Simulation parallèle d'écoulements instationnaires par la méthode SPH

Rémi Croquet, Jean-Marc Cherfils, Grégory Pinon

Simulation parallèle d'écoulements instationnaires par la méthode SPH

R. CROQUET[a], J.-M. CHERFILS[a], G. PINON[a]

a. Laboratoire Ondes et Milieux Complexes, 53 rue de Prony, 76058 LE HAVRE

Résumé :

Cet article présente une implémentation parallèle d'un canal à houle numérique, modélisé par la méthode Smoothed Particle Hydrodynamics. La répartition des calculs entre les processeurs est assurée par une décomposition de domaine, optimale pour les problèmes traités : propagation de houle, interactions fluides-structures. L'équilibre de la charge est alors maintenu au cours des simulations en modifiant la taille des sous-domaines. Dans la dernière section, les performances obtenues sont présentées pour plusieurs applications.

Abstract :

This article presents a parallel implementation of a wave flume, modelled by means of the Smoothed Particle Hydrodynamics method. Distribution of computations between processors is carried out by a domain decomposition, optimal for the considered problems : wave propagation, fluid-structure interactions. Load balancing is ensured during the whole simulation by dynamic modification of subdomains size. In the last section, performances will be presented for several applications.

Mots clefs : Méthode particulaire, décomposition de domaine, calcul parallèle, Smoothed Particle Hydrodynamics

1 Introduction

La méthode Smoothed Particle Hydrodynamics (SPH) est une méthode particulaire permettant notamment la modélisation d'écoulements instationnaires grandes échelles. Toutefois, la mise en oeuvre de simulations mettant en jeu ces phénomènes est rendue difficile par les temps de calcul important lorsque le nombre de particules devient grand. Pour éviter cette difficulté, nous avons donc produit un code parallèle basé sur une décomposition du domaine. Lors du développement, un effort tout particulier a été porté au maintien de l'équilibre de la charge de calcul entre les différents processeurs, problème difficile du fait de l'instationnarité. Nous présenterons donc ici quelques éléments sur la méthode SPH avant de justifier la démarche adoptée pour la parallélisation. Enfin, nous présenterons les performances du code parallèle sur deux types de problème.

2 La méthode SPH

La méthode Smoothed Particle Hydrodynamics a été introduite en 1977 par Monaghan, Gingold et Lucy [1] pour les besoins de l'astrophysique et a été étendue pour les écoulements hydrodynamiques par Monaghan [2]. Elle permet la discrétisation d'un milieu fluide en particules porteuses d'information. Ces particules peuvent alors servir de points d'interpolation pour discrétiser les équations aux dérivées partielles régissant le comportement du milieu. Dans le cas d'écoulements à surface libre, ces équations sont les équations d'Euler pour un fluide compressible auxquelles on ajoute une équation d'état, l'équation de Tait, permettant, avec des conditions limites, de clore le système (Cf. *Système continu*, équation 1) :

$$\begin{array}{rcl@{\qquad}c@{\qquad}rcl}
\frac{D\mathbf{x}}{Dt} &=& \mathbf{u} & & \frac{D\mathbf{x}_i}{Dt} &=& \mathbf{u}_i \\
\frac{D\mathbf{u}}{Dt} &=& -\frac{1}{\rho}\nabla P + \mathbf{g} & & \frac{D\mathbf{u}_i}{Dt} &=& -\frac{1}{\rho_i}\sum_j (P_j + P_i)\,\nabla W(\mathbf{x}_j - \mathbf{x}_i)V_j + \mathbf{g} \\
\frac{D\rho}{Dt} &=& -\rho\nabla.\mathbf{u} & \Longrightarrow & \frac{D\rho_i}{Dt} &=& -\rho_i\sum_j (\mathbf{u}_j - \mathbf{u}_i)\,\nabla W(\mathbf{x}_j - \mathbf{x}_i)V_j \\
P &=& \frac{\rho_0 c_0^2}{7}\left(\left(\frac{\rho}{\rho_0}\right)^7 - 1\right) & & P_i &=& \frac{\rho_0 c_0^2}{7}\left(\left(\frac{\rho_i}{\rho_0}\right)^7 - 1\right)
\end{array} \tag{1}$$

Système continu Système discrétisé

En modifiant la vitesse nominale du son c_0, il est alors possible de limiter la variation de la masse volumique et ainsi de reproduire le comportement d'un fluide incompressible, ce qui est généralement le cas en hydrodynamique.

La méthode SPH offre alors un schéma d'interpolation des variables s'appuyant sur la convolution par un noyau W des seconds membres des équations précédentes pour passer au système discrétisé (Cf. *Système discrétisé*, équation 1)). Ce noyau a la propriété de tendre vers la distribution de Dirac lorsque le pas de l'interpolation h tend vers 0. Dans notre implémentation, nous utilisons le noyau gaussien à support compact suivant :

$$W(s,h,\delta) = \frac{e^{-\left(\frac{s}{h}\right)^2} - e^{-\left(\frac{s}{\delta}\right)^2}}{2\pi \int_0^{\delta} r \left(e^{-\left(\frac{r}{h}\right)^2} - e^{-\left(\frac{\delta}{h}\right)^2}\right) dr} \tag{2}$$

où h correspond au pas d'interpolation et δ au rayon du support, ici $\delta = 3h$.

Après quelques manipulations algébriques destinées à rétablir le principe de réciprocité des actions entre les particules, on obtient le schéma numérique déterminé par les équations 1.

Les quantités $\frac{D\mathbf{x}_i}{Dt}$, $\frac{D\mathbf{u}_i}{Dt}$ et $\frac{D\rho_i}{Dt}$ sont alors intégrées en temps par un schéma numérique type Euler ou Runge-Kutta explicite. En itérant ce procédé, il est possible de simuler l'évolution du fluide au cours du temps.

3 Parallélisation par décomposition de domaine

3.1 Principe de la méthode

La méthode adoptée pour la parallélisation est basée sur la propriété de compacité du support du noyau W (Cf. eq. 2). Cette propriété implique en effet que deux particules suffisament éloignées n'auront pas d'influence immédiate l'une sur l'autre. Le principe de la méthode est alors de décomposer l'ensemble du domaine en autant de sous-domaines que l'on a de processeurs, chacun des sous-domaine possédant à peu près le même nombre de particules.

La figure (1) illustre cette idée et met en évidence l'apparition de bandes de largeur $3h$ (les zones rouges et vertes) à la frontière entre deux sous-domaines où les particules seront potentiellement en interactions avec des particules du sous-domaine voisin. Nous porterons donc une attention toute particulière au traitement de ces interfaces pour s'assurer que chaque processeur possède bien les informations nécessaires sur les particules situées dans les franges voisines.

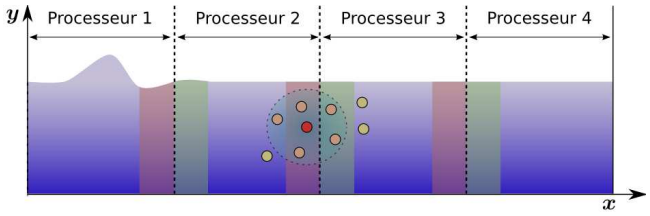


FIG. 1 – Principe de la décomposition du domaine

Dans le cas des applications que nous avons été amenées à traiter, et notamment la géométrie du canal à houle, la décomposition du domaine se fait, à l'instar de la figure (1), en bandes verticales. D'autres approches plus générales existent [3] et [4] mais ce choix est justifié par la forme allongée du domaine (de l'ordre de plusieurs mètres en longueur pour quelques dizaines de centimètres de hauteur) qui permet de minimiser la taille des interfaces et donc le nombre de particules situées dans ces zones.

3.2 Mise en place de l'algorithme

L'algorithme que nous avons mis en place est représenté sur la figure (2). Sur ce schéma, les cellules vertes représentent les actions réalisées simultanément par chaque processeur et les cellules rouges correspondent aux phases de communications.

Après que chaque processeur ait initialisé le jeu de particules dont il a la charge (étape 1), la phase itérative de résolution des équations et d'intégration en temps commence par la détection des particules de bord.

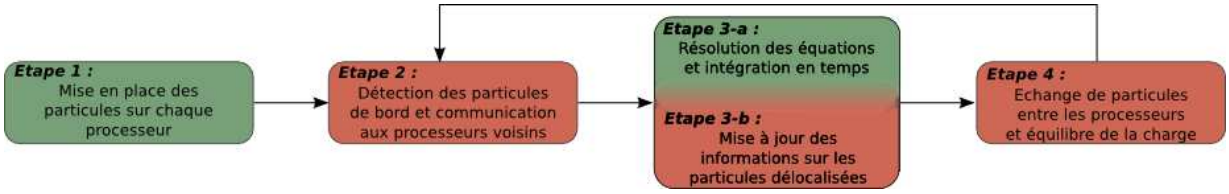


FIG. 2 – Mise en place de la parallélisation

Chaque processeur effectue une boucle sur l'ensemble des particules de son sous-domaine (étape 2). Si la coordonnée en x d'une particule est suffisament près de celle de la frontière, l'indice de cette particule est ajouté à une liste permettant de retrouver toutes les particules du bord du sous-domaine. Toutes les particules repérées comme telles sont alors répliquées à l'identique sur le processeur voisin (étape 2).

Dans le cas d'un schéma d'intégration en temps de type Runge-Kutta, les informations sur ces particules répliquées devront être mises à jour à chaque sous-pas de temps (étape 3b). A la fin de chaque itération (étape 3a), les particules qui passent d'un sous-domaine à son voisin voient leur gestion transféré d'un processeur à l'autre, tout en s'assurant que les processeurs conservent un nombre de particules quasi-constant (étape 4).

L'ensemble des communications est assuré au moyen d'une bibliothèque spécifique de routines utilisant le paradigme MPI. Ces routines permettent notamment d'envoyer par un simple appel de fonction l'ensemble des informations sur toutes les particules de bord aux processeurs voisins, la mise à jour des informations concernant une particule répliquée ou encore le transfert d'une particule d'un processeur vers un autre. De plus, afin de pouvoir faire évoluer facilement le code, une structure *ad hoc* a été créée afin de regrouper toutes les variables liées à la parallélisation.

Lors de leur mise en place, nous avons fait particulièrement attention à l'optimisation des routines de communication. En effet, nous avons d'une part pris soin d'utiliser au maximum des communication non bloquantes afin de masquer les phases de communications par des calculs. D'autre part, nous avons fait l'hypothèse de n'utiliser qu'un nombre pair de processeurs afin de pouvoir faire l'ensemble des communications en seulement quatre phases de communication. Sur la figure (3), nous présentons l'organisation des communications correspondantes (Cf. étape 2, 3b et 4 de la figure (2)) en représentant les processeurs en mode réception par des ronds verts, ceux en émission par des ronds rouges et ceux inactifs par des ronds blancs.

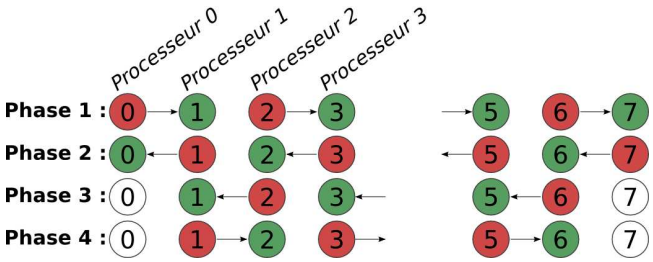


FIG. 3 – Organisation des communications

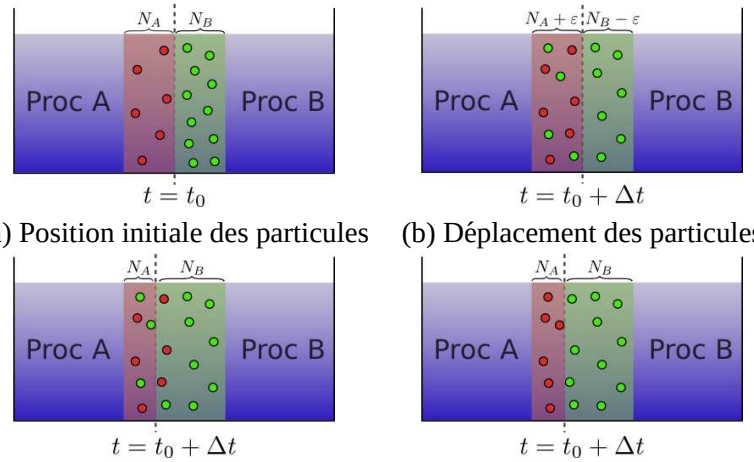
L'écriture des résultats dans des fichiers se fait processeur par processeur, la synchronisation étant effectuée par de simples fonctions MPI. La lecture est, quant à elle, réalisée simultanément par chaque processeur. Les fonctions de lecture et d'écriture de MPI-2 ont été testées avec différentes implémentations mais leur utilisation n'a pas été fructueuse.

3.3 Equilibre dynamique de la charge sur chaque processus

Le transfert d'une particule sortant de son sous-domaine au processeur voisin (Cf. étape 4 de la figure (2)) permet d'éviter des communications inutiles. Toutefois, ce choix introduit le risque de voir apparaitre à terme un déséquilibre dans le nombre de particules gérées par chacun des processeurs (figure (4-b)). Un algorithme a été mis en place pour pallier ce problème en adaptant la taille de chacun des sous-domaines.

Cet algorithme repose sur une propriété imposée par une condition de type Courant-Friedrich-Lévy (CFL). Pour que le schéma soit stable, une particule ne doit pas se déplacer d'une distance supérieure au pas d'interpolation h. Dans notre cas, le rayon du support du noyau est fixé à $3h$. Cette distance correspond également à la largeur de chacune des bandes. Une particule n'est donc susceptible de passer d'un domaine à l'autre que si

elle se trouvait initialement dans la frange.



(a) Position initiale des particules (b) Déplacement des particules

(c) Positionnement de la frontière (d) Transfert des particules

FIG. 4 – Algorithme d'équilibrage de la charge

Le principe est alors de conserver le même nombre de particules dans chacune des deux bandes avant (figure (4-a)) et après (figure (4-b)) leur déplacement. Sur les figures (4), on appelle N_A (resp. N_b) le nombre de particules présentes dans la *bande* à gauche (resp. à droite) de la frontière. Pour conserver ces quantités de particules, la frontière est repositionnée de manière à conserver le nombre de particules dans les franges (figure (4-c)). Cette nouvelle position est obtenue en déterminant la particule ayant la N_A ième plus grande abscisse parmi les $N_A + N_B$ particules du bord. Ce calcul, qui n'est réalisé que par l'un des deux processeurs, est fait en utilisant un algorithme rapide, Quickselect [5], dont la complexité est en $O(log(N_A + N_B))$. Le temps consacré à cette tâche est alors négligeable devant les temps de calcul.

Une notion de "bilan de particules par processeur" a également été mise en place. En effet, nous avons pu observer qu'un déséquilibre d'une particule pourrait advenir à certaines occasions. De tels déséquilibres se cumulant au fil des itérations, ils pouvaient entrainer, à l'échelle d'une simulation, d'importantes répercussions en terme de performances. Nous avons alors introduit une mémoire permettant à chaque processeur de connaitre son défaut ou son excès de particules par rapport au début de la simulation, le nombre total de particules restant inchangé. En cas d'excès, le processeur aura alors tendance à chercher à rapprocher la frontière vers lui afin de réduire son nombre de particule et inversement.

Afin de tester la robustesse de notre procédure d'équilibrage, celle-ci a été étendue à la simulation d'une rupture de barrage suivie d'un impact sur un mur. Ce cas, souvent employé pour la validation des méthodes SPH [6], présentant des propriétés géométriques proches du canal à houle, l'adaptation a donc été tout à fait naturelle. Du fait de la cinématique beaucoup plus complexe, l'équilibrage des charges est un problème difficile. La figure (5) représente une simulation parallèle. Les images de gauche représentent l'équilibrage des charges, chaque bande de couleur correspondant à un ensemble de particules gérées par le même processeur. On voit sur la figure de gauche pour $t = 0.6s$ que les domaines s'étirent pour conserver un nombre de particule constant. A l'inverse, après l'impact, les particules ont tendance à être concentrées près du mur, contraignant les processeurs les plus à droite (en orange et rouge) à réduire considérablement leur sous-domaine.

4 Performances

Une étude des performances du code a été menée pour observer le coût de la parallélisation. Les figures (6-a) et (6-b) représentent respectivement l'accélération ($A(p) = \frac{\text{Temps de calcul séquentiel}}{\text{Temps de calcul avec p processeurs}}$) et l'efficacité ($eff(p) = \frac{A(p)}{p}$) du code pour des discrétisations différentes. Ces résultats sont obtenus sur le cluster de notre laboratoire (le LOMC) composé de 4 noeuds possédant chacun deux Xeon quad core.

Tout d'abord, nous pouvons constater que les performances de la parallélisation s'améliorent très largement lorsque le nombre de particules augmente. Ce résultat s'explique par l'évolution de la répartition du temps de calcul en fonction du nombre de particules. Sur des problèmes avec peu de particules, le temps consacré au calcul est suffisament faible pour que l'effet des routines liées à la parallélisation se fasse ressentir. En revanche,

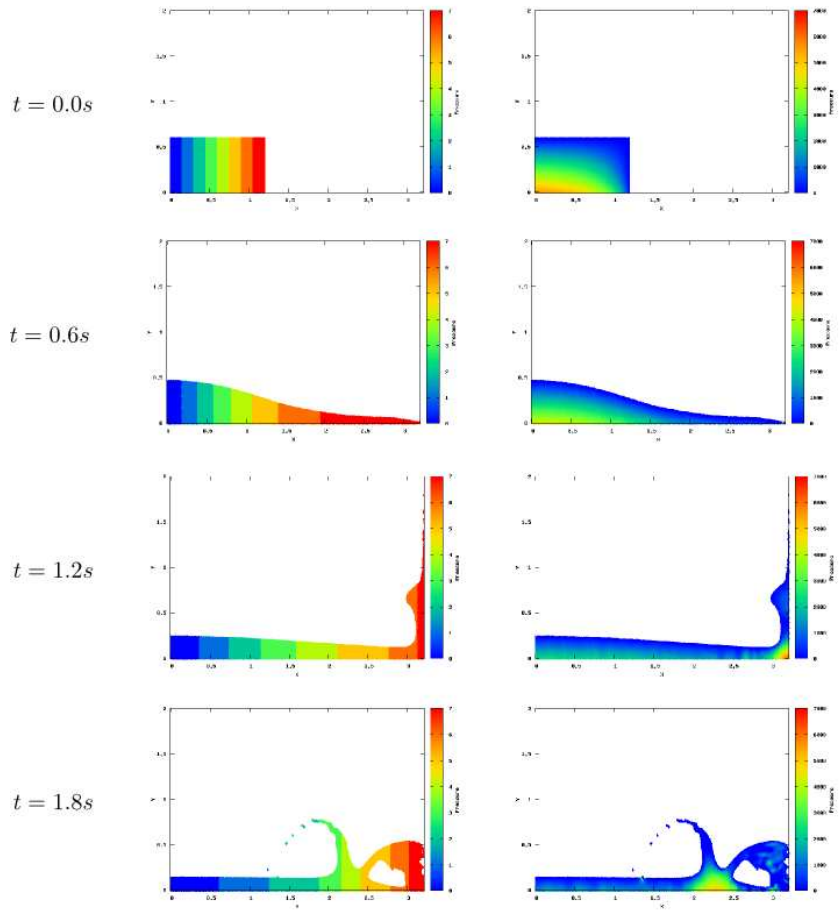


FIG. 5 – Exemple d'équilibrage sur une rupture de barrage : à gauche, la répartition des particules par processeur et à droite, les champs de pression associés.

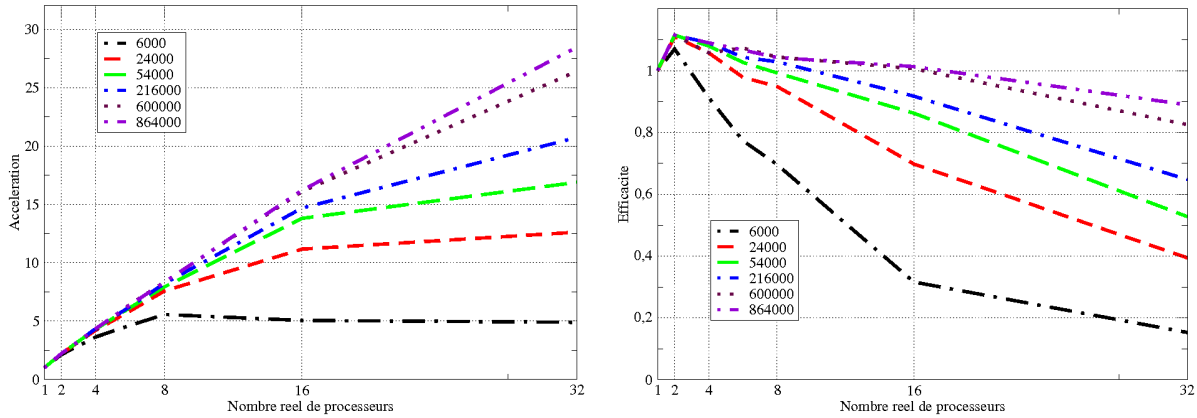


FIG. 6 – Mesure de l'efficacité du code parallèle sur une simulation de type canal à houle : à gauche, l'accélération $A(p)$ et à droite, l'efficacité $eff(p)$.

on voit sur la figure (6) qu'à partir de $600\,000$ particules, l'efficacité devient supérieure à 80%. Avec peu de processeurs, on peut avoir une efficacité supérieure à 1 ; ceci peut s'expliquer pour une question de vitesse d'accès aux données stockées en tableaux (Cf. [4] pour plus de détails).

A titre de comparaison, nous avons fait le même type de mesures pour le problème de la rupture de barrage. On constate alors que les performances de la parallélisation sont moins bonnes pour ces simulations que pour le canal à houle. Cette différence s'explique par la grande déformation du domaine inhérente au problème de

la rupture de barrage. En effet, pour maintenir les charges de calcul constantes sur chaque processeur, de nombreux échanges de particules sont nécessaires.

Afin de confirmer les résultats déjà obtenus, nous avons procédé à une analyse plus fine de la répartition du temps de calcul grâce au logiciel Xprofiler, mis à notre disposition par le CRIHAN. Afin de bien prendre en compte les effets de l'équilibrage de la charge, cette étude a été menée sur le problème de la rupture de barrage. Pour une simulation de $10\,000$ itérations avec $5\,000$ particules et 4 processeurs, nous obtenons la répartition suivante :

	Communications	Routine liée à la parallélisation	Détection des interactions	Résolution du schéma numérique
Temps (en ms)	8,72	1,98	1017	605.47
Proportion	0.53%	0.12%	62.27%	37,07%

Ce tableau indique d'une part que le coût de la parallélisation est très faible, et d'autre part que la détection des interactions entre les particules est la phase la plus contraignante. Diverses optimisations ont alors été menées afin de pallier ce dernier problème. D'un côté des méthodes rapides de détection de ces interactions ont été implémentées (*list search* [6] et *tree search* [7]) et comparées. Dans notre cas, le *list search* s'avère être beaucoup plus performant. Sa complexité est effectivement en N (contre $Nlog(N)$ pour le *tree search*) sous réserve que toutes les particules possèdent le même pas d'interpolation. Par ailleurs, afin de réduire la fréquence de recherche des interactions, nous avons mis en place une méthode de Verlet. Toutefois, cette dernière approche ne s'est pas révélées aussi fructueuse que pour Colagrossi [6]. Ces résultats s'expliquent d'une part par la parallélisation du code et d'autre part par notre traitement des conditions aux bords.

5 Remerciements

Nous remercions le Centre de Ressources Informatiques de Haute-Normandie (CRIHAN) pour leur apport en ressources, support technique, etc.

6 Conclusion

Une méthode parallèle efficace et bien adaptée aux problèmes traités à été mise en place. Pour un niveau de discrétisation réaliste, l'accélération produite est en effet de l'ordre du nombre de processeurs utilisés. Ces résultats nous confortent dans l'idée que la méthode SPH est bien adaptée à la parallélisation. Ces performances rendent aussi envisageables des simulations sur de grands domaines ou le passage à un code 3D, bien plus exigeant en terme de temps de calculs. Néanmoins, ce code est déjà utilisé pour l'étude de la propagation d'ondes et des interactions fluide-structure (Cf. [8]).

Références

[1] Gingold R. and Monaghan J. Smoothed particle hydrodynamics : theory and application to non-spherical stars. Mon. Not. R. astr. Soc., 181, 375–389, 1977.

[2] Monaghan J. Simulating free surface flows with SPH. J. Comput. Phys., 110(2), 399–406, 1994.

[3] Dubinski J. A parallel tree code. New Astronomy, 1, 133–147, 1996.

[4] Oger G. Aspects théoriques de la méthodes SPH et applications à l'hydrodynamique à surface libre. Thèse, Ecole Centrale de Nantes et Université de Nantes, 2006.

[5] Press W., Flannery B., Teukolsky S., and Vetterling W. Numerical recipes in FORTRAN. Cambridge Univ. Pr., Cambridge, Mass. [u.a.], 2. ed. edition, 1992.

[6] Colagrossi A. A Meshless Lagrangian Method for Free-Surface and Interface Flows with Fragmentation. Thèse, Universita di Roma, 2005.

[7] Hernquist L. and Katz N. Treesph : A Unification of SPH with the Hierarchical Tree Method. Astrophys. Sp. Science, 70, 419–446, June 1989.

[8] Cherfils J. Modélisation de l'interaction houle-plaque horizontale immergée par la méthode SPH. Soumis au 19ième Congrès Français de Mécanique, 2009.

[9] Davé R., Dubinski J., and Hernquist L. Parallel TreeSPH. New Astronomy, 2, 277–297, 1997.