

A novel generalized diffusive SPH model: Theoretical analysis and 3D HPC implementation



Jose Luis Cercos-Pita

Escuela Técnica Superior de Ingenieros Aeronáuticos (ETSIA)
UNIVERSIDAD POLITÉCNICA DE MADRID

Tesis doctoral

Director de tesis:
Antonio Souto-Iglesias

February 2016

If all else fails, immortality can always be assured by spectacular error.

John Kenneth Galbraith

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Jose Luis Cercos-Pita
February 2016

Acknowledgements

And I would like to acknowledge ...

Antonio Souto-Iglesias

CEHINAV

Family

Tony and Andrea

Krishna Kumar (Thesis template)

Calderon

Table of contents

Nomenclature	xiii
1 Introduction	1
1.1 Motivation and background	1
1.1.1 General	1
1.1.2 GPGPU, a new computing paradigm	2
1.1.3 State of the art of SPH	4
1.1.4 The SPH community	6
1.2 Objectives	6
1.3 Structure of the thesis	7
2 Numerical model	9
2.1 Governing equations	9
2.1.1 Boundary conditions	10
2.1.2 Initial conditions	12
2.2 SPH methodology	13
2.2.1 SPH Continuous model	13
2.2.2 SPH Discretized model	14
2.2.3 Boundary conditions	15
2.3 Numerical scheme	20
3 Diffusive terms in the mass conservation equation	21
3.1 General	21
3.2 Consistency conditions and desirable features	21
3.2.1 Condition 1. Conservation of mass consistency	22
3.2.2 Condition 2. Consistency close to the free surface	22
3.2.3 Condition 3. Intrinsically global mass conservation	22
3.2.4 Condition 4. No tuning of parameters	22

3.3	Existing diffusive terms in the literature	23
3.3.1	The diffusive term proposed by Vila (1999)	23
3.3.2	The diffusive term proposed by Ferrari et al. (2009)	25
3.3.3	The diffusion term proposed by Molteni and Colagrossi (2009) and Antuono et al. (2010)	26
3.3.4	A diffusion term inspired by Fatehi and Manzari (2011) and Hashemi et al. (2011)	26
3.4	Summary of diffusive δ -SPH terms	27
4	Energy equations	29
4.1	Viscous dissipation function	31
4.2	δ -SPH dissipation function	31
4.2.1	Molteni and Colagrossi (2009)	32
4.2.2	Fatehi and Manzari (2011)	32
4.3	BC dissipation function	32
4.3.1	Kernel correction	33
4.3.2	Boundary forces	33
4.3.3	Fluid extensions	36
4.3.4	Boundary Integrals	38
5	AQUAgpusph features	39
5.1	Free software	39
5.2	Modular application	40
5.2.1	General	40
5.2.2	Variables	41
5.2.3	Tools	41
5.2.4	Reports	42
5.2.5	Modularization	43
5.3	OpenCL acceleration	44
5.3.1	General	44
5.3.2	More powerful devices and lower costs	44
5.3.3	Hardware diversification	45
5.3.4	Applications with the capability of being modified	45
5.4	Python extensible	46
5.5	muParser fast math parser	47
5.6	Wide variety of boundary conditions	47
5.6.1	General	47

5.6.2	Solid boundary conditions	48
5.6.3	Inflow/outflow	53
5.6.4	Periodic	53
5.6.5	Symmetry	53
6	Verification and validation	57
6.1	General	57
6.2	Lid-driven cavity flow	57
6.2.1	Case description	57
6.2.2	Implementation details	59
6.2.3	Results	60
6.3	Standing wave	62
6.3.1	Case description	62
6.3.2	Implementation details	63
6.3.3	Results	64
6.4	Taylor-Green vortex	64
6.4.1	Case description	64
6.4.2	Implementation details	67
6.4.3	Results	68
6.5	Moving square inside a rectangular box	69
6.5.1	Case description	69
6.5.2	Implementation details	72
6.5.3	Results	75
7	Practical applications	79
7.1	Simulation of Earthquake Sloshing Loads in a Nuclear Reactor	79
7.1.1	General	79
7.1.2	Case description	80
7.1.3	Results	83
7.2	Coupling free surface tanks with ship motions codes	83
7.2.1	General	83
7.2.2	Co-simulation strategy	86
7.2.3	Linear ship motions model (SeaFEM)	89
7.2.4	Non-linear ship motions model (SHIXDOF)	91
8	Conclusions and future work	95

Thesis publications	101
Refereed papers	101
Conference papers	102
Other publications	105
References	107
References	109

Nomenclature

Symbols

Co	Courant factor
c_s	Speed of sound
Δr	Initial distance between particle
\mathcal{E}_c	Energy due to the compressibility
\mathcal{E}_d	Energy dissipated
\mathcal{E}_i	Internal energy
\mathcal{E}_k	Kinetic energy
\mathcal{E}_m	Mechanical energy
\mathcal{E}_p	Potential energy
$\mathcal{W}_{\Omega \rightarrow \partial\Omega}$	Work done by the fluid on the boundary
\mathbf{g}	Generic external volumetric force -Usually the gravity-
h	Characteristic length of the kernel
m	Mass
\mathbf{n}	Normal of a generic surface. Outward pointing normals are considered unless something else is stated
Ω	Compact support of the kernel, unless something else is stated
p	Pressure
\mathbf{r}	Particle position
ρ	Density
\mathbb{T}	Stress tensor
\mathbf{u}	Flow velocity
μ	Dynamic viscosity of the fluid
ν	Kinematic viscosity of the fluid, $\nu = \mu/\rho$

Operators

$\langle \dots \rangle$ SPH operator

Acronyms / Abbreviations

APU Accelerated Processing Unit

ART	Anti-Rolling Tank
BC	Boundary Condition
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
CSS	Conventional Serial Staggered
δ -SPH	WC-SPH with a diffusive term added to the mass conservation equation
EOS	Equation of State
GPGPU	General-Purpose computing on Graphics Processing Unit
GPU	Graphics Processing Unit
I-SPH	Truly Incompressible SPH
MPS	Moving Particle Semi-implicit Method
RAO	Response Amplitude Operator
R-SPH	Riemann solvers based SPH
SPH	Smoothed-particle hydrodynamics
TLD	Tuned Liquid Damper
WC-SPH	Weakly Compressible SPH

Abstract

The subject of the present thesis is the development of a new free Smoothed Particle Hydrodynamics (SPH) code oriented to the researchers, conversely to the already available free/open source SPH codes. The new software should be competitive in performance terms, allowing at the same time to be modified and extended with a minimum amount of effort. Such code should be later applied to analyze SPH models where a diffusive term is added in the mass conserving equation, which are usually known as δ -SPH models.

Therefore, the thesis can be split in 3 main parts: The theoretical aspects of the SPH models considered, the numerical model implementation, and their verification and validation.

During the theoretical analysis, the possibility of adding a diffusive term inside the continuity equation in order to significantly reduce the characteristic numerical instabilities of the SPH model is investigated, compiling a list of the already existing models, discussing their features and relationships, and pointing out their main benefits and drawbacks. A close relation between the δ -SPH models and the Riemann solver based ones have been unveiled. Also, a new diffusive term not requiring neither a tuning parameter nor a Courant condition is presented. Such term is resulting from a slightly modification of an already existing one.

Later, as part of the same theoretical investigation of the model, an energy conservation analysis has been carried out, considering the interactions with the boundary conditions, such that several extra energy terms are resulting, some of them representing extra energy dissipations if a set of conditions are fulfilled.

All this theoretical aspects have been implemented in a new free SPH solver, accelerated with OpenCL, modular, and extensible by Python scripts. Implementation details regarding the software itself, as well as the SPH models considered have been provided. The main benefits of the new code, compared with other already existing free/open source alternatives, have been discussed.

Finally, a set of 4 verification and validation test cases is described, covering all the theoretical and implementation aspects discussed along the thesis. In such verification and validation cases the implementation details, performance, and results quality are discussed. The verification and validation cases are complemented with two practical applications,

where the capabilities of the new software to scale to significantly complex problems are shown.

Resumen

En esta tesis se documenta el desarrollo de un nuevo software de SPH (Smoothed Particle Hydrodynamics), que tiene la particularidad de estar orientado al trabajo de investigación, en lugar de enfocarse a aplicaciones industriales como los otros códigos libres/abiertos disponibles en la actualidad. El nuevo código debe ser competitivo en términos de coste computacional, al tiempo que provee una herramienta modular y extensible. Dicho código se empleará más tarde para analizar los modelos conocidos como δ -SPH, que se caracterizan por la presencia de un término difusivo en la ecuación de conservación de masa.

Por tanto, el trabajo llevado a cabo durante el desarrollo de la tesis se puede dividir en tres partes: Los aspectos teóricos, la implementación del modelo numérico, y la verificación y validación de los mismos.

A lo largo del análisis teórico se investigará la posibilidad de añadir un término difusivo a la ecuación de continuidad, con el objetivo de reducir significativamente las inestabilidades numéricas que vienen caracterizando al modelo de SPH. Para ello se recopilarán todos los modelos existentes actualmente, discutiendo sus propiedades principales, las relaciones que guardan entre ellos, y las principales ventajas a destacar de cada uno de ellos. Ello nos servirá para establecer una relación entre los modelos δ -SPH, y los modelos basados en *Riemann solvers*. Además se presentará un nuevo término, resultante de la modificación de uno ya existente. Dicho término tendrá la ventaja de no requerir ni un parámetro de tuneado, ni una condición de Courant adicional sobre el paso de tiempo.

Como parte del mismo análisis teórico, se investigarán las propiedades de conservación de los modelos, en donde se considerará la interacción con los contornos. Como resultado, se demostrará la existencia de unos términos de energía adicionales, algunos de los cuales pueden ser considerados disipaciones *extra* si se cumplen unas determinadas condiciones.

Todos estos aspectos teóricos se implementan en un nuevo código libre de SPH, el cual está acelerado con OpenCL, además de ser modular y extensible con códigos de Python. Se documentan algunos detalles sobre la implementación del software en si mismo, y sobre la implementación de los modelos SPH a considerar. El nuevo software se compara con otras alternativas ya existentes, resaltando sus ventajas.

Finalmente, se describen 4 casos de verificación y validación que permiten cubrir todos los aspectos teóricos y computacionales abordados durante la tesis. En ellos se proveen algunos detalles sobre como se han desarrollado, el coste computacional, y la calidad de los resultados. Estos casos de verificación y validación se complementan con dos aplicaciones prácticas, donde se muestran las capacidades del software para afrontar casos significativamente complejos.

Chapter 1

Introduction

1.1 Motivation and background

1.1.1 General

When in a CFD discussion someone mentions the SPH numerical method, the seeds of controversy are sown. Probably, at some point of the conversation someone will try to support SPH arguing that it is a young methodology, which is still waiting to enjoy a large amount of improvements. However, it does not seem sensible to claim that SPH is a young technique since it was introduced for the first time in 1977 (Gingold and Monaghan, 1977; Lucy, 1977).

In order to find a justification to the lack of maturity of the method, the number of publications per year related with SPH are depicted in Fig. 1.1. It can be appreciated that during the first ten years the methodology received a marginal attention by the research community. Moreover, the interest in SPH has not started to significantly grow until the 21st century.

Therefore, even though considering SPH as a young methodology could be strictly incorrect, we can assert that the rate of publications is currently still significantly growing. To illustrate such statement, it can be remarked that just in the last year more papers have been published than during the first 23 years of the entire history of the method.

The reasons that have led the research community to start paying attention to SPH are really foggy, but probably they are widely related to the GPGPU technique introduction, which has critically modified the computer simulations paradigm. Next section is dedicated to discuss this disruptive technique.

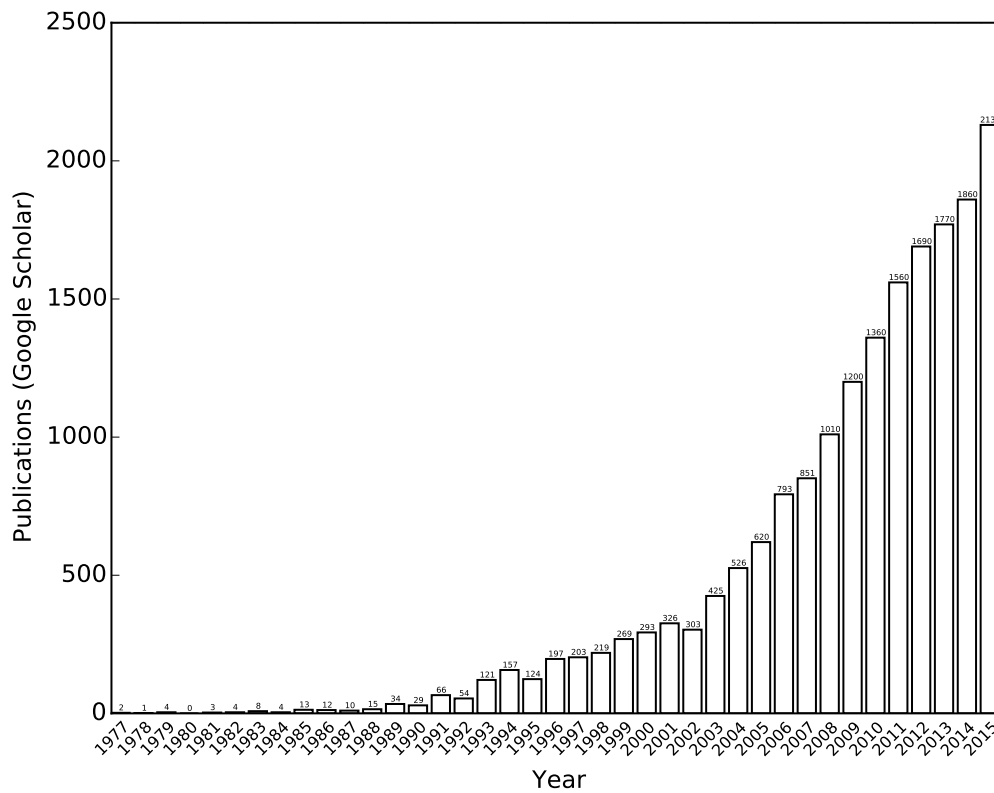


Fig. 1.1 Number of SPH related publications per year. Extracted from scholar.google.com (Thomson Reuters Science citation Index indexed and non-indexed publications are considered)

1.1.2 GPGPU, a new computing paradigm

The GPGPU term was coined for the first time by Harris (2002). By then, such technique was based on the use of shaders -small pieces of code to be executed in the GPU- that operate data artificially introduced as textures.

Such new paradigm was motivated by the technical limitation reached in the CPU development a few years ago, due to the impossibility to increase the processor's clock-rate. The manufacturers dealt with this situation focusing their new designs on the multi-core chips with a clock-rate stabilized around 3 GHz. In order to use this new feature, specific new libraries were mandatory. Nowadays, the Open Multi-Processing standard (OpenMP) (OpenMP Architecture Review Board, 2013) is usually applied in all operating systems and architectures.

Of course, the extremely low-level approach for developing GPGPU accelerated applications was quickly superseded by sets of libraries and compilers specifically designed

for that purpose, e.g. Brook (Stanford University graphics group, 2004), CUDA (NVidia, 2006) or AMD-Stream (AMD, 2006). At that time (circa 2006), a GPU device had similar computational power than a middle sized CPU based cluster for a fraction of its cost, a feature that quickly attracted the attention of science and engineering researchers (Maintz et al., 2011; Martinsen et al., 2009; Molero-Armenta et al., 2014).

SPH community has not remained unaware of this computer sciences revolution. In fact, SPH may result in relatively large computational costs due to the large number of interactions per particle and the small time steps usually required, which made SPH simulations become a constant struggle, and therefore heavily criticized. The first proofs of concept regarding the acceleration of a SPH code with a GPU device, in the context of the computer graphics, were described by Amada et al. (2004) and Kipfer et al. (2004), deploying moderate speed-ups. They were followed by several implementations which proved the capabilities of graphic oriented devices to perform massive computations using the CUDA language (Crespo et al., 2011; Dominguez et al., 2013; Herault et al., 2010), deploying this time large speed-ups which transformed SPH into a competitive CFD alternative in performance terms, in a variety of problems.

However, in recent years the gap between the CPU and the GPU performance is fading. Moreover, the hybridization of both technologies -so-called APU- is becoming a very promising tendency, which significantly increases the computational performance due to the extreme reduction of the overhead due to the memory transfers. Unfortunately, the available CUDA based implementations present some limitations which put them aside of these new technologies:

1. CUDA is a proprietary framework.
2. Heterogeneous platforms are not supported in CUDA.
3. Hardware from vendors other than NVidia cannot be used within the CUDA framework yet.

To partially work around this problem, DualSPHysics package (Crespo et al., 2011; Dominguez et al., 2013) includes both a CUDA and an OpenMP implementations, with the main drawback of the large amount of code duplication. In this sense, the open standard OpenCL (Khronos group, 2009) can be used to completely shortcut the problem, due to its specific design to unify the acceleration of codes when heterogeneous platforms are considered.

To illustrate the relevance of the APU irruption in the numerical simulations, Daga et al. (2011) compared the performance of the AMD Zacate APU with two different AMD graphic cards, benchmarking against 4 classic cases: the Fast Fourier Transform, the Scan,

the Reduction, and the Leonard-Jones Molecular Dynamics. Even though the theoretical computational capabilities of the considered GPUs were several times larger than the APU's, the latter showed similar performance in those benchmarks characterized by their high computational demands, becoming significantly faster when the bandwidth was a critical factor. Along the same line, Conti et al. (2012) compared the performance deployed by a high-end NVidia Tesla GPU and a mid-end desktop APU, for a complex practical application, demonstrating that similar performances are achieved.

1.1.3 State of the art of SPH

Regardless the reasons that motivated the critical change of tendency shown in Fig. 1.1, it is interesting to analyze the consequences. There is no doubt that the constantly increasing research efforts dedicated to the SPH method in recent years are helping the community to get a much better understanding of such method, and therefore its strengths and weaknesses, implying at the same time an unpredicted rate of improvements. Therefore, it is not unreasonable to suggest that SPH itself, and all the related ecosystem in general, is currently in a really healthy situation. Unfortunately, such situation is inexorably resulting in a large rate of new formulations and methodologies. To illustrate that, the main contributions of recent years are reviewed below.

To start with, the incompressibility of the model, and the associated numerical noise, is a SPH aspect which is receiving a large amount of interest by the community. One significant contribution in this area was carried out by Vila (1999), who formalized the R-SPH methodology (see also Rafiee et al. (2012) and Koukouvinis et al. (2013)). Such formulation is usually criticized by the large computational costs resulting from the integrated Riemann solver. Ferrari et al. (2009) simplified it, significantly reducing the required computational efforts, introducing a Rusanov (upwind flux-based) diffusive term directly in the conservation of mass equation of the WC-SPH formulation. In parallel, Molteni and Colagrossi (2009) proposed a different diffusive term to be added to the mass conservation equation as well. Both terms suffer from inconsistencies close to the free surface, as it was demonstrated by Antuono et al. (2010), who also suggested a correction to recover the consistency.

The term proposed by Molteni and Colagrossi (2009), and corrected by Antuono et al. (2010), has been later deeply investigated (Antuono et al., 2012, 2011, 2015). Recently, as part of this thesis, Cercos-Pita et al. (2016b) have revisited all those terms, as well as the one proposed by Fatehi and Manzari (2011), investigating their features. In the same work the relation between R-SPH and WC-SPH methodologies, already hinted by Ferrari et al. (2009), was documented.

In a different fashion, some other authors have investigated the use of truly incompressible approaches in SPH (Lee et al., 2008; Shao and Lo, 2003), with the main drawback that a large and dense linear system of equations has to be solved each time step, resulting again in large computational costs. Related to that, the equivalence of those approaches with the MPS method (Khayyer and Gotoh, 2009) was demonstrated by Souto-Iglesias et al. (2013) and Souto-Iglesias et al. (2014).

Another topic which is acquiring a significant amount of prominence in recent years is the boundary conditions treatment, becoming in fact one of the four grand challenges of SPH, according to the SPHERIC SPH Numerical Development Working Group.

Colagrossi et al. (2010) profusely studied the behavior of the operators close to the free surface, including to this end the wide variety of different formulations used to approximate each differential operator involved in the Navier-Stokes equations, providing a theoretic background for the inconsistencies suffered by the SPH model. A further analysis of the viscous term was carried out later by Colagrossi et al. (2011), demonstrating that it may locally diverge close to the free surface, becoming anyway consistent in an integral sense.

Regarding the solid boundaries, on top of the wide variety of techniques already existing (Cercos-Pita, 2015), which can be grouped as repulsive forces and fluid extensions, a new methodology was described by Ferrand et al. (2013), based on the works of Campbell (1989) and De Leffe et al. (2009), so-called Boundary Integrals. In the original work, a semi-analytical approach to approximate the boundary integrals involved was suggested, while in some other recent works a purely numerical approach has been practiced (Cercos-Pita, 2015; Macià et al., 2012).

Macià et al. (2011a), Macià et al. (2012) and Merino-Alonso et al. (2013) performed a set of analysis on the consistency of the differential operators close to the solid boundaries, when different techniques are applied to impose the boundary conditions.

Actually the SPHERIC SPH Numerical Development Working Group is doing a great job regarding the SPH grand challenges, not only motivating new research works, but also collecting and compiling the most significant related publications. Hence, the SPH grand challenges site is a good place to look for literature where new formulations, models and schemes are introduced. For instance, within the stability section topic the work carried out by Dehnen and Aly (2012) can be remarked. In such work the role of the kernel in the numerical scheme stability was discussed, demonstrating the benefits of switching to the Wendland kernel, instead of the Gaussian or Cubic splines typically applied at that time. This idea was already noticed by Macià et al. (2011b) through numerical simulations.

Regardless the theoretical aspects commented above, one SPH grand challenge is focused on the adaptivity. Adaptivity refers to the possibility of increasing the resolution, i.e. the

number of particles, in specific regions of the simulation domain where more detail is required. The term is inherited from the adaptive meshes already used for long time in Eulerian methods. Of course the adaptivity is a desirable feature to the model in order to significantly reduce the computational costs. The first dynamic particles splitting process, conserving both mass and momentum was described by Bonet and Rodriguez-Paz (2005) and Feldman and Bonet (2007). Later, Vacondio et al. (2013) proposed a model to split and coalesce particles conserving momentum, even when variable smoothing length is considered.

1.1.4 The SPH community

The state of the art of the SPH methodology described in the previous section has depicted a dynamic ecosystem, with a constant increase of new actors, formulations, models, schemes and implementations. However, such healthy situation may become untenable if a convenient set of resources can not be eventually provided to the community.

Limiting to available software implementations, it can be asserted that the industry currently enjoys a number of open-source implementations (Dominguez et al., 2013; Gomez-Gesteira et al., 2012; Herault et al., 2010), which have significantly contributed to boil down the technical barriers to access the model. However, there is an actual lack of tools strictly oriented to the researchers. It is clear that the features of the software mainly dedicated to industrial applications should significantly differ from the features of the tools oriented to research activities, in which some performance may be sacrificed for the sake of great capabilities to develop, test and deploy new formulations and schemes. In fact, the low-level programming language used by the fully accelerated SPH alternatives oriented to industrial applications (Dominguez et al., 2013; Herault et al., 2010), which are strictly written in C++ and CUDA, may excessively hamper the developing tasks, and therefore the implementation of modifications to the model. Therefore, the SPH community is facing the challenge of providing software packages specifically designed to aid the researchers.

1.2 Objectives

During this thesis the AQUAgpusph tool is presented, with the following main objectives:

1. Analyze the wide variety of boundary conditions already developed for the SPH method, carrying out a formalization which allows to provide a common framework to investigate all of them in the same context.
2. Break down the energy components of the discrete system, comparing it with the continuous one in order to analyze the effect of the selected viscous function, the

δ -SPH term, and the boundary conditions. Such analysis is focused on the investigation of the consistency of the different formulations, as well as the eventual extra dissipation or spurious energy terms.

3. Develop a novel free SPH solver, which can be executed in a wide variety of platforms and architectures. This implementation should be clearly oriented to the research activities, allowing the users to easily modify the formulations to be applied, or even to extend the solver using a Python interface.
4. Carry out verification and validation of the solver with simple test cases.
5. Explore the possibilities offered by this new code to solve complex problems, namely forced motion 3D sloshing in a nuclear reactor, and non-linear coupled sloshing and ship motions problems.

1.3 Structure of the thesis

This thesis is structured as follows.

1. The SPH numerical model is described in chapter 2, including all the formulations supported “out of the box” by the AQUAgnusph package.
2. In chapter 5, the key AQUAgnusph features are discussed, focusing on the differences with other existing SPH solvers.
3. In chapter 6, the new code, AQUAgnusph, is verified and validated against simple benchmark tests.
4. In chapter 7, a set of practical applications will be presented in order to demonstrate the capabilities of AQUAgnusph to be used both in industrial and research contexts.
5. Finally, a conclusions summary together with future work targets are provided.

Chapter 2

Numerical model

Even though AQUA_{gpusph} has been designed to be able to solve a wide variety of problems, just an incompressible Navier-Stokes equations solver is provided “out of the box”, based on the WC-SPH formulation. In this chapter, the governing equations and the associated numerical model built on top of that are presented.

2.1 Governing equations

In the WC-SPH formulation, the incompressibility is sought by modelling the flow with a compressible fluid which, in the expected flow regime, presents very small density fluctuations. The fluid is assumed to be barotropic, implying that the internal energy equation is decoupled from the continuity and momentum equations.

The compressible Navier-Stokes equations for a barotropic fluid in Lagrangian formalism are:

$$\frac{D\rho}{Dt} = -\rho \operatorname{div}(\mathbf{u}), \quad (2.1)$$

$$\frac{D\mathbf{u}}{Dt} = \mathbf{g} + \frac{\operatorname{div}(\mathbf{T})}{\rho}, \quad (2.2)$$

$$p = p(\rho). \quad (2.3)$$

The flow velocity, \mathbf{u} , is defined as the material derivative of a fluid particle position, \mathbf{r} :

$$\frac{D\mathbf{r}}{Dt} = \mathbf{u}. \quad (2.4)$$

The stress tensor of a Newtonian fluid, \mathbb{T} , is defined as follows:

$$\mathbb{T} = (-p + \lambda \text{tr} \mathbb{D}) \mathbb{1} + 2\mu \mathbb{D}, \quad (2.5)$$

with \mathbb{D} as the rate of strain tensor, i.e. $\mathbb{D} = (\nabla \mathbf{u} + \nabla \mathbf{u}^T)/2$; $\mathbb{1}$ the identity matrix, and μ, λ are the viscosity coefficients.

An EOS relating the pressure and density fields must be imposed. Monaghan (2012) discussed the most convenient EOS for weakly compressible simulations, suggesting the usage of the following stiff equation (see also (MacDonald, 1966)):

$$p = p_0 + \frac{c_s^2 \rho_0}{\gamma} \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right), \quad (2.6)$$

where ρ_0 is the reference density, p_0 is the ambient pressure, and c_s is the sound speed. However, assuming the weakly-compressibility hypothesis, a Taylor series around $\rho = \rho_0$ demonstrates that the following linear form is the main contribution in the previous equation (see for instance (Antuono et al., 2010)):

$$p = p_0 + c_s^2 (\rho - \rho_0). \quad (2.7)$$

Hereinafter, just the EOS (2.7) will be considered, even though more complicated expressions can be easily used in AQUAplusph.

2.1.1 Boundary conditions

A number of BCs can be considered depending on the particular problem to solve. In this section all the boundary conditions provided within the AQUAplusph package are described.

Free surface BC

Along the free surface, kinematic and dynamic BCs must be satisfied. The kinematic BC implies that material points already on the free surface must remain on $\partial\Omega_F$ since this region evolves with the fluid flow.

$$\mathbf{u}_n(\mathbf{x}) = \mathbf{V}_n(\mathbf{x}) \quad \forall \mathbf{x} \in \partial\Omega_F, \quad (2.8)$$

where \mathbf{u}_n is the fluid velocity projected over the solid normal at point \mathbf{x} , and \mathbf{V}_n is the solid velocity projected over the solid normal.

The dynamic free-surface BC is a consequence of the continuity of the stresses across the free surface. Assuming that surface tension is negligible, a free surface does not withstand neither perpendicular normal stresses nor parallel/tangential shear stresses. For a Newtonian fluid, the dynamic free-surface BC can be expressed as:

$$\mathbf{T} \cdot \mathbf{n} = (-p + \lambda \operatorname{tr} \mathbf{D}) \mathbf{n} + 2\mu \mathbf{D} \cdot \mathbf{n} = 0. \quad (2.9)$$

Solid boundary conditions

In the solid boundaries an impenetrability condition must be imposed, meaning that

$$\mathbf{u}_n(\mathbf{x}) = \mathbf{V}_n(\mathbf{x}) \quad \forall \mathbf{x} \in \partial\Omega_B. \quad (2.10)$$

Regarding the tangential velocity on the solid, either a free-slip or no-slip BC can be imposed, depending on the interest on resolving the boundary layers. The no-slip condition implies:

$$\mathbf{u}_t(\mathbf{x})|_{\text{no-slip}} = \mathbf{V}_t(\mathbf{x}) \quad \forall \mathbf{x} \in \partial\Omega_B, \quad (2.11)$$

where \mathbf{u}_t and \mathbf{V}_t are the fluid and solid velocities respectively, both of them projected on the tangent plane of the solid boundary. Conversely, with a free-slip BC no restriction is imposed to the tangential velocity.

Regarding the pressure, the following Neumann boundary condition is considered:

$$\nabla p(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = \rho \left[\mathbf{g} - \frac{d\mathbf{V}(\mathbf{x})}{dt} + \nu \Delta \mathbf{u}(\mathbf{x}) \right] \cdot \mathbf{n}(\mathbf{x}) \quad \forall \mathbf{x} \in \partial\Omega_B, \quad (2.12)$$

Inflow/Outflow

While in solid boundaries a Dirichlet BC is applied to the velocity field, and a Neumann BC to the pressure one, in the case of Inflow BC just Dirichlet boundary conditions are considered,

$$\begin{cases} \mathbf{u}(\mathbf{x}) = \mathbf{V}(\mathbf{x}) \\ p(\mathbf{x}) = P(\mathbf{x}) \end{cases} \quad \forall \mathbf{x} \in \partial\Omega_I, \quad (2.13)$$

where P is the prescribed pressure field at the inflow BC.

On the other hand, in the outflow just Neumann BCs are imposed:

$$\begin{cases} \nabla \mathbf{u}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = \mathbf{0} \\ \nabla p(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}) = \mathbf{g} \cdot \mathbf{n}(\mathbf{x}) \end{cases} \quad \forall \mathbf{x} \in \partial\Omega_O. \quad (2.14)$$

Periodic boundary conditions

Some problems may admit a spatial periodic description, such that a generic scalar or vector field can be defined as follows:

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x} + \Delta\mathbf{x}), \quad (2.15)$$

where $\Delta\mathbf{x}$ is the spatial period.

Symmetric boundary conditions

In a similar way, some problems may admit a symmetric description such that a generic scalar field can be defined as follows:

$$f(\mathbf{x}) = f(\mathbf{x} - 2\mathbf{n}_{\partial\Omega}((\mathbf{x} - \mathbf{x}_{\partial\Omega}) \cdot \mathbf{n}_{\partial\Omega})), \quad (2.16)$$

where $\mathbf{x}_{\partial\Omega}$ and $\mathbf{n}_{\partial\Omega}$ are a generic point and the normal of the symmetry plane respectively. Regarding the treatment of vector fields, the normal and tangential components should, if needed, be extended in a different way:

$$f_n(\mathbf{x}) = \mathbf{f}(\mathbf{x}) \cdot \mathbf{n}_{\partial\Omega} = -\mathbf{f}(\mathbf{x} - 2\mathbf{n}_{\partial\Omega}((\mathbf{x} - \mathbf{x}_{\partial\Omega}) \cdot \mathbf{n}_{\partial\Omega})) \cdot \mathbf{n}_{\partial\Omega}, \quad (2.17)$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{f}(\mathbf{x} - 2\mathbf{n}_{\partial\Omega}((\mathbf{x} - \mathbf{x}_{\partial\Omega}) \cdot \mathbf{n}_{\partial\Omega})) + 2f_n(\mathbf{x})\mathbf{n}_{\partial\Omega}, \quad (2.18)$$

such that the tangential component is preserved while the normal component orientation is swapped.

2.1.2 Initial conditions

Since the governing equations (2.1)-(2.3) must be solved as an initial value problem, the pressure, density and velocity fields must be known at an initial time t_0 , from which a forward time integration can be performed.

2.2 SPH methodology

2.2.1 SPH Continuous model

SPH method is entirely based on a convolution integral. To this end a kernel $W_h(\mathbf{x})$ is defined as a positive even function such that

$$\int_{\mathbb{R}^d} W_h(\mathbf{x} - \mathbf{y}) d\mathbf{y} = 1, \quad (2.19)$$

$$W_h(\infty) = 0. \quad (2.20)$$

For practical purposes, the kernel $W_h(\mathbf{x})$ vanishes for $|\mathbf{x}| > sh$, where s is an integer greater than 0, and h is the characteristic length of the kernel. Hence, the SPH convolution integral of a scalar or vector function $f(\mathbf{x})$ with respect to the kernel $W_h(\mathbf{x})$ is defined as

$$\langle f(\mathbf{x}) \rangle = \int_{\Omega} f(\mathbf{y}) W_h(\mathbf{x} - \mathbf{y}) d\mathbf{y}, \quad (2.21)$$

where the integral is restricted to the compact support of the kernel, Ω , of radius sh . It should be noticed that we are not considering truncated compact supports yet, where boundary conditions should be imposed. It can be demonstrated that $\langle f(\mathbf{x}) \rangle - f(\mathbf{x}) = O(h^2)$ (see Monaghan (2012)).

Of course the expression (2.21) can be applied to a generic first order differential operator as well:

$$\langle \mathcal{D}f(\mathbf{x}) \rangle = \int_{\Omega} \mathcal{D}f(\mathbf{y}) W_h(\mathbf{x} - \mathbf{y}) d\mathbf{y}. \quad (2.22)$$

Unfortunately, $\mathcal{D}f(\mathbf{y})$ is generally unknown, turning the expression above useless in principle. However, it is possible to expand it, and use the divergence theorem to get a new form where the value of $\mathcal{D}f(\mathbf{y})$ is not required anymore,

$$\langle \mathcal{D}f(\mathbf{x}) \rangle = \int_{\Omega} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} + \int_{\partial\Omega} f(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y}, \quad (2.23)$$

where the symmetry property of the kernel has been already applied. However, provided that we are not considering truncated kernel supports yet, and since the kernel vanishes for a radius bigger or equal to the compact support one by definition, the second term of the right hand side can be neglected, leading to the well known SPH convolution continuous expression:

$$\langle \mathcal{D}f(\mathbf{x}) \rangle = \int_{\Omega} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y}. \quad (2.24)$$

Therefore the SPH convolution integral has allowed us to conveniently move the differential operator to the kernel function, which can be analytically differentiated.

2.2.2 SPH Discretized model

For practical purposes, the convolution integral of the equation (2.24) is solved numerically, introducing therefore another error (Amicarelli et al., 2011; Quinlan et al., 2006). To this end, the space is conveniently discretized by placing a set of particles separated a distance Δr which are treated from a Lagrangian point of view. The differential volume element, $d\mathbf{y}$, is then discretized as the volume of the particles, and as follows for a generic particle i :

$$d\mathbf{y} \approx \frac{m_i}{\rho_i}, \quad (2.25)$$

where m_i is the particle mass, and ρ_i its density. The discretized version of the SPH convolution (2.21), and its differential developed form (2.24), read as follows:

$$\langle f \rangle_i = \sum_{j \in \text{Fluid}} \frac{f_j}{\rho_j} W_h(\mathbf{r}_j - \mathbf{r}_i) m_j, \quad (2.26)$$

$$\langle \mathcal{D}f \rangle_i = \sum_{j \in \text{Fluid}} \frac{f_j}{\rho_j} \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) m_j, \quad (2.27)$$

where it can be appreciated that the expression (2.27) offers us a tool to compute the value of first order differential operators of a generic scalar or vector field from the already known value of such generic field for the surrounding particles (hereinafter neighbours). The convolution process described above is schematically shown in the Fig. 2.1.

Even though the same procedure can be extended to higher order differential operators, the error resulting from that may become unacceptable. As a workaround, a combination of a finite difference scheme and the SPH model is usually applied. At the moment there exists two different formulations:

1. Monaghan and Gingold (1983): Specifically designed to model an artificial viscosity term, and shown by Hu and Adams (2006) to be the continuous version of the incompressible flow Newtonian viscous term.
2. Morris et al. (1997): A more general Laplacian operator. Unfortunately, this model diverges close to the boundaries if a BC is not properly imposed (see Colagrossi et al. (2011)).

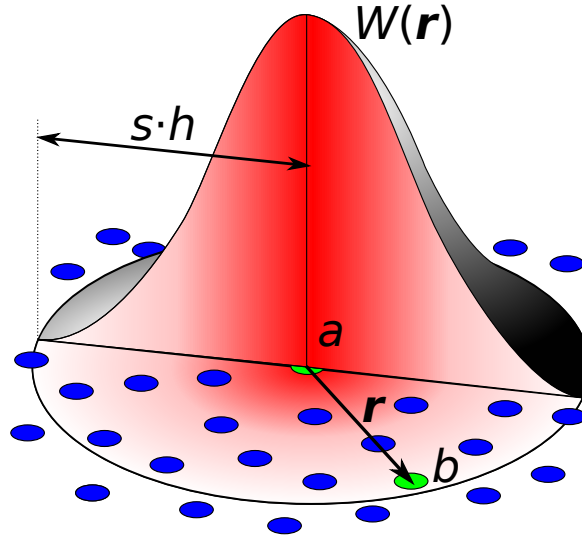


Fig. 2.1 Discrete SPH convolution scheme

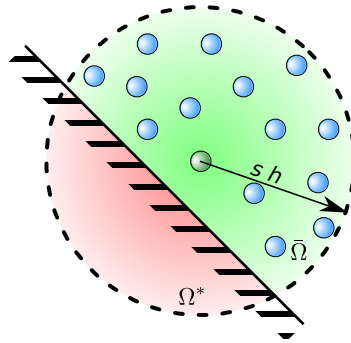


Fig. 2.2 Schematically view of a generic particle affected by a boundary. The compact support has been divided in 2 subdomains: $\bar{\Omega}$, highlighted with a green background, where fluid information is already available, and Ω^* , highlighted with a red background, where fluid data is in principle unknown

2.2.3 Boundary conditions

Both in sections 2.2.1 and 2.2.2 we have considered that the compact support is completely filled. However in a number of problems it is required to deal with boundary conditions where the compact support will be inexorably truncated. Such situation has been schematically depicted in Fig 2.2. As it can be appreciated, the kernel compact support, Ω , lacks of fluid particles in the subdomain Ω^* .

There currently exist 4 methodologies that have been developed to deal with this situation, i.e. to impose boundary conditions. They are described in the following subsections.

Kernel corrections

The obvious solution is letting the kernel compact support become truncated, eventually replacing the kernel by a deformed version of itself, which is in fact able to partially recover the differential operators consistency. Indeed, when this approach is practised the kernel function, $W_h(\mathbf{r}_j - \mathbf{r}_i)$, is replaced by an alternative function, $W_h^{\mathcal{L}}(\mathbf{r}_j - \mathbf{r}_i)$, with the following form:

$$\nabla W_h^{\mathcal{L}}(\mathbf{r}_j - \mathbf{r}_i) = \mathcal{L}(\mathbf{r}_i) \nabla W_h(\mathbf{r}_j - \mathbf{r}_i). \quad (2.28)$$

Of course, In case $\mathcal{L}(\mathbf{r}_i) = \mathcal{I}$ the original undeformed kernel is recovered.

Along the line of the kernel deformations, the simplest formulation is the Shepard renormalization (Belytschko et al., 1998),

$$\mathcal{L}(\mathbf{r}_i) = \frac{1}{\gamma(\mathbf{r}_i)} \mathcal{I}, \quad (2.29)$$

where $\gamma(\mathbf{r}_i)$ is the Shepard renormalization factor, defined as follows:

$$\gamma(\mathbf{r}_i) = \sum_{j \in \text{Fluid}} W_h(\mathbf{r}_j - \mathbf{r}_i) \frac{m_j}{\rho_j}. \quad (2.30)$$

It should be noticed that the Shepard renormalization can be used to recover the 0^{th} order consistency in the fields values convolution, remaining anyway inconsistent for the differential operators computation. In order to recover 1^{st} order consistency, even for the differential operators, Randles and Libersky (1996) described the MLS approach, where the deformation matrix is defined as follows:

$$\mathcal{L}(\mathbf{r}_i) = \left[\sum_{j \in \text{Fluid}} (\mathbf{r}_j - \mathbf{r}_i) \otimes \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \frac{m_j}{\rho_j} \right]^{-1}. \quad (2.31)$$

This kind of boundary condition has been widely applied in the literature to impose the free surface BC, when multiphase simulations are not considered. Moreover, Colagrossi et al. (2009) already analyzed the consistency and conservation features shown by multiple formulations, eventually including some kernel deformations.

Boundary forces

The previous technique may not be applicable in some contexts. For instance, even if MLS deformed kernels (Randles and Libersky, 1996) are applied, the solid boundaries untraversing feature cannot be asserted. As a simple solution a set of dummy particles

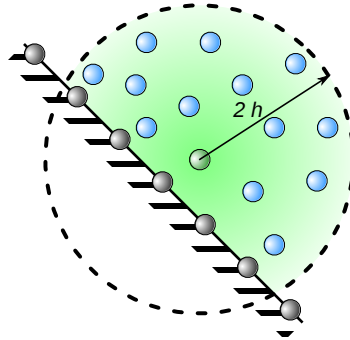


Fig. 2.3 Schematically view of the boundary forces methodology

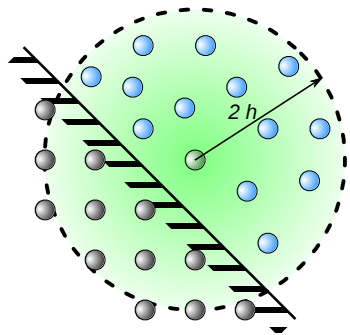


Fig. 2.4 Schematically view of the fluid extensions methodology

can be inserted along the boundary, such that they will exert a repulsion force over the fluid particle. It has been schematically depicted in Fig. 2.3.

The main benefit of this technique is that it can be easily implemented, which has brought the spawn of a number of formulations in the past (see for instance the boundary forces (Cleary, 1997, 1998; Monaghan and Kajtar, 2009), the Dynamic boundaries (Crespo et al., 2007), or the elastic bounce (Cercos-Pita, 2015)). However, this kind of boundary conditions are generally damaged by the lack of consistency.

Fluid extensions

A completely different approach to impose the boundary condition is based on completing again the kernel compact support, virtually extending the fluid domain to this end. The process can be appreciated in Fig. 2.4. Of course, these new particles, usually called virtual

or ghost particles, should be added to the SPH operators described in the section 2.2.2:

$$\langle f \rangle_i = \sum_{j \in \text{Fluid}} \frac{f_j}{\rho_j} W_h(\mathbf{r}_j - \mathbf{r}_i) m_j + \sum_{j \in \text{GP}} \frac{f_j}{\rho_j} W_h(\mathbf{r}_j - \mathbf{r}_i) m_j, \quad (2.32)$$

$$\langle \mathcal{D}f \rangle_i = \sum_{j \in \text{Fluid}} \frac{f_j}{\rho_j} \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) m_j + \sum_{j \in \text{GP}} \frac{f_j}{\rho_j} \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) m_j, \quad (2.33)$$

where GP denotes the appended Ghost particles.

The first non-trivial topic to become solved, in order to apply this type of this boundary condition, is the process to select the number and position of the new ghost particles. Three popular approaches have been documented in the literature:

1. Fixed particles attached to the boundary (Bouscasse et al., 2013).
2. Mirrored particles (Randles and Libersky, 1996).
3. Templates of particles (Fourtakas et al., 2014).

Once the positions of the ghost particles are set, the second challenge is defining a procedure to assign field values to those new particles, using for that the already available information from the fluid particles. The applicable field values extension algorithms, and their consistency, has been reviewed by Macià et al. (2011a) and Merino-Alonso et al. (2013).

Boundary Integrals

The fluid extensions approach to impose boundary conditions has the main drawback that dealing with complex geometries can become excessively complex. Even though the first try to use the normal flux to impose boundary conditions was described by Campbell (1989) for the first time, revisited later by Kulasegaram et al. (2004) and De Lefte et al. (2009), the first consistent Boundary Integrals formulation was introduced by Ferrand et al. (2013). In fact, the consistency of the Boundary Integrals methodology was analyzed by Macià et al. (2012).

Ferrand et al. (2013) initially proposed a semi-analytic formulation. However, such formulation depends on the usage of the mesh connectivity data, implying a prohibitive overhead in 3D GPU-based simulations. Therefore, during this work just the purely numerical Boundary Integrals scheme will be considered (see for instance (Cercos-Pita, 2015)).

It is convenient for us to develop the Boundary Integrals formulation in a different fashion to the usually found in literature. Coming back to Fig. 2.2, the first order differential operator continuous convolution of the equation (2.24) can be rewritten as follows:

$$\langle \mathcal{D}f(\mathbf{x}) \rangle = \int_{\bar{\Omega}} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} + \int_{\Omega^*} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y}, \quad (2.34)$$

where the second integral is in principle unknown. However, for a regular enough generic field f , and using the kernel properties described in equation (2.19), the following relation can be imposed between both integrals of the right hand side:

$$\begin{aligned} \frac{1}{\gamma(\mathbf{x})} \left(\int_{\bar{\Omega}} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} + \int_{\partial\bar{\Omega}} f(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} \right) = \\ \frac{1}{1 - \gamma(\mathbf{x})} \left(\int_{\Omega^*} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} + \int_{\partial\Omega^*} f(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} \right) + O(h), \end{aligned} \quad (2.35)$$

where $\mathbf{n}(\mathbf{y})$ is the outward normal (with respect to the integration domain boundary), and $\gamma(\mathbf{x})$ is the continuous Shepard renormalization factor,

$$\gamma_h(\mathbf{x}) := \int_{\Omega} W_h(\mathbf{x} - \mathbf{y}) d\mathbf{y}. \quad (2.36)$$

Taking into account that the kernel vanishes in all the boundary except in the intersection of the 2 subdomains, $\partial\bar{\Omega} \cap \partial\Omega^*$, the second integral of the left hand side is the same than the second integral of the right hand side, with the sign inverted. Hence, the second integral of the right hand side of the equation (2.34) can be expanded,

$$\begin{aligned} \int_{\Omega^*} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} \simeq \\ \frac{1}{\gamma(\mathbf{x})} \left((1 - \gamma(\mathbf{x})) \int_{\bar{\Omega}} f(\mathbf{y}) \cdot \nabla W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} + \int_{\partial\bar{\Omega}} f(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y}) W_h(\mathbf{y} - \mathbf{x}) d\mathbf{y} \right). \end{aligned} \quad (2.37)$$

The equation above is really interesting, because it clearly shows that the boundary effect, when the Boundary Integrals approach is applied, is composed by a term strictly depending on the boundary (surface integral of the right hand side), and a volume integral in the fluid domain. Such assertion contrasts with the common point of view where the effect of the boundary is restricted to the integral along the boundary.

Inserting equation (2.37) into equation (2.34), and discretizing, the well known purely numerical Boundary Integrals expression is achieved,

$$\langle \mathcal{D}f \rangle_i = \frac{1}{\gamma_i} \left(\sum_{j \in \text{Fluid}} \frac{f_j}{\rho_j} \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) m_j + \sum_{j \in \text{BE}} f_j \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) s_j \right), \quad (2.38)$$

where BE denotes a set of boundary elements distributed along $\partial\bar{\Omega}$, and \mathbf{n}_j and s_j their normal and area respectively.

Therefore, the BC described above is actually a combination of a boundary forces based BC (see 2.2.3), as it can be appreciated by the second summation of the right hand side, and

a Shepard kernel deformation (see 2.2.3). Indeed, it shares a lot of benefits of the boundary forces and kernel deformations, like the good capabilities to deal with complex geometries. Unfortunately, the main drawback imported from the kernel correction is the non-symmetric final kernel expression (in general $\frac{W_h(\mathbf{r}_j - \mathbf{r}_i)}{\gamma_i} \neq \frac{W_h(\mathbf{r}_i - \mathbf{r}_j)}{\gamma_j}$), which is affecting negatively the conservation properties of the model.

2.3 Numerical scheme

In principle, the SPH model described in the section 2.2 can be directly applied to the differential operators of the governing equations, discussed along the section 2.1. However, it is more convenient to consider the following numerical scheme:

$$\left\{ \begin{array}{l} \frac{d\rho_i}{dt} = -\rho_i \sum_{j \in \text{Fluid}} [(\mathbf{u}_j - \mathbf{u}_i) + \mathbf{F}_{ij}] \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \frac{m_j}{\rho_j}, \\ \frac{d\mathbf{u}_i}{dt} = - \sum_{j \in \text{Fluid}} \frac{p_i + p_j}{\rho_i \rho_j} \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) m_j \\ \quad - \mu K \sum_{j \in \text{Fluid}} \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{\rho_i \rho_j |\mathbf{r}_j - \mathbf{r}_i|^2} \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) m_j + \mathbf{g} \\ \frac{d\mathbf{r}_i}{dt} = \mathbf{u}_i \\ p_i = c_s^2 (\rho_i - \rho_0) \end{array} \right. \quad (2.39)$$

where \mathbf{F}_{ij} is a numerical diffusive term usually known as δ -SPH model (see Antuono et al. (2012, 2010, 2015); Cercos-Pita et al. (2016b)), which is discussed in the next chapter, and K is a spatial dimension parameter ($K = 6, 8, 15$, in 1D, 2D and 3D respectively). Of course the previous numerical scheme should be conveniently modified depending on the BCs techniques used, according to the models discussed in the section 2.2.3.

When neither the δ -SPH term, \mathbf{F}_{ij} , nor the viscous term, nor the BCs are considered, the numerical scheme (2.39) has the main benefit that is fully conservative. Therefore, the role of those terms in the energy balance of the numerical scheme should be carefully analyzed later, in chapter 4.

Chapter 3

Diffusive terms in the mass conservation equation

3.1 General

As it has been discussed in the section 2.3, a generic diffusive term, F_{ij} , has been introduced in the mass conservation equation. Such diffusive term has the main objective of reducing the characteristic SPH pressure field noise. Here in, several already existing terms are revisited, analyzing their relationships and principal features, providing a glossary where the main benefits and drawbacks of each one.

More details were published as part of this thesis in the work of Cercos-Pita et al. (2016b).

3.2 Consistency conditions and desirable features

In order to can discriminate the benefits and drawbacks of each diffusive model, a set of conditions to check the consistency, as well as some other desirable properties should be provided. More specifically, we are introducing 2 consistency conditions, and 2 desirable features. It should be remarked that in the work of Cercos-Pita et al. (2016b) an additional condition regarding the energy balance is stated, which for the sake of a more readable document has been intentionally relegated to the section 4.2.

3.2.1 Condition 1. Conservation of mass consistency

In order to assert the consistency of the model, the original governing equation (2.1) should be recovered in the continuum. This condition can be written as

$$\lim_{\substack{\Delta x/h \rightarrow 0 \\ h \rightarrow 0}} \left\langle \frac{d\rho_{F_{ij}}}{dt} \right\rangle_a = 0, \quad (3.1)$$

where Δx is the distance between particles. As it will be shown later, almost authors fulfill the condition (3.1) defining terms of the following form:

$$F(x, y) = h^p \hat{F}(x, y) \leftrightarrow p \geq 1. \quad (3.2)$$

3.2.2 Condition 2. Consistency close to the free surface

Antuono et al. (2010) introduced this condition for the first time. In such work the authors demonstrated that the term originally proposed by Molteni and Colagrossi (2009) becomes inconsistent close to the free-surface due to the singularity of the Morris formula (Colagrossi et al., 2009; Morris et al., 1997).

In a similar way, any other term based on the Laplacian of the density or pressure fields may be affected by such singularity, as Antuono et al. (2012) demonstrated for the term proposed by Ferrari et al. (2009).

3.2.3 Condition 3. Intrinsically global mass conservation

Even though the condition 3.2.1 is enough to assert the mass conservation in the continuum, for practical purposes intrinsically conserving the global mass, even at a discrete level, becomes a desirable feature of the diffusive model, i.e.:

$$\sum_{i \in \text{Fluid}} \left\langle \frac{d\rho_{F_{ij}}}{dt} \right\rangle_i = 0. \quad (3.3)$$

Antuono et al. (2012) already demonstrated that symmetric forms of the term F_{ij} are already fulfilling this condition.

3.2.4 Condition 4. No tuning of parameters

Antuono et al. (2010) and Antuono et al. (2012) discuss the process to set a convenient δ parameter to their model, and an associated Courant condition for the time step. The stability

analysis performed requires an expertise regarding the model to can successfully apply it in the simulations. Hence it is a desirable -but not mandatory- property of a model if it could be used without needing to tune parameters in order to have a stable time integration. The author incidentally remarks that the commonly used artificial viscosity depends on a parameter α to be tuned.

3.3 Existing diffusive terms in the literature

3.3.1 The diffusive term proposed by Vila (1999)

Vila (1999) included the Riemann solvers in the SPH context, where the Godunov scheme (Godunov, 1959) is approximately solved (see (Koukouvinis et al., 2013; Toro, 1997)). However, this model can be studied in the context of the δ -SPH model, provided that the mass variation rate used in R-SPH is related with the numerical diffusive term.

In fact, in R-SPH model the domain is discretized in particles as well, but the following transport equation is used:

$$\frac{dV_i}{dt} = V_i \operatorname{div}(\mathbf{u}(\mathbf{x}_i, t)), \quad (3.4)$$

where the volume of the generic particle i can be written as the division of mass and density,

$$V_i = \frac{m_i}{\rho_i}. \quad (3.5)$$

On the other hand, in R-SPH a mass rate of change equation is proposed in the following general form,

$$\frac{d}{dt}(V_i \rho_i) = - \sum_j V_i V_j \mathbf{G}_j \nabla W_{ij}. \quad (3.6)$$

We are trying to relate this proposed model with the discussed along the section 2.3. First let's expand the derivative of the volume, V , in terms of the mass and the density:

$$\frac{dV_i}{dt} = \frac{1}{\rho_i} \frac{dm_i}{dt} - \frac{m_i}{\rho_i^2} \frac{d\rho_i}{dt}. \quad (3.7)$$

Solving for the density, the conservation of mass equation in the context of the R-SPH model reads

$$\frac{d\rho_i}{dt} = -\rho_i \operatorname{div}(\mathbf{u}(\mathbf{x}_i, t)) + \frac{\rho_i}{m_i} \frac{dm_i}{dt}, \quad (3.8)$$

where the transport equation (3.4) has already been applied. Comparing this conservation of mass equation with the mass conservation equation in (2.39), we can relate the diffusive term

inside the conservation of mass equation with the mass variation rate in the R-SPH approach:

$$\langle \text{div}(\mathbf{F}_{ij}) \rangle_i = \frac{\rho_i}{m_i} \frac{dm_i}{dt}. \quad (3.9)$$

Regarding the term \mathbf{G}_j , it can be related with the mass variation rate as well, because from the equation (3.6) we can write

$$\frac{\rho_i}{m_i} \frac{dm_i}{dt} = - \sum_j \frac{1}{\rho_j} \mathbf{G}_j \nabla W_{ij} m_j. \quad (3.10)$$

Combining the equations (3.9) and (3.10) the relation between R-SPH model and the numerical diffusive term added to the conservation of mass equation can be obtained:

$$\mathbf{F}_{ij} = -\mathbf{G}_j. \quad (3.11)$$

Indeed the term proposed for the mass rate of change equation in R-SPH is the same than the introduced in the conservation of mass equation. This idea had been hinted by Ferrari et al. (2009), who did not formalized it.

Hence, the δ -SPH equivalent term can be written as follows:

$$\mathbf{F}_{ij}^{\text{Vi}} = -2\rho_{i,ij} (\mathbf{u}_{i,ij} - \mathbf{u}(\bar{\mathbf{r}}_{ij}, t)), \quad (3.12)$$

where $\rho_{i,ij}$ and $\mathbf{u}_{i,ij}$ are the approximate solutions of a Riemann problem for the density and velocity respectively, and

$$\bar{\mathbf{r}}_{ij} = \frac{\mathbf{r}_i + \mathbf{r}_j}{2}. \quad (3.13)$$

In this term the condition 1 is fulfilled if the approximate Riemann problem solution consistency can be demonstrated:

$$\lim_{\substack{\Delta x/h \rightarrow 0 \\ h \rightarrow 0}} \mathbf{u}_{a,ab} = \mathbf{u}(\bar{\mathbf{r}}_{ab}, t). \quad (3.14)$$

Such demonstration is out of the scope of this work. However, becoming the term symmetric, it can be asserted that the condition 3 is fulfilled.

Regarding the condition 2, since this term is proposed as a function of the divergence of the velocity resulting from the Riemann problem solution, not using therefore the Morris formula (Morris et al., 1997), it is not affected by the singularity described by Antuono et al. (2010). Vila (1999) has already analyzed the stability of the time integration, which has been

proven later in several practical applications (see for instance Molteni and Bilello (2003) and Koukouvini et al. (2013)), not requiring therefore a tuning parameter.

It should be remarked that the solution of the Riemann problem adds a significant computational cost to the model.

3.3.2 The diffusive term proposed by Ferrari et al. (2009)

Ferrari et al. (2009) proposed a diffusive term directly inside the conservation of mass equation as a simplification of the model introduced by Vila (1999). In this case Ferrari et al. (2009) chose an upwind Rusanov (1962) flux, such that, taking a constant sound speed for simplicity, the model can be written as

$$\mathbf{F}_{ij}^{\text{Fe}} = -c_s (\rho_j - \rho_i) \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|}. \quad (3.15)$$

In order to check the condition 1, a finite differences approximation can be applied,

$$\mathbf{F}_{ij}^{\text{Fe}} \simeq -c_s \langle \nabla \rho \rangle_j |\mathbf{r}_{ij}|, \quad (3.16)$$

which can be expressed in a continuous form,

$$\left\langle \frac{d\hat{\rho}}{dt} \right\rangle = -c_s \left\langle \text{div} (\nabla \rho(\mathbf{x}) |\mathbf{r}_{xy}|) \right\rangle. \quad (3.17)$$

Unfortunately, equation above is not vanishing at the limit of $h \rightarrow 0$:

$$\left\langle \frac{d\hat{\rho}(\mathbf{x})}{dt} \right\rangle = c_s \left(\int_{\Omega} \Delta \rho(\mathbf{y}) |\mathbf{r}_{xy}| W_h(\mathbf{r}_{xy}) d\mathbf{y} + \int_{\Omega} \nabla \rho(\mathbf{y}) \cdot \frac{\mathbf{r}_{xy}}{|\mathbf{r}_{xy}|} W_h(\mathbf{r}_{xy}) d\mathbf{y} \right). \quad (3.18)$$

Therefore the condition 1 is not fulfilled. However, even though the term is inconsistent with the mass conserving equation, it has a symmetric form (at least if a constant sound speed is considered), and therefore it is intrinsically conserving the global mass.

Regarding the condition 2, Antuono et al. (2012) already demonstrated that this model is affected by the singularity of the Morris formula close to the free surface.

In this term no tuning parameters are used, fulfilling in principle the condition 4. Unfortunately, this term may turn the simulation unstable if a conveniently tuned parameter is not introduced in the equation (3.15).

3.3.3 The diffusion term proposed by Molteni and Colagrossi (2009) and Antuono et al. (2010)

This term was initially introduced by Molteni and Colagrossi (2009) as a function of the Laplacian of the density field, and corrected later by Antuono et al. (2010) to fix the problems arising on the free surface, transforming it into a fourth order differential operator. The term can be written as follows:

$$\mathbf{F}_{ij}^{\text{An}} = -2\delta h c_s \left((\rho_j - \rho_i) \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^2} + f_L(\rho_i, \rho_j) \right), \quad (3.19)$$

where δ is a non-dimensional factor, and $f_L(\rho_i, \rho_j)$ is a MLS based correction term (Antuono et al., 2010; Hashemi et al., 2011).

In this model the smoothing length is specifically inserted into the term, fulfilling, for regular enough density fields, the condition 1 at the limit of $h \rightarrow 0$. The model is also intrinsically conserving the global mass, i.e. is fulfilling the condition 3, due to its symmetric form.

Antuono et al. (2010) specifically added the term $f_L(\rho_i, \rho_j)$ to the model to fix the singularity close to the free surface.

This term is proposed with an specific dependency on a δ parameter which should be tuned for each simulation. Hence the condition 4 is not fulfilled for this term. Related with that, it should be remarked that Antuono et al. (2010) performed a linear stability analysis to provide a way to tune the parameter, but adding an heuristically found limit for 2D simulations. Later a new linear stability analysis was performed (Antuono et al., 2012), such that this time the resulting δ parameter is a function of the applied kernel too, and it requires introducing another Courant condition to set the time step depending on the selected value for such δ parameter.

3.3.4 A diffusion term inspired by Fatehi and Manzari (2011) and Hashemi et al. (2011)

The last proposed term to include in the conservation of mass equation was proposed by Fatehi and Manzari (2011), who were the first to use an approximation the Laplacian of the pressure instead of the Laplacian of the density. The term was later formalized by Hashemi et al. (2011), introducing a correction concerning the boundaries, in a similar way as Antuono et al. (2010) did for the term proposed by Molteni and Colagrossi (2009). During this work a new modified version of the originally defined model is used, where the averaged density

value $(\rho_i + \rho_j)/2$ is replaced by density of reference ρ_0 :

$$\mathbf{F}_{ij}^{\text{Ha}} = -\frac{\Delta t \rho_i}{\rho_0} \left((p_j - p_i) \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^2} + \mathbf{f}_L(p_i, p_j) \right), \quad (3.20)$$

which is really a more convenient formulation to analyse the consistency.

The condition 1 is satisfied due to the involved time step Δt , which in SPH is usually set such that:

$$\Delta t \sim \frac{h}{c_s}, \quad (3.21)$$

However, it should be remarked that, not becoming symmetric, this term is not intrinsically conserving the global mass at the discrete level, which is an undesirable property. However, due to the weakly compressibility assumption ($\rho_i \simeq \rho_0$), small errors have to be expected.

In a similar way to the previous model, this model requires a correction to avoid the singularity close to the free surface, turning it consistent under the condition 2.

This term satisfies the conditions 4 as there are no tuning parameters.

Actually, considering the equation (3.21), and the equation of state in (2.39), it can be demonstrated that the terms (3.19) and (3.20) are widely related (see Cercos-Pita et al. (2016b)). Along this line, explicitly introducing the time step in the diffusive term takes the place of the tuning process and Courant condition imposition described by (Antuono et al., 2012).

3.4 Summary of diffusive δ -SPH terms

In Table 3.1 the main features of each term, discussed in the previous section 3.3, are compiled. As it can be appreciated, the new term \mathbf{F}^{Ha} , resulting from the convenient modification of the term proposed by Fatehi and Manzari (2011) and formalized later by Hashemi et al. (2012), is the only one which, not requiring a tuning parameter, its consistency has been proven. Actually, such term only fails regarding the intrinsically global mass conservation desirable feature. However, as it has been mentioned in the section 3.3.4, not large errors can be expected due to the weakly compressibility hypothesis. Effectively, the lack of either a tuning parameter or an additional associated Courant condition, reduce the required expertise to implement and apply the model, which is a great benefit.

Hereinafter, just the last 2 terms are considered, \mathbf{F}^{An} and \mathbf{F}^{Ha} , becoming the only ones in which have been proven that the 2 consistency conditions are fulfilled.

Condition	F^{Vi}	F^{Fe}	F^{An}	F^{Ha}
1. Mass conserving equation consistency	unknown*	no	yes	yes
2. Consistency close to the free-surface	no	yes	yes**	yes**
3. Intrinsically global mass conservation	yes	yes	yes	no***
4. No tuning parameters	yes	no****	no	yes

Table 3.1 Properties of existing diffusive terms proposed for the Conservation of mass equation in SPH. F^{Vi} = Vila (1999), F^{Fe} = Ferrari et al. (2009), F^{An} = Antuono et al. (2010), F^{Ha} = modified Hashemi et al. (2011).

* The consistency of the Riemann approximated solution should be analyzed. ** Far away from the boundaries, where the correction term contribution can be neglected. *** Small errors have to be expected due to the weakly compressibility hypothesis. **** Even though it has no tuning parameters, without them the simulation may turn unstable.

Chapter 4

Energy equations

During this chapter an analysis of the sources of energy dissipation of the numerical fluid system, depending on the formulation used, is carried out. This kind of analysis, including the boundaries and the external works, is an original contribution of this thesis, extending previous work by Antuono et al. (2015).

For a system where the heat transfers can be neglected, the following energy balance can be stated:

$$\frac{d\mathcal{E}_i}{dt} + \frac{d\mathcal{E}_m}{dt} = -\dot{W}_{\Omega \rightarrow \partial\Omega}, \quad (4.1)$$

where the left hand side is the total energy variation rate -i.e. power- of the fluid, while the right hand side is the delivering energy rate of the fluid to the boundaries. The mechanical energy variation rate can be computed as the sum of the potential and the kinetic energy,

$$\frac{d\mathcal{E}_m}{dt} = \frac{d\mathcal{E}_p}{dt} + \frac{d\mathcal{E}_k}{dt} = \sum_{i \in \text{Fluid}} m_i \mathbf{u}_i \cdot \left(-\mathbf{g} + \frac{d\mathbf{u}_i}{dt} \right). \quad (4.2)$$

Of course, the acceleration term, $\frac{d\mathbf{u}_i}{dt}$, is affected by the boundary condition applied (see the section 2.2.3), which is discussed below. For the time being, it is more convenient disregarding the boundary effects in the mechanical energy, reading as follows:

$$\begin{aligned} \frac{d\mathcal{E}_m}{dt} = & - \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} \mathbf{u}_i \cdot \left((p_i + p_j) + \mu K \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \right) \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} m_i \mathbf{u}_i \cdot \left\langle \frac{d\mathbf{u}_i}{dt} \right\rangle_{\partial\Omega}. \end{aligned} \quad (4.3)$$

Regarding the internal energy, the discrete system considered can be split in the dissipated/irreversible energy and the energy due to the compressibility,

$$\frac{d\mathcal{E}_i}{dt} = \frac{d\mathcal{E}_d}{dt} + \frac{d\mathcal{E}_c}{dt} = \frac{d\mathcal{E}_d}{dt} + \sum_{i \in \text{Fluid}} m_i \frac{p_i}{\rho_i^2} \frac{d\rho_i}{dt}, \quad (4.4)$$

where the rate of dissipated energy, $\frac{d\mathcal{E}_d}{dt}$, is in principle unknown.

As it happened to the kinetic energy, the compressibility term, $\frac{d\rho_i}{dt}$, is affected by the specific BC used. The effect of such BC is being intentionally relegated to a further discussion in section 4.3.

Taking into account the following relation (see for instance Antuono et al. (2015)),

$$\sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} (p_i + p_j) \mathbf{u}_i \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) = - \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} p_i (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i), \quad (4.5)$$

the variation rate of the total energy of the fluid can be written as follows:

$$\left\{ \begin{array}{l} \frac{d\mathcal{E}_i}{dt} + \frac{d\mathcal{E}_m}{dt} = \frac{d\mathcal{E}_d}{dt} - \frac{d\mathcal{E}_\mu}{dt} - \frac{d\mathcal{E}_\delta}{dt} - \frac{d\mathcal{E}_{\partial\Omega}}{dt}, \\ \frac{d\mathcal{E}_\mu}{dt} = \mu K \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \mathbf{u}_i \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i), \\ \frac{d\mathcal{E}_\delta}{dt} = - \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} \frac{p_i}{\rho_i} \mathbf{F}_{ij} \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i), \\ \frac{d\mathcal{E}_{\partial\Omega}}{dt} = - \sum_{i \in \text{Fluid}} m_i \left(\mathbf{u}_i \cdot \left\langle \frac{d\mathbf{u}_i}{dt} \right\rangle_{\partial\Omega} + \frac{p_i}{\rho_i^2} \left\langle \frac{d\rho_i}{dt} \right\rangle_{\partial\Omega} \right). \end{array} \right. \quad (4.6)$$

Finally, combining the equations (4.1) and (4.6), the expression for the total energy dissipation in the discrete system is obtained,

$$\frac{d\mathcal{E}_d}{dt} = \frac{d\mathcal{E}_\mu}{dt} + \frac{d\mathcal{E}_\delta}{dt} + \left(\frac{d\mathcal{E}_{\partial\Omega}}{dt} - \dot{\mathcal{W}}_{\Omega \rightarrow \partial\Omega} \right), \quad (4.7)$$

such that the total energy dissipated by the fluid is the sum of the power dissipated by the viscous function, the power degraded by the numerical diffusive term added to the mass conservation equation, and the power exchanged with the boundary, conveniently subtracting the energy effectively absorbed by such boundary.

4.1 Viscous dissipation function

The first term of the right hand side of Equation (4.7), $\frac{d\mathcal{E}_\mu}{dt}$, is the power associated to the viscous dissipation function, which is actually the only one with a physical meaning. In this sense, it should be demonstrated that it is a positive function, consuming therefore mechanical energy to produce irreversible internal energy. To this end, such term can be rewritten as a function of the particles pair interactions,

$$\frac{d\mathcal{E}_\mu}{dt} = -\frac{1}{2}\mu K \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i), \quad (4.8)$$

where the symmetry property of the kernel has been already applied. However, the kernel gradient can be expressed as follows by construction:

$$\nabla W_h(\mathbf{r}_j - \mathbf{r}_i) = -(\mathbf{r}_j - \mathbf{r}_i) \cdot w_h(\mathbf{r}_j - \mathbf{r}_i), \quad (4.9)$$

where $w_h(\mathbf{r}_j - \mathbf{r}_i) \geq 0$. Hence, the energy dissipation by the viscous function of the equation (4.8) is positive, *quod erat demonstrandum*.

4.2 δ -SPH dissipation function

The δ -SPH dissipation of energy, $\frac{d\mathcal{E}_\delta}{dt}$, is caused by the term F_{ij} , artificially added into the mass conservation equation. In this sense both terms, (3.19) and (3.20), vanish in the limit $h \rightarrow 0$, thus allowing to assert that,

$$\lim_{h \rightarrow 0} \frac{d\mathcal{E}_\delta}{dt} = 0. \quad (4.10)$$

However, the sign of the power term above should be analyzed again, in order to find out whether the δ -SPH term is always consuming energy due to the compressibility to produce irreversible internal energy, or it may eventually pump energy into the system, which is a really undesirable situation. To this end, we can proceed in a similar way than the described in the subsection 4.1, expressing the δ -SPH energy dissipation term as a function of the particles pair interactions:

$$\frac{d\mathcal{E}_\delta}{dt} = \frac{1}{2} \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} \left(\frac{p_j}{\rho_j} \mathbf{F}_{ji} - \frac{p_i}{\rho_i} \mathbf{F}_{ij} \right) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i), \quad (4.11)$$

Of course, the sign of the energy term above depends on the specific diffusive term \mathbf{F}_{ji} applied.

4.2.1 Molteni and Colagrossi (2009)

Considering the term of Equation (3.19), and using the kernel feature (4.9), the equation (4.11) reads:

$$\frac{d\mathcal{E}_\delta^{\text{Mo}}}{dt} = \delta h c_s \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} \left(\frac{p_j}{\rho_j} - \frac{p_i}{\rho_i} \right) (\rho_j - \rho_i) w_h(\mathbf{r}_j - \mathbf{r}_i), \quad (4.12)$$

where the correction term $f_L(\rho_i, \rho_j)$ has been neglected, as it was done by Antuono et al. (2015) and Cercos-Pita (2015). Provided that $\frac{p(\rho)}{\rho}$ is non-decreasing, the energy term, $\frac{d\mathcal{E}_\delta^{\text{Mo}}}{dt}$, becomes positive, hence dissipating energy.

4.2.2 Fatehi and Manzari (2011)

Considering the term of the equation (3.20), and using the kernel feature (4.9), the equation (4.11) reads:

$$\frac{d\mathcal{E}_\delta^{\text{Fa}}}{dt} = \frac{\Delta t}{\rho_0} \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} (p_j - p_i)^2 w_h(\mathbf{r}_j - \mathbf{r}_i), \quad (4.13)$$

where the correction term $f_L(p_i, p_j)$ has been neglected again. It can be noticed that the expression above is positive, becoming an energy dissipation term as well.

4.3 BC dissipation function

The last energy dissipation source in the equation (4.7) is the one associated with the difference between the energy variations due to the fluid interaction with the boundary, and the effective work done over such boundary. Of course, this analysis should be carried out on top of each boundary condition technique from the ones described in the section 2.2.3.

4.3.1 Kernel correction

When kernel corrections are applied, the following expressions become valid,

$$\left\langle \frac{d\rho_i}{dt} \right\rangle_{\partial\Omega} = -\rho_i \sum_{j \in \text{Fluid}} (\mathbf{u}_j - \mathbf{u}_i) \cdot [(\mathcal{L}(\mathbf{r}_i) - \mathcal{I}) \nabla W_h(\mathbf{r}_j - \mathbf{r}_i)] \frac{m_j}{\rho_j}, \quad (4.14)$$

$$\begin{aligned} \left\langle \frac{d\mathbf{u}_i}{dt} \right\rangle_{\partial\Omega} = & - \sum_{j \in \text{Fluid}} \frac{p_i + p_j}{\rho_i \rho_j} [(\mathcal{L}(\mathbf{r}_i) - \mathcal{I}) \nabla W_h(\mathbf{r}_j - \mathbf{r}_i)] m_j \\ & - \sum_{j \in \text{Fluid}} \mu K \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{\rho_i \rho_j |\mathbf{r}_j - \mathbf{r}_i|^2} [(\mathcal{L}(\mathbf{r}_i) - \mathcal{I}) \nabla W_h(\mathbf{r}_j - \mathbf{r}_i)] m_j, \end{aligned} \quad (4.15)$$

such that the energy variation due to the interaction with the boundary can be expressed as follows:

$$\begin{aligned} \frac{d\mathcal{E}_{\partial\Omega}}{dt} = & \frac{1}{2} \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} (\mathbf{u}_j - \mathbf{u}_i) \cdot (p_j \mathcal{L}(\mathbf{r}_j) + p_i \mathcal{L}(\mathbf{r}_i)) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & - \frac{1}{2} \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} (p_j + p_i) (\mathbf{u}_j \cdot \mathcal{L}(\mathbf{r}_j) + \mathbf{u}_i \cdot \mathcal{L}(\mathbf{r}_i)) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & - \frac{1}{2} \sum_{i \in \text{Fluid}} \sum_{j \in \text{Fluid}} \frac{m_i m_j}{\rho_i \rho_j} \mu K \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} (\mathbf{u}_j \cdot \mathcal{L}(\mathbf{r}_j) + \mathbf{u}_i \cdot \mathcal{L}(\mathbf{r}_i)) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & - \frac{d\mathcal{E}_v}{dt}. \end{aligned} \quad (4.16)$$

This boundary condition technique has a peculiar behavior, replacing the energy variations due to the fluid particles interactions by an approximation affected by the kernel deformation. Of course, the consistency of this BC is granted by a deformation function converging to the identity,

$$\lim_{\substack{\Delta x/h \rightarrow 0 \\ h \rightarrow 0}} \mathcal{L}(\mathbf{r}_i) = \mathcal{I}. \quad (4.17)$$

Unfortunately, if Shepard or MLS methodologies are considered, such limit is not fulfilled close to the boundaries, becoming locally inconsistent. On top of that, it is not possible to guarantee that the sign of Equation (4.16) is always negative, thus implying that this BC technique may result in spurious energy pumped into the system.

4.3.2 Boundary forces

Due to the large number of Boundary forces implementations already existing in the literature (e.g. Monaghan (1989), Souto-Iglesias et al. (2006), Crespo et al. (2007) and Kajtar and

Monaghan (2008)), carrying out an analysis of each of them becomes out of the scope of this work. However, as it was mentioned in section 2.2.3, the boundary integrals incorporate a kernel correction, and an equivalent boundary force. Such boundary force is based on the normal flux of the equation (2.38). Indeed, the density and velocity variation rates, due to the interaction with the boundary, read:

$$\left\langle \frac{d\rho_i}{dt} \right\rangle_{\partial\Omega} = -\frac{1}{\gamma_i} \sum_{j \in \text{BE}} \rho_i (\mathbf{u}_j - \mathbf{u}_i) \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) s_j, \quad (4.18)$$

$$\begin{aligned} \left\langle \frac{d\mathbf{u}_i}{dt} \right\rangle_{\partial\Omega} = & -\frac{1}{\gamma_i} \sum_{j \in \text{BE}} \frac{p_i + p_j}{\rho_i} \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) s_j \\ & - \frac{1}{\gamma_i} \sum_{j \in \text{BE}} \mu K \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{\rho_i |\mathbf{r}_j - \mathbf{r}_i|^2} \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) s_j. \end{aligned} \quad (4.19)$$

With these values, the fluid power due to the interaction with the boundary can be written as follows:

$$\begin{aligned} \frac{d\mathcal{E}_{\partial\Omega}}{dt} = & \sum_{i \in \text{Fluid}} \sum_{j \in \text{BE}} \frac{m_i}{\gamma_i \rho_i} s_j (p_i + p_j) \mathbf{u}_i \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \sum_{j \in \text{BE}} \mu K \frac{m_i}{\gamma_i \rho_i} s_j \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \mathbf{u}_i \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \sum_{j \in \text{BE}} \frac{m_i}{\gamma_i \rho_i} s_j p_i (\mathbf{u}_j - \mathbf{u}_i) \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i). \end{aligned} \quad (4.20)$$

Conversely to the case of the kernel deformations, where the boundary does not exert/receive work, $\dot{\mathcal{W}}_{\Omega \rightarrow \partial\Omega}$, due to the lack of an actual boundary to interact with, in this specific case such work should be addressed. Imposing momentum conservation, the force over a generic boundary element, j , can be computed,

$$\begin{aligned} \mathbf{F}_{j, \Omega \rightarrow \partial\Omega} = & \sum_{i \in \text{Fluid}} \frac{m_i}{\gamma_i \rho_i} s_j (p_i + p_j) \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \mu K \frac{m_i}{\gamma_i \rho_i} s_j \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i), \end{aligned} \quad (4.21)$$

and the work received by the boundary can be computed multiplying the force by the boundary velocity, and integrating along the boundary surface:

$$\begin{aligned} \dot{W}_{\Omega \rightarrow \partial\Omega} = & \sum_{i \in \text{Fluid}} \sum_{j \in \text{BE}} \frac{m_i}{\gamma_i \rho_i} s_j (p_i + p_j) \mathbf{u}_j \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \sum_{j \in \text{BE}} \mu K \frac{m_i}{\gamma_i \rho_i} s_j \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \mathbf{u}_j \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i). \end{aligned} \quad (4.22)$$

Therefore, a mismatching energy variation rate, with respect to the power effectively absorbed by the fluid, due to the interaction with the boundary, has been obtained.

$$\begin{aligned} \frac{d\mathcal{E}_{\partial\Omega}}{dt} - \dot{W}_{\Omega \rightarrow \partial\Omega} = & - \sum_{i \in \text{Fluid}} \sum_{j \in \text{BE}} \frac{m_i}{\gamma_i \rho_i} s_j p_j (\mathbf{u}_j - \mathbf{u}_i) \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & - \sum_{i \in \text{Fluid}} \sum_{j \in \text{BE}} \mu K \frac{m_i}{\gamma_i \rho_i} s_j \frac{(\mathbf{u}_j - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} (\mathbf{u}_j - \mathbf{u}_i) \cdot \mathbf{n}_j W_h(\mathbf{r}_j - \mathbf{r}_i). \end{aligned} \quad (4.23)$$

The expression above should vanish at the continuum, consistently with the no-slip boundary condition. Unfortunately, as for the kernel correction techniques, the expression above has not a defined sign, implying that spurious energy could be artificially pumped into the fluid particles system due to the interaction with the boundary.

At this point, it is worth recalling the expression (4.21). Neglecting the viscous term, the force of the fluid over a generic boundary element, j , should be equal to:

$$\mathbf{F}_{j, \Omega \rightarrow \partial\Omega} = p_j \mathbf{n}_j s_j, \quad (4.24)$$

which is true only if the pressure on the boundary element is computed as follows:

$$p_j = \frac{\sum_{i \in \text{Fluid}} \frac{m_i}{\gamma_i \rho_i} p_i W_h(\mathbf{r}_j - \mathbf{r}_i)}{1 - \sum_{i \in \text{Fluid}} \frac{m_i}{\gamma_i \rho_i} W_h(\mathbf{r}_j - \mathbf{r}_i)}, \quad (4.25)$$

conversely to the usually applied Shepard filtered extrapolation (e.g. Cercos-Pita (2015); Ferrand et al. (2013); Macià et al. (2012)),

$$p_j = \frac{1}{\gamma_j} \sum_{i \in \text{Fluid}} \frac{m_i}{\rho_i} p_i W_h(\mathbf{r}_j - \mathbf{r}_i). \quad (4.26)$$

In Fig. 4.1 both extrapolation methods are compared, considering an infinite lattice of particles cropped by the infinite plane $x = 0$, and different pressure fields. As it can be

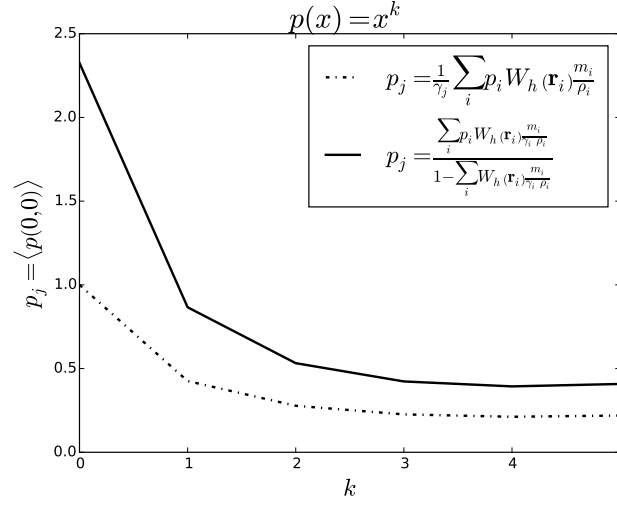


Fig. 4.1 Comparison between the pressure using the extrapolation methods at the boundary, for different pressure fields. 2D infinite lattice of particles, cropped by $x = 0$ infinite plane. $h = 1$, $\Delta x = \Delta y = \frac{h}{8}$.

appreciated, if momentum conservation is imposed, inconsistent pressure fields will be retrieved, turning the differential operators inconsistent as well (see Macià et al. (2012)).

4.3.3 Fluid extensions

In the case of the fluid extensions the effect of the boundary in the fluid particles is just an extension of the sums along the neighbours, where the ghost particles should be included as well,

$$\left\langle \frac{d\rho_i}{dt} \right\rangle_{\partial\Omega} = - \sum_{j \in \text{GP}} \rho_i (\mathbf{u}_j^\rho - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \frac{m_j}{\rho_j}, \quad (4.27)$$

$$\begin{aligned} \left\langle \frac{d\mathbf{u}_i}{dt} \right\rangle_{\partial\Omega} = & - \sum_{j \in \text{GP}} \frac{p_i + p_j}{\rho_i} \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \frac{m_j}{\rho_j} \\ & - \sum_{j \in \text{GP}} \mu K \frac{(\mathbf{u}_j^\mu - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{\rho_i |\mathbf{r}_j - \mathbf{r}_i|^2} \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \frac{m_j}{\rho_j}, \end{aligned} \quad (4.28)$$

where the velocities \mathbf{u}_j^ρ and \mathbf{u}_j^μ may differ from the body velocity, \mathbf{u}_j , depending on the mirroring/extension process applied, as it was discussed in section 2.2.3. It should be noticed that the density and pressure fields, ρ_j and p_j , depend as well on the mirroring/extension model.

The analogy between the expressions above and the ones presented in the numerical scheme in section 2.3, where the BCs were not considered, can be appreciated. With fluid extensions, the power of the fluid particles system, due to the interaction with the boundary, can be expressed as follows:

$$\begin{aligned} \frac{d\mathcal{E}_{\partial\Omega}}{dt} = & \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} (p_i + p_j) \mathbf{u}_i \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} \frac{(\mathbf{u}_j^\mu - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \mathbf{u}_i \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} p_i (\mathbf{u}_j^\rho - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i). \end{aligned} \quad (4.29)$$

In this case, the work on the boundary should be computed as well to close the balance. To this end, the expressions described in Marrone et al. (2013) and Bouscasse et al. (2013), to compute the force of the fluid over the boundary, are considered. Such expressions result from the momentum conservation imposition, similarly to the procedure discussed in the previous subsection, 4.3.2. Therefore, the work exerted by the fluid on the boundary can be written as follows:

$$\begin{aligned} W_{\Omega \rightarrow \partial\Omega} = & \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} (p_i + p_j) \mathbf{u}_j \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} \frac{(\mathbf{u}_j^\mu - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \mathbf{u}_j \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i). \end{aligned} \quad (4.30)$$

Hence, the extra energy term, led by the differences between the power of the fluid resulting from the interaction with the boundaries, and the effective work received by the solid, reads:

$$\begin{aligned} \frac{d\mathcal{E}_{\partial\Omega}}{dt} - \dot{W}_{\Omega \rightarrow \partial\Omega} = & - \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} (p_j + p_i) (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & - \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} \frac{(\mathbf{u}_j^\mu - \mathbf{u}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} (\mathbf{u}_j - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i) \\ & + \sum_{i \in \text{Fluid}} \sum_{j \in \text{GP}} \frac{m_i m_j}{\rho_i \rho_j} p_i (\mathbf{u}_j^\rho - \mathbf{u}_i) \cdot \nabla W_h(\mathbf{r}_j - \mathbf{r}_i). \end{aligned} \quad (4.31)$$

Each differential operator leaves its imprint on the energy, which strongly depends on the mirroring/extension model. For consistency the terms above should converge to zero at the continuum, which is fulfilled by the first 2 terms due to the no-slip BC imposition. Regarding the latter, it should be consistent provided the consistency of the velocity divergence operator.

As it will be numerically demonstrated later, while the first and last terms are actually negligible, the second term may become significant. Fortunately, if the body velocity is used to compute the viscous term, $\mathbf{u}_j^\mu = \mathbf{u}_j$, the extra energy term becomes positive, and therefore, an energy dissipation (see section 4.1).

4.3.4 Boundary Integrals

As it has been remarked in the section 2.2.3, the boundary integrals methodology is actually a combination of the kernel deformations and the boundary forces. In this sense, the extra energy terms associated to this BC are just the sum of the right-hand side of the equations (4.16) and (4.23).

Unfortunately, the kernel deformation requirement cause that this formulation is not able to conserve momentum or energy, as it has been discussed in section 4.3.1.

Chapter 5

AQUAgpusph features

5.1 Free software

AQUAgpusph is, and ever will be, free software. It is currently licensed under the GPLv3 terms, so that, the users are welcome to read, edit and redistribute the code, with the main limitation that the redistributed versions must be shared with the same license. Of course, the possibility of accessing the source code of the program, as well as the source code of the tools packaged within it, is not only a desirable feature for the software in general, but also a mandatory one if the research community is the main target.

At this point, the difference between the free and open-source software should be pointed out. Citing Richard Stallman, founder of the free software foundation, when we call software “free”, we mean that it respects the users’ essential freedoms: the freedom to run it, to study and change it, and to redistribute copies with or without changes. Along this line, it can be asserted that free software is moved by ethical/philosophical values, resulting in pragmatical benefits, while open-source software is directly motivated by such pragmatical benefits. The difference in the values results in actual differences in practice. For instance, both DualSPHysics (Crespo et al., 2015) and GPUSPH (Herault et al., 2010) are licensed under the same GPLv3 terms. However, they cannot be called free but open-source software, as they are actually self-defining such packages, since both are accelerated using CUDA, requiring therefore a privative library, and a specific hardware owned by an unique manufacturer.

On the other hand, sometimes the gap between the open-source and the free software can be built with the license itself. For instance, PySPH software (Ramachandran and Kaushik, 2013) is provided with the full source code, and does not require any privative piece of software to work. However, such software is licensed under BSD license terms, stating that the software can be modified and redistributed in binary format, i.e. without the source code, automatically truncating the original freedoms.

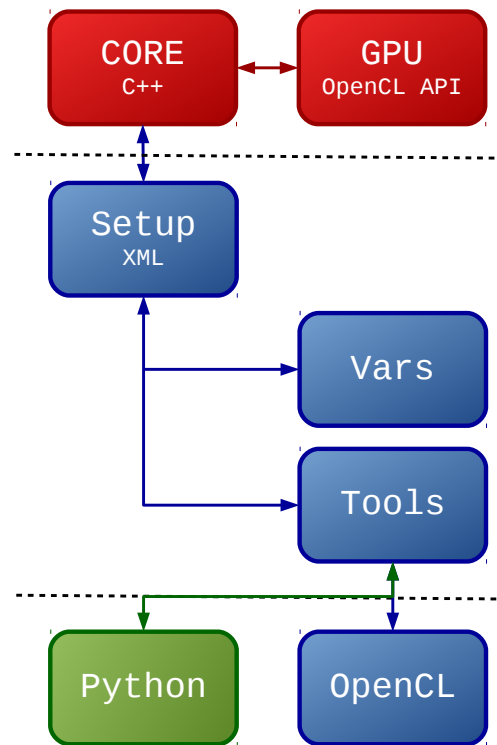


Fig. 5.1 Schematic view of the AQUAgnusph modular implementation. The levels are separated by dashed lines. The colors denote the complexity of the development, Red: High complexity, Blue: Intermediate complexity, Green: Low complexity

5.2 Modular application

5.2.1 General

As it has been discussed in section 5.1, AQUAgnusph (Cercos-Pita, 2015) is free software, and therefore the source code is available for the researchers, who are welcomed to modify and adapt it to their necessities. However, becoming a desirable feature, it is clearly insufficient if the user has to deal with the C++ core, and even worst, with the OpenCL and Python APIs. AQUAgnusph has been divided in 3 different levels, as it is illustrated in Fig. 5.1, in order to provide a software where the user can remain unaware of those high-level programming environments.

The top level is the AQUAgnusph core, where the most complex development pieces of the whole package are located. Actually, the AQUAgnusph executable binary is compiled on top of the code of such level. Accordingly, normal users and developers would never need to modify the source code at this level, or even to recompile the AQUAgnusph binary.

On the other hand, the actual computational tools, i.e. the pieces of code to be executed during each time step, are located in the bottom level.

In order to join both levels, the modular boosted XML simulations definition has been inserted. As it is discussed below, in such level all the variables and tools that will be used during the simulation can be specified and organized.

5.2.2 Variables

A set of variables can be asked to be generated to AQUAplus in the XML definition files. Those variables can be later read and written by the tools, as it will be discussed below.

Currently, there are 2 main groups of variables: Scalars and Arrays. The former are allocated in the host memory, with the main benefit that they can be accessed to either read or write, without a significant performance penalty. The Arrays, on the other hand, are lists allocated in the computational tool memory, with a fixed length. The great advantage of the arrays is that they can be processed in parallel in the OpenCL scripts, as it will be discussed in section 5.3. However, accessing this memory outside of the OpenCL codes is requiring a memory transfer between the host and the computational device, which is usually a slow operation due to the relatively low bandwidth available.

In Table 5.1, the full list of variable types that can be defined in AQUAplus are listed. As it can be appreciated, in the 3D simulations 4D vectors and matrices are used, which may result in significant performance improvements depending on the specific computational device used.

5.2.3 Tools

In parallel to the variables discussed above, the tools to be executed at each time step can be defined as well. Several kinds of tools can be used in AQUAplus, that can be grouped by the library which drives them:

1. muParser tools
2. OpenCL tools
3. Python tools
4. Dummy tools

Each specific tool will be further discussed later.

At the time of defining a tool, it can be appended at the end of the already existing tools stack, or it can be eventually inserted before/after an specific tool, identified by its name or

Table 5.1 List of variable types accepted in AQUA_{gpusph}. N can take the values 2, 3 or 4

Type	Description
int	Integer scalar variable
unsigned int	Unsigned integer scalar variable
float	Floating point scalar variable
ivec N	Vector of N integer components
ivec	ivec2 for 2D simulations, ivec4 for 3D simulations
uivec N	Vector of N unsigned integer components
uivec	uivec2 for 2D simulations, uivec4 for 3D simulations
vec N	Vector of N floating point components
int*	Array of integers
unsigned int*	Array of unsigned integers
float*	Array of floating point values
ivec N *	Array of vectors of N integer components
ivec*	ivec2* for 2D simulations, ivec4* for 3D simulations
uivec N *	Array of vectors of N unsigned integer components
uivec*	uivec2* for 2D simulations, uivec4* for 3D simulations
vec N *	Array of vectors of N floating point components
vec*	vec2* for 2D simulations, vec4* for 3D simulations
matrix*	Array of floating point matrices. 2x2 for 2D simulations, 4x4 for 3D ones

position. Optionally, the tools can be replaced, redefined, or even removed from the tools stack.

5.2.4 Reports

The reports are quite similar to the tools. More specifically, they can be managed as a second stack of tools to be executed right after the tools one, discussed in the previous subsection. However, the reports are not designed to perform actual computational tasks, but to print some data at the end of each time step, either at the terminal or into an output file.

The reports may constitute a really powerful helper since they can extract some data from the simulation at runtime. Among others, it can be mentioned the fluid energy, the forces over an specific body, performance indicators, or simulation progress.

Since the reports are just extracting already computed data, it is not relevant how are they sorted on the execution stack. Hence, the reports can just be appended at the end of such stack.

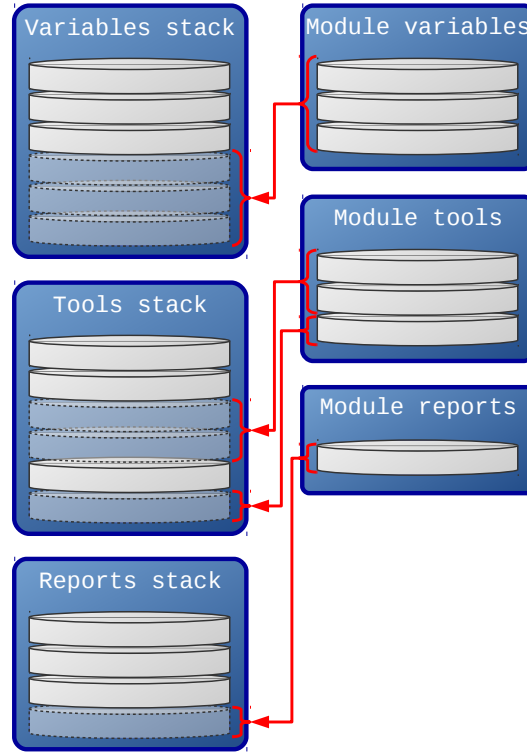


Fig. 5.2 Schematic example of a generic module loading process

5.2.5 Modularization

In order to enhance the variables, tools, and reports definition described in sections 5.2.2-5.2.4, the possibility to include XML files from another XML files has been implemented. Therefore, the variables, tools, and even the reports, can be grouped in modules, also called presets. Such process is shown in Fig. 5.2. As it can be appreciated, the module may contain sets of variables, tools, and reports that are inserted into the already existing stacks when it is loaded. As it has been discussed before, while the reports can be only appended at the end of the stack, the variables may eventually replace an already existing one, and the tools may be inserted in an arbitrary position.

To make easier the tools stack management, dummy tools can be added in order to mark events to other modules. For instance, a 3rd party module can be designed to insert a bunch of tools after the particles interactions process, i.e. differential operators computation, but before such differential operators are used to compute the magnitudes variation rates. To this end, dummy particles can be defined to highlight when the particles interactions computation

have finished, and when the rates of computation would begin. Hence, the modules can be easily split, and the dependencies between them well tracked.

Examples of this modularization will be provided during the practical applications in the section 6.

5.3 OpenCL acceleration

5.3.1 General

As aforementioned, AQUA_{Agpusph} has been accelerated with OpenCL. Moreover, AQUA_{Agpusph} can run all the simulation on the computational device (CPU, GPU,...), falling back to the host exclusively for printing output files, significantly reducing the performance penalties due to the eventually low bandwidth assigned to the computational device.

The key features that have led the author to use OpenCL instead of other alternatives, such as CUDA (NVidia, 2006), OpenMP (OpenMP Architecture Review Board, 2013), or MPI (MPI Forum, 2013) (already used in other free available SPH implementations (Cherfils et al., 2012; Dominguez et al., 2013; Herault et al., 2010; Liu and Liu, 2003)) are described below. However, it should be remarked that some other SPH implementations using OpenCL already exist, namely:

1. Blender (Blender foundation, 2013) has implemented its own SPH fluids simulator, designed to obtain good visual effects.
2. Bullet (Bullet physics library, 2013) has a similar implementation focused on providing great visual results. This engine additionally has fluid-solid interaction capabilities implemented.
3. PySPH (Ramachandran and Kaushik, 2013) is a framework for Python partially accelerated with OpenCL.

While the first two aforementioned alternatives are not designed for researches or engineers, the latter one can be actually considered for science and engineering, despite the fact that it is not fully accelerated with OpenCL yet.

5.3.2 More powerful devices and lower costs

When using the CUDA framework, only NVidia devices can be used, but with OpenCL hardware coming from other vendors can be applied, which may result in a substantial cost and power difference. Currently AMD commercializes graphic devices that are approximately

50% times more powerful and 50% times less expensive than their NVidia counterpart. For instance an AMD ATI HD 7970 card (approximately 280€), has a computational power of 4.2 Tflops, compared to an NVidia GTX 680 card (approximately 450€) with a computational capability of 3.0 tflops.

These costs and power differences are really volatile, being dependant on the game industry or on the global IT market. However, they illustrate the importance of not restricting your developments to a unique vendor.

For this reason, NVidia is currently promoting a new standard for Open Accelerators (OpenACC), which has been successfully applied in the context of Lagrangian simulations (Niemeyer and Sung, 2013). OpenACC, in front of CUDA, is designed to simplify parallel programming of heterogeneous CPU/GPU systems, and in front of OpenCL, is based in annotations in the source code to specify the areas to be accelerated.

It should be mentioned that although CUDA is designed to work just with NVidia devices, becoming heavily optimized for such hardware, and therefore using OpenCL (or OpenACC either) in a NVidia device has the drawback that lower performance can be expected.

5.3.3 Hardware diversification

OpenCL is not restricted to GPUs, CPUs or IBM Cell architecture, but is able to deal with all of them. This feature must be considered seriously when selecting the software development framework since these architectures are in continuous evolution.

Furthermore, in other SPH solvers 3 versions of the software need to be maintained. Those are:

1. A serial CPU version.
2. A parallel CPU version based.
3. A CUDA based implementation.

With the OpenCL standard a unique version of the code can cover all the aforementioned versions, and more over, can do it simultaneously, allowing the usage of GPUs and CPUs in the same simulation. It should be noticed that using OpenMP or MPI the serial and the parallel CPU codes can be joined in a single implementation.

5.3.4 Applications with the capability of being modified

In a similar way to the Open Graphics Library (OpenGL), a set of libraries to develop hardware-accelerated 2-D and 3-D vector graphics applications, OpenCL based programs

can optionally execute codes which are loaded and compiled at runtime, and could therefore be modified without the need to recompile the main program.

In AQUAopusph, all the OpenCL source codes used to perform the computation are available as independent files, and their execution is organized using the boosted XML definition files, as it has been discussed in section 5.2. Since AQUAopusph-2.0, the program is able to automatically recognize and provide the variables required by the OpenCL code, from the list of defined ones (see section 5.2.2). It is also able to detect when a variable has been modified, such that a variable will not be resent to the OpenCL code until it should not be updated, slightly reducing the overhead.

Creating these kind of customizable applications with other frameworks, like CUDA, is really complex and may result in an undesirable messy coded software.

5.4 Python extensible

As it has been mentioned, AQUAopusph core has been developed in C++ language, allowing the deployment of a high performance application, which is actually a critical factor of a SPH code. Unfortunately, the usage of such language imply an unacceptable coding overhead as well as a high level of complexity, which may expel a significant part of the research community, depending on their expertise in the computational sciences.

Such frictional issues can be significantly lightened by the integration of Python scripts extensibility capabilities. Python is a script based language that has become very popular in recent years due to its simplicity, power and increasing support. It is a free language that does not require a private platform to be executed, and has been implemented in the most widely used operating systems.

The Python extensibility in this context, and its main benefits, were already described by Cercos-Pita (2015). However, since AQUAopusph-2.0 it is possible to call Python scripts at an arbitrary point of the time loop. Also, the Python scripts are not restricted to the solid motions management anymore, but all the simulation data can be read and written.

For instance, the following source code is increasing a generic counter, c , defined as an unsigned integer scalar variable in the variables stack described in the section 5.2.2:

```
import numpy as np
import aquaopusph as aqua

def main():
    c = aqua.get("c")
    aqua.set("c", c + 1)
```



```
return True
```

Indeed, the Python module, `aquagpusph`, provides access to all the variables defined. Eventually, whole array variables can be downloaded, as well as subsets of them. Unfortunately, the performance deployed by the Python scripts is in general several orders of magnitude lower than the OpenCL accelerated codes. In this sense, Python scripts are strongly recommended when arbitrary complex operations should be carried out, provided that they do not require to read or write a large amount of data. Otherwise, OpenCL scripts should be considered.

As a similar approach PySPH (Ramachandran and Kaushik, 2013) can be considered. However, it should be remarked that AQUA`gpusph` is a C++ application, extended with Python, while PySPH is mainly implemented in Python, eventually extended with C and Fortran languages.

5.5 muParser fast math parser

Even though AQUA`gpusph` provides a powerful Python environment, which can be used to operate scalar variables (see section 5.2.2), in order to reduce the overhead the fast math parser, `muParser`, has been integrated. Such tool can be used to perform operations between scalars, without requiring a full Python environment.

Incidentally we must remark that the `muParser` library is used as well at the initial condition loading process from ASCII files, so that some fields can be expressed as a math function.

5.6 Wide variety of boundary conditions

5.6.1 General

As it is aforementioned, SPH methodology is growing fast leaded by the increasing interest on the model, bringing a rate of changes and new modifications not seen before. Along this line, AQUA`gpusph` package brings a wide variety of formulations and tools “out of the box”. Probably, the most representative example of that is the wide variety of boundary conditions deployed within the software. During this section such boundary conditions will be presented, providing as well some details regarding the algorithm used to this end.

5.6.2 Solid boundary conditions

While other boundary conditions have been implemented using just the fluid extensions approach, significant efforts have been invested to introduce several alternatives for the solid boundary condition, using all the approaches described in section 2.2.3.

Shepard and MLS

The 2 most popular kernel deformation functions have been implemented, the Shepard renormalization factor (Belytschko et al., 1998), and the MLS function (Randles and Libersky, 1996). Conversely to the Shepard renormalization factor implementation, which is a trivial operation provided that a good enough SPH framework has been deployed, MLS is requiring the introduction of matrix operations, which are not supported “out of the box” by OpenCL. As a workaround, the `matrix*` variable type has been introduced in AQUA_{gpusph}-3.0. This type is an abstraction of the floating point variables alignment, i.e. it is equivalent to `float4` in 2D, and `float16` in 3D. Hereinafter, we are only describing the 2D matrix implementation, for the sake of simplicity.

Unfortunately, accessing the matrix elements is not a direct operation, since OpenCL is internally applying a vector description: $M_{11} = \mathbf{M}.\mathbf{s0}$, $M_{12} = \mathbf{M}.\mathbf{s1}$, $M_{21} = \mathbf{M}.\mathbf{s2}$, $M_{22} = \mathbf{M}.\mathbf{s3}$. So in order to keep the user unaware from this relatively complex matrix manipulation system, a complete linear algebra set of operators is provided. For instance, in order to perform the MLS matrix for each particle, an outer product, and matrix pseudo-inversion operators are required. The latter is compound itself by several operators, namely matrix transposition, matrix multiplication, and matrix inversion. Finally matrix inversion requires matrix determinant operators. All these operators are implemented as follows:

```
matrix outer(const vec v1, const vec v2)
{
    matrix m;
    m.s01 = v1.x * v2;
    m.s23 = v1.y * v2;
    return m;
}

float det(const matrix m)
{
    return m.s0 * m.s3 - m.s1 * m.s2;
}
```

```

matrix inv(const matrix m)
{
    const float d = 1.f / det(m);
    return ((matrix)( m.s3 , -m.s1 ,
                      -m.s2 ,  m.s0 )) * d;
}

#define TRANSPOSE s0213

#define MATRIX_INV(_M) \
MATRIX_MUL( inv (MATRIX_MUL(_M.TRANSPOSE, _M)) , _M.TRANSPOSE)

```

It should be remarked that in both cases, the Shepard renormalization factor and the MLS matrix are ephemeral properties of the particles, i.e. they should be computed once per time step.

Elastic bounce

Actually the elastic bounce is the simplest solid boundary condition model (Ihmsen et al., 2011; Simpson and Wood, 1996). In such boundary condition the force necessary to prevent the particles trespassing through a solid wall is computed, such that at the end of the time step

$$\mathbf{u}_i \cdot \mathbf{n} = (\mathbf{U} - \mu_e \mathbf{u}_i) \cdot \mathbf{n}, \quad (5.1)$$

where \mathbf{u}_i is the resulting velocity for a generic particle i , \mathbf{n} and \mathbf{U} are the normal and the solid boundary speed respectively, and μ_e is the elastic bounce factor ($\mu_e = 1$ corresponds to a full kinetic energy conservative interaction, and $\mu_e = 0$ to a full kinetic energy dissipative interaction). This technique can therefore be included into the boundary forces category. Indeed, to efficiently implement this method, the boundary is discretized in a set of area elements as is depicted in Fig. 2.3, where the position, normal, velocity and acceleration of the boundary are stored. Then the interactions of a generic particle, i , with the boundary can be described with the following pseudo-code:

```

for j in neighbours:
    if j is not elastic_bounce:
        continue
    # Check if the particle would trespass the boundary
    dist = dot(r_j - r_i , n_j)

```

```

    vel = dot(u_i - u_j , n_j)
    if dist > vel * dt:
        continue
    # Correct the normal velocity
    u_i += dot(u_j - (1 + mu_e) * u_i , n_j) * n_j

```

Some benefits may be remarked for this boundary condition:

1. It is the simplest one.
2. It is efficient both in memory consumption and in computational time.
3. It is really robust, being the best way to ensure that the particles do not trespass the solid walls.
4. It is able to deal with complex geometries.
5. It can be combined with all other boundary conditions.

Nonetheless, it is also important to remark that the accuracy of the model is low. For this reason, this boundary condition is often used in combination with other boundary condition models.

Ghost particles

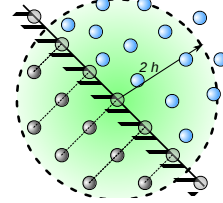
In AQUA_{Agpusph}-2.0 main fluid extensions models can be applied. Along this line, the simplest approach is setting a set of dummy particles outside the fluid domain, as it was schematically shown in Fig. 2.4. The body velocity is imposed to such dummy particles, while the density and pressure fields derive from the continuity equation (e.g., Issa et al. (2005), Crespo et al. (2007)). After that, those dummy particles can be used as regular neighbour particles during the interactions process.

However, the improved model described by Marrone et al. (2011a) and Marrone et al. (2011b), has been implemented as well. In such model the pressure, density and velocity fields result from the application of mirroring processes. To this end, on top of the dummy particles placed outside the fluid domain, the boundary is discretized again in a set of area elements as depicted in Fig. 2.3, such that each dummy particle is associated to an area element, which should be its projection over the wall, as it is depicted in Fig. 5.3. It can be noticed that several dummy particles may share the same boundary element, which could be in fact an elastic bounce boundary element.

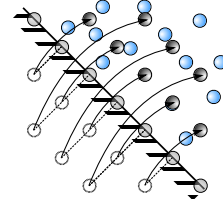
Hence, the process can be divided in several stages, described in Table 5.2.

Table 5.2 Ghost particles process

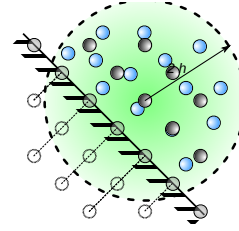
The Laplacian of the velocity is computed along the boundary elements



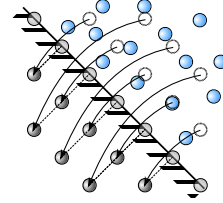
The dummy particles are mirrored with respect the area elements



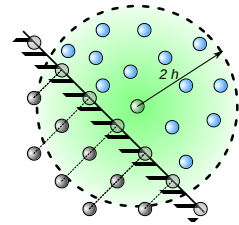
The mirrored pressure, and velocity fields, p^*, u^* are interpolated for the mirrored dummy particles. These values should be renormalized, such that the Shepard renormalization factor should be computed as well. Regarding the density field, it is resulting from the application of the EOS



The dummy particles positions are restored, and the pressure, velocity, and density fields conveniently computed using the mirrored interpolated values. For instance, the pressure field is computed imposing the following Neumann boundary condition: $\nabla p \cdot \mathbf{n} = \rho \left[\mathbf{g} \cdot \mathbf{n} - \frac{dU}{dt} \cdot \mathbf{n} + \nu \Delta \mathbf{u} \cdot \mathbf{n} \right]$, where \mathbf{n} and \mathbf{U} are the normal and velocity of the associated area element.



The dummy particles can be now considered as usual fluid neighbour particles during the interactions computation.



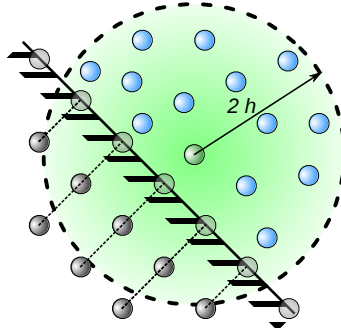


Fig. 5.3 Schematic view of the ghost particles methodology

This boundary condition has demonstrated to be very robust and reliable (Bouscasse et al., 2013; Marrone et al., 2013, 2011b). However, some drawbacks should be considered that may lead to the usage of a different boundary condition:

1. As it can be appreciated, it is hard to be implemented, making room to errors.
2. It would become non-trivial to dispose the boundary particles, making harder to deal with complex geometries.
3. In the traditional Ghost particles implementation, the memory required can be quite significant and is heavily dependent on the geometry.
4. The computational effort grows as $O((h/\Delta x) \cdot (1/\Delta x)^{d-1})$. In addition to that, the complexity of the implementation is inexorably imposing a penalty to the performance.

Boundary integrals

In AQUA_{gpusph} the purely numerical model or the equation 2.38 is applied, conversely to the original semi-analytical approach discussed by Ferrand et al. (2013). Such purely numerical approach allows to efficiently extend the boundary integrals methodology to 3D applications, eventually parallelized.

Hence, as it happened with the elastic bounce and the ghost particles, the boundary should be discretized in area elements as depicted in Fig. 2.3. The velocity of the boundary elements is equivalent to the body velocity, while the pressure is extrapolated from the fluid available information, conveniently applying the Shepard renormalization factor. The density is obtained just using the EOS.

A direct consequence is that this boundary condition can be easily applied to complex geometries. Also it is relatively efficient both in memory consumption and in computational time. Unfortunately, this model has the drawback that the Shepard renormalization breaks

the symmetric form of the operators shown in the equation 2.39, having a dramatic effect on the energy conservation, as it was discussed in the section 4.3.

5.6.3 Inflow/outflow

In AQUAplus the inflow and outflow BCs have been implemented only for uniform flows, i.e. the velocity, pressure and density are constant magnitudes along the inflow and outflow planes. Outflow BC is just a plane. When a generic fluid particle, i , trespass such plane, its properties are replaced by the uniform outflow ones, $\mathbf{u}_i = \mathbf{U}_o$, $\rho_i = \rho_0$, $p_i = p_0$. After some time steps, when the particle is far enough from the outflow plane to become neglected from the interactions with the fluid particles, it is removed.

In AQUAplus, the particles are not actually removed, but stored in a buffer where they are ignored. This feature is conveniently used to implement the inflow plane BC. Effectively, tracking the last layer of particles injected in the inflow, it can be detected when the inflow is starving, extracting a new layer of particles from the buffer.

This implementation is similar to the described by Federico et al. (2012).

5.6.4 Periodic

In AQUAplus there is not implemented an actual periodic BC, but a portal one. When a particle is close enough to the entrance of the portal, it may interact with the particles close to the exit of the portal. Also, when a particle trespass the portal entrance, it is automatically teleported to the exist. The process is illustrated in Table 5.3

Of course, 2 portals with the entrance and the exit swapped result in a periodic boundary condition, as it is described in the section 2.1.1. The possibility of using single portals is just adding more generality. An example of the practical application of this boundary condition will be discussed in the section 6.

5.6.5 Symmetry

Symmetry BCs are actually very similar to the portal BCs described above. However, this time the particles are virtually mirrored respect the symmetry plane, i.e. the particles are not really affected, just the mirrored positions and velocities are computed, carrying out a second pass on interactions, centered in such new mirrored virtual position. The process is illustrated in Table 5.4

Table 5.3 Portal boundary condition process

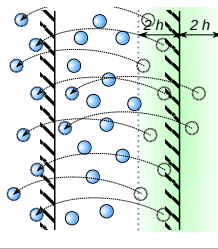
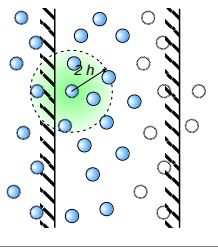
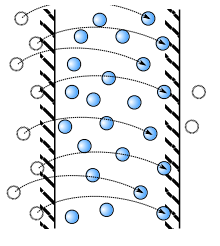
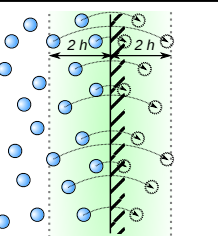
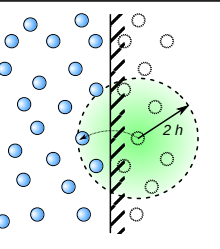
<p>After the usual fluid interactions process, the particles close enough to the portal entrance are moved to the portal exit, keeping the same relative position.</p>	 <p>The diagram shows a vertical dashed line representing a portal entrance on the left and a solid vertical line representing the portal exit on the right. A green shaded region between them represents the portal. Blue circles representing particles are shown on the left side of the entrance. Arrows indicate that particles within a distance $2h$ from the entrance are being moved to the exit, maintaining their relative positions. The distance $2h$ is marked on both sides of the entrance.</p>
<p>The interactions of the teleported particles are computed again. Since the neighbours information has not been refreshed, the transformed particles may not interact between them in this stage.</p>	 <p>The diagram shows the same portal setup. A green circle of radius h is centered on a particle that has been teleported to the exit. This circle represents the interaction range of that particle. Other particles are shown in the region, but they are not yet interacting with the teleported particle because their neighbor information is outdated.</p>
<p>The particles which were not trespass the portal yet are send again to their original position.</p>	 <p>The diagram shows the portal setup. Particles that were not within the $2h$ range of the portal entrance are shown being moved back to their original positions on the left side of the entrance. The green shaded region and the $2h$ distance markers are still present.</p>

Table 5.4 Symmetry boundary condition process

<p>After the usual fluid interactions process, the particles close enough to the symmetry plane are virtually mirrored, i.e. the mirrored position and velocity is computed and stored in different Arrays.</p>	 <p>The diagram shows a vertical dashed line representing a symmetry plane. Blue circles representing particles are shown on both sides of the plane. A green shaded region of width $2h$ is centered on the symmetry plane. Arrows indicate that particles within this region are being mirrored across the plane. The distance $2h$ is marked on both sides of the plane.</p>
<p>The interactions of the particles close enough to the symmetry plane are computed again. However, in the numerical scheme 2.39 the position \mathbf{r}_i and the velocity \mathbf{u}_i are replaced by the mirrored values. After that, the mirrored values are discarded.</p>	 <p>The diagram shows the same symmetry plane setup. A green circle of radius h is centered on a particle near the symmetry plane. This circle represents the interaction range of that particle. Other particles are shown in the region, but they are not yet interacting with the mirrored particle because their neighbor information is outdated.</p>

Incidentally, it should be remarked that currently it is not possible to use the implementation of the portal BC, discussed in section 5.6.4, to simulate Symmetry BCs. An example of the practical application of this boundary condition will be discussed in the section 6.

Chapter 6

Verification and validation

6.1 General

Even though a wide variety of simple test cases have been successfully carried out with AQUAplus, for the sake of simplicity, only 4 representative verification and validation cases are documented in this thesis.

We'll take the advantage of the relatively low complexity of the cases to show some details about the application modularization. In addition to that, the moving square inside a box of the section 6.5 is perfectly suited to verify the Energy components discussed in the section 4, while the moving square inside a fixed box of the section 6.5 can be used to show the Python extension capabilities.

6.2 Lid-driven cavity flow

6.2.1 Case description

The lid-driven cavity flow has been widely applied in the past to verify CFD tools, becoming in fact a standard (see Ghia et al. (1982), Ku et al. (1987) and Chern et al. (2005)). Moreover, it has been included in the set of SPHERIC validation tests.

In this test case a square cavity filled with an incompressible fluid is considered, where the top boundary is horizontally moving with a constant velocity U . The flow starts from rest, letting it to evolve until a stationary solution is reached. To this end, no-slip boundary conditions should be imposed at all the walls, including the non-moving ones. The problem is schematically depicted in Fig. 6.1.

In Table 6.1 all the selected variable values are specified. Even though 2 Reynolds numbers are usually selected, $Re = 10^3$, $Re = 10^4$, in this thesis just the lowest variable will

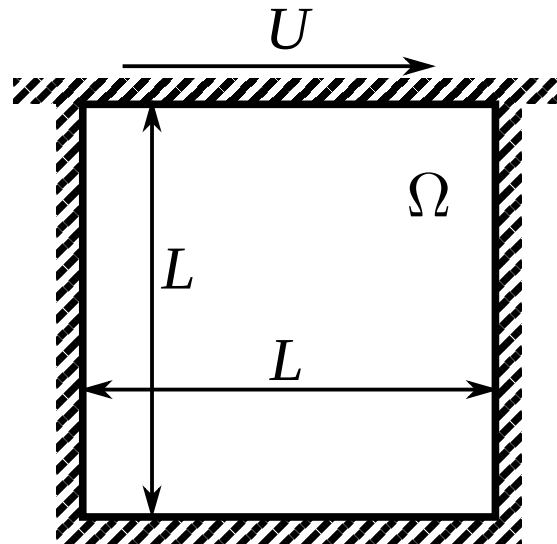


Fig. 6.1 Lid-driven cavity definition sketch.

Table 6.1 Variables selected in the lid-driven cavity test case

Variable	Value
L	1 [m]
U	1 [m / s]
Re	1000
ρ_0	1 [kg / m ²]
μ	10^{-3} [Pa · s]
c_s	50 [m / s ²]
p_0	3 [Pa]
Δr	0.01 [m]
$h/\Delta r$	4
Co	0.25

be considered. With the selected distance between particles, Δr , the square cavity is indeed discretized by 100 fluid particles in each direction. With such resolution, the simulation can be carried out without adding an extra dissipation, i.e. without artificially increasing the viscosity or adding the δ -SPH term either.

6.2.2 Implementation details

Because of its simplicity, the lid-driven cavity test case is perfect to show the modular design of AQUA_{gpusph}, discussed in section 5.2. All the simulations in AQUA_{gpusph} should be defined by a XML file, in this case filled with the following content:

```
<?xml version="1.0" ?>
<sphInput>
  <Include file="PATH/basic.xml" />
  <Include file="PATH/basic/domain.xml" />
  <Include file="PATH/cfd.xml" />
  <Include file="PATH/cfd/BI.xml" />
  <Include file="PATH/cfd/BINoSlip.xml" />
  <Include file="PATH/cfd/elasticBounce.xml" />
  <Include file="PATH/basic/timing.report.xml" />
  <Include file="PATH/basic/performance.report.xml" />
  <Include file="Settings.xml" />
  <Include file="SPH.xml" />
  <Include file="Time.xml" />
  <Include file="Fluids.xml" />
  <Include file="BCs.xml" />
</sphInput>
```

Where the *PATH* is the path where the AQUA_{gpusph} provided modules are placed. As it can be appreciated, such XML file is only asking AQUA_{gpusph} to load other XML files, which are the modules. The AQUA_{gpusph} modules used in this simulation have the following targets:

1. **basic.xml**: This module is loading a set of common variables, like the position, velocity and acceleration, or the density and density variation rate. This module is also creating a basic structure for the SPH solver, including an Improved Euler time integrator.
2. **basic/domain.xml**: Helper module which allows to set a computational domain, such that if a particle moves out of such domain, it is removed. Loading this module

is strongly recommended, since a single fugitive particle may lead to the simulation crash due to the constantly growing number of link-list cells.

3. `cfid.xml`: Build up the WC-SPH numerical scheme of Eq. (2.39), on top of the structure provided by `basic.xml`.
4. `cfid/BI.xml`: Add the boundary integrals BC to the numerical scheme, as it was described in section 2.2.3.
5. `cfid/BINoSlip.xml`: Add no-slip capabilities to the boundary integrals BC.
6. `cfid/elasticBounce.xml`: Add the elastic bounce BC to the numerical scheme, as it was described in section 2.2.3. It is very common to combine the elastic bounce BC with other BCs in order to grant the unpenetrable walls.
7. `basic/timing.report.xml`: Add a report of the simulation time.
8. `basic/performance.report.xml`: Add a performance report, showing the memory required, the average required time to compute a single time step, the percentage of simulation already performed, and estimated time to finish the simulation.

After loading the AQUAplusph modules, 3 more XML definition files are loaded to set some required values, like the sound speed, c_s , or the computational domain bounds, r_{min} and r_{max} .

The last 2 XML files, namely `Fluids.xml` and `BCs.xml`, are used to specify the files to be read in order to load the initial condition, i.e. the initial values of the particles fields. The fields to become read from each file can be configured as well.

For the sake of simplicity, we are not providing details on how are generated the fluid particles and the boundary area elements.

6.2.3 Results

Numerical results are commonly assessed by plotting the steady-state streamlines and velocity profiles down the center line of the tank. In Fig. 6.2, the velocity field resulting from AQUAplusph, at time instant $t \cdot U/L = 30$, is compared with the streamlines obtained by Chern et al. (2005). As it can be appreciated, a very good agreement exists between both solutions, with a small difference in the shape of the bottom vortexes.

In Fig. 6.3 the velocity profile has been plot, showing again a good agreement.

Regarding the time required to carry out the simulation, it should take about 1.5 hours in a mid-term computational device.

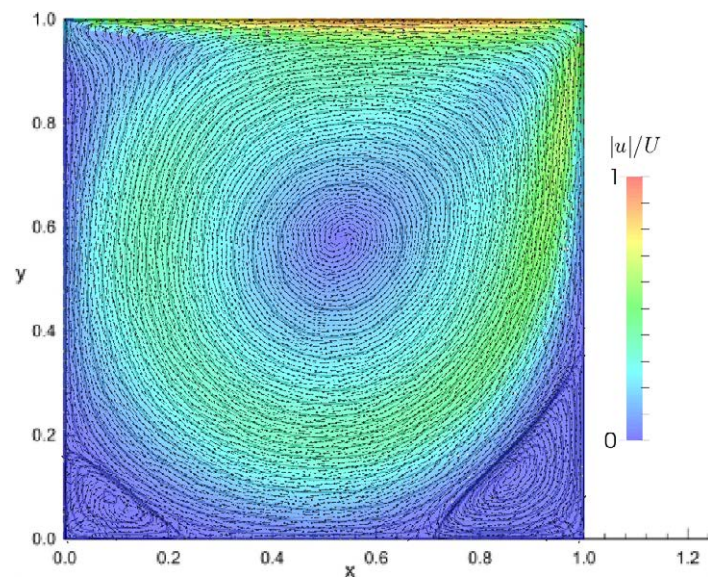


Fig. 6.2 Lid-driven cavity velocity field. Background streamlines obtained by Chern et al. (2005). The arrows denote the velocity direction of each fluid particle, while the colour bar represents the normalized velocity module.

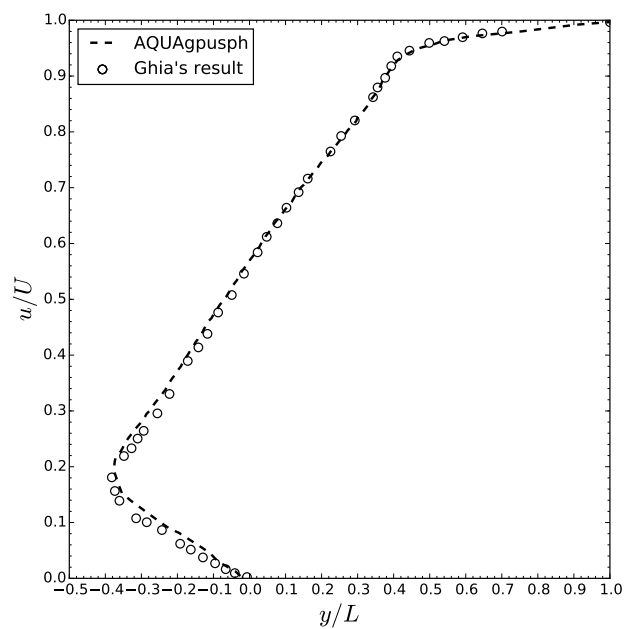


Fig. 6.3 Velocity profile down the center line of the tank.

Table 6.2 Variables selected in the standing wave test case

Variable	Value
\mathbf{g}	(0, -1) [m / s ²]
L	2 [m]
ε	0.1
ρ_0	1 [kg / m ²]
Re	250
μ	4×10^{-3} [Pa · s]
c_s	50 [m / s ²]
Δr	0.01 [m]
$h/\Delta r$	4
Co	0.2

6.3 Standing wave

6.3.1 Case description

The standing wave is also becoming a popular validation test case, because of the great balance between simplicity and complex physical phenomena. More specifically, it is a problem which involves a free-surface, accepting anyway an analytical solution of the kinetic energy decay.

In the work of Souto-Iglesias et al. (2013), the standing wave is defined on top of an horizontally periodic fluid domain of dimensions $L \times H$, with $L = 2H$. In such domain a wave with length $\lambda = L$ and amplitude A is imposed. To this end, the fluid starts from a regular lattice distribution, with $y = 0$ corresponding to the flat free-surface, imposing an initial velocity field according to the following velocity potential:

$$\begin{cases} \phi(x, y, t) &= \phi_0(x, y) \cos(\omega t), \\ \phi_0(x, y) &= -\varepsilon \frac{H |\mathbf{g}| \cosh(k(y+H))}{2\omega \cosh(kH)} \cos\left(k\left(x + \frac{L}{2}\right)\right), \end{cases} \quad (6.1)$$

where $\varepsilon = 2A/H$, k is the wavenumber, and ω is the angular frequency given by the dispersion relation of gravity waves:

$$\omega = |\mathbf{g}| k \tanh(kH). \quad (6.2)$$

The expression (6.1) is valid only for small ε ratio values. Therefore, the velocity field can be computed as $\mathbf{u} = \nabla \phi(x, y, t)$, for an arbitrary time instant $t \geq 0$.

In Table 6.1 all the selected variable values are specified. Hence, the domain has been

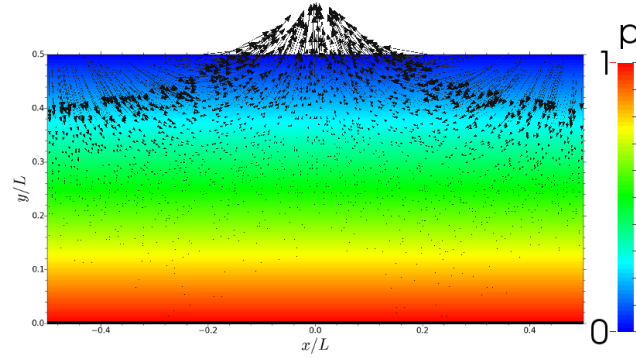


Fig. 6.4 Initial condition of the Standing wave test case. The arrows denote the velocity of the particles

discretized with 100 particles in the vertical direction. The resulting initial condition has been depicted in Fig. 6.4.

With the conditions above, it is possible to find an analytical solution of the kinetic energy (see Lighthill (2001)):

$$\mathcal{E}_k = \varepsilon^2 |g| \frac{LH^2}{32} e^{-4\nu k^2 t} (1 + \cos(2\omega t)). \quad (6.3)$$

6.3.2 Implementation details

Restricting to the implementation context, this problem is characterized by the presence of 2 different symmetry planes, $x = -L/2, L/2$. Therefore this time it would not be enough loading the symmetry BC module, but it should be loaded 2 times, modifying the symmetry plane data right before each tool is executed. To this end, a `Symmetries.xml` file can be loaded from the main XML simulation definition file, with the following content:

```
<?xml version="1.0" ?>
<sphInput>
  <Include file="PATH/cfd/symmetry.xml" prefix="left_" />
  <Include file="PATH/cfd/symmetry.xml" prefix="right_" />

  <Tools>
    <Tool name="left_symmetry_r" action="insert"
          before="left_cfd symmetry mirror"
          type="set_scalar" in="symmetry_r"
          value="-1.0,0.0"/>
    <Tool name="left_symmetry_n" action="insert"
```

```

        before="left_cfd symmetry mirror"
        type="set_scalar" in="symmetry_n"
        value="-1.0,0.0"/>

    <Tool name="right_symmetry_r" action="insert"
        before="right_cfd symmetry mirror"
        type="set_scalar" in="symmetry_r"
        value="1.0,0.0"/>
    <Tool name="right_symmetry_n" action="insert"
        before="right_cfd symmetry mirror"
        type="set_scalar" in="symmetry_n"
        value="1.0,0.0"/>

</Tools>
</sphInput>

```

The file above is loading the symmetry module twice, with a different prefix each time. Such prefix is inserted at the beginning of the name of every tool loaded, such that it can be identified when the mirroring process of each symmetry plane is executed. Indeed, 2 tools are inserted before each mirroring process to conveniently overwrite the infinite plane position and normal.

This is a great example to show the AQUAgpusph modular capabilities, and the consequent flexibility offered.

6.3.3 Results

In Fig. 6.5 the resulting kinetic energy is shown, comparing it with the envelope of the function (6.3). As it can be appreciated, a good agreement between the analytical solution and the one obtained with AQUAgpusph is achieved.

In terms of performance, with the currently selected parameters a simulation of 8 periods would take about 1 hour in a mid-term computational device.

6.4 Taylor-Green vortex

6.4.1 Case description

The Taylor-Green vortex is another test case that has been widely applied in the SPH context (see for instance Xu et al. (2009) and Vacondio et al. (2013)). In this case, an infinite periodic

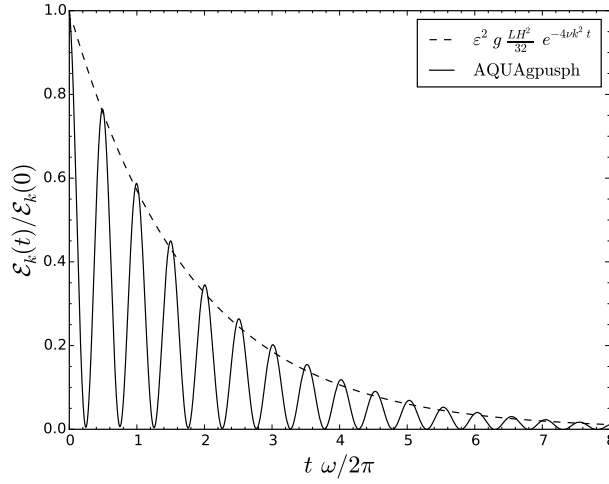


Fig. 6.5 Standing wave kinetic energy decay, compared with the analytical solution described in Lighthill (2001)

domain (in x and y directions) is seeded by co-rotating vortexes of length $\lambda = \pi$. In order to carry out the simulation, a square subdomain of length $L = 2\lambda = 2\pi$ is selected, such that only 4 co-rotating vortex are considered, with periodic boundary conditions. In the initial condition, the particles are distributed in a regular lattice, with the following velocity field:

$$\begin{cases} u_x &= \sin(x) \cos(x), \\ u_y &= -\cos(x) \sin(x), \end{cases} \quad (6.4)$$

which is a smooth function. In Fig. 6.6 the initial condition is shown. Even though the vortex are co-rotating, the system becomes unstable after a while. During the stable stage of the simulation, the velocity and pressure should decay with the following functions:

$$\begin{cases} u_x &= F(t) \sin(x) \cos(x), \\ u_y &= -F(t) \cos(x) \sin(x), \\ p &= F^2(t) \frac{\rho}{4} \cos(2x) \sin(2x). \\ F(t) &= e^{-2\nu t}. \end{cases} \quad (6.5)$$

Therefore, the velocity field should be rescaled each time step to restore the lost energy. Otherwise the instability would not become never excited.

In this specific case, the parameters of Table 6.3 have been chosen, such that the domain is discretized by 800×800 particles. As it happened in the previous test cases, the selected resolution means that an additional dissipation is not required.

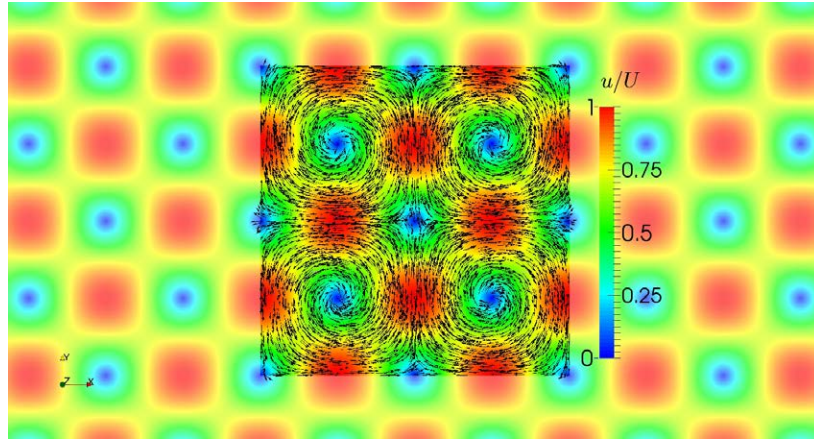


Fig. 6.6 Initial condition of the Taylor-Green vortex test case. The arrows denote the velocity direction of the particles, while the color denote the velocity magnitude

Table 6.3 Variables selected in the Taylor-Green test case

Variable	Value
L	2π [m]
U	1 [m / s]
Re	2000
ρ_0	1 [kg / m ²]
μ	0.52 [Pa · s]
c_s	50 [m / s ²]
p_0	3 [Pa]
Δr	7.8×10^{-3} [m]
$h/\Delta r$	4
Co	0.2

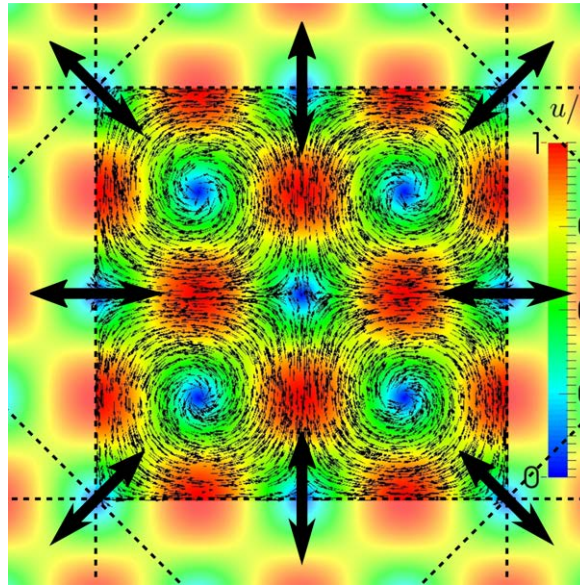


Fig. 6.7 Periodic BCs required in the Taylor-Green test case.

6.4.2 Implementation details

In this test case, the implementation has 2 main peculiarities:

1. Periodic boundary conditions in both directions, x and y
2. Velocity rescaling at each time step

The former condition is quite similar to the symmetry planes, taking into account that in this case portal BCs should be used (see the section 5.6.4). However, in this particular case simultaneous periodic BCs should be considered both at x , and y directions. Even though it should be in principle equivalent to impose 4 portal BCs, actually 8 ones should be used. In Fig. 6.7 the required boundaries have been depicted over the initial condition to illustrate the fact that 4 additional portals are effectively required at the corners.

The other characteristic of this test case, in terms of implementation, is the velocity rescaling process. To carry out this operation, we can load from the main XML definition file a `Rescale.xml` file with the following content:

```
<?xml version="1.0" ?>
<sphInput>
  <Tools>
    <Tool name="velocity rescale" action="insert"
      after="Sort" type="kernel" path="Rescale.cl"/>
  </Tools>
```

</sphInput>

Which is asking for the execution of an OpenCL script, `Rescale.cl`, right after the Link-list and sorting process. The following OpenCL script can be considered:

```
#include "PATH/ types /2D.h"

__kernel void entry(const __global uint* iset ,
                    __global vec* u,
                    __constant float* visc_dyn ,
                    __constant float* refd ,
                    unsigned int N,
                    float dt)
{
    unsigned int i = get_global_id(0);
    if(i >= N)
        return ;

    const float visc_kin = visc_dyn[iset[i]] / refd[iset[i]];
    u[i].XYZ *= exp(2.f * visc_kin * dt);
}
```

Where *PATH* should be replaced by the folder where the OpenCL scripts provided by AQUAgpusph are located. The script above is executed once per particle (the number of threads to be launched can be configured in the tool definition), such that the index *i* is assigned to each particle. After that the kinematic viscosity is computed from the already known fluid properties, and rescaling the velocity field.

This test case demonstrate the powerful combination of the modular design of the tool, and the OpenCL acceleration to perform particle by particle operations.

6.4.3 Results

In Fig. 6.8 the kinetic energy along the simulation is depicted. As it can be appreciated, the flow is quite unstable. It can be appreciated as well that there is a very short period of time where the energy is completely conserved.

Even though several flow instabilities appear along the simulation, the vortex structure is not broken until the time instant $tU/L > 20$. In Fig. 6.9 snapshots of the this part of the simulation, where the vortex structure is broken, are depicted.

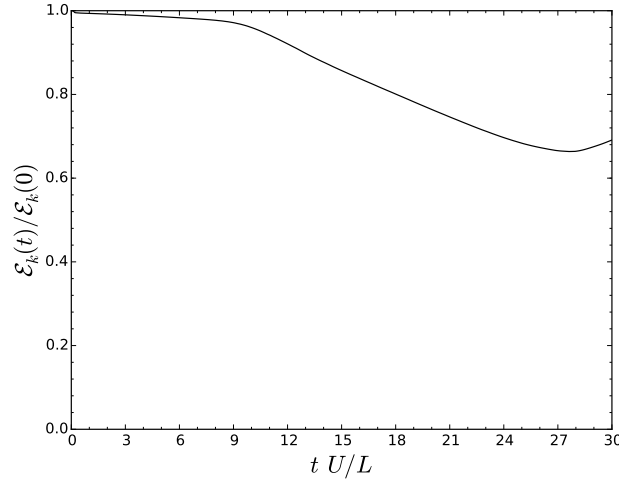


Fig. 6.8 Kinetic energy evolution in the Taylor-Green vortex simulation

It should be remarked that, even though a symmetric distribution of particles have been generated as initial condition, the instability have been effectively excited.

Regarding the performance, this simulation is highly demanding due to the large number of particles and simulation time required. In a mid-term computational device about 1 day may be consumed to carry out this test-case.

6.5 Moving square inside a rectangular box

6.5.1 Case description

Another SPHERIC validation test case, worth described here, is the test number 6, 2D Incompressible flow around a moving square inside a rectangular box.

Actually this case cannot be considered a validation but a verification, since there is not experimental data or analytical solution either. However, this test case becomes really interesting to verify the energy model described in the section 4. To this end, the simulation has been carried out twice, separately applying ghost particles and boundary integrals.

In this case a solid square is introduced into a rectangular box filled with an incompressible fluid, as it is depicted in Fig. 6.10. Both, the outer box and the inner square, are unpenetrable. Rest condition is considered at the begin of the simulation. The outer box is fixed, while the inner square is moving along x direction, with the prescribed motion shown in Fig. 6.11.

The simulation has been performed using the parameters listed in Table 6.4 has been chosen, such that the domain is discretized by 400 particles in the vertical direction.

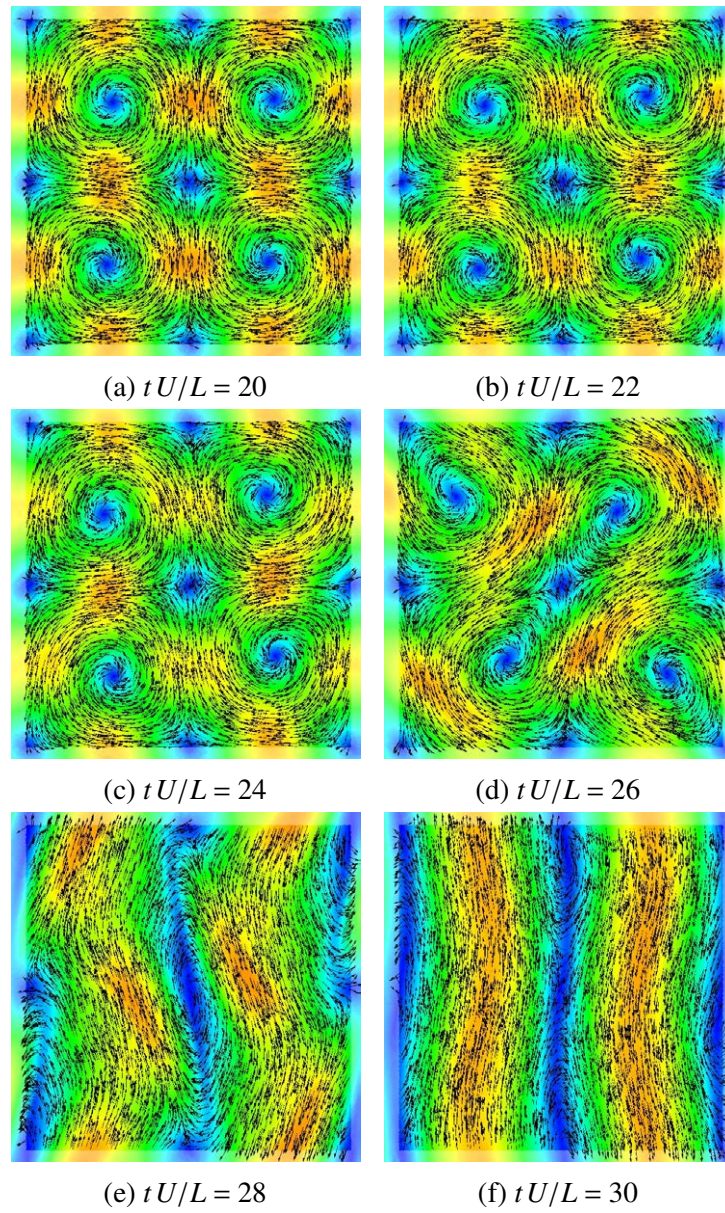


Fig. 6.9 Snapshots of the Taylor-Green vortex simulation

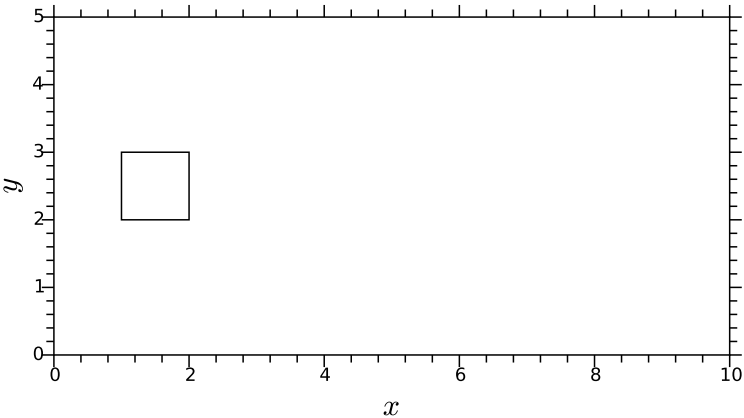


Fig. 6.10 Definition geometry of the moving square inside a rectangular box test case

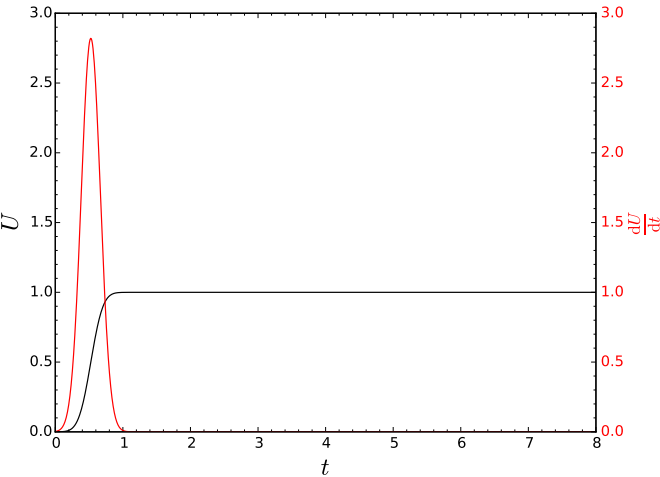


Fig. 6.11 Velocity and acceleration of the inner solid square

Table 6.4 Variables selected in the moving square inside a rectangular box test case

Variable	Value
L	2π [m]
U	1 [m / s]
Re	100
ρ_0	1 [kg / m ²]
μ	10^{-2} [Pa · s]
c_s	50 [m / s ²]
p_0	3 [Pa]
Δr	1.25×10^{-2} [m]
$h/\Delta r$	4
Co	0.2

6.5.2 Implementation details

As this case implies a motion, this is the first example extended with Python. In fact, AQUAplusph provides a module, `motion.xml`, to easily define motions through a Python script. When such module is loaded, a set of variables regarding the position, angle, velocity and acceleration are generated, as well as a Python tool calling a script called `Motion.py`, which should be provided by the user. In this case the following script has been considered:

```

import numpy as np
import aquagplusph as aqua

f = open('Motion_Body.dat', 'r')
lines = f.readlines()
f.close()
T = []
X = []
U = []
DUDT = []
for l in lines[2:]:
    l = l.strip()
    while l.find(' ') != -1:
        l = l.replace(' ', '')
    if l == '':
        continue
    t, dudt, u, x = map(float, l.split(' '))

```

```

        T.append(t)
        X.append(x)
        U.append(u)
        DUDT.append(dudt)
del f, lines

T = np.asarray(T)
X = np.asarray(X)
U = np.asarray(U)
DUDT = np.asarray(DUDT)

F = open('Motion.dat', 'w')

def main():
    t = aqua.get("t")

    r = np.zeros(2, dtype=np.float32)
    r[0] = np.interp(t, T, X)
    u = np.zeros(2, dtype=np.float32)
    u[0] = np.interp(t, T, U)
    dudt = np.zeros(2, dtype=np.float32)
    dudt[0] = np.interp(t, T, DUDT)

    aqua.set("motion_r", r)
    aqua.set("motion_drdt", u)
    aqua.set("motion_ddrddt", dudt)

    F.write('{}\t{}\t{}\t{}\n'.format(
        t, r[0], u[0], dudt[0]))
    F.flush()

    return True

```

Indeed, when the script is loaded, it is reading the prescribed motion and opening an output file. Later, in the `main` function, which is called each time step, it is getting the time step, interpolating the motion variables from the already known data. Of course, in this case angular motion variables are not required.

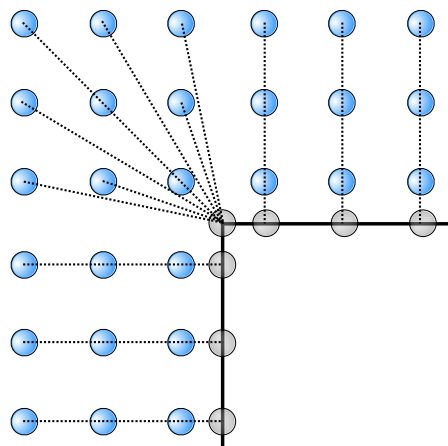


Fig. 6.12 Ghost particles distribution close to the convex (from the point of view of the ghost particles) corner

At this point, it is mandatory to discuss about the ghost particles distribution. As it was discussed in section 5.6.2, the ghost particles formulation used in AQUA_{gpusph}, described by Marrone et al. (2011a) and Marrone et al. (2011b). In such formulation, each ghost particle is associated to a boundary element, which acts as pivoting point to perform the velocity, pressure and density fields interpolation from the fluid data. Distributing the ghost particles in planar walls is quite simple. However, close to the corners such distribution could become significantly more complex. In case of the outer box, characterized by convex angles (from the side of the ghost particles), we can proceed in a similar way than we done for the portals in the previous example, i.e. we can create a lattice of ghost particles sharing a common area element at the corner. This workaround is depicted in Fig. 6.12.

Conversely, in the case of concave (from the point of view of the ghost particles) corners, like the ones of the inner square, a different approach has to be practised. Some models to deal with corners, in the ghost particles context, were introduced by Colagrossi et al. (2007). Later, Marrone et al. (2011a) revisited the problem in order to provide a more robust model to perform the mirroring. Such new model was applied to a flow around a rectangular body in Marrone et al. (2013). Herein a simpler approach is used, based on duplicating the Ghost particles close to the corner, halving their mass. Then each particle of the pair will pivot against a different area element, or in other words, one particle of the pair will be horizontally mirrored, while the other one will be vertically mirrored. This process is illustrated in Fig. 6.13.

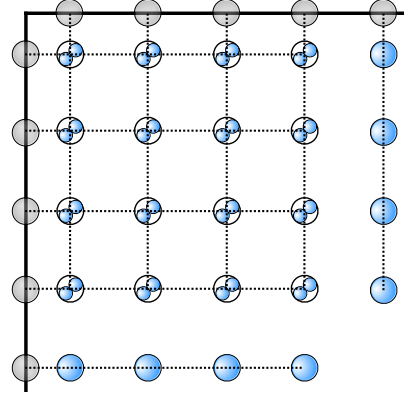
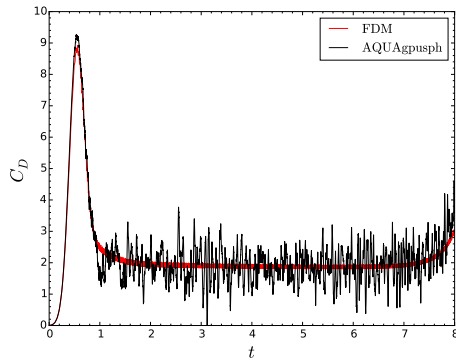
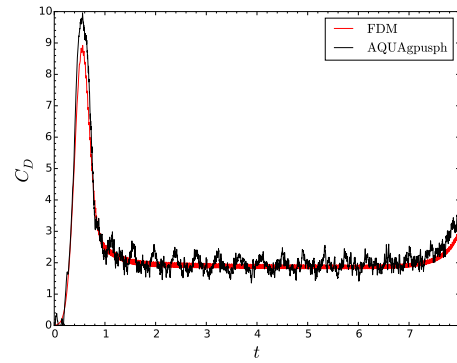


Fig. 6.13 Ghost particles distribution close to the concave (from the point of view of the ghost particles) corner



(a) AQUAagpusph, Ghost Particles



(b) AQUAagpusph, Boundary Integrals

Fig. 6.14 Drag coefficient computed with AQUAagpusph, and compared with the Finite Difference Method.

6.5.3 Results

The main result of this test case is the drag force. Since there is not available experiments or analytical solution either, the simulation performed applying a Navier-Stokes Finite Difference solver, with a Level-Set algorithm to capture fluid/solid interfaces, is used as reference result. In Fig. 6.14 the drag coefficient computed with AQUAagpusph is compared with the reference solution, both when the Ghost Particles BC is applied (Fig. 6.14a), and when Boundary Integrals are considered instead (Fig. 6.14b). A good agreement can be appreciated with both BCs. It can be appreciated the effect of the acoustic waves, mainly dominated by the harmonic of period $T = L/c_s = 0.2[s]$. In this sense it can be noticed that

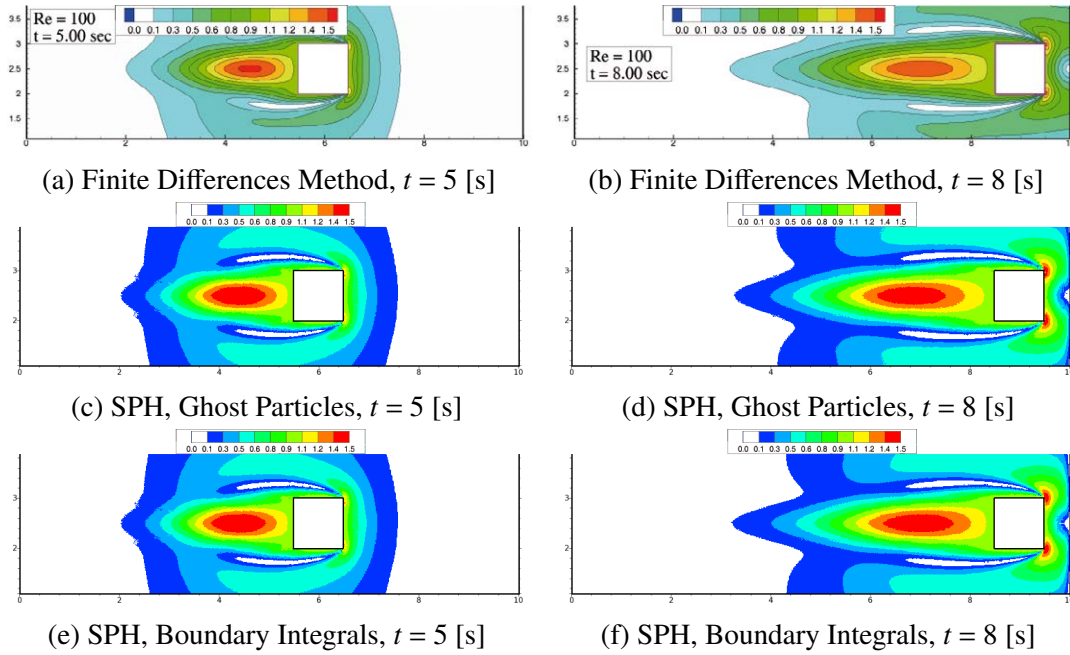


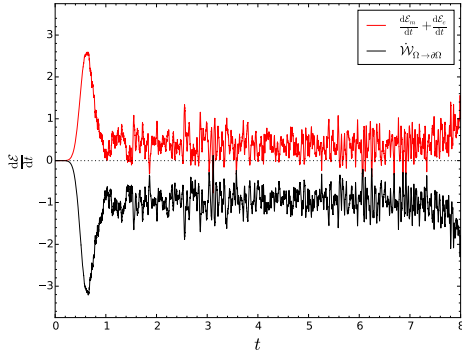
Fig. 6.15 Snapshots of the moving square inside a fixed rectangular box simulation

Ghost Particles are more affected by such acoustic effects, probably due to the poor approach practised to model the corners of the inner square, described above.

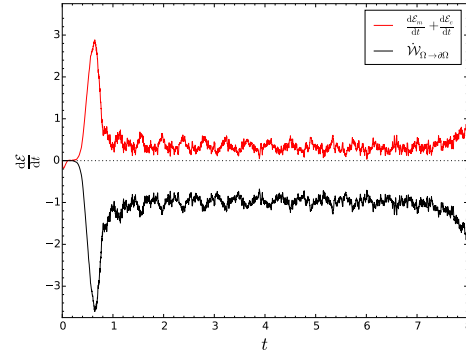
In Fig. 6.15 snapshots of the simulations, including the reference case, are shown. There is a very good agreement between the solutions resulting from SPH, although the velocity profile is slightly different to the one of the reference solution.

Regardless the comparison with the reference solution, we are interested in the verification of the energy components decomposition carried out in section 4, and more specifically in the extra dissipation terms of Eq. (4.7). In Fig. 6.16 the Mechanical and compressibility power of the fluid are depicted, as well as the effective mechanical power assimilated by the solid square. Even though the acoustic waves affecting the Ghost Particles are clearly reflected in the energy, in general both boundary conditions have a similar behavior.

The difference between the mechanical work assimilated by the solid square, and the reversible energy accumulated by the fluid, is the dissipated energy. In Fig. 6.17 the components of such dissipated power are shown. Both the viscous dissipation and the dissipation due to the δ -SPH term are always positive, such that they are consuming mechanical and compressibility energies respectively. However, as it was discussed in section 4.3, in general the sign of the term $\frac{d\mathcal{E}_{\partial\Omega}}{dt} - \dot{\mathcal{W}}_{\Omega \rightarrow \partial\Omega}$ can not be defined *a priori*. Effectively, such term is pumping energy into the system, an undesirable feature. Moreover, while it turns negative for a small lapse of time when the Boundary Integrals are considered, for the Ghost particles it is pumping energy into the system during a large part of the simulation. This issue is

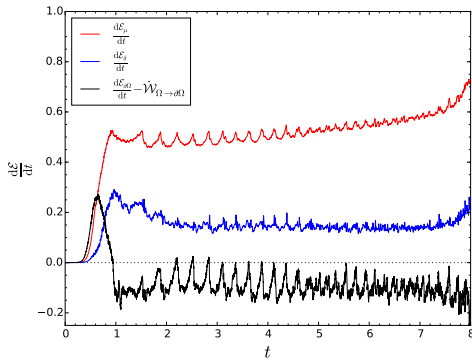


(a) AQUAplusph, Ghost Particles

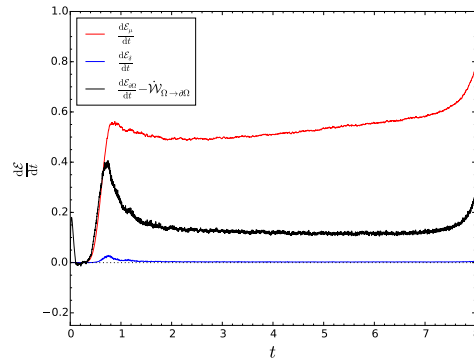


(b) AQUAplusph, Boundary Integrals

Fig. 6.16 Reversible energy variation rate of the fluid, and effective work done by the fluid on the solid

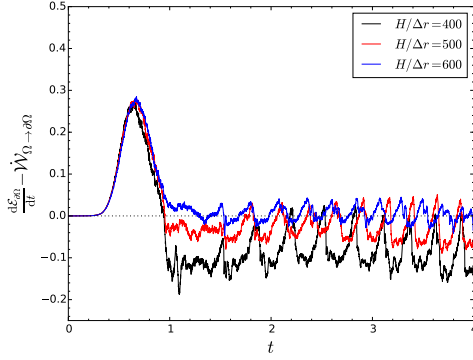


(a) AQUAplusph, Ghost Particles

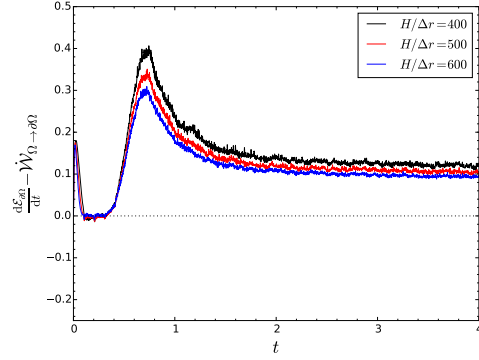


(b) AQUAplusph, Boundary Integrals

Fig. 6.17 Extra power terms



(a) AQUAplusph, Ghost Particles



(b) AQUAplusph, Boundary Integrals

Fig. 6.18 Convergence numerical analysis of the boundary interaction extra energy term

probably caused by the poor modelization of the corners of the moving square, described in previous subsection. Comparing the figures, it is not unreasonable to suggest that δ -SPH is significantly dissipating the artificially pumped energy by the solids.

It should be remarked as well that, while the δ -SPH term dissipation is in fact very small, the extra energy term of the boundaries is of the same order of magnitude of the viscous dissipation, and therefore non-negligible.

A further analysis on the extra energy term due to the interactions by the solid boundaries, $\frac{dE_{\partial\Omega}}{dt} - \dot{W}_{\Omega \rightarrow \partial\Omega}$, has been analyzed by Cercos-Pita et al. (2016a), considering just the Ghost Particles methodology. In such work the consistency of the model is demonstrated, that is, the term vanishes at the continuum. In Fig. 6.18 this term is depicted for different resolutions. Indeed, it converges to zero for the larger resolutions, both when Ghost Particles and Boundary Integrals are considered.

This simulation does not demands large computational resources, taking less than 1 hour in a mid-term computational device.

Chapter 7

Practical applications

Even though AQUAgpusph is a code designed by researchers for researchers, it has been successfully applied to industrial applications. Herein, 2 remarkable industrial applications are described.

7.1 Simulation of Earthquake Sloshing Loads in a Nuclear Reactor

7.1.1 General

Since “Chicago Pile-1”, the world’s first nuclear reactor, almost all the nuclear energy facilities have been invariably designed on top of the thermal-neutron fission reactor technology. However, a different approach could be eventually considered: the fast-neutron reactors, characterized by the lack of a neutron moderator to slow-down the neutrons. The main benefit of such fast reactors is the boosted resources efficiency (about 200 times more efficient than a classic thermal reactor), even allowing to breed new fuel with the same facilities (see Waltar and Reynolds (1981)), which may dramatically enhance the sustainability of the nuclear power in general. Of course, such benefit is leading the generation of research programs to develop this technology, some of them including experimental reactors (see for instance Cochran et al. (2010)).

Unfortunately, in order to operate fast reactors, some sort of exotic coolant derived from heavy atoms are required. Along this line, the most popular coolant is the liquid Sodium. However, due to the high reactivity of the liquid Sodium with air and water, much less reactive liquid lead based coolants are becoming considered as well (see among others Gromov et al. (1997) and Alemberti et al. (2011b)).

On the other hand, after the violent earthquakes in Japan, which caused the collapse of Kashiwazaki-Kariwa (2007) and Fukushima (2011) nuclear plants, the nuclear engineering community has significantly increased the research efforts dedicated to the seismic initiated events, in which the hottest topic is clearly the power plants isolation (the works of Frano and Forasassi (2012) and Qin et al. (2014) are good examples of that). The isolators are designed to dramatically reduce the accelerations -i.e. the loads- induced by the seismic event in the reactor.

Unfortunately, the extreme reduction of high frequency motions results in large amplitude and period oscillations, which may eventually excite severe sloshing phenomena and flow impacts. Such fluid-structure interaction has been numerically investigated in the past by Chellapandi et al. (2012), for a liquid Sodium cooled vessel. However, the loads induced by the fluid may become significantly bigger in the case of molten lead coolant system. Indeed, Molten lead, at a temperature of 753K, have the following density and dynamic viscosity:

$$\begin{aligned}\rho_0 &= 1.047 \times 10^4 \frac{\text{kg}}{\text{m}^3}, \\ \mu &= 1.882 \times 10^{-3} \text{Pa} \cdot \text{s},\end{aligned}\tag{7.1}$$

becoming in fact approximately 10 times denser than water or liquid Sodium, with a 10 times smaller kinematic viscosity. In this practical application a numerical investigation of the sloshing phenomena in a nuclear reactor vessel, considering molten lead, is carried out.

7.1.2 Case description

A simulation of a Lead-cooled Fast Nuclear Reactor (hereinafter LFR) is described herein. In Fig. 7.1, the complete ELSY reactor geometry is shown (see Alemberti et al. (2011b) and Alemberti et al. (2011a)), which is in fact one of the geometries considered in the SILER EU project (see Forni and De Grandis (2012)). On top of such geometry, a 3D simplified model has been generated, depicted in Fig. 7.2. The following elements are highlighted in that figure:

1. Grey: Structural vessel
2. Green: Cylindrical reactor kernel.
3. Blue: Active heat exchangers.
4. Red: Passive heat exchangers (for safety reasons).

The cooled lead flows in the kernel from the bottom of the cylinder, becoming heated by the nuclear reaction. Then the heated lead is driven to the active heat exchangers (blue

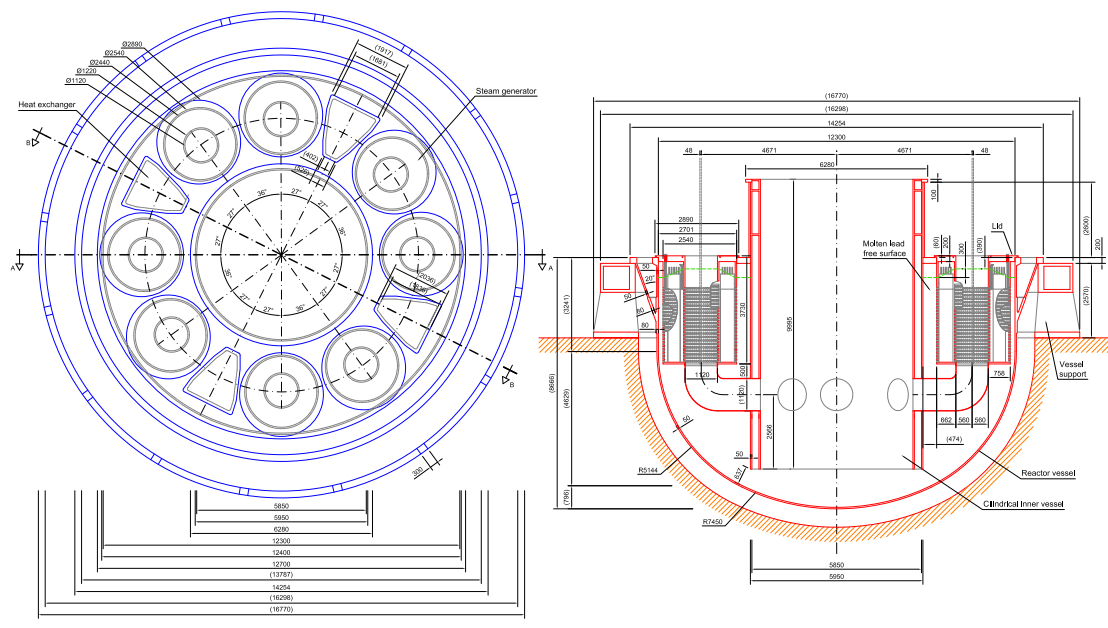


Fig. 7.1 A plant view of the SILER LFR (left) and a front view showing the lead level with a dashed line (right). Dimensions are in millimetres.

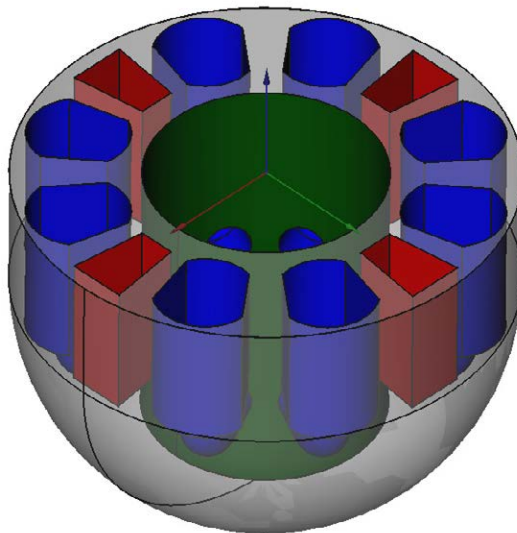


Fig. 7.2 Simplified geometry of the LFR.

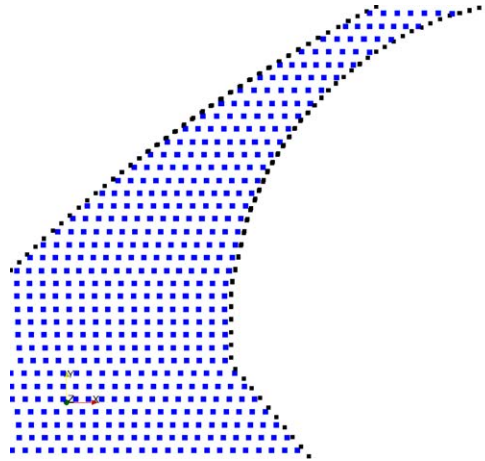


Fig. 7.3 XY plane section of the generated particles from the geometry in Fig. 7.2. Black: Boundary integrals area elements, discussed in section 5.6.2. Blue: Fluid particles.

structures around the kernel) where it is cooled again, such that the heat given by the lead is used later to produce energy. The four passive heat exchangers, separated 90 degrees, are included in the reactor for safety reasons. Herein, the fluid dynamics inside the kernel and the heat exchangers during the earthquake is neglected, and therefore they will be considered as solid impenetrable structural elements.

In order to setup the initial condition of the problem, a surface mesh of the model shown in Fig. 7.2 is generated. Within AQUAgusph package, a tool to generate a set of particles from a surface mesh is provided. In Fig. 7.3 a detail of the generated particles in an arbitrary XY plane section is shown.

Regarding the motions induced by the earthquake on the structure, it has been precomputed, becoming in fact an input to the computation. Its time history and a spectrum of its z component are depicted in Fig. 7.4. It can be appreciated that, while in the x and y directions the motion is widely dominated by long period functions, in the z direction some high frequency components remain. They are illustrated in the spectrum of Fig. 7.4b with some small components in frequencies close to 8 Hz.

The motions are associated to a extreme earthquake event affecting a facility equipped with dampers. Effectively, as it has been commented during the subsection 7.1.1, the dampers are efficiently reducing the accelerations transmitted to the structure, creating long period motions. This is the most relevant situation from the point of view of the flow dynamics.

To capture the flow impacts, numerical pressure probes are placed along the reactor top slab in a radial distribution, at $r = 2.975, 4.3284, 5.6819$ m, separating them 15 degrees.

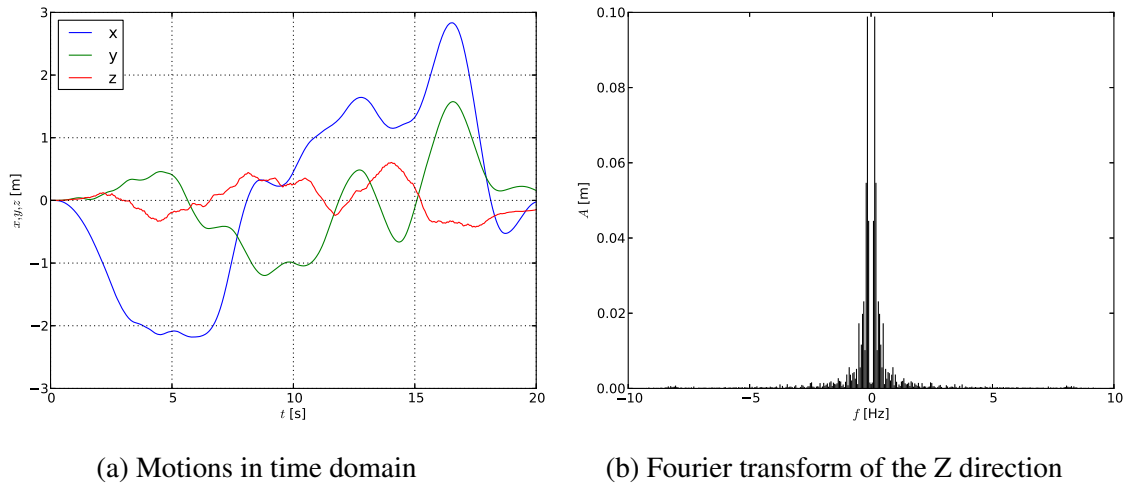


Fig. 7.4 Motions induced by the earthquake on the LFR structure

7.1.3 Results

During this simulation the δ -SPH formulation has been applied. A set of 3.6×10^6 particles were used in the simulation ($\Delta r = 0.05$ m), with a kernel height ratio of $h/\Delta r = 4$. In this case, the considered sound speed was $c_s = 100$ m/s, imposing a Courant factor of $C_f = 0.25$, resulting in a time step of $\Delta t = 3.75 \times 10^{-4}$ s.

In Fig. 7.5, a snapshot of the simulation is shown for the time instant $t = 19$ s, where a severe flow impact was detected at the LFR top slab.

In Fig. 7.6, the pressure records in the sensors for the 45 and 225 degree angles, for the last 5 seconds of the simulation, are depicted. It can be appreciated that significant sloshing impact phenomena are happening along the reactor top slab which might be considered in the reactor structural design.

This simulation is characterised by a large number of particles and neighbours, as well as a small time step, becoming significantly expensive in computational terms, requiring time lapses of the order of days to carry out a single simulation.

7.2 Coupling free surface tanks with ship motions codes

7.2.1 General

It is well known in Naval architecture the effect on the static stability caused by onboard liquid partially filled tanks (see for instance Lewis (1989)), at a point that nowadays it is a routine task.

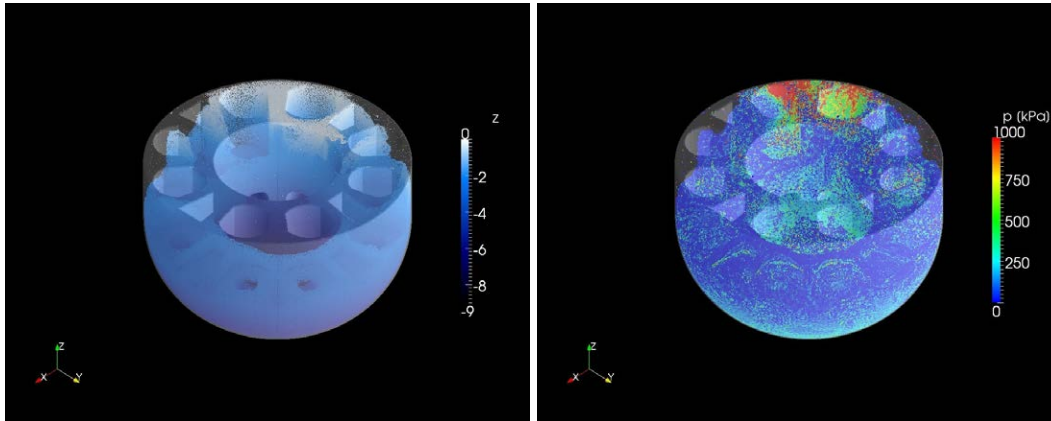


Fig. 7.5 A simulation snapshot at time instant $t = 19$ s. Left: particles z coordinate relative to the LFR roof, in meters. Right: Pressure of the same particles.

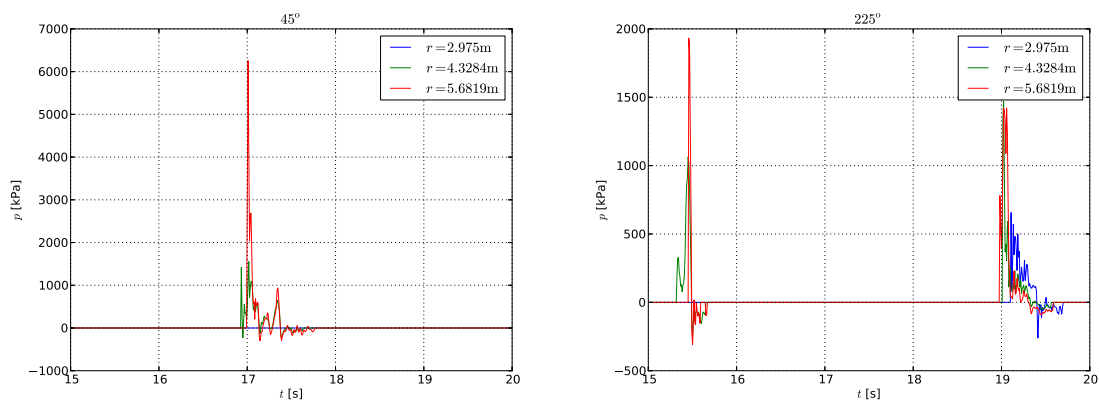


Fig. 7.6 Pressure measured in the reactor roof at 45 degrees (left) and 225 degrees (right).

Unfortunately, the state of the art regarding such effects in dynamic situations has a degree of development several times lower, becoming in fact a cutting-edge technology.

However, the normal operating condition of ships and offshore floating structures is almost invariably characterised by the presence of liquids, which sometimes may represent a significant part a large percentage of the deadweight. This is the case of tankers or LNG ships, among others. Along this line, sometimes the naval structures may be equipped with Anti-Rolling Tanks (ART). Such tanks are usually representing a small extra load on the ship, significantly affecting anyway the dynamic stability, analogously to the Tuned Liquid Dampers (TLD) in civil engineering (see Bulian et al. (2010)).

Due to the complexity of the physical phenomena considered herein, the first approaches practiced by researchers in this regards are clearly oriented towards the coupling of independent solvers for the external and internal hydrodynamics. The simplest developments we can consider are the works of Malenica et al. (2003) and Kim and Shin (2008), where both, the internal flow and the ship motions, are approximated by using fully linear solvers. As a consequence, the whole system can be formulated in frequency domain, dramatically reducing the computational costs. However, such approaches can be only applied in mild-sea conditions.

When more severe sea conditions should be considered, fully linear seakeeping solvers becomes insufficient, and must be replaced by more complex methodologies. This can be the case of the analysis of parametric roll situation, pure loss of stability, surf riding and broaching or large amplitude roll in dead-ship condition. Several developments along this line have been discussed in the literature (see for instance Francescutto et al. (1999), Spanos and Papanikolaou (2001))

Much more interesting, in the context of the present work, is the case where severe sloshing phenomena is considered, invalidating the possibility of using linear approaches for the internal flow solver. The first approach where non-linearities are introduced in the internal flow, keeping a linear solver for the ship motions computation, was described by Kim et al. (2007). A similar approach was practiced later by Zhao et al. (2014). However, the first work where a fully non-linear approach for the internal flow computation is applied, becoming able to treat non-single valued free surface shapes, is attributed to Bunnik and Veldman (2010). Later some authors have investigated the possibility of using non-linear solvers both for the ship motions and the internal flow computation (see Holden and Fossen (2012) and Mitra et al. (2012)).

Herein the coupling of AQUAgusph with two different seakeeping solvers, SeaFEM (A linear seakeeping solver described in Serván-Camas and García-Espinosa (2013)) and SHIXDOF (A non-linear seakeeping solver described in Bulian et al. (2012)) is documented.

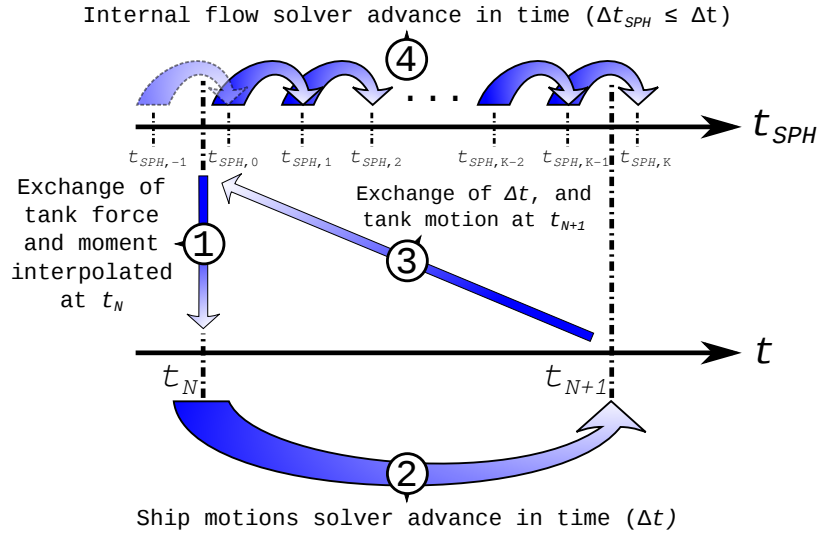


Fig. 7.7 Implemented time stepping algorithm, based on a CSS purely explicit numerical scheme. The indices enclosed by circles denote the order of processes

7.2.2 Co-simulation strategy

In this specific case, where the solvers may potentially run in different computers, facilities, operating systems, or even architectures, a co-simulation strategy should be considered (see Breuer et al. (2012) and Gomes et al. (2011)). Such strategy is also ideally suited for all cases where it is provided a limited or null access to the source code of one or more involved programs, like it is the case.

Actually, the same co-simulation strategy has been followed to couple AQUAgpusph with the 2 different seakeeping solvers. Moreover, the implementation in the AQUAgpusph side has not significant differences.

Almost every co-simulation strategy can be split in the 3 main parts discussed below.

The numerical scheme, or coupling

In the co-simulation methodologies, also known as partitioned treatment, each subsystem is isolated, modelling the interactions as forcing effects that are communicated between the individual components. This type of co-simulation strategy is usually applied in case of so-called loosely coupled systems, like those where the interaction between two subsystems is achieved by communicating the interface force and displacement, as it is the case in this study. Along this line, the Conventional Serial Staggered (CSS) purely explicit numerical scheme (see Farhat et al. (2006)) depicted in Fig. 7.7 has been used. Such integration scheme can be summarized as follows:

1. AQUA_{gpusph} send the interpolated force and moment to the seakeeping code.
2. SHIXDOF advances in time by Δt while AQUA_{gpusph} is waiting.
3. SHIXDOF sends the time step Δt and the new motions at time t_{N+1} to AQUA_{gpusph}.
4. Finally, AQUA_{gpusph} advances in time Δt as well, but using a different time step, Δt_{SPH} . During this process, the seakeeping solver is waiting at $t_{N+1} = t_N + \Delta t$ for the force and moment, in order to start the next time step computation.

The communication layer

Once the numerical scheme has been defined, a two-way channel between both subsystems should be provided in order to exchange the necessary information. In this case, a TCP/IP network communication model has been selected, which is better suited to exploit all the benefits of the partitioned treatment of the problem. Indeed, an exchange of data based on network communication is much less intrusive and potentially more scalable.

The implementation

In order to close the system, both the numerical scheme and the communication channel should be joined in a common implementation. In Fig. 7.8 the co-simulation implementation is schematically shown. The working flow of each simulation can be summarized as follows:

1. Initialization stage. This phase consists of the following tasks:
 - (a) The daemon detects that there is not any simulation running, taking the control and setting up a server waiting for a seakeeping client.
 - (b) When the seakeeping solver is launched, it connects with the daemon and sends the simulation data (geometry, fluid properties, filling level, target number of particles, etc.)
 - (c) While SHIXDOF is waiting, the daemon setups the simulation.
 - (d) The daemon sends back a small report about the simulation, including the actual number of particles resulting from the packing. Such information is used as a flag to inform that the daemon work has finished and the connection should be closed.
2. After the system has been initialized, the control of the server is assigned to AQUA_{gpusph}, which opens a new port through a Python extension.

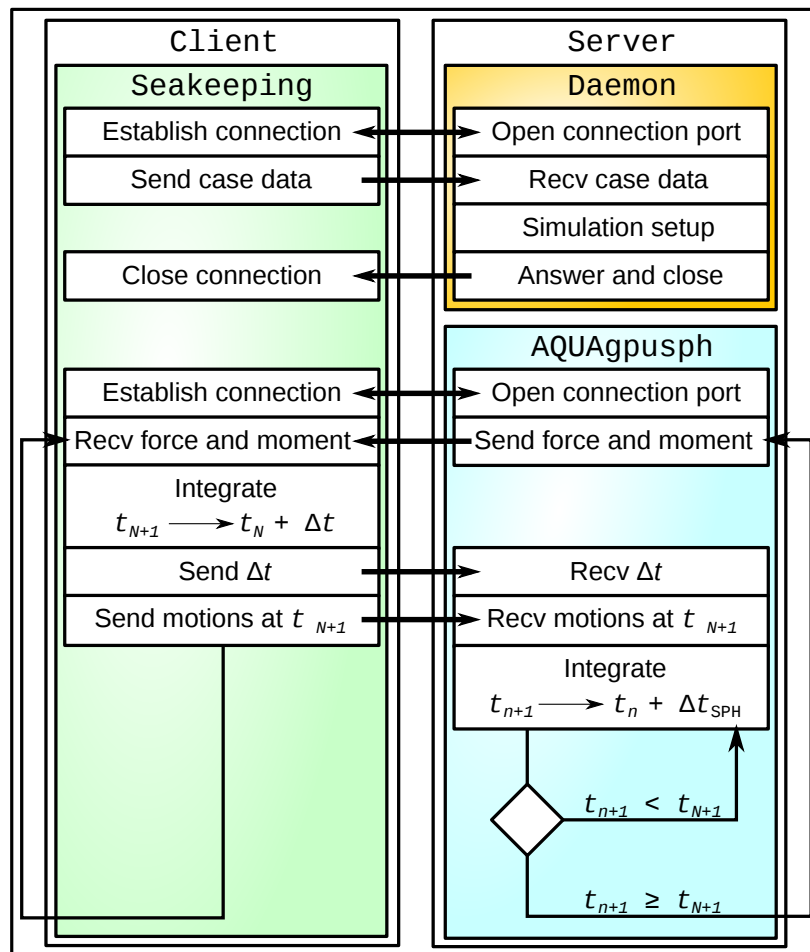


Fig. 7.8 Implementation of the co-simulation strategy

Table 7.1 Main features of the modified S175 hull

Length between perpendiculars	47.6 m
Breadth	13.7 m
Draft	4.0 m
Displacement	1498 tons
Z_G	-0.25 m
Radius of roll gyration	6.26 m

3. Eventually, the actual coupled simulation can start by carrying out the time stepping loop described above.

In AQUAgpusph the coupling implementation described herein can be carried out with a Python extension of 684 lines, from which 309 lines of code are dedicated to the numerical scheme implementation, while the remaining 375 lines are dedicated to the network communication server instance. 404 lines of code should be added to implement the daemon, composed by 297 lines of Python, and 107 lines of Bash. Therefore, a total of 1088 lines of code have been required for this specific practical application.

7.2.3 Linear ship motions model (SeaFEM)

The first seakeeping solver considered is SeaFEM, a potential FEM time domain suite of tools for seakeeping simulation (see Serván-Camas and García-Espinosa (2013)). In order to verify the tool, the same numerical simulations performed by Kim et al. (2007) can be carried out. The work of Kim et al. (2007) is based on the experimental campaign described by Bai and Rhee (1987), regarding a supply vessel equipped with an ART. However, Kim et al. (2007) is not using the same hull geometry considered by Bai and Rhee (1987), but a S175 supply ship modified to fit to the dimensions of the one considered in the experiments. The results of the simulations shown similar trends between the numerical and the experimental results.

The considered ship hull, as well as the equipped ART considered in this work are depicted in Fig. 7.9. In Tables 7 and 7 the main features of both, the hull and the tank, are related. The ART is filled with fresh water, $\rho_0 = 998 \text{ kg / m}^3$.

Two series of simulations with monochromatic waves have been carried out, one without considering the ART, i.e. running SeaFEM alone, and one equipped with the ART, requiring therefore the coupling with AQUAgpusph. In Fig. 7.10 the obtained Roll RAOs, normalized with respect to the maximum wave slope, are compared with the experimental ones. A good

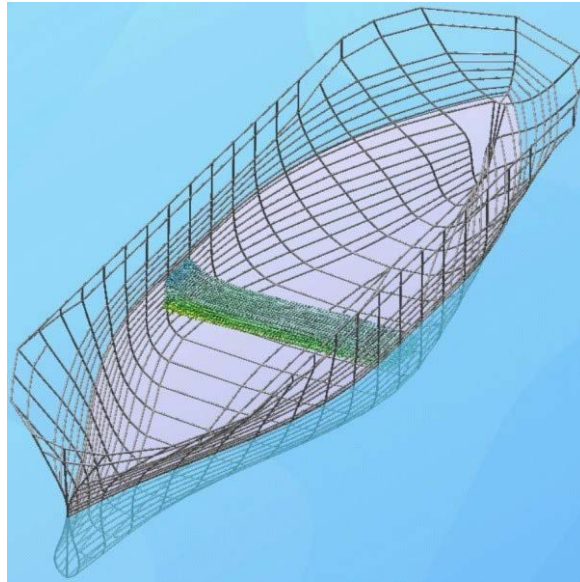


Fig. 7.9 Modified S175 hull, equipped with an ART

Table 7.2 Main features of the ART for the modified S175 hull

Length	2.8 m
Breadth	13.7 m
Height	2.4 m
Filling level	50%
X_T (mid tank)	-0.73 m
Z_T (base tank)	2.14 m

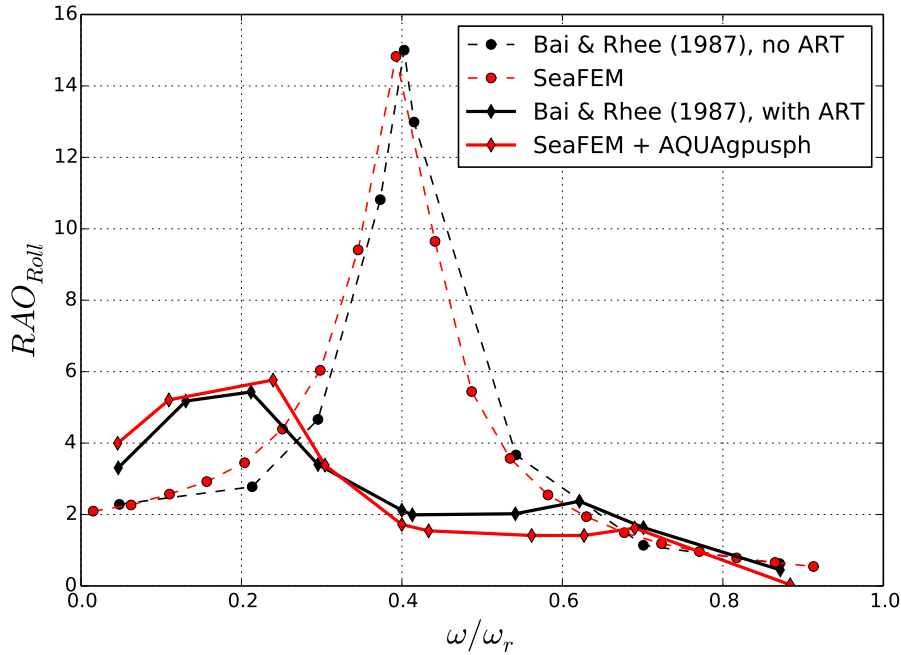


Fig. 7.10 Experimental vs present work. Roll RAOs for monochromatic wave tests.

degree of agreement can be appreciated, even when the ART is considered, consistently with the model calibration performed to achieve a similar behavior to the supply vessel used by Bai and Rhee (1987). It can be appreciated the large effect of the ART in the ship roll motions. During the next subsection AQUAgnusph is coupled with a non-linear seakeeping code, which will be useful to numerically investigate the effect of the sea state on the effectiveness of an ART.

7.2.4 Non-linear ship motions model (SHIXDOF)

During the previous subsection, the ship motions resulting for a supply vessel equipped with an ART, have been analyzed for mild sea conditions, comparing them with the experimental results. However, herein we focus on investigate the effectiveness of the ART when more severe sea conditions are considered. As it has been mentioned above, to this end linear seakeeping codes cannot be considered anymore. Indeed, AQUAgnusph has been coupled with SHIXDOF (see Bulian et al. (2012)), a fully non-linear ship motions computation tool.

For the sake of reproducibility, a well known series-60 hull will be considered this time, with the main characteristics listed in Table 7.3. As in the case of the coupling with SeaFEM, an ART is equipped, with the main features described in Table 7.4.

Table 7.3 Main features of the modified series-60 hull

Length between perpendiculars	162.5 m
Breadth	25.0 m
Draft	10.0 m
Displacement	32500 tons
Z_G	8.59 m
Radius of roll gyration	9.1 m

Table 7.4 Main features of the ART for the series-60 hull

Length	10.0 m
Breadth	25.0 m
Height	5.0 m
Filling level	21.6%
X_T (mid tank)	0.0 m
Z_T (base tank)	22.0 m

In Fig. 7.11 the maximum roll angles obtained for different regular beam waves, with and without the ART, are depicted. It can be appreciated the non-linear behavior of the system in the dependency of the normalized roll angle depends with the considered wave steepness. Such phenomena can not be captured by linear approaches.

Regarding the effect of the ART in the global system, it should be remarked the large effect of the tank, which represents a 0.4% of the total ship displacement, when moderate wave steepnesses are considered. It is also mandatory to point out the quick change of behavior when the largest wave steepness, $s_W = 1/50$, is considered. In such severe sea condition the efficiency of the ART is critically affected. The system shows as well the classical double peak behavior when ART devices are applied.

Regarding the flow inside the tank, in Table 7.5 several snapshots of a simulations subset are depicted. The snapshots are taken for the tank in its maximum, minimum, and zero roll angles, during the last wave period of the simulation. Consistently with the Fig. 7.11, a significant difference can be clearly appreciated in the behavior of the fluid inside the tank for different wave steepness.

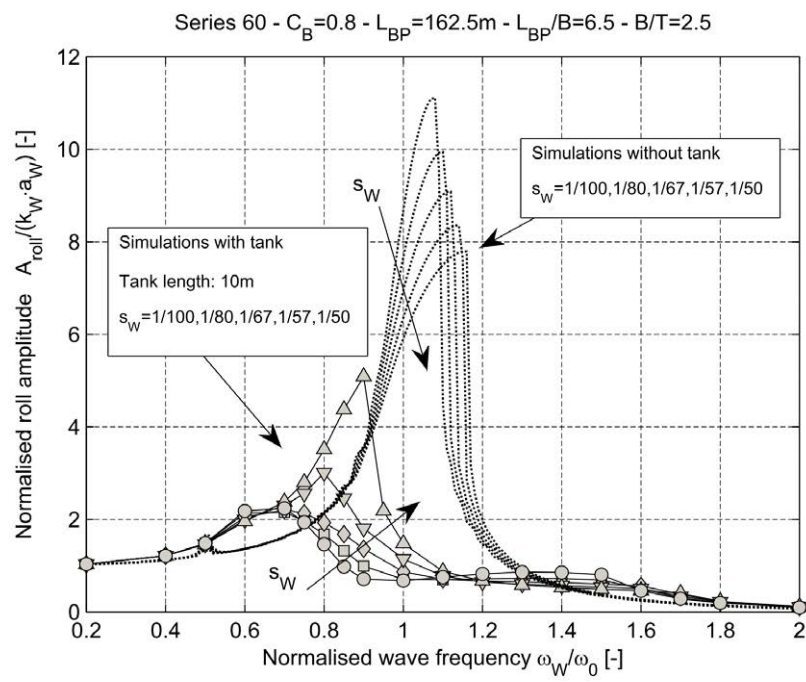
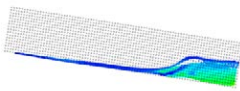
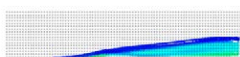
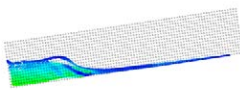
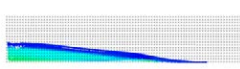
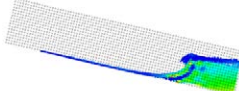
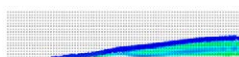
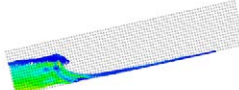
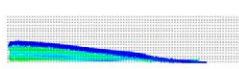
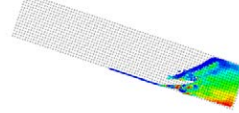
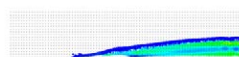
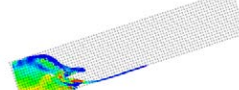
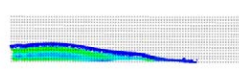

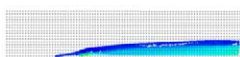

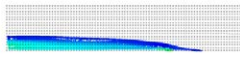

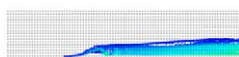

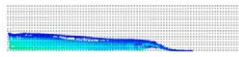
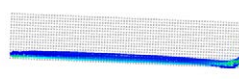
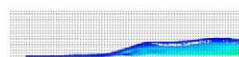

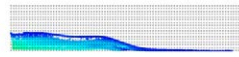


Fig. 7.11 Normalised roll response in regular beam waves. Vessel without tank and vessel with tank, for different forcing wave steepnesses.

Table 7.5 Snapshots of the flow inside the ART, corresponding to a subset of the series-60 simulations

	$\omega/\omega_0 = 0.7$	$\omega/\omega_0 = 0.8$	$\omega/\omega_0 = 0.9$
$s_W = 1/50$	   	   	   
$s_W = 1/80$	   	   	   

Chapter 8

Conclusions and future work

In the present Ph.D. thesis an analysis of the state of the art of the SPH methodology has been carried out. Such analysis has not been restricted to the numerical technique itself, but also to the ecosystem of researchers and industry supporting it. In such analysis it has been pointed out that SPH has received a marginal attention during its first 20 years, situation that has dramatically changed in recent years, with a significant growing rate of the related publications. A natural conclusion arising from this fact is that a large development of the methodology is still required.

During the review of the state of the art, we have also demonstrated that several free/open source codes already exist for industrial applications. Conversely, there is a lack of software oriented to the researching tasks, which should be clearly characterised by the capacity of modifying and extending the software with a minimum amount of effort. Of course, such lack of software, combined with a growing research community, may result in a unsustainable situation. As part of this thesis, the software AQUAgnusph has been released as free software, licensed under GPL version 3 terms. Due to the acceleration -and optimization- with OpenCL, AQUAgnusph is competitive in performance terms, becoming as well modular and extensible. Indeed, AQUAgnusph is called to fill the gap between the low performances codes designed to carry out small tests of new formulations, and the high performance codes clearly designed to perform industrial applications.

In order to demonstrate the capabilities of AQUAgnusph to aid the research works, contributing at the same time to the SPH methodology development, an analysis of the diffusive terms in the mass conservation equation, usually called δ -SPH model, has been carried out, as well as an analysis of the energy conservation when the interaction with solid boundaries is considered.

In the former analysis, several already existing formulations have been compared. The usually applied δ -SPH model is based in tuning parameters and imposing an additional

Courant condition. Conversely, a model which is not requiring neither tuning parameters nor additional Courant conditions has been developed. The eventual relations between the diffusive models have been investigated as well. From such study, it can be concluded that the Riemann solver based SPH models (R-SPH) can be analysed in the same context of the δ -SPH models. In fact, the mass variation rate used in R-SPH is equivalent to the mass conserving equation used in δ -SPH, where a diffusive terms is appended.

From the energy analysis, it can be concluded that the δ -SPH model implies an extra energy dissipation, while some other extra energy terms are arising from the interaction with solid boundaries, which unfortunately can not be classified as dissipation terms since their signs cannot be defined *a priori*.

Of course, the extra terms arising from the interaction with the boundaries depend on the specific BC technique applied. Along this line, and in order to provide as many details as possible regarding those extra energy terms, during this thesis the already existing BC techniques have been classified in four groups: Kernel corrections, Boundary forces, Fluid extensions, and Boundary integrals. Using such classification, it has been demonstrated that the Boundary integrals technique is actually a combination of a kernel correction and a boundary force.

Four verification/validation tests have been discussed in the document. On top of the cases description and results, mandatory to prove that the tool is verified and validated, the most relevant implementation details have been provided for all the test cases. Also some notes regarding the performance are provided as well.

The first case consists of a Lid-driven cavity flow, being in fact one of the SPHERIC validation test cases. Such case has been selected due to its simplicity, ideally suited therefore to verify the package as a CFD solver, and to demonstrate the benefits of the modular implementation of AQUA_{gpusph}.

The second test case, a 2D standing wave, is used to verify the symmetry boundary condition provided “out of the box”, while the third test case, consisting in a Taylor-Green vortex, has been selected to verify the periodic boundary condition. The Taylor-Green test case is used at the same time to show how to take advantage of the AQUA_{gpusph} modular design to extend it, inserting a velocity field rescaling OpenCL script directly by the simulation XML definition files, i.e. without editing a single line of the AQUA_{gpusph} source code.

The last verification case is clearly the most complex one. Such case, extracted as well from the SPHERIC validation test cases, consists in a 2D moving box moving into a rectangular box confined fluid. It was selected to verify the energy theoretical analysis, including the effect of the δ -SPH term and the interaction with the solid boundaries. Along

this line, the case is implemented using both Fluid extensions and Boundary integrals. Such case is used as well to show the Python extension capabilities.

Regarding the results obtained, a good agreement has been achieved for all the described test cases, such that AQUAgnusph can be considered successfully verified and validated.

Finally, two practical applications have been described in order to show the capability of AQUAgnusph to scale to significantly more complex problems.

First, a simulation of earthquake sloshing loads in a fast-neutron nuclear reactor has been carried out. Effectively, due to the combination of the molten lead coolant considered and the presence of dampers, which dramatically reduce the acceleration at the expense of large amplitude displacements, large sloshing phenomena can be expected, which may eventually imply high loads that should be considered during the design. Such practical application is characterized by the very complex 3D geometry considered, which inexorably imply the use of a large number of particles. To partially fix the issues when complex geometries are considered, AQUAgnusph has been equipped with a tool to generate an initial particles packing -i.e. and initial condition- from a surface mesh. Significant sloshing loads have been predicted at the reactor top slab. In this specific reactor design the top slab is a critical part, becoming in fact the only link point of the reactor cylinder and heat exchangers with the structural vessel. Therefore, the collapse of such piece is an unacceptable critical situation.

The second practical application is a complex coupling between AQUAgnusph, and two different seakeeping/ship motions solvers. In such coupling, AQUAgnusph is used to model the flow inside an arbitrary number of tanks, with the main objective of computing the forces and moments exerted by that on the ship, such that the seakeeping codes may take into account such information during the ship motions computation. Of course it is a 2 way coupled system, where AQUAgnusph reports the forces to the seakeeping codes, and the seakeeping codes enforce the motions of the tanks for the AQUAgnusph instances. To carry out such coupling, a TCP/IP network connection based GRID system has been setup, such that an arbitrary number of AQUAgnusph servers can be deployed in an arbitrary number of facilities, aiding SeaFEM or SHIXDOF clients, conforming a virtual synergistic organization. This practical application is probable the best example of how to take advantage on the modular design and extensibility of AQUAgnusph. Moreover, just 1088 lines of code, almost of them of Python, have been required to build up such a really complex interaction system.

Hence, during this thesis a new free SPH code has been developed. Such code has been designed bearing in mind the researcher common tasks, clearly dominated by the continuous implementation of new formulations, as well as the modification of already existing ones. Along this line, AQUAgnusph is a optimized application, accelerated with OpenCL, modular,

and extensible, which may eventually reduce the efforts required to the researchers to develop their new advances on the model.

In order to demonstrate that AQUA_{gpusph} may fit better to the necessities of the SPH researchers, theoretical aspects of the model, mainly related to the energy equations, have been analyzed, implemented, and verified. Details regarding such implementations have been provided in order to demonstrate that the required efforts have been minimized, while the performance of the tool has not been significantly penalized. Also, such theoretical investigations represents a contribution to the SPH model development.

However, several improvements and future works are still required.

Regarding the software, other models should be implemented on top of the already provided “out of the box”. In particular, the author would remark the incompressible SPH formulations and the linear elasticity structural models. The former imply the introduction of a simple way to setup and solve linear system of equations. The later require a fully new model, which is in fact widely related with the CFD one.

Regarding the theoretical aspects, there are three main points that have been treated in this thesis which should be further investigated in the future:

1. The boundary conditions.
2. The δ -SPH terms.
3. The energy balance.

The boundary conditions are in fact one of the SPHERIC grand challenges. As it has been discussed along this work, several boundary condition techniques have been already described in the literature, almost of them affected by consistency issues, energy conservation issues, or even momentum conservation ones. Therefore, improvements are still required to gain knowledge about the commented issues, and to eventually control or even fix them.

Regarding the δ -SPH terms, they have represented an actual revolution in the SPH model, allowing to dramatically increase the model stability with a minimal impact on the performance. However, as it has been demonstrated, the δ -SPH terms are introducing a new term in the energy balance, which should be controlled. Effectively, the candidate has pointed out that several δ -SPH terms, including the most popular one, represent an extra energy dissipation, which is a very good property. However, finding a term which is not altering the global energy balance would become a dramatic improvement.

Related to that, during the analysis of the δ -SPH term, it has been demonstrated that the R-SPH models are widely related with them, and moreover, they could be analyzed in the same context. Unfortunately, such analysis analysis has not been carried out. Therefore, a

further investigation of the R-SPH models in the context of the common δ -SPH models is left for future work.

Finally, the energy analysis should be extended to some other thermodynamic phenomena. In this sense, the temperature may play a critical role in the energy balance. However, in the present work such thermodynamic variable has been completely neglected.

Thesis publications

Refereed papers

Cercos-Pita, J. (2015). AQUAghusph, a new free 3D SPH solver accelerated with OpenCL. *Computer Physics Communications*, 192(0):295–312.

Cercos-Pita, J. L., Antuono, M., Colagrossi, A., and Souto-Iglesias, A. (2016a). SPH energy conservation for fluid-solid interactions. (Submitted for publication).

Cercos-Pita, J. L. and Bulian, G. (2016). Co-simulation of ship motions and sloshing in tanks. (Under review).

Cercos-Pita, J. L., Bulian, G., Pérez-Rojas, L., and Francescutto, A. (2016b). Coupled simulation of nonlinear ship motions and a free surface tank. *Ocean Engineering*.

Cercos-Pita, J. L., Herault, A., and Dalrymple, R. A. (2016c). Diffusive terms for the conservation of mass equation in SPH. *Applied Mathematical Modelling*. (Accepted for publication).

Macià, F., González, L. M., Cercos-Pita, J. L., and Souto-Iglesias, A. (2012). A boundary integral SPH formulation. Consistency and applications to ISPH and WCSPH. *Progress of Theoretical Physics*, 128(3).

Serván-Camas, B., Cercos-Pita, J., J., C., García-Cobb, J., and Souto-Iglesias, A. (2016). Time domain simulation of coupled sloshing-seakeeping problems. (Under review).

Souto-Iglesias, A., Macià, F., González, L. M., and Cercos-Pita, J. L. (2013). On the consistency of MPS. *Computer Physics Communications*, 184(3):732–745.

Souto-Iglesias, A., Macià, F., González, L. M., and Cercos-Pita, J. L. (2014). Addendum to On the consistency of MPS [Comput. Phys. Comm. 184 (3) (2013) 732-745]. *Computer Physics Communications*, 185(2):595–598.

Conference papers

- Barrera, G., Dinoi, P., Cercós, J., González, L., Guerrero, A., Beltrán, F., and Moreno, A. (2013). Sloshing Effects in LFR Systems. In *SILER Workshop*. UE FP7.
- Cercos-Pita, J. L., Bulian, G., and Souto-Iglesias, A. (2015). Time domain assessment of nonlinear coupled ship motions and sloshing in free surface tanks. In *ASME 2015 34th International Conference on Ocean, Offshore and Arctic Engineering*, pages V001T01A039–V001T01A039. American Society of Mechanical Engineers.
- Cercos-Pita, J. L., Gonzalez, L. M., Moreno, A., Guerrero, A., and Salgado, S. (2014). Simulation of earthquake sloshing loads in a nuclear reactor. In *9th International SPHERIC SPH Workshop*.
- Cercos-Pita, J. L., Souto-Iglesias, A., Gonzalez, L. M., and Macià, F. (2013). AQUA_{gpusph}, a free 3D SPH solver accelerated with OpenCL. In *8th International SPHERIC SPH Workshop*.
- Cheng, L. Y., Souto-Iglesias, A., Simos, A., Cercos, J. L., Tsukamoto, M. M., Endo, C. Y., Marin, M. A., and Botia, E. (2009). Hydrodynamic Impact Pressure Computations and Experiments in an LNG Tank Section. In *III International Conference on Computational Methods in Marine Engineering*. CIMNE.
- Colagrossi, A., Delorme, L., Colicchio, G., Souto-Iglesias, A., and Cercós-Pita, J. L. (2008). Reynolds number and Shallow Depth Sloshing. In *3rd ERCOFTAC SPHERIC workshop on SPH applications*, pages 221–228.
- González, L., Cercos-Pita, J., and Macià, F. (2012). On the boundary condition enforcement in SPH methods. In *7th ERCOFTAC SPHERIC workshop on SPH applications*, pages 303–310. Monash University, INSEAN, University of Pavia.
- Gonzalez, L. M. and Cercos-Pita, J. L. (2013). An implicit SPH solution of the Burgers equation. In *8th International SPHERIC SPH Workshop*.
- Macià, F., Souto-Iglesias, A., Gonzalez, L. M., and Cercos-Pita, J. L. (2013). MPS \equiv SPH. In *8th International SPHERIC SPH Workshop*.
- Pérez-Rojas, L., Bulian, G., Botía-Vera, E., Cercos-Pita, J. L., Souto-Iglesias, A., and Delorme, L. (2009). A combined experimental and SPH approach to sloshing and ship roll motions. In *10th International Conference on Stability of Ships and Ocean Vehicles (STAB 2009)*.

- Pérez-Rojas, L. and Cercos-Pita, J. (2012). 3D GPU SPH analysis of coupled sloshing and roll motion. In *11th International Conference on Stability of Ships and Ocean Vehicles (STAB 2012)*.
- Rey-Villaverde, A., Cercos-Pita, J. L., Souto-Iglesias, A., and González, L. M. (2011). Particle Methods Parallel Implementations by GP-GPU Strategies. In *II International Conference on Particle-based Methods - Fundamentals and Applications, PARTICLES 2011*.

Other publications

Other publications, not directly related to the thesis main topic, but carried during the thesis are reported below:

Bouscasse, B., Colagrossi, A., Souto-Iglesias, A., and Cercos-Pita, J. L. (2014a). Mechanical energy dissipation induced by sloshing and wave breaking in a fully coupled angular motion system. I. Theoretical formulation and numerical investigation. *Physics of Fluids (1994-present)*, 26(3).

Bouscasse, B., Colagrossi, A., Souto-Iglesias, A., and Cercos-Pita, J. L. (2014b). Mechanical energy dissipation induced by sloshing and wave breaking in a fully coupled angular motion system. II. Experimental investigation. *Physics of Fluids (1994-present)*, 26(3).

Gomez-Goni, J., Garrido-Mendoza, C. A., Cercos-Pita, J., and Gonzalez, L. (2013). Two phase analysis of sloshing in a rectangular container with Volume of Fluid (VOF) methods. *Ocean Engineering*, 73(0):208–212.

Gutiérrez, L. M. G., Pita, J. L. C., and Burgos, D. E. (2016). Implantación de metodologías de cálculo a través del lenguaje python para asignaturas impartidas en las titulaciones de la etsin. *Modelling in Science Education and Learning*, 9(1):151–160.

References

References

- Alemberti, A., Carlsson, J., Malambu, E., Orden, A., Cinotti, L., Struwe, D., Agostini, P., and Monti, S. (2011a). ELSY-European LFR Activities. *Journal of Nuclear Science and Technology*, 48(4):479–482.
- Alemberti, A., Carlsson, J., Malambu, E., Orden, A., Struwe, D., Agostini, P., and Monti, S. (2011b). European lead fast reactor-ELSY. *Nuclear Engineering and Design*, 241(9):3470–3480.
- Amada, T., Imura, M., Yasumuro, Y., Manabe, Y., and Chihara, K. (2004). Particle-based fluid simulation on GPU. In *ACM workshop on general-purpose computing on graphics processors*, volume 41, page 42. Citeseer.
- AMD (2006). Accelerated Parallel Processing (APP) SDK. <http://developer.amd.com/tools/heterogeneous-computing/amd-accelerated-parallel-processing-app-sdk>.
- Amicarelli, A., Marongiu, J.-C., Leboeuf, F., Leduc, J., and Caro, J. (2011). SPH truncation error in estimating a 3D function. *Computers & Fluids*, 44(1):279–296.
- Antuono, M., Colagrossi, A., and Marrone, S. (2012). Numerical diffusive terms in weakly-compressible SPH schemes. *Computer Physics Communications*, 183(12):2570–2580.
- Antuono, M., Colagrossi, A., Marrone, S., and Lugni, C. (2011). Propagation of gravity waves through an SPH scheme with numerical diffusive terms. *Computer Physics Communications*, 182(4):866–877.
- Antuono, M., Colagrossi, A., Marrone, S., and Molteni, D. (2010). Free-surface flows solved by means of SPH schemes with numerical diffusive terms. *Computer Physics Communications*, 181(3):532–549.
- Antuono, M., Marrone, S., Colagrossi, A., and Bouscasse, B. (2015). Energy balance in the δ -SPH scheme. *Computer Methods in Applied Mechanics and Engineering*, 289.
- Bai, K. and Rhee, K. (1987). Roll-damping tank test. *Seoul National University, Project report*.
- Belytschko, T., Krongauz, Y., Dolbow, J., and Gerlach, C. (1998). On the completeness of meshfree particle methods. *Int. J. Numer. Methods Engineering*, 43(5):785–819.
- Blender foundation (2013). Free and open-source 3D computer graphics software product Blender.

- Bonet, J. and Rodriguez-Paz, M. (2005). Hamiltonian formulation of the variable-h SPH equations. *J. Comp. Phys.*, 209:541–558.
- Bouscasse, B., Colagrossi, A., Marrone, S., and Antuono, M. (2013). Nonlinear water wave interaction with floating bodies in SPH. *Journal of Fluids and Structures*, 42:112–129.
- Breuer, M., De Nayer, G., Münsch, M., Gallinger, T., and Wüchner, R. (2012). Fluid-structure interaction using a partitioned semi-implicit predictor-corrector coupling scheme for the application of large-eddy simulation. *Journal of Fluids and Structures*, 29:107–130.
- Bulian, G., Francescutto, A., and Sinibaldi, M. (2012). Roll motion of a ship with low metacentric height in bi-chromatic beam waves. In *11th International Conference on the Stability of Ships and Ocean Vehicles (STAB2012)*, pages 187–200.
- Bulian, G., Souto-Iglesias, A., Delorme, L., and Botia-Vera, E. (2010). SPH simulation of a Tuned Liquid Damper with angular motion. *Journal of Hydraulic Research*, 48(Extra Issue):28–39.
- Bullet physics library (2013). Open source physics engine.
- Bunnik, T. and Veldman, A. (2010). Modelling the effect of sloshing on ship motions. In *ASME 2010 29th International Conference on Ocean, Offshore and Arctic Engineering*, pages 279–286. American Society of Mechanical Engineers.
- Campbell, P. (1989). Some new algorithms for boundary value problems in Smooth Particle Hydrodynamics. Technical report, Defende Nuclear Agency.
- Cercos-Pita, J. (2015). AQUAghusph, a new free 3D SPH solver accelerated with OpenCL. *Computer Physics Communications*, 192(0):295–312.
- Cercos-Pita, J. L., Antuono, M., Colagrossi, A., and Souto-Iglesias, A. (2016a). SPH energy conservation for fluid-solid interactions. *Computer Methods in Applied Mechanics and Engineering*. (Submitted for publication).
- Cercos-Pita, J. L., Herault, A., and Dalrymple, R. A. (2016b). Diffusive terms for the conservation of mass equation in SPH. *Applied Mathematical Modelling*. (Accepted for publication).
- Chellapandi, P., Chetal, S., and Raj, B. (2012). Numerical simulation of fluid-structure interaction dynamics under seismic loadings between main and safety vessels in a sodium fast reactor. *Nuclear Engineering and Design*, 253(0):125–141. {SI} : CFD4NRS-3.
- Cherfils, J., Pinon, G., and Rivoalen, E. (2012). JOSEPHINE: A parallel SPH code for free-surface flows. *Computer Physics Communications*, 183(7):1468–1480.
- Chern, M., Borthwick, A., and Eatock Taylor, R. (2005). Pseudospectral element model for free surface viscous flows. *International Journal of Numerical Methods for Heat & Fluid Flow*, 15(6):517–554.
- Cleary, P. (1997). Boundary force and damping : Modes of fluid oscillation and their dependence on properties of the boundary force. Technical report, CMIS.

- Cleary, P. (1998). A boundary force based on the gradient of the kernel. Technical report, CMIS.
- Cochran, T. B., Feiveson, H. A., Patterson, W., Pshakin, G., Ramana, M., Schneider, M., Suzuki, T., and von Hippel, F. (2010). Fast breeder reactor programs: history and status. *International Panel on Fissile Materials*.
- Colagrossi, A., Antuono, M., and Le Touzé, D. (2009). Theoretical considerations on the free-surface role in the Smoothed-particle-hydrodynamics model. *Physical Review E*, 79(5):056701.
- Colagrossi, A., Antuono, M., Souto-Iglesias, A., and Le Touzé, D. (2011). Theoretical analysis and numerical verification of the consistency of viscous smoothed-particle-hydrodynamics formulations in simulating free-surface flows. *Physical Review E*, 84:026705.
- Colagrossi, A., Colicchio, G., and Le Touzé, D. (2007). Enforcing boundary conditions in SPH applications involving bodies with right angles. In *2nd ERCOFTAC SPHERIC Workshop*.
- Colagrossi, A., Colicchio, G., Lugni, C., and Brocchini, M. (2010). A study of violent sloshing wave impacts using an improved SPH method. *Journal of Hydraulic Research*, 48(Extra Issue):94–104.
- Conti, C., Rossinelli, D., and Koumoutsakos, P. (2012). GPU and APU computations of Finite Time Lyapunov Exponent fields. *Journal of Computational Physics*, 231(5):2229–2244.
- Crespo, A., Dominguez, J., Rogers, B., Gomez-Gesteira, M., Longshaw, S., Canelas, R., Vacondio, R., Barreiro, A., and Garcia-Feal, O. (2015). DualSPHysics: Open-source parallel {CFD} solver based on Smoothed Particle Hydrodynamics (SPH). *Computer Physics Communications*, 187(0):204–216.
- Crespo, A., Gómez-Gesteira, M., and Dalrymple, R. A. (2007). Boundary conditions generated by dynamic particles in SPH methods. *CMC-TECH SCIENCE PRESS*-, 5(3):173.
- Crespo, A. C., Dominguez, J. M., Barreiro, A., Gómez-Gesteira, M., and Rogers, B. D. (2011). GPUs, a New Tool of Acceleration in CFD: Efficiency and Reliability on Smoothed Particle Hydrodynamics Methods. *PLoS ONE*, 6(6).
- Daga, M., Aji, A. M., and Feng, W.-c. (2011). On the efficacy of a fused CPU+GPU processor (or APU) for parallel computing. In *Application Accelerators in High-Performance Computing (SAAHPC), 2011 Symposium on*, pages 141–149. IEEE.
- De Leffe, M., Le Touzé, D., and Alessandrini, B. (2009). Normal flux method at the boundary for SPH. In *4th SPHERIC*, pages 149–156.
- Dehnen, W. and Aly, H. (2012). Improving convergence in smoothed particle hydrodynamics simulations without pairing instability. *Monthly Notices of the Royal Astronomical Society*, 425(2):1068–1082.

- Dominguez, J. M., Crespo, A. J. C., and Gomez-Gesteira, M. (2013). Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method. *Computer Physics Communications*, 184(3):617–627.
- Farhat, C., Van der Zee, K. G., and Geuzaine, P. (2006). Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity. *Computer methods in applied mechanics and engineering*, 195(17):1973–2001.
- Fatehi, R. and Manzari, M. (2011). A remedy for numerical oscillations in weakly compressible smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 67(9):1100–1114.
- Federico, I., Marrone, S., Colagrossi, A., Aristodemo, F., and Antuono, M. (2012). Simulating 2D open-channel flows through an SPH model. *European Journal of Mechanics-B/Fluids*, 34:35–46.
- Feldman, J. and Bonet, J. (2007). Dynamic refinement and boundary contact forces in SPH with applications in fluid flow problems. *International Journal for Numerical Methods in Engineering*, 72(3):295–324.
- Ferrand, M., Laurence, D. R., Rogers, B. D., Violeau, D., and Kassiotis, C. (2013). Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method. *International Journal for Numerical Methods in Fluids*, 71(4):446–472.
- Ferrari, A., Dumbser, M., Toro, E. F., and Armanini, A. (2009). A new 3D parallel SPH scheme for free surface flows. *Computers & Fluids*, 38(6):1203–1217.
- Forni, M. and De Grandis, S. (2012). Seismic-Initiated events risk mitigation in LEad-cooled Reactors: the SILER Project. *Proc. of the 15WCEE, Lisbon*, pages 24–28.
- Fourtakas, G., Dominguez, J., Vacondio, R., Nasar, A., and Rogers, B. (2014). Local Uniform Stencil (LUST) boundary conditions for 3-D irregular boundaries in DualSPHysics. In *Proc. 9th Int. SPHERIC Workshop, Paris*, pages 103–110.
- Francescutto, A., Contento, G., et al. (1999). An investigation on the applicability of simplified mathematical models to the roll-sloshing problem. *International Journal of Offshore and Polar Engineering*, 9:97–104.
- Frano, R. L. and Forasassi, G. (2012). Preliminary evaluation of the seismic response of the {ELSY} {LFR}. *Nuclear Engineering and Design*, 242(0):361–368.
- Ghia, U., Ghia, K. N., and Shin, C. (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of computational physics*, 48(3):387–411.
- Gingold, R. and Monaghan, J. (1977). Smoothed Particle Hydrodynamics: theory and application to non-spherical stars. *Mon. Not. Roy. Astron. Soc. (MNRAS)*, 181:375–389.
- Godunov, S. K. (1959). A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Matematicheskii Sbornik*, 89(3):271–306.

- Gomes, J. P., Yigit, S., Lienhart, H., and Schäfer, M. (2011). Experimental and numerical study on a laminar fluid-structure interaction reference test case. *Journal of Fluids and Structures*, 27(1):43–61.
- Gomez-Gesteira, M., Crespo, A., Rogers, B., Dalrymple, R., Dominguez, J., and Barreiro, A. (2012). SPHysics - development of a free-surface fluid solver - Part 2: Efficiency and test cases. *Computers & Geosciences*, 48(0):300–307.
- Gromov, B., Belomitcev, Y. S., Yefimov, E., Leonchuk, M., Martinov, P., Orlov, Y. I., Pankratov, D., Pashkin, Y. G., Toshinsky, G., Chekunov, V., et al. (1997). Use of lead-bismuth coolant in nuclear reactors and accelerator-driven systems. *Nuclear Engineering and Design*, 173(1):207–217.
- Harris, M. (2002). General-Purpose Computation on Graphics Hardware.
- Hashemi, M., Fatehi, R., and Manzari, M. (2011). SPH simulation of interacting solid bodies suspended in a shear flow of an Oldroyd-B fluid. *Journal of Non-Newtonian Fluid Mechanics*, 166(21):1239–1252.
- Hashemi, M., Fatehi, R., and Manzari, M. (2012). A modified SPH method for simulating motion of rigid bodies in Newtonian fluid flows. *International Journal of Non-Linear Mechanics*, 47(6):626–638.
- Herauld, A., Bilotta, G., and Dalrymple, R. A. (2010). SPH on GPU with CUDA. *Journal of Hydraulic Research*, 48(sup1):74–79.
- Holden, C. and Fossen, T. I. (2012). A nonlinear 7-DOF model for U-tanks of arbitrary shape. *Ocean Engineering*, 45:22–37.
- Hu, X. and Adams, N. A. (2006). Angular-momentum conservative Smoothed Particle Hydrodynamics for incompressible viscous flows. *Physics of Fluids*, 18:702–706.
- Ihmsen, M., Bader, J., Akinci, G., and Teschner, M. (2011). Animation of air bubbles with SPH. In *GRAPP*.
- Issa, R., Lee, E. S., Violeau, D., and Laurence, D. R. (2005). Incompressible separated flows simulations with the smoothed particle hydrodynamics gridless method. *International journal for numerical methods in fluids*, 47(10-11):1101–1106.
- Kajtar, J. and Monaghan, J. (2008). SPH simulations of swimming linked bodies. *Journal of Computational Physics*, 227(19):8568–8587.
- Khayyer, A. and Gotoh, H. (2009). Modified Moving Particle Semi-implicit methods for the prediction of 2D wave impact pressure. *Coastal Engineering*, 56(4):419–440.
- Khronos group (2009). Open Computing Language (OpenCL). <http://www.khronos.org/opencl>.
- Kim, B. and Shin, Y. (2008). Coupled seakeeping with liquid sloshing in ship tanks. In *ASME 2008 27th International Conference on Offshore Mechanics and Arctic Engineering*, pages 247–257. American Society of Mechanical Engineers.

- Kim, Y., Nam, B., Kim, D., and Kim, Y. (2007). Study on coupling effects of ship motion and sloshing. *Ocean Engineering*, 34(16):2176–2187.
- Kipfer, P., Segal, M., and Westermann, R. (2004). UberFlow: A GPU-based Particle Engine. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware*, HWWS '04, pages 115–122, New York, NY, USA. ACM.
- Koukouvini, P., Anagnostopoulos, J., and Papantonis, D. E. (2013). An improved MUSCL treatment for the SPH-ALE method: comparison with the standard SPH method for the jet impingement case. *International Journal for Numerical Methods in Fluids*, 71:1152–1177.
- Ku, H. C., Hirsh, R. S., and Taylor, T. D. (1987). A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations. *Journal of Computational Physics*, 70(2):439–462.
- Kulasegaram, S., Bonet, J., Lewis, R., and Profit, M. (2004). A variational formulation based contact algorithm for rigid boundaries in two-dimensional SPH applications. *Computational Mechanics*, 33:316–325.
- Lee, E. S., Moulinec, C., Xu, R., Violeau, D., Laurence, D., and Stansby, P. (2008). Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. *Journal of Computational Physics*, 227(18):8417–8436.
- Lewis, E. V. (1989). *Principles of naval architecture (vol. II)*. Principles of Naval Architecture. Society of Naval Architects and Marine Engineers.
- Lighthill, J. (2001). *Waves in fluids*. Cambridge University Press.
- Liu, G. and Liu, M. (2003). *Smoothed particle hydrodynamics. A meshfree particle method*. World Scientific Publishing Co. Pte. Ltd.
- Lucy, L. (1977). A numerical approach to the testing of the fission hypothesis. *Astronomical Journal*, 82:1013–1024.
- MacDonald, J. R. (1966). Some Simple Isothermal Equations of State. *Rev. Mod. Phys.*, 38:669–679.
- Macià, F., Antuono, M., González, L. M., and Colagrossi, A. (2011a). Theoretical Analysis of the No-Slip Boundary Condition Enforcement in SPH Methods. *Progress of Theoretical Physics*, 125(6):1091–1121.
- Macià, F., Colagrossi, A., Antuono, M., and Souto-Iglesias, A. (2011b). Benefits of using a Wendland kernel for free-surface flows. In *6th ERCOFTAC SPHERIC workshop on SPH applications*.
- Macià, F., González, L. M., Cercos-Pita, J. L., and Souto-Iglesias, A. (2012). A boundary integral SPH formulation. Consistency and applications to ISPH and WCSPH. *Progress of Theoretical Physics*, 128(3).
- Maintz, S., Eck, B., and Dronskowski, R. (2011). Speeding up plane-wave electronic-structure calculations using graphics-processing units. *Computer Physics Communications*, 182(7):1421–1427.

- Malenica, S., Zalar, M., and Chen, X. (2003). Dynamic coupling of seakeeping and sloshing. In *Proceedings of the 13th international offshore and polar engineering conference*, pages 484–490.
- Marrone, S., Antuono, M., Colagrossi, A., Colicchio, G., Le Touzé, D., and Graziani, G. (2011a). Delta-SPH model for simulating violent impact flows. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16):1526–1542.
- Marrone, S., Colagrossi, A., Antuono, M., Colicchio, G., and Graziani, G. (2013). An accurate SPH modeling of viscous flows around bodies at low and moderate Reynolds numbers. *Journal of Computational Physics*, 245:456–475.
- Marrone, S., Colagrossi, A., Antuono, M., Lugni, C., and Tulin, M. (2011b). A 2D+t SPH model to study the breaking wave pattern generated by fast ships. *Journal of Fluids and Structures*, 27(8):1199–1215.
- Martinsen, P., Blaschke, J., Künemeyer, R., and Jordan, R. (2009). Accelerating Monte Carlo simulations with an NVIDIA graphics processor. *Computer Physics Communications*, 180(10):1983–1989.
- Merino-Alonso, P., Macia, F., Souto-Iglesias, A., and Colagrossi, A. (2013). Consistency analysis of flow field extension models into ghost fluid regions for SPH solid body boundary condition implementations. In *8th ERCOFTAC SPHERIC workshop on SPH applications*.
- Mitra, S., Wang, C., Reddy, J., and Khoo, B. (2012). A 3D fully coupled analysis of nonlinear sloshing and ship motion. *Ocean Engineering*, 39:1–13.
- Molero-Armenta, M., Iturrarán-Viveros, U., Aparicio, S., and Hernández, M. (2014). Optimized OpenCL implementation of the Elastodynamic Finite Integration Technique for viscoelastic media. *Computer Physics Communications*, 185(10):2683–2696.
- Molteni, D. and Bilello, C. (2003). Riemann solver in SPH. *Memorie della Societa Astronomica Italiana Supplementi*, 1:36.
- Molteni, D. and Colagrossi, A. (2009). A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH. *Computer Physics Communications*, 180:861–872.
- Monaghan, J. (1989). On the problem of penetration in particle methods. *J. Comp. Phys.*, 82.
- Monaghan, J. (2012). Smoothed Particle Hydrodynamics and Its Diverse Applications. *Annual Review of Fluid Mechanics*, 44(1):323–346.
- Monaghan, J. and Gingold, R. A. (1983). Shock Simulation by the particle method SPH. *Journal of Computational Physics*, 52(2):374–389.
- Monaghan, J. and Kajtar, J. (2009). SPH particle boundary forces for arbitrary boundaries. *Computer Physics Communications*, 180(10):1811–1820.
- Morris, J. P., Fox, P. J., and Zhu, Y. (1997). Modeling Low Reynolds Number Incompressible Flows Using SPH. *Journal of Computational Physics*, 136:214–226.

- MPI Forum (2013). The Message Passing Interface (MPI). <http://www.mpi-forum.org>.
- Niemeyer, K. and Sung, C.-J. (2013). Recent progress and challenges in exploiting graphics processors in computational fluid dynamics. *The Journal of Supercomputing*, pages 1–37.
- NVidia (2006). Compute Unified Device Architecture (CUDA). http://www.nvidia.com/object/cuda_home_new.html.
- OpenMP Architecture Review Board (2013). OpenMP API specification for parallel programming. <http://www.openmp.org>.
- Qin, C., Liu, W., and He, W. (2014). Seismic response analysis of isolated nuclear power plants with friction damper isolation system. *AASRI Procedia*, 7:26–31.
- Quinlan, N. J., Lastiwka, M., and Basa, M. (2006). Truncation error in mesh-free particle methods. *International Journal for Numerical Methods in Engineering*, 66(13):2064–2085.
- Rafiee, A., Cummins, S., Rudman, M., and Thiagarajan, K. (2012). Comparative study on the accuracy and stability of SPH schemes in simulating energetic free-surface flows. *European Journal of Mechanics - B/Fluids*, 36(0):1–16.
- Ramachandran, P. and Kaushik, C. (2013). PySPH: A Python framework for Smoothed Particle Hydrodynamics. In *8th ERCOFTAC SPHERIC Workshop on SPH Applications*, pages 191–197.
- Randles, P. and Libersky, L. (1996). Smoothed Particle Hydrodynamics: some recent improvements and applications. *Computer methods in applied mechanics and engineering*, 39:375–408.
- Rusanov, V. V. (1962). *Calculation of interaction of non-steady shock waves with obstacles*. NRC, Division of Mechanical Engineering.
- Serván-Camas, B. and García-Espinosa, J. (2013). Accelerated 3D multi-body seakeeping simulations using unstructured finite elements. *Journal of Computational Physics*, 252(0):382–403.
- Shao, S. and Lo, E. Y. (2003). Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in Water Resources*, 26(7):787–800.
- Simpson, J. and Wood, M. (1996). Classical kinetic theory simulations using smoothed particle hydrodynamics. *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, 54(2):2077–2083.
- Souto-Iglesias, A., Delorme, L., Pérez-Rojas, L., and Abril-Pérez, S. (2006). Liquid moment amplitude assessment in sloshing type problems with smooth particle hydrodynamics. *Ocean Engineering*, 33(11-12):1462–1484.
- Souto-Iglesias, A., Macià, F., González, L. M., and Cercos-Pita, J. L. (2013). On the consistency of MPS. *Computer Physics Communications*, 184(3):732–745.
- Souto-Iglesias, A., Macià, F., González, L. M., and Cercos-Pita, J. L. (2014). Addendum to On the consistency of MPS [Comput. Phys. Comm. 184 (3) (2013) 732-745]. *Computer Physics Communications*, 185(2):595–598.

- Spanos, D. and Papanikolaou, A. (2001). On the stability of fishing vessels with trapped water on deck. *Ship Technology Research-Schiffstechnik*, 48(3):124–133.
- Stanford University graphics group (2004). BrookGPU. <http://graphics.stanford.edu/projects/brookgpu>.
- Toro, E. F. (1997). *Riemann solvers and numerical methods for fluid dynamics*. Springer, Berlin.
- Vacondio, R., Rogers, B., Stansby, P., Mignosa, P., and Feldman, J. (2013). Variable resolution for SPH: a dynamic particle coalescing and splitting scheme. *Computer Methods in Applied Mechanics and Engineering*.
- Vila, J. (1999). On particle weighted methods and Smooth Particle Hydrodynamics. *Mathematical Models & Methods in Applied Sciences*, 9(2):161–209.
- Waltar, A. E. and Reynolds, A. B. (1981). *Fast breeder reactors*. Alan E. Waltar.
- Xu, R., Stansby, P., and Laurence, D. (2009). Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics*, 228(18):6703–6725.
- Zhao, W., Yang, J., Hu, Z., Xiao, L., and Tao, L. (2014). Hydrodynamics of a 2D vessel including internal sloshing flows. *Ocean Engineering*, 84(0):45–53.

