

Contents

I Inria Bordeaux - Modèles d'adaptation de mailages	4
1 Introduction	5
2 Le modèle	6
2.1 Description du problème	6
2.2 Discrétisation en éléments finis	7
2.3 Quelques éléments pour le calcul de K	8
2.4 Résolution du système linéaire	10
3 Application à la fonction Level Set	11
3.1 Construction de la fonction à adapter (avec ω donné par l'expression (1))	12
3.1.1 Choix des paramètres	14
3.2 Calcul des tailles désirées	16
3.2.1 Résultats	16
3.3 Relaxation et qualité du maillage	17
3.4 Adaptation 3D	18
4 Application à l'adaptation physique	21
4.1 Calcul des tailles désirées	22
4.2 Couplage entre adaptation Level Set et adaptation physique .	23
4.3 Résultats	24
5 Adaptation à un cas non stationnaire	28
II Meric / Inria Chile - Méthodes de Décomposition de Domaine	30
1 Introduction	31
2 L'équation de KdV	33
2.1 Le modèle	33
2.2 Discrétisation	34
2.2.1 Premier pas du <i>splitting</i> - advection	34
2.2.2 Deuxième pas du <i>splitting</i> - dispersion	36
2.2.3 Choix de la taille de maille	37

2.3	Analyse d'échelle	37
2.3.1	Caractérisation de la non linéarité	38
2.3.2	Caractérisation de la dispersion	38
2.4	Le critère pour choisir une solution initiale appropriée	39
2.4.1	Choix de la longueur d'onde	39
2.4.2	Choix de l'amplitude de l'onde	40
2.4.3	Résumé	40
2.5	Tests numériques	41
3	Les équations de Serre	42
3.1	Le modèle	42
3.1.1	Les équations de Serre dans les variables (h, hu)	43
3.2	Discrétisation	44
3.2.1	Première système d'équations (pas d'advection)	45
3.2.2	Deuxième système d'équations (pas de dispersion)	46
3.3	Tests numériques	47
3.3.1	Description de la solution initiale	47
3.3.2	Results	48
4	Premier étude des conditions aux limites transparentes	49
4.1	Introduction et quelques exemples de motivation	49
4.2	Optimisation de conditions aux limites de Robin pour simuler des TBCs	54
4.2.1	Conditions aux limites de Robin jusqu'à la dérivée première	54
4.2.2	Conditions aux limites de Robin jusqu'à la dérivée seconde	57
4.2.3	Conclusion partielle	58
5	Conditions aux limites transparentes approximées pour l'équation de dispersion	58
5.1	Approximation des TBCs utilisant des polynômes constants . .	60
5.1.1	Tests de validation de l'approximation	61
5.2	Approximation des TBCs utilisant un polynôme linéaire . . .	63
5.2.1	Tests de validation de l'approximation	65
5.3	Conclusion partielle	65
6	Application des TBCs approximées à une Méthode de Décomposition de Domaine	66
6.1	The Schwarz Methods	66
6.2	ASM avec des TBCs approximées pour l'équation de dispersion	68
6.3	Discrétisation du problème et erreur dans la solution convergée	69

6.3.1	Numerical verification of the error	72
6.3.2	Corrections pour les TBCs approximées	73
6.4	Optimisation des IBCs (vitesse de convergence)	75
6.4.1	Tests variant l'instant initial et la position de l'interface	76
6.4.2	Tests variant Δt and Δx	79
6.5	Partial conclusion	81

Part I

Inria Bordeaux - Modèles d'adaptation de maillages

1 Introduction

L'objectif de l'adaptation de maillages développée dans ce stage est de modifier la position de ses noeuds par rapport à une fonction donnée, ou à une solution d'un problème physique, de façon qu'on puisse avoir un raffinement variable au long du domaine et qui représente bien la présence de fortes gradients, surfaces, etc., sans modifier le nombre de points ni la connectivité du maillage.

Dans ce rapport, on va présenter dans un premier moment l'application de l'algorithme à l'adaptation à des fonctions Level Set, à fin d'obtenir une bonne représentation d'un objet. Cette fonction, définie pour tout point du maillage, est la distance signée entre le point et l'objet, où la signe indique s'il est à son intérieur ou extérieur [12]. Ainsi, la ligne de niveau 0 de la fonction Level Set représente la surface de l'objet, et on utilisera ce fait pour orienter l'adaptation du maillage.

Ensuite, on résoudra des problèmes de la mécanique des fluides sur les maillages adaptés et, avec les résultats obtenus, on fera des nouvelles adaptations, mais cette fois-ci en utilisant au même temps la fonction Level Set et la solution physique du problème. Avec cette procédure, on sera capable d'obtenir des maillages encore plus appropriées au calcul envisagé.

Le rapport est organisé de la façon suivante : dans la section 2, on va d'abord présenter, de façon plus générale, le modèle utilisé pour l'adaptation de maillages, selon la formulation développée par [2], et des détails concernant son implémentation. L'application du modèle à l'adaptation à des fonctions Level Set sera décrite dans la section 3, avec quelques exemples de tests réalisés et une indication des paramètres qui ont produit les meilleurs résultats. Le couplage avec l'adaptation physique et les résultats obtenus sont présentés dans la section 4. Enfin, dans la section 5, on utilise le modèle pour adapter le maillage à des objets en mouvement. Par ailleurs, on remarque que la plupart du contenu de ce rapport se réfère à des cas 2D; pourtant, nous présentons aussi quelques résultats pour des exemples 3D.

Les exemples présentés dans ce rapport ne sont qu'une petite partie de l'ensemble des tests réalisées au cours de ce stage. Ces tests ont eu une grande importance pour valider, corriger et développer le modèle, tester la bibliothèque et les plusieurs parties du code, et trouver les paramètres et stratégies d'adaptation qui nous permettent d'obtenir les meilleurs résultats en tenant compte des objectifs décrits ci-dessus. La section ?? présente une liste des rapports rédigées pour décrire et présenter ces tests.

2 Le modèle

`<sec:modele>`

2.1 Description du problème

On va faire la distinction entre deux domaines :

- **Domaine physique ou réel** ($\mathbf{x} = (x, y)$) : domaine déformable, noté Ω ;
- **Domaine computationnel ou de référence** ($\boldsymbol{\xi} = (\xi, \eta)$) : domaine fixé, noté $\Omega_{\boldsymbol{\xi}}$

On cherche une fonction

$$\mathbf{x} = \mathbf{x}(\boldsymbol{\xi})$$

Dans le modèle utilisé, on va considérer que la position \mathbf{x} des noeuds du maillage est régie par l'équation

$$\nabla_{\xi\eta} \cdot (\omega \nabla_{\xi\eta} \mathbf{x}) = 0$$

où $\nabla_{\xi\eta}$ est le gradient par rapport aux coordonnées de référence et ω est une fonction de \mathbf{x} qui contient l'information qui déterminera le mouvement des noeuds. Dans le travail développé au cours du stage, on a implémenté deux modèles différents pour le calcul de cette fonction :

1. Dans un première moment, on a implémenté utilisé dans [2] : en supposant qu'on fait l'adaptation par rapport à une fonction u , ω est donné par l'expression

$$\omega(\mathbf{x}) = \sqrt{1 + \alpha \|\nabla_{\xi\eta} u(\mathbf{x})\| + \beta \|H(u)(\mathbf{x})\|}$$

$H(u)$ est le hessien de u , et α et β sont des paramètres choisis par l'utilisateur. Pour que ce choix soit moins dépendant du problème, on va toujours considérer les gradients et le hessiens normalisés.

2. On a ensuite utilisé un modèle où on fournit directement à chaque noeud i la taille de maille h_{des} qu'on désire, selon la formulation présentée par [3] :

$$\omega(\mathbf{x}) = \frac{1}{h_{des}(\mathbf{x})} \quad (1) \boxed{\text{eq:omega2}}$$

La façon dont on calcule les tailles désirées dépende du type d'adaptation qu'on fait (adaptation à une fonction Level Set ou à une solution physique), comme on précisera dans les sections suivantes de ce rapport.

Pour que le problème soit bien posé, il faut définir des conditions aux bords appropriées :

$$\begin{cases} \nabla_{\xi\eta} \cdot (\omega \nabla_{\xi\eta} \mathbf{x}) = 0 & \text{dans } \Omega_{\boldsymbol{\xi}} \\ \mathbf{x} = \mathbf{g} & \text{sur } \partial\Omega_{\boldsymbol{\xi}}^D \\ \nabla_{\xi\eta} \mathbf{x} \cdot \mathbf{n} = 0 & \text{sur } \partial\Omega_{\boldsymbol{\xi}}^N \end{cases} \quad (2) \boxed{\text{eq:systeme}}$$

Ainsi, les conditions aux limites utilisées sont de deux types, de Dirichlet et de Neumann, définies sur des parties disjointes du bord, ($\partial\Omega_{\boldsymbol{\xi}}^D$ et $\partial\Omega_{\boldsymbol{\xi}}^N$, respectivement). Pour les conditions de Dirichlet, on impose $\mathbf{g} = \boldsymbol{\xi}$, indiquant que les points de $\partial\Omega_{\boldsymbol{\xi}}^D$ ne doivent pas bouger (ce qu'on impose, par exemple, dans les coins d'un domaine rectangulaire). En revanche, les conditions de Neumann (imposées par exemple dans les côtés du domaine), indiquent que les points de $\partial\Omega_{\boldsymbol{\xi}}^N$ doivent glisser sur le bord, i.e., bouger parallèlement au bord (de façon que, dans notre exemple, le domaine reste toujours rectangulaire).

La formulation faible du problème, avec une fonction test $v \in H_1^0(\Omega_{\boldsymbol{\xi}})$, s'écrit comme

$$0 = \int_{\Omega_{\boldsymbol{\xi}}} v \nabla_{\xi\eta} \cdot (\omega \nabla_{\xi\eta} \mathbf{x}) d\boldsymbol{\xi} = - \int_{\Omega_{\boldsymbol{\xi}}} \omega \nabla_{\xi\eta} v \cdot \nabla_{\xi\eta} \mathbf{x} d\boldsymbol{\xi} + \int_{\partial\Omega_{\boldsymbol{\xi}}} v \omega \nabla_{\xi\eta} \mathbf{x} \cdot \mathbf{n} ds \quad (3) \boxed{\text{eq:faible}}$$

Les conditions aux bords définies en (2) annulent le dernier terme en (3), et on arrive ainsi à

$$\int_{\Omega_{\boldsymbol{\xi}}} \omega \nabla_{\xi\eta} v \cdot \nabla_{\xi\eta} \mathbf{x} d\boldsymbol{\xi} = 0$$

2.2 Discrétisation en éléments finis

On utilise une discrétisation en élément finis P1, avec une base de fonctions $\{\varphi_i\}$ définis pour chacun des N noeuds du maillage. Ainsi, \mathbf{x} et la fonction test $v \in H_0^1(\Omega)$ se discrétisent sous la forme

$$\begin{aligned}\mathbf{x}_h &= \sum_{j=1}^N \mathbf{x}_j \varphi_j = \sum_{j=1}^N \begin{pmatrix} x_j \\ y_j \end{pmatrix} \varphi_j \\ v_h &= \sum_{i=1}^N v_i \varphi_i\end{aligned}\tag{4} \boxed{\text{eq:u_disc}}$$

En utilisant (4) dans (3), on arrive à

$$\sum_{j=1}^N \left[\left(\int_{\Omega_\xi^h} \omega \nabla_{\xi\eta} \varphi_i \cdot \nabla_{\xi\eta} \varphi_j d\xi \right) \mathbf{x}_j \right] = 0 \quad \forall i \in \{1, \dots, N\}$$

On voit ainsi que la discréétisation en éléments finis se ramène à la résolution de deux systèmes linéaires indépendants et de la même forme, un pour les coefficients $\{x_j\}$ et l'autre pour $\{y_j\}$:

$$\begin{cases} Kx = 0 \\ Ky = 0 \end{cases} \tag{5} \boxed{\text{eq:syst_final}}$$

Les éléments de la matrice K ont la forme

$$k_{ij} = \int_{\Omega_\xi^h} \omega \nabla_{\xi\eta} \varphi_i \cdot \nabla_{\xi\eta} \varphi_j d\xi \tag{6} \{?\}$$

2.3 Quelques éléments pour le calcul de K

`(subsec:calculK)` Le calcul des éléments de K est fait de la manière usuelle, par assemblage des contributions des éléments pour les coefficients k_{ij} . On précise dans la liste suivante quelques détails de l'implémentation de ce calcul :

- Le gradient de φ_i sur l'élément T sera donnée par $(\nabla_{\xi\eta} \varphi_i)^T = \frac{\mathbf{n}_i^T}{d|T|}$, où $|T|$ est l'aire de T , $d = 2$ est le nombre de dimensions spatiales et \mathbf{n}_i^T est le vecteur entrant à T , dans le côté opposé à i et de norme égale à la longueur de ce côté [9].
- La fonction ω sera considérée constante dans chaque élément T et égale à la moyenne ω^T de sa valeur sur les noeuds de T .
- Comme l'intégrale est calculée dans le domaine de référence, on utilisera toujours les vecteurs normaux et les aires du maillage initial. La fonction ω , en revanche, sera actualisée pour exprimer l'évolution du maillage, et son calcul sera faite en utilisant la solution interpolée à la fin de chaque itération.

On a, ainsi :

$$\begin{aligned}
k_{ij} &= \int_{\Omega^h} \omega \nabla_{\xi\eta} \varphi_i \cdot \nabla_{\xi\eta} \varphi_j d\boldsymbol{\xi} = \sum_{T \ni i} \int_T \omega \nabla_{\xi\eta} \varphi_i \cdot \nabla_{\xi\eta} \varphi_j d\boldsymbol{\xi} = \\
&= \sum_{T \ni i} |T| \omega^T \frac{\mathbf{n}_i^T \cdot \mathbf{n}_j^T}{4|T|^2} = \sum_{T \ni i} \omega^T \frac{\mathbf{n}_i^T \cdot \mathbf{n}_j^T}{4|T|} \tag{7} \boxed{\text{eq:calculK_2d}}
\end{aligned}$$

Remarque 1 : comme montre le développement du modèle fait ci-dessus, le calcul de l'intégrale qui donne les éléments de la matrice K est faite **toujours sur le maillage de référence**, ce qui implique que, dans (7), on utilisera toujours les mêmes vecteurs normaux et les mêmes aires pour le calcul discret. Néanmoins, **la fonction ω est actualisée à chaque itération dans le maillage physique**, afin de bien indiquer que, dans ses nouvelles positions, les noeuds sont dans des régions de raffinement ou pas.

Remarque 2 : Pour le calcul des éléments de K , on a profité du fait que, dans la méthode d'éléments finis P1, on a $k_{ij} = 0$ si les noeuds i et j n'appartient pas au même élément. Ainsi, la matrice est creuse, ce que nous a motivé à le stocker avec une structure adaptée (au lieu d'allouer une matrice de taille $N \times N$), afin de réduire la consommation d'espace mémoire, comme décrit ci-dessous :

- On identifie, d'abord, le nombre maximal de voisins d'un noeud du maillage, N_{neig} , sous la convention qu'un noeud est toujours voisin de lui-même;
- On alloue deux vecteurs :
 - Un vecteur d'entiers, A , de taille $(N_{neig} + 1).(N + 1)$;
 - Un vecteur de doubles, \tilde{K} , de taille $N_{neig}.(N + 1)$;
- Le vecteur A contient les index des voisins de chaque noeud. Par convention,
 - $A_{(N_{neig}+1).i} = N_{neig}^i$ contient le nombre de voisins de i ;
 - $A_{(N_{neig}+1).i+1}$ jusqu'à $A_{(N_{neig}+1).i+N_{neig}^i}$ contiennent les index des voisins de i (pour commodité, on garde toujours $A_{(N_{neig}+1).i+N_{neig}^i} = i$, mais cela n'est pas nécessaire)
- Le vecteur \tilde{K} contient les éléments de la matrice :
 - Si $A_{(N_{neig}+1).i+z} = j$, alors $\tilde{K}_{N_{neig}.i+z-1} = K_{ij}$

Le stockage ici proposé n'est pas optimal, car, dans des maillages non structurés, les noeuds n'ont pas tous le même nombre de voisins. Par ailleurs, les $N_{neig} + 1$ premier éléments de A et les N_{neig} premiers éléments de \tilde{K} ne sont pas utilisés, pour être en concordance avec les conventions d'index utilisés dans la bibliothèque MMG. Néanmoins, par rapport au stockage de la matrice complète, on a vérifié des très grandes avantages :

- Par exemple, dans un test avec environ 28000 noeuds et 165 mille éléments non nulles dans la matrice K , on est passé d'un stockage de 783 million de doubles (soit 0.02% d'utilisation) à un stockage de 308 mille doubles (soit 54% d'utilisation) et 336 mille entiers, avec un considérable gain en temps d'exécution.
- On a observé aussi des gains en temps d'exécution du programme. Dans la formulation originale, sans garder les index des voisins de chaque noeud, il fallait appeler une fonction qui retourne la liste de voisins. Cet appel était fait une fois par itération pour chaque noeud, lors de la résolution du système linéaire.

2.4 Résolution du système linéaire

(subsec:jacobi) Le système linéaire (5) est résolu de façon itérative, avec la méthode de Jacobi. Dans les tests, on utilise en général un nombre petit d'itérations (dix ou vingt), ce qui donne des bons résultats pour un temps de calcul raisonnable. La solution est calculée en terme des déplacements $\boldsymbol{\delta} = \mathbf{x} - \boldsymbol{\xi}$. Ainsi, on réécrit le système linéaire (5) sous la forme

$$Kx = K(x - \delta + \delta) = 0 \longrightarrow K\delta = -K\xi$$

où $\boldsymbol{\delta} = \mathbf{x} - \boldsymbol{\xi}$ est le déplacement des points. Ainsi, $\forall i \in \{1, \dots, N\}$:

$$k_{ii}\delta_i^{[n+1]} = - \sum_{j=1, j \neq i}^N k_{ij}\delta_j^{[n]} - \sum_{j=1}^N k_{ij}\xi_j = - \sum_{j=1}^N k_{ij}(\xi_j + \delta_j^{[n]}) + k_{ii}\delta_i^{[n]}$$

Donc, finalement,

$$\delta_i^{[n+1]} = \delta_i^{[n]} - \frac{1}{k_{ii}} \sum_{j=1}^N k_{ij}(x_j^{[n]})$$

Remarque : avant d'actualiser la position des points, il faut vérifier si le déplacement calculé ne conduit pas à un croisement des noeuds. Pour faire

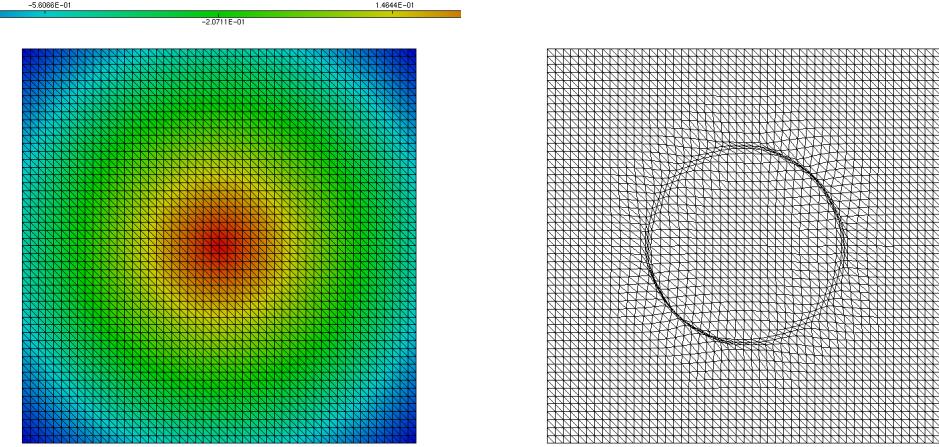
dans cette vérification, on calculé les aires signés des éléments (i.e., en considérant l'ordre des noeuds). Si nécessaire, on relaxe le déplacement :

$$\boldsymbol{x}^{[n+1]} = \boldsymbol{x}^{[n]} + \theta \left(\boldsymbol{\xi} + \boldsymbol{\delta}^{[n+1]} - \boldsymbol{x}^{[n]} \right)$$

avec $\theta \in [0, 1]$. La relaxation est ainsi appliquée sur la différence entre deux positions successives, non pas sur le déplacement par rapport à la position initiale, car dans ce cas-ci on peut avoir de "retours en arrière" des points du maillage.

3 Application à la fonction Level Set

`(sec:application)` Le modèle décrit dans la section précédente sera utilisé pour adapter le maillage par rapport à une fonction Level Set (LS). Cette fonction définit une distance signée entre chaque point du maillage et la surface qu'on représente: si le point \boldsymbol{x} est à l'extérieur de la surface, $LS(\boldsymbol{x}) > 0$; s'il est à l'intérieur, $LS(\boldsymbol{x}) < 0$; et si il est sur l'interface, $LS(\boldsymbol{x}) = 0$. On cherche alors à bouger les points pour avoir une meilleure représentation de la ligne de niveau 0 de la fonction Level Set, i.e., de la surface de l'objet [12]. Pour illustrer notre objectif, on présente dans la figure 1 un exemple d'adaptation à une fonction Level Set :



(a) Fonction Level Set représentée sur le maillage initial (b) Maillage adapté à la fonction Level Set

Figure 1: Exemple d'adaptation à la fonction Level Set

`(fig:exLS)`

3.1 Construction de la fonction à adapter (avec ω donné par l'expression (1))

Le calcul de ω à partir de l'expression (1) est fait en utilisant le gradient et le hessien de la fonction à adapter. Pourtant, dans le cas de la fonction Level Set, l'adaptation ne sera pas faite en considérant $u = LS$, parce qu'on n'aurait pas de forts gradients, en tenant compte de la lisseté de cette fonction. Ainsi, ω ne serait pas capable d'exprimer le raffinement désiré dans chaque partie du maillage. De cette façon, à partir de la fonction Level Set, on va construire une nouvelle fonction qui ait un très fort gradient sur les bords de l'objet.

Idéalement, on construirait une fonction avec un saut, valant 1 à l'extérieur et -1 à l'extérieur de l'objet, ce que fournit des très bons raffinements sur la ligne de niveau 0 de la fonction Level Set. Néanmoins, dans la résolution numérique de problèmes de la mécanique des fluides, il est intéressant d'avoir aussi un bon raffinement du maillage sur une couche autour de la surface (la couche limite de l'écoulement), où l'interaction fluide-objet n'est pas négligeable. En effet, pour avoir un calcul précis, un maillage approprié sur la couche limite est requis par la plupart des schémas numériques et logiciels pour les problèmes de fluides [20]. Ainsi, on a cherché une fonction plus lisse.

Dans un premier moment, on a défini une fonction avec une variation linéaire entre les deux étages (figure 3). Néanmoins, les tests réalisées ont fourni des résultats de mauvaise qualité : étant le gradient constant à l'intérieur de la pente, les points dans cette région (y compris les points les proches du bord de l'objet) n'avaient pas la tendance de bouger, au contraire des points proches des bords de la pente, dû à la discontinuité du gradient de u . Par conséquent, l'adaptation produisait une maillage avec deux petites couches raffinées, qui ne représentaient pas la surface de l'objet (figure 2).

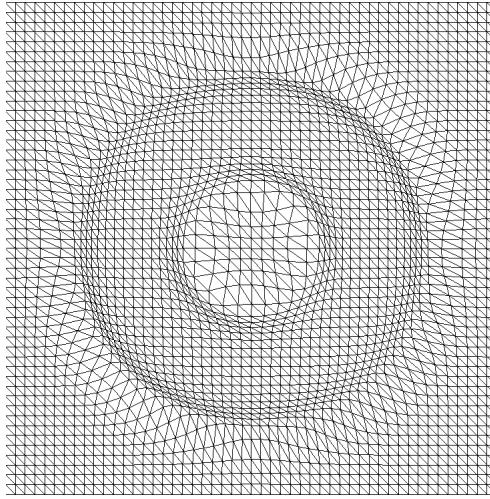


Figure 2: Adaptation à une fonction Level Set modifié avec une variation linéaire
 (fig:LSlin) On a ainsi cherché une fonction dont le gradient est plus variable et continue afin de réduire cet effet. On a obtenu des bons résultats en utilisant

$$u(\mathbf{x}) = \frac{1}{1.1.107} \operatorname{atan}\left(\frac{2LS(\mathbf{x})}{\delta}\right) \quad (8) \quad \text{eq:atan}$$

étant 2δ la taille de la couche, choisie par l'utilisateur. Cette fonction est représentée dans la figure 3.

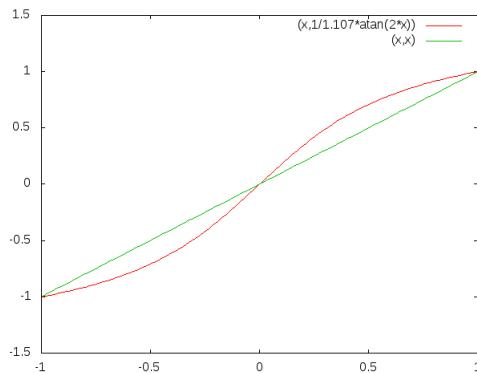


Figure 3: Comparaison des options pour la lissage de la fonction à adapter
 (fig:atan) On peut jouer aussi avec le paramètre multiplicateur de $LS(\mathbf{x})/\delta$ à l'intérieur de la fonction tangent. Avec 1, par exemple, on obtient un résultat plus proche de la pente linéaire; avec 5 ou 10, la fonction devient plus discontinue et la couche de raffinement est plus petite. On restera ainsi avec l'option donnée par (8).

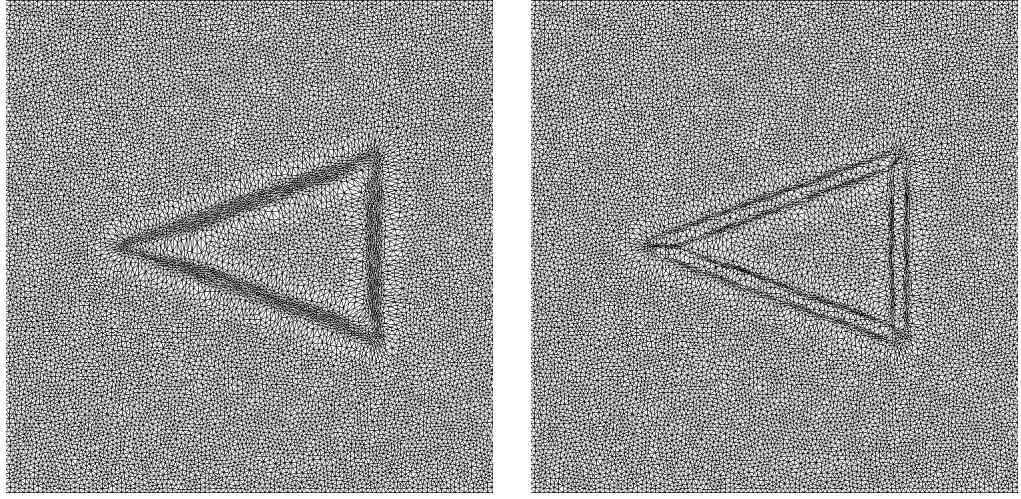
3.1.1 Choix des paramètres

Afin de chercher les paramètres qui génèrent le meilleur maillage, plusieurs tests ont été réalisés :

- Cas test (objet cible):
 - Cercle;
 - Triangle;
 - Profil d'une aile (NACA)
- Type de maillage
 - Structuré;
 - Non structuré

Les meilleurs maillages obtenus ont été utilisés pour résoudre un problème d'écoulement autour de l'objet, afin de vérifier l'influence sur les résultats. Les conclusions des tests sont les suivants :

- Pour le calcul de ω , il faut plutôt utiliser le gradient de u . Le hessien peut être aussi utilisé, mais il a la tendance de créer deux couches raffinées dans les bords de la région de variation de la fonction u (figure 4);
- Le raffinement d'une certaine région du maillage cause un étirement dans la direction normale à surface des éléments voisins à cette région. Ce résultat n'est pas désirable car on veut plutôt une anisotropie dans la direction de l'écoulement du fluide (figure 5) : l'erreur d'interpolation est plus petit si on a des triangles anisotropes dont le côté le plus long est dans la direction des petites dérivées de deuxième ordre de la solution [22]. Ainsi, il est important de avoir une valeur de δ assez grande.
- Néanmoins, δ ne peut être trop grand, car la fonction arctangente s'approche de la pente linéaire, et on n'obtient pas un bon raffinement à l'intérieur de la couche.



(a) $\alpha = 1000, \beta = 0$

(b) $\alpha = 0, \beta = 1000$

Figure 4: Influence des paramètres α et β sur l'adaptation du maillage (les deux cas avec $\delta = 0.02$)

`(fig:alphabeta)`

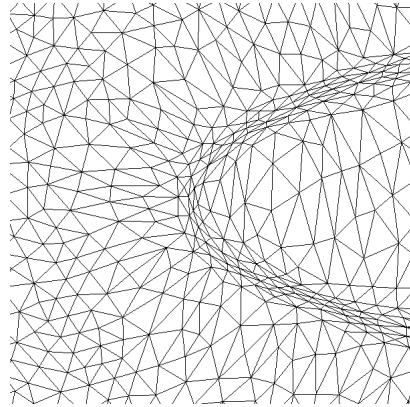


Figure 5: Détail d'un maillage obtenu avec $\delta = 0.005$

`(fig:anisoNormal)`

En tenant compte de ces conclusions, on recommande l'utilisation des paramètres suivants (on rappelle qu'on utilise toujours les normes du gradient et du hessien normalisés entre 0 et 1) :

$$\alpha = 1000, \quad \beta = 0, \quad \delta = 0.01, 0.02$$

On peut aussi faire des adaptations successives, en faisant varier les paramètres, afin d'obtenir d'autres résultats. Par exemple, on peut faire une première adaptation avec $\delta = 0.02$ pour obtenir une bonne couche raffinée, et après une deuxième avec $\delta = 0.01$ pour obtenir un raffinement plus précis autour de la surface.

La figure 6 présente quelques exemples d'adaptation à une fonction Level Set, avec les paramètres présentés ci-dessus, sur un maillage non structuré

avec environ 28000 noeuds :

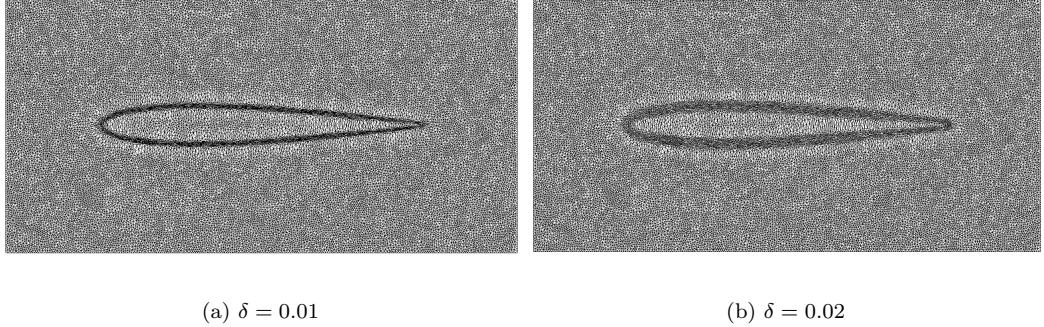


Figure 6: Exemples d’adaptation à une fonction Level Set
(fig:example)

3.2 Calcul des tailles désirées

Le calcul des tailles désirées est basée sur la définition d’une métrique pour chaque point du maillage. Il s’agit d’une matrice 2×2 (dans le cas bidimensionnel) définie positive symétrique, construite de façon à contrôler l’erreur de la représentation de la solution sur le maillage modifiée [21].

La description détaillé de la définition, calcul et interprétation géométrique de la métrique seront faites dans la section 4, pour l’adaptation à une solution physique. Dans le cas de l’adaptation Level Set, on a implémenté un calcul simple des tailles désirées, en se basant sur la méthode utilisée par [12], qui permet de contrôler la distance de Hausdorff entre la surface de l’objet et sa représentation. En faisant attention toujours à l’importance, pour une bonne adaptation, de créer une fonction ω qui soit au même temps assez lisse et qui présente des variations qui permettent le mouvement des noeuds, on définit, avec le paramètre δ , des couches pour l’imposition des tailles :

$$h_{des}(\mathbf{x}_i) = \begin{cases} h_{min} & si \quad |LS(\mathbf{x}_i)| \leq \frac{\delta}{2} \\ 2h_{min} & si \quad \frac{\delta}{2} \leq |LS(\mathbf{x}_i)| \leq \delta \\ h_{max} & sinon \end{cases}$$

et on lisse les tailles sur le maillage avec le logiciel *MMG*, en imposant une gradation de 10%.

3.2.1 Résultats

La figure 7 illustre l’adaptation Level Set basée sur l’imposition de tailles désirées à chaque noeud.

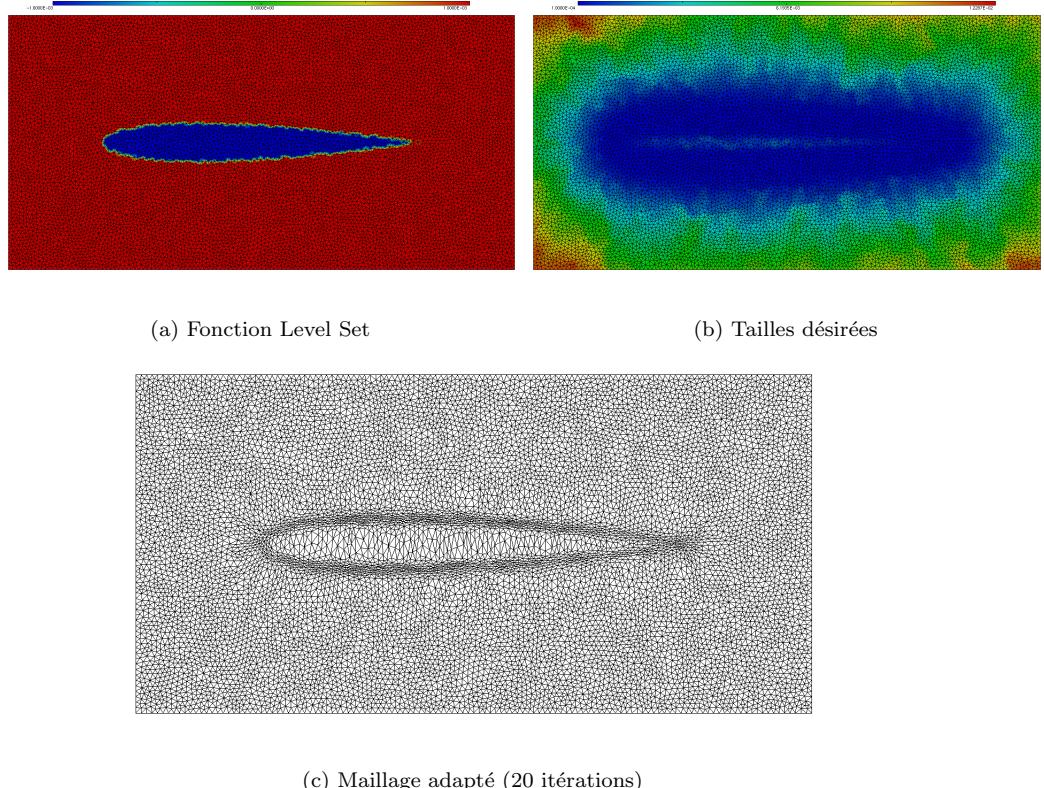


Figure 7: Adaptation à une fonction Level Set à partir de l'imposition des tailles désirées
 (fig:adaptMet)

3.3 Relaxation et qualité du maillage

Pendant la résolution du système linéaire, on fait, si nécessaire, une relaxation des déplacements calculés afin d'éviter le croisement des noeuds et la conséquente occurrence d'aires négatives. Néanmoins, en dépendant du maillage de départ (par exemple, un maillage déjà raffiné), la relaxation est trop contraignant et impose l'arrêt de l'algorithme lors des premières itérations. Parfois cela est provoqué par un petit nombre d'éléments.

On a ainsi implémenté, dans les premières itérations, un prétraitement d'optimisation, selon le schéma suivant :

- Si le numéro de l'itération actuelle n'est pas supérieur à $iter_{optim}$:
 1. Identifier tous les éléments qui demanderont de la relaxation;
 2. Classifier ces éléments selon un critère de qualité (mesure de l'anisotropie);
 3. Optimiser les n_{optim} éléments les plus critiques de cette liste

L'optimisation réalisée consiste en, itérativement, approcher chaque noeud i de l'élément concerné du barycentre des noeuds voisins de i . L'optimisation est arrêtée dès qu'aucun des trois noeuds de l'élément bouge au dessus d'un seuil, ou qu'au moins un des noeuds n'arrive pas à réduire l'anisotropie au dessous d'une certaine pourcentage, par rapport au début de l'itération.

La figure 8 montre le résultat de son application sur un élément :

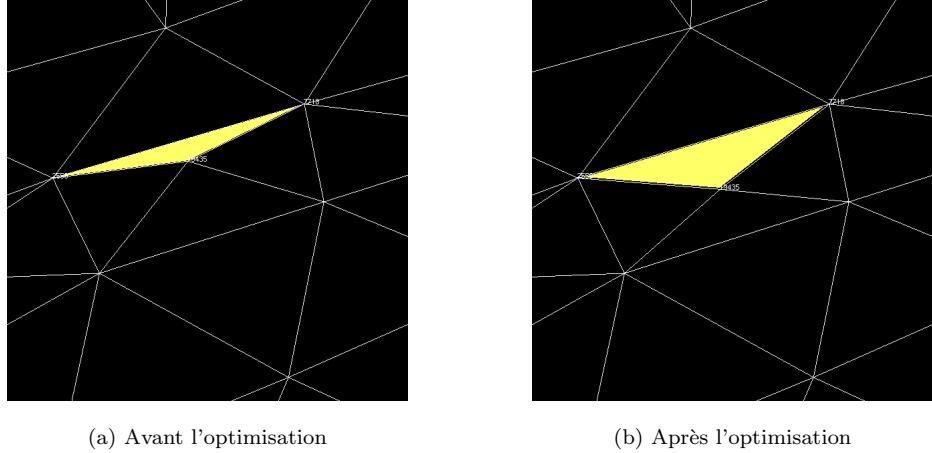


Figure 8: Résultat de l'optimisation d'un élément

(fig:optim) Pour que le prétraitement ne soit pas très coûteux, on a choisi $n_{optim} = 10$ et $iter_{optim} = 2$. Les tests réalisés montrent des très bons résultats quand l'algorithme s'arrête au début de l'exécution, et avec l'optimisation on arrive à tourner tous les itérations. Néanmoins, si appliqué sur un maillage de bonne qualité et qui ne demande pas une forte relaxation, les gains sont en général petits (ou même on perd un peu de la qualité du résultat final).

3.4 Adaptation 3D

Après le développement et validation du modèle bidimensionnel décrit jusqu'ici, avec l'obtention de bons résultats d'adaptation des maillages, on l'a étendu au cas tridimensionnel. Du point de vue théorique, cette extension est assez simple, étant le modèle toujours décrit par le problème (2). Le développement de sa formulation faible reste aussi la même, et on se ramène ainsi à la résolution des systèmes linéaires

$$\begin{cases} Kx = 0 \\ Ky = 0 \\ Kz = 0 \end{cases}$$

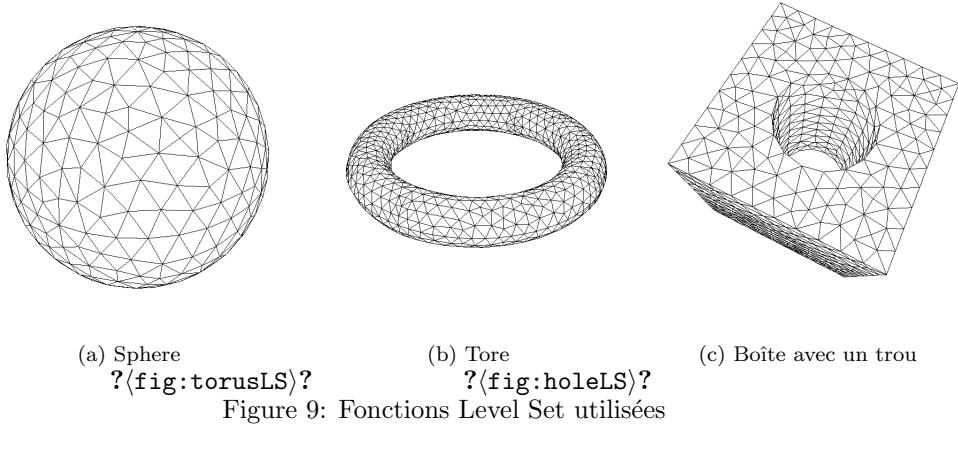
Comme ces systèmes sont indépendants et ont la même forme, l'extension du cas 2D au 3D a été également simple du point de vue de l'implémentation.

En effet, parmi les calculs décrits en détail dans les sous-sections 2.3 et 2.4, l'unique modification concerne le calcul des vecteurs normaux ($(\nabla_{\xi\eta}\varphi_i)^T = \frac{\mathbf{n}_i^T}{d!|T|}$, avec $d = 3$). Ainsi, les éléments de la matrice K sont donnés par

$$\begin{aligned} k_{ij} &= \int_{\Omega_{\boldsymbol{\xi}}^h} \omega \nabla_{\xi\eta}\varphi_i \cdot \nabla_{\xi\eta}\varphi_j d\boldsymbol{\xi} = \sum_{T \ni i} \int_T \omega \nabla_{\xi\eta}\varphi_i \cdot \nabla_{\xi\eta}\varphi_j d\boldsymbol{\xi} = \\ &= \sum_{T \ni i} |T| \omega^T \frac{\mathbf{n}_i^T \cdot \mathbf{n}_j^T}{(3!|T|)^2} = \sum_{T \ni i} \omega^T \frac{\mathbf{n}_i^T \cdot \mathbf{n}_j^T}{36|T|} \end{aligned}$$

On présente dans les figures 10 à 12 quelques résultats de l'adaptation Level Set tridimensionnelle, à partir de l'imposition des tailles désirées. Les fonctions Level Set utilisées sont celles de la figure 9. Malgré les résultats analogues au cas 2D et cohérents avec les objectifs ici proposés, l'adaptation 3D a des limitations plus fortes concernant le nombre de points du maillage, qui peut devenir très grande pour qu'on ait un maillage suffisamment fine pour la bonne représentation de la fonction Level Set, et la relaxation de l'adaptation, puisque le degré de liberté additionnel peut conduire plus rapidement au croisement des noeuds.

Les domaines de départ sont plus raffinées dans les régions où se trouve l'objet, afin de minimiser l'influence des bords sur l'adaptation. Le logiciel *MMG* a été utilisé pour les générer, en imposant une taille minimale $hmin = 0.05$ et une gradation de $hgrad = 1.1$ de la variation de las tailles. Les adaptations ont été faites avec δ entre 0.05 et 0.07.



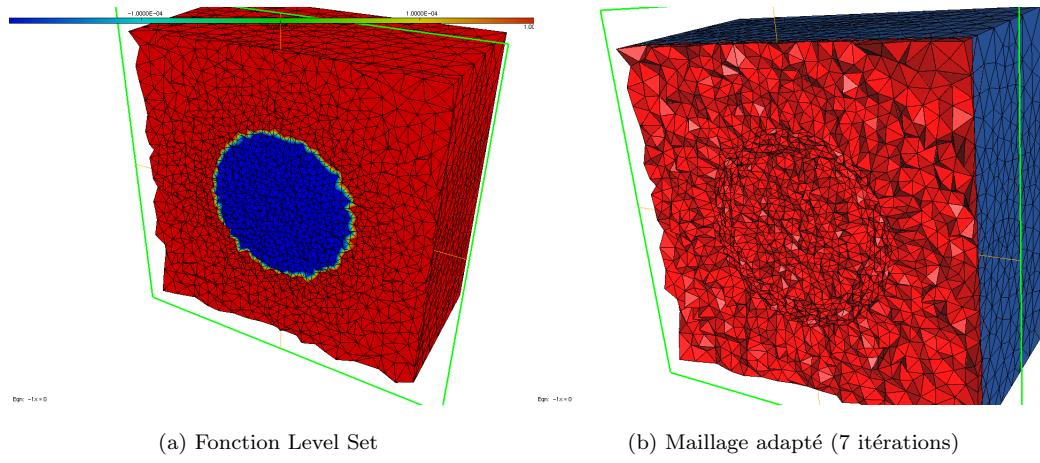


Figure 10: Adaptation du maillage à la sphère (environ 34000 noeuds)

{fig:adaptSphere}

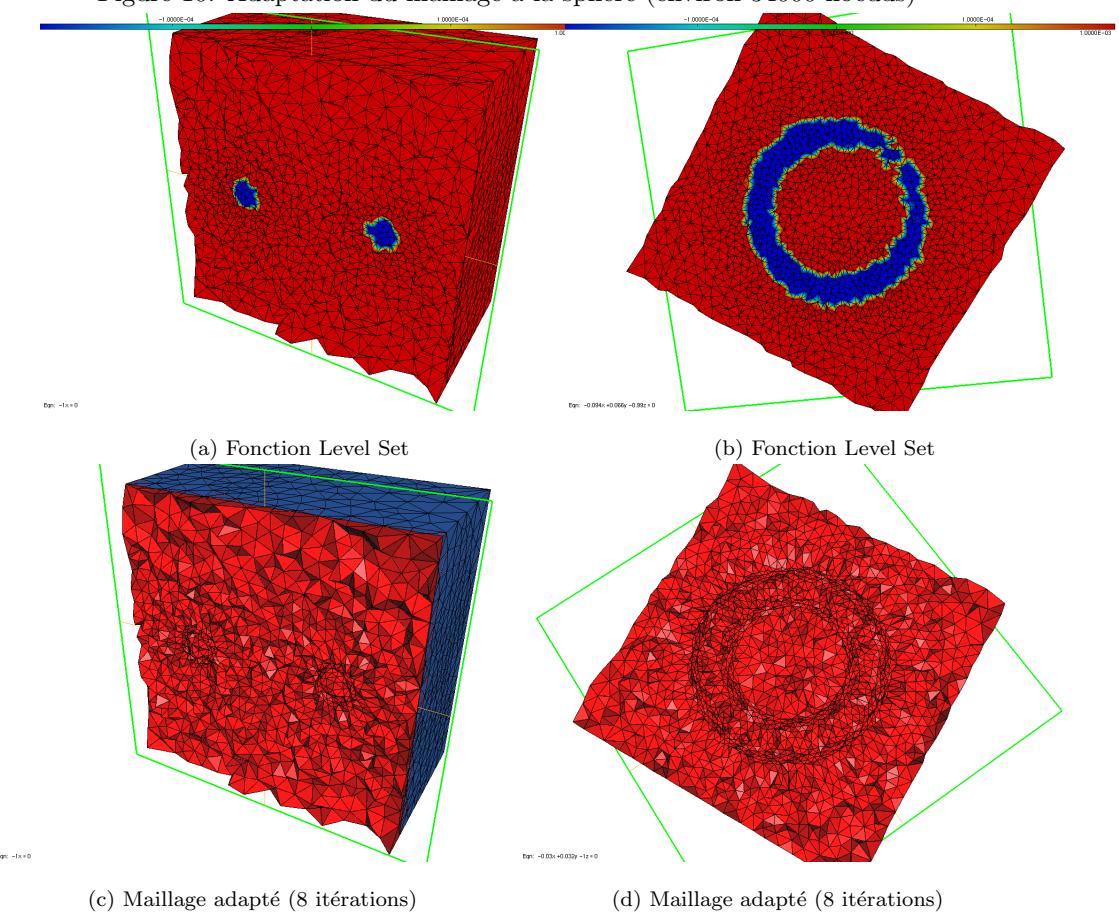


Figure 11: Adaptation du maillage au tore (environ 25000 noeuds)

?{fig:adaptTorus}?

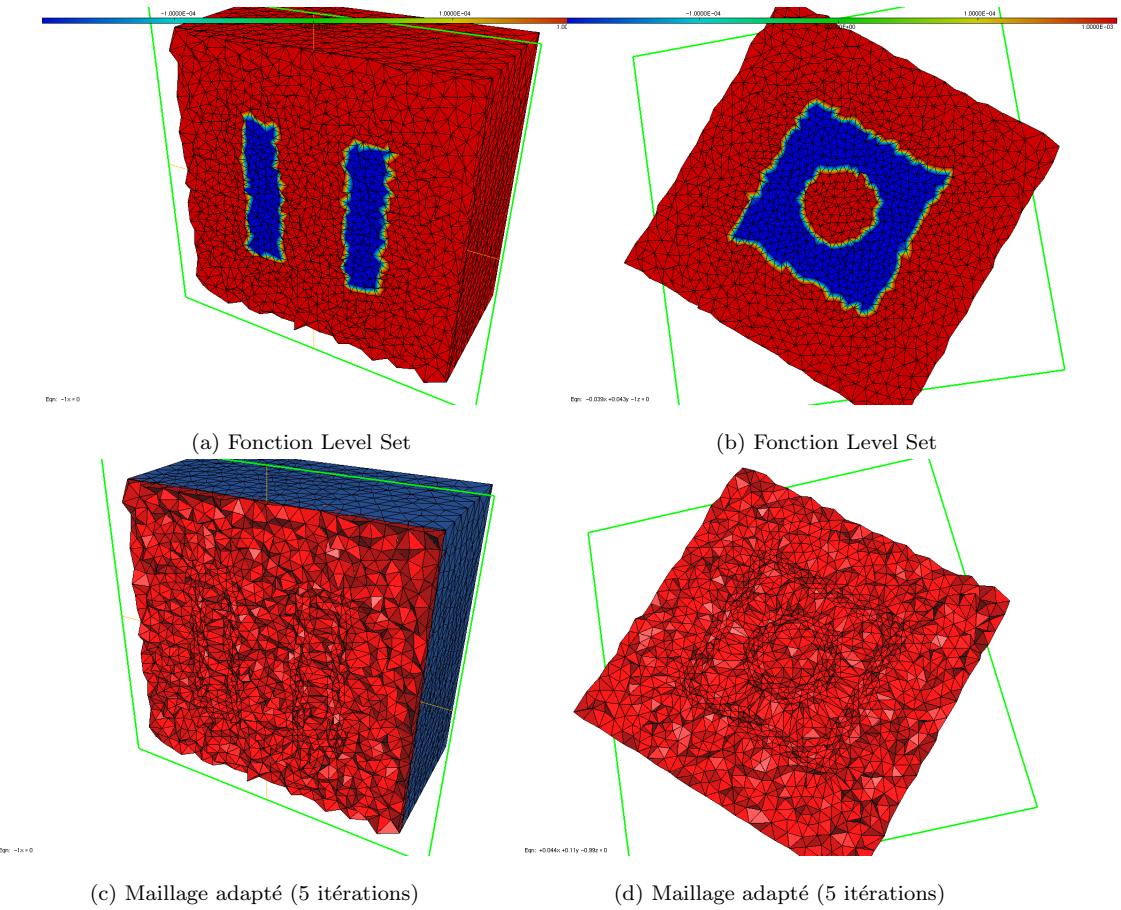


Figure 12: Adaptation du maillage à une boîte avec un trou (environ 34000 noeuds)

`(fig:adaptHole)`

4 Application à l'adaptation physique

De la même façon que l'adaptation Level Set, l'adaptation physique (i.e., à une variable physique calculée sur le maillage, comme le champs de vitesse de l'écoulement) peut être faite avec les deux formulations de ω . Pour la première, basée sur la variation de la fonction, on n'a pas de remarques supplémentaires à faire : au contraire de la fonction Level Set, qui est toujours lisse, les variables physiques dans les problèmes de mécanique de fluides qui nous intéressent présentent en général des discontinuités, ayant ainsi des fortes gradients et hessiens qui permettent une bonne adaptation. Alors, il ne faut pas construire une autre fonction à adapter.

On va ainsi détailler seulement le deuxième calcul de ω :

4.1 Calcul des tailles désirées

Le calcul des tailles présenté dans la suite s'inspire dans [11] et [13], et s'est basée sur la notion de métrique : pour chaque noeud i du maillage, on définit une matrice \mathcal{M}_i 2×2 (dans le cas bidimensionnel) qui minimise un estimateur de l'erreur d'interpolation de la solution sur le maillage.

Étant $\Pi_h u$ l'interpolation de u sur le maillage, cet estimateur est donné, pour chaque élément K du maillage, par

$$\|u - \Pi_h u\|_{\infty, K} \leq c_d \max_{x \in K} \max_{\vec{e} \in E_K} \langle \vec{e}, |H_u(\mathbf{x})| \vec{e} \rangle \quad (9) \boxed{\text{eq:erreur_interp}}$$

où $c_d = \frac{2}{9}$ est obtenu avec un développement de Taylor de (9) dans son point de maximum. On veut imposer une limite ε à cette erreur :

$$\|u - \Pi_h u\|_{\infty, K} = \varepsilon$$

En définissant la métrique

$$\mathcal{M} = \frac{c_d}{\varepsilon} H_u(\mathbf{x}) \quad (10) \boxed{\text{eq:def_métrique}}$$

la longueur des arêtes selon cette métrique est alors

$$l_{\mathcal{M}_i} = \langle \vec{e}, |\mathcal{M}_i| \vec{e} \rangle = 1$$

c'est à dire, la métrique qui assure l'erreur d'interpolation désirée est celle telle que le maillage soit unitaire [11, 13].

Étant la métrique une matrice positive définie symétrique, elle est toujours diagonalisable et ses valeurs propres $\tilde{\lambda}_i^j$, $j = 1, 2$, sont toujours réelles. En effet, la métrique peut être représentée par une ellipse (ou un ellipsoïde, en 3D), dont les vecteurs propres donnent la direction des axes et les valeurs propres sa taille, selon la relation $h_j = \frac{1}{\sqrt{\tilde{\lambda}_i^j}}$ [21]. Ainsi, les caractéristiques géométriques des éléments (taille, forme et orientation) est contenue dans la métrique, et comme on a plus d'un valeur propre, on en définit ainsi des éléments anisotropes.

Néanmoins, on va considérer un cas isotrope, en prenant la plus grande valeur propre de la métrique, ce qui donne la plus petite taille de chaque élément. Enfin, en établissant des seuils minimal et maximal pour la taille désirée (h_{min} et h_{max}), et en tenant compte de l'équation (10), les tailles désirées sont calculées à partir de

$$h_i = \frac{1}{\sqrt{\min \left(\max \left(\frac{c_d}{\varepsilon} \lambda_i, h_{max}^{-2} \right), h_{min}^{-2} \right)}}$$

où $\lambda_i = \max_{j=1,2} |\lambda_i^j|$ est la plus grande valeur propre du hessien $H_u(\mathbf{x}_i)$, en valeur absolue.

Également au cas de l'adaptation Level Set, les tailles définies sur le maillage sont lissées avec une gradation de 10%.

La figure 13 présente un exemple de métrique calculée à partir d'une variable physique .

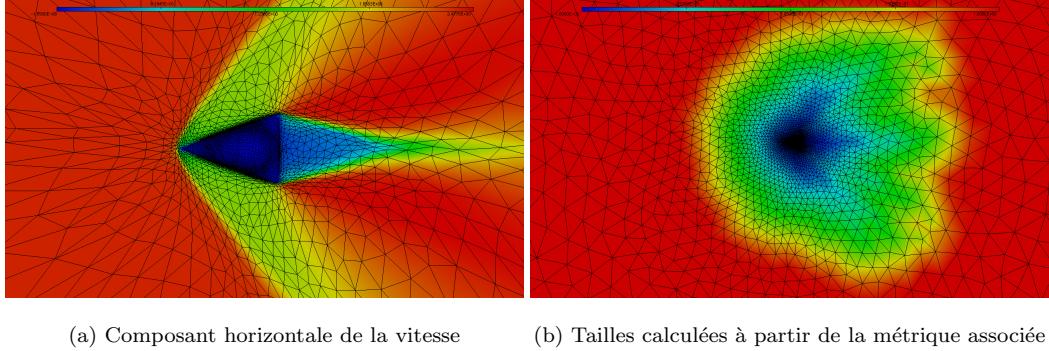


Figure 13: Écoulement autour d'un triangle - calcul des tailles désirées à partir d'une variable physique

`(fig:metPhys)`

4.2 Couplage entre adaptation Level Set et adaptation physique

L'objectif principal de l'ensemble du travail réalisée dans ce projet est l'application du modèle et de la bibliothèque développés à la résolution de problèmes de la mécanique de fluides, afin d'améliorer la précision des résultats. Ainsi, un bon maillage serait adapté au même temps à la surface de l'objet dans le domaine et au profil de la variable physique calculée (la vitesse, par exemple), et par conséquent les deux types de adaptation présentées ci-dessus doivent être considérées.

Pour coupler les deux adaptations, on va intersector ses respectives métriques : en notant h_i^{LS} et h_i^{phys} la taille calculée pour le noeud i à partir des métriques, respectivement, de la fonction Level Set et de la variable physique (avant le lissage), on choisit le taille

$$h_i = \min(h_i^{LS}, h_i^{phys})$$

Ensuite, on applique le même lissage avec une gradation de 10%.

Pour le couplage des deux adaptation, on a proposé la procédure itérative suivante :

1. Calcul de la métrique associée à la fonction Level Set;

- `(item:resolution)`
2. Adaptation du maillage à cette métrique;
 3. Résolution du problème physique sur le maillage adapté;
 4. Calcul de la métrique associée à la variable physique et intersection avec la métrique de la fonction Level Set
 5. Adaptation du maillage à cette métrique, en partant du dernier maillage obtenu, mais toujours avec le même maillage de référence (le maillage de départ, non adapté);
 6. Retour au pas 3

4.3 Résultats

La procédure décrite ci-dessus a été utilisé dans la résolution d'un flot supersonique 2D autour d'un triangle, avec une méthode de distribution de résidus appliquée aux équations de Navier Stokes pénalisées. Ce cas test est présenté et résolu dans [21], dont les paramètres de l'écoulement sont les mêmes utilisés ici.

Étant le modèle pénalisé, le triangle n'est pas discrétisé dans le domaine, ce qui signifie qu'il y a des mailles à son intérieur. En effet, le terme de pénalisation est appliqué exactement aux noeuds à l'intérieur de l'objet, identifiés par la signe de la fonction Level Set. Ainsi, la première adaptation de notre procédure (adaptation à la ligne de niveau 0 de la fonction Level Set) est de grande importance pour la représentation de la surface et la résolution précise des équations dans ses voisinages.

La solution physique utilisée pour les adaptations suivantes est la composante horizontale de la vitesse.

Les tests ont été réalisées avec l'objectif de vérifier l'évolution de la qualité du résultat dans chaque itération de la procédure. Cette vérification sera faite de façon qualitative et quantitative. Pour la première, on regardera par exemple la régularité du profil de la vitesse et de ses lignes de niveau et la résolution autour des coins du triangle. Pour la deuxième, on adoptera le même critère utilisé par [21], en calculant l'angle entre le choc et l'axe $y = 0$, en utilisant un point du choc proche à cet axe, et identifiée par la proximité des lignes de niveau de la vitesse. La résolution analytique du problème fournit un angle $\beta = 53.33^\circ$, qu'on utilisera pour la comparaison.

Le résultat présenté dans les figures 14 à 17 a été fait sur un domaine circulaire, de rayon vingt fois plus grand que la taille du triangle, afin de représenter un domaine infini, qui minimise l'influence des bords. Afin d'éviter un nombre trop grand de points, mais encore avec un bon raffinement dans

les régions les plus proches des bords pour bien capturer le choc, les tailles des éléments ont été définies en fonction de la distance au centre du domaine, augmentant linéairement entre 0.002 et 1. Le résultat est un maillage avec 12664 noeuds et 25200 éléments.

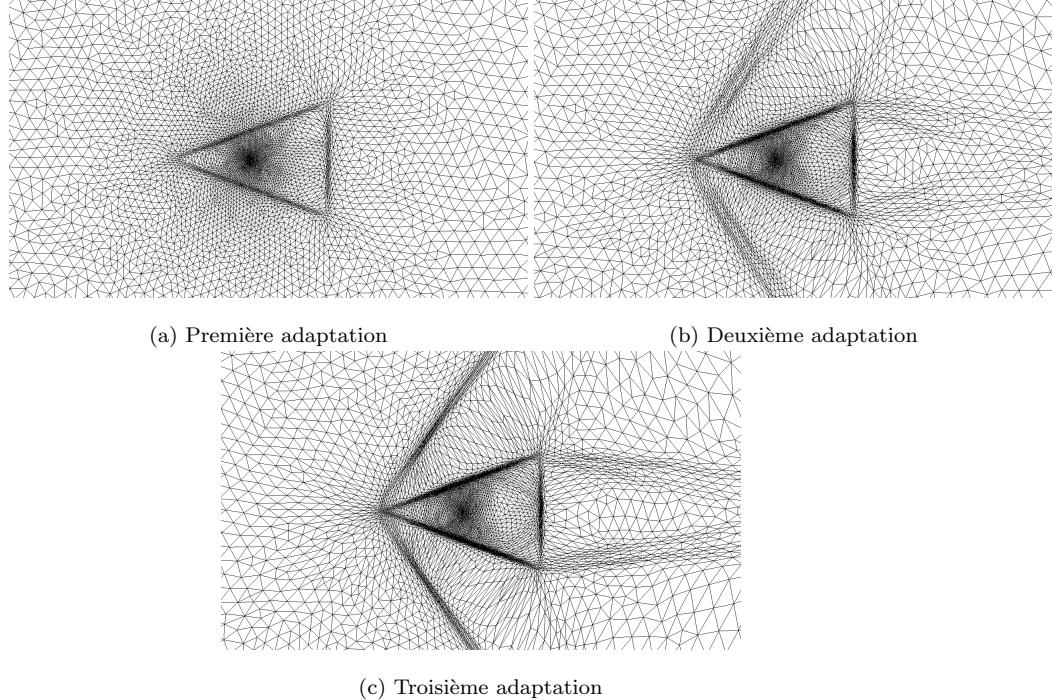
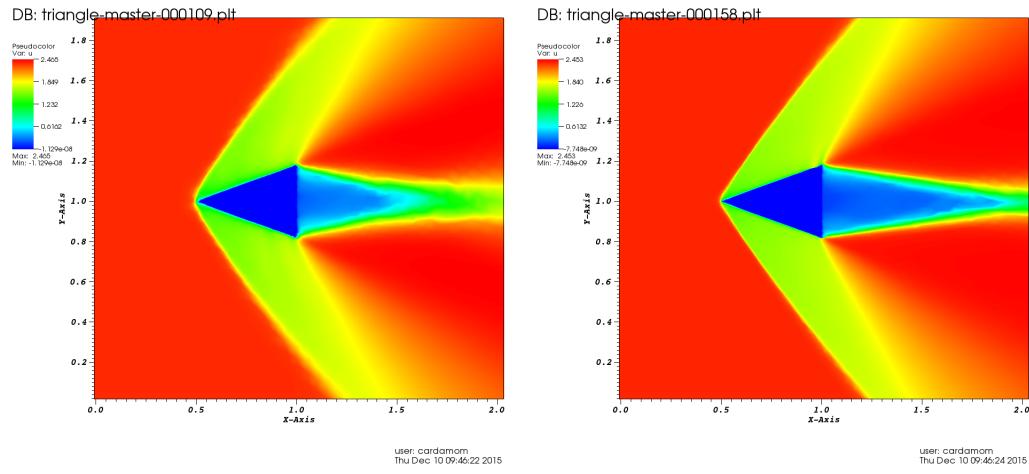
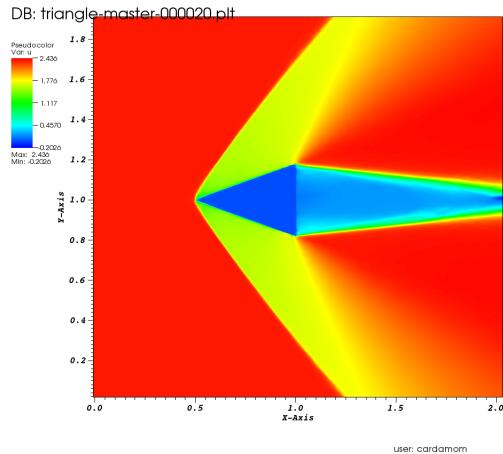


Figure 14: Séquence de maillages adaptés obtenus

`(fig:adapPhysDoms)`

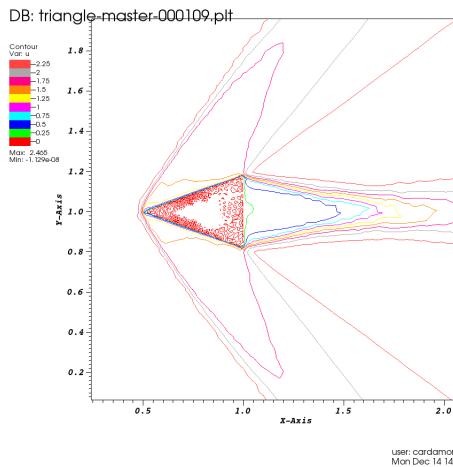




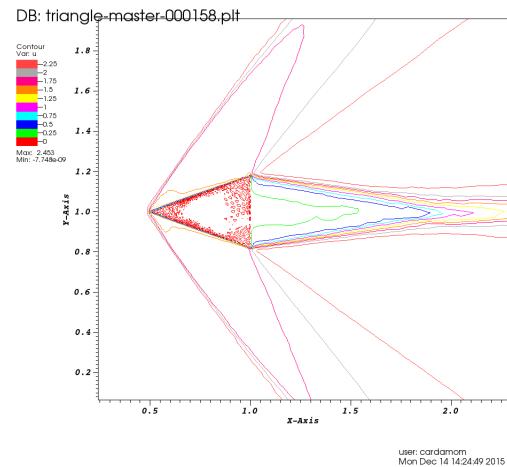
(c) Troisième résolution

Figure 15: Composante horizontale de la vitesse

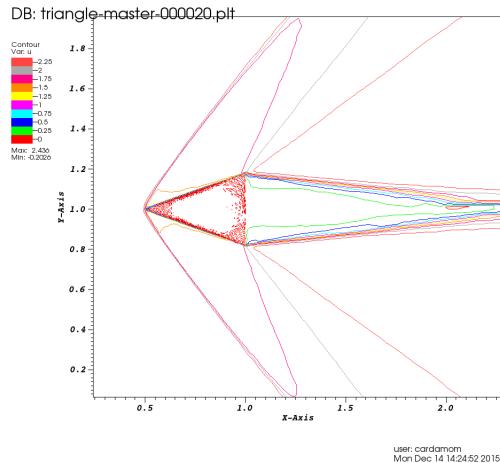
`<fig:adapPhysPlotA>`



(a) Première résolution

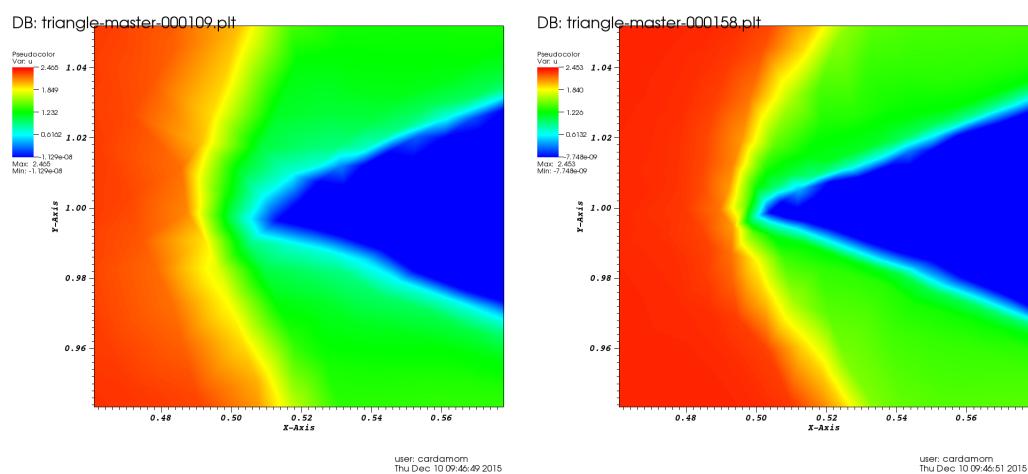


(b) Deuxième résolution



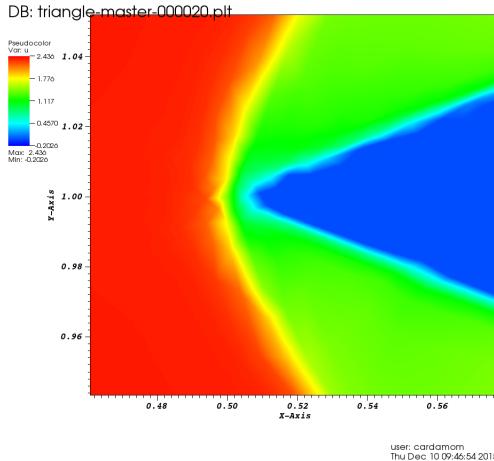
(c) Troisième résolution

Figure 16: Lignes de niveau de la composante horizontale de la vitesse



(a) Première résolution

(b) Deuxième résolution



(c) Troisième résolution

Figure 17: Détail - Composante horizontale de la vitesse

L’angle β a évolué selon les valeurs présentées dans le tableau 1

Résolution	β
1	55.68°
2	53.92°
3	53.01°

Table 1: Angle entre le choc et l’axe $y = 0$

`(tab:beta)`

Ces résultats montrent que la procédure proposée permet en effet d’améliorer la résolution du problème physique : à chaque itération, le champ de vitesse calculée devient mieux définie, avec des lignes de niveau plus régulières, et l’angle entre le choc et l’axe $y = 0$ s’approche de la valeur analytique. Néanmoins, quelques difficultés sont encore présentes dans les tests réalisées, notamment la génération de maillages permettant un calcul stable (considérant les fortes discontinuités qui caractérisent ce cas test) et le fait que, comme montre la figure 17, le choc n’est pas attaché au triangle, au contraire du résultat théorique, ce qui s’explique par les limitations du raffinement du maillage.

5 Adaptation à un cas non stationnaire

`(sec:nonstat)` Les résultats présentés dans les sections précédentes se réfèrent à des cas stationnaires, avec les surfaces fixes, mais une situation qu’on veut considérer aussi est celle où les objets bougent. Dans ce cas, il faut à chaque pas de

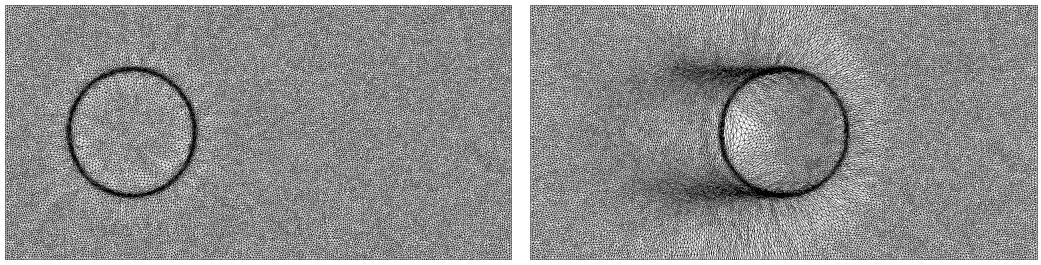
temps actualiser la fonction Level Set et adapter le maillage autour de la nouvelle position de la surface. La procédure adoptée est la suivante :

1. Pour tout noeud du maillage, calculer la fonction Level Set par rapport à la position actuelle de l'objet, et la fonction u à adapter (equation (8));
2. Adapter le maillage à la fonction u ;
3. Faire bouger la surface de l'objet (indépendamment de la position des noeuds)

L'objectif de cette procédure est de faire chaque adaptation à partir du dernier maillage adapté, au lieu de partir du maillage non adapté à chaque pas du mouvement. Ainsi, on peut éviter de garder en mémoire toute la structure du maillage de référence et du maillage adapté. Il faut pourtant garder au moins les aires et les vecteurs normaux du maillage, parce que, en raison du modèle adopté, toutes les adaptations doivent utiliser le même maillage de référence.

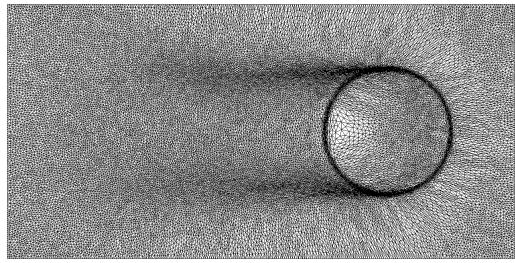
Les figures 18 et 19 présentent deux exemples de cette procédure, le premier pour l'advection d'un cercle et le deuxième pour l'oscillation de l'aile Naca. Les mouvements ont été discrétisées en 20 pas, et chaque itération a été réalisée avec 20 ou 30 itérations, respectivement.

On peut observer qu'on obtient à chaque pas une bonne adaptation à la position actuelle de la fonction Level Set, mais que les positions précédentes ne sont pas complètement "désadaptées", même que son trace soit faible. En effet, les tests montrent que le passage d'une adaptation à la prochaine se fait de deux manières, soit par le mouvement des points d'une région adaptée à l'autre, soit par le "dérafinement" d'une région. On a observé que le deuxième est plus lente que le premier, et ainsi on aurait besoin d'un plus grand nombre d'itérations. Néanmoins, pour justifier l'application de cette procédure, il faut trouver un équilibre entre le temps d'exécution et la qualité des résultats.



(a) Pas n. 0

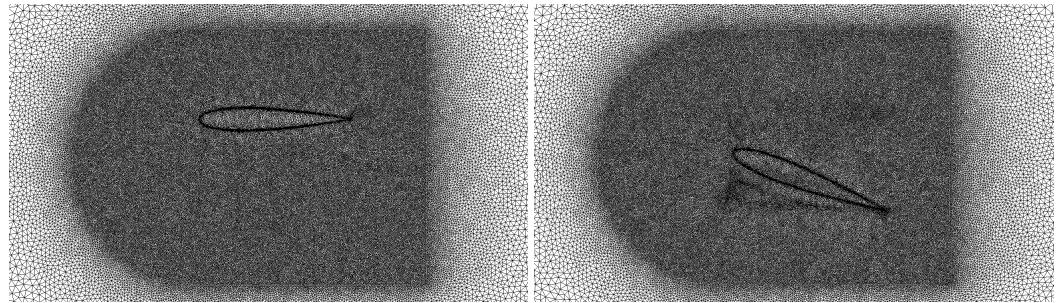
(b) Pas n. 10



(c) Pas n. 20

Figure 18: Advection d'un cercle

`(fig:circleLSAdv)`



(a) Pas n. 0

(b) Pas n. 10



(c) Pas n. 20

Figure 19: Naca oscillant

`(fig:nacaLSAdv)`

Part II

Meric / Inria Chile - Méthodes de Décomposition de Domaine

1 Introduction

Le travail réalisée dans le deuxième stage de mon année de césure s’insère dans le contexte et les objectives de Meric : la recherche scientifique multidisciplinaire liée à la production d’énergie Marine au Chili. Plus spécifiquement, je faisait partie de l’équipe responsable pour la ligne d’investigation appelé “Modélisation avancée pour l’énergie marine”, dont le principal objectif sont le “couplage de modèles de petites résolutions spectrales a des grandes résolutions dans le domaine du espace-temps” et “la modélisation de grande résolution spatiale et temporelle proche de la côte”¹.

Les sujets étudiés dans ce stage constituent une introduction à la recherche envisageant ces objectifs et peuvent être divisés en deux branches principales: la première consiste dans l’étude, implémentation numérique et simulation de modèles de propagation d’ondes d’eau; le deuxième, plus directement lié à la ligne de recherche de Meric, envisage l’étude, développement et application numérique de méthodes de décomposition de domaine (DDMs) pour la résolution de tels modèles.

Concernant le premier point, on a étudié plusieurs modèles qui prennent en compte les effets non linéaires et dispersives dans la propagation des ondes. Cet étude s’est focalisé notamment sur l’équation de KdV et l’équation de Serre. En considérant les objectifs de la recherche développé par l’équipe, l’étude de différentes modèles mathématiques a une grande importance : premièrement, chacun d’eux est dérivée à partir d’hypothèses différents, et, ainsi, ses domaines de validité physique sont différents. Des critères comme les caractéristiques de l’onde (amplitude et longueur), la profondeur de l’eau et le fond océanique guident le choix d’un ou autre modèle. Dans le contexte de la production de l’énergie marine, par exemple, la distance à la côte est très importante, et le choix du bon modèle est essentielle pour que les variables d’intérêt, comme la vitesse et l’amplitude des ondes, peuvent être calculées de manière fiable.

Par ailleurs, en raison des différentes complexités de chaque modèle, les approches nécessaires pour son étude dans le contexte des méthodes de

¹<http://www.meric.cl/que-hacemos/>

décomposition de domaine sont aussi différents. Dans ce stage, cet étude concernant l'équation de KdV a été bien avancé (entraînant même l'élaboration du papier scientifique qu'on a envoyé pour son publication); dans les dernières semaines du stage on a débuté l'étude de la résolution des équations de Serre avec des DDMs, mais encore sans des résultats pour présenter dans ce rapport.

Les DDMs consistent en diviser un domaine computationnel en plusieurs subdomaines et résoudre un problème en chacun d'eux. Pour que la solution du problème décomposé soit la même que la solution du problème calculé dans le monodomaine, il est essentiel qu'on défini des conditions appropriés sur l'interface entre les subdomaines.

Ainsi, dans la deuxième ligne de notre travail, on a d'abord fait un étude des conditions aux limites transparentes (TBCs) pour l'équation de KdV, en tenant compte que ces types de conditions aux bords jouent un rôle essentiel dans les méthodes de décomposition de domaine. Afin d'introduire ce sujet, on a suivi au début un approche plus simple, en optimisant des conditions aux bords classiques que fournissaient des conditions les plus proches des TBCs pour l'équation de KdV.

Après, on s'est concentré dans l'étude des TBCs exactes pour l'équation de KdV linéarisée, en se basant sur l'es études développés par [25] et [6]. Étant non locaux en temps, ces TBCs exactes ne peuvent pas être implémentées en pratique; alors notre but était de les proposer des approximation simples, qu'on a ensuite appliqué à une méthode de décomposition de domaine.

À ce moment, il est nécessaire de clarifier et délimiter nos objectifs et les différences par rapport aux objectifs de [25] et [6]. Pour cela, on fait une brève description des sources d'erreur et incertitudes liées aux simulations numériques de modèles physiques.

De façon générale, ces erreurs peuvent être classifiés en des erreurs de modélisation conceptuelle et des erreurs numériques [23]. Dans le premier groupe, il y a les assumptions de modélisation conceptuelle (pour le phénomène physique et les conditions aux bords) et les incertitudes dans la géométrie, les donnés initiales, les donnés aux bords et les paramètres qui jouent un rôle dans le modèle [23, 4]. Dans ce qui concerne les erreurs numériques, on peut citer ceux liés à la finitude do domaine computationnel, les erreurs temporales et les erreurs spatiales dues à la discréétisation des équations[16, 23] et d'autres possibles sources d'erreur liées à la méthode numérique, comme dans des procès itératives (comme est le cas de la DDM implémentée ici).

L'erreur totale de simulation numérique est une somme des contributions de chacune de ces sources. Ainsi, la connaissance et quantification d'eux sont essentielles pour améliorer la description numérique du procès physique et, dans ce contexte, l'étude séparé de chaque contribution a une grande

importance.

Parmi les erreurs mentionnées ci-dessus, [25] et [6] cherchaient à minimiser celui lié à la finitude du domaine computationnel. En fait, en tronquant le domaine avec l'introduction de bords artificielles, il faut utiliser des conditions aux bords additionnelles, qui doivent être choisies proprement afin d'assurer la stabilité et la précision du problème aux valeurs initiales et aux bords tronqué [25]. Même en utilisant des approches différents, les deux auteurs envisageaient à construire des conditions aux limites absorbantes (ABCs), qui simulent l'absorption d'une onde sortant du domaine computationnel, ou des TBCs, qui assurent que la solution approximée dans le domaine computationnel coïncide avec la solution de tout le domaine.

Ainsi, notre travail ne doit pas utiliser la même solution de référence que celle utilisée par [25] et [6] : pour valider leurs approches, ils comparent ces solutions approximées avec la solution exacte dans tout le domaine. En revanche, notre solution de référence sera la solution approximée calculée dans le monodomaine computationnel. En suivant le principe d'étudier séparément chaque type d'erreur numérique, on ne cherche pas à minimiser l'erreur due à l'introduction de bords externes au domaine computationnel, mais seulement l'erreur due à l'introduction d'une interface entre les subdomaines.

2 L'équation de KdV

?⟨sec:KdV⟩? 2.1 Le modèle

Le premier modèle de propagation d'ondes étudié et implémenté dans ce projet est l'équation de Korteweg-de Bries (KdV) , qui prend en compte les effets non linéaires et dispersives et qui est une bonne approximation pour des ondes de petite amplitude et large longueur. [5]

Plusieurs formes de cette équation peuvent être trouvées dans la littérature, variant notamment dans les facteurs d'échelle pour chacun des processus physiques présents dans l'équation (non linéarité et dispersion). On va considérer la formulation présentée par [5], qui est écrite en termes de variables sans dimensions mais non échalonées :

$$u_t + u_x + (u^2)_x + u_{xxx} = 0$$

2.2 Discrétisation

Le problème qu'on veut résoudre, avec une condition initial Φ et des conditions aux bords appropriées, est

$$\begin{cases} u_t + u_x + (u^2)_x + u_{xxx} = 0 , \quad x \in [x_{min}, x_{max}], \quad t \in [0, t_{max}] \\ u(x, 0) = \Phi(x) \\ + \text{boundary conditions} \end{cases}$$

Dans un premier moment, pour valider l'implémentation du modèle, sans influence des bords, on va considérer des conditions aux bords périodiques, ou des conditions de Dirichlet et/ou de Neumann homogènes mais suffisamment éloignées de l'onde (c'est-à-dire, du support de la solution).

La résolution numérique du problème sera faite avec un schéma de *splitting*, en séparant les termes d'advection et de dispersion. Alors, en définissant les opérateurs

$$\begin{aligned} T_a(u) &= u_t + u_x + (u^2)_x \\ T_d(u) &= u_t + u_{xxx} \end{aligned}$$

on va résoudre, dans chaque pas de temps $[t_n, t_{n+1}]$:

$$\begin{cases} T_a(v) = 0 , \quad t \in [t_n, t_{n+1}], \quad v^n = u^n \\ T_d(w) = 0 , \quad t \in [t_n, t_{n+1}], \quad w^n = v^{n+1} \\ u^{n+1} = w^{n+1} \end{cases}$$

Les schémas numériques utilisés dans chacun de ces pas sont décrites ci-dessous :

2.2.1 Premier pas du *splitting* - advection

?<sec:KdVSplitted1> Le premier pas du schéma de splitting pur l'équation de KdV est une loi de conservation hyperbolique, qui peut être écrite en termes d'une fonction de flot f :

$$v_t + f(v)_x = 0, \quad f(v) = v + v^2 \tag{11} \boxed{\text{eq:conservationLaw}}$$

Cette équation sera résolue avec une méthode de volumes finis, avec les cellules $[x_{i-1/2}, x_{i+1/2}]$ centrées aux points discrets x_i et avec des valeurs moyennes égales à la solution dans ces points. La dérivée spatiale dans (11) sera discrétisée avec une méthode de Runge-Kutta d'ordre 4 :

$$\begin{aligned}
k_1 &= -f(v_i^n)_x \\
k_2 &= -f \left(v_i^n + k_1 \frac{\Delta t}{2} \right)_x \\
k_3 &= -f \left(v_i^n + k_2 \frac{\Delta t}{2} \right)_x \\
k_4 &= -f(v_i^n + k_3 \Delta t)_x \\
v_i^{n+1} &= v_i^n + \frac{\Delta t}{6} (k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}$$

et la dérivée spatiale sera approximée en termes du flot dans les interfaces de las cellules :

$$f(v_i^n)_x = \frac{f(v_{i+1/2}^n) - f(v_{i-1/2}^n)}{\Delta x}$$

Alors, il faut calculer las valeurs de u dans chaque interface. Cela sera fait en résolvant le problème de Riemann suivant :

$$\begin{cases} v_t + f(v)_x = 0 \\ v(x, 0) = v^- , \quad x < 0 \\ v(x, 0) = v^+ , \quad x > 0 \end{cases}$$

où l'interface se trouve dans $x = 0$ et v^- et v^+ sont les solutions dans les cellules voisines.

La fonction de flot f es uniformément convexe; alors le problème de Riemann a une unique solution faible admissible [24] :

- If $v^- > v^+$ (choc) :

$$v(x, t) = \begin{cases} v^- , & f(v^-) > f(v^+) \\ v^+ , & f(v^-) < f(v^+) \end{cases}$$

- If $v^+ > v^-$ (onde de raréfaction) :

$$v(x, t) = \begin{cases} v^- , & f'(v^-) > 0 \\ (f')^{-1}(v) , & f'(v^-) < 0 < f'(v^+) \\ v^+ , & f'(v^+) < 0 \end{cases}$$

2.2.2 Deuxième pas du *splitting* - dispersion

Deux schémas seront proposés pour la résolution du deuxième pas de l'équation de KdV :

$$w_t + w_{xxx} = 0 \quad (12) \boxed{\text{eq:dispersion}}$$

en dépendant des conditions aux bords (périodiques ou pas).

Le cas périodique Les dérivées spatiales et la linéarité de l'équation (12) nous motivent à mettre en œuvre une méthode spectrale de Fourier, ce qui est possible avec des conditions aux bords périodiques. En fait, la méthode est assez simple :

Soit $\hat{w}(k, t_n)$ les coefficients de Fourier de $w(x, t_n)$. La transformée de Fourier de l'équation (12) fournit

$$\hat{w}_t(k, t) - ik^3\hat{w}(k, t) = 0$$

C'est une EDO dans t , dont la solution est

$$\hat{w}(k, t) = e^{ik^3(t-t_n)}\hat{w}(k, t_n)$$

Finalement, la transformée inverse de Fourier utilisant les coefficients $\hat{w}(k, t_{n+1})$ donne $w(x, t_{n+1})$.

Le cas non périodique Dans ce cas, l'équation (12) sera résolue avec un schéma de différences finies implicite, avec une discrétisation de quatrième ordre de la troisième dérivée spatiale, sauf dans les points les plus proches des bords, pour lesquels un schéma décentré d'ordre 1 sera utilisé :

$$\begin{aligned} \frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{\frac{1}{8}u_{i-3}^{n+1} - u_{i-2}^{n+1} + \frac{13}{8}u_{i-1}^{n+1} - \frac{13}{8}u_{i+1}^{n+1} + u_{i+2}^{n+1} - \frac{1}{8}u_{i+3}^{n+1}}{\Delta x^3} &= 0, \quad i = 3, \dots, N-3 \\ \frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{-u_i^{n+1} + 3u_{i+1}^{n+1} - 3u_{i+2}^{n+1} + u_{i+3}^{n+1}}{\Delta x^3} &= 0, \quad i = 0, 1, 2 \\ \frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{u_i^{n+1} - 3u_{i-1}^{n+1} + 3u_{i-2}^{n+1} - u_{i-3}^{n+1}}{\Delta x^3} &= 0, \quad i = N-2, N-1, N \end{aligned}$$

ce qui ramène à la résolution d'un système linéaire, avec les modifications appropriées pour prendre en compte les conditions aux bords.

2.2.3 Choix de la taille de maille

Le pas de temps a été choisi en se basant sur le premier pas du schéma de *splitting*. Dans la forme non conservative, l'équation (11) peut être écrite comme

$$v_t + (1 + 2v)v_x = 0$$

ce qui s'agit d'un problème d'advection avec une vitesse $1 + 2v$.

Les pas spatial et temporal doivent vérifier la condition CFL :

$$(1 + 2v)\frac{\Delta t}{\Delta x} \leq 1$$

pour éviter des comportements non physiques (e.g. création de masse). Alors, dans chaque pas de temps, on va choisir, pur un ε petit :

$$\Delta t = \frac{\Delta x}{1 + 2 \max_x |v|} - \varepsilon$$

2.3 Analyse d'échelle

En envisageant la correcte simulation des phénomènes physiques qui jouent un rôle dans l'équation de KdV, la solution initiale doit satisfaire les hypothèses faites lors de la dérivation du modèle. Ayant cet objectif en tête, on fait des paragraphes suivants une analyse d'échelle en suivant les arguments présentés par [5]. Cette analyse nous permettra d'étudier la validité physique du modèle et de dériver un critère pour sélectionner les conditions initiales pour quelques exemples de simulations.

On veut écrire l'équation de KdV dans la forme suivante (sans dimension et échallonnée), selon la description de [5] :

$$U_T + U_X + \frac{\varepsilon}{2}(U^2)_X + \varepsilon\alpha^2 U_{XXX} = 0 \quad (13) \boxed{\text{eq:scaledKdV}}$$

et la lier aux paramètres du modèle d'ondes de surface dans forme dimensionnelle [17]

$$u_{t^*}^* + c_0 u_{x^*}^* + \frac{3}{4} \frac{c_0}{h_0} (u^{*2})_{x^*} + \frac{1}{6} c_0 h_0^2 u_{x^* x^* x^*}^* = 0$$

où le \cdot^* indique les variables physiques, h_0 la profondeur de l'eau non perturbée pour un fond plat, et $c_0 = \sqrt{gh_0}$ la vitesse d'onde longue.

2.3.1 Caractérisation de la non linéarité

D'après [5], la non linéarité est caractérisée par un paramètre ε tel que les caractéristiques sont écrites dans la forme

$$\frac{1}{c_0} \frac{dx}{dt} = 1 + bu$$

alors on peut choisir un $\varepsilon \ll 1$ tel que $bu = \varepsilon U$, avec u d'ordre 1. Pour le cas particulier d'ondes d'eau, ceci peut être représenté par

$$\frac{1}{c_0} \frac{dx}{dt} = 1 + \frac{3}{2h_0} u$$

alors

$$b = \frac{3}{2h_0} \quad (14) \boxed{\text{eq:analysisb}}$$

Ce paramètre sera utilisé dans les arguments suivants.

2.3.2 Caractérisation de la dispersion

La caractérisation de la dispersion vient de la dérivation de l'équation de KdV. D'après [5], si la propagation de l'onde suit une loi de la forme

$$u_{t^*}^* + u^* u_{x^*}^* + (\mathcal{L}u^*)_{x^*} = 0$$

avec \mathcal{L} tel que

$$\widehat{\mathcal{L}u^*} = \frac{c(k)}{c_0} \widehat{u^*}(k, t)$$

où $\widehat{\cdot}$ dénote la transformée de Fourier et $c(k)$ la célérité de phase, alors, pour κ suffisamment petit, la vitesse d'onde

$$c(\kappa) = c_0 + c_0 \sum_{n=1}^{\infty} A_n \varepsilon^n \kappa^{2n} \quad (15) \boxed{\text{eq:expansionc}}$$

peut être approchée par $c(\kappa) = c_0(1 - \kappa^2)$, ce qui motive les remplacements $X = \sqrt{\varepsilon}x^*$, $T = c_0\sqrt{\varepsilon}t^*$, et $u^* = \frac{\varepsilon}{b}U$ pour obtenir l'équation équivalente

$$U_T + \varepsilon U U_x + (\mathcal{L}_\varepsilon U)_X = 0 \quad (16) \boxed{\text{eq:scaledEquation}}$$

avec \mathcal{L}_ε tel que

$$\widehat{\mathcal{L}_\varepsilon U} = \frac{c(\varepsilon^{1/2}K)}{c_0} \widehat{U}(K, T) \quad (17) \boxed{\text{eq:operatorL}}$$

où $K = \sqrt{\varepsilon}k$. Après le remplacement de l'expansion (15) dans (17), on obtient

$$\mathcal{L}_\varepsilon U = U + \sum_{n=1}^{\infty} (-1)^n A_n \varepsilon^n \partial_X^{2n} U \quad (18) \boxed{\text{eq:expansionLe}}$$

et si les termes pour $n \geq 2$ sont négligeables (ce qui est le cas pour $\varepsilon \ll 1$) et si on suppose que toutes les dérivées de U sont de la même ordre de magnitude, alors on obtient

$$\begin{aligned} \mathcal{L}_\varepsilon U &= U + A_n \varepsilon \frac{\partial^2 U}{\partial x^2} \\ &= U - \alpha^2 \varepsilon \frac{\partial^2 U}{\partial x^2} \end{aligned}$$

avec $\alpha^2 = -A_n$. En remplaçant dans l'équation échelonnée (16), on obtient

$$U_T + U_X + \frac{\varepsilon}{2} (U^2)_X + \varepsilon \alpha^2 U_{XXX} = 0$$

L'application du même échelonnement $X = \sqrt{\varepsilon}x^*$, $T = c_0 \sqrt{\varepsilon}t^*$, et $u^* = \frac{\varepsilon}{b} U$ à l'équation physique fournit

$$U_T + U_X + \frac{3\varepsilon}{4h_0 b} (U^2)_X + \frac{h_0^2 \varepsilon}{6} U_{XXX} = 0$$

d'où, en comparant avec (13), on conclue que $\alpha^2 = \frac{h_0^2}{6}$.

2.4 Le critère pour choisir une solution initiale appropriée

En se basant sur l'analyse d'échelle faite ci-dessus, on va proposer un critère pour choisir des conditions initiales (i.e., la longueur et l'amplitude de l'onde initiale) pour lesquelles l'équation de KdV est physiquement valide.

2.4.1 Choix de la longueur d'onde

Une condition suffisante pour que les termes d'ordre supérieur à 1 dans l'expansion en série de puissances de \mathcal{L}_ε (équation 18) soient négligeables est que ces termes soient aussi négligeables pour $c(k)$ (dans l'équation 15), étant

donné que toutes les dérivées de U aient ordre de magnitude 1 (une assumption faite par [5]).

D'après [5], l'expression suivante est applicable à des ondes de surface :

$$c(\kappa) = c_0 \left(\frac{\tanh(\kappa h_0)}{\kappa h_0} \right) = c_0 \left(1 - \frac{1}{6}(\kappa h_0)^2 + \frac{19}{360}(\kappa h_0)^2 + \dots \right)$$

d'où on voit qu'il faut choisir $\kappa h_0 \ll 1$ pour que les dérivées d'ordre $n > 1$ soient négligeables.

En dénotant la longueur d'onde par λ , et en choisissant une constante B telle que $\kappa h_0 = B \ll 1$, on a que $h_0 = \frac{B\lambda}{2\pi}$, et, de la relation $\alpha^2 = \frac{h_0^2}{6}$, on obtient $\alpha^2 = \frac{B^2\lambda^2}{6(2\pi)^2}$.

2.4.2 Choix de l'amplitude de l'onde

Du changement de variables $bu^* = \varepsilon U$, avec U d'ordre de magnitude 1, et en considérant $b = \frac{3}{2h_0}$ (expression 14), la variable physique est écrite comme $u^* = \frac{2}{3}h_0\varepsilon U$, ($\varepsilon > 0$). Alors

$$A = \frac{2}{3}\varepsilon h_0$$

est l'amplitude de l'onde. La détermination de l'amplitude est fait en considérant $\varepsilon \ll 1$.

2.4.3 Résumé

Alors, le critère proposé ici pour construire la condition initial pour l'équation de KdV est

1. Adopter une profondeur h_0 (e.g. à partir du data disponible);
2. Choisir une amplitude d'onde $A = \frac{2}{3}h_0\varepsilon$, alors la restriction $\varepsilon \ll 1$ ramène à $\frac{A}{h_0} = \frac{2}{3}\varepsilon \ll 1$.
3. Choisir une longueur d'onde λ telle que $\kappa h_0 = \frac{2\pi}{\lambda}h_0 = B \ll 1$, ce qui ramène à $\frac{h_0}{\lambda} = \frac{B}{2\pi} \ll 1$
4. L'effet de dispersion sur la non linéarité peut être mesurée par le quotient des respectifs coefficients, $\frac{\varepsilon\alpha^2}{\varepsilon} = \alpha^2$.

D'un autre point de vue, on peut définir une onde d'amplitude A et longueur λ et déterminer l'intervalle de profondeurs dans lequel cette condition initiale est valide, pour une précision (ε, B) donnée :

$$h_0^{\text{valid}} = \left[\frac{3A}{2\varepsilon}, \frac{B\lambda}{2\pi} \right] \quad (19) \boxed{\text{eq:hvalid}}$$

ce qui est consistant avec le fait de que le modèle de KdV est valide pour des ondes de petite amplitude et grande longueur [5].

2.5 Tests numériques

On présente ci-dessous deux exemples pour tester la résolution numérique proposée pour l'équation de KdV. Ces exemples sont inspirés dans ceux présentés dans [24]: les solutions initiales sont des gaussiennes (suffisamment lointaines des bords afin d'éviter son influence) avec des différentes amplitudes et longueurs (la longueur d'onde est adoptée comme étant l'écart standard). L'idée est de vérifier l'influence de ces caractéristiques sur les effets non linéaires et dispersives, et aussi de vérifier le range de profondeur d'eau dans lequel la propagation de chaque solution peut être modelée par l'équation de KdV (selon l'expression 19). Les deux tests ont été faits en considérant $B = 0.1$ et $\varepsilon = 0.001$.

Les solutions initiales utilisées sont

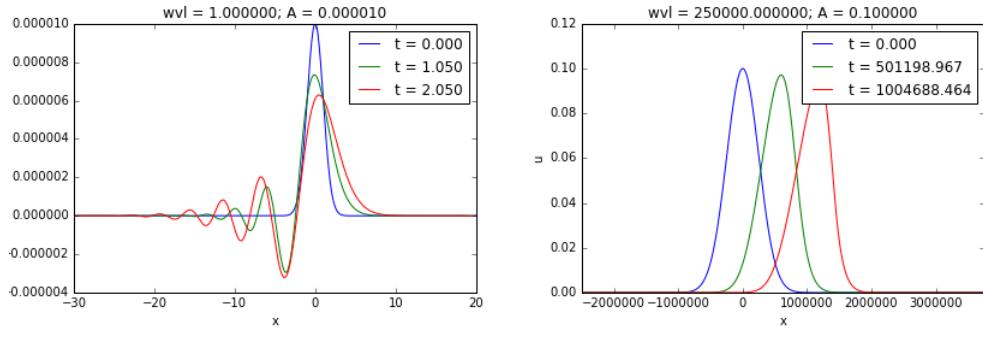
1. Onde courte (figure 20a)

- $\lambda = 1$
- $A = 10^{-5}$
- $h_0^{\text{valid}} = [0.15, 0.16]$

2. Onde longue (figure 20b)

- $\lambda = 250000$
- $A = 0.1$
- $h_0^{\text{valid}} = [150, 3979]$

La figure 20 présente les solutions dans quelques instants de simulation :



(a) Onde gaussienne courte

(fig:KdVcriteriaShort)

(b) Onde gaussienne longue

(fig:KdVcriteriaLong)

Figure 20: Simulations avec l'équation de KdV

(fig:KdVcriteria)

Conclusions partiales :

- Les résultats obtenus sont cohérents avec les observations et exemples faits par [24], qui affirme que les effets dispersifs sont plus fortes dans des ondes plus courtes; pour des ondes plus longues, les effts non linéaires sont plus évidents.
- Le range de validité de l'équation de KdV est très petit dans le cas de l'onde courte, et beaucoup plus grand dans le cas de l'onde large, ce qui est cohérent avec la validité du modèle de KdV pour les ondes de petite amplitude et grande longueur. Cela peut être observé dans la définition de h_0^{valid} (équation (19)) : en réduisant A et augmentant λ , la taille du range de validité augmente.
- Néanmoins, on n'a pas validé une des conclusions qu'on a faites lors de la dérivation du critère proposé. On a affirmé que l'importance relative des effets dispersifs et non linéaires peut être mesurée par $\alpha^2 = \frac{h_0^2}{6}$, mais, dans les exemples testés, si on adopte h_0 comme étant la médiane de h_0^{valid} , l'onde courte (qui est très dispersive) a un α^2 plus petite que l'onde large.

3 Les équations de Serre

?<sec:Serre>? 3.1 Le modèle

Les équations de Serre constituent un modèle qui décrit la propagation d'ondes fortement non linéaires dans des eaux peu profondes. En considérant un fond plat, ces équations s'écrivent sous la forme suivante, pour les variables (h, u) :

$$\begin{cases} h_t + (hu)_x = 0 \\ u_t + uu_x + gh_x - \frac{1}{3h} (h^3 (u_{xt} + uu_{xx} - (u_x)^2))_x = 0 \end{cases} \quad (20) \boxed{\text{eq:serrehu}}$$

où $u = u(x, t)$, $h = h(x, t)$ et g sont, respectivement, la vitesse horizontal moyennée au long de la profondeur, la profondeur de l'eau et l'accélération de la gravité. Cette formulation s'est basée sur [8].

3.1.1 Les équations de Serre dans les variables (h, hu)

Afin de mettre en œuvre une méthode de *splitting* pour la résolution numérique de les équations de Serre (ainsi comme il est fait dans [7] pour les équations de Green-Naghdi, qui sont la version en 2D de les équations de Serre), on va réécrire le système (20) dans les variables (h, hu) , afin d'avoir une formulation analogue à celle utilisé dans ce papier.

On va utiliser les identités

$$(hu^2)_x = (huu)_x = (hu)_x u + huu_x$$

et, de la première équation de (20),

$$hu_t = (hu)_t - h_t u = (hu)_t + (hu)_x u$$

En multipliant la deuxième équation de (20) par h , on obtient

$$\begin{aligned} hu_t + huu_x + ghh_x - \frac{1}{3} (h^3 (u_{xt} + uu_{xx} - (u_x)^2))_x &= 0 \\ \implies (hu)_t + (hu)_x u + (hu^2)_x - (hu)_x u + ghh_x - \frac{1}{3} (h^3 (u_{xt} + (uu_x)_x - 2(u_x)^2))_x &= 0 \\ \implies (hu)_t + (hu^2)_x + ghh_x - \frac{1}{3} (h^3 (u_{xt} + (uu_x)_x - 2(u_x)^2))_x &= 0 \end{aligned} \quad (21) \boxed{\text{eq:serreTimesh}}$$

On remarque que

$$\begin{aligned} u_{xt} &= \left(\frac{1}{h} (hu) \right)_{xt} = \left(-\frac{h_t}{h^2} (hu) + \frac{1}{h} (hu)_t \right)_x = \left(-\frac{h_t u}{h} + \frac{1}{h} (hu)_t \right)_x = \\ &\quad \left(\frac{1}{h} ((hu)_x u) \right)_x + \left(\frac{1}{h} (hu)_t \right)_x = 0 \end{aligned}$$

et

$$(uu_x)_x = \left(\frac{1}{h} (huu_x) \right)_x = \left(\frac{1}{h} ((hu^2)_x - (hu)_x u) \right)_x = \left(\frac{1}{h} (hu^2)_x \right)_x - \left(\frac{1}{h} (hu)_x u \right)_x$$

alors, dans (21), on obtient

$$(hu)_t + (hu^2)_x + ghh_x - \frac{1}{3} \left[h^3 \left(\left(\frac{1}{h}(hu)_t \right)_x + \left(\frac{1}{h}(hu^2)_x \right)_x - 2(u_x)^2 \right) \right]_x = 0$$

Cette dernier équation peut être écrite sous la forme

$$\begin{aligned} & \left(I + h\mathcal{T} \frac{1}{h} \right) (hu)_t + \left(I + h\mathcal{T} \frac{1}{h} \right) (hu^2)_x + ghh_x + h\mathcal{Q}_1(u) + \\ & \quad \left(I + h\mathcal{T} \frac{1}{h} \right) (ghh_x) - \left(I + h\mathcal{T} \frac{1}{h} \right) (ghh_x) = 0 \end{aligned}$$

où le terme $\left(I + h\mathcal{T} \frac{1}{h} \right) (ghh_x)$ a été ajouté et soustrait pour avoir une formulation équivalente à de [7]. Les opérateurs \mathcal{T} et \mathcal{Q} , aussi définis par [7], sont donnés par

$$\begin{aligned} \mathcal{T}(w) &= -\frac{1}{3h}(h^3 w_x)_x = -\frac{h^2}{3} w_{xx} - hh_x w_x \\ \mathcal{Q}(w) &= \frac{2}{3h}(h^3(w_x)^2)_x = \frac{4h^2}{3}(w_x w_{xx}) + 2hh_x(w_x)^2 \end{aligned}$$

Ainsi, le système qu'on va résoudre est

$$\left\{ \begin{array}{l} h_t + (hu)_x = 0 \\ \left(I + h\mathcal{T} \frac{1}{h} \right) (hu)_t + \left(I + h\mathcal{T} \frac{1}{h} \right) (hu^2)_x + ghh_x + h\mathcal{Q}_1(u) + \\ \quad \left(I + h\mathcal{T} \frac{1}{h} \right) (ghh_x) - \left(I + h\mathcal{T} \frac{1}{h} \right) (ghh_x) = 0 \end{array} \right. \quad (22) \boxed{\text{eq:serrehhu}}$$

3.2 Discréétisation

Comme on a fait antérieurement pour la résolution numérique des équations de KdV et BBM, les équations de Serre (22) seront numériquement résolues en utilisant une méthode de *splitting*, dans laquelle le système d'équations sera décomposé dans deux parties: la première contiendra les termes d'advection, et la deuxième, les dérivées d'ordre supérieur.

Alors, la résolution numérique demandera la résolution, dans chaque pas de temps $[t_n, t_{n+1}]$, du problème suivant :

$$\begin{cases} \tilde{h}_t + (\tilde{h}\tilde{u})_x = 0 \\ (\tilde{h}\tilde{u})_t + (\tilde{h}\tilde{u}^2)_x + g\tilde{h}\tilde{h}_x = 0 \end{cases} \quad (23) \boxed{\text{eq:splitSerre1}}$$

$$\begin{cases} \bar{h}_t = 0 \\ (\bar{h}\bar{u})_t - g\bar{h}\bar{h}_x + (I + h\mathcal{T}\frac{1}{h})^{-1} [g\bar{h}\bar{h}_x + \bar{h}\mathcal{Q}_1(\bar{u})] = 0 \\ (h, u)(x, t_{n+1}) = (\bar{h}, \bar{u})(x, t_{n+1}) \end{cases} \quad (24) \boxed{\text{eq:splitSerre2}}$$

En dénotant les systèmes (23) et (24) par les opérateurs $T_a^{\Delta t}$ et $T_d^{\Delta t}$, respectivement, où le superindice indique que l'opérateur est appliqué sur un pas de temps Δt , le problème peut s'écrire comme :

$$(h, u)(x, t_{n+1}) = T_d^{\Delta t} (T_a^{\Delta t} ((h, u)(x, t_n)))$$

Quelques variations de ce schéma de *splitting* seront également implémentés. Par exemple, en inversant l'ordre des opérateurs; ou encore la méthode connue comme "*Strang splitting*", où trois problèmes sont résolus dans chaque pas de temps :

$$(h, u)(x, t_{n+1}) = T_a^{\frac{\Delta t}{2}} \left(T_d^{\Delta t} \left(T_a^{\frac{\Delta t}{2}} (h, u)(x, t_n) \right) \right)$$

Dans la suite, le tilde et la barre supérieur seront supprimées pour clarifier la notation.

3.2.1 Première système d'équations (pas d'advection)

La première partie des équations de Serre correspond au *Non linear Shallow Water equation* (NSWE), qui, en tenant compte que

$$\begin{aligned} (hu)_t &= uh_t + hu_t = -u(hu)_x - h(uu_x + gh_x) \\ &= -u(h_xu + 2hu_x) - ghh_x \\ &= -(hu^2)_x - \frac{1}{2}g(h^2)_x = -\left(hu^2 + \frac{1}{2}gh^2\right)_x \end{aligned}$$

peut être écrit comme une loi de conservation sous la forme

$$U_t + F(U)_x = 0 \quad (25) \boxed{\text{?serre:conservative}}$$

Alors, on peut résoudre la première pas du *splitting* en utilisant un schéma de volumes finies.

Décrire de façon très résumé.

3.2.2 Deuxième système d'équations (pas de dispersion)

Dans le deuxième système (24) des équations de Serre splittées, la profondeur d'eau h est constante en temps, et par conséquent seulement la vitesse u doit être mise à jour. La résolution numérique proposée consiste en résoudre, à chaque pas de temps, le système linéaire

$$\left(I + h\mathcal{T} \frac{1}{h} \right)^{-1} [hh_x + h\mathcal{Q}_1(u)] = z \implies \left(I + h\mathcal{T} \frac{1}{h} \right) z = hh_x + h\mathcal{Q}_1(u) \quad (26) \boxed{\text{eq:systemhu}}$$

Le côté à gauche de (26) s'écrit

$$\begin{aligned} \left(I + h\mathcal{T} \frac{1}{h} \right) z &= z - \frac{h^3}{3} \left(\frac{1}{h} z \right)_{xx} - h^2 h_x \left(\frac{1}{h} z \right)_x = \\ &= z - \frac{h^3}{3} \left[\left(2 \frac{(h_x)^2}{h^3} - \frac{h_{xx}}{h^2} \right) z - 2 \frac{h_x}{h^2} z_x + \frac{z_{xx}}{h} \right] - h^2 h_x \left[-\frac{h_x}{h^2} z + \frac{z_x}{h} \right] = \\ &= \left(1 + \frac{1}{3}(h_x)^2 + \frac{1}{3}hh_{xx} \right) z - \left(\frac{1}{3}hh_x \right) z_x - \left(\frac{1}{3}h^2 \right) z_{xx} \end{aligned}$$

En utilisant des différences finies d'ordre quatre pour discréteriser les dérivées spatiales de z , on résoudre, pour chaque $i = 1, \dots, N-1$ dans le pas de temps t_n :

$$\begin{aligned} &\left(1 + \frac{1}{3}((h_x)_i^n)^2 + \frac{1}{3}h_i^n(h_{xx})_i^n + \frac{1}{\Delta x^2} \frac{5}{6}(h_i^n)^2 \right) z_i^n + \\ &\frac{1}{3} \left(-\frac{2}{3} \frac{h_i^n(h_x)_i^n}{\Delta x} - \frac{4}{3} \frac{(h_i^n)^2}{\Delta x^2} \right) z_{i+1}^n + \frac{1}{3} \left(\frac{2}{3} \frac{h_i^n(h_x)_i^n}{\Delta x} - \frac{4}{3} \frac{(h_i^n)^2}{\Delta x^2} \right) z_{i-1}^n + \\ &\frac{1}{36} \left(\frac{h_i^n(h_x)_i^n}{\Delta x} + \frac{4}{3} \frac{(h_i^n)^2}{\Delta x^2} \right) z_{i+2}^n + \frac{1}{36} \left(-\frac{h_i^n(h_x)_i^n}{\Delta x} + \frac{4}{3} \frac{(h_i^n)^2}{\Delta x^2} \right) z_{i-2}^n = \\ &h_i^n(h_x)_i^n + h_i^n(\mathcal{Q}_1(u))_i^n \end{aligned}$$

Ainsi, pour chaque $i = 1, \dots, N-1$, la solution est mise à jour en temps selon l'expression

$$(hu)_i^{n+1} = (hu)_i^n + \Delta t (gh_i^n(h_x)_i^n - z_i^n)$$

3.3 Tests numériques

3.3.1 Description de la solution initiale

Afin de valider l'implémentation des équations de Serre, on les résoudra en utilisant la solution analytique comme solution initiale. D'après [8], les équations de Serre admettent la famille de solutions périodiques suivante :

$$h(x, t) = a_0 + a_1 dn^2(\kappa(x - ct), k), \quad u(x, t) = c \left(1 - \frac{h_0}{h(x, t)}\right)$$

$$\kappa = \frac{\sqrt{3a_1}}{2\sqrt{a_0(a_0 + a_1)(a_0 + (1 - k^2)a_1)}}, \quad c = \frac{\sqrt{ga_0(a_0 + a_1)(a_0 + (1 - k^2)a_1)}}{h_0}$$

avec $k \in (0, 1)$, $a_0 > 0$, $a_1 > 0$ et $dn(\cdot, k)$ une fonction elliptique de Jacobi de module k .

La relation entre la longueur d'onde λ et $k \in (0, 1)$ est

$$\lambda = \frac{2K(k)}{\kappa}$$

et la profondeur moyenne de l'eau, h_0 , es calculée à partir de

$$h_0 = \frac{1}{\lambda} \int_0^\lambda h(x, t) dx = a_0 + a_1 \frac{E(k)}{K(k)}$$

où $K(k)$ et $E(k)$ sont des intégrales elliptiques complètes de premier et deuxième types.

La limite pour $k \rightarrow 0^+$ est le niveau constant de l'eau $a_0 + a_1$ en équilibre. Si $k \rightarrow 1^-$, (h, u) converge vers la solution de Rayleigh (onde solitaire). On testera aussi ce dernier cas, dans lequel la solution est décrite par

$$h(x, t) = a_0 + a_1 sech^2(\kappa(x - ct), k), \quad u(x, t) = c \left(1 - \frac{a_0}{h(x, t)}\right)$$

$$\kappa = \frac{\sqrt{3a_1}}{2\sqrt{a_0(a_0 + a_1)}}, \quad c = \sqrt{ga_0(a_0 + a_1)}$$

Les expressions para la longueur d'onde λ et la profondeur moyenne de l'eau h_0 sont les mêmes que pour le cas général de la solution cnoïdale.

3.3.2 Results

With the objective to observe the nonlinear and the dispersive processes in the model, we solved the Serre equation and the Nonlinear Shallow Water Equation (NSWE), which is the first step of the proposed split scheme. The figures 21 and 22 shows the evolution of (h, u) for the cnoidal solution; and the figures 23 and 24 for the solitary solution. In this last case, we also solved the problem with a first order finite volume solver for the resolution of the first step of the Serre equation.

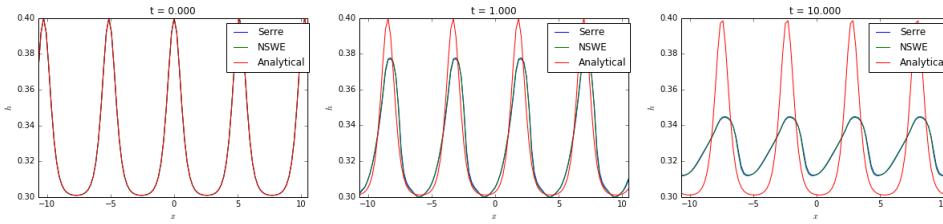


Figure 21: Evolution of h for the cnoidal solution in the Serre equation. Comparison between the analytical solution (in red) and the solutions (practically overlapped) computed with the Serre (in blue) and the NSWE (in green) models

`(fig:cnoidalh)`

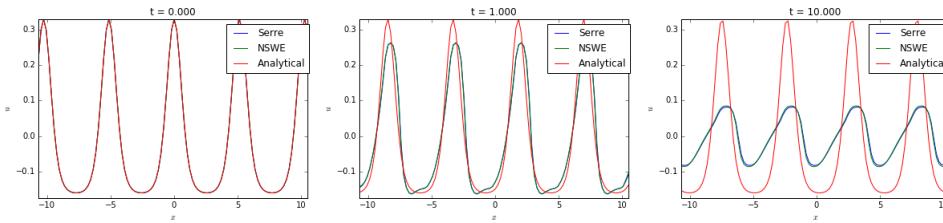


Figure 22: Evolution of u for the cnoidal solution. Comparison between the analytical solution (in red) and the solutions (practically overlapped) computed with the Serre (in blue) and the NSWE (in green) models

`(fig:cnoidalu)`

The results show the existence of modeling or programming errors. In both cases tested, the analytical solution is not preserved : we observe a strong dissipation of the solution, and, in the solitary wave case, an inversion of the velocity that causes the formation of secondary waves. The utilization of a higher-order solver for the Finite Volume scheme did not correct this last problem, but showed a lower dissipation.

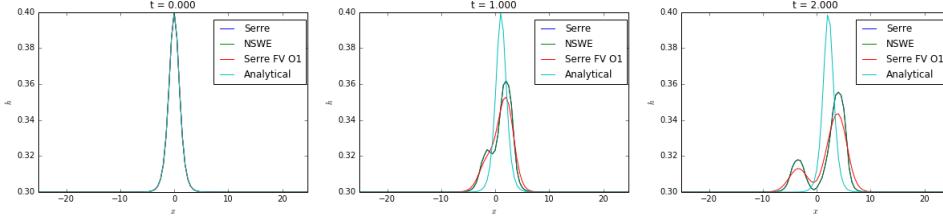


Figure 23: Evolution of h for the solitary solution. Comparison between the analytical solution (in light blue) and the solutions computed with the Serre model with first order resolution for the finite volume scheme (in red), the Serre model with second order resolution for the finite volume scheme (in dark blue) and the second order NSWE model (in green). The last two solutions are practically overlapped

`(fig:solitaryh)`

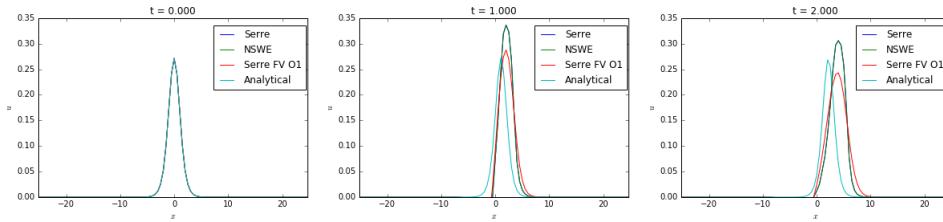


Figure 24: Evolution of u for the solitary solution. Comparison between the analytical solution (in light blue) and the solutions computed with the Serre model with first order resolution for the finite volume scheme (in red), the Serre model with second order resolution for the finite volume scheme (in dark blue) and the second order NSWE model (in green). The last two solutions are practically overlapped

`(fig:solitaryu)`

4 Premier étude des conditions aux limites transparentes

?<sec:TBC>? 4.1 Introduction et quelques exemples de motivation

Le contenu présenté dans cette section est une introduction aux objectifs qu'on envisage dans ce stage, concernant l'étude des conditions aux limites transparentes (en anglais, *transparent boundary conditions* - TBCs) afin de les appliquer à des méthodes de décomposition de domaine. Cet étude a été développé sur l'équation de KdV, parce que c'était, parmi les modèles étudiés et implémentés jusqu'au moment où on a commencé cette partie du projet, celui pour lequel on a obtenu la meilleure et mas fiable implémentation numérique. En plus, la linéarisation de l'équation de KdV est beaucoup plus évidente que pour les équations de Serre - comme on discutera plus tard, l'étude des TBCs pour des équations linéaires est beaucoup plus développé que pour les non linéaires.

Les TBCs sons construites de façon que la solution calculée dans le do-

maine comptuationnel fini Ω coïncide avec la solution du problème dans tout l'espace, restreinte à Ω . En général, ces conditions aux bords son non locales en temps, alors elles doivent être approximées pour permettre une implémentation numérique efficiente [1].

Les TBCs exactes pour le problème

$$\begin{cases} \mathcal{A}(u) = f & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

où \mathcal{A} est un opérateur différentiel partiel, sont données par [15]

$$B_i(u) = \frac{\partial}{\partial n_i} u + D2N(u) = 0 \quad (27) \boxed{\text{eq:exactTBC}}$$

où ∂n_i est le vecteur normal sortant à Ω_i sur Γ , et l'opérateur D2N (*Dirichlet to Neumann*) est défini par

$$D2N : \alpha(x) \mapsto \left. \frac{\partial}{\partial n_i^c} v \right|_{\Gamma}$$

avec α défini sur Γ . v est solution du problème suivant, résolu dans le complémentaire de Ω_i , dénoté Ω_i^c :

$$\begin{cases} \mathcal{A}(v) = f & \text{in } \Omega_i^c \\ v = 0 & \text{on } \partial\Omega_i \setminus \Gamma \\ v = \alpha & \text{on } \Gamma \end{cases}$$

Ainsi, on peut interpréter l'opérateur D2N comme une imposition de continuité de la dérivée de la solution dans la direction normale à l'interface. Par ailleurs, comme son nom indique, cet opérateur construit une dérivée (une condition de Neumann) à partir de la solution dans Ω_i^c (une condition de Dirichlet).

Présentons d'abord un exemple simple, avec de solution y TBCs analytiques connues. On va considérer le problème 1D suivant :

$$\begin{cases} -u''(x) = 1 & \text{in } \Omega = [0, 2] \\ u(0) = 0 \\ u(2) = 0 \end{cases}$$

dont la solution est

$$u(x) = -\frac{x^2}{2} + x$$

et, en considérant la partition de Ω en $\Omega_1 = [0, 1]$ et $\Omega_2 = [1, 2]$, on va considérer le problème

$$\begin{cases} -u_1''(x) = 1 & \text{in } \Omega_1 \\ u_1(0) = 0 \\ \mathcal{B}(u_1) = 0 & \text{at } \Gamma = \{1\} \end{cases}$$

où la TBC $\mathcal{B}(u)$ est telle que $u|_{\Omega_1} = u_1$.

La fonction v , pour la détermination de l'opérateur D2N, est solution de

$$\begin{cases} -v''(x) = 1 & \text{in } \Omega_2 \\ v(2) = 0 \\ v(1) = \alpha & \text{at } \Gamma = \{1\} \end{cases}$$

alors

$$v(x) = -\frac{x^2}{2} + \left(\frac{3}{2} - \alpha\right)x + 2\alpha - 1$$

et

$$\left. \frac{\partial}{\partial x} v \right|_{x=1} = \frac{1}{2} - \alpha$$

Finalement, le TBC est

$$B(u_1) = \frac{\partial}{\partial x} u_1 + D2N(u_1) = \frac{\partial}{\partial x} u_1 + \frac{1}{2} - u_1$$

Le problème a été résolu avec une méthode de différences finies, avec des approximations centrées de seconde ordre pour la dérivée spatial, dans deux grilles différentes. La figure 25 montre que le TBC construit fournit effectivement une solution convergente à la solution de référence restreinte à Ω .

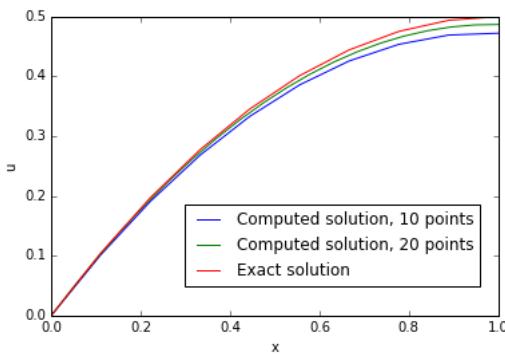


Figure 25: Solutions pour l'équation de Laplace avec TBC

(fig:TBClaplace)

En retournant aux modèles d'ondes, on rappelle que, dans les premières simulations avec les méthodes de *splitting* adoptées pour la résolution de l'équation de KdV, on n'a pas utilisé une application rigoureuse de conditions aux bords appropriées. En fait, notre objectif principal était de valider la méthode; alors, en imposant des conditions périodiques ou des conditions de Dirichlet ou Neumann homogènes, on a analysé l'évolution de la solution seulement avant son arrivée aux bords.

Avant de commencer l'étude des TBCs pour l'équation de KdV, on va présenter deux exemples de motivation à ce travail. Le premier exemple montre très clairement l'influence de conditions aux bords inappropriées sur la solution. On a résolu deux fois le même problème, avec la même solution initiale, conditions aux bords et discréétisations spatiales et temporales :

$$\begin{cases} u_t + u_x + (u^2)_x + u_{xxx} = 0, & x \in \Omega = [a, b] \quad t \in [0, t_{max}] \\ u(x, 0) = \Phi(x) \\ u(a, t) = 0 \\ u(b, t) = 0 \\ u_x(b, t) = 0 \end{cases}$$

La seule différence entre les deux problèmes est la taille de ses domaines: ils ont été choisis de façon que l'onde arrive aux bords (dans le temps de simulation) dans le première problème, mais pas dans le deuxième. La figure 26 montre comme la différence entre les deux solutions augmente avec le temps, en partant du bord et se propageant pour tout le domaine:

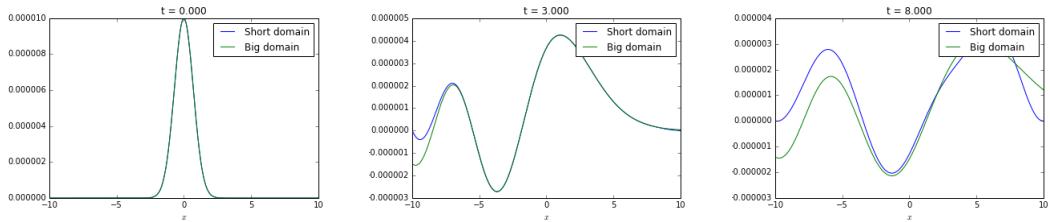


Figure 26: Premier exemple de motivation : comparaison entre les solutions dans un petit et un large domaines

(fig:motivational1) Alors, on va chercher des conditions aux bords qui puissent bien simuler les TBCs, c'est-à-dire, de façon que la solution calculé dans Ω soit le même que la solution de tout le domaine restreint à Ω . Par conséquent, on veut des bords qui n'ont pas d'influence sur la solution, de façon que, quand l'onde arrive au bord, elle puisse simplement "sortir" du domaine.

L'exemple suivant montre une deuxième motivation pour ce travail. On veut résoudre le problème

$$(P_1) \begin{cases} u_t + u_x + (u^2)_x + u_{xxx} = 0, & x \in \Omega_1 = [0, L], \quad t \in [0, t_{max}] \\ u(x, 0) = \Phi(x) \\ u(0, t) = 0 \\ u_x(0, t) = 0 \\ u(L, t) = g(t) \end{cases}$$

On cherche une fonction $g(t)$ pour simuler le TBC. Pour atteindre cet objectif, on va d'abord résoudre le problème

$$(P_2) \begin{cases} u_t + u_x + (u^2)_x + u_{xxx} = 0, & x \in \Omega_2 = [0, 2L], \quad t \in [0, t_{max}] \\ u(x, 0) = \Phi(x) \\ u(0, t) = 0 \\ u_x(0, t) = 0 \\ u(2L, t) = 0 \end{cases}$$

et on impose $g(t) = u_2(t)$, où u_2 est la solution de (P_2) . Afin d'obtenir des résultats plus précis, les deux calculs sont réalisés avec les mêmes pas de temps et taille du maillage.

Supposons qu'il y a une unique solution u_1 pour (P_1) . On peut facilement voir que $u_2|_{\Omega_1}$ est également solution de (P_1) . Alors, $u_1 = u_2|_{\Omega_1}$. Ce fait justifie pourquoi notre procédure marche comme une TBC, en fournissant la solution "exacte", comme montre la figure 27 (détail sur la région proche du bord droit).

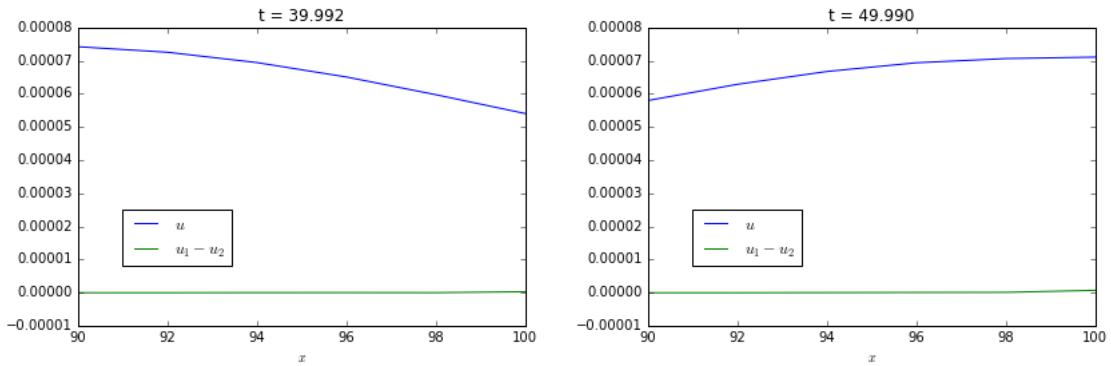


Figure 27: Deuxième exemple de motivation : solution avec une condition de Dirichlet "exacte" sur le bord à droite

`(fig:motivation2)`

4.2 Optimisation de conditions aux limites de Robin pour simuler des TBCs

Malgré le très bon résultat présenté dans le dernier exemple, on ne peut pas appliquer cette procédure en pratique. En fait, calculer la solution dans un domaine plus grand et l'utiliser comme solution exacte pour construire les conditions aux bords n'est qu'une "triche". Alors, on veut plutôt déterminer des approximations pour les TBCs sans avoir une solution de référence.

4.2.1 Conditions aux limites de Robin jusqu'à la dérivée première

Dans un approche initial, l'équation de KdV sera résolue dans le domaine $[-L, L]$ avec les conditions aux bords suivants (imposés dans la résolution du deuxième pas de la méthode de *splitting*):

$$\begin{cases} u(-L) = 0 \\ u_x(-L) = 0 \\ \alpha u(L) + \beta u_x(L) = 0, \quad \alpha, \beta > 0 \end{cases}$$

La troisième condition aux bords consiste en des conditions de Robin, avec des paramètres α et β (ou, de façon équivalente, le paramètre β/α) qui seront optimisés afin de simuler une TBC sur le bord à droite. Dans un premier moment, on va considérer des conditions de Robin contenant jusqu'à la première dérivée de la solution.

Afin de trouver les coefficients optimaux, on va tester plusieurs paires $(1, \beta/\alpha)$ (y compris les limites $\beta/\alpha \rightarrow 0$ et $\beta/\alpha \rightarrow \infty$, correspondant respectivement aux conditions de Dirichlet et de Neumann) et calculer l'erreur par rapport à la solution de référence u_{ref} , calculée dans le domaine $[-2L, 2L]$. Deux erreurs seront calculées, pour chaque pas de temps t_n :

$$e_1^n = \sqrt{\sum_{i=0}^N (u_i^n - (u_{ref})_i^n)^2}$$

$$e_2^n = u_N^n - (u_{ref})_N^n$$

e_2^n est calculé pour montrer que la plus grande partie de l'erreur e_1^n de tout le domaine est due aux bords.

Les figures 28 à 31 montrent la solution dans quelques instants et l'évolution de e_1 et e_2 pour certains valeurs de β/α . La figure 32 compare e_2 pour plusieurs autres valeurs, y compris le cas de pure Dirichlet (avec $\alpha = 1, \beta = 0$, alors $\log(\beta/\alpha) = -\infty$) et pure Neumann (avec $\alpha = 0, \beta = 1$).

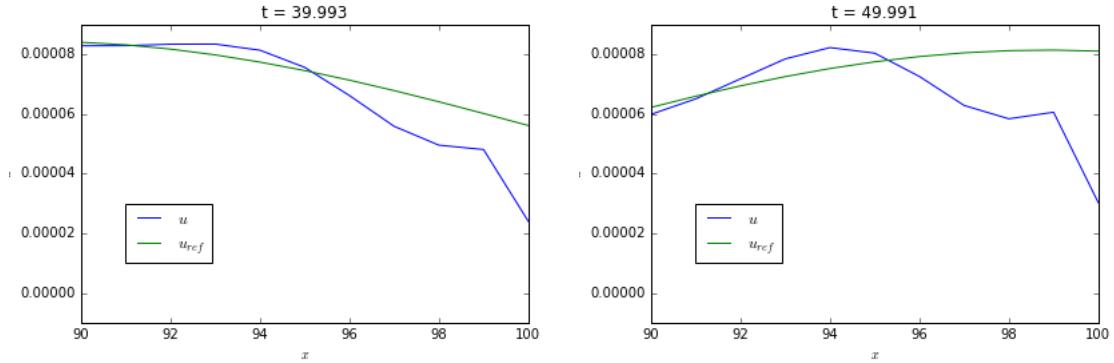


Figure 28: Solution de référence et solution approximée pour $\beta/\alpha = 1$

`(fig:robin1)`

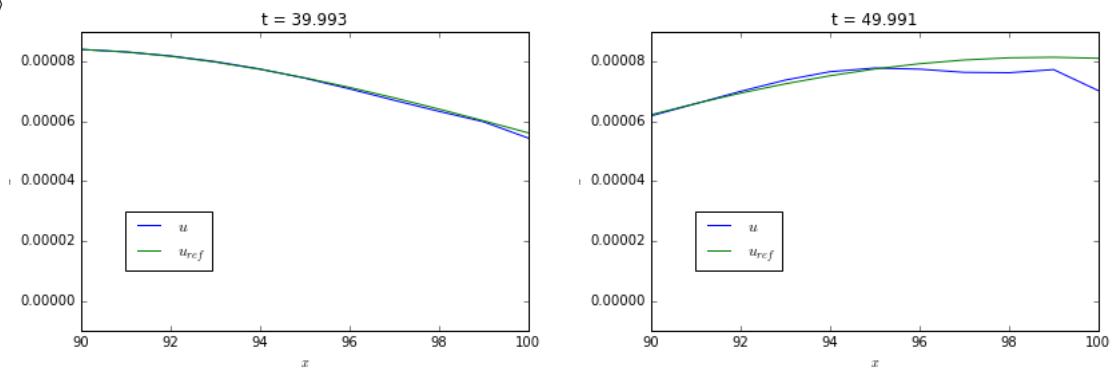


Figure 29: Solution de référence et solution approximée pour $\beta/\alpha = 10$

`?(fig:robin10)?`

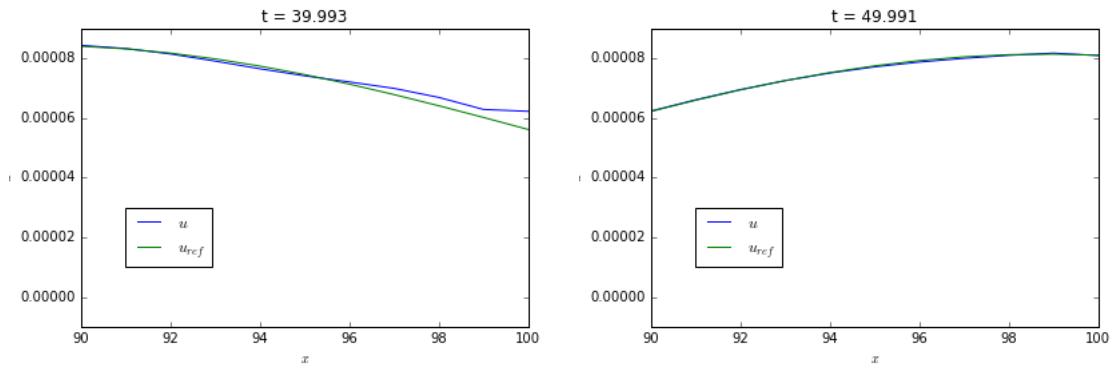


Figure 30: Solution de référence et solution approximée pour $\beta/\alpha = 100$

`?(fig:robin100)?`

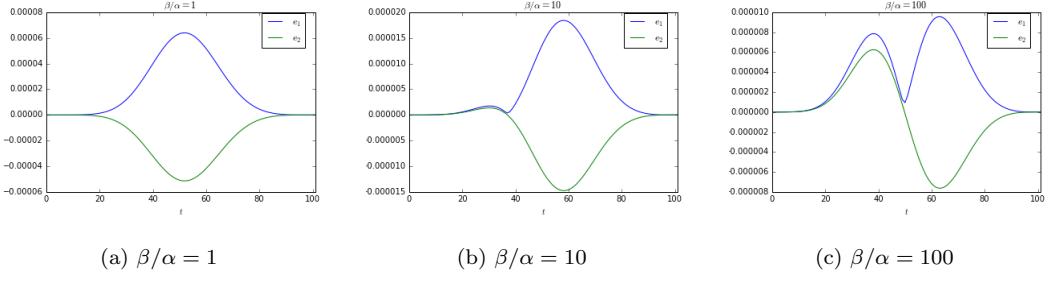


Figure 31: Erreurs entre la solution approximée et la solution de référence

:robinErrorsExample)

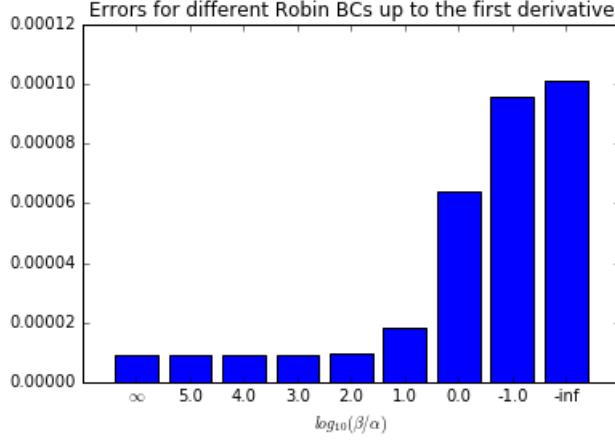


Figure 32: Erreur $\|e_1\| = \sum_{n=0}^{n_{max}} (e_1^n)^2$ entre la solution approximée et la solution de référence pour des plusieurs valeurs de β/α , avec des conditions aux limites de Robin jusqu'à la dérivée première

:fig:robinErrorsAll)

Les résultats présentés dans les figures 28 jusqu'à 32 montrent que des conditions aux bord avec un caractère de Neumann plus fort produisent des meilleures approximations pour le TBC, en comparaison avec des conditions plus proches du pure Dirichlet. Les résultats pour le pure Neumann et pour le Neumann avec un petit mais pas nul Dirichlet terme sont très proches, comme présenté dans le tableau 2 pour un étude plus raffiné autour des meilleures valeurs pour β/α . En fait, imposer la solution nulle aux bord est une condition trop forte, et la condition de Neumann peut capturer de façon plus satisfaisant la lisséité de la solution.

$\log(\beta/\alpha)$	Error ($\times 10^{-6}$)
2.5	8.93
3.0	8.87
3.5	8.95
4.0	8.98
4.5	8.99
5.0	8.99
∞	8.99

Table 2: Erreur $\|e_1\| = \sum_{n=0}^{n_{max}} (e_1^n)^2$ pour quelques valeurs de β/α autour des meilleurs résultats

tab:robinErrorsZoom

4.2.2 Conditions aux limites de Robin jusqu'à la dérivée seconde

On a répété les tests décrits ci-dessus, mais en remplaçant les conditions au bord droit par $\alpha u(L) + \beta u_x(L) + \gamma u_{xx}(L) = 0$, $\alpha, \beta, \gamma > 0$.

Les valeurs de α et β sont fixés et égaux à celui qui a donné l'erreur minimal dans les simulations précédents ($(\alpha, \beta) = (1, 1000)$). On montre directement le graphe contenant les erreurs pour des plusieurs valeurs de γ/β (figure 33). De façon similaire aux conclusions faites ci-dessus, on a observé une meilleure approximation du TBC pour des valeurs plus fortes de γ/β (étant l'erreur presque constante à partir d'un certain seuil, comme montre le tableau 3). En fait, même l'erreur le plus grande dans la figure 33 ($\|e_1\|(\gamma/\beta = 0.01) = 8.78 \times 10^{-6}$) est plus petit que le meilleur cas présenté dans le tableau 2.

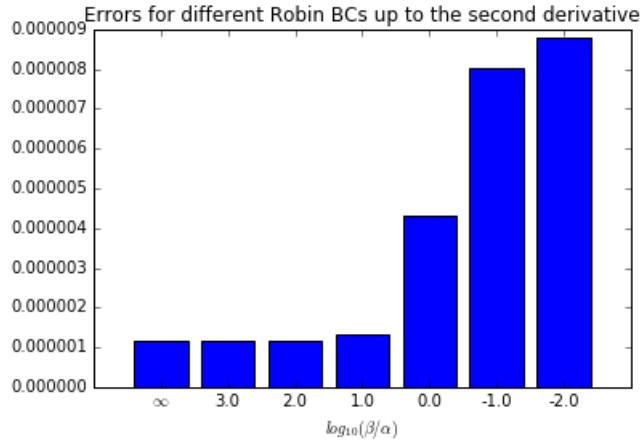


Figure 33: Erreur $\|e_1\| = \sum_{n=0}^{n_{max}} (e_1^n)^2$ entre la solution approximée et la solution de référence pour des plusieurs valeurs de γ/β , avec des conditions aux limites de Robin jusqu'à la dérivée seconde

fig:robinErrorsAll2

$\log(\gamma/\beta)$	Error ($\times 10^{-6}$)
2.0	1.157665
2.5	1.157382
3.0	1.157280
3.5	1.157247
4.0	1.157236
4.5	1.157233
∞	1.157231

Table 3: Error $\|e_1\| = \sum_{n=0}^{n_{max}} (e_1^n)^2$ for some values of $\gamma/beta$ around the best ones

ab:robinErrors2Zoom}

4.2.3 Conclusion partielle

Pour résumer cet étude initial des conditions transparentes, on a cherché à les approximer par des conditions aux bords de Robin, écrites en fonction de coefficients ajustables afin d'attribuer différentes importances à chacun des termes (la solution et ses dérivées). L'idée était de tester plusieurs combinaisons de ces coefficients afin de les optimiser (au sens de minimiser l'erreur entre la solution calculée dans $[-L, L]$ et la solution de référence, calculée dans $[-2L, 2L]$). Comme on a décrit ci-dessus, les conditions de Robin qui prennent en compte la continuité de la solution testée (i.e., avec des coefficients plus grands pour les termes des dérivées) ont produit des meilleurs résultats. Alors, la TBC basée sur la première dérivée a été plus efficiente que celle basée seulement sur la solution; et la TBC basée sur la seconde dérivée a été encore meilleure. Finalement, on a observé que l'amélioration de la solution est négligeable au-dessus une certaine relation entre les coefficients.

5 Conditions aux limites transparentes approximées pour l'équation de dispersion

Dans [6], des conditions aux limites transparents exactes sont dérivées pour l'équation de KdV linéarisée (ou équation de Airy) :

$$u_t + U_1 u_x + U_2 u_{xxx} = h(t, x), \quad t \in \mathbb{R}^+, \quad x \in \mathbb{R} \quad (28) \quad \boxed{\text{eq:LKdV}}$$

où $U_1 \in \mathbb{R}$, $U_2 \in \mathbb{R}_*^+$ et h est un terme de source.

Pour le problème homogène à valeur initiale

$$\begin{cases} u_t + U_1 u_x + U_2 u_{xxx} = h(t, x), & t \in \mathbb{R}^+, \quad x \in [a, b] \\ u(0, x) = u_0(x), & x \in [a, b] \\ \text{+boundary conditions} \end{cases}$$

les TBCs sont données [6, équations 2.17,2.18] par

$$\begin{aligned} u(t, a) - U_2 \mathcal{L}^{-1} \left(\frac{\lambda_1(s)^2}{s} \right) * u_x(t, a) - U_2 \mathcal{L}^{-1} \left(\frac{\lambda_1(s)}{s} \right) * u_{xx}(t, a) &= 0 \\ u(t, b) - \mathcal{L}^{-1} \left(\frac{1}{\lambda_1(s)^2} \right) * u_{xx}(t, b) &= 0 \\ u_x(t, b) - \mathcal{L}^{-1} \left(\frac{1}{\lambda_1(s)} \right) * u_{xx}(t, b) &= 0 \end{aligned} \quad (29) \boxed{\text{eq:continuousTBC}}$$

où \mathcal{L}^{-1} dénote la transformée inverse de Laplace, $s \in \mathbb{C}$, $\operatorname{Re}(s) > 0$ est la fréquence de Laplace et λ_1 est, parmi les trois racines du polynôme caractéristique cubique obtenu en résolvant (28) dans l'espace de Laplace, la seule avec partie réelle négative. [25]

On va concentrer nos efforts dans le cas $U_1 = 0, U_2 = 1$, qui fournit l'équation de KdV linéarisée avec seulement la partie dispersive (qu'on appellera *équation de dispersion*) :

$$u_t + u_{xxx} = 0 \quad (30) \boxed{\text{eq:DKdV}}$$

Dans ce cas, aussi d'après [25], la seule racine avec partie réelle négative est

$$\lambda(s) = \lambda_1(s) = -\sqrt[3]{s} \quad (31) \boxed{\text{eq:lambda}}$$

Cependant, le calcul des TBCs (29) n'est pas simple en raison de la transformée inverse de Laplace, qui donnent à ces conditions un caractère nonlocal en temps. Ainsi, on va proposer des approximations pour la racine (31) qui évitent les intégrations en temps, en générant des TBC considérablement plus simples.

Évidemment, comme on va à vérifier dans les résultats présentés dans cette section, les TBCs approximées ne sont pas si précises comme les TBCs proposées par [6] (qui les dérive pour la version discrète de l'équation de KdV linéarisée). Néanmoins, nos objectifs sont très différents de ceux de [6] : tandis qu'ils cherchent à minimiser l'erreur de la solution calculée (comparée avec la solution analytique) due aux conditions aux bords, on cherche ici à utiliser les TBCs approximées comme conditions aux limites à l'interface (IBCs), dans le contexte d'une Méthode de Décomposition de Domaine (DDM). Ainsi, notre objectif s'appuie sur la convergence de la solution fournie par la DDM à la solution du même problème résolu dans le monodomain, indépendamment des erreurs dans les bords extérieurs.

Pour dériver nos approximations, on rappelle les propriétés suivantes [10] de la transformée de Laplace:

- Linearity :

$$\mathcal{L}^{-1} [a_1 \hat{u}_1(s, x) + a_2 \hat{u}_2(s, x)] = a_1 u_1(t, x) + a_2 u_2(t, x)$$

(32) [eq:linearityLaplace]

- First Derivative :

$$\mathcal{L}^{-1} [s \hat{u}(s, x)] = u_t(t, x) + \mathcal{L}^{-1} [u(0, x)] = u_t(s, t) + u(0, x)\delta(t)$$

(33) ?eq:firstDerivativeLaplace

- Second Derivative :

$$\begin{aligned} \mathcal{L}^{-1} [s^2 \hat{u}(s, x)] &= u_{tt}(t, x) + \mathcal{L}^{-1} [su(0, x) + u_t(0, x)] = \\ &= u_{tt}(s, t) + u(0, x)\delta_t(t) + u_t(0, x)\delta(t) \end{aligned}$$

(34) ?eq:secondDerivativeLaplace

- Convolution :

$$\mathcal{L}^{-1} [\hat{u}_1(s, x) \hat{u}_2(s, x)] = \mathcal{L}^{-1} [\hat{u}_1(s, x)] * \mathcal{L}^{-1} [\hat{u}_2(s, x)]$$

(35) [eq:convolutionLaplace]

où $\delta(t)$ est la fonction delta de Dirac et $*$ note l'opérateur de convolution.

Les propriétés (32) jusqu'à (35) nous motivent à approximer les opérandes des transformées inverses de Laplace dans (29) par des polynômes dans s . On va implémenter et tester des approximations avec des polynômes constants et des polynômes linéaires :

5.1 Approximation des TBCs utilisant des polynômes constants

On va utiliser le polynôme constant $P_0(s) = c$ pour approximer $\frac{\lambda^2}{s}$. Par ailleurs, en raison de (31), on peut approximer les autres opérandes des transformées inverses de Laplace dans (29) uniquement en fonction de c :

$$\frac{\lambda^2}{s} = c, \quad \frac{\lambda}{s} = -c^2, \quad \frac{1}{\lambda_1(s)^2} = c^2, \quad \frac{1}{\lambda_1(s)} = -c$$

En remplaçant (36) dans (29) et en considérant des approximations différentes pour les bords à gauche et à droite (respectivement avec les coefficients c_L et c_R), on obtient les TBC approximées :

$$\begin{aligned} \Theta_1^{c_L}(u, x) &= u(t, x) - c_L u_x(t, x) + c_L^2 u_{xx}(t, x) = 0 \\ \Theta_2^{c_R}(u, x) &= u(t, x) - c_R^2 u_{xx}(t, x) = 0 \\ \Theta_3^{c_R}(u, x) &= u_x(t, x) + c_R u_{xx}(t, x) = 0 \end{aligned}$$

(37) [eq:appTBCP0]

En considérant un domaine discret avec une taille de maille Δx et des points x_0, \dots, x_N , et en utilisant des approximations par différences finies, les TBC approximées (37) sont discrétisées comme

$$\begin{aligned}
u_0 - c_L \frac{u_1 - u_0}{\Delta x} + c_L^2 \frac{u_0 - 2u_1 + u_2}{\Delta x^2} &= 0 \\
u_N - c_R^2 \frac{u_N - 2u_{N-1} + u_{N-2}}{\Delta x^2} &= 0 \\
\frac{u_N - u_{N-1}}{\Delta x} + c_R^2 \frac{u_N - 2u_{N-1} + u_{N-2}}{\Delta x^2} &= 0
\end{aligned} \tag{38) ?eq:appDiscTBCP0?}$$

5.1.1 Tests de validation de l'approximation

Afin de valider nos approximations, observer son comportement général quand on fait varier les coefficients c_L et c_R , et comparer nos résultats avec ceux obtenus par [25] et [6], on va résoudre le même test numérique considéré dans ces papiers :

$$\begin{cases} u_t + u_{xxx} = 0, & x \in \mathbb{R} \\ u(0, x) = e^{-x^2}, & x \in \mathbb{R} \\ u \rightarrow 0, & |x| \rightarrow \infty \end{cases} \tag{39a) ?eq:testCaseBesse1}$$

(39b) ?eq:testCaseBesse2?

(39c) ?eq:testCaseBesse3?

La solution fondamentale de (39a) et la solution exacte du problème (39a) - (39c) sont donnés, respectivement, par

$$E(t, x) = \frac{1}{\sqrt[3]{3t}} Ai\left(\frac{x}{\sqrt[3]{3t}}\right) \tag{40) {?}}$$

$$u_{exact}(t, x) = E(t, x) * e^{-x^2} \tag{41) ?eq:exactSolution?}$$

où Ai est la fonction d'Airy.

En suivant [25] et [6], le problème sera résolu dans le domaine spatiale $[-6, -6]$, pour $0 \leq t \leq T_{max}$, avec $T_{max} = 4$. La taille de maille est $\Delta x = 12/500 = 0.024$ et, aussi comme fait [6], le pas de temps est $\Delta t = 4/2560 = 0.0015625$.

Pour une évaluation quantitative des résultats, on calculera les mêmes erreurs définies par [6]: pour chaque instant $t_n = n\Delta t$, on définit l'erreur l^2 -relative

$$e^n = \frac{\|u_{exact}^n - u_{computed}^n\|_2}{\|u_{exact}^n\|_2}$$

et, dans tout l'intervalle de temps, l'erreur maximale e^n et la norme l^2 de e^n , données respectivement par

$$e_{Tm} = \max_{0 < n < T_{max}} (e^n), \quad e_{L2} = \sqrt{\Delta t \sum_{n=1}^{T_{max}} (e^n)^2} \quad (42) \{?\}$$

Afin de vérifier l'influence de c_L et c_R sur les solutions calculées (et, possiblement, identifier des intervalles de valeurs qui donnent les meilleures approximations pour les TBCs), on a fait des tests avec tous les paires possibles $(c_L, c_R) \in \{-10, -1, -0.1, 0, 0.1, 1, 10\}^2$. Les résultats ont été classifiés selon leurs erreurs e_{L2} (un critère basé sur l'erreur e_{Tm} fournit un résultat similaire). La figure 34 montre, pour quelques instants, une comparaison entre la solution exacte, la meilleure et la pire. Pour nommer la pire solution, on n'a pas considéré les solutions divergentes (selon le critère arbitraire $e_{L2} > 10$).

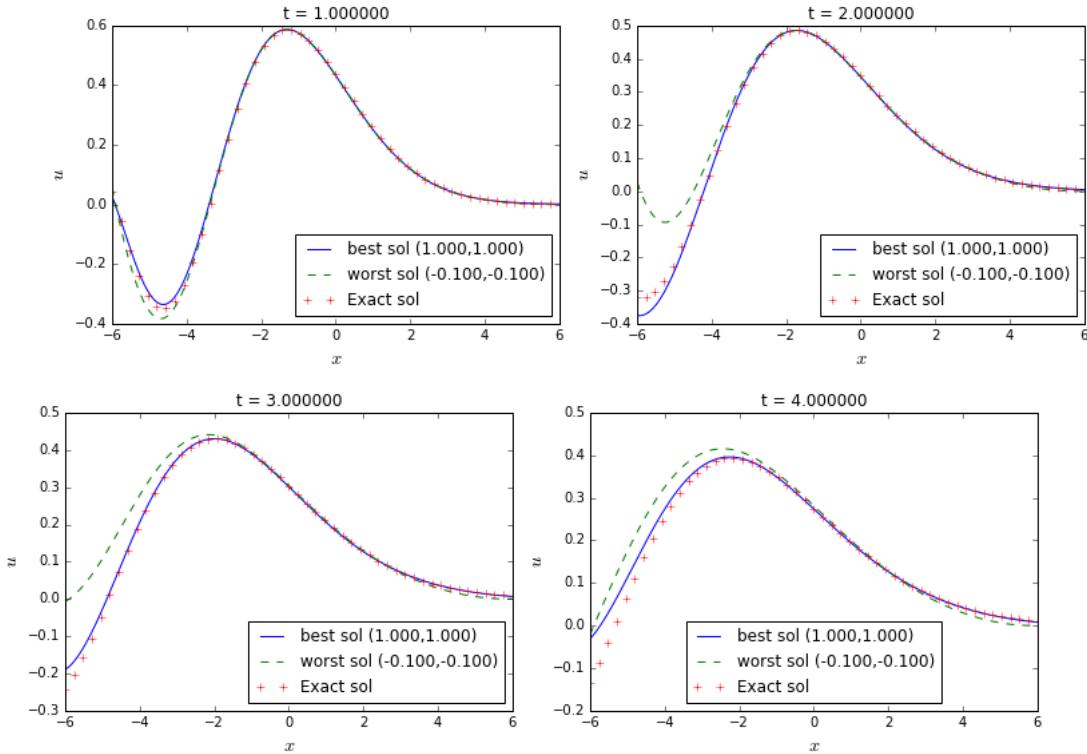


Figure 34: Best and worst solution compared with analytical solution, for the constant polynomial approximation

`(fig:firstTestsP0)` Le tableau 4 présente les dix tests qui ont donné les erreurs e_{L2} les plus petites :

c_L	c_R	e_{L2}
1.0	1.0	0.0947
1.0	10.0	0.0973
1.0	0.1	0.0984
1.0	0.0	0.0992
1.0	-10.0	0.0994
1.0	-0.1	0.1000
1.0	-1.	0.1016
10.0	1.0	0.3470
10.0	0.1	0.3474
10.0	0.0	0.3475

Table 4: Best results (smallest e_{L2}) for the constant polynomial approximation

(tab:firstTestsP0)

On remarque que les résultats sont beaucoup plus sensitives au coefficient pour le bord à gauche : pour un c_L fixé, l'erreur est très similaire pour tout c_R . Cela est une conséquence du fait que la solution de ce problème est pratiquement constante et égale à zéro aux voisinages de $x = 6$, mais présente des fortes variations dans la région proche de $x = -6$. Le meilleur résultat, comme montre la figure 34, est capable d'imposer ce condition dans le bord à droite, tandis que la pire solution n'en peut pas. Dans le cas du bord à gauche, malgré une erreur plus évidente, les meilleures solutions suivent approximativement bien le comportement de la solution exacte.

5.2 Approximation des TBCs utilisant un polynôme linéaire

De façon similaire à ce qu'on a fait ci-dessus, on approxime $\frac{\lambda^2}{s}$ par $P_1(s) = ds + c$. Puis, en établissant des relations comme (36), les convolutions dans (29) s'écrivent :

$$\begin{aligned}
\mathcal{L}^{-1} \left(\frac{\lambda^2}{s} \right) * u_x(s, x) &= \mathcal{L}^{-1} \left(\frac{\lambda^2}{s} \hat{u}_x(s, x) \right) = \mathcal{L}^{-1} [(ds + c)\hat{u}_x(t, s)] = \\
&\quad d\mathcal{L}^{-1} [\hat{u}_{xt}(t, s)] + c\mathcal{L}^{-1} [\hat{u}_x(t, s)] = \\
&\quad du_{xt}(x, t) + du_x(0, x)\delta(t) + cu_x(x, t) \\
\mathcal{L}^{-1} \left(\frac{\lambda}{s} \right) * u_{xx}(s, x) &= \mathcal{L}^{-1} \left(\frac{\lambda}{s} \hat{u}_{xx}(s, x) \right) = \mathcal{L}^{-1} [-(ds + c)^2 \hat{u}_{xx}(t, s)] = \\
&\quad -d^2 u_{xxtt}(x, t) - d^2 \delta_t(t) u_x x(0, x) - d^2 \delta(t) u_{xxt}(0, x) + \\
&\quad -2dcu_{xxt}(x, t) - 2dcu_x x(0, x)\delta(t) - c^2 u_{xx}(x, t) \\
\mathcal{L}^{-1} \left(\frac{1}{\lambda^2} \right) * u_{xx}(s, x) &= \mathcal{L}^{-1} \left(\frac{1}{\lambda^2} \hat{u}_{xx}(s, x) \right) = \mathcal{L}^{-1} [(ds + c)^2 \hat{u}_{xx}(t, s)] = \\
&\quad d^2 u_{xxtt}(x, t) + d^2 \delta_t(t) u_x x(0, x) + d^2 \delta(t) u_{xxt}(0, x) + \\
&\quad 2dcu_{xxt}(x, t) + 2dcu_x x(0, x)\delta(t) + c^2 u_{xx}(x, t) \\
\mathcal{L}^{-1} \left(\frac{1}{\lambda} \right) u_{xx}(s, x) &= \mathcal{L}^{-1} \left(\frac{1}{\lambda} \hat{u}_{xx}(s, x) \right) = \mathcal{L}^{-1} [-(ds + c)\hat{u}_{xx}(t, s)] = \\
&\quad -du_{xxt}(x, t) - du_{xx}(0, x)\delta(t) - cu_{xx}(x, t)
\end{aligned}$$

En utilisant des différences finies pour les dérivées en temps, et en considérant que la solution initiale est nulle sur les bords (ce qui est le cas de l'exemple testé ici), on obtient les TBC approximées discrètes :

$$\begin{aligned}
u_0^{n+1} - \left(\frac{d_L}{\Delta t} + c_L \right) \left(\frac{u_1^{n+1} - u_0^{n+1}}{\Delta x} \right) + \left(\frac{d_L^2}{\Delta t^2} + \frac{2d_L c_L}{\Delta t} + c_L^2 \right) \left(\frac{u_0^{n+1} - 2u_1^{n+1} + u_2^{n+1}}{\Delta x^2} \right) = \\
-\frac{d_L}{\Delta t} \left(\frac{u_1^n - u_0^n}{\Delta x} \right) + \left(2\frac{d_L^2}{\Delta t^2} + \frac{2d_L c_L}{\Delta t} \right) \left(\frac{u_0^n - 2u_1^n + u_2^n}{\Delta x^2} \right) - \frac{d_L^2}{\Delta t^2} \left(\frac{u_0^{n-1} - 2u_1^{n-1} + u_2^{n-1}}{\Delta x^2} \right) \\
u_N^{n+1} - \left(\frac{d_R^2}{\Delta t^2} + \frac{2d_R c_R}{\Delta t} + c_R^2 \right) \left(\frac{u_N^{n+1} - 2u_{N-1}^{n+1} + u_{N-2}^{n+1}}{\Delta x^2} \right) = \\
-\left(2\frac{d_R^2}{\Delta t^2} + \frac{2d_R c_R}{\Delta t} \right) \left(\frac{u_N^n - 2u_{N-1}^n + u_{N-2}^n}{\Delta x^2} \right) + \frac{d_R^2}{\Delta t^2} \left(\frac{u_N^{n-1} - 2u_{N-1}^{n-1} + u_{N-2}^{n-1}}{\Delta x^2} \right) \\
\frac{u_N^{n+1} - u_{N-1}^{n+1}}{\Delta x} + \left(\frac{d_R}{\Delta t} + c_R \right) \left(\frac{u_N^{n+1} - 2u_{N-1}^{n+1} + u_{N-2}^{n+1}}{\Delta x^2} \right) = \frac{d_R}{\Delta t} \left(\frac{u_N^n - 2u_{N-1}^n + u_{N-2}^n}{\Delta x^2} \right)
\end{aligned}$$

5.2.1 Tests de validation de l'approximation

On répète les tests faites pour l'approximation avec P_0 , en faisant varier les coefficients c_L et c_R parmi les valeurs dans $\{-10, -1, -0.1, 0, 0.1, 1, 10\}$. Afin d'éviter un calcul trop longue, et aussi en utilisant la remarque fait ci-dessus concernant la faible influence des coefficients du bord à droite sur les résultats, tous les tests seront faits avec $c_R = c_L$ et $d_R = d_L$.

Les dix meilleurs résultats sont présentés dans le tableau 5. On observe que le meilleur résultat est celui avec $d_L = d_R = 0$ et $c_L = c_R = 1.$, ce qui correspond au meilleur résultat parmi les approximations utilisant des polynômes constants.

$d_L = d_R$	$c_L = c_R$	e_{L2}
0.	1.0	0.1075
0.1	1.0	0.1405
1.0	1.0	0.1920
10.0	0.1	0.2279
-10.0	0.1	0.3771
10.0	1.0	0.2716
-10.0	0.0	0.2480
-10.0	1.0	0.3004
10.0	0.0	0.2721
0.0	0.1	0.3674

Table 5: Meilleurs résultats (les erreurs e_{L2} les plus petites) pour l'approximation avec le polynôme linéaire

(tab:firstTestsP1)

5.3 Conclusion partielle

Il faut être claire que notre approximation ne fournit pas des meilleures TBCs que celles proposées par [6] (et cela n'est pas non plus l' objectif du travail développé ici, comme il a été discuté dans la introduction de ce rapport). En fait, [6] dérive des TBCs pour deux schémas discrets, et le pire résultat parmi eux, en utilisant les mêmes Δt et Δx utilisés ici, présente une erreur $e_{L2} \approx 0.005$ pour $t = 4$, tandis que notre mieux résultat fournit $e_{L2} \approx 0.1$ pour le même instant. Néanmoins, en considérant qu'on souhaite appliquer les TBCs à une Méthode de Décomposition de Domaine, on voudra minimiser l'erreur due à les conditions limites à l'interface, mais pas l'erreur liée aux conditions aux limites extérieures.

Cependant, enc considérant les résultats présentés jusqu'ici, on peut dire que les conditions aux limites proposées simulent relativement bien des TBCs, avec une implémentation assez simple, en comparaison avec celles proposées

par [6] (qui demandent par exemple le calcul de \mathcal{Z} -transformées, dans le rôle de versions discrètes des Transformées de Laplace). Par ailleurs, on remarque que l'approximation qu'on a fait avec le polynôme linéaire, malgré l'augmentation de la complexité du schéma (où il faut garder la solution des itérations précédentes), ne fournit pas des meilleures résultats par rapport au cas du polynôme constant.

Ainsi, dans la suite, on va utiliser toujours l'approximation avec P_0 . On notera les TBCs correspondantes avec les opérateurs θ_i^c , $i = 1, 2, 3$, définis dans (37).

$$\begin{aligned}\Theta_1^{c_L}(u, x) &= u(t, x) - c_L u_x(t, x) + c_L^2 u_{xx}(t, x) \\ \Theta_2^{c_R}(u, x) &= u(t, x) - c_R^2 u_{xx}(t, x) \\ \Theta_3^{c_R}(u, x) &= u_x(t, x) + c_R u_{xx}(t, x)\end{aligned}$$

6 Application des TBCs approximées à une Méthode de Décomposition de Domaine

L'approximation des conditions aux limites transparentes (TBCs) utilisant un polynôme constant sera appliquée dans une Méthode de Décomposition de Domaine (*Domain Decomposition Method - DDM*). On va d'abord décrire le DDM qu'on utilisera ici, et ensuite on décrira et testera l'incorporation des TBCs.

6.1 The Schwarz Methods

La description suivante s'appuie sur [15]. Les Méthodes de Décomposition de Domaines, comme indique son nom, permettent de décomposer un domaine Ω en plusieurs subdomaines Ω_i (superposés ou pas) et de résoudre le problème dans chacun d'eux. Ainsi, il faut trouver des fonctions qui satisfassent la PDE dans chacun des subdomaines et que soient égales dans les interfaces.

La première DDM développée a été la méthode de Schwarz [15, 14], qui consiste en une méthode itérative : dans le cas d'un problème d'évolution, la solution $u_i^{n,\infty}$, dans chaque pas de temps t_n et chaque subdomaine Ω_i , est calculée comme étant la convergence de la solution obtenue dans chaque itération, $u_i^{n,k}$, $k \geq 0$. Il y a deux types de méthodes de Schwarz, en dépendant de la manière avec laquelle les conditions aux interfaces sont construites pour calculer $u_i^{n,k}$.

Dans la méthode de Schwarz additive (*Additive Schwarz Method - ASM*), les conditions aux interfaces sont toujours calculées en utilisant la solution

$u_j^{n,k-1}$, $j \neq i$ de la dernière itération. Ainsi, dans l'interface entre les domaines Ω_i et Ω_j , les conditions à l'interface pour le problème Ω_i s'écrivent comme

$$\mathcal{B}_i(u_i^{n,k+1}) = \mathcal{B}_i(u_j^{n,k})$$

où \mathcal{B} est l'opérateur de la TBC.

En revanche, la méthode de Schwarz multiplicative (*Multiplicative Schwarz Method - MSM*) utilise toujours l'information la plus récente pour le calcul des conditions aux interfaces. Ainsi, dans le cas d'une DDM avec deux subdomaines, les TBCs s'écriraient (par exemple) sous la forme

$$\mathcal{B}_1(u_1^{n,k+1}) = \mathcal{B}_1(u_2^{n,k})\mathcal{B}_2(u_2^{n,k+1}) = \mathcal{B}_2(u_1^{n,k+1})$$

pour la résolution du problème dans Ω_1 et Ω_2 , respectivement. En fait, la MSM a été la forme originale proposée par Schwarz (avec une condition du type Dirichlet, $\mathcal{B}_j(u) = u$) [15, 19]. L'ASM est une modification proposée par [18], et qui présente comme principale avantage (notamment quand le nombre de subdomaines augmente) le fait d'être un algorithme naturellement parallèle (et qui peut ainsi être implémenté en utilisant la computation parallèle) [18].

Dans le stage, on n'a travaillé qu'avec l'ASM. Par ailleurs, on a toujours considéré une DDM décomposant $\Omega \subset \mathcal{R}$ dans deux subdomaines Ω_1 et Ω_2 , non superposants (sauf par un point en commun). La description faite dans ce rapport considère toujours ces hypothèses, mais elle serait équivalente dans le cas de DDMs plus générales.

On veut qu'une DDM présente les propriétés suivantes :

- Il y a une unique solution Ω_i dans chaque subdomaine Ω_i .
- La solution u_i dans chaque subdomaine Ω_i converge vers $u|_{\Omega_1}$, i.e., la solution u du monodomaine Ω restreinte à Ω_1
- La méthode converge rapidement

Selon [?], l'ASM optimale est celle qui utilise les exactes TBCs (27) comme conditions aux interfaces : dans ce cas, la méthode converge dans deux itérations, et aucune autre ASM ne peut converger plus rapidement. Néanmoins, comme discuté avant dans ce rapport, la dérivation analytique et l'implémentation numérique des TBCs exactes sont, en général, impraticables, en raison de son caractère non local en temps. Ainsi, on propose ici l'implémentation des TBCs approximées (37) dans la DDM.

6.2 ASM avec des TBCs approximées pour l'équation de dispersion

La résolution de l'équation de dispersion avec la Méthode de Schwarz Additive, en utilisant l'approximation des TBCs construite à partir d'un polynôme constant, s'écrit comme

$$\begin{cases} (u_1^{n,k+1})_t + (u_1^{n,k+1})_{xxx} = 0, & x \in \Omega_1, \quad t \geq 0 \\ u_1^{n,0} = u_1^{n-1,\infty}, & x \in \Omega_1 \\ \Upsilon_1^{c_L}(u_1^{n+1,k+1}, -L) = 0, \\ \Theta_2^{c_R}(u_1^{n+1,k+1}, 0) = \Theta_2^{c_R}(u_2^{n,k}, 0), \\ \Theta_3^{c_R}(u_1^{n+1,k+1}, 0) = \Theta_3^{c_R}(u_2^{n,k}, 0) \end{cases} \quad (43) \boxed{\text{eq:problemDDM1}}$$

$$\begin{cases} (u_2^{n,k+1})_t + (u_2^{n,k+1})_{xxx} = 0, & x \in \Omega_2, \quad t \geq 0 \\ u_2^{n,0} = u_2^{n-1,\infty}, & x \in \Omega_2 \\ \Theta_1^{c_L}(u_2^{n+1,k+1}, 0) = \Theta_1^{c_L}(u_1^{n,k}, 0) \\ \Upsilon_2^{c_R}(u_2^{n+1,k+1}, L) = 0 \\ \Upsilon_3^{c_R}(u_2^{n+1,k+1}, L) = 0 \end{cases} \quad (44) \boxed{\text{eq:problemDDM2}}$$

où Υ_i , $i = 1, 2, 3$, sont les conditions aux limites sur les bords extérieurs (i.e., définies sur $\partial\Delta_i \setminus \Gamma$). Ces conditions externes sont indépendantes des conditions à l'interface. On va considérer ici $\Upsilon_1 = \Theta_1^{1,0}$, $\Upsilon_2 = \Theta_2^{0,0}$ and $\Upsilon_3 = \Theta_3^{0,0}$, ce qui donne

$$\begin{aligned} \Upsilon_1(u, x) &= u - u_x + u_{xx} \\ \Upsilon_2(u, x) &= 0 \\ \Upsilon_3(u, x) &= 0 \end{aligned} \quad (45) \boxed{\text{eq:externalBCsDDM}}$$

Ce choix a été fait en se basant sur son implémentation simple et sur les bons résultats fournis par les coefficients $c_L = 1.0$ et $c_R = 0.0$ lors de l'approximation de la solution analytique dans Ω (comme montre le tableau 4). Néanmoins, cela n'a pas une vraie importance dans l'étude qu'on propose ici, parce qu'on va comparer les résultats de la DDM avec une solution numérique de référence calculée dans le monodomaine. La seule exigence pour que cet étude soit cohérent est que les conditions aux limites externes pour le calcul de u^{ref} soient les mêmes Υ_i , $i = 1, 2, 3$ utilisées dans la DDM.

Ainsi, la solution de référence utilisé dans cet étude est la solution du problème

$$\begin{cases} u_t + u_{xxx} = 0, \quad x \in \Omega, \quad t \in [t_0, t_0 + \Delta t] \\ u(t_0, x) = u^{exact}(t_0, x), \quad x \in \Omega \\ \Upsilon_1(u, -L) = 0, \quad t \in [t_0, t_0 + \Delta t] \\ \Upsilon_2(u, L) = 0, \quad t \in [t_0, t_0 + \Delta t] \\ \Upsilon_3(u, L) = 0, \quad t \in [t_0, t_0 + \Delta t] \end{cases} \quad (46) \quad \text{[eq:problemMonodomain]}$$

Remarques sur la notation Toujours en considérant les objectifs de ce travail, on remarque qu'on va comparer les solutions calculées seulement dans un pas de temps. Cela est nécessaire pour que l'erreur due à la DDM soit étudiée séparément (sans influence, par exemple, de l'erreur accumulée au long des pas de temps, due à la discrétisation temporelle).

Par conséquent, on peut alléger la notation. À parti d'ici jusqu'à la fin de ce rapport, on notera par u_j^i la solution du DDM, où i indique le subdomaine Ω_i (ou, dans le cas de la solution de référence, $i = ref$, et à convergence de la méthode, $i = *$) et j indique la position spatiale discrète. Dans les cas où le processus itératif doit être considéré, on ajoutera le superindice k pour indiquer l'itération.

En ce qui concerne la discrétisation spatiale, le monodomaine Ω sera divisé en $2N + 1$ points distribués de façon homogène, numérotés de 0 jusqu'à $2N$. Dans les descriptions suivantes, on va considérer toujours que les deux subdomaines Ω_1 et Ω_2 ont le même nombre de points, respectivement x_0, \dots, x_N et x_N, \dots, x_{2N} . Le point à l'interface, x_N , est commun aux deux subdomaines, ayant des solutions différentes dans chacun d'eux : u_N^1 et u_N^2 . Évidemment, on espère que, à convergence, $u_N^1 = u_N^2 = u_N^*$.

6.3 Discrétisation du problème et erreur dans la solution convergée

Lors de l'application de TBCs approximés dans une ASM, on doit assurer que la solution convergée u^* satisfait la même équation discrète que la solution u^{ref} du problème dans le monodomaine. Dans les paragraphes suivants, on va montrer que cette propriété n'est pas vérifiée par la méthode (43) - (44) proposée ici, et, en se basant sur cette démonstration, on proposera des corrections pour ce problème.

Pour les points intérieurs de chacun des domaines, on considérera une discrétisation spatiale de seconde ordre pour l'équation (30) :

$$\frac{u_j^i - \alpha_j^i}{\Delta t} + \frac{-\frac{1}{2}u_{j-2}^i + u_{j-1}^i - u_{j+1}^i + \frac{1}{2}u_{j+2}^i}{\Delta x^3} = 0 \quad (47) \quad \text{[eq:FDdiscretization]}$$

ce qui est valide pour $j = 2, \dots, N - 2$ dans le cas $i = 1$; pour $j = N + 2, \dots, 2N - 2$ dans le cas $i = 2$; et pour $j = 2, \dots, 2N - 2$ dans le cas $i = ref$. Dans (47), α_j^i est une donnée (par exemple, la solution convergée du pas de temps précédent).

Pour les points proches aux bords, on utilise une discréétisation décentrée ou les TBCs appropriées. En fait, en considérant qu'une TBC est écrite pour le bord à gauche et deux pour le bord à droite, on doit imposer une discréétisation décentrée seulement pour le deuxième point le plus proche du bord à gauche. Par exemple, pour le point x_1 :

$$\frac{u_1^2 - \alpha_1^2}{\Delta t} + \frac{-\frac{5}{2}u_1^2 + 9u_2^2 - 12u_3^2 + 7\frac{1}{2}u_4^2 - \frac{3}{2}u_5^2}{\Delta x^3} = 0 \quad (48) \quad \text{?eq:uncenteredFDdisc}$$

et de façon similaire pour les autres points proches des bords.

Pour résoudre le problème dans Ω_1 , two interface boundary conditions are imposed (corresponding to Θ_2 and Θ_3) to the discrete equations for the points x_{N-1} and x_N :

$$\begin{aligned} \Theta_2^{c_R}(u_N^1) &= \Theta_2^{c_R}(u_N^2) \implies \\ \implies u_N^1 - c_R^2 \frac{u_N^1 - 2u_{N-1}^1 + u_{N-2}^1}{\Delta x^2} &= u_N^2 - c_R^2 \frac{u_N^2 - 2u_{N+1}^2 + u_{N+2}^2}{\Delta x^2} \end{aligned} \quad (49) \quad \text{?eq:TBCsIterOmega1A}$$

$$\begin{aligned} \Theta_3^{c_R}(u_N^1) &= \Theta_3^{c_R}(u_N^2) \implies \\ \implies \frac{u_N^1 - u_{N-1}^1}{\Delta x} + c_R \frac{u_N^1 - 2u_{N-1}^1 + u_{N-2}^1}{\Delta x^2} &= \frac{u_{N+1}^2 - u_N^2}{\Delta x} + c_R \frac{u_N^2 - 2u_{N+1}^2 + u_{N+2}^2}{\Delta x^2} \end{aligned} \quad (50) \quad \text{?eq:TBCsIterOmega1B}$$

En revanche, pour résoudre le problème dans Ω_2 , seulement une condition à l'interface est utilisée (correspondant à Θ_1), étant imposée pour le point x_N :

$$\begin{aligned} \Theta_1^{c_L}(u_N^2) &= \Theta_1^{c_L}(u_N^1) \implies \\ \implies u_N^2 - c_L \frac{u_{N+1}^2 - u_N^2}{\Delta x} + c_L^2 \frac{u_N^2 - 2u_{N+1}^2 + u_{N+2}^2}{\Delta x^2} &= \\ u_N^1 - c_L \frac{u_N^1 - u_{N-1}^1}{\Delta x} + c_L^2 \frac{u_N^1 - 2u_{N-1}^1 + u_{N-2}^1}{\Delta x^2} & \end{aligned} \quad (51) \quad \text{?eq:TBCsIterOmega2}$$

À convergence, les expressions (49) jusqu'à (51) donnent respectivement

- $u_N^* - c_R^2 \frac{u_N^* - 2u_{N-1}^* + u_{N-2}^*}{\Delta x^2} = u_N^* - c_R^2 \frac{u_N^* - 2u_{N+1}^* + u_{N+2}^*}{\Delta x^2} \implies$

$$\implies 2c_R^2 \frac{-\frac{1}{2}u_{N-2}^* + u_{N-1}^* - u_{N+1}^* + \frac{1}{2}u_{N+2}^*}{\Delta x^2} = 0$$

- $$\frac{u_N^* - u_{N-1}^*}{\Delta x} + c_R \frac{u_N^* - 2u_{N-1}^* + u_{N-2}^*}{\Delta x^2} =$$

$$\frac{u_{N+1}^* - u_N^*}{\Delta x} + c_R \frac{u_N^* - 2u_{N+1}^* + u_{N+2}^*}{\Delta x^2} \implies$$

$$\implies -\frac{u_{N-1}^* - 2u_N^* + u_{N+1}^*}{\Delta x} - 2c_R \frac{-\frac{1}{2}u_{N-2}^* + u_{N-1}^* - u_{N+1}^* + \frac{1}{2}u_{N+2}^*}{\Delta x^2} = 0$$
- $$u_N^* - c_L \frac{u_{N+1}^* - u_N^*}{\Delta x} + c_L^2 \frac{u_N^* - 2u_{N+1}^* + u_{N+2}^*}{\Delta x^2} =$$

$$u_N^* - c_L \frac{u_N^* - u_{N-1}^*}{\Delta x} + c_L^2 \frac{u_N^* - 2u_{N-1}^* + u_{N-2}^*}{\Delta x^2} \implies$$

$$\implies -c_L \frac{u_{N-1}^* - 2u_N^* + u_{N+1}^*}{\Delta x} + 2c_L^2 \frac{-\frac{1}{2}u_{N-2}^* + u_{N-1}^* - u_{N+1}^* + \frac{1}{2}u_{N+2}^*}{\Delta x^2} = 0$$

Alors, on peut observer que la solution convergée de la DDM ne satisfait pas la même équation discrète que la solution de référence dans les points $x_{N-1}, x_N \in \Omega_1$, et dans $x_N \in \Omega_2$. Dans tous les autres points les équations satisfaites sont identiques, à l'exception du point $x_{N_1} \in \Omega_2$, comme détaillé dans la remarque suivante :

Remarque : modification de la solution de référence : Même si la DDM fournissait une solution compatible avec celle du problème dans le monodomain (ce qu'on va imposer avec des corrections proposées dans la suite de ce rapport), la solution de la DDM ne convergerait exactement vers u^{ref} , pour une raison qui ne dépendent pas de l'expression des IBCs, mais si du fait qu'on écrit deux IBCs pour le bord à gauche et une pour le droit. Comme on utilise une discréttisation centrée pour la troisième dérivative dans l'espace (ce qui demande un stencil de deux points de chaque côté du point au milieu), il faut écrire une discréttisation décentrée pour le point x_{N+1} lors de la résolution du problème dans Ω_2 . Ainsi, ce point ne satisfait pas la même équation discrète que dans le problème de référence. Afin d'éviter cette incompatibilité et de nous permettre de bien étudier le comportement de la DDM, on va modifier la discréttisation du point x_{N+1} dans problème du monodomain, en utilisant la même discréttisation décentrée de seconde ordre :

$$\frac{u_{N+1}^2 - \alpha_{N+1}^2}{\Delta t} + \frac{-\frac{5}{2}u_{N+1}^2 + 9u_{N+2}^2 - 12u_{N+3}^2 + 7\frac{1}{2}u_{N+4}^2 - \frac{3}{2}u_{N+1}^2}{\Delta x^3} = 0$$

Étant faite cette remarque, la figure 35 résume les discrétisations imposées pour chaque point dans les problèmes du monodomain et de la DDM, selon da description faite ci-dessus :

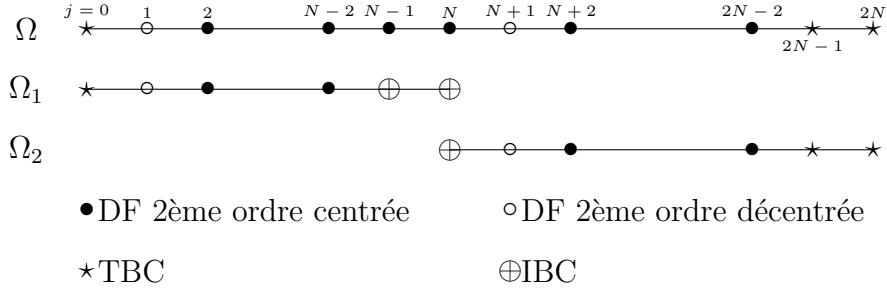


Figure 35: Schéma indiquanla discrétisation spatiale imposée pour chaque point du monodomaine et des subdomaines de la DDM

`fig:discretizations`

6.3.1 Numerical verification of the error

Le problème (43) - (44) a été résolu jusqu'à convergence avec cinq discrétisations spatiales uniformes différentes, au long d'un pas de temps (dans l'intervalle $[0, \Delta t]$). Dans chaque cas, la solution de référence u^{ref} était la solution du problème du monodomain (46), résolu avec la même taille de maille. Deux erreurs ont été calculées :

$$e^{N,*} = |u_N^{ref} - u_N^*|$$

$$e^{\Omega,*} = \|u_N^{ref} - u_N^*\|_2 = \sqrt{\Delta x \left[\sum_{j=0}^N (u_j^{ref} - u_j^{1,\infty})^2 + \sum_{j=N}^{2N} (u_j^{ref} - u_j^{2,\infty})^2 \right]} \quad (52) \quad \text{[eq:errorDDM]}$$

correspondant respectivement à l'erreur sur l'interface et à l'erreur dans tout le domaine.

On s'intéresse au comportement de ces erreurs en fonction de la taille de maille. Comme montré la figure 36, on vérifie que la DDM proposée par nous produit une erreur d'ordre $O(\Delta x)$:

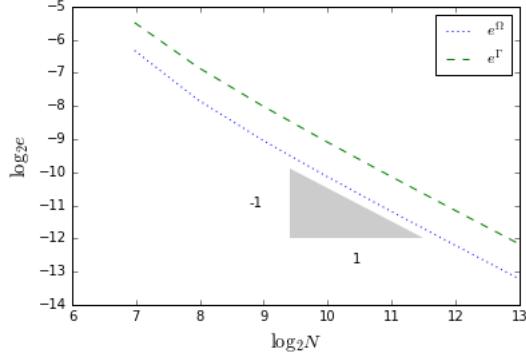


Figure 36: Vérification numérique de l'ordre de convergence de l'erreur due à la DDM

6.3.2 Corrections pour les TBCs approximées

On va formuler des modifications pour les TBCs utilisées dans l'ASM afin d'annuler ces erreurs :

$$\begin{aligned}\Theta_1^{cL}(u_2^{n+1,k+1}) + \theta_1 &= \Theta_1^{cL}(u_1^{n,k}) + \theta'_1 \\ \Theta_2^{cR}(u_1^{n+1,k+1}) + \theta_2 &= \Theta_2^{cR}(u_2^{n,k}) + \theta'_2 \\ \Theta_3^{cR}(u_1^{n+1,k+1}) + \theta_3 &= \Theta_3^{cR}(u_2^{n,k}) + \theta'_3\end{aligned}$$

avec θ_i, θ'_i donnés par

$$\begin{aligned}\theta_1 &= \Delta x c_L \frac{u_{N+1}^2 - 2u_N^2 + u_{N-1}^1}{\Delta x^2} + c_L^2 \frac{\Delta x}{\Delta t} (u_N^2 - \alpha_N^2) \\ \theta'_1 &= -c_L^2 \frac{\Delta x}{\Delta t} (u_N^1 - \alpha_N^1)\end{aligned}$$

$$\begin{aligned}\theta_2 &= \frac{\Delta x}{\Delta t} c_R^2 (u_N^1 - \alpha_N^1) \\ \theta'_2 &= -\frac{\Delta x}{\Delta t} c_R^2 (u_N^2 - \alpha_N^2)\end{aligned}$$

$$\begin{aligned}\theta_3 &= 2 \frac{\Delta x}{\Delta t} [-\Delta x (u_{N-1}^1 - \alpha_{N-1}^1) - c_R (u_N^1 - \alpha_N^1)] + \Delta x \frac{u_{N-3}^1 - 2u_{N-2}^1 + u_{N-1}^1}{\Delta x^2} \\ \theta'_3 &= 0\end{aligned}$$

En fait, on a, dans la convergence de la première condition à l'interface :

$$\begin{aligned}
& \Theta_1^{c_L}(u_N^*) + \theta_1 = \Theta_1^{c_L}(u_N^*) + \theta'_1 \\
\implies & u_N^* - c_L \frac{u_{N+1}^* - u_N^*}{\Delta x} + c_L^2 \frac{u_N^* - 2u_{N+1}^* + u_{N+2}^*}{\Delta x^2} + \\
& \Delta x c_L \frac{u_{N+1}^* - 2u_N^* + u_{N-1}^*}{\Delta x^2} + c_L^2 \frac{\Delta x}{\Delta t} (u_N^* - \alpha_N^*) = \\
& u_N^* - c_L \frac{u_N^* - u_{N-1}^*}{\Delta x} + c_L^2 \frac{u_N^* - 2u_{N-1}^* + u_{N-2}^*}{\Delta x^2} - c_L^2 \frac{\Delta x}{\Delta t} (u_N^* - \alpha_N^*) \\
\implies & 2c_L^2 \frac{-\frac{1}{2}u_{N-2}^* + u_{N-1}^* - u_{N+1}^* + \frac{1}{2}u_{N+2}^*}{\Delta x^2} + 2c_L^2 \frac{\Delta x}{\Delta t} (u_N^* - \alpha_N^*) = 0 \\
\implies & \frac{u_N^* - \alpha_N^*}{\Delta t} + \frac{-\frac{1}{2}u_{N-2}^* + u_{N-1}^* - u_{N+1}^* + \frac{1}{2}u_{N+2}^*}{\Delta x^3} = 0
\end{aligned}$$

ce qui est identique à (47) satisfait dans x_N .

Pour la deuxième IBC :

$$\begin{aligned}
& \Theta_2^{c_R}(u_N^*) + \theta_2 = \Theta_2^{c_R}(u_N^*) + \theta'_2 \\
\implies & u_N^* - c_R^2 \frac{u_N^* - 2u_{N-1}^* + u_{N-2}^*}{\Delta x^2} + \frac{\Delta x}{\Delta t} c_R^2 (u_N^* - \alpha_N^*) = \\
& u_N^* - c_R^2 \frac{u_N^* - 2u_{N+1}^* + u_{N+2}^*}{\Delta x^2} - \frac{\Delta x}{\Delta t} c_R^2 (u_N^* - \alpha_N^*) \tag{53} \boxed{\text{eq:modifiedTBC2}} \\
\implies & 2 \frac{\Delta x}{\Delta t} c_R^2 (u_N^* - \alpha_N^*) + 2c_R^2 \frac{-\frac{1}{2}u_{N-2}^* + u_{N-1}^* - u_{N+1}^* + \frac{1}{2}u_{N+2}^*}{\Delta x^2} = 0 \\
\implies & \frac{u_N^* - \alpha_N^*}{\Delta t} + \frac{-\frac{1}{2}u_{N-2}^* + u_{N-1}^* - u_{N+1}^* + \frac{1}{2}u_{N+2}^*}{\Delta x^3} = 0
\end{aligned}$$

ce qui correspond également à (47) satisfait dans x_N .

Finalement, pour la troisième IBC, on utilise (53) dans (50) pour obtenir

$$-\frac{u_{N-1}^* - 2u_N^* + u_{N+1}^*}{\Delta x} + 2c_R \Delta x \frac{u_N^* - \alpha_N^*}{\Delta t} = 0$$

Ainsi

$$\begin{aligned}
& \Theta_3^{c_R}(u_N^*) + \theta_3 = \Theta_3^{c_R}(u_N^*) + \theta'_3 \\
\implies & \frac{u_N^* - u_{N-1}^*}{\Delta x} + c_R \frac{u_N^* - 2u_{N-1}^* + u_{N-2}^*}{\Delta x^2} + 2 \frac{\Delta x}{\Delta t} [-\Delta x(u_{N-1}^* - \alpha_{N-1}^*) - c_R(u_N^* - \alpha_N^*)] + \\
& \frac{u_{N-3}^* - 2u_{N-2}^* + u_{N-1}^*}{\Delta x} = \frac{u_{N+1}^* - u_N^*}{\Delta x} + c_R \frac{u_N^* - 2u_{N+1}^* + u_{N+2}^*}{\Delta x^2} \\
\implies & -\frac{u_{N-1}^* - 2u_N^* + u_{N+1}^*}{\Delta x} + 2c_R \Delta_x \frac{u_N^* - \alpha_N^*}{\Delta t} + \\
& 2 \frac{\Delta x}{\Delta t} [-\Delta x(u_{N-1}^* - \alpha_{N-1}^*) - c_R(u_N^* - \alpha_N^*)] + \frac{u_{N-3}^* - 2u_{N-2}^* + u_{N-1}^*}{\Delta x} = 0 \\
\implies & -2 \frac{-\frac{1}{2}u_{N-3}^* + u_{N-2}^* - u_N^* + \frac{1}{2}u_{N+1}^*}{\Delta x} - 2 \frac{\Delta x^2}{\Delta t} (u_{N-1}^* - \alpha_{N-1}^*) = 0 \\
\implies & \frac{u_{N-1}^* - \alpha_{N-1}^*}{\Delta t} + \frac{-\frac{1}{2}u_{N-3}^* + u_{N-2}^* - u_N^* + \frac{1}{2}u_{N+1}^*}{\Delta x^3} = 0
\end{aligned}$$

ce qui est la discréétisation (47) écrite pour le point x_{N-1} .

6.4 Optimisation des IBCs (vitesse de convergence)

Notre objectif maintenant est d'optimiser les IBCs, dans le sens de minimiser le nombre de itérations de l'ASM pour arriver à la convergence. Ainsi, de façon similaire à l'optimisation des TBCs faite dans la section ??, on va faire un très large ensemble de tests, afin de trouver les coefficients c_L et c_R qui fournissent la convergence la plus rapide. Dans un permier moment, on fera cet étude avec un pas de temps et un pas de espace fixés, afin d'analyser exclusivement l'influence du coefficient; en suite, on va introduire ces deux paramètres dans l'étude.

Comme on connaît une solution de référence, le critère de convergence utilisé est

$$e^{\Omega,k} \leq \varepsilon$$

avec $\varepsilon = 10^{-9}$ et l'erreur $e^{\Omega,k}$, pour chaque itération k , définie comme dans (52).

Afin de simplifier les tests et d'éviter des coûts de calcul trop élevés, on va considérer toujours $c_L = c_R = c$ dans le procès d'optimisation. Le range des coefficients testés est $[-10.0, 20.0]$ (choisi après des tests initiaux pour identifier un intervalle approprié), avec un pas égal à 0.1 entre eux (ou encore plus petit, jusqu'à 0.005, dans les régions proches des coefficients optimaux). Le nombre maximal d'itérations est 100.

Comme une dernière remarque, on rappelle que tous les tests seront réalisées au long d'un seul pas de temps.

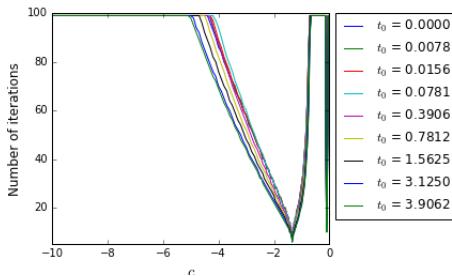
6.4.1 Tests variant l'instant initial et la position de l'interface

On va utiliser un pas de temps $\Delta t = 20/2560 = 0.0078125$ et une taille de maille $\Delta x = 12/500 = 0.024$ fixés. Par ailleurs, on va mettre en place deux ensembles de tests, qui nous permettront d'étudier la vitesse de convergence avec des différentes conditions initiales et différents tailles des subdomaines :

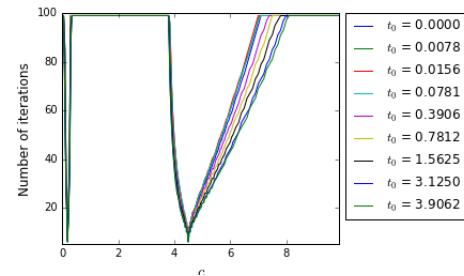
1. Tests variant l'instant initial t_0 , avec l'interface fixée sur le centre du monodomaine $\Omega = [-6, 6]$;
2. Tests variant la position de l'interface ($x_{interface} = -L + \alpha 2L$, où $L = 6$ et $0 < \alpha < 1$), pour un instant initial $t_0 = 0.78125$ fixé.

Dans tous les cas, la solution de référence u^{ref} sera la solution du problème dans le monodomaine (46), calculée dans $[t_0, t_0 + \Delta t]$.

Les résultats obtenus sont résumés dans les figures 37 and 38, avec le nombre d'itérations en fonction du coefficient c . Par souci de clarité, les résultats pour les coefficients négatifs et positifs sont présentés dans des graphes séparés. Ils montrent des comportements très similaires pour toutes les courbes, avec deux minima for $c < 0$ et deux autres for $c > 0$, avec approximativement la même valeur dans tous les cas (environ -1.35, -0.10, 0.20 and 4.50). Les minima les plus proches de zéro sont associés à des courbes très discontinues, tandis que les autres deux minima sont associés à des courbes plus lisses (voir les détails dans les figures 36c-36f et 37c-37f). Finalement, on remarque que, pour quelques courbes, le minimum est associée aux coefficients les plus proches de zéro, et pour les autres courbes, il est associée aux autres coefficients. Néanmoins, dans tous ces cas, les nombres optimales d'itérations sont similaires (entre cinq et sept).



(a) Vue générale des coefficients négatifs



(b) Vue générale des coefficients positifs

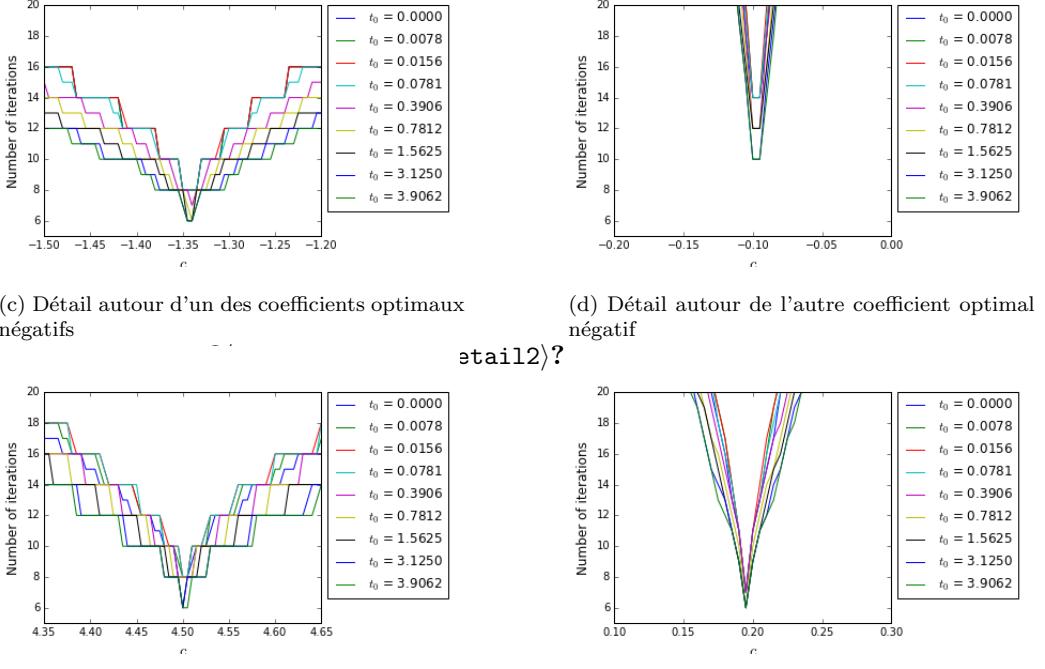
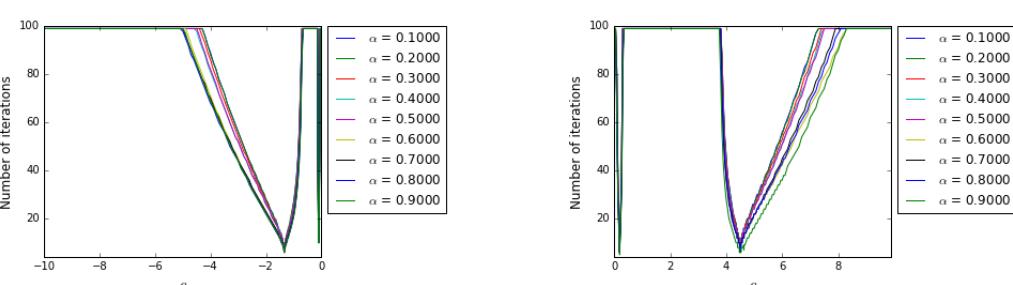


Figure 37: Nombre d'itérations jusqu'à convergence en fonction du coefficient des TBCs, pour une interface fixée et des différentes valeurs de t_0



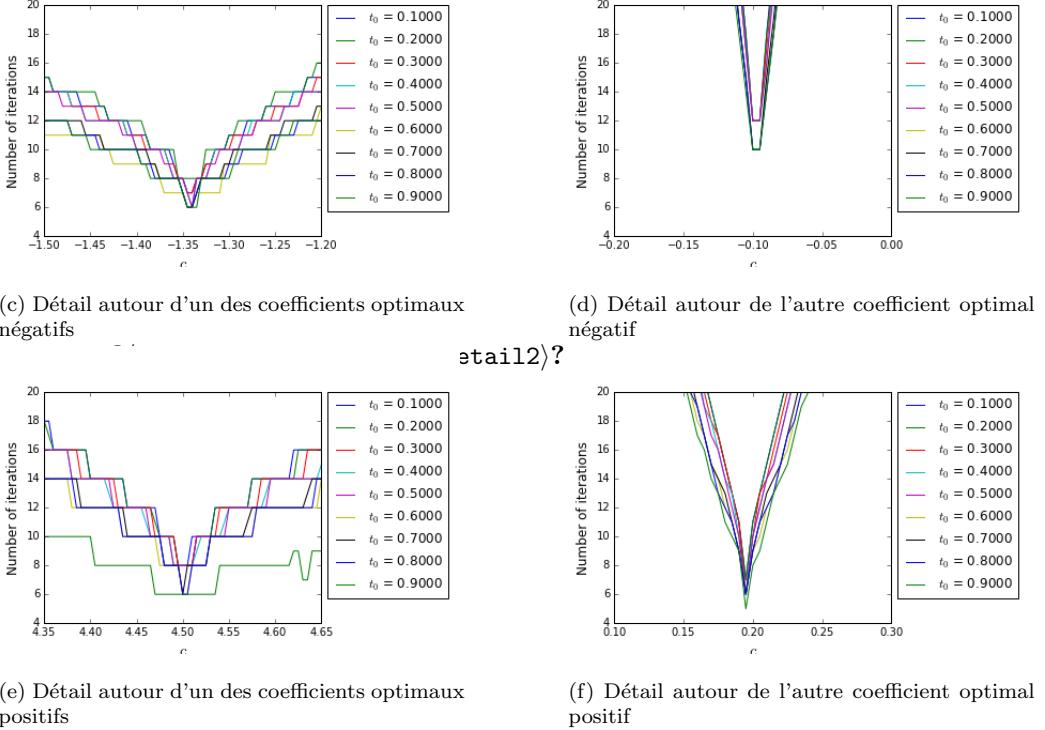


Figure 38: Nombre d'itérations jusqu'à convergence en fonction du coefficient des TBCs, pour t_0 fixé et des différentes positions de l'interface

La figure 39 montre l'évolution de l'erreur, en fonction des itérations, pour cinq coefficients c qui ont fourni les convergences les plus rapides, pour un temps initial et une position de l'interface fixés. Pour des autres valeurs de t_0 et α , le graph est similaire, en ce qui concerne le nombre d'itérations et le fait que la convergence est plus régulière pour les coefficients les plus proches de zéro, en comparaison aux autres coefficients optimaux.

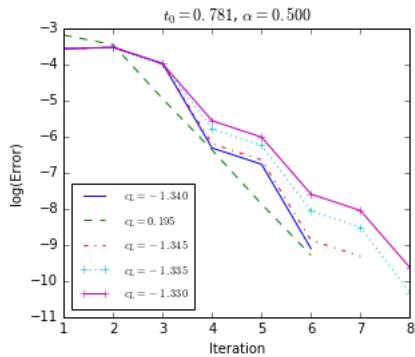


Figure 39: Évolution de l'erreur, en fonction des itérations, pour les tests les plus rapides

6.4.2 Tests variant Δt and Δx

Après vérifier que la méthode se comporte de façon similaire pour toute condition initiale (i.e., pour tout t_0) et toute position de l'interface, on a fixé ces paramètres ($t_0 = 0$ and $\alpha = 0.5$) et on a fait des nouveaux tests avec des différentes valeurs de Δt (avec $\Delta x = 12/250$ fixé) et des différentes valeurs de Δx (avec $\Delta t = 0.02$ fixé).

Le nombre d'itérations en fonctions des coefficients, pour quelques tests, est montré dans les figures 40 et 41. La figure 42 présente le coefficient optimal pour chaque Δt ou Δx . En considérant la remarque qu'on a fait concernant les résultats similaires (i.e., le nombre d'itérations jusqu'à convergence) pour les quatre coefficients optimaux, on a tenu en compte, pour la construction des courbes de la figure 42, seulement les minima les plus lointains de zéro: ceci a été fait parce que, comme montre les figures 40 et 41, ces minima ont une forte dépendance de Δt et Δx , et on va chercher à étudier cette relation.

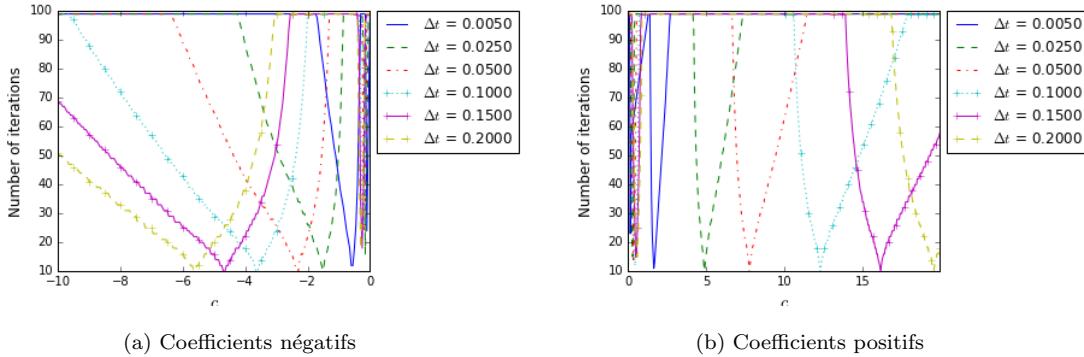


Figure 40: Nombre d'itérations jusqu'à convergence en fonction du coefficient pour $2N = 250$ fixé et des différentes valeurs de Δt

fig:niterxCoefVarDt

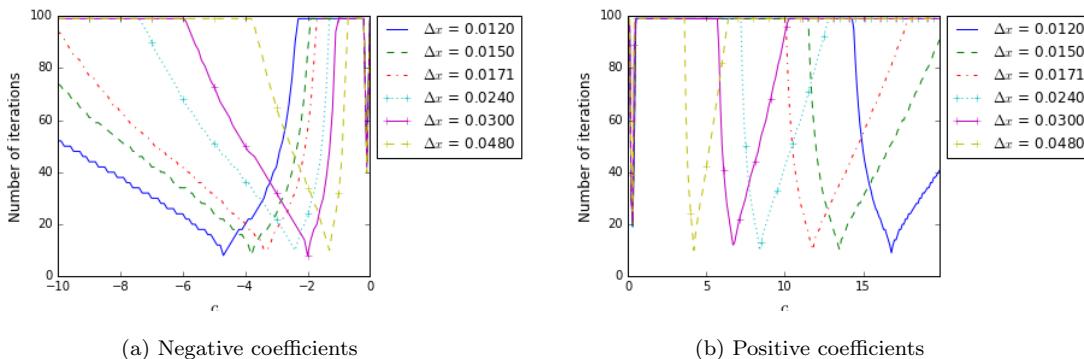


Figure 41: Nombre d'itérations jusqu'à convergence en fonction du coefficient des TBCs pour $\Delta t = 0.02$ et des différentes valeurs de Δx

fig:niterxCoefVarDx

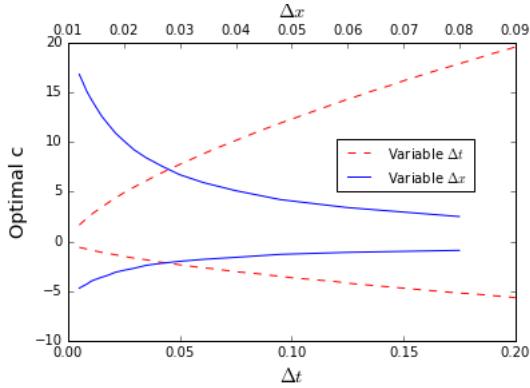


Figure 42: Coefficients optimaux en fonction du pas de temps et de la taille de maille

`CoeffVarDxDtCorrectN`

La figure 42 suggère que le coefficient optimal dépende de $(\Delta t)^\nu$ et $(\Delta x)^\eta$, avec $0 \leq \nu \leq 1$ et $\eta < 0$. En fait, en faisant quelques régressions avec Δt ou Δx fixé, on peut conclure que $\nu = \frac{2}{3}$ et $\eta = -1$ fournissent des courbes de régression très bien adaptées (avec des coefficients de détermination R^2 plus grandes que 0.99), pour les cas des coefficients positifs et négatifs (même que chacun de ces cas corresponde à des courbes différents). Ainsi, on va chercher à modéliser une fonction

$$c_{opt}(\Delta t, \Delta x) = \kappa + \alpha(\Delta t)^{\frac{2}{3}} + \beta \frac{1}{\Delta x} + \gamma \frac{(\Delta t)^{\frac{2}{3}}}{\Delta x} \quad (54) \text{ ?eq:regression2D?}$$

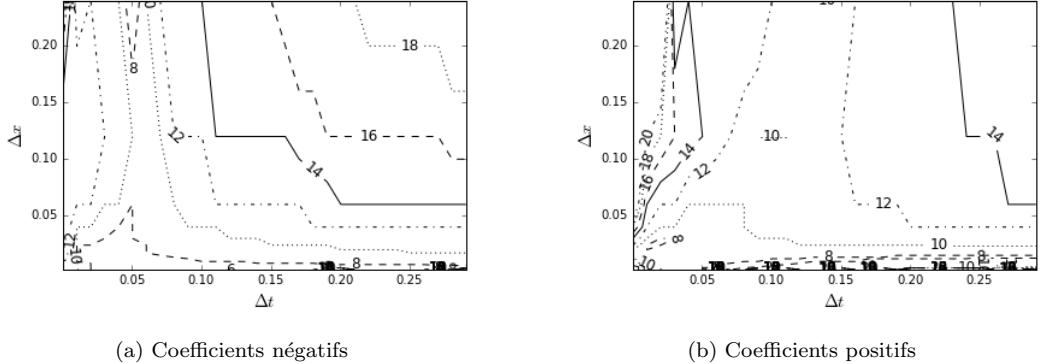
Une régression utilisant les coins du rectangle $[0.001, 0.1] \times [\frac{12}{100}, \frac{12}{1000}]$ et quinze points à l'intérieur fournit les surfaces

$$c_{opt}^+(\Delta t, \Delta x) = 0.0775 - 0.3353(\Delta t)^{\frac{2}{3}} - 0.0012 \frac{1}{\Delta x} + 2.7407 \frac{(\Delta t)^{\frac{2}{3}}}{\Delta x} \quad (55) \text{ [eq:regression2DPos]}$$

$$c_{opt}^-(\Delta t, \Delta x) = -0.0583 - 1.5024(\Delta t)^{\frac{2}{3}} - 0.0006 \frac{1}{\Delta x} - 0.7287 \frac{(\Delta t)^{\frac{2}{3}}}{\Delta x} \quad (56) \text{ [eq:regression2DNeg]}$$

respectivement pour les coefficients optimaux positifs et négatifs. Les coefficients de détermination de chaque régression sont $R^{2,+} = 0.9999894$ et $R^{2,-} = 0.9998993$, ce qui indique une bonne représentation.

Afin de valider les expressions (55) and (56), elles ont été utilisées pour calculer le coefficient optimal pour des plusieurs points $(\Delta t, \Delta x)$, avec $\Delta t \in [0.0005, 0.3]$ et $\Delta x \in [\frac{12}{5000}, \frac{12}{50}]$. Comme montre la figure 43, pour la plupart des points dans le domaine considéré, le coefficient optimal calculé fournit une convergence rapide vers la solution du monodomaine, avec moins de 20 itérations (ou encore moins que 12 itérations), sur une grande région du domaine).



(a) Coefficients négatifs

(b) Coefficients positifs

Figure 43: Lignes de contour du nombre d’itérations jusqu’à convergence, en utilisant les coefficients optimaux $c_{opt}^+(\Delta t, \Delta x)$ obtenus à partir des régressions.

`egressionValidation`

Les nombres d’itérations montrées dans la figure 43 ne sont pas les plus petites qu’on est capable de trouver (cf. les figures ?? jusqu’à ??), parce que les expressions (55) et (56) sont des régressions construites à partir de coefficients optimaux obtenus parmi un ensemble discret de valeurs possibles. Néanmoins, elles donnent des très bonnes approximations pour le c optimal pour chaque $(\Delta t, \Delta x)$, et ainsi on peut chercher dans une petite région autour du c_{opt} calculé pour obtenir une convergence encore plus rapide.

6.5 Partial conclusion

The results presented in this section show that the Domain Decomposition Method proposed here, consisting in the Additive Schwarz Method with our approximate TBCs, is able to provide a fast convergence toward the solution of the monodomain problem. Therefore, we reached our goals of solving the dispersion equation in a finite domain divided in two subdomains.

Moreover, the results of the optimization tests are very satisfying regarding a more general application of our method. Firstly, for fixed spatial and temporal discretizations, we obtained optimal coefficients for the method independently of the initial solution and the size of the subdomains (i.e., independently of the initial instant and the position of the interface). Secondly, we obtained good regression curves for the optimal coefficient as function of Δt or Δx , which could allow the application of the model, with fast convergence, for tests different for the ones made in this study.

References

- [Xavieretal2008] [1] X. Antoine, A. Arnold, C. Besse, M. Ehrhardt, and C. Schädle. A review of Transparent Boundary Conditions or linear and nonlinear Schrödinger equations. *Communications in Computational Physics*, 4(4):729–796, October 2008.
- [arpaia] [2] L. Arpaia and M. Ricchiuto. Mesh adaptation by continuous deformation. Basics : accuracy, efficiency, well balancedness. Technical report, INRIA Bordeaux Sud-Ouest, 2015. RR-8666, [hal-01102124](https://hal.inria.fr/hal-01102124).
- [askes] [3] H. ASKES, L. J. SLUYS, and B. C. C. JONG. Remeshing techniques for R-adaptives and combined H/R-adaptive analysis with application to 2D/3D crack propagation. *European Congress on Computational Methods in Applied Sciences and Engineering*, 2000.
- [balagurusamy2008] [4] E. Balagurusamy. *Numerical methods*. Tata McGraw-Hill, New Delhi, 2008.
- [BBM1971] [5] T. B. Benjamin, J. L. Bona, and J. J. Mahony. Model equations for long waves in nonlinear dispersive systems. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 272(1220):47–78, 1972.
- [besse2015] [6] C. Besse, M. Ehrhardt, and I. Lacroix-Violet. Discrete Artificial Boundary Conditions for the Korteweg-de Vries Equation. working paper or preprint, Jan. 2015.
- [Bonneton2011] [7] P. Bonneton, F. Chazel, D. Lannes, F. Marche, and M. Tissier. A Splitting approach for the fully nonlinear and weakly dispersive Green-Naghdi model. *J. Comput. Phys.*, 230:1479–1498, 2011.
- [CarterCienfuegos2011] [8] J. D. Carter and R. Cienfuegos. The kinematics and stability of solitary and cnoidal wave solutions of the Serre equations. *European Journal of Mechanics B/Fluids*, 2011.
- [vecNormal] [9] L. Chen. Programmim of finite elements in matlab. Notes du cours Math 226 : Computational PDEs (Univeristy of California Irvine). <http://www.math.uci.edu/~chenlong/226/Ch3FEMCode.pdf>.
- [laplaceTransform] [10] P. Cheung. Lecture 6 - Frequence-domain analysis : Laplace Transform. [Imperial College of London - Signals and Linear Systems.

- [cecile_these]** [11] C. Dobrsynski. *Adaptation de maillage anisotrope 3D et application à l'aéro-thermique des bâtiments*. PhD thesis, Université Pierre et Marie Curie - Paris VI, 2005.
- [ducrot]** [12] V. Ducrot and P. Frey. Anisotropic level set adaptation for accurate interface capturing. In R. Garimella, editor, *Proceedings of the 17th International Meshing Roundtable*, pages 159–176. Springer Berlin Heidelberg, 2008.
- [frey_alauzet]** [13] P. J. Frey and F. Alauzet. Anisotropic metrics for mesh adaptation. *Computational Fluid and Solid Mechanics*, 2003.
- [Gander2008]** [14] M. J. Gander. Schwarz methods over the course of time. *ETNA. Electronic Transactions on Numerical Analysis [electronic only]*, 31:228–255, 2008.
- [Japhet2003]** [15] C. Japhet and F. Nataf. The best interface conditions for Domain Decomposition methods : Absorbing Boundary Conditions. <http://www.ann.jussieu.fr/nataf/chapitre.pdf>.
- [karniadakis1995]** [16] G. Karniadakis. Toward a numerical error bar in CFD. *ASM Journal of Fluids Engineer*, 117(1):7–9, 1995.
- [Khorsand2014]** [17] Z. Khorsand and H. Kalisch. On the shoaling of solitary waves in the kdv equation. *Coastal Engineering Proceedings*, 1(34):44, 2014.
- [Lions1988]** [18] P.-L. Lions. On the Schwarz alternating method.I. In G. J. R.Glowinski, G.Golub, editor, *Proceedings of the First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 1–42, 1988.
- [Lions1990]** [19] P.-L. Lions. On the Schwarz alternating method III: a variant for nonoverlapping sub-domains. In J. P. O. W. T. Chan, R. Glowinski, editor, *Proceedings of the Third International Conference on Domain Decomposition Methods*, page 202–223, 1990.
- [loseille]** [20] A. Loseille and R. Löhner. Robust boundary layer mesh generation. In X. Jiao and J.-C. Weill, editors, *Proceedings of the 21st International Meshing Roundtable*, pages 493–511. Springer Berlin Heidelberg, 2013.
- [leo]** [21] L. Nouveau, H. Beaugendre, C. Dobrzynski, R. Abgrall, and M. Ricchiuto. An explicit residual distribution scheme combined to mesh adaptation for solving the unsteady penalized Navier Stokes equations with a splitting approach. **????**, Juin 2015. Preprint.

- [**rippa**] [22] S. Rippa. Long and thin triangles can be good for linear interpolation. *SIAM Journal on Numerical Analysis*, 29(1):pp. 257–270, 1992.
- [**roache1997**] [23] P. J. Roache. Quantification of uncertainty in Computational Fluid Dynamics. *Annual Review of Fluid Mechanics*, 29:123–160, 1997.
- [**conservationLaws2002**] [24] University of Stanford. Partial differential equations of applied mathematics - lecture 3 : Conservation laws. <http://web.stanford.edu/class/math220a/handouts/conservation.pdf>, 2002.
- [**zheng2008**] [25] C. Zheng, X. Wen, and H. Han. Numerical solution to a linearized kdv equation on unbounded domain. *Numerical Methods for Partial Differential Equations*, 24(2):383–399, 2008.