RESEARCH ARTICLE

Large-scale solitary wave simulation with implicit incompressible SPH

Ramprasad Sampath¹ · Niels Montanari¹ · Nadir Akinci¹ · Steven Prescott² · Curtis Smith²

Received: 7 December 2015 / Accepted: 6 June 2016 / Published online: 24 June 2016
© Springer International Publishing Switzerland 2016

Abstract Simultaneously tackling a large-scale simulation domain, high resolution, and a high level of incompressibility presents a computational challenge for smoothed-particle hydrodynamics. We present a method involving a combination of a recently developed implicit incompressible SPH scheme, a fluid-rigid boundary handling technique, and a data structure for neighborhood search to address the problem. We implemented the method on multi-core CPUs and performed simulations of tsunami wave impacts on coastal nuclear facilities to advance the state of the practice in probabilistic risk analysis modeling. Preliminary validation was conducted on a dam break problem test and a solitary wave past a conical island test. In addition to reduced memory consumption and the ability to handle arbitrarily large simulation domains, the technique exhibits fast simulation times. Performance of the method is evaluated for different parameter values related to time-stepping and neighborhood search, and several levels

of resolution and incompressibility enforcement. The results demonstrate its applicability for large-scale simulations relevant to ocean and coastal engineering, and, thereby, as a valuable tool for risk-informed decision-making related to external flooding events.

Keywords Smoothed-particle hydrodynamics · Incompressible SPH · Solitary wave · Performance analysis · Acceleration data structure

1 Introduction

Hydrodynamical models such as shallow water models (e.g., Boussinesq equations), or models based on the potential flow theory are widely used in ocean and coastal engineering for things such as representing ocean surface waves or generating inundation maps. They enable fast simulations to be realized, which is sometimes crucial such as for tsunami early warning or geophysical-scale simulations (Behrens and LeV-[eque 2011](#)). However, they also rely on strong assumptions which makes them questionable in some contexts. Nearshore detailed simulations often require 3D models based on the solution of the full Navier–Stokes equations that incorporate more comprehensive physics, deal with fluid-structure interactions, and complex flows and topologies.

Besides its traditional applications, fluid simulation is increasingly used for risk-based modeling of external hazards, such as tsunami-induced flooding, to advance the capabilities of safety analysis. The Risk-Informed Safety Margins Characterization (RISMC) toolkit, developed at Idaho National Laboratory, includes accounting for external flooding events (Mandelli et al. 2015; Prescott et al. 2015), which motivates the presented work.

Idaho National Laboratory, U.S. Department of Energy. This project was developed as part of an integrated RISMC toolkit.

✉ Ramprasad Sampath
ram@centroidlab.com

Niels Montanari
niels@centroidlab.com

Nadir Akinci
nadir@centroidlab.com

Steven Prescott
steven.prescott@inl.gov

Curtis Smith
curtis.smith@inl.gov

¹ Centroid LAB Inc, 12777 W Jefferson Blvd, Bldg D, Los Angeles, CA 90066, USA

² Idaho National Laboratory, 2525 Fremont Ave, Idaho Falls, ID 83415, USA

While mesh-based and Eulerian methods for simulating free-surface flows are well-matured and widely used, they are challenged by flows with high dynamics, over complex geometries and topologies, and around moving or deformable bodies. This type of scenario would typically occur when dealing with a tsunami wave hitting a coastline. Employing a mesh-free, Lagrangian method where the fluid is discretized as a set of particles without any explicit connection, is more powerful in these kinds of situations. Indeed, on the one hand, the Lagrangian approach enables avoidance of performing an otherwise numerically error-prone advection, especially in the case of violent flows. On the other hand, handling of complex boundaries and interaction with rigid and deformable bodies and of breaking waves and splashes, are not limited by a mesh resolution or the necessity of a remeshing operation.

Smoothed-particle hydrodynamics (SPH) is a common mesh-free Lagrangian method. It was originally introduced for dealing with astrophysical, compressible fluid flows (Gingold and Monaghan 1977; Lucy 1977), and later, its range of application domain extended to a wider variety of flows, such as non-Newtonian fluids (Xenakis et al. 2015), granular flows (Monatti and Paris 2015), and even solid deformation and fracture (Das and Cleary 2010). SPH has been applied to incompressible, free-surface flows for more than two decades (Monaghan 1994), along with the closely related moving particle semi-implicit (MPS) method (Koshizuka and Oka 1996). Mainly, two approaches have been used in SPH for enforcing incompressibility: (1) state equation-based SPH (SESPH), in which a compressible approach is employed with an equation of state, relating density, pressure and speed of sound in the fluid, and (2) in incompressible SPH (ISPH), in which pressure forces are implicitly computed and iteratively refined by solving a pressure Poisson equation until the desired compressibility is reached.

Simultaneously tackling large-scale simulation domain, high resolution, and a high level of incompressibility enforcement is a computational challenge for SPH. The issue has been partially circumvented through the use of massively parallel computing, relying on distributed and/or GPU-accelerated computing (Dominguez et al. 2013; Cercos-Pita et al. 2013). However, most of these highly parallel SPH solvers are based on SESP, which tends to yield noisy pressure fields and leads to less reliable pressure predictions (Lee et al. 2008). Conversely, ISPH features more accurate pressure computations; however, despite a few recent successes (Rogers et al. 2013; Asai et al. 2013), its ability to be used for large-scale simulations is still questioned.

Different projection schemes have been used in ISPH. The incompressibility condition can be interpreted either as a constant density (Shao and Lo 2003) or as a null divergence velocity (Cummins and Rudman 1999), leading to two variants for the source term in the pressure Poisson equation.

Solving two Poisson equations that account for both incompressibility criteria is also possible (Hu and Adams 2007; Gui et al. 2014), and enhances the accuracy of pressure predictions at the cost of extra computations. Higher-order source terms (Khayyer and Gotoh 2009), higher-order Laplacian operators (Gotoh et al. 2014; Ikari et al. 2015), and error-compensating source terms (Gotoh et al. 2014) have been proposed for enhancing accuracy and stability. Skew-adjointness of gradient and divergence operators (Leroy 2014) and the use of a gradient correction technique (Khayyer and Gotoh 2011) have been reported as important for energy conservation. An analytical formula for determining the critical time step for numerical stability has been recently derived for ISPH (Violeau and Leroy 2015). For more details, the latest progress in ISPH and MPS are thoroughly described in the survey paper (Gotoh and Khayyer 2016). A novel ISPH solver called implicit incompressible SPH (IISPH) has been introduced recently (Ihmsen et al. 2014). In spite of its origin in the computer graphics community, it does not assume any unacceptable approximation, thereby it seems promising when physical accuracy is desired. Its good efficiency, scalability, and ability to handle large time steps have been reported. Furthermore, IISPH has been implemented on distributed architecture (Thaler et al. 2014) and GPU (Goswami et al. 2015).

The fluid-rigid boundaries require special attention. First, discontinuities of physical quantities that occur at boundaries are problematic for the usual forms of SPH. Proper boundary handling is necessary to avoid underestimated densities and nonphysical pressure forces. Furthermore, pressure and friction forces between the fluid and the rigid bodies must be accounted for, and non-penetration must be ensured. Handling of thin objects or with complex geometries is particularly challenging, as well as when multiple dynamic objects are involved. Different strategies have been proposed through use of distance-based penalty forces (Monaghan and Kajtar 2009), frozen particles (Crespo et al. 2007), mirror particles (Liu et al. 2013) or a wall renormalisation factor (Ferrand et al. 2013). In this work, an efficient technique based on frozen particles and able to cope with complex boundaries, multiple bodies and the discontinuity issue is employed (Akinci et al. 2012). Moreover, the technique uses fluid-rigid pressure and friction force models conserving linear and angular momentum.

SPH requires the computation of sums over dynamically changing sets of neighboring particles. Spatial data structures are employed to accelerate the neighborhood search, and efficient construction and query are thus essential. While hierarchical data structures are used in simulations with a variable particle interaction radius (Adams et al. 2007), uniform grids are more efficient when the radius is fixed. The cell size is usually set to the interaction radius. Index sort-based grids (Green 2008) are often preferred, because the

basic grid suffers from low cache hit rate and race conditions during parallel construction.

A further increase of the cache hit rate is realized through use of a space-filling curve, such as the Z-curve or the Hilbert curve (Goswami et al. 2010). While very efficient, the memory consumption then scales with the simulation domain. To handle large-scale simulation domains without compromising performance, a combination of compact spatial hashing and Z-curve index sorting has been proposed (Ihmsen et al. 2011) and is used in this work. It features performance similar to index sort-based grids using the Z-curve, optimized cache-hit rate, reduced memory consumption, and the ability to handle an arbitrarily large simulation domain.

In this paper, we present simulations produced by IISPH (Ihmsen et al. 2014), combined with the fluid-rigid boundary handling technique (Akinci et al. 2012) and the neighborhood search based on compact hashing and Z-curve index sorting (Ihmsen et al. 2011). This method is implemented on multi-core CPUs, and its applicability for large-scale simulations relevant to ocean and coastal engineering is demonstrated. The impact of a tsunami wave on a coastal nuclear power plant serves as the test problem. The tsunami wave is produced using a piston-type wave maker based on Goring's methodology for generating solitary waves (Goring 1978). The performance of the method is evaluated for several parameters related to time-stepping and neighborhood search, and for different levels of resolution and incompressibility enforcement.

Basics of SPH are first introduced in Sect. 2. The IISPH discretization of the Navier–Stokes equations is then presented in Sect. 3. Explanations of the IISPH pressure solver, fluid-rigid boundary handling, and neighborhood search are located in Sects. 4, 5, and 6, respectively. For more details about these specific components, the reader is invited to refer to the original papers (Ihmsen et al. 2014; Akinci et al. 2012; Ihmsen et al. 2011). The solitary wave generation based on Goring's work is presented in Sect. 7. Prior validation performed on a dam break problem test and a solitary wave past a conical obstacle test are presented in Sect. 8. Finally, the simulation results of the tsunami wave impacts are available in Sect. 9.

2 Smoothed-particle hydrodynamics

SPH is a numerical method for both interpolating quantities and approximating spatial differential operators from a set of known quantities at sampled positions. Among the different ways to interpolate from scattered data, SPH relies on the use of radial basis functions. The value of a field at any point of space is estimated based on the value of this field at neighbor sampling points and its distance to them.

Let A be a scalar field defined on a domain Ω . The value of A at position $\mathbf{x}_i$ can be exactly estimated through the convolution with the Dirac delta distribution δ :

$$A(\mathbf{x}_i) = \int_{\Omega} A(\mathbf{x}) \delta(\mathbf{x}_i - \mathbf{x}) d\mathbf{x} \quad (1)$$

The Dirac delta distribution cannot be defined numerically, and thus has to be approximated by the so-called kernel function W . Furthermore, the integration is also approximated, replaced by a discrete sum over the sampling points, indexed j :

$$A(\mathbf{x}_i) = \sum_j V_j A(\mathbf{x}_j) W(\mathbf{x}_i - \mathbf{x}_j) \quad (2)$$

where V_j is the volume represented by the sampling point j . Let us consider the sampling points as a set of fluid particles carrying physical quantities, e.g., a mass, a velocity, a pressure, etc. In the following equation, we assume the volume of a particle j to be expressed as $V_j = \frac{m_j}{\rho_j}$, where m_j and ρ_j are its mass and density. Considering constant particle mass, fluid rest density ρ_0 , initial volume δr^3 in 3D, with δr the initial inter-particle distance, and an initial particle configuration on a Cartesian grid, the mass is calculated as $m_j = \rho_0 \delta r^3$. Therefore, the SPH interpolation reads as:

$$A_i = \sum_j \frac{m_j}{\rho_j} A_j W_{ij} \quad (3)$$

where the simplified notations $A_i = A(\mathbf{x}_i)$ and $W_{ij} = W(\mathbf{x}_i - \mathbf{x}_j)$ are used.

The kernel function W must satisfy a certain number of properties. First, it must be defined on a compact support, as a kernel with infinite support would be computationally impractical. It must also be radial, feature a Gaussian-like shape, be sufficiently smooth, and tend to the Dirac delta distribution when its support radius tends to zero. Finally, choosing a kernel function that is even and normalized leads to a first-order consistent continuous SPH interpolation. Note that the particle distribution has an impact on the accuracy. A regular and isotropic distribution is desired. Besides, the trade-off between interpolation accuracy and computational cost should also motivate the choice of the kernel function. In this work, the cubic B-spline kernel is employed:

$$W_{ij} = \alpha_W \begin{cases} 6q^3 - 6q^2 + 1 & \text{if } 0 \leq q < \frac{1}{2} \\ 2(1 - q)^3 & \text{if } \frac{1}{2} \leq q < 1 \\ 0 & \text{if } q \geq 1 \end{cases} \quad (4)$$

where $\alpha_W = \frac{8}{\pi h^3}$ is the normalization constant in 3D and $q = \frac{|\mathbf{x}_i - \mathbf{x}_j|}{h}$, with h the support radius that is set as twice the

Fig. 1 Illustration of the SPH interpolation. W denotes a Gaussian-like shaped kernel function, and h is the support radius

rest inter-particle distance δr . Figure 1 illustrates the SPH interpolation.

There are several ways to build spatial differential operators in SPH. In this work, the following SPH gradient and SPH divergence are considered:

$$\nabla A_i = \rho_i \sum_j m_j \left(\frac{A_i}{\rho_i^2} + \frac{A_j}{\rho_j^2} \right) \nabla W_{ij} \quad (5)$$

$$\nabla \cdot A_i = -\frac{1}{\rho_i} \sum_j m_j (A_i - A_j) \cdot \nabla W_{ij} \quad (6)$$

The SPH divergence (6) is zero-order consistent. As for the SPH gradient (5), it is not even zero-order consistent. However, since it is involved in the pressure force computations, its pairwise symmetric nature is necessary for momentum conservation and optimizes the regularity and isotropy of particle distribution (Price 2012).

3 IISPH discretization of the Navier–Stokes equations

The single-phase, isothermal, incompressible, Newtonian fluid flows are the considered physical model. In addition, the viscosity is assumed constant in space, and the surface tension forces are ignored. The Navier–Stokes equations in their Lagrangian, velocity-pressure formulation then reads:

$$\frac{d\mathbf{v}}{dt} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \mathbf{v} + \mathbf{g} \quad (7)$$

$$\frac{d\rho}{dt} = -\rho \nabla \cdot \mathbf{v} \quad (8)$$

where $\mathbf{v}$, p , ν and $\mathbf{g}$ stand for velocity, pressure, kinematic viscosity, and gravitational acceleration, respectively.

Equation (7) comes from the momentum conservation, and involves pressure, viscosity, and gravity forces. Equation (8) is derived from the mass conservation. The incompressibility condition is satisfied either by $\frac{d\rho}{dt} = 0$ or $\nabla \cdot \mathbf{v} = 0$; these two ways are equivalent in the continuous representation.

Algorithm 1 IISPH solver

```

1: while simulating do
2:   construct neighborhood search data structure
3:   for all fluid particle  $i$  do
4:     query neighbor particles  $j$ 
5:   for all fluid particle  $i$  do
6:     compute density  $\rho_i(t)$  (9)
7:   update time step  $\Delta t$  (10)
8:   for all fluid particle  $i$  do
9:     compute intermediate velocity  $\mathbf{v}_i^*$  (13)
10:  for all fluid particle  $i$  do
11:    compute intermediate density  $\rho_i^*$  (15)
12:  solve pressure Poisson equation (16)
13:  for all fluid particle  $i$  do
14:    compute velocity  $\mathbf{v}_i(t + \Delta t)$  (17)
15:    compute position  $\mathbf{x}_i(t + \Delta t)$  (18)

```

As a preliminary step, since it is involved in all other SPH computations, the densities ρ_i are calculated using the SPH interpolation:

$$\rho_i(t) = \sum_j m_j W_{ij}(t) \quad (9)$$

where: t represents the time. Next, the time step Δt is updated according to two constraints:

$$\Delta t = \min(\Delta t_{CFL}, \Delta t_{\max}) \quad (10)$$

Δt_{CFL} represents the Courant–Friedrichs–Lewy (CFL) condition, and $\Delta t_{\max}$ is empirically designed to avoid unreasonable numbers of iterations of the pressure solver. These are expressed as:

$$\Delta t_{CFL} = \begin{cases} \frac{\lambda_{CFL} h}{|\mathbf{v}_{\max}|} & \text{if } |\mathbf{v}_{\max}| > \epsilon \\ \sqrt{\frac{\lambda_{CFL} h}{|g|}} & \text{otherwise} \end{cases} \quad (11)$$

$$\Delta t_{\max} = \frac{\lambda_{\max} h}{|g|} \quad (12)$$

where λ_{CFL} is the CFL coefficient, $\lambda_{\max}$ is a coefficient for $\Delta t_{\max}$, $\mathbf{v}_{\max}$ is the maximum velocity over all particles, and ϵ is the small positive number close to 0. IISPH makes use of the pressure projection method. Velocity and pressure are decoupled by splitting the resolution of the momentum equation (7) into two steps. First, intermediate velocities $\mathbf{v}_i^*$ are computed by solving the momentum equation with only the non-pressure force terms. The Monaghan artificial viscosity formulation (Monaghan and Gingold 1983) is used in this

work to discretize the viscosity term, and the time integration is realized with the explicit Euler scheme:

$$\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \left(- \sum_j m_j \Pi_{ij} \nabla W_{ij}(t) + \mathbf{g} \right) \quad (13)$$

where

$$\Pi_{ij} = -\nu \frac{\min((\mathbf{v}_i(t) - \mathbf{v}_j(t)) \cdot (\mathbf{x}_i(t) - \mathbf{x}_j(t)), 0)}{|\mathbf{x}_i(t) - \mathbf{x}_j(t)|^2 + 0.01h^2} \quad (14)$$

with the viscous factor $\nu = \frac{2\alpha h c_s}{\rho_i(t) + \rho_j(t)}$, α a viscosity constant and c_s standing for the speed of sound in the fluid. Intermediate densities ρ_i^* are then computed using the mass equation (8) and the implicit Euler scheme:

$$\rho_i^* = \rho_i(t) + \Delta t \sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \cdot \nabla W_{ij}(t) \quad (15)$$

In the second step, the momentum equation with only the pressure force term is combined with the mass equation and the incompressibility condition. Following a time discretization, it leads to the following Poisson equation for the pressures p_i :

$$\nabla^2 p_i(t) = \frac{\rho_0 - \rho_i^*}{\Delta t^2} \quad (16)$$

The resolution of this Poisson equation is detailed in Sect. 4. Finally, the new velocity and position are computed by taking into account the pressure force term, and the first-order, symplectic Euler–Cromer scheme is employed for the time integration:

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i^* + \Delta t \sum_j m_j \times \left(\frac{p_i(t)}{\rho_i(t)^2} + \frac{p_j(t)}{\rho_j(t)^2} \right) \nabla W_{ij}(t) \quad (17)$$

$$\mathbf{x}_i(t + \Delta t) = \mathbf{x}_i(t) + \Delta t \mathbf{v}_i(t + \Delta t) \quad (18)$$

Prior to performing the computations, a spatial data structure for accelerating the neighborhood search is constructed, and the neighbors are queried and stored afterwards. Details are available in Sect. 6. The steps of the IISPH solver are summarized in Algorithm 1.

4 IISPH pressure solver

The pressure Poisson equation (16) is reformulated as:

$$\rho_i(t) \nabla \cdot \left(\frac{1}{\rho_i(t)} \nabla p_i(t) \right) = \frac{\rho_0 - \rho_i^*}{\Delta t^2} \quad (19)$$

where the Laplacian is expressed as the divergence of the gradient and the approximation $\nabla \rho_i(t) = 0$ is done. The discretized Poisson equation in IISPH is obtained using the SPH divergence (6) and SPH gradient (5):

$$\begin{aligned} & - \sum_j m_j \left[\sum_j m_j \left(\frac{p_i(t)}{\rho_i(t)^2} + \frac{p_j(t)}{\rho_j(t)^2} \right) \nabla W_{ij}(t) \right. \\ & \left. - \sum_k m_k \left(\frac{p_j(t)}{\rho_j(t)^2} + \frac{p_k(t)}{\rho_k(t)^2} \right) \nabla W_{jk}(t) \right] \cdot \nabla W_{ij}(t) \\ & = \frac{\rho_0 - \rho_i^*}{\Delta t^2} \end{aligned} \quad (20)$$

It can be reformulated as a linear system with pressure values as unknowns and matrix coefficients a_{ij} , and each equation reads:

$$\sum_j a_{ij} p_j(t) = \rho_0 - \rho_i^* \quad (21)$$

The conjugate gradient method would be the method of choice to solve the system, given its high convergence rate. However, a few issues plague its use. First, it requires the matrix to be symmetric. This can be enforced by assuming $\rho_i = \rho_j = \rho_0$ and $m_i = m_j$ for all particles, but only for single-phase, uniform mass simulations. Additionally, the computational cost per iteration is high. Relaxed Jacobi is employed instead, which iterates for each individual pressure p_i as:

$$p_i^{l+1} = (1 - \omega) p_i^l + \omega \frac{\rho_0 - \rho_i^* - \sum_{j \neq i} a_{ij} p_j^l}{a_{ii}} \quad (22)$$

where: l denotes the iteration index and ω is the relaxation factor, chosen as 0.5. The discretized pressure Poisson equation (20) can be rewritten as:

$$\begin{aligned} & \sum_j m_j (\mathbf{d}_{ii} - \mathbf{d}_{ji}) \cdot \nabla W_{ij}(t) p_i(t) \\ & + \sum_j m_j \left(\sum_j \mathbf{d}_{ij} p_j(t) - \mathbf{d}_{jj} p_j(t) - \sum_{k \neq i} \mathbf{d}_{jk} p_k(t) \right) \\ & \cdot \nabla W_{ij}(t) \\ & = \rho_0 - \rho_i^* \end{aligned} \quad (23)$$

where

$$\mathbf{d}_{ii} = -\Delta t^2 \sum_j \frac{m_j}{\rho_i(t)^2} \nabla W_{ij}(t) \quad (24)$$

$$\mathbf{d}_{ij} = -\Delta t^2 \frac{m_j}{\rho_j(t)^2} \nabla W_{ij}(t) \quad (25)$$

Algorithm 2 IISPH pressure solver with relaxed Jacobi

```

1: for all fluid particle  $i$  do
2:   compute  $d_{ii}$  (24)
3:   compute  $a_{ii}$  (26)
4:   initialize  $p_i^0$  (28)
5:  $l = 0$ 
6: while  $\rho_{err}^l > \xi$  (29) or  $l < l_{min}$  do
7:   for all fluid particle  $i$  do
8:     compute  $\sum_j \mathbf{d}_{ij} p_j^l$  (25)
9:   for all fluid particle  $i$  do
10:    compute  $p_i^{l+1}$  (22)
11:     $p_i(t) = p_i^{l+1}$ 
12:     $l = l + 1$ 

```

The matrix coefficients a_{ii} can then be identified:

$$a_{ii} = \sum_j m_j (\mathbf{d}_{ii} - \mathbf{d}_{ji}) \cdot \nabla W_{ij}(t) \quad (26)$$

as well as:

$$\sum_{j \neq i} a_{ij} p_j^l = \sum_j m_j \left(\sum_j \mathbf{d}_{ij} p_j^l - \mathbf{d}_{jj} p_j^l - \sum_{k \neq i} \mathbf{d}_{jk} p_k^l \right) \cdot \nabla W_{ij}(t) \quad (27)$$

The relaxed Jacobi requires initializing the pressure field. In this work, it is done as:

$$p_i^0 = 0.6 p_i(t - \Delta t) \quad (28)$$

which, in combination with setting the relaxation factor as $\omega = 0.5$, has been experimentally found to lead to an optimal convergence. The relative density error ρ_{err}^l at iteration l of the average density ρ_{avg}^l from the rest density is used as a convergence criterion:

$$\rho_{err}^l = \frac{\rho_{avg}^l}{\rho_0} - 1 \quad (29)$$

Additionally, a minimum number of iterations l_{min} is defined as 3. The steps of the IISPH pressure solver are summarized in Algorithm 2. The threshold for the density error to be reached is denoted by ξ .

5 Fluid-rigid boundary handling

At the boundaries, the kernel support domain might not be sufficiently sampled, which leads to an underestimation of densities. The neighboring boundary particles are thus taken into account when computing densities and forces for fluid particles. The surface of the rigid objects is sampled using a single layer of boundary particles. The particle sampling is realized either directly for analytical shapes, or from a mesh

(Akinci et al. 2013). As possible, the spacing of the boundary particles is enforced equal to the rest inter-particle distance δr of the fluid particles. However, oversampling might occur in some regions. To cope with this, the relative contribution Ψ_b of a boundary particle b to a physical quantity is computed as:

$$\Psi_b = \frac{\rho_0}{\sum_k W_{bk}(t)} \quad (30)$$

where k denotes boundary particle neighbors. The boundary particles contribute to the densities of fluid particles i as:

$$\rho_i(t) = \sum_j m_j W_{ij}(t) + \sum_b \Psi_b W_{ib}(t) \quad (31)$$

The pressure force $\mathbf{F}_{i \leftarrow b}^p$ and friction force $\mathbf{F}_{i \leftarrow b}^v$ applied from a boundary particle b to a fluid particle i are expressed as:

$$\mathbf{F}_{i \leftarrow b}^p = -m_i \Psi_b \frac{p_i(t)}{\rho_i(t)^2} \nabla W_{ib}(t) \quad (32)$$

$$\mathbf{F}_{i \leftarrow b}^v = -m_i \Psi_b \Pi_{ib} \nabla W_{ib}(t) \quad (33)$$

where the viscous factor is reformulated as $\nu = \frac{\sigma h c_s}{2 \rho_i(t)}$, with σ the fluid-rigid viscosity coefficient.

Incorporating the fluid-rigid boundary handling requires a few modifications in the IISPH solver besides the SPH interpolation of the density (31). The fluid-rigid friction must be taken into account in the intermediate velocity estimate:

$$\mathbf{v}_i^* = \mathbf{v}_i(t) + \Delta t \left(- \sum_j m_j \Pi_{ij} \nabla W_{ij}(t) - \sum_b \Psi_b \Pi_{ib} \nabla W_{ib}(t) + \mathbf{g} \right) \quad (34)$$

As for the intermediate density, assuming a constant rigid velocity $\mathbf{v}_b$ throughout the pressure iterations leads to:

$$\rho_i^* = \rho_i(t) + \Delta t \left(\sum_j m_j (\mathbf{v}_i^* - \mathbf{v}_j^*) \nabla W_{ij}(t) + \sum_b \Psi_b (\mathbf{v}_i^* - \mathbf{v}_b) \nabla W_{ib}(t) \right) \quad (35)$$

In the pressure solver, additional terms appear, inducing the following modifications:

$$d_{ii} = -\Delta t^2 \left(\sum_j \frac{m_j}{\rho_i(t)^2} \nabla W_{ij}(t) + \sum_b \frac{\Psi_b}{\rho_i(t)^2} \nabla W_{ib}(t) \right) \quad (36)$$

$$\sum_{j \neq i} a_{ij} p_j^l = \sum_j m_j \left(\sum_j d_{ij} p_j^l - d_{jj} p_j^l - \sum_{k \neq i} d_{jk} p_k^l \right) \cdot \nabla W_{ij}(t) - \sum_b \Psi_b \sum_j d_{ij} p_j^l \nabla W_{ib}(t) \quad (37)$$

Finally, the fluid-rigid pressure must be taken into consideration when computing the new velocity:

$$\begin{aligned} \mathbf{v}_i(t + \Delta t) = & \mathbf{v}_i^* \\ & + \Delta t \left(\sum_j m_j \left(\frac{p_i(t)}{\rho_i(t)^2} + \frac{p_j(t)}{\rho_j(t)^2} \right) \nabla W_{ij}(t) \right. \\ & \left. + \sum_b \Psi_b \frac{p_i(t)}{\rho_i(t)^2} \nabla W_{ib}(t) \right) \end{aligned} \quad (38)$$

6 Neighborhood search

To accelerate the neighborhood search, a combination of compact hashing and Z-curve index sorting is employed. Besides the particle array, a spatial data structure consisting in a hash table of dimension m and a compact array of dimension n is employed. m is a prime number; n is lesser than the number of cells filled with particles, itself lesser than N . The cell size is set to the support radius h . For each particle i , a cell index c_i is calculated:

$$\begin{aligned} c_i = & c_{xi} + c_{yi}K + c_{zi}L \\ (c_{xi}, c_{yi}, c_{zi}) = & \left(\frac{x_i - x_{min}}{h}, \frac{y_i - y_{min}}{h}, \frac{z_i - z_{min}}{h} \right) \end{aligned} \quad (39)$$

where (x_i, y_i, z_i) is the position of the particle, $(x_{min}, y_{min}, z_{min})$ the minimum coordinates over all particles, and K and L the number of cells in the x and y directions of the fluid axis-aligned bounding box. Arbitrarily large simulation domains are handled by mapping each used cell, of index coordinates (c_x, c_y, c_z) , to the hash table using the following hashing function:

$$H(c_x, c_y, c_z) = [(c_x p_1) \oplus (c_y p_2) \oplus (c_z p_3)] \bmod m \quad (40)$$

where p_1 , p_2 and p_3 are large prime numbers taken as 73,856,093, 19,349,663 and 83,492,791. Each cell of the compact array stores its index in the hash table and a list of handles, consisting in a reference to a particle and its corresponding cell index. The memory consumption is $O(m + n + N)$, independently of the whole simulation domain size.

Compared to Ihmsen et al. (2011), the particle array is reordered according to a space-filling Z-curve at every time step. We found that doing so provides good results

for all scenes, especially large ones. Unnecessary copies are avoided by sorting first the handles according to the Z-curve. All the other attributes are reordered accordingly afterwards. The insertion of the particle references is performed serially. Indeed, it is required for ensuring consistency between used cell and Z-curve order. Further reduction of memory consumption could be realized by storing only a reference to the first particle in the sorted array and the number of particles located within the corresponding cell, instead of references to all particles in the cell. This would lead to an $O(m + 2n)$ memory complexity.

7 Solitary wave generation

Solitary waves represent a good model for tsunami waves. Among the various ways of generating a solitary wave, a piston-type wave maker based on Goring's (1978) methodology is used. Thanks to the accurate fluid-rigid boundary handling, such a wave maker is applicable. Goring's methodology for solitary wave generation starts by assuming that the average vertical velocity of the adjacent water equals the wave paddle velocity. That is, the wave paddle displacement X is linked to the free-surface displacement η as follows:

$$\frac{dX}{dt} = \frac{C\eta(t)}{h + \eta(t)} \quad (41)$$

where C is the wave celerity and h is the water depth. The free-surface displacement and the wave celerity are determined using the solitary wave solution of Boussinesq:

$$\eta(t) = H \operatorname{sech}^2(\theta) \quad (42)$$

$$\theta = \kappa(Ct - X(t))$$

with the wave celerity and the decay coefficient κ expressed as:

$$\kappa = \sqrt{\frac{3H}{4h^3}} \quad (43)$$

$$C = \sqrt{|g|(h + H)} \quad (44)$$

where H denotes the wave height. Combining Eqs. (41) and (42) leads to:

$$X(t) = \frac{H}{Kh} \tanh(\kappa(Ct - X(t))) \quad (45)$$

which is solved by applying Newton's method, for which an iteration indexed by l reads:

$$\theta^{l+1} = \theta^l - \frac{\theta^l - \kappa Ct + \frac{H}{h} \tan(\theta^l)}{1 + \frac{H}{h} \operatorname{sech}^2(\theta^l)} \quad (46)$$

Finally, the wave paddle displacement is deduced as:

$$X(t) = Ct - \frac{\theta}{\kappa} \quad (47)$$

The wave paddle total displacement S and the corresponding duration T can be calculated as:

$$S = \frac{2H}{\kappa h} \quad (48)$$

$$T = \frac{2}{\kappa C} \left(3.80 + \frac{H}{h} \right) \quad (49)$$

8 Preliminary validation tests

Two validation tests were performed to assess the ability of the solver to accurately predict force impact on structures and wave propagation. The first one is a dam break problem test; the second one is devoted to a breaking solitary wave past a conical obstacle. For both tests, the simulation results are confronted to experimental data.

8.1 Dam break

We consider the dam break problem as carried out by Yeh and Petroff (Raad and Bidoae 2005), which is a popular benchmarking test case (Cummins et al. 2012; Pan et al. 2016). The simulation is in a one-to-one scale with the experiment. The scene consists in a rectangular tank with 0.61 m width, 1.6 m length, and 0.75 m height; a vertical column with a 0.12 m squared base; and a planar gate for suddenly releasing water. The column is located 0.9 m from one of the extremity of the tank, and the gate is located in-between at a distance of 0.5 m from the column. A volume of water with 0.3 m height is initially contained behind the gate, while a 0.01 m layer of residual water is present elsewhere in the tank. The configuration is schematized in Fig. 2. For the simulation, we set the rest inter-particle distance δr to 0.01 m, leading to about 64 k fluid particles; the time-stepping parameters λ_{CFL} and λ_{max} to 0.4 and 0.5, respectively; and the tolerated density error ξ to 0.1 %.

The gate is removed at the beginning of the simulation, and the water starts to flow under the influence of the gravity. The first, strong impact occurs at 0.3 s. The wave reaches the other side of the tank at 0.6 s. Then, the reflected wave propagates and hits the column at approximately 1.5 s. We measured the horizontal component of the impact force on the column and compared it to the experimental data. Plots are displayed in Fig. 3. The simulation data are in good agreement with those of the experiment until a simulated time of 1.3 s, although the initial wave impact at 0.3 s is slightly overestimated. The second impact corresponding to the reflected wave is less well predicted with the impact profile too sharp in the simulation.

Fig. 2 Dam break problem configuration (from Cummins et al. 2012)

Fig. 3 Evolution of the horizontal force applied on the column. The blue line represents the simulation measurements; the dark line is for the experimental data. High-frequency noise of the simulation data was smoothed by averaging over the ten simulation steps before and after the current one (color figure online)

However, for the real experiment, a significant presence of bubbles could be expected at that time, which we do not take into account in our physical model. This could explain the observed discrepancy.

8.2 Solitary wave past a conical island

We consider here the case of a solitary wave propagating up a triangular shelf with an island feature located at the offshore point of the shelf. This test has been used for benchmarking tsunami models (Horillo et al. 2015). In the physical experiment, free-surface elevation (FSE) was recorded via resistance-type wave gages and sonic wave gages, while velocity information was recorded via acoustic Doppler

velocimetry (ADV) probes. A piston-type wave maker was used to generate a wave of a height of 0.39 m measured at a location where the water depth reached 0.78 m.

To simulate the physical experiment, bathymetry data were converted to 3D geometry. The piston-type wave maker described in Sect. 7 was used to generate the wave. We measured the FSE at several gage points and horizontal components of the velocity at a specific ADV point. The simulation was performed in a one-to-one scale with the experiment and for a total duration of 10 s. The resolution of the simulation was defined by a rest inter-particle distance of 0.0475 m, leading to about 2320 k fluid particles. The other numerical parameters are set to the same values as for the dam break problem test.

The motion of the wave maker is initiated at the beginning of the simulation. The wave height is first measured at its reference height of 0.39 m to match the specifications of the experiment. At 4.9 s, the wave reaches the conical island, and the wave front splits in two parts. At 7.5 s, the two fronts collide on the back side of the island, exhibiting turbulent behavior. Figure 4 displays the run of the simulation from two different viewpoints.

Results of the simulation with its comparison to the physical experiment (Lynett 2015) are indicated in Figs. 5, 6, and 7. We noticed reasonably good agreement with the experimental data for the FSE and the velocity component in the direction of flow (U Component), despite a slight delay for some of the measurements. Regarding the other horizontal component, orthogonal to the direction of flow (V Component), we observed a certain discrepancy. Lack of turbulence modeling perhaps could be attributed to this.

9 Results and discussion

We did a performance analysis, investigating the impact of several parameters: time-stepping coefficient $\lambda_{\max}$, hash table size m , tolerated density error ξ , and rest inter-particle distance δr . For this purpose, we performed four sets of simulations, totaling 15 simulation runs. The scene consists of a model representing the facilities of a Fukushima Dai-ichi look-alike coastal nuclear power plant, and a section of the ocean nearshore. The whole simulation domain has a size of $1300 \times 1150 \times 260 \text{ m}^3$. For generating the tsunami wave, a wave paddle is modeled by means of a large rigid plane positioned at 800 m away from shore. The motion of the wave paddle is such that it delivers a wave with a height measured as approximately 25 m at 700 m away from the shore. The CFL coefficient λ_{CFL} is set to 0.4; we do not explore its influence in this paper, as it would require a stability analysis. The fluid-rigid friction model, expressed by Eq. (33), could reasonably be ignored in this scene, but is kept for performance analy-

sis. The experiments are performed for a simulated time of 90 s each. We run the simulations on an Intel Xeon E5-2680, v2, 2.8 GHz dual 10-core CPU, with 40 threads being used. A simulation run is illustrated in Fig. 8. The wave is in the process of being generated at time $t = 0$ s. Then, it starts propagating and reaches a location 700 m away from the shore at $t \simeq 15$ s. The impact occurs at $t \simeq 55$ s, from which we can deduce a speed of propagation of 17.5 m.s^{-1} . At $t = 60$ s, a reflected wave can be observed. Finally, at $t \simeq 90$ s, when the simulation stops, the front of the reflected wave is about 300 m away from the shore, corresponding to a speed of propagation of 10 m.s^{-1} . Zoom-in snapshots showing the velocity field and flow patterns near the structure can be seen in Fig. 9.

In the first set, we compare four simulations with different time-stepping coefficients $\lambda_{\max}$: 0.050, 0.075, 0.100, and 0.200. We set the rest inter-particle distance δr , tolerated density error ξ , and hash table size m as shown by Table 1. *No.* is used to identify the simulation run, and N_f and N_b denote the approximate numbers of fluid particles and rigid particles, respectively. We measured the average time step Δt_{avg} , average number of iterations l_{avg} of the pressure solver, and total simulation duration T . Additionally, we timed the computations spent on constructing the data structure for neighborhood search, querying the neighboring particles, performing the advection and solving the pressure Poisson equation; the corresponding duration are denoted by T_c , T_q , T_a , and T_p , respectively. The advection is to be understood here as including all the computations done in Eqs. (9), (13), (15), (17), and (18). The measurements are displayed in Table 2. It is worth noting that run No. 4 corresponds to a situation for which the time step is almost always constrained by the CFL condition (11). The results show that a larger time step does not necessarily yield to a better performance when employing a pressure projection scheme. Indeed, advecting with a larger time step also increases the deviation from the incompressibility condition; thus restoring it requires more endeavors, i.e., iterations, from the pressure solver. On the other hand, the overall cost of searching neighbors and advecting increases with the number of simulation steps. A trade-off must be found, which seems optimally realized using $\lambda_{\max} = 0.075$, corresponding to about 15 iterations of the pressure solver on average and about 45 % of the total simulation time spent on enforcing incompressibility, as shown in Table 3.

In the second set, we use another set of four simulations to investigate the influence of the hash table size m . We set its value to 180,001, 360,007, 4,360,001, and 8,360,003, corresponding to prime numbers close to N_f , $2N_f$, $2N_f + N_b$, and $2N_f + 2N_b$, respectively. The choice of using $2N_f$ has

Fig. 4 Top (*left*) and side (*right*) views of the evolution of the wave propagating and passing the conical obstacle. The images correspond to the wave front at the location of the measurement fields, which are reached at simulated times of 3.2 s (*first row*), 4.9 s (*second row*), 7.5 s (*third row*), and 9.1 s (*fourth row*)

Fig. 5 Visual comparison with a video of a similar physical experiment, available at <http://coastal.usc.edu/plynnett/research/exp/index.html> (accessed May 10, 2016). Courtesy of Lynett (2015)

Fig. 6 Evolution of the horizontal components of the velocity at a position located at a distance of 13 m from the wave maker along the *centerline*. The simulation data are represented by the *solid line*; the experimental data are represented by the *dashed line*

been found sufficient to ensure a low hash collision rate (Ihm-sen et al. 2011). The parameters used for these simulations can be seen in Table 4. The construction of the data structure for neighborhood search and the neighbor query are the only operations for which the duration notably varies, the other timings we measured being almost identical to those of the run No. 2. As shown in Table 5, using a large enough size for the hash table is important for ensuring a high efficiency. The choice of $2N_f$ leads to a gain close to 3 min compared to that of N_f , that is, a 17.3 % decrease of the time spent on neighborhood search. Furthermore, increasing m by N_b brings an additional important speedup of 2 min 16 s, corresponding to a further 13.2 % reduction; it indicates that the hash collision rate was still quite high. However, adding another N_b to m gives a more moderate gain, of only 27 s, which is just an extra 2.7 %. The results thus suggest employing at least $m = 2N_f + N_b$

when memory consumption is not much of an issue, and that the worthiness of going beyond this value is more debatable.

The third set deals with the tolerated density error ξ . Five runs are compared, with respective values 0.025, 0.050, 0.100, 0.200, and 0.400 %. Table 6 shows the parameters used for this set of simulations. The influence of ξ is restricted to the pressure solver that is, to its number of iterations and the time spent on enforcing incompressibility, for which the results are displayed in Table 7. Their evolution with ξ can be visualized in Fig. 10. Unsurprisingly, both vary in the same manner, the computational cost per iteration being basically constant at a given simulation step. The results exhibit an almost perfect linear scaling of l_{avg} and T_p , which increase by a factor of ~ 1.78 when halving ξ . It is also to be noted that the time steps are approximately the same for all the runs of this set, as the time step constraint expressed by

Fig. 7 Evolution of the free-surface elevation at the different measurement locations. The simulation data are represented by the *solid line*; the experimental data are represented by the *dashed line*

Fig. 8 A 25-m-high tsunami wave hitting a coastal nuclear power plant and inundating facility structures. The resolution corresponds to a 1-m rest inter-particle distance. The fluid particles are *color-coded* according to their velocities. The images correspond to a simulated time of 0 s (*top left*), 20 s (*top right*), 60 s (*bottom left*) and 90 s (*bottom right*) (color figure online)

Fig. 9 Zoom-in snapshots: of the tsunami wave propagating (*top left*) and impacting the nuclear facilities (*top right*), with visualization of the velocity magnitude field; and of the formation of a vortex near a cavity-shaped structure (*bottom*), with visualization of velocity vector field and streamlines. The simulation run with 2-m rest inter-particle distance was used

Table 1 Setup parameters for the varying $\lambda_{\max}$ simulation set

No.	δr (m)	N_f	N_b	ξ (%)	m	$\lambda_{\max}$
1	4	180 k	4000 k	0.100	4,360,001	0.050
2						0.075
3						0.100
4						0.200

Table 2 Performance measurements (1) for the varying $\lambda_{\max}$ simulation set

No.	Δt_{avg} (s)	l_{avg}	T (s)	T_c (s)	T_q (s)	T_a (s)	T_p (s)
1	0.041	5.5	2080	112	1145	364	459
2	0.060	14.7	1711	73	641	239	758
3	0.077	27.9	1776	58	411	190	1117
4	0.095	44.3	2070	50	354	166	1500

Table 3 Performance measurements (2) for the varying $\lambda_{\max}$ simulation set

No.	T_c (%)	T_q (%)	T_a (%)	T_p (%)
1	5.4	55.0	17.5	22.1
2	4.3	37.5	14.0	44.3
3	3.3	23.1	10.7	62.9
4	2.4	17.1	8.0	72.5

Table 4 Setup parameters for the varying m simulation set

No.	δr (m)	N_f	N_b	ξ (%)	m	$\lambda_{\max}$
5	4	180 k	4000 k	0.100	180,001	0.075
6					360,007	
2					4,360,001	
7					8,360,003	

Table 5 Performance measurements for the varying m simulation set

No.	T (s)	T_c (s)	T_q (s)	$T_c + T_q$ (s)
5	2025	67	961	1028
6	1847	69	781	850
2	1711	73	641	714
7	1684	76	611	687

Equation (12) is independent from ξ . Better total simulation times could be expected with an improved formulation that takes the level of incompressibility enforcement into account.

In the last set, we consider different levels of resolution by varying the rest inter-particle distance δr , from 1 m to 5 m. The parameters used for this set are displayed in Table 8. We adapted the hash table size to the number of fluid parti-

Table 6 Setup parameters for the varying ξ simulation set

No.	δr (m)	N_f	N_b	ξ (%)	m	$\lambda_{\max}$
8	4	180 k	4000 k	0.025	4,360,001	0.075
9				0.050		
2				0.100		
10				0.200		
11				0.400		

Table 7 Performance measurements for the varying ξ simulation set

No.	l_{avg}	T (s)	T_p (s)
8	44.7	3317	2364
9	26.4	2278	1325
2	14.7	1711	758
10	8.7	1424	471
11	5.0	1228	275

cles and rigid particles and chose the value $m = 2N_f + N_b$, as suggested by our findings of the second set. The results are shown in Tables 9 and 10. The simulations with moderate resolution could be run in a reasonable amount of time, e.g., about 4 h 35 min for the 2 m resolution, and less than 29 min for the 4 m resolution. On the other hand, the 1-m resolution simulation was completed within 65 h, and simulating at a finer scale can be considered out of reach in practice. It can be seen that incompressibility enforcement and neighbor search might both be bottlenecks. Contrary to the neighborhood search, the weight of the pressure solving rises with the resolution of the simulation, while that of the advection is more or less constant. The time step criteria (12) is resolution-dependent and able to enforce approximately the same number of iterations of the pressure solver per time step irrespective of the resolution, i.e., ~ 15 on average in this scene. However, because the pressure solver has a higher weight on the simulation duration when a higher resolution is used, it is not clear if $\lambda_{\max} = 0.075$, determined from 4 m resolution simulations, is the optimal choice for all resolutions. Instead of enforcing a given number of iterations, it might be more efficient to make use of time-stepping aiming at a given weight of the pressure solving, e.g., 45 % of the total simulation time, as suggested by the results of the first set. This would require further investigation. Lastly, the evolution of the total simulation duration with N_f is drawn in Fig. 11. A good scaling is observed at moderate resolution if we consider less than about 500 k fluid particles, doubling N_f results in an increase of the simulation duration by a factor lesser than 2. On the other hand, the results suggest a less favorable trend for significantly higher resolutions, with a factor beyond 2. Not much conclusion can be drawn with regards to the scalability of the method

Fig. 10 Variation of the average number of iterations of the pressure solver (*left*) and of the time spent on enforcing incompressibility (*right*) with the tolerated density error. The *bottom charts* correspond to a $\log_2$ -rescaling and a recentering on the data of run No. 2

Table 8 Setup parameters for the varying δr simulation set

No.	δr (m)	N_f (k)	N_b (k)	ξ (%)	m	$\lambda_{\max}$
12	1	18,020	18,190	0.100	54,230,023	0.075
13	2	2040	7370		11,450,003	
14	3	660	4890		6,210,011	
2	4	180	4000		4,360,001	
15	5	100	3330		3,530,011	

Table 9 Performance measurements (1) for the varying δr simulation set

No.	Δt_{avg} (s)	I_{avg}	T (s)	T_c (s)	T_q (s)	T_a (s)	T_p (s)
12	0.015	14.7	233,317	23,578	34,580	25,479	149,680
13	0.030	14.2	16,506	1369	3103	2175	9859
14	0.046	13.3	4858	350	1628	588	2292
2	0.060	14.7	1711	73	641	239	758
15	0.075	17	1198	34	527	139	498

with increasing resolutions in the general case though, as N_b does not follow the same evolution as N_f in this specific scene.

Table 10 Performance measurements (2) for the varying δr simulation set

No.	T_c (%)	T_q (%)	T_a (%)	T_p (%)
12	10.1	14.8	10.9	64.2
13	8.3	18.8	13.2	59.7
14	7.2	33.5	12.1	47.2
2	4.3	37.5	14.0	44.3
15	2.8	44.0	11.6	41.6

10 Conclusion

We presented the application of an SPH method for large-scale simulations in a coastal engineering context. The Navier–Stokes equations are discretized using the IISPH scheme, and fluid-rigid boundary conditions are properly enforced. Reduced memory consumption is accomplished, and arbitrarily large simulation domains are handled thanks to an efficient neighborhood search based on compact spatial hashing and Z-curve index sorting. We simulated tsunami waves impacting a coastal nuclear power plant and inundating facility structures, employing Goring’s methodology for

Fig. 11 Variation of the total simulation duration with the number of fluid particles. The *right chart* correspond to a $\log_2$ -rescaling and a recentering on the data of run No. 14. For the sake of readability, the data of run No. 12 is not represented in the non-rescaled chart

generating solitary waves. We implemented the method on multi-core CPUs. We evaluated its performance for different values of parameters linked to time-stepping and neighborhood search, and also for several levels of resolution and incompressibility enforcement. Preliminary validation tests were conducted on two cases of dam break and solitary wave past a conical island, exhibiting a good agreement with the experimental data.

Although this work evaluates capability and performance, these simulations provide methods for subsequent analysis. The ability to measure particle contact and pressure forces on structures over time provides a rich set of data for risk analysis methods.

While the applicability of the method for large-scale simulations is demonstrated, simulating with over thousand million fluid particles is in practice still out of reach. Several ways to speedup the computations could be investigated. A superior adaptive time-stepping scheme could be devised, attempting to use a time step closer to the optimal one. Coupling the method with a shallow water simulation code is another possibility. The shallow water code would handle the propagation of the tsunami wave until near the coastline, where dealing with the flow complexity and the fluid-structure interactions requires solving the Navier–Stokes equations. Lastly, higher levels of parallelization could be realized by taking advantage of distributed (Thaler et al. 2014) and/or general-purpose GPU computing (Goswami et al. 2015).

The work to date has focused on detailing the different components of the method and performance analysis. Rigorous and comprehensive benchmarking must constitute future work for in-depth verification and validation. Also, more accurate wave impact forces could be expected by including a velocity divergence part to the source term of the pressure Poisson equation (Gui et al. 2014).

Acknowledgments We would like to thank our funding sources, the Idaho National Laboratory (U.S. Dept. of Energy) under U.S. GOVERNMENT CONTRACT NO. DE-AC07-05ID14517. We would also like to thank Dr. Nam Dinh, North Carolina State University, and the reviewers for their valuable suggestions, and Linyu Lin, North Carolina State University, for his help with the dam break validation experiment.

References

- Adams B, Pauly M, Keiser R, Guibas L (2007) Adaptively sampled particle fluids. *ACM Trans Graph (SIGGRAPH Proc)* 26(3):48–54
- Akinci N, Ihmsen M, Akinci G, Solenthaler B, Teschner M (2012) Versatile rigid-fluid coupling for incompressible SPH. *ACM Trans Graphics (SIGGRAPH Proc)* 31(4):62:1–62:8
- Akinci N, Cornelis J, Akinci G, Teschner M (2013) Coupling elastic solids with smoothed particle hydrodynamics fluids. *J Comp Animat Virtual Worlds* 24(3–4):195–203
- Asai M, Isshiki M, Tanabe S (2013) Large scale tsunami simulation by the incompressible SPH with real geography generated from aero-survey data. In: *Proc. International Conf. on Particle-Based Method, Stuttgart, Germany*
- Behrens J, LeVeque R (2011) Modeling and simulating tsunamis with an eye to hazard mitigation. *SIAM News* 44(4)
- Cercos-Pita JL, Souto-Iglesias A, Gonzalez L, Macia F (2013) AQUAgpusph, a free 3D SPH solver accelerated with OpenCL. In: *Proc. 8th International SPHERIC Workshop, Trondheim, Norway*, vol 192, p 295–312
- Crespo AJC, Gómez-Gesteira M, Dalrymple RA (2007) Boundary conditions generated by dynamic particles in SPH methods. *Comp Mater Continua* 5(3):173–184
- Cummins S, Rudman M (1999) An SPH projection method. *J Comput Physics* 152(2):584–607
- Cummins SJ, Silvester TB, Cleary PW (2012) Three-dimensional wave impact on a rigid structure using smoothed particle hydrodynamics. *Int J Numer Methods Fluids* 68(12):1471–1496. doi:10.1002/fld.2539
- Das R, Cleary P (2010) Effect of rock shapes on brittle fracture using smoothed particle hydrodynamics. *Theor Appl Fract Mech* 53(1):47–60
- Dominguez J, Crespo A, Valdez-Balderas D, Rogers B, Gomez-Gesteira M (2013) New multi-GPU implementation for smoothed particle

- hydrodynamics on heterogeneous clusters. *Comp Phys Commun* 184(8):1848–1860
- Ferrand M, Laurence DR, Rogers BD, Violeau D, Kassiotis C (2013) Unified semi-analytical wall boundary conditions for inviscid, laminar or turbulent flows in the meshless SPH method. *Int J Numer Methods Fluids* 71:446–472. doi:10.1002/fld.3666
- Gingold R, Monaghan J (1977) Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices Royal Astron Soc* 181:375–398
- Goring DG (1978) Tsunamis - the propagation of long waves onto a shelf. Report no. kh-r-38, California Institute of Technology, Keck Laboratory of Hydraulics and Water Resources, California Institute of Technology, Pasadena, California
- Goswami P, Schlegel P, Solenthaler B, Pajarola R (2010) Interactive SPH Simulation and Rendering on the GPU. SCA '10: Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on Computer animation. Norrköping, Sweden, pp 1–10
- Goswami P, Eliasson A, Franzen P (2015) Implicit incompressible SPH on the GPU. In: Proc. of 12th Workshop on Virtual Reality Interaction and Physical Simulation, Lyon, France, Nov 4–5, 2015, Eurographics Association, pp 23–29
- Gotoh H, Khayyer A (2016) Current achievements and future perspectives for projection-based particle methods with applications in ocean engineering. *Journal of Ocean Engineering and Marine Energy* pp 1–28
- Gotoh H, Khayyer A, Ikari H, Arikawa T, Shimosako K (2014) On enhancement of incompressible SPH method for simulation of violent sloshing flows. *Appl Ocean Res* 46:104–115
- Green S (2008) Particle-based fluid simulation. In: Game Developers Conference, pp 1–33
- Gui Q, Shao S, Dong P (2014) Wave impact simulations by an improved ISPH model. *Journal of Waterway Port, Coastal, and Ocean Engineering* 140(3)
- Horrillo J, Grili ST, Nicolsky D, Roeber V, Zhang J (2015) Performance benchmarking tsunami models for NTHMP's inundation mapping activities. *Pure Appl Geophy* 172(3):869–884
- Hu X, Adams N (2007) An incompressible multi-phase SPH method. *J Comp Phys* 227(1):264–278
- Ihmsen M, Akinci N, Becker M, Teschner M (2011) A parallel SPH implementation on multi-core CPUs. *Comp Graph Forum* 30(1):99–112
- Ihmsen M, Cornelis J, Solenthaler B, Horvath C, Teschner M (2014) Implicit incompressible SPH. *IEEE Trans Visual Comp Graph* 20(3):426–435
- Ikari H, Khayyer A, Gotoh H (2015) Corrected higher order Laplacian for enhancement of pressure calculation by projection-based particle methods with applications in ocean engineering. *J Ocean Eng Marine Energy* 1(4):361–376
- Khayyer A, Gotoh H (2009) Wave impact pressure calculations by improved sph methods. *Int J Offshore Polar Eng* 19(4):300–307
- Khayyer A, Gotoh H (2011) Enhancement of stability and accuracy of the moving particle semi-implicit method. *J Comput Phys* 230(8):3093–3118
- Koshizuka S, Oka Y (1996) Moving-particle semi-implicit method for fragmentation of incompressible fluid. *Nuclear Sci Eng* 123(3):421–434
- Lee ES, Moulinec C, Xu R, Violeau D, Laurence D, Stansby P (2008) Comparisons of weakly compressible and truly incompressible algorithms for the SPH mesh free particle method. *J Comput Phys* 227(10):8417–8436
- Leroy A (2014) A new incompressible SPH model: towards industrial applications. PhD thesis, Université Paris-Est
- Liu X, Xu H, Shao S, Lin P (2013) An improved incompressible SPH model for simulation of wave-structure interaction. *Comp Fluids* 71:113–123
- Lucy L (1977) A numerical approach to the testing of the fission hypothesis. *Astron J* 82:1013–1024
- Lynett P (2015) Solitary wave past a conical island. http://coastal.usc.edu/currents_workshop/all_data.zip
- Mandelli D, Prescott S, Smith C, Alfonsi A, Rabiti C, Cogliato J, Kinoshita R (2015) A flooding induced station blackout analysis for a pressurized water reactor using the RISMIC toolkit. *Sci Technol Nucl Install* 2015
- Monaghan J, Gingold R (1983) Shock simulation by the particle method SPH. *J Comput Physics* 52(2):374–389
- Monaghan J, Kajtar J (2009) SPH particle boundary forces for arbitrary boundaries. *Comp Phys Commun* 180(10):1811–1820
- Monaghan JJ (1994) Simulating free surface flows with SPH. *J Comput Phys* 110:399–406. doi:10.1006/jcph.1994.1034
- Monatti L, Paris E (2015) A SPH model for the simulation of free surface granular flows in a dense regime. *Appl Math Model* 39(1):363–382
- Pan K, IJzermans RHA, Jones BD, Thyagarajan A, van Beest BWH, Williams JR (2016) Application of the SPH method to solitary wave impact on an offshore platform. *Comput Particle Mecha* 3(2):155–166
- Prescott S, Smith C, Sampath R (2015) Incorporating dynamic 3D simulation into PRA. In: International Topical Meeting on Probabilistic Safety Assessment and Analysis, PSA 2015, vol 2, pp 1053–1062
- Price D (2012) Smoothed particle hydrodynamics and magnetohydrodynamics. *J Comp Phys Spl Issue Comput Plasma Phys* 231(3):759–794
- Raad PE, Bidoae R (2005) The three-dimensional Eulerian–Lagrangian marker and micro cell method for the simulation of free surface flows. *J Comput Phys* 203(2):668–699
- Rogers B, Ashworth M, Stansby P, Lind S, Guo X (2013) Efficient massive parallelisation for incompressible smoothed particle hydrodynamics with 10e8 particles. In: Proc. 8th International SPHERIC Workshop, Trondheim, Norway, pp 397–402, <http://www.sintef.no/projectweb/spheric-2013/>
- Shao S, Lo EYM (2003) Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Adv Water Resour* 26:787–800
- Thaler F, Solenthaler B, Gross M (2014) A parallel architecture for IISPH fluids. In: Proc. of 11th Workshop on Virtual Reality Interaction and Physical Simulation, pp 119–124
- Violeau D, Leroy A (2015) Optimal time step for incompressible SPH. *J Comput Phys* 288:119–130
- Xenakis A, Lind S, Stansby P, Rogers B (2015) An incompressible SPH scheme with improved pressure predictions for free-surface generalised Newtonian flows. *J Non-Newtonian Fluid Mech* 218:1–15