés après qu'ils aient tous été calculés. Cela ne correspond donc plus à Gauss-Seidel qui calcule les flux en se basant sur les valeurs de densité de masse u déjà mises à jour. Pour améliorer les performances du programme, nous pouvons utiliser la mémoire partagée au lieu de la mémoire globale du GPU. Chaque bloc partage cet espace de la mémoire entre ses threads et ces derniers peuvent y accéder ou la modifier. Ce n'était pas possible de l'utiliser dans le cas de Gauss-Seidel car il faut que l'ensemble des *threads* connaissent les valeurs des $u$ mises à jour par les différents flux partiels. Dans le cas du bloc-flux par contre, nous avons besoin des valeurs de $u$ au pas de temps précédent uniquement. En terme de vitesse d'accès, la mémoire partagée est à peu de chose près équivalente au cache L1 et donc bien plus rapide que la mémoire globale. L'avantage de cette implémentation est qu'elle diminue largement le rapport byte/flops comme la figure 22 permet de l'observer. En effet lorsqu'on désire calculer un seul flux, il est nécessaire de connaître les valeurs de u de 8 cases au total. En regroupant en blocs de 4x4 par exemple, un thread va calculer 32 flux (16 horizontaux et 16 verticaux) en utilisant 49 valeurs de u. Cela revient à diviser par 5 le nombre de valeur de u à aller chercher en mémoire par flux calculé. En implémentant cela, nous nous sommes rendu compte que bien que le temps nécessaire pour l'accès mémoire était réduit, l'utilisation moins importante des *threads* diminuait davantage nos performances. Notre choix s'est porté sur le calcul du flux vertical et horizontal rentrant dans une case par *thread* formant ainsi des "blocs" de 1x1.

Le tableau 4 indique le temps d'exécution en fonction de la taille de ces blocs pour 1000 itérations sur une grille de 512 par 512. on observe qu'il n'y a pas de réel gain de temps. De plus, le fait de ne plus utiliser Gauss-Seidel diminue la stabilité de nos simulations. Pour une même discrétisation spatiale, l'implémentation bloc-flux nécessite une contrainte sur $\tau$ 4 fois

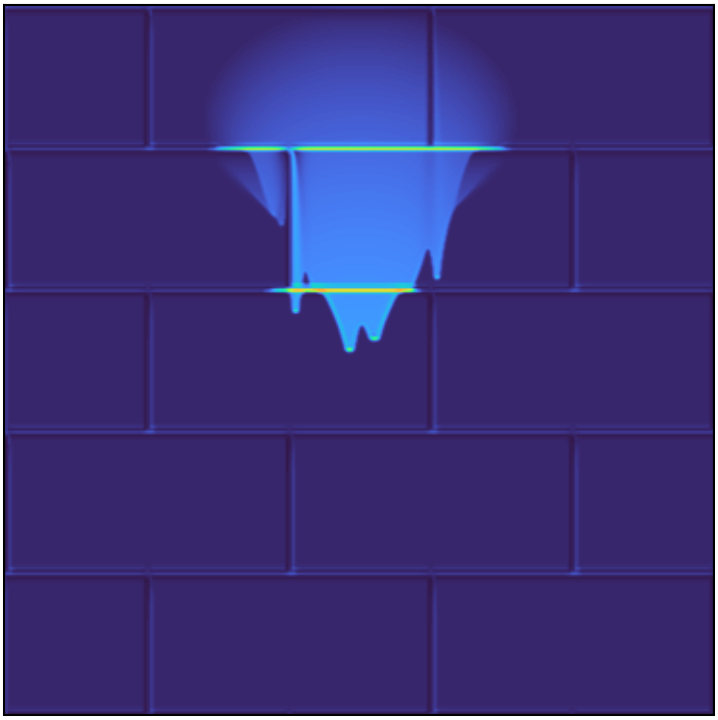


FIGURE 24 – Écoulement d'une Gaussienne sur des briques avec l'implémentation 3D simplifiée

plus petite. Il faut donc arriver à obtenir une simulation au moins 4 fois plus rapide pour y voir de l'intérêt. Nous avons donc une réelle amélioration des performances de la simulation mais cela n'est pas suffisant face à la baisse de stabilité.

| Implémentation | temps $[s]$ |
|---|---|
| Gauss-Seidel CPU | 2.35 |
| Gauss-Seidel GPU | 0.49 |
| Bloc-flux GPU | 0.14 |

TABLE 4 – Performance sur 1000 itérations pour différentes implémentations en 3D

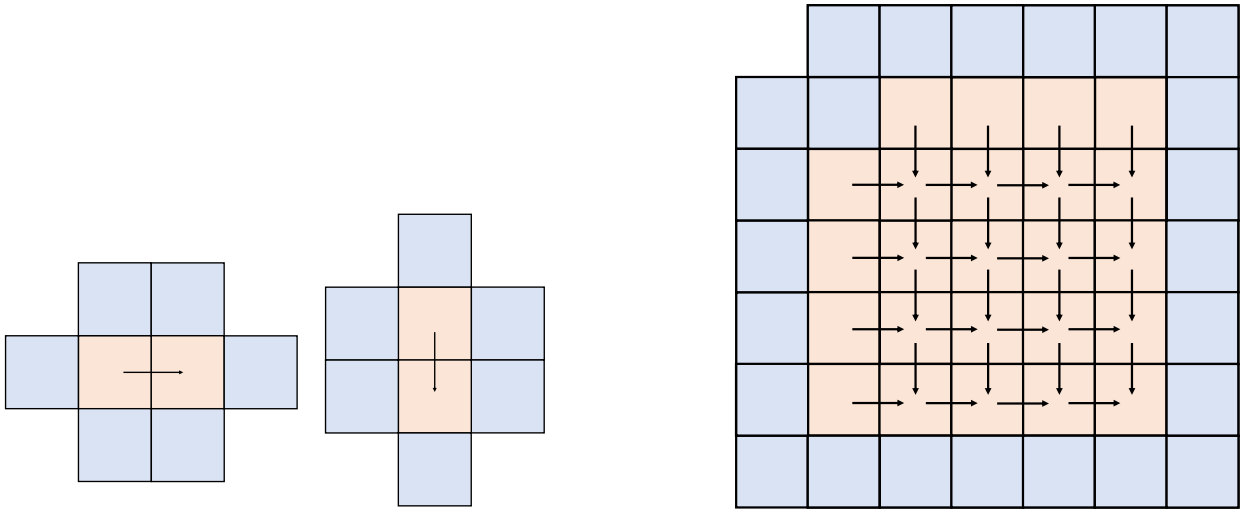


FIGURE 25 – Accès à la mémoire : à gauche, la méthode avec Gauss-Seidel 8 cases/flux. A droite, la méthode Bloc-flux 1.5 cases/flux

# 6 Visualisation

Dans ce travail, les figures d'écoulement du fluide ont été obtenues au moyen de *matplotlib* et de la *colormap* "turbo". Dans la visualisation en temps réel nous utilisons ce qui s'appelle un *shader* pour implémenter cette *colormap* avec OpenGL. Un *shader* est un petit programme faisant partie de la chaîne de traitement (ou *pipeline*) qu'OpenGL exécute pour calculer le rendu graphique des pixels. Plusieurs *shaders* sont nécessaires au bon fonctionnement d'OpenGL comme le *vertex shader* pour la création des points ou le *geometry shader* pour la géométrie, mais ici nous nous intéressons au *fragment shader* qui s'occupe de calculer la couleur finale du pixel. C'est dans ce programme que les effets visuels sont implémentés et où nous avons codé notre *colormap*.

Dans l'article [2], les auteurs implémentent les effets visuels de réfraction et de caustique du fluide grâce à la méthode décrite dans [18]. Cette méthode leur permet d'obtenir des simulations en temps réel plus réalistes comme à la figure 26.

Sortons du cadre des simulations en temps réel et intéressons-nous à la visualisation la plus réaliste possible. Les logiciels de rendu graphique comme Blender sont parfaits pour cet objectif. Ces logiciels utilisent un modèle 3D réalisé par l'utilisateur et lui permet de le placer dans un environnement 3D. La position et l'intensité des lumières sont définies par l'utilisateur ; Blender s'occupe alors de simuler cette lumière en prenant en compte ses multiples réflections, réfractions, ... sur les différents objets et en fonction de leur matériaux. Cela permet d'obtenir une visualisation très proche de la réalité mais nécessite un énorme coût en calcul.

Un choix judicieux des paramètres du modèle 41 et de son pas de temps, permet d'obtenir des sortes de vagues dans l'écoulement comme nous pouvons le voir à la figure 27. La même simulation rendue avec Blender pour de l'eau est présentée à côté. Lorsque la surface du fluide est localement plane, il n'est pas simple de percevoir le fluide mais lorsque des vagues se forment

FIGURE 26 – Écoulement d'eau sur une vitre, directement tiré de [2]

nous voyons les reflets de la lumière sur ceux-ci.

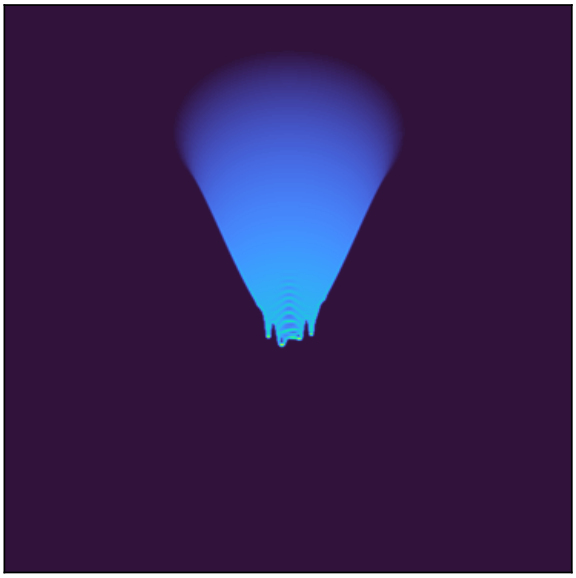


FIGURE 27 – Écoulement d'une gaussienne d'eau sur une vitre avec le modèle 41

# 7 Conclusion

L'étude de la physique du film et de ses équations adimensionnelles nous a permis de simplifier les équations de Navier-Stokes pour obtenir une expression en terme de la hauteur. En gardant uniquement les effets de gravité et de tension de surface nous obtenons un flux rampant comme désiré. Nous avons également observé l'intérêt de diminuer l'influence de la tension de surface pour avoir des simulations plus stables et plus rapides.

La méthode des volumes finis pour résoudre les équations s'est naturellement imposée. En effet, l'opérateur divergence dans l'équation du film peut être transformé en intégrale du flux. Pour obtenir ce flux, nous avons vu qu'il était possible de le faire au moyen d'un flux de gradient mais également plus simplement avec des différences finies. Le calcul du flux entre deux volumes prend la forme d'un problème de minimisation et grâce à la méthode de pas fractionnaires la fonction à minimiser est quadratique ce qui facilite la résolution du problème.

Cette dissipation négative contre-balance la dissipation présente naturellement dans la physique du film. En conséquence nous conservons bien plus longtemps les variations de courte longueur d'onde.

Les modèles finaux sont très locaux et ne nécessitent pas de résolution de système ; ces caractéristiques nous ont permis d'optimiser l'implémentation grâce au calcul parallèle. Notre division adéquate de la grille permet d'éviter toute concurrence entre les *threads*. Dans le cas d'écoulement sur une surface 3D, la principale difficulté consiste à accéder efficacement à la mémoire du GPU. Nous tentons de résoudre ce problème à l'aide de la mémoire partagée mais nous ne pouvons alors plus utiliser l'aspect Gauss-Seidel du modèle.

Pour répondre à la question centrale de ce travail, la méthode des pas fractionnaires nous a permis d'améliorer la stabilité du modèle sans devoir résoudre de système. Le modèle développé dans [2] et repris à l'équation 37 laissait espérer une bien meilleure stabilité que des modèles explicites (voire implicites). En travaillant sur cet article, nous nous sommes rendu compte que le modèle décrit dans l'article et le modèle implémenté ne coïncident pas. La réelle amélioration apportée par la méthode des pas fractionnaires n'est pas franche et au vu de la différence de stabilité entre un modèle explicite et implicite, elle n'est pas suffisante. Directement agir sur le facteur limitant, la tension de surface, se révèle plus efficace. En diminuant l'impact des effets de tension de surface dans l'équation du film, nous avons réussi à améliorer grandement la stabilité sans pour autant altérer de trop le rendu. Grâce au modèle de frottement 41 et en appliquant une "dissipation négative" en fonction de la discrétisation spatiale nous obtenons la meilleure stabilité et un pas de temps se comportant en $\mathcal{O}(n^2)$ au lieu de $\mathcal{O}(n^4)$.

Les résultats obtenus dans ce travail montrent qu'il est possible de simuler un écoulement ressemblant fortement à un film visqueux en temps réel. Bien que la physique du film n'est pas complètement respectée nous conservons les propriétés théoriques de ces écoulements. Dans l'optique de faire du graphisme, les simulations sont visuellement acceptables et représentent une avancée qui n'avait pas encore été atteinte.

Certaines pistes intéressantes n'ont pas été analysées dans ce travail par manque de temps.

Nous avons utilisé le langage CUDA dans l'implémentation mais il serait intéressant de directement utiliser OpenGL et stocker les valeurs de $u$ sous forme de *texture* et d'appliquer la mise à jour des $u$ au moyen d'un *shader*. Cette façon de faire permettrait de stocker les informations sur de la mémoire plus rapide tout en gardant l'aspect Gauss-Seidel. Nous aurions également pu nous intéresser à la résolution de système non-linéaire de manière parallèle et utiliser un modèle implicite. Il est possible que cela soit suffisamment rapide pour des simulations en temps réel dans le cas de petits maillages.

# 8 Annexe

# A Conditions à l'interface

L'équilibre des forces à l'interface sous forme tensorielle peut s'écrire :

$$\mathbf{n} \cdot \mathbf{T} \cdot \mathbf{n} = \sigma\kappa$$
$$\mathbf{n} \cdot \mathbf{T} \cdot \mathbf{t} = 0$$

Avec $\sigma$ la constante de tension de surface et $\kappa = \nabla \cdot n$, le rayon de courbure.
Nous considérons le tenseur de contrainte pour une fluide incompressible newtonien :

$$\mathbf{T} = \begin{bmatrix} 2\mu\partial_x u - p & \mu(\partial_x w + \partial_z u) \\ \mu(\partial_x w + \partial_z u) & 2\mu\partial_z w - p \end{bmatrix}$$

Le vecteur normal est quant à lui obtenu en prenant le vecteur unitaire perpendiculaire à la dérivée de h selon x :

$$(n_x, n_z) = \frac{-(\partial_x h, 1)}{(1 + (\partial_x h)^2)^{\frac{1}{2}}}$$

Et le vecteur tangent est par définition perpendiculaire à $\mathbf{n}$ :

$$(t_x, t_z) = \frac{(1, \partial_x h)}{(1 + (\partial_x h)^2)^{\frac{1}{2}}}$$

La première équation peut donc être calculée en utilisant les composantes de $\mathbf{T}$ et $\mathbf{n}$ définies ci-dessus.

$$-p + \frac{2\mu}{(1 + \partial_x h^2)} \left[\partial_x u \left(\partial_x h^2 - 1\right) - \partial_x h \left(\partial_z u + \partial_x w\right)\right] = \sigma \frac{\partial_{xx} h}{(1 + \partial_x h^2)^{\frac{3}{2}}}$$

où nous avons utilisé la propriété $w_z = -u_x$ pour simplifier l'équation.
Avec les variables adimensionnelles 4 et 5, nous retombons bien sur la seconde équation de 8 :

$$\frac{C^{-1}\epsilon^3 \partial_x^2 h}{\left[1 + \epsilon^2 \left(\partial_x h\right)^2\right]^{3/2}} = -p + \frac{2\epsilon^2}{\left[1 + \epsilon^2 \left(\partial_x h\right)^2\right]} \left\{\partial_x u \left[\epsilon^2 \left(\partial_x h\right)^2 - 1\right] - \partial_x h \left(\partial_z u + \epsilon^2 \partial_x w\right)\right\}$$

L'équation d'équilibre des forces dans la direction tangente à l'interface peut être développée :

$$4\partial_x h \partial_x u + \left(\partial_x h^2 - 1\right)\left(\partial_z u + \partial_x w\right) = 0$$

Nous pouvons alors adimensionnaliser cette équation avec les nombres adimensionnels définis en 4 et 5. Nous retombons alors sur la première équation de 8 :

$$0 = \left(\partial_z u + \epsilon^2 \partial_x w\right)\left[1 - \epsilon^2 \left(\partial_x h\right)^2\right] - 4\epsilon^2 \left(\partial_x h\right)\left(\partial_x u\right)$$

# B Substrat 3D

Nous supposons que la surface solide décrite par la variété $\Gamma$ sur laquelle le film s'écoule est inscrite dans $\mathbb{R}^3$ par la cartographie $\phi_0 : \Gamma \mapsto \mathbb{R}^3$. Cette cartographie est isométrique ce qui implique :

$$\langle v, w\rangle_\Gamma = \langle d\phi_0(v), d\phi_0(w)\rangle$$

Pour tout $v, w \in T\Gamma$, où $d\phi_0$ est la différentielle de $\phi_0$, $\langle\cdot,\cdot\rangle_\Gamma$ et $\langle\cdot,\cdot\rangle$ sont le produit interne de $T\Gamma$ et de $\mathbb{R}^3$ respectivement avec $|\cdot|_\Gamma = \sqrt{\langle\cdot,\cdot\rangle_\Gamma}$ la norme associée.
Pour une fonction $f : \Gamma \mapsto \mathbb{R}$, nous définissons la cartographie $\phi_f : \Gamma \mapsto \mathbb{R}^3$ as $\phi_f(x) := \phi_0(x) + f(x)n(x)$ avec $n(x)$ le vecteur unitaire normal à $\Gamma$. Si nous considérons un sous-ensemble $U \subseteq \Gamma$, $\phi_f(U)$ est alors son image selon $\phi_f$ et l'extrusion entre $\Gamma$ et $\phi_f$ est définie comme étant $\mathcal{X}_f(U) := \bigcup_{\xi=0}^{1} \phi_{\xi f}(U)$ comme représenté à la figure 4

Dans le cas d'un film, nous considérons la fonction $h(\cdot, t)$, l'interface liquide-solide correspond à $\phi_0(\Gamma)$ et l'interface air-liquide à $\phi_{\epsilon h}(\Gamma)$. Le domaine du film correspond donc à $\mathcal{X}_{\epsilon h}(\Gamma)$ et est caractérisé par la cartographie :

$$\phi : \Gamma \times \mathbb{R} \to \mathbb{R}^3; (x, \xi) \mapsto \phi_0(x) + \epsilon\xi n(x)$$

où $0 \leq \xi \leq h$, avec $h$ la hauteur adimensionnelle.
Nous pouvons alors écrire la différentielle de $\phi$ pour une variation de la position $\delta x$ et de a distance normale à la surface $\delta\xi$ : $d\phi(\delta x, \delta\xi) = d\phi_0(\Lambda\delta x) + \epsilon\delta\xi n(x),$ pour $(\delta x, \delta\xi) \in T\Gamma \times \mathbb{R}$ avec $\Lambda :=$ id $- \epsilon\xi S$ où S est l'opérateur de forme de la surface.

On peut alors exprimer une loi de conservation en terme de distribution de masse et de flux. La distribution de masse $u : \Gamma \mapsto \mathbb{R}$ représente le volume de liquide par unité de surface tel que :

$$\int_{\mathcal{X}_{\epsilon h}(U)} \mathrm{d}V = \epsilon \int_U u \, \mathrm{d}a$$

$$u = \int_0^h \lambda \mathrm{d}\xi \qquad \text{avec } \lambda := \det\Lambda$$

Si $\kappa_1$ et $\kappa_2$ correspondent aux principaux rayons de courbure de S alors $\lambda = \det(\mathrm{id} - \epsilon\xi S) = (1 - \epsilon\xi\kappa_1)(1 - \epsilon\xi\kappa_2) = 1 - \epsilon\xi H + \epsilon^2\xi^2 K$. Avec $H$ et $K$ respectivement le rayon de courbure moyen et Gaussien. En intégrant $\lambda$, on obtient alors

$$\begin{aligned} u &= h - \frac{\epsilon}{2}Hh^2 + \frac{\epsilon^2}{3}Kh^3 \\ \implies h &= u + \frac{\epsilon}{2}Hu^2 + \mathrm{O}\left(\epsilon^2\right) \end{aligned} \tag{43}$$

Le flux $F$ est un champ vectoriel tangent à $\Gamma$ que l'on peut définir par $\int_{\mathcal{X}_{\epsilon h}(\partial U)} \langle v, n\rangle \mathrm{d}a = \epsilon \int_{\partial U} \langle F, \nu\rangle_\Gamma \mathrm{d}l$ pour tout $U \subset \Gamma$, où $\partial U$ désigne la frontière de $U$, $n$ le vecteur normal unitaire de $\mathcal{X}_{\epsilon h}(\partial U)$ et $\nu$ le vecteur unitaire conormal de $\partial U$. En utilisant la cartographie $\phi$, on obtient alors $F = \int_0^h \mathrm{cof}\,\Lambda v_\Gamma \mathrm{d}\xi$. Avec $v_\Gamma$, la composante tangente à $\Gamma$ de la vitesse et $\mathrm{cof}\,\Lambda = \lambda\Lambda^{-T} = \lambda\Lambda^{-1}$, la matrice des cofacteurs. Avec $F$ et $u$ définis pour notre film, on peut alors utiliser l'équation du transport :

$$\frac{\partial u}{\partial t} + \mathrm{div}_\Gamma F = 0$$

On introduit aussi le flux $\mathfrak{F}(\xi) = \int_0^\xi \mathrm{cof}\,\Lambda v_\Gamma \mathrm{d}\xi$ tel que $F = \mathfrak{F}(h)$ et $v_\Gamma = \mathrm{cof}\,\Lambda^{-1}\frac{\partial\phi}{\partial\xi}$

# C L'énergie du film

L'énergie adimensionnelle totale du film peut s'exprimer comme :

$$\mathfrak{E}(u) = \mathfrak{E}_{\text{gravity}} + \mathfrak{E}_{\text{surface}} = \epsilon^{-1} \int_{\mathcal{X}_{\text{eh}}\,(\Gamma)} \zeta \mathbf{z} \mathrm{d}V + \epsilon^{-1} \int_{\phi_{\text{eh}}\,(\Gamma)} \mathrm{d}a$$

où $\zeta$ est le nombre de Bond, $\mathcal{X}_{\text{eh}}\,(\Gamma)$ est le volume du film délimité par $\gamma$ et $\phi_{\text{eh}}\,(\Gamma)$ et la surface du film à l'interface film/gaz.
Le premier terme est développé grâce aux outils de mapping développés précédemment :

$$\begin{aligned}
\frac{1}{\epsilon} \int_{\mathcal{X}_{\epsilon h}(\Gamma)} \zeta \mathbf{z} \mathrm{d}V &= \int_\Gamma \int_0^h \zeta \mathbf{z}(\phi(x,\xi)) \lambda(x,\xi) \mathrm{d}\xi \mathrm{d}a(x) \\
&= \int_\Gamma \int_0^h \zeta \left( \mathbf{z}\left(\phi_0(x)\right) + \epsilon\xi \left\langle \nabla \mathbf{z}\left(\phi_0(x)\right), n(x) \right\rangle + \mathrm{O}\left(\epsilon^2\right) \right) \lambda(x,\xi) \mathrm{d}\xi \mathrm{d}a(x) \\
&= \int_\Gamma \zeta \left( zu + \epsilon \frac{u^2}{2} \cos\theta + \mathrm{O}\left(\epsilon^2\right) \right) \mathrm{d}a
\end{aligned}$$

Avec $\cos\theta(x) := \left\langle \nabla \mathbf{z}\left(\phi_0(x)\right), n(x) \right\rangle = -\langle \hat{g}, n(x) \rangle$ et $z$ qui correspond au *pullback* (image réciproque ou tiré en arrière) de $\mathbf{z}$ sur $\Gamma$.

Pour le second terme, nous devons calculer l'aire $A[h]$ de l'interface libre du film $\phi_{\epsilon h}$. Nous allons utiliser le développement de Taylor suivant une variation normale de second ordre. Cette approximation de la surface prend donc la forme :

$$A[h] = A[0] + \epsilon A'[0](h) + \frac{\epsilon^2}{2} A''[0](h) + \mathrm{O}\left(\epsilon^3\right)$$

La variation de l'aire selon une variation normale est un résultat classique de géométrie différentielle :

$$\begin{aligned}
A'[0](h) &= -\int_\Gamma Hh \ \mathrm{d}a \\
A''[0](h) &= \int_\Gamma 2Kh^2 + |\mathrm{grad}_\Gamma\, h|_\Gamma^2 \ \mathrm{d}a \\
\implies \frac{1}{\epsilon} \int_{\phi_{eh}(\Gamma)} \mathrm{d}a &= \frac{1}{\epsilon} \int_\Gamma \mathrm{d}a + \int_\Gamma -Hu - \frac{\epsilon}{2} T u^2 + \frac{\epsilon}{2} |\mathrm{grad}_\Gamma\, u|_\Gamma^2 \ \mathrm{d}a + \mathrm{O}\left(\epsilon^2\right)
\end{aligned}$$

Le premier terme de cette expression est l'aire de $\Gamma$ et est constant. Il n'intervient donc pas dans la dynamique du film et peut être ignoré.

En additionnant les expressions développées de l'énergie adimensionnelle de tension de surface et de gravité, l'expression de l'énergie adimensionnelle devient donc :

$$E[u] = \int_\Gamma (\zeta z - H)u + \frac{\epsilon}{2}(\zeta \cos\theta - T)u^2 + \frac{\epsilon}{2} |\mathrm{grad}_\Gamma\, u|_\Gamma^2 \ \mathrm{d}a \quad + O(\epsilon^2)$$

Cela correspond bien à l'équation 20 et rappelons que dans le cas d'un écoulement sur surface plane, les rayons de courbure sont nuls et $\theta$ correspondant à l'angle entre $\nabla \boldsymbol{g}$ et $\hat{\boldsymbol{n}}$ vaut 90°. Cela donne donc pour le cas plan :

$$E[u] = \int_\Gamma \zeta z u + \frac{\epsilon}{2} |\mathrm{grad}_\Gamma\, u|_\Gamma^2 \ \mathrm{d}a \quad + O(\epsilon^2) \tag{44}$$

# D Dissipation visqueuse et mobilité

Le taux de dissipation visqueuse dans le film est donné par :

$$2\mu \int_{\mathcal{X}_{\epsilon h}(\Gamma)} \|D[v]\|_F^2 \, \mathrm{d}V$$

où $\mu$ est la viscosité du fluide, $D[v] = \frac{1}{2}\left(\nabla v + \nabla v^T\right)$ correspond au tenseur de taux de déformation et $\|\cdot\|_F^2$ est la norme de Frobenius. Nous pouvons alors développer l'expression de la dissipation visqueuse grâce au tenseur $D[v]$ exprimé avec des coordonnées curviligne et pour lequel on développe en expansion de $\epsilon$. Le terme lié à la contrainte de cisaillement due au flux laminaire domine et une fois adimensionnalisé avec le taux de dissipation caractéristique $\frac{\sigma^2 L \epsilon^3}{\mu}$ [10] obtient :

$$\mathfrak{g}(v,v) = \int_\Gamma \int_0^h \left| \frac{\partial v_\Gamma}{\partial \xi} + \epsilon S v_\Gamma \right|_\Gamma^2 \lambda \mathrm{d}\xi \mathrm{d}a + \mathrm{O}\left(\epsilon^2\right)$$

Cette expression de la dissipation visqueuse dépend de la vitesse mais pour convenir à notre problème de minimisation nous allons l'exprimer en fonction de $u$ et de $f$.

A partir de cette expression, nous pouvons réaliser une optimisation en considérant des flux partiels.
Considérons que le profil de vitesse dans la direction normale s'ajuste pour minimiser la dissipation. Cela nous permet d'exprimer la dissipation visqueuse en fonction de $F$ le flux total.
Rappelons que $\epsilon S = -\frac{\partial \Lambda}{\partial \xi}$, l'expression devient alors :

$$\begin{aligned}
\mathfrak{g}(v,v) &= \int_0^h \left| \frac{\partial v_\Gamma}{\partial \xi} - \frac{\partial \Lambda}{\partial \xi} v_\Gamma \right|_\Gamma^2 \lambda \mathrm{d}\xi \\
&= \int_0^h \lambda \left| e^{\Lambda} \frac{\partial}{\partial \xi}\left(e^{-\Lambda} v_\Gamma\right) \right|_\Gamma^2 \mathrm{d}\xi \\
&= \int_0^h \lambda \left| e^{\Lambda} \frac{\partial}{\partial \xi}\left(e^{-\Lambda} \operatorname{cof} \Lambda^{-1} \frac{\partial \mathfrak{F}}{\partial \xi}\right) \right|_\Gamma^2 \mathrm{d}\xi \\
&= \int_0^h \left\langle \frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial \mathfrak{F}}{\partial \xi}\right), A(\Lambda)\frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial \mathfrak{F}}{\partial \xi}\right) \right\rangle_\Gamma \mathrm{d}\xi
\end{aligned}$$

avec $A(\Lambda) := \lambda e^{2\Lambda}$ et $B(\Lambda) := e^{-\Lambda} \operatorname{cof} \Lambda^{-1}$

Nous pouvons intégrer cette expression par partie à deux reprises et pour deux flux partiels $\mathfrak{F}$ et $\tilde{\mathfrak{F}}$ :

$$\begin{aligned}
&\int_0^h \left\langle \frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial \tilde{\mathfrak{F}}}{\partial \xi}\right), A(\Lambda)\frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial \mathfrak{F}}{\partial \xi}\right) \right\rangle \mathrm{d}\xi \\
&= \left[ \left\langle B(\Lambda)\frac{\partial \tilde{\mathfrak{F}}}{\partial \xi}, A(\Lambda)\frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial \mathfrak{F}}{\partial \xi}\right) \right\rangle_\Gamma - \left\langle \tilde{\mathfrak{F}}, B(\Lambda)\frac{\partial}{\partial \xi}\left(A(\Lambda)\frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial \mathfrak{F}}{\partial \xi}\right)\right) \right\rangle_\Gamma \right]_0^h \\
&\quad + \int_0^h \left\langle \tilde{\mathfrak{F}}, \frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial}{\partial \xi}\left(A(\Lambda)\frac{\partial}{\partial \xi}\left(B(\Lambda)\frac{\partial \mathfrak{F}}{\partial \xi}\right)\right)\right) \right\rangle_\Gamma \mathrm{d}\xi.
\end{aligned}$$

Considérons alors le flux qui minimise la dissipation sous les conditions frontières comme étant $\mathfrak{F}$ et une variation de ce flux partiel $\mathfrak{F} + \delta\mathfrak{F}$. Nous pouvons également considérer le flux partiel $\tilde{\mathfrak{F}} = \delta\mathfrak{F}$. Or si $\mathfrak{F}$ est optimal, alors nous avons comme condition :

$$0 = \int_0^h \left\langle \frac{\partial}{\partial\xi} B(\Lambda) \frac{\partial \delta\mathfrak{F}}{\partial\xi}, A(\Lambda) \frac{\partial}{\partial\xi} B(\Lambda) \frac{\partial \mathfrak{F}}{\partial\xi} \right\rangle_\Gamma \mathrm{d}\xi \tag{45}$$

L'expression précédente peut être exprimée pour tout flux partiel $\delta\mathfrak{F}$ et les termes à la frontière se simplifient :

$$0 = \int_0^h \left\langle \delta\mathfrak{F}, \frac{\partial}{\partial\xi} \left( B(\Lambda) \frac{\partial}{\partial\xi} \left( A(\Lambda) \frac{\partial}{\partial\xi} \left( B(\Lambda) \frac{\partial \mathfrak{F}}{\partial\xi} \right) \right) \right) \right\rangle_\Gamma \mathrm{d}\xi$$

Nous pouvons définir $f$ tel que

$$f := -\left( B(\Lambda) \frac{\partial}{\partial\xi} \left( A(\Lambda) \frac{\partial}{\partial\xi} \left( B(\Lambda) \frac{\partial \mathfrak{F}}{\partial\xi} \right) \right) \right) \tag{46}$$

En intégrant trois fois, l'expression de $\mathfrak{F}$ devient :

$$\mathfrak{F} = \left( \int_0^\xi B\left(\Lambda_c\right)^{-1} \int_0^c A\left(\Lambda_b\right)^{-1} \int_b^h B\left(\Lambda_a\right)^{-1} \, \mathrm{d}a \, \mathrm{d}b \, \mathrm{d}c \right) f$$

où a,b,c sont des "dummies variables" et où $\Lambda_a$ désigne la valeur de *Lambda* en $\xi = a$. Les équations 45 et 46 ainsi que les conditions frontières permettent de simplifier l'expression du taux de dissipation pour le cas $\tilde{\mathfrak{F}} = \mathfrak{F}$ et nous obtenons :

$$-\left\langle \mathfrak{F}(h), B(\Lambda) \frac{\partial}{\partial\xi} \left( A(\Lambda) \frac{\partial}{\partial\xi} \left( B(\Lambda) \frac{\partial \mathfrak{F}}{\partial\xi} \right) \right) \right\rangle_\Gamma = -\langle \mathfrak{F}(h), -f \rangle_\Gamma = \langle \mathfrak{M}(h) f, f \rangle_\Gamma$$

Avec $F = \mathfrak{F}(h) = \mathfrak{M}(h) f$ et donc $\mathfrak{M}(h)$ peut être définie comme :

$$\mathfrak{M}(h) := \int_0^h B\left(\Lambda_c\right)^{-1} \left( \int_0^c A\left(\Lambda_b\right)^{-1} \left( \int_b^h B\left(\Lambda_a\right)^{-1} \, \mathrm{d}a \right) \mathrm{d}b \right) \mathrm{d}c$$

Ce développement a permis d'obtenir l'expression de la dissipation non plus en fonction de la vitesse mais bien du flux f. Nous pouvons tout de même encore simplifier en développant $\mathfrak{M}(h)$ en puissance de $\epsilon$ et se limiter à l'ordre $\mathrm{O}(\epsilon)$ :

$$\begin{aligned} A(\Lambda)^{-1} &= \lambda^{-1} e^{-2\Lambda} = e^{-2}(\mathrm{id} + \epsilon\xi(H\mathrm{id} + 2S)) + \mathrm{O}\left(\epsilon^2\right) \\ B(\Lambda)^{-1} &= e^{\Lambda} \operatorname{cof} \Lambda = e(\mathrm{id} - \epsilon\xi H\mathrm{id}) + \mathrm{O}\left(\epsilon^2\right) \\ \mathfrak{M}(h) &= \frac{h^3}{3} + \frac{\epsilon}{6} h^4 (S - 2H\mathrm{id}) + \mathrm{O}\left(\epsilon^2\right) \end{aligned}$$

Finalement, il est possible de remplacer $h$ par $u$ grâce à l'expression 43 et nous obtenons :

$$M(u) := \frac{u^3}{3} + \frac{\epsilon}{6} u^4 (H \ \mathrm{id} \ + S)$$

Cela permet de conclure avec l'expression approchée de la dissipation visqueuse :

$$g_u(f, f) := \int_\Gamma \langle f, M(u) f \rangle_\Gamma \mathrm{d}a$$

tel que $g_u(f, f) = \mathfrak{g}(v, v) + \mathrm{O}\left(\epsilon^2\right)$. Cela correspond bien à l'expression 22.

# Références


[1] O. Reynolds, "On theory of lubrification and its application to mr. beauchamp tower's experiments," *Philosophical Transactions of the Royal Society of London*, vol. 177, pp. 157–234, 1886.

[2] O. Vantzos, S. Raz, and M. Ben-Chen, "Real-time viscous thin films," *ACM Transaction on Graphics*, 2018.

[3] O. Alexander, D. Stephen, and G. Bankoff, "Long-scale evolution of thin liquid films," *Reviews of Modern Physics*, vol. 69, pp. 157–234, 1997.

[4] C. Wang, "Thin film flowing down a curved surface," *Journal of applied mathematics and physics*, vol. 35, 1984.

[5] Roy, Roberts, and Simpson, "A lubrification model of coating flows over a curved substrate in space," *Journal of Fluid Mechanics*, vol. 454, pp. 235–261, 2002.

[6] L. Wiryanto, "A finite difference method for a thin film model," *Proceedings of international conference on Mechanical, Electrical and Medical intelligent system 2017*, 2017.

[7] L. Zhornitskaya and A. L. Bertozzi, "Positivity-preserving numerical schemes for lubrication-type equations," *SIAM Journal on Numerical Analysis*, vol. 37, pp. 523–555, 2000.

[8] G. Grün and M. Rumpf, "Nonnegativity preserving convergent schemes for the thin film equation," *Numerische Mathematik*, vol. 87, no. 1, pp. 113–152, 2000.

[9] C. Ma, "Finite element simulations of thin films flowing down planes or cylinders," *Progress in Computational fluid dynamics an international journal*, 2019.

[10] O. Vantzos and M. Rumpf, "Numerical gradient flow discretization of viscous thin films on curved geometries," *Mathematical models  methods in applied sciences 23 n°05*, 2013.

[11] R. Keunings, "Linma2720 : Mathematical modeling of physical systems," 2020.

[12] P.-G. De Gennes and F. Brochard-Wyart, *Gouttes, bulles, perles et ondes*. Belin, 2015.

[13] R. J. LeVeque *et al.*, *Finite volume methods for hyperbolic problems*. Cambridge university press, 2002, vol. 31.

[14] G. Grün, M. Lenz, and M. Rumpf, "A finite volume scheme for surfactant driven thin film flow," 2002.

[15] J. B. Perot, "An analysis of the fractional step method," *Journal of Computational Physics*, vol. 108, no. 1, pp. 51–58, 1993.

[16] R. Temam, "Sur la stabilité et la convergence de la méthode des pas fractionnaires," *Annali di Matematica pura ed applicata*, vol. 79, no. 1, pp. 191–379, 1968.

[17] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven, "Nodal discontinuous galerkin methods on graphics processors," *Journal of Computational Physics*, vol. 228, no. 21, pp. 7863–7882, 2009.

[18] C. Yuksel and J. Keyser, "Fast real-time caustics from height fields," *The Visual Computer*, vol. 25, no. 5, pp. 559–564, 2009.

**UNIVERSITÉ CATHOLIQUE DE LOUVAIN**
**École polytechnique de Louvain**
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**