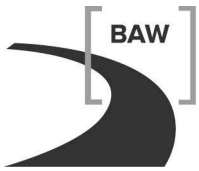
Report, Published Version

Jankowski, Jacek

Evaluation and adaption of the SPH method for hydraulic engineering problems on federal waterways. FuE-Abschlussbericht B3953.05.04.70002



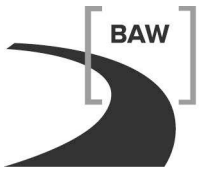
Bundesanstalt für Wasserbau
Kompetenz für die Wasserstraßen

R&D-Project

Evaluation and adaption of the SPH method for hydraulic engineering problems on federal waterways

Final Report

B3953.05.04.70002



Bundesanstalt für Wasserbau
Kompetenz für die Wasserstraßen

R&D-Project

Evaluation and adaption of the SPH method for hydraulic engineering problems on federal waterways

Final Report

Client:	BAW
Order Date:	26th January 2010
Order Number:	BAW-No. B3953.05.04.70002
Prepared by:	Department: W Section: W2,W5,W4 Project lead: Jacek A. Jankowski Person in charge: Eugenio Rustico

Karlsruhe, *October* 2016

Bundesanstalt für Wasserbau · Kußmaustr. 17 · D-76187 Karlsruhe · ([+49] (0)721 9726 - 0

Contents

List of Figures

Summary

Smoothed Particle Hydrodynamics (SPH) is a Lagrangian method for fluid dynamic simulations. In the past decades it drew the attention of the scientific community for its versatility and the possibility to simulate complex phenomena such as e.g. surface tension and fluid-solid interactions with floating objects. SPH has in general higher computational requirements than the most common Eulerian methods and several different models have been proposed for the treatment of boundaries, each with advantages and limitations.

The project aimed to develop, validate and increase the computational efficiency of an existing SPH code for hydraulic engineering problems. As base code for the development, the GPUSPH simulator has been chosen. As the name suggests, it is a SPH implementation which exploits the Graphic Processing Units (GPU) to perform the computation of the particle to particle interactions. The GPUSPH simulator has been developed within the ATHOS Consortium, whose members include universities and research institutes from different nations working on different aspects of the code. A development branch external to BAW but of particular interest for the project regards the newer "Semi-Analytical" boundary model, the development of which has been completed only recently.

In the first phase, several features have been improved or introduced in the simulator, especially the capability of splitting a simulation across multiple nodes of a cluster each equipped with one or more GPU devices. This additional level of parallelism has enabled the simulation of high-resolution or spatially big scenarios (more than 100 million particles).

In the second phase, two test cases have been developed. The first is a fish pass, that has been modeled following the plan of a physical scaled model situated in the laboratories of BAW Karlsruhe. Simple Lennard-Jones boundaries have been used, featuring "pseudo-open" boundaries capable of controlling the inlet flow and specifically developed for the test case. A comparison of the simulated flow and a set of laboratory measurements highlighted a very good agreement in terms of the stream shape, including the recirculation areas. The water level were however too high in the simulated model because of an excess of friction due to the boundary model.

A ship lock has been modeled as second test case replicating the new Kiel-Holtenau lock, of which a scaled model was built at BAW Karlsruhe. The first implementation used the Semi-Analytical boundaries, featuring pressure- or velocity-driver open boundaries. Although the first results appeared to be promising, the low computational performance of the boundary model made a continuation of the tests on a large scale infeasible. Since a performance-optimized version of the model was not yet available (and is still in progress within the consortium), a second attempt has been done with the Dynamic Boundaries, a lighter boundary model featuring correct pressure at the boundaries but no inlets. The water inflow has been implemented either with "pseudo-inlet" and with a piston chamber. Both attempts showed a strong artifact causing the stream flow to go upward and hit the ship frontally rather than passing under it. This yielded the forces acting on the ship to be stronger than the forces measured in a laboratory model at BAW. The causes of the difference have been identified with high probability in the artificial stream lift caused by the non-physical inlet and the excess of friction along the ship hull. Both factors could be elimi-

nated with a more physics-adherent boundary model.

The test cases highlighted the limits of the simpler boundary models and the need for an optimized, industrial-level version of the newly developed Semi-Analytical boundaries. Although the current development status and the first results are very promising for the applicability of the method to complex real-life engineering problems, and are already being used in the scientific community for many simpler problem scenarios, an optimization of the new boundary model is crucial to free the high potential of the method and accurately determine its application range and characteristics. An additional time of at least one year would be required to re-implement both test cases with the new model and produce a definitive comparison, granted the availability of the consortium members involving the optimization of the boundary model.

1 Introduction

Smoothed Particle Hydrodynamics (SPH) is a free-surface, Lagrangian method for fluid dynamic simulations. For its versatility and features, like the possibility to simulate complex phenomena such as surface tension, crust formation and fluid-solid interactions with floating objects, it attracted in the past fifteen years a particular attention in several fields such as civil and naval engineering, natural disaster risk assessment and oceanography.

SPH has in general higher computational requirements than the most common Eulerian methods both in terms of memory, for the high number of particles needed to achieve a significant resolution, and of computation, due to the small timesteps often required by the stability conditions. Many software packages implementing an SPH-based model exploit some degree of hardware parallelism (multiple CPU cores, single or multiple GPUs, multiple nodes in a network) to fulfill these requirements by keeping the computation times in a practicable scale.

The scientific community discussed for long time about the convergence of the SPH method and about the treatment of the boundaries, which appear to be the most important and different aspects with respect to Eulerian methods. Several solutions have been proposed in particular for the treatment of the boundaries, for which a variety of boundary models have been validated and used in different applications.

The project described here aimed to increase the computational efficiency, development and validation of an existing SPH code for hydraulic engineering problems in which the traditional grid-based methods reveal its applicability limits.

The existing code used as basis for the project is GPUSPH (`https://www.gpusph.org`), a powerful SPH simulator based on the GPGPU (General Purpose Graphic Processing Unit) paradigm, i.e. exploiting the computational power of the graphic cards to perform fluid dynamic simulations.

GPUSPH is being developed within the scope of the ATHOS research program, which brings together international partners interested in the development of "advanced tools and methods for computational fluid-dynamics".

The current ATHOS partners are:

- Istituto Nazionale di Geofisica e Vulcanologia (INGV), Sezione di Catania, Italy,
- Johns Hopkins University (JHU), Baltimore (MD), USA,
- Bundesanstalt für Wasserbau (BAW), Karlsruhe, Germany,
- Univeristät für Bodenkultur Wien (BOKU), Wien, Austria,
- Coastal Studies Institute, North Carolina University (CSI),
- Laboratoire d'Hydraulique Saint-Venant (LHSV),
- Conservatoire National des Arts et Métiers, Paris, France (CNAM),
- EDF R&D, Chatou, France.

2 Smoothed Particle Hydrodynamics (SPH)

2.1 Introduction

Smoothed Particle Hydrodynamics (SPH) is a meshless Lagrangian numerical method for computational fluid-dynamics, originally developed by Gingold, Monaghan and Lucy (Gingold and Monaghan, 1977; Monaghan, 1977) for astrophysics, and that has recently seen applications in a number of fields so different as astrophysics, volcanology and oceanography. It has recently seen increasing interest with application to several aspects of fluid dynamics, from free surface to thermal problems (Cleary and Monaghan, 1999), from gaseous media to hair and deformable objects. It has been applied to simulate phase transitions (Keiser et al., 2005; Solenthaler et al., 2007), viscous fluids and interaction of miscible and immiscible fluids (Müller et al., 2005; Solenthaler and Pajarola, 2008). Other partially explored fields include fluid-solid interaction and solid mechanics, where it is often referred to as SPAM (Smooth-Particle Applied Mechanics) (Hoover, 2006).

Among the key features of SPH it is worth mentioning that:

- it guarantees conservation of mass without extra computation,
- the pressure is computed from weighted contributions of neighboring particles rather than by solving linear systems of equations,
- unlike grid-base technique, with SPH it is not necessary to track fluid boundaries either in single- or multi-phase simulations,
- can be used to solved purely dynamic problems for Newtonian flows as well as thermal problems and to model non-Newtonian fluids,
- it is possible to interpolate the scalar fields in arbitrary points of the domain,
- free-surface problems are easily managed.

We refer the reader to recent reviews on SPH to (Gomez-Gesteira et al., 2010; Monaghan, 2005; Violeau, 2012) for further information.

2.2 Discretization of Navier-Stokes equations

In SPH, the fluid is discretized as a set of particles, each representing a virtual volume of fluid and thus carrying all the relevant properties. The value of any field A at a point r of the domain is determined by convolving the field value at the particle locations with a smoothing kernel $W(\cdot, h)$ with smoothing length h:

$$A(r) \approx \sum_j m_j \frac{A_j}{\rho_j} W(r - r_j, h)$$

where m_j, ρ_j, A_j are respectively the mass, density and value of the field A for particle j, r denotes position and summation is extended to all particles.

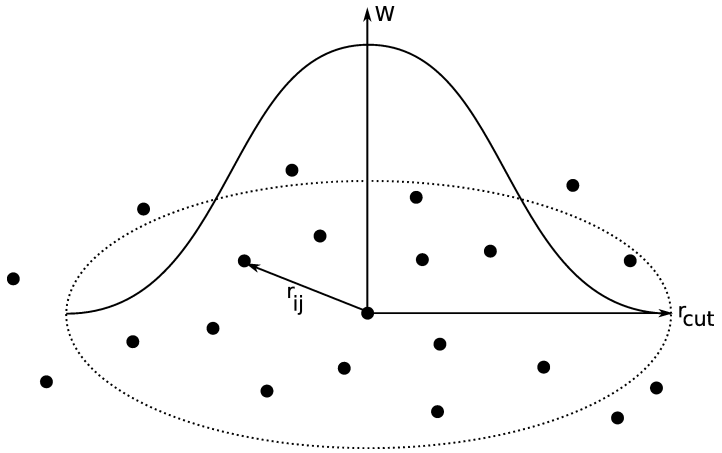


Figure 1: *Each particle only interacts with a small set of neighbors.*

The smoothing kernel is usually chosen positive (to ensure non-negative density throughout the simulation), symmetric (to ensure first-order accuracy of the method,) and with compact support. The latter choice limits the summations to the particles in a small neighborhood of the location (fig. 1), and reduces the computational complexity of the method from $O(N^2)$ to $O(N)$, where N is the number of particles in the system.

Using the properties of convolutions, it can be shown that for gradient computations the expressions can be arranged in such a way that the ∇ operator acts only on the kernel, so that, for example, the continuity equation:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot u$$

(where D/Dt indicates the material derivative and u is the velocity) takes the SPH form

$$\frac{D\rho_i}{Dt} = \rho_i \sum_j \frac{m_j}{\rho_j}(u_i - u_j) \cdot \nabla W(||r_i - r_j||, h)$$

(this is actually only one of the possible forms that the continuity equation can take in SPH), showing how the original partial differential equation has been turned into a simple ordinary differential equation in time with a right-hand side that can be directly computed from the mass, density, position and velocity of all the particles. It has been already mentioned that the kernel is chosen with compact support; it is also convenient to choose a kernel such that the factor

$$F(r,h) = \frac{1}{r}\frac{\partial W(r,h)}{\partial r}$$

can be computed analytically without an actual division by r. This allows us to write the Laplacian of a vector field v in the form

$$\nabla^2 v_i = \sum_j m_j \frac{\rho_i + \rho_j}{\rho_i \rho_j} F_{ij} v_{ij}$$

where $v_{ij} = v_j - v_i$ and $F_{ij} = F(r_{ij}, h)$. The standard SPH formulation models fluid as weakly-compressible, with an equation of state linking pressure P to density ρ. Usually the Tait equation is used in the form

$$P = \rho_0 \frac{c_s^2}{\gamma} \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right)$$

with ρ the density, ρ_0 the at-rest density of the fluid, γ the polytropic characteristic value and c_s the speed of sound of the fluid. In most applications, explicit integration methods are used, so that the timestep is limited by the speed of sound. Hence, rather than using the physical speed of sound, a fictitious speed of sound is used, which is at least one order of magnitude higher than the maximum velocity of the fluid in the given problem. This ensures that density fluctuations are kept small (less than 1 %) while allowing for larger timesteps. The Navier-Stokes equation of motion

$$\frac{Du}{Dt} = \frac{-\nabla P}{\rho} + \nu \nabla^2 u + g$$

is then discretized in SPH in the form

$$\frac{Du_i}{Dt} = -\sum_j m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \nabla_i W_{ij} + \sum_j m_j \frac{4\nu}{\rho_i + \rho_j} F_{ij} u_i + g$$

where ν is the kinematic viscosity coefficient and g represents external forces, in our case gravity.

The motion of a weakly compressible inviscid fluid can be described by the Navier-Stokes equations in the Lagrangian form

$$\frac{D\vec{u}}{Dt} = -\frac{\nabla P}{\rho} + \vec{g}$$
$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \vec{u},$$

where D/Dt denotes the total (Lagrangian) derivative with respect to time, $\vec{u}$ the fluid velocity, P the pressure, ρ the density, ν the kinematic viscosity coefficient, and $\vec{g}$ the external forces per unit mass (gravity in our case).

2.3 Time integration

The SPH model implemented in GPUSPH uses an explicit predictor/corrector integration scheme with a dynamic timestep. At each iteration, the timestep is controlled by standard CFL-type conditions determined by speed of sound, viscosity and force magnitude, so that for each iteration the the timestep taken is the largest possible that does not violate these constraints.

The predictor/corrector scheme is as follows. From the particle positions X_n and velocities V_n at time t_n we compute the particle accelerations $F^*_{n+1}(X_n, V_n)$ and we choose the

maximum timestep Δt^* allowable by all particles according to the CFL condition. The predicted new positions and velocities are then:

$$V^*_{n+1} = V_n + F^*_{n+1}\frac{\Delta t_n}{2},$$
$$X^*_{n+1} = X_n + V^*_{n+1}\frac{\Delta t_n}{2},$$

with the timestep Δt_n selected from the previous iteration. The correction step then computes $F^{**}_{n+1}(X^*_{n+1}, V^*_{n+1})$ and a new maximum timestep Δt^{**}. An intermediate corrected velocity V_c is computed as

$$V_c = V_n + F^{**}_{n+1}\frac{\Delta t_n}{2}$$

and the final velocities and positions are computed as

$$V_{n+1} = V_n + F^{**}_{n+1}\Delta t_n$$
$$X_{n+1} = X_n + V_c\Delta t_n$$

and we chose as the next timestep $\Delta t_{n+1} = \min(\Delta t^*, \Delta t^{**})$.

2.4 Boundary models

The treatment of the boundaries with SPH is still an open problem with no general consensus in the scientific community. Among the factors that characterize different models there can be, for example, a different way to preserve the compactness of the particle support at the edge of the domain, or the possibility to simulate open boundaries with pressure or velocity imposed. In many cases, the more advanced and physics-adherent a boundary model is, the more computationally expensive it is.

SPH-based fluid simulators sometimes support multiple boundary models to enable the user to use the model that suits the characteristics of the simulation most or to run the same simulation with different models and compare the results (e.g. pressure gradients at the boundaries).

Here follows an overview of the most common boundary models:

Lennard-Jones. The boundary particles exert a repulsive force on the fluid particles according to a Lennard-Jones potential. The support of the fluid particle is not complete and its scalar fields are therefore underestimated. This kind of boundary is particularly straightforward to implement and simple and it is suitable for quick tests or simulations where the pressure profile near the boundaries is not taken into account.

Dynamic boundaries. The boundary is made by multiple layers of fluid particles fixed in space. The number of layers is chosen so that the support of any fluid particles coming close to the boundary remains complete. It is a simple and light model capable of preserving a correct pressure field near the boundary. Disadvantages are the higher memory footprint for keeping multiple layers of boundaries, the thickness of the boundaries and the generation of such a boundary from an arbitrary triangular mesh, which is not trivial and often must be approximated.

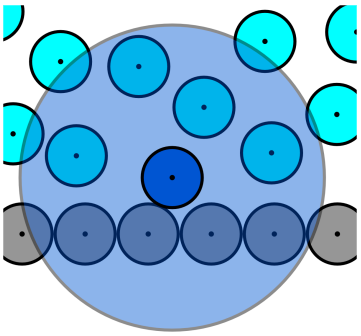


Figure 2: *A fluid particle and its support close to Lennard-Jones boundaries.*

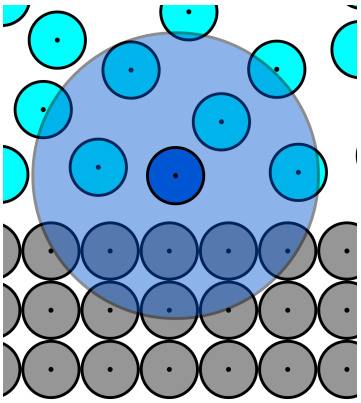


Figure 3: *A fluid particle and its support close to dynamic-boundary boundaries. Notice multiple layers of boundary are necessary.*

Mirror particles. More than a boundary model, this is a class of models. Multiple kinds of mirror particles have been proposed, ranging from planes providing plane-symmetric virtual particles to point-based boundaries producing point-symmetric ones. The main disadvantage of these techniques is the complexity of modeling an arbitrary shape and their usually high computational cost.

Semi-analytical boundaries. The boundary is represented as a triangular mesh and needs some preprocessing (e.g. resizing triangles, connectivity, normals). The completeness of the support of any fluid particle is analytically computed and stored in a parameter called γ; this is subsequently used to correct all fields. The model also features velocity- and pressure-driven open boundaries (inlets and outlets). The advantage of the method are the superior realism, quantified with dedicated validation tests, and its intrinsic capability of loading arbitrarily-shaped triangular meshes; its main disadvantage is its high computational cost.

The GPUSPH simulator supports the Lennard-Jones, Dynamic and Semi-Analytical model boundaries. Mirror-particles based models have been used for temporary tests and were only implemented in experimental code branches.

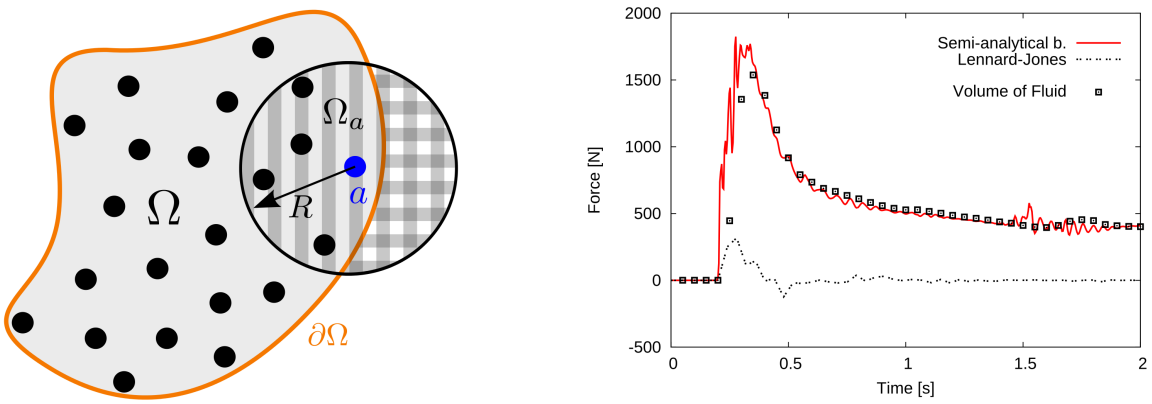


Figure 4: *Schematic depiction of the kernel support with the Semi-Analytical boundaries (left) and validation of the pressure field against Lennard-Jones particles and laboratory measurements (right).*

2.5 Implementation challenges

The aim of distributing the computational load of a simulation is twofold: increasing the size of the problem being simulated, in terms of absolute dimensions or in terms of resolution, and decreasing the wall-clock time required for the completion of a simulation. Similarly to grid methods, also Lagrangian methods (and thus SPH) exhibit a high level of parallelism and are therefore particularly suitable to distributed implementations. Unlike grid methods, however, where the set of neighbor cells of a simulation unit is static, the neighborhood of the particles in a SPH simulation is highly dynamic and needs to be continuously computed and updated. This introduces an additional level of complexity in the design of a distributed implementation. In the implementations storing a list of neighbors for each particle, the construction and update of this list is usually performed in a dedicated computation unit, separate from the classical SPH *kernels*.

The mathematical operations required for the computation of one step of a SPH simulation are dense but relatively few. It is therefore not convenient to distribute computational load across computing units by arranging a pipeline of the model steps. Computations are usually distributed according to the spatial partition either of the simulation domain or of the list of particles.

3 GPUSPH

Here a basic description of the GPUSPH simulator is provided. For a more detailed description of the simulator and its performance results we refer the reader to (Hérault et al., 2010a).

3.1 The GPGPU paradigm

Graphics processing units (GPUs) were developed to satisfy the need for fast rendering of three-dimensional images. As the requirements for their capabilities rose with the ever-growing request for more realism in games, so did their processing power.

In 2001, the introduction of the first *programmable shader* marked a turning point, by offering the possibility to insert custom functions in the previously fixed rendering pipeline of GPUs. This capability was soon exploited to harness the computing power of GPUs for purposes beyond rendering, thus marking the beginnings of GPGPU: General-purpose Programming on GPUs.

Coding for the GPU, however, still required extensive knowledge of the graphics stack, and sophisticated tricks to map arbitrary problems into something that would fit in the rendering pipeline, significantly affecting performance.

The growing interest for this kind of applications soon led the two major hardware manufacturers to ship products that allowed more direct and flexible control of GPUs, resulting in a break-through for GPGPU with the introduction of the NVIDIA CUDA (Nickolls et al., 2008) and ATI Stream (ATI, 2008) platforms in late 2006 and early 2007.

At the software level, the offers from NVIDIA and ATI were however significantly different. In particular, CUDA also describes the software stack offered by NVIDIA to assist programming its GPUs. This includes a host-side API and compiler that is familiar and comfortable for programmers with C/C++ programming experience, and a device programming language based on C++. The possibility to use a high level programming language interface was a key advantage of the framework when the OpenCL standard had not been released yet.

Many of the essential principles of GPGPU can also be applied to modern multi-core vector-capable CPUs. This has led to the birth of the OpenCL computing standard (Group, 2011) in 2008. Initially proposed by Apple and subsequently developed by a consortium including Intel, AMD, NVIDIA, IBM and others, OpenCL offers a host API and a C99-based programming language for multicore computing devices designed around the essential principles of GPGPU (kernels, blocks and threads – respectively called work-groups and work-items in OpenCL –, global and shared memory spaces, etc.), while remaining at a level abstract enough to allow execution on heterogeneous hardware.

For historical reasons, GPUSPH is entirely based on CUDA, but a port to OpenCL is being discussed for the future, to allow its deployment on a wider variety of hardware.

3.2 SPH on GPU

Direct per-particle computation of physical quantities and their derivatives ensures that SPH is easy to parallelize (one particle per thread), making it ideal for implementation on parallel architectures such as GPUs. One of the first GPU-based SPH solvers was developed by Amada (Amada et al., 2004), which offloaded force computation to the GPU while the CPU took care of other tasks such as neighbor search.

The first implementations of the SPH method completely on the GPU is due to Kolb and

Cuntz (Kolb and Cuntz, 2005) and Harada et al. (Harada et al., 2007). Their use of OpenGL and Cg (C for graphics) to program the GPU due to the lack of a better solution lead to some technical limitations that reduced the implementation performance.

The introduction of the CUDA architecture for NVIDIA cards in 2007 allowed to fully exploit the computational power of modern GPUs without the limitations imposed by having to go through the graphical engine. The first CUDA implementation of the SPH model (Hérault et al., 2010a) was developed at the Sezione di Catania of the Istituto Nazionale di Geofisica e Vulcanologia (INGV-CT) in cooperation with the Department of Civil Engineering of the Johns Hopkins University. It was inspired by the Fortran SPHysics code (Gómez-Gesteira et al., 2007) and has been published as the open source project GPUSPH (Hérault et al., 2016), used in applications ranging from coastal engineering (e.g. (Hérault et al., 2009; Dalrymple and Hérault, 2009; Dalrymple et al., 2010)) to lava flow simulation, e.g. (Bilotta et al., 2010; Hérault et al., 2010b; 2011). In the mean time, other CUDA implementations of SPH have emerged, e.g. (Goswami et al., 2010; Crespo et al., 2012).

3.3 Kernels

A GPU device is often referred to as device while the machine hosting the device as host. Device and the host have their own computational capabilities and memory and can somehow be seen as two separate computers. A function executed in parallel on several cores of a GPU is called *kernel*. The kernels are uploaded to the GPU from the host code, which triggers their execution and handles the transfer of the input and output data. The organization of the SPH method in CUDA kernels directly reflects the computing phases of the model:

1. **`BuildNeibs`** — for each particle, build the list of neighbors,
2. **`Forces`** — for each particle, compute the interaction with neighbors and its maximum allowed Δt ,
3. **`MinimumScan`** — select the minimum Δt among the maxima provided by each particle,
4. **`Euler`** — for each particle, update the scalar properties integrating over selected Δt .

All but the first kernel are executed twice for each iteration, due to the predictor/corrector integration scheme.

3.4 Fast neighbor search

During a single iteration the neighbors of each particle have to be accessed multiple times. It is thus faster to prepare a neighbor list once and iterate over it rather than searching for neighbors whenever they are needed. The naive $O(N^2)$ approach can be sped up by partitioning the domain in virtual cells with side equal to the influence radius, and sorting particles by a hash value computed as the linearized index of the cell they belong to. Neighbors

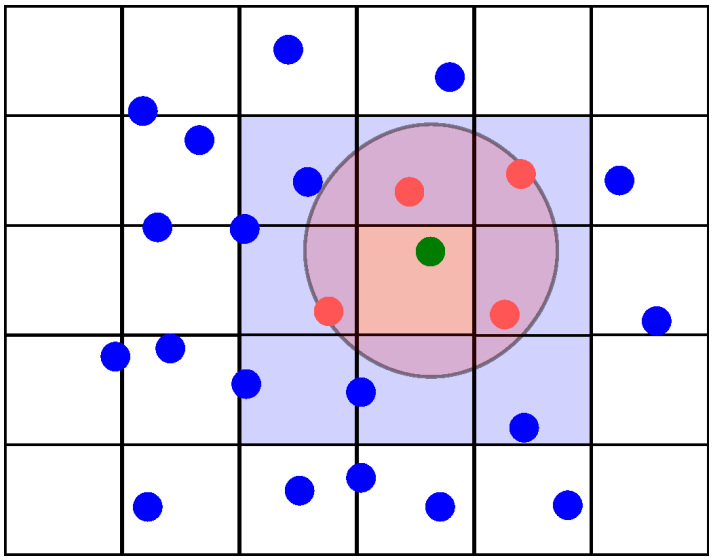


Figure 5: *If cells have side equal to the influence radius, neighbors of the green particle must reside inside in the immediate neighbor cells (light blue).*

of a particle can then be sought by only looking at the cells which are in neighboring *cells* (Green, 2010). Fig. 5 is a schematic representation of the virtual cells in 2D. We remark that this grid serves only to speed up the neighbor search, and has nothing to do with the SPH model.

Even with this approach, building the neighbor list for each particle still requires about 50 % of the total simulation time. As a further optimization we only rebuild the neighbor list every k-th iteration. With $k = 10$, this causes a negligible loss of precision while decreasing the time required for the neighbor list construction from 50 % to about 10 % of the total simulation time.

3.5 Multi-GPU

Although the implementation of the SPH method on GPUs already benefits from the computational parallelism offered by these devices, the memory and performance requirements of a high-resolution simulation often overcome the capability of a single device. This limit, together with the decreasing cost of medium to high level GPUs, naturally resulted in the extension of the simulator to an additional level of parallelism by featuring the possibility to exploit several GPUs connected to the same host machine simultaneously.

In GPUSPH, the simulation domain is split into partitions by means of the virtual grid of cells used for the neighbor search. The previous versions of the simulator only offered the possibility to split the domain across parallel sections. The data of the particles of every partition is then uploaded to a GPU device together with a copy of the particles situated in the cells neighboring the partition. These cells, referred to as *external*, need to be updated after every iteration as the contained particles and their values change throughout the whole

simulation. Moreover, the reduction performed to find the biggest allowable Δt for the next iteration must be performed not only on every device but also among the relative maxima of all the devices.

The design of a multi-device simulator must therefore face several complications including, to cite a few:

- the neighbors of a particle (and their scalar values) sometime reside in the memory of another device,
- the inter-device transfers needed to make the particles values available to their neighbors require a non negligible time, during which the computation of these particles cannot be performed,
- the overhead of transferring input and output data between devices and host multiplies with the number of devices,
- an uneven distribution of the computational load across different devices will result in a lower speedup,
- the timestep reduction must take into account (and therefore wait for the completion of) all the devices.

A crucial feature helping the programmer to overcome these complications is the possibility for the underlying hardware to perform duplex data transfers simultaneously with the computations, feature often referred to as *asynchronous computation*. This allows to transfer the particle values after they are ready while at the same time computing the values of other, non-neighboring particles, essentially "hiding" the overhead times due to the additional transfers with respect to single-GPU simulations (Fig. 6). It is mainly thanks to this feature that the implementation of multi-GPU capabilities in GPUSPH led to a performance speedup close to the theoretically optimal one, where the optimum refers to the maximum speedup obtainable according to the Amdahl's law (Amdahl, 1967).

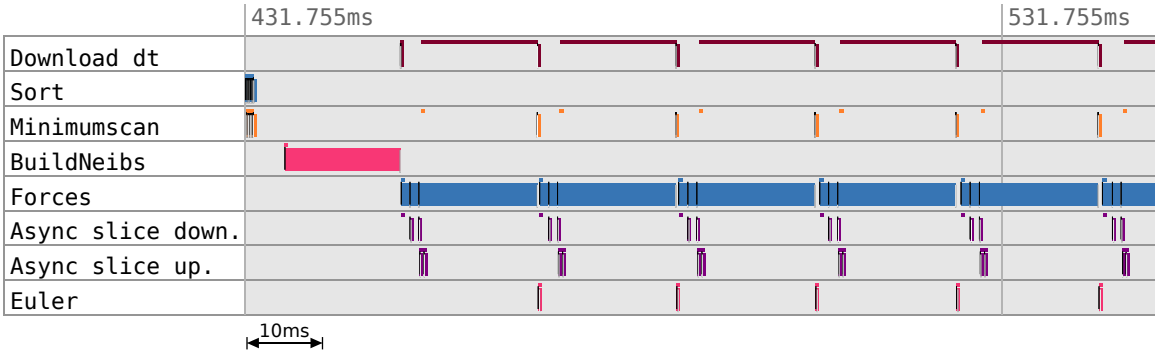


Figure 6: *Timeline obtained with a custom-made profiler for a visual representation of the kernel execution times and concurrency. Note the purple-coled data transfers are "hidden" by partitions of the forces kernel, hiding the transfer overheads.*

While simulations with strongly dynamically evolving topologies took benefit from the load balancing techniques implemented in the previous versions of the simulator, the majority of the simulation scenarios implemented in GPUSPH only featured a semi-static distribution of particles in space. For this reason, the later introduction of the support for splitting the domain along arbitrary geometries complicated the load balancing problem to an extent for which it was not worth the effort anymore (we recall that the problem of an optimal graph split with even node distribution and minimal number of edging nodes is NP-complete). A static load balancing performed in the pre-processing stage is now preferred.

Details about the techniques used to solve the mentioned problems and the implemented optimizations, together with the resulting performance, can be found in (Rustico et al., 2013).

3.6 Development at BAW

The first stage of the work at the BAW has been the completion of features and the implementation of additional ones to prepare the simulator to the needs of the project and to suit it the capabilities of the available hardware platform.

3.6.1 Multi-node version

At the time the project was being started, it was planned to extend the capabilities of the computing cluster at BAW by adding a set of GPU-based nodes (8 $\times$ BullX 515 *accelerator blades*, each carrying 2 $\times$ K20 NVIDIA GPUs). The hardware and software installation of the nodes was completed around middle 2013 but it was already in 2012 that the design and development of multi-node capabilities was realised. Adding a further level of parallelism required structural changes to the code, since:

- compared to the intra-node device-to-device transfers, the communication between devices attached to different nodes passes through a completely different layer of technologies exhibiting different speeds, latencies and asynchronous behaviour,
- the high-performance inter-node communication must be performed through a MPI (Message Passing Interface) implementation, which has specific requirements and takes over the main compilation toolchain,
- the domain partitioning must be aware of the network topology to optimize the split,
- many important details must be carefully taken into account such as the split of the output data in multiple files that is later possible to merge or open together in a post-processing tool.

The extension from multi-GPU to multi-node multi-GPU allowed to increase substantially the maximum number of particles allowed in a single simulation scenario. It is now possible to run a single simulation with over 100 millions of particles, although in the test cases later described the highest number of particles was about half this peak.

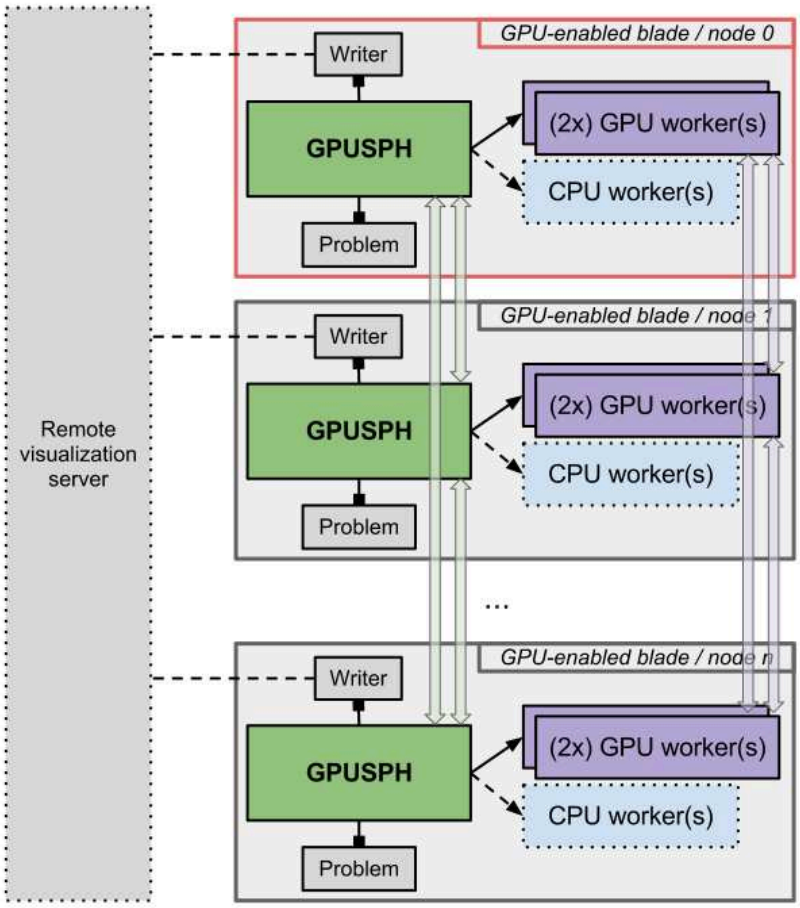


Figure 7: *Schematic representation of the simulator structure on a multi-node environment.*

Two optimizations implemented in the multi-node architecture are worth being mentioned:

- the option to use asynchronous transfers, similarly to the asynchronous computation techniques used to hide the intra-device transfers,
- the possibility to enable RDMA transfers with the underlying MPI library (i.e. passing device pointers to the MPI calls with no staging buffer), an advanced feature offered by a few MPI implementations.

Although the simulator was capable of running in any configuration of multiple processes per nodes, each one potentially handling multiple devices (e.g. 1 process per node, each using 2 devices, or 2 processes per node, each using 1 device), some difficulties were encountered in running a single process per node using multiple GPUs, in particular while RDMA was active. After extensive analysis it was discovered that although the underlying MPI library fully supported RDMA tranfers, it did not support using multiple devices in the same process. It was stated by the developers of the MPI library that this feature was too advanced and only required by GPUSPH, so it might not even be implemented in the future.

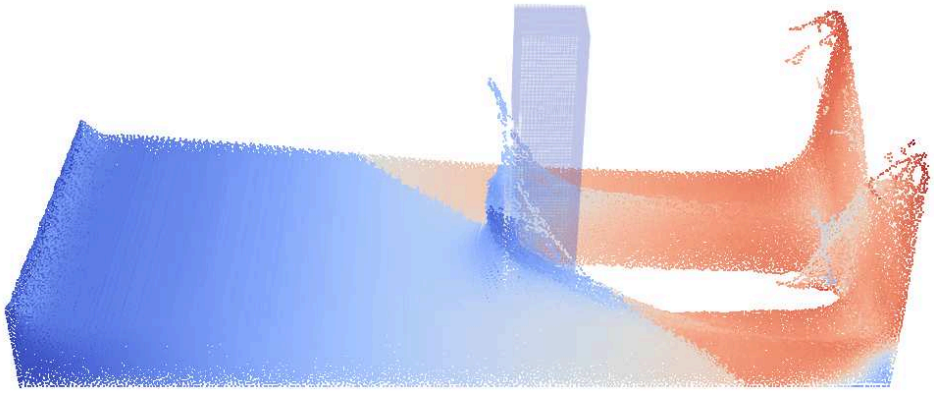


Figure 8: *Dam-break problem distributed across two devices by a plane which is not parallel to any of the axis. Particles are colored by device and shaded by velocity.*

As a result, GPUSPH can be run in multi-GPU multi-node as long as multiple processes are started in each node, each using only one GPU.

3.6.2 Arbitrary split

In the versions of the simulator presented in (Rustico et al., 2012; 2013), the domain split was further restricted to the granularity of slices of cells, orthogonal to the main direction in the cell index linearization, in order to ensure that edge cells were stored consecutively and could therefore be transferred more efficiently. For multi-node simulations, which can be potentially distributed across tens or even hundred of devices, such a restriction would have severely limited the applicability and efficiency of GPUSPH, in particular for simulations featuring fluid topologies displaced in such a way that a balanced linear subdivision is not feasible. The restriction was then lifted, and the support for any domain decomposition at the granularity of individual cells was implemented. The algorithm used for the initial domain split can be user-defined and some common algorithms based on parallel and orthogonal splits are also offered as pre-defined options (see Fig. 8).

Arbitrary splitting comes at the cost of some additional memory consumption, since the particle system was augmented by a `DeviceMap`, which encodes the index of the host node and GPU to which each cell is assigned.

While splitting along parallel planes guaranteed that all the external cells to be transferred from/to different devices were consecutive in memory and thus readable with a single transfer, with the arbitrary domain decomposition external cells can be sparse to some extent. To keep the efficiency of data transfers and avoid penalizing the performance of particular decompositions, external cells with the same recipients are sent in compact bursts. For more details about the implementation of the arbitrary decomposition and the techniques applied please refer to (Rustico et al., 2014).

3.6.3 XProblem interface

Among the corrections and renovations the GPUSPH source code underwent in the past three years, some of them did not concern the functionality of the simulation engine but rather its interface with the users and programmers. A problem scenario was typically defined by extending a C++ class named `Problem` and implementing all the necessary virtual and optional methods. Loading a problem definition from file was excluded because of the many variants and computation options offered by GPUSPH, most of which need a compile-time option for performance optimization and cannot be loaded at run time. The implementation of such class was not trivial because of the many advanced details that had to be explicitly implemented, such as the generation of particles in space, their grouping in boundaries and floating objects, the assignment of IDs and their full initialization. The Problem class was then restructured. To keep compatibility with previous problems, a new class `XProblem` was defined as child class of `Problem`. Newly defined problems can directly descend from the latter.

The new class led to many advantages for the programming interface and thus for the final user, including:

- more compact scenario definition (e.g. the ship lock problem was shrunk from about 2 thousands lines by a factor of ten),
- more flexibility in the run-time options (e.g. changing from one boundary model to another does not require rewriting entire methods anymore),
- less prone to errors thanks to compile-time notifications of incompatible combinations of options,
- less pre-processing needed for minor changes in the problem definition (e.g. translating or rotating a mesh is done by means of one function call and does not require remeshing the original data),
- clearer distinction between problem parts by the introduction of the `Geometry` entity, whose type and scalar fields values can be changed with one call,
- powerful geometry definition mechanism allowing even “hidden” entities solely defined to unfill fluid particle or punch boundaries.

The new problem interface simplified the definition of new problem scenarios and accelerated the development as well as the debugging of test cases or a practical project.

3.6.4 Other features

Additional features where implemented to address specific needs of the evaluation experiments. They are descrided as follows:

Pseudo-inlets and outlets. The Fishpass test case, later described in detail, was modeled using Lennard-Jones boundaries, which do not feature open boundaries. However, the possibility to control the exact inflow rate was fundamental for a proper physical simulation. A *pseudo-inlet* was then implemented, providing inflow control through the capability of generating fluid particles and imposing their velocity until the entrance in the fish pass structure. The initial implementation only provided the generation of particles and a velocity field; particles leaving the field underwent a violent discontinuity where a high density aggregation of particles evolved near the inlet, causing instabilities and pressure singularities. To solve this problem, the inlet field was divided into two virtual halves, the first consisting in a linear velocity field generating a particle for each particle leaving it, the second applying a gradually fading velocity field with null intensity at the end.

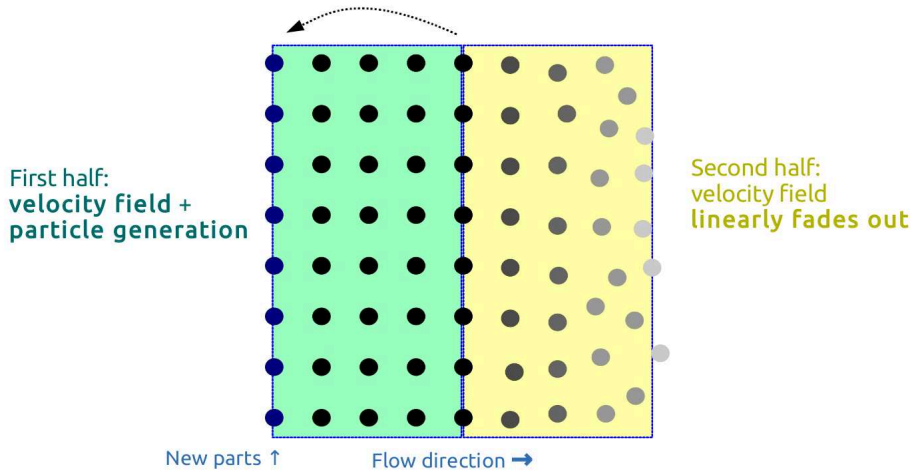


Figure 9: *Section of a pseudo-inlet; the first part generates new particles and applies a velocity fields, the second part gradually releases the field.*

The pseudo-inlet does not provide any pressure nor density correction and it is only possible to impose the particle velocity (and therefore the flow rate). These fields are let to evolve throughout the simulation and slowly adjust their initially artificial values; for this reason, this type of inlet is not recommended in problems where an accurate pressure field near the open boundary is determinant for a correct physical simulation.

The pseudo-outlet simply consisted in a rectangular field destroying any particle entering it; it was only used at the end of the fish pass where particles have super critical speed (after a small waterfall) to avoid pressure shocks and discontinuities in general.

Frictionless dynamic boundaries. In the later described ship lock test case, the drag force on the ship appeared to be influenced by the friction between the water and the dynamic boundaries. Frictionless boundaries have been developed, ignoring the tangential component of the force resulting from the interaction with the boundary, to quantify the effect of the friction. Since dynamic boundary particles are treated like water

particles, an alternative version was also implemented ignoring the viscous component of the interaction force, to ensure that the influence of the water viscosity parameter was not stronger than the friction merely due to the aforementioned tangential force.

Both variants were first tested on a classic DamBreak3D test case and then applied to the ship lock; in both cases, the influence of the friction resulted not to be determinant in the computation of the forces on the ship.

Mesh unfill For ship placement. GPUSPH featured from the beginning an "unfill" operation capable of erasing fluid particles where solid obstacles are placed. If, for example, a simulation scenario features a column on a wet floor, first the water parallelepiped will be filled with water particles, then the column will be filled with boundary particles and eventually the water will be we "unfilled" in the volume overlapping with the column (water particles positioned within the column volume will be erased before the simulation begins). This operation was only implemented for solid primitives, for which it is easy to assess if an arbitrary point lies inside or outside their volume. The same problem is less trivial for arbitrary shaped meshes, even if we reduce the set of possible meshes to rectangular, well-connected, closed ones. For the tests using Dynamic boundaries, and thus the mesh of a ship loaded in form of a three-layered point cloud, a dedicated unfilling technique was developed for the ship lock problem. The technique consists in three steps:

1. Unfill a sphere with radius dp around every particle of the ship hull.
2. Unfill a particle in the geometrical center of the bounding box of the ship, which lies inside the hull, and all its neighbour fluid particles.
3. Repeat step 2 for every fluid particle directly encountered; the process will automatically stop when the edges of the hull are encountered (since they were unfilled from water particles in step 1.

The technique allows to move the ship mesh at a different position at run time before the simulation begins; it is not used in the Semi-Analytical version of the test case, that needs instead an already unfilled set of particles produced with an external preprocessing tool (Mayrhofer, 2016).

Support for point clouds. It has been mentioned that dynamic boundaries require three layers to fill the support of any particle getting closer to them. While populating these three layers is trivial for simple regular solids (spheres, parallelepiped, cylinders...), the operation is much more complex to be abstracted for arbitrary shapes. A mesh requires being shrunk along the face normals, while keeping a constant boundary particle density and avoiding self-intersections. This operation has been performed with external software tools specialized in mesh analysis and modeling such as Meshlab (Cignoni et al., 2008). A XYZ file format reader has been implemented in GPUSPH to enable loading three-dimensional point clouds resulting from the elaboration of such tools. GPUSPH can load a point cloud either as fluid or as boundary particles. The

reader preserves the normals of the points being loaded, which are not currently used but might be used in the future to refine the unfilling techniques.

Floating objects with joints. The simulator initially used the external library ODE (Smith, 2014) to handle body dynamics and collisions. The ODE library suffered a few problems:

- Its support was discontinued by the developers.
- Resuming an interrupted simulation led to different numerical results due to some internal recomputations in the library.
- There was limited support for complex joints between bodies.

Because of these limitations, the subsystem handling the body dynamics was ported to the Chrono library (Mazhar et al., 2013), which is actively developed, offers high accuracy and a wide range of different joint types.

Fig. 10 shows a simulation scenario of a floating platform for the study of energy-generating devices, courtesy of the Costal Studies Institute of the University of North Carolina, simulated with the use of the Chrono library in GPUSPH.

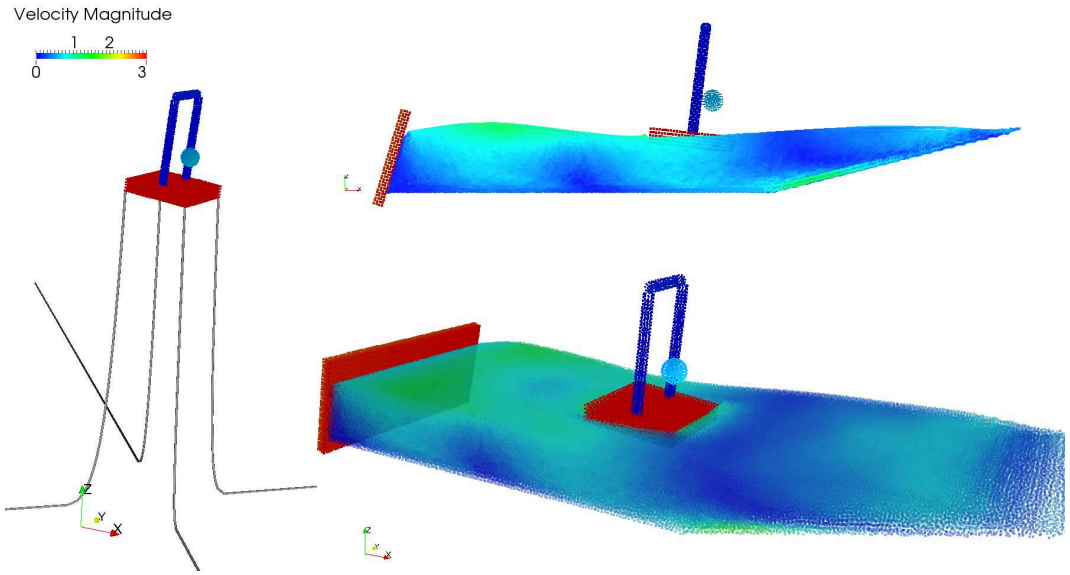


Figure 10: *Floating platform for the study of energy-generating devices, courtesy of C.S.I., University of North Carolina; the platform generates energy from the rotational moment of the pendulum, swinging between two fixed poles, and it is anchored to the sea bottom to avoid overturning while retaining all degrees of freedom. Moorings are simulated by dedicated functions.*

One limitation introduced by the Chrono library is the absence of the Torus and Disk geometrical primitives, which have been discontinued in GPUSPH as well. These solids can of course still be simulated if loaded from an external mesh file.

4 Test cases

4.1 Fish pass

4.1.1 Introduction

The first test case was a vertical-slot fish pass. Hydraulic research on vertical slot fish passes has shown that the hydraulic conditions within the pools of such facilities are mostly determined by the pool geometry and the slope of the fish pass. While previous SPH-based approaches to the numerical simulation of fish passes were mostly in 2 dimensions, the use of SPH allows to run fully three-dimensional simulations, without constraints on the domain shape and with great accuracy. This comes at the cost of significant requirements in terms of memory and computational power, so that only low-resolution attempts have been done so far (Violeau et al., 2008; Marivela, 2009).

To cope with this, we have extended GPUSPH, which previously supported single-node multi-GPU computing, to exploit the computational power of clusters of graphic devices. By exploiting 12 devices across 6 nodes simultaneously we have been able to run a fine-grained simulation with a resolution of about 4 mm per particle and a total of 50 millions particles.

4.1.2 Model

A laboratory model of the modeled fish pass has already existed in the laboratory of the Federal Waterways Engineering and Research Institute (BAW) as part of ongoing R&D activities.

The laboratory model consists of 9 pools with width of 78.5 cm and length of 99 cm installed in a flume. The slot width is 12.2 cm; the slope of the flume is 2.8 %. The side walls and the bottom of the flume are made of plexiglass, the cross-walls and the baffles of the model are made of wood.

The SPH model was designed to reproduce the same geometry, inlet and outlet conditions of the laboratory model. Influx rate, sizes, wall offsets, initial filling and other geometric details are fully parametrized to allow for fast comparative tests and efficient maintenance in case of changes in the physical model (Fig. 11).

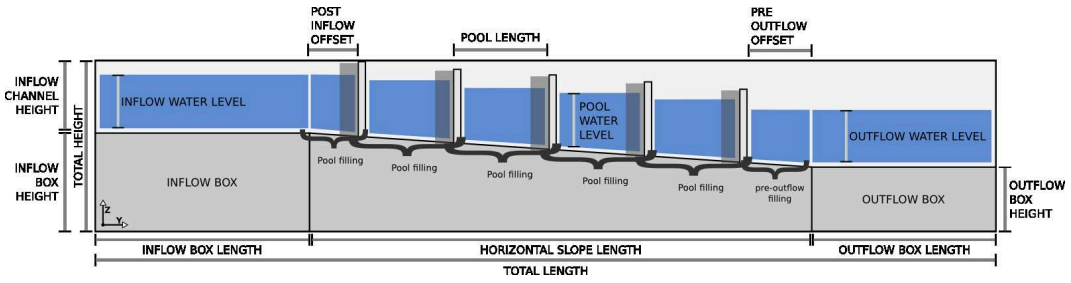


Figure 11: *Schematic description of the SPH model of the fish pass.*

The physical boundaries of the domains are modeled with a combination of Lennard-

Jones repulsive particles and Lennard-Jones repulsive planes. Planes are used to model the side walls of the entire fish pass, as they yield no friction to the fluid stream, and their usage reduces the total number of particles required for the simulation. However, since the current implementation only allows infinite planes, they cannot be used to model other parts of the fish pass (inner walls and floor), for which the standard Lennard-Jones repulsive particles are used instead, with a linear density which is twice as high as that of the fluid, in order to reduce the artificial friction introduced by this kind of boundaries.

Inflow is modeled by reserving a section (inlet) at the beginning of the domain for particle generation. These particles are generated with an initial velocity such that the inlet achieves the same inflow rate as the physical model. The "pseudo-inlet" described in section 3.6.4 has been used to model such inlet.

A ramp at the end of the outflow channel is used to model the flap gate, which was not moved for this test case but was set to impose the desired water level at one end of the pass. In the computation model, an outflow field is placed right after the freefall (Fig. 12). The only task of this "pseudo-outlet" is to destroy the particles that fall into it.

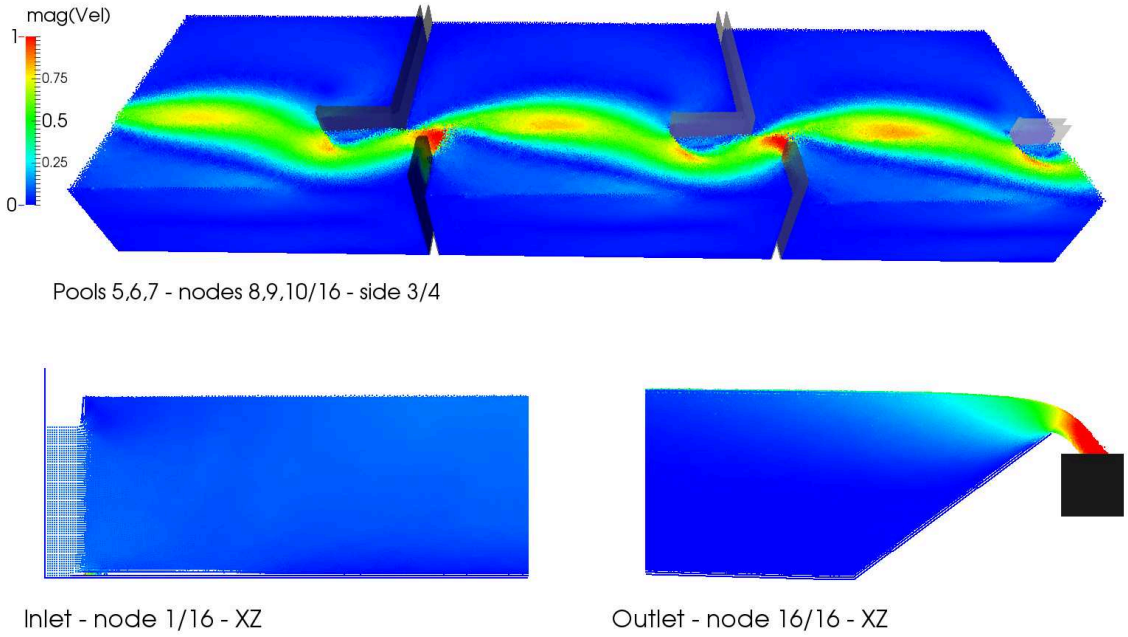


Figure 12: *Three pools, inlet and outlet of a 50M particles simulation.*

Water levels are captured with special "gage particles", which are floating particles with fixed X and Y coordinates. Two gages are set in each pool in the same positions where the water level are measured in the physical model.

4.1.3 Validation and laboratory measurements

The flow velocity measurements were carried out in the sixth pool of the model 12 cm above bottom level along 134 gridpoints by an Acoustic Doppler Velocimeter (ADV) of type side-looking Vectrino with 200 Hz sampling rate.

For the validation of the flow velocities, velocities were captured at the same gridpoints in the numerical model as in the laboratory model. The flow velocities were recorded for every time-step in the numerical model. The obtained velocity time series show fluctuations, so time averaged values over 10 seconds were used during the comparison for stationarity reasons. The measured and modeled velocity fields are presented in Figure 13. It can be observed, that the main flows interconnecting the slots have a similar shape and the magnitudes of the velocities show a good agreement.

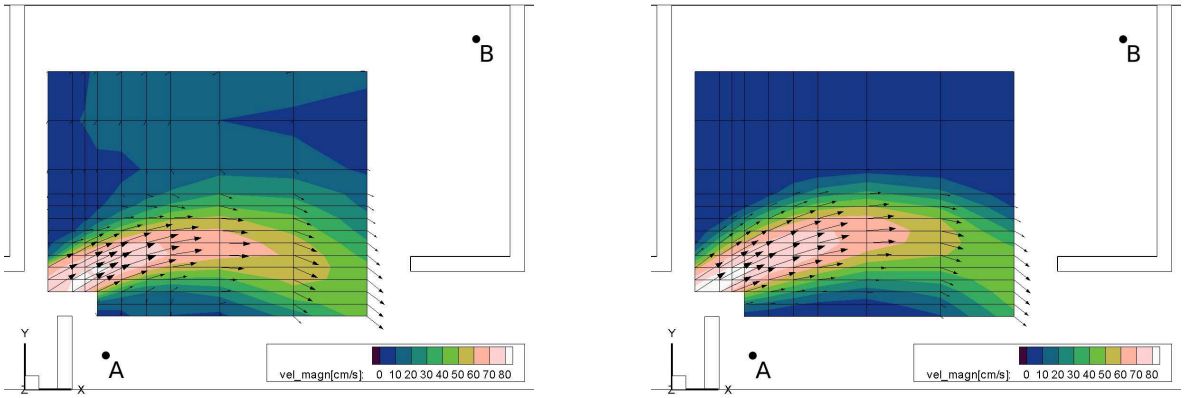


Figure 13: *Velocity field measured in the sixth pool of the laboratory model (left) and captured in the sixth pool of the numerical model (right).*

The water depths measurements were performed in points A and B in each of the nine pools by ultrasonic water level sensors with a sampling rate of 40 Hz. Water depths in points A and B nearly represent the minimal and maximal water depths within the pools, so that the average of the two values was used as the characteristic water depth in the pool. The water levels captured in the numerical model were transformed to water depth values, and were then averaged over 10 seconds to get statistical stationary values.

The water depths obtained from the measurements and the numerical model are presented in Figure 14. It can be observed that the water depths in the numerical model are higher than in the laboratory model, with a larger difference close to the inflow. This indicates that our SPH model produces more hydraulic loss than the laboratory model, which is probably due to a combination of boundary effects, such as the wall friction and inflow conditions, as well as to the non-physical viscosity applied in the model.

4.1.4 Results

The stream shape shows a good agreement with the experimental data both in the overall velocity and in the local recirculation areas (Fig. 15). The velocity profile shows slightly slower

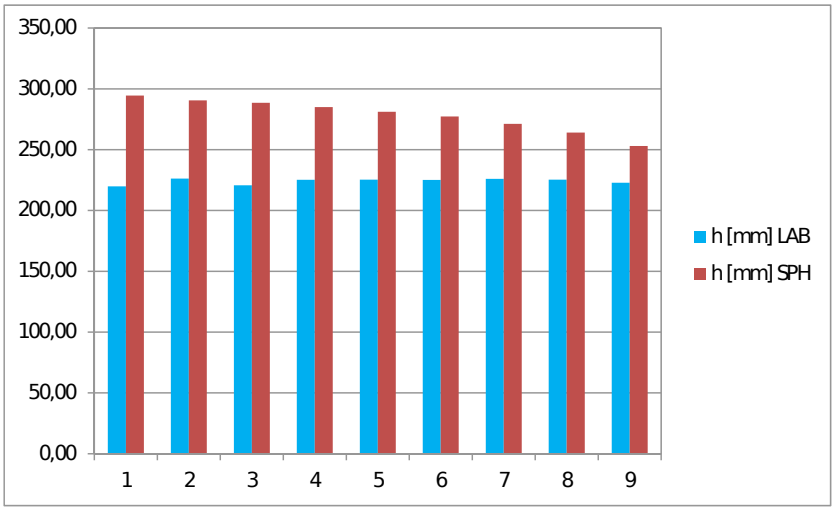


Figure 14: *Characteristic water depths in the pools in the laboratory model and SPH.*

speed especially in contact with the boundary, near the walls and in between the passes. The peak speeds are comparable but are influenced by the high friction caused by the simple boundary model. The loss in kinetic energy affect the water levels, that are increasingly higher than the measured ones the farther we move up from the outlet.

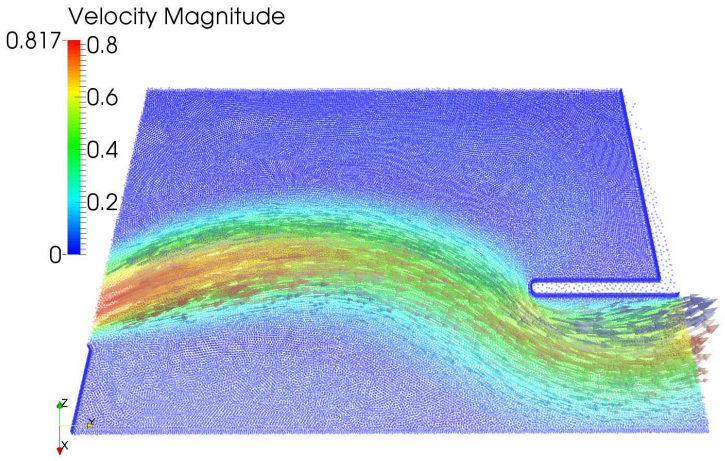


Figure 15: *Cross section of a pool of the same simulation.*

To reduce the energy loss of the fluid, a first attempt was done by reducing the viscosity, which led however to an excess of noise at the water surface. Further tests with a more physical boundary model featuring less friction and turbulence (Ferrand et al., 2013; Mayrhofer

et al., 2015) were planned. This boundary would also allow better modeling of inlet and outlet conditions. The tests require porting the model to the new boundary and were scheduled to be performed after its implementation would have been completed.

4.2 Ship lock

4.2.1 Introduction

The large ship locks of the Kiel-Canal underwent a significant renovation and it is planned to adopt a through-the-gate filling system. The Waterways Engineering and Research Institute (BAW) was commissioned to evaluate the filling and emptying times of the new system and its impact on the forces acting on the ships, which might undermine the safety of the transit. Figure 16 is an aerial view of the lock complex.

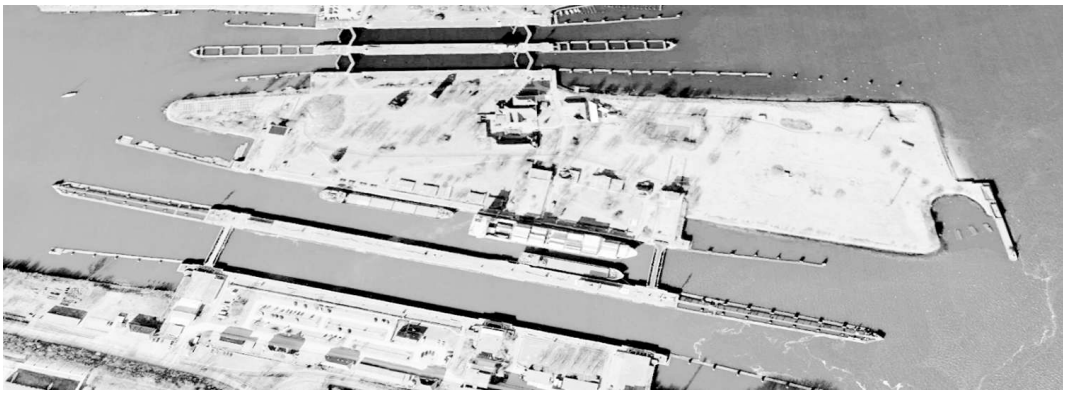


Figure 16: *The Kiel Holtenau Lock Complex.*

In (Thorenz and Anke, 2013) a scaled physical model and a purely numerical approach are used to simulate the filling and emptying of the lock. The numerical model used a Volume-Of-Fluid (VOF) Eulerian approach and was in acceptable agreement with the laboratory measurements. The relative movement of the ship with respect to the lock chamber was simulated by grid-morphing or by defining an extra mesh containing the vessel and moving the latter with respect to the background one. The study concluded underlining the relevance of the transversal forces acting on the ships for the planning of the schedule of the valves, especially in extraordinary load conditions. However, it is stated that the most complete way to simulate the behavior of a lock is still a scaled physical model, as the numerical modeling showed several difficulties in respect to the movement of the ship.

SPH is particularly interesting for this application for its support for floating objects, which allows us to model the ship in the basin and its fully coupled interaction with the water during operation.

4.2.2 Model

Size and resolution

The ship lock has been modeled in 1:1 proportion. Figure 17 shows the ground plan of the lock while Figure 18 is an overview of the SPH model. The main chamber is 330 m long and 45 m wide. The smallest geometrical size is the height of the small gate channels (1.3 m). We used a linear particle size of 0.2 m in order to fit at least 5 particles in the shortest geometrical size. The initial amount of water is 12.5 m $\times$ 45 m $\times$ 330 m $= 185.625\ \mathrm{m}^3$, which equals to about 23 millions particles. The total amount of fluid particles does not overcome 30 millions including the fluid being added in the filling process.

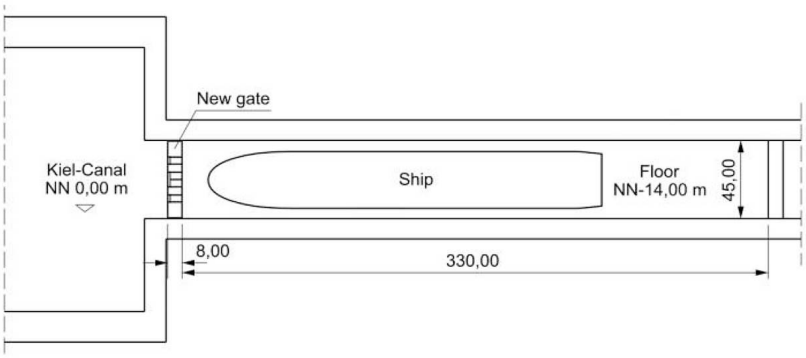


Figure 17: *Ground plan of the lock complex.*

Floating ship

Thanks to the mesh-less, Lagrangian nature of SPH, fluid/solid interaction and thus the modeling of floating objects is quite straightforward in SPH, compared to e.g. VOF or other mesh-based methods. GPUSPH uses the thin-shell approach to floating objects: the object is modeled as a thin layer of boundary particles. Any boundary model implemented in GPUSPH can be used for floating bodies. The interaction of these particles with the fluid particles is used to compute the total force and torque acting on the object, which is then used in the integration phase to determine the motion of the object (as a rigid body), and consequently correct the position and velocities of the particles that model its boundary.

In GPUSPH the actual motion of the rigid body is delegated to an external library dedicated to handling the body dynamics and collisions; the force applied from the fluid to the boundary is input to the library and the resulting movement and rotation is read and used to update the position of its particles. GPUSPH used the open-source Object Dynamics Engine library (Smith, 2014) up to version 3.0; GPUSPH v. 4.0, release in October 2016, uses instead the Chrono library (Mazhar et al., 2013), but was not used for the test case.

4.2.3 First test: cylinder, pseudo-inlet

The first feasibility test was conducted when the simulator did not support yet loading arbitrary meshes but only a combination of a set of solid primitives. A cylinder has been used to approximate the shape of the ship hull. Since Lennard-Jones boundary particles do not allow us to compute physically correct forces in the case of slow dynamics, for the cylinder we have used the computationally more expensive dynamic boundary approach described in (Crespo et al., 2007); the cylinder is filled with three layers of dynamic boundary particles, since we use a Wendland kernel of radius 2 and smoothing coefficient 1.3; the standard SPH discretization of the Navier-Stokes equations is used to compute the forces acting on these particles, and thus ultimately the force and torque acting on the ship, whose motion then prescribes the motion of the boundary particles, as mentioned.

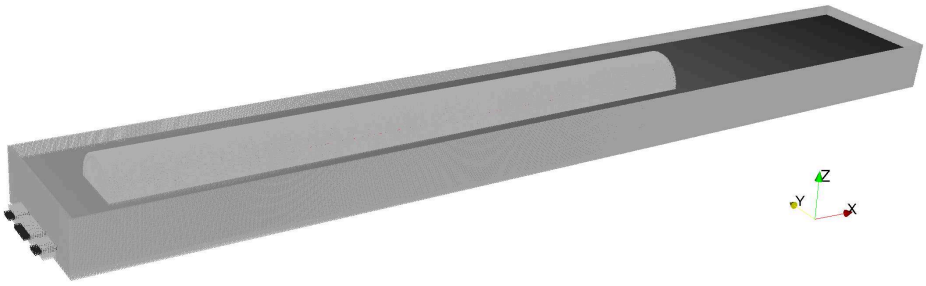


Figure 18: *Overview of the first test.*

Instead of modeling and allocating a tank of water behind the channels, three fluid inlets capable of particle generation have been placed inside the three channels. This allows for halving the number of particles to be simulated and thus decreases the memory requirements and the simulation times. The pseudo-inlet that can impose a time-dependent velocity but not a pressure value. The inflow rate computed in (Thorenz and Anke, 2013) has been used to compute the velocity over time; it raises linearly up to a maximum of 74 m^3/s, reached at 228 s, and then decreases linearly again to 0 m^3/s in about 392 s. Before the inlets start to generate fluid particles, 60 seconds of simulation are run as settling time; this has been estimated as a period sufficient to let the Z force of the ship stabilize.

The movement of the ship, in this as well as in the following tests, has been constrained to resemble the tests described in (Thorenz and Anke, 2013). It is free to move only along the Z axis and to rotate only around X and Y axes. The force and torque contributions applied by the fluid to the remaining axes are ignored.

Results

The simulation took 72.5 hours, on a cluster of 16 GPU devices distributed across 8 nodes, for 680 s of simulated time (60 s settling plus 620 s of fluid injection). In this preliminary test we focused our attention on the shape of the injected stream and on the longitudinal forces acting on the ship.

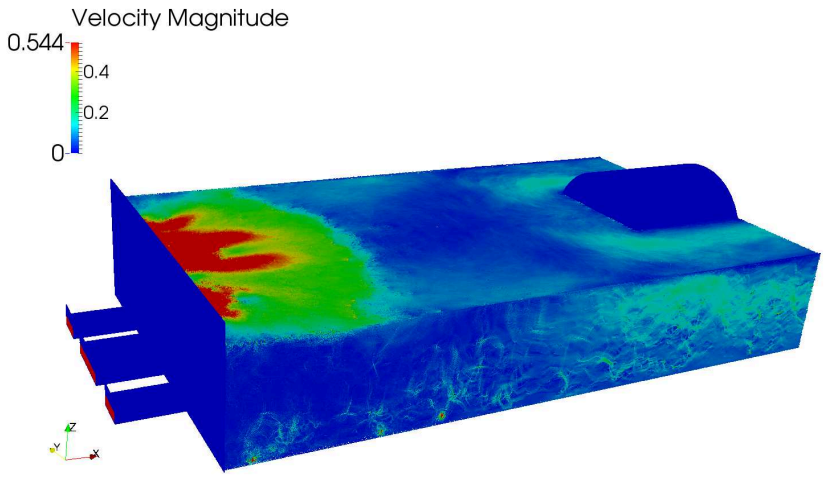


Figure 19: *Overview of the inlet part of the simulation in a higher resolution test (about 30M particles).*

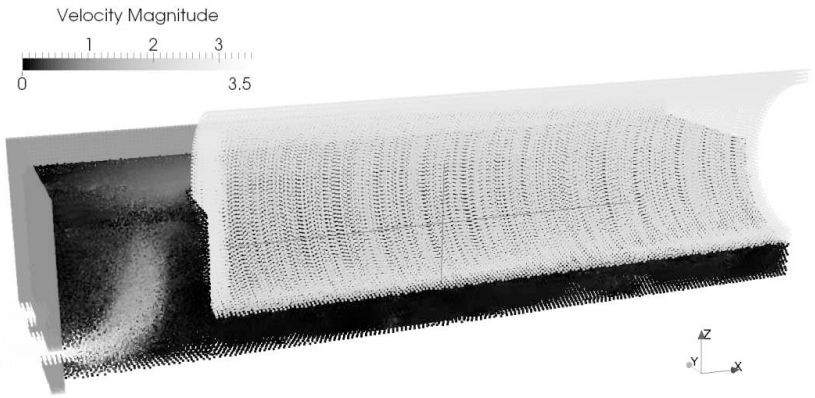


Figure 20: *Section of the inlet part showing the flow pattern at $t = 240$ s.*

In a first phase, the injected particles tend to move towards the fluid surface soon after being injected (Figure 20). This causes a vertical recirculation in front of the cylinder and, after the peak velocity is reached, the fluid starts to flow under it. We have currently no specific measurements to assess if a similar phenomenon is present also in laboratory experiments but we believe the stream behavior is anomalous. This behavior could be attributed to the experimental nature of the inlets, where we can impose velocity but not pressure, and the non-hydrodynamic shape of the cylindrical "ship", which opposes a significant resistance to the inlet stream.

The main difference with respect to the values measured in the laboratory tests is the trend of the longitudinal force acting on the ship. The comparison plot is shown in Fig. 21.

The force measured in laboratory is positive for the first 180 s of the experiment, with a peak value of about 45 kN, and drops to almost -100 kN in the next 100 s. The sign change is due to the main stream passing under the hull of the ship and pushing from the back of the chamber. Forces in the SPH simulation, instead, exhibit an always positive trend, with a

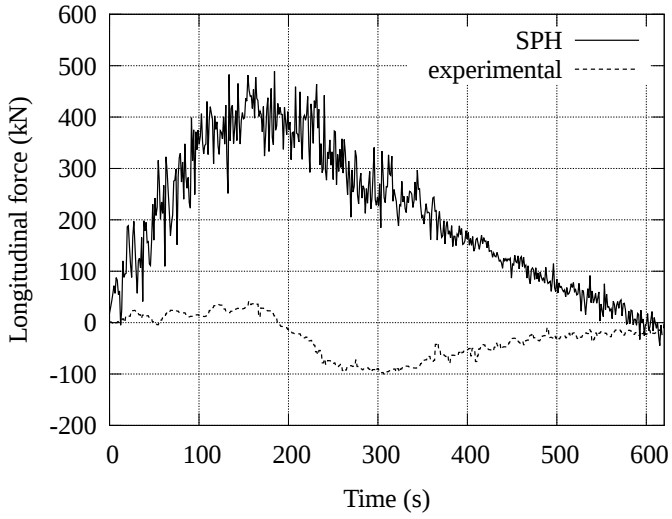


Figure 21: *Longitudinal forces in the model and in the laboratory experiments. A positive longitudinal force, by convention, pushes the ship away from the inlet channels.*

peak force of about 400 kN achieved near the peak positive force of the experimental data; it then slowly decreases to zero without going negative. The difference in magnitude and sign between the laboratory and SPH force measurements is explained by the difference in the stream of inlet fluid and the less hydrodynamic profile of the cylinder with respect to the actual shape of the ship, which plays a key role in hindering the transit of the main stream to the back of the chamber. The measured force, both in the physical experiment and in the SPH simulation, is the sum of all the forces acting on the ship. However, while in the experimental set-up the stream passes around and below the ship and the forces partially cancel out, in SPH, where the stream shape is different, the forces are focused on the front of the ship and do not cancel, resulting in a larger overall forces, always with a positive sign.

4.2.4 Second test: Semi-Analytical boundaries

A second test was performed by re-modeling with the recently developed Semi-Analytical boundaries. The implementation of this boundary model was not complete yet when the test was performed, but a sufficient portion of its features was already usable for an evaluation test.

Both the lock walls and the ship have been modeled and loaded as triangular mesh. The mesh connectivity and specific particle properties have been set in the preprocessing phase with the software Crixus (Mayrhofer, 2016). The inlets have been transformed in pressure inlets, simulating a constant water level behind the culverts.

Unfortunately, the slow performance of the Semi-Analytical boundaries did not allow a complete simulation, which would have taken more than a month (we recall that usually more than one run is required to check the simulation settings and the evolution of simulation). The first, partial run showed a jet with a clearly more straight jet stream, which is

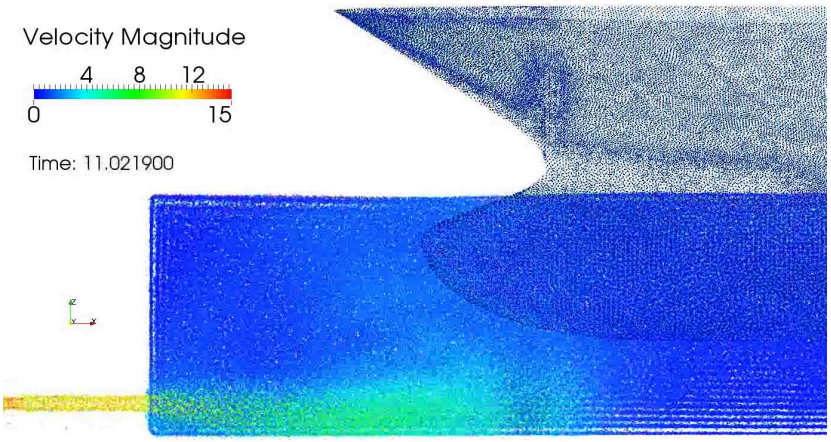


Figure 22: *Section of the lock with Semi-Analytical boundaries in the second test.*

however still not completely undergoing the ship. The fluid particles near the hull boundary are faster and show a smoother profile but it was not possible to complete all the analysis for a proper evaluation of the forces.

4.2.5 Third test: dynamic boundaries, point cloud

For the third test, the dynamic boundaries were again chosen but this time the new GPUSPH features that make it possible to load point clouds was exploited to reproduce the actual ship shape and not approximate it with a cylinder. A point cloud representing three layers of boundary particles was produced from the ship mesh with the Meshlab software (Cignoni et al., 2008). The layers are δp distant from each other and proportionally shrunk. The conversion between the triangle mesh and the points is done with a Poisson sampling algorithm. The cloud point is then loaded into GPUSPH as floating object.

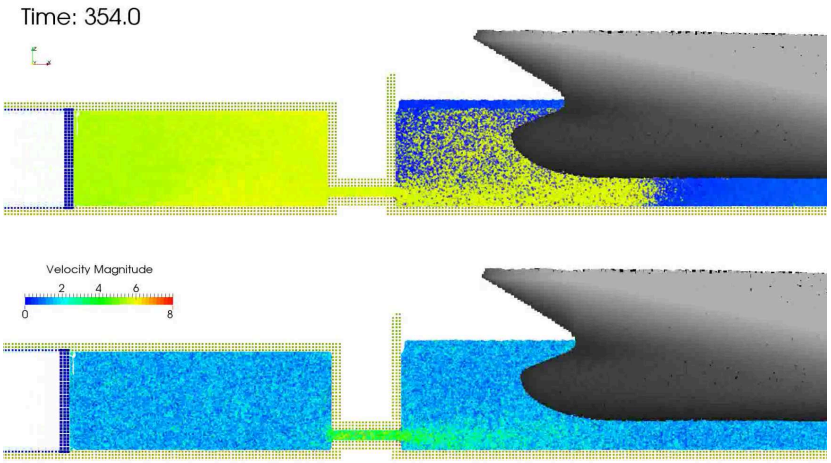


Figure 23: *Section of the lock with dynamic boundaries and piston (third test).*

Since dynamic boundaries do not feature a physical inlet and the pseudo-inlet was one of the possible causes of the anomalous jet, a different approach has been chosen for the inflow: a reservoir chamber contains the water estimated to fill the lock up to the desired water level; a piston pushes the water into the lock chamber with a velocity computed according to the desired inflow rate.

The shape of the jet showed a clear improvement, although it was not yet straight. The forces acting on the ship are comparable to the ones of the first test, meaning that the shape of the jet is not the principal factor affecting the push on the ship. There appear however to be still a high friction with the water. Up to our knowledge, this was the first time such boundary model was used for non-trivial geometries; it is therefore also possible that its interaction with non-flat surfaces is affected by some kind of artifacts.

The test also suggested a connection between the jet stream shape and the speed of sound, a SPH parameter affecting the compressibility of the fluid and the timestep (thus the computational performance). Increasing the speed of sound by a factor 2 or 4 appears to be beneficial for the straightness of the jet, without bringing an advantage that would justify the proportionally increased computational times.

It would be recommended to run the test with a much higher resolution, currently very challenging without variable resolution nor coupling with another model, both with dynamic and Semi-Analytical boundaries.

4.3 Other test cases

4.3.1 XVases

The aim of the `XVases` problem was to simulate two communicating vessels and analyze the jet resulting from a mere water level difference, with no inlet artificially generating and pushing fluid particles. The test showed that the jet problem can occur with a free surface problem but only if the water level difference is very small compared to the absolute levels. This causes the jet to have a small velocity and it is likely that this reduced push is not able to contrast the SPH discretization artifacts and the hydrostatic pressure gradient. A larger difference in the water level causes a stronger push and lets the jet run perfectly horizontally and symmetrically (Fig. 24).

The communicating vessels were simulated with different sizes and resolutions. Similarly to the ship lock test case, we do not own laboratory measurements to compare to.

4.3.2 XInjection

`XInjection` is a test case deriving from `XVases` with a different setting. There are three chambers, vertically stacked, connected through holes in the separating walls. One chamber contains a piston and injects water in the second; the second chamber lets water pass in the third; the third chamber lets the water overflow freely. The aim of the `XInjection` problem was to evaluate the influence of the orientation of a piston on the shape of the injection stream; radically different stream shapes or directions would have suggested the

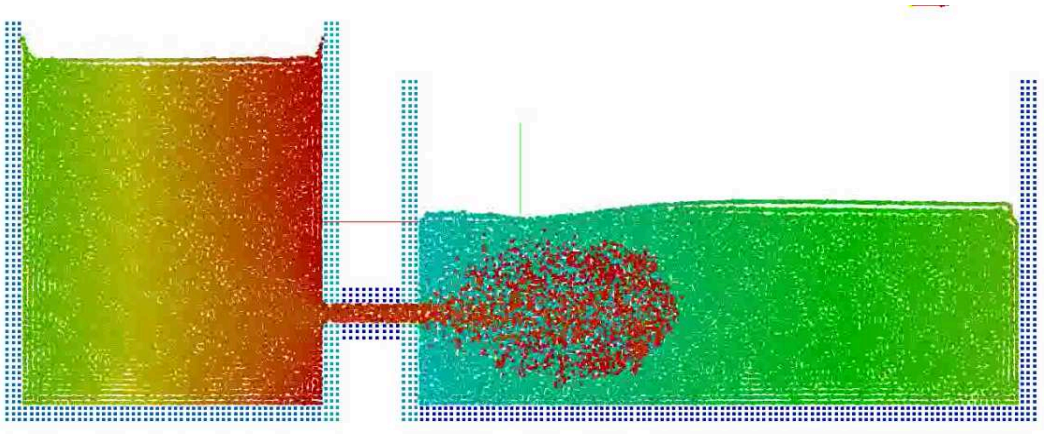


Figure 24: *Section of the XVases (communicating vessels) test case with horizontal jet.*

existence of an intrinsic problem in the SPH model or in the GPUSPH implementation rather than a setting or resolution problem confined to the ship lock test case.

It is possible to run the `XInjection` in two different configurations: with gravity parallel or perpendicular to the piston direction. The first configuration reproduces the injection of 1 m^3 cubic meter of water into a 1 m^3 chamber through a round hole ($\oslash$ 0.2 m) on its bottom wall. Another hole is placed in the center of the top wall. The jet is directed upwards, opposite to gravity, and the water overflows freely 0.5 m above the hole level. The second configuration uses the same geometry with two modifications:

- the top chamber is closed on the top and in three of the sides,
- gravity is directed towards the opposite side of the open wall of the top chamber and it is thus orthogonal to the jet direction.

Note that the configurations are designed to present the same distance between the center of the top hole and the free surface from which the water flows out (0.5 m).

The first configuration was used as reference configuration, where the jet is naturally symmetric. The second configuration reproduces a scenario more similar to the water inflow in the ship lock, as the injection is perpendicular to the direction of the gravity. Fig. 25 shows a section of the two configurations shortly after the simulation has started. The jet is generally symmetric in both configurations; in the second configuration, it bends towards the free surface only in an advanced stage of the simulation, when a movement towards the free surface is necessary because of the increasing water level.

The results showed that the orientation of the piston affects the shape of the jet only to a small extent; its influence is weaker than the influence of other factors such as the speed of sound and above all the resolution of the simulation. No evident problem in the model nor in the implementation can be deduced from this simulation.

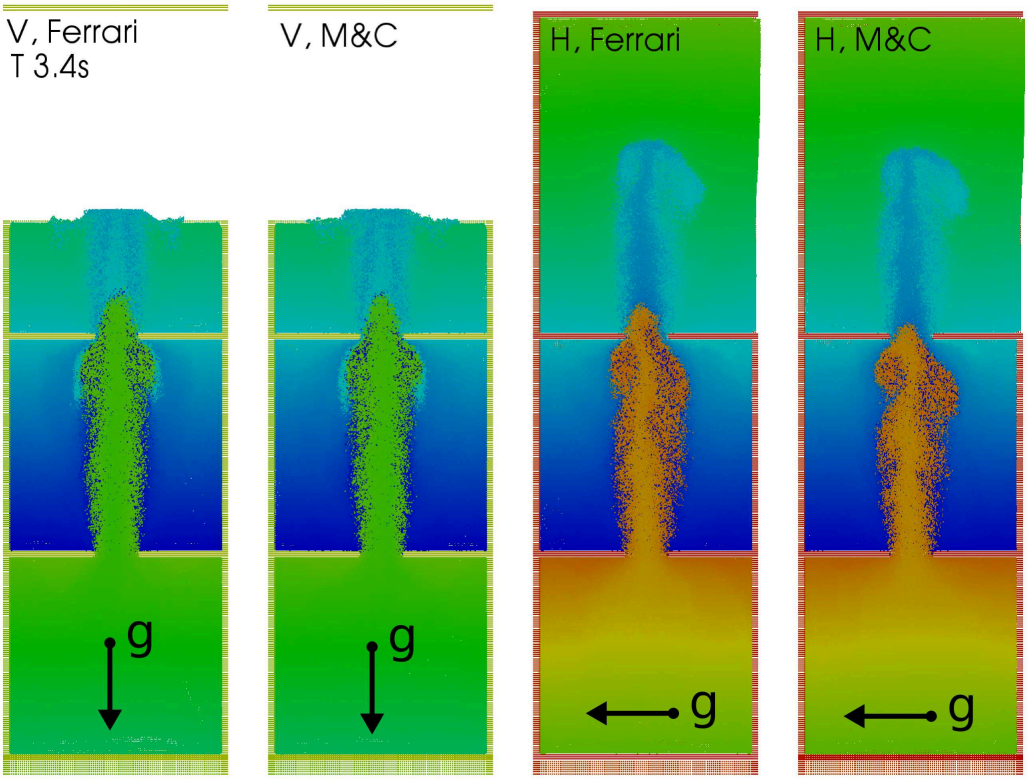


Figure 25: *Section of the XInjection test case with vertical and horizontal gravity, respectively.*

4.3.3 River

With the extension of the simulator to multi-node capabilities, it would be possible to simulate a river. With a resolution of 1 particle per cubic meter of water, a river with a width of 200 m could be simulated for about 10 km on a single GPU device. The river bed would have the same resolution and be loaded from a mesh or a DEM topography. The development of a test case modeling a river was only evaluated but not planned.

5 Publications and workshops

Here follows the list of publications and workshops conducted during the project:

5.1 Journal papers

- Z. Wei, R. A. Dalrymple, E. Rustico, A. Hérault and G. Bilotta, Simulation of Nearshore Tsunami Breaking by Smoothed Particle Hydrodynamics Method, Journal of Waterway, Port, Coastal and Ocean Engineering, July 2016, Vol. 142
- Z. Wei, R. A. Dalrymple, A. Hé, G. Bilotta, E. Rustico and H. Yeh SPH modeling of dynamic impact of tsunami bore on bridge piers, Coastal Engineering, Volume 104,

October 2015, Pages 26-42

- E. Rustico, G. Bilotta, A. Hérault, C. Del Negro and G. Gallo, Advances in multi-GPU Smoothed Particle Hydrodynamics simulations, IEEE Transactions on Parallel and Distributed Systems, vol. 99, 2013

5.2 Conference proceedings

- E. Rustico, T. Brudy-Zippelius, A. Hérault and G. Bilotta, Modelling the Holtenau ship lock with SPH, 11th International Conference on Hydroscience & Engineering (ICHE), September 28th - October 2nd, 2014, Hamburg (DE)
- E. Rustico, B. Sokoray-Varga, G. Bilotta, A. Hérault and T. Brudy-Zippelius, Full 3D numerical simulation and validation of a fih pass with GPUSPH, The 18th European Conference on Mathematics for Industry (ECMI), June 9-13, 2014, Taormina (IT)
- E. Rustico, J. Jankowski, A. Hérault, G. Bilotta and C. Del Negro, Multi-GPU, multi-node SPH implementation with arbitrary domain decomposition, 9th SPHERIC International Workshop, June 2-5, 2014, Paris (FR)
- R. Jalali Farahani, R. A. Dalrymple, A. Hérault, G. Bilotta and E. Rustico, Modeling the Coherent Vortices in Breaking Waves, 9th SPHERIC International Workshop, June 2-5, 2014, Paris (FR)

5.3 Workshops

- GPUSPH International Workshop - Conservatoire des Arts et Métiers (CNAM), Paris, France - October 2016
- GPUSPH International Workshop - Coastal Studies Institute (CSI), University of North Carolina, Manteo, NC, USA - April 2016
- GPUSPH International Workshop - Istituto Nazionale di Geofisica e Vulcanologia (INGV), Catania, Italy - October 2015
- GPUSPH International Workshop - Coastal Studies Institute (CSI), University of North Carolina, Manteo, NC, USA - May 2015
- GPUSPH International Workshop - Conservatoire des Arts et Métiers (CNAM), Paris, France - December 2014
- SPH workshop - Bundesanstalt für Wasserbau, Karlsruhe, in collaboration with the University of Bochum - June 2014
- GPUSPH International Workshop - Coastal Studies Institute (CSI), University of North Carolina, Manteo, NC, USA - May 2013

6 Conclusions

The SPH model is nowadays mature enough for practical applications and a number of academic as well as commercial software solutions are already available and currently being used. Only a few kinds of problems have particular requirements that push the method to the limits of its range of applicability. One includes those applications where a variation of density of 1 % in a fluid at rest, which is intrinsic of any compressible-SPH implementation, is already considered excessive. Another example is the requirement of a very large amount of particles, due to very big or very dense simulation scenarios, which confines the feasibility of a simulation to the small number of implementations capable of simulating a number of particles higher than hundreds of millions or featuring advanced techniques to reduce the number of required particles.

The first part of the project consisted in the extension of the GPUSPH simulator for the execution on a cluster of GPU nodes. The extension was successful and allowed for the simulation of the fish pass test case with 50 million particles, allowing for a comparison with the physical model at BAW. The friction artifacts with the boundaries resulted to be crucial in the flow speed and induced to delay further experiments until the development of a boundary model with less influence on the water levels.

However, even an implementation capable of simulating an arbitrarily big number of particles would be limited by the hardware capabilities (i.e. memory), which increased exponentially in the past twenty years but are of course far from being arbitrarily big. A reformulation of the problem is therefore necessary in some applications, for example by means of coupling with different models (e.g. in (Liu, 2016) a two-dimensional wave impact model interfacing a three-dimensional SPH simulation only in proximity with the impact to structures). Another technique, rather than the simple extension of the software and/or hardware platform, could be the use of variable resolution within the same SPH simulation; this is however an advanced feature offered in very few implementations on an experimental basis only. Its support is at an early stage of evaluation and design for GPUSPH.

The ship lock test case is an example of problems for which the number of particles has been one of the factors driving the development choices. The higher resolution required to correctly simulate the water flow in the culverts and the lack of capabilities for variable resolution led to an enormous computational cost in terms of both performance and memory. This excluded the use of the Semi-Analytical boundaries, which featured physically-correct pressure inlets and would have been likely to generate less friction along the ship hull. It is suspected that the insufficient resolution in the area where the inflow is generated is also one of the causes of the lift of the stream jet.

The presence of several different boundary models in the relevant literature could be considered as an advantage of the method as well as a problem since the lack of consensus requires choosing between slightly different traits and levels of adherence to physics. The appearance and validation of the Semi-Analytical boundaries seemed to have solved the dilemma; their development, however, is still at the level of a functional code having not yet reached the performance and stability of an industrial-level solution. With a computational cost of about 8× slower performance, unbearable for time span for the present project, the

boundary model has only been used for smaller evaluation tests.

Although the current development status and the first results are very promising for the applicability of the method to complex real-life engineering problems, and are already being used in the scientific community for many simpler problem scenarios, an optimization of the new boundary model is crucial to free the high potential of the method and accurately determine its application range and characteristics. An additional time of at least one year would be required to re-implement both test cases with the new model and produce a definitive comparison, granted the availability of the consortium members involving the optimization of the boundary model.

References

Amada, T.; Imura, M.; Yasumuro, Y.; Manabe, Y. and Chihara, K. Particle-based fluid simulation on GPU, 2004.

Amdahl, G. M. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483–485, New York, NY, USA, 1967. ACM. doi: 10.1145/1465482.1465560.

ATI. *ATI Stream Computing — User Guide*, December 2008.

Bilotta, G.; Hérault, A.; Del Negro, C.; Russo, G. and Vicari, A. Complex fluid flow modeling with SPH on GPU. *EGU General Assembly 2010, held 2-7 May, 2010 in Vienna, Austria, p.12233*, 12:12233, May 2010.

Cignoni, P.; Corsini, M. and Ranzuglia, G. MeshLab: an Open-Source 3D Mesh Processing System. *ERCIM News*, April(73):45–46, April 2008. URL `http://vcg.isti.cnr.it/Publications/2008/CCR08`.

Cleary, P. W. and Monaghan, J. J. Conduction Modelling Using Smoothed Particle Hydrodynamics. *Journal of Computational Physics*, 148(1):227 – 264, 1999. ISSN 0021-9991. doi: 10.1006/jcph.1998.6118.

Crespo, A. J. C.; Gómez-Gesteira, M. and Dalrymple, R. A. Boundary conditions generated by dynamic particles in SPH methods. *Computers, Materials, & Continua*, 5(3):173–184, 2007.

Crespo, A.; Domínguez, J.; Gómez-Gesteira, M.; Barreiro, A. and Rogers, B. *User Guide for DualSPHysics Code v2.0*, 2012. URL `http://dual.sphysics.org/index.php/downloads/`.

Dalrymple, R. and Hérault, A. Levee breaching with GPU-SPHysics code. In *Proc. Fourth Workshop, SPHERIC, ERCOFTAC, Nantes*, 2009.

Dalrymple, R.; Hérault, A.; Bilotta, G. and Farahani, R. J. GPU-accelerated SPH model for water waves and other free surface flows. In *Proc. 31st International Conf. Coastal Engineering, Shanghai*, 2010.

Ferrand, M.; Laurence, D. R.; Rogers, B. D.; Violeau, D. and Kassiotis, C. 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, 2013. ISSN 1097-0363. doi: 10.1002/fld.3666.

Gingold, R. A. and Monaghan, J. J. Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Mon. Not. Roy. Astron. Soc.*, 181:375–389, November 1977.

Gomez-Gesteira, M.; Rogers, B. D.; Dalrymple, R. A. and Crespo, A. J. State-of-the-art of classical SPH for free-surface flows. *Journal of Hydraulic Research*, 48(sup1):6–27, 2010. doi: 10.1080/00221686.2010.9641242.

Goswami, P.; Schlegel, P.; Solenthaler, B. and Pajarola, R. Interactive SPH simulation and rendering on the GPU. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on ComputerAnimation*, SCA '10, pages 55–64, 2010. URL `http://dl.acm.org/citation.cfm?id=1921427.1921437`.

Green, S. Particle simulation using CUDA, 2010. URL `http://developer.download.nvidia.com/compute/DevZone/C/html/C/src/particles/doc/particles.pdf`.

Group, T. K. OpenCL - the open standard for parallel programming of heterogeneous systems, November 2011. URL `http://www.khronos.org/opencl/`.

Gómez-Gesteira, M.; Rogers, B.; Dalrymple, R.; Crespo, A. and Narayanaswamy, M. User guide for the SPHysics code v1.2, 2007.

Harada, T.; Koshizuka, S. and Kawaguchi, Y. Smoothed particle hydrodynamics on GPUs. In *Computer Graphics International*, pages 63–70, 2007.

Hérault, A.; Vicari, A.; Del Negro, C. and Dalrymple, R. Modeling water waves in the surf zone with GPU-SPHysics. In *Proc. Fourth Workshop, SPHERIC, ERCOFTAC, Nantes*, 2009.

Hérault, A.; Bilotta, G. and Dalrymple, R. A. SPH on GPU with CUDA. *Journal of Hydraulic Research*, 48(Extra Issue):74–79, 2010a. ISSN 0022-1686.

Hérault, A.; Bilotta, G.; Dalrymple, R.; Rustico, E. and Del Negro, C. GPU-SPH, 2016. URL `http://www.gpusph.org`.

Hérault, A.; Bilotta, G.; Del Negro, C.; Russo, G. and Vicari, A. *SPH modeling of lava flows with GPU implementation*, volume 15 of *World Scientific Series on Nonlinear Science, Series B*, pages 183–188. World Scientific Publishing Company, 2010b.

Hérault, A.; Bilotta, G.; Vicari, A.; Rustico, E. and Del Negro, C. Numerical simulation of lava flow using a GPU SPH model. *Annals of Geophysics*, 54(5):600–620, 2011. doi: 10.4401/ag-5343.

Hoover, W. G. *Smooth Particle Applied Mechanics: The State of the Art*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2006. ISBN 9789812700025, 9812700021.

Keiser, R.; Adams, B.; Gasser, D.; Bazzi, P.; Dutre, P. and Gross, M. A unified lagrangian approach to solid-fluid animation. *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics*, 0:125–148, 2005. ISSN 1511-7813. doi: 10.1109/PBG.2005.194073.

Kolb, A. and Cuntz, N. Dynamic particle coupling for GPU-based fluid simulation. In *In Proc. of the 18th Symposium on Simulation Technique*, pages 722–727, 2005. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.89.2285`.

Liu, M. Coupling SPH with a potential Eulerian model for wave propagation problems. In *Proceedings of the 11th SPHERIC International Workshop*. Technische Universitaet Muenchen, Germany, Munich, Germany, June, 13-16 2016 2016.

Marivela, R. Applications of the SPH model to the design of fishways. In *33rd IAHR Congress*, Vancouver, Canada, 08/2009 2009.

Mayrhofer, A. Crixus, 2016. URL `https://github.com/Azrael3000/Crixus`.

Mayrhofer, A.; Ferrand, M.; Kassiotis, C.; Violeau, D. and Morel, F.-X. Unified semi-analytical wall boundary conditions in sph: analytical extension to 3-d. *Numerical Algorithms*, 68(1):15–34, 2015. ISSN 1572-9265. doi: 10.1007/s11075-014-9835-y.

Mazhar, H.; Heyn, T.; Pazouki, A.; Melanz, D.; Seidl, A.; Bartholomew, A.; Tasora, A. and Negrut, D. Chrono: a parallel multi-physics library for rigid-body, flexible-body, and fluid dynamics. *Mechanical Sciences*, 4(1):49–64, 2013. doi: 10.5194/ms-4-49-2013.

Monaghan, J. J. Smoothed particle hydrodynamics. In *Annual Review of Astronomy and Astrophysics 30*, pages 543–574, 1977.

Monaghan, J. J. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703, 2005.

Müller, M.; Solenthaler, B.; Keiser, R. and Gross, M. Particle-based fluid-fluid interaction. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '05, pages 237–244, New York, NY, USA, 2005. ACM. ISBN 1-59593-198-8. doi: 10.1145/1073368.1073402.

Nickolls, J.; Buck, I.; Garland, M. and Skadron, K. Scalable parallel programming with CUDA. *Queue*, 6:40–53, March 2008. ISSN 1542-7730. doi: 10.1145/1365490.1365500.

Rustico, E.; Bilotta, G.; Hérault, A.; Del Negro, C. and Gallo, G. Smoothed particle hydrodynamics simulations on multi-GPU systems. In *Proceedings of the International Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 543–574, Garching, DE, February 2012.

Rustico, E.; Jankowski, J.; Hérault, A.; Bilotta, G. and Del Negro, C. Multi-GPU, multi-node SPH implementation with arbitrary domain decomposition. In Damien Violeau, A. J., Alexis Hérault, editor, *Proceedings of the 9th SPHERIC International Workshop*, pages 127–133. CNAM, Paris, France, Paris, France, June, 03-05 2014 2014.

Rustico, E.; Bilotta, G.; Hérault, A.; Negro, C. D. and Gallo, G. Advances in Multi-GPU Smoothed Particle Hydrodynamics simulations. *IEEE Transactions on Parallel & Distributed Systems*, 25(1):43–52, 2013. ISSN 1045-9219. doi: doi.ieeecomputersociety.org/10.1109/TPDS.2012.340.

Smith, R. Open Dynamics Engine, 2014. URL `http://www.ode.org`.

Solenthaler, B. and Pajarola, R. Density contrast SPH interfaces. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '08, pages 211–218, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association. ISBN 978-3-905674-10-1. URL http://portal.acm.org/citation.cfm?id=1632592.1632623.

Solenthaler, B.; Schläfli, J. and Pajarola, R. A unified particle model for fluid–solid interactions: Research articles. *Comput. Animat. Virtual Worlds*, 18:69–82, February 2007. ISSN 1546-4261. doi: 10.1002/cav.v18:1.

Thorenz, C. and Anke, J. Evaluation of ship forces for a through-the-gate filling system. In *Proceedings of the PIANC - SMART Rivers Conference 2013*, Maastricht, NL, 2013. CNAM, Paris, France.

Violeau, D.; Issa, R.; Benhamadouche, S.; Saleh, K.; Chorda, J. and Maubourguet, M. M. Modelling a fish passage with SPH and Eulerian codes: the influence of turbulent closure. In *Proceedings of the 3rd SPHERIC International Workshop*, 2008.

Violeau, D. *Fluid mechanics and the SPH method: theory and applications*. Oxford University Press, Oxford, 2012. URL https://cds.cern.ch/record/1540142.