International Journal of Computational Methods
Vol. 16, No. 4 (2019) 1843012 (20 pages)

DOI: 10.1142/S0219876218430120

Numerical Simulation of Three-Dimensional Violent Free Surface Flows by GPU-Based MPS Method

Xiang Chen and Decheng Wan*

State Key Laboratory of Ocean Engineering
School of Naval Architecture, Ocean and Civil Engineering
Shanghai Jiao Tong University
Collaborative Innovation Center for Advanced Ship
and Deep-Sea Exploration Shanghai 200240, P. R. China
**dcwan@sjtu.edu.cn*

Received 13 December 2017
Accepted 12 September 2018
Published 16 October 2018

The Moving Particle Semi-implicit (MPS) method has been widely used in the field of computational fluid dynamics in recent years. However, the inefficient drawback of MPS method limits its three-dimensional (3D) large-scale applications. In order to overcome this disadvantage, a novel acceleration technique, graphics processing unit (GPU) parallel computing, is applied in MPS. Based on modified MPS method and GPU technique, an in-house solver MPSGPU-SJTU has been developed by using Compute Unified Device Architecture (CUDA) language. In this paper, 3D dam break and sloshing, two typical violent flows with large deformation and nonlinear fragmentation of free surface are simulated by MPSGPU-SJTU solver. In dam break case, the results of fluid flied, water front, wave height and impact pressure by GPU simulation are compared to those by CPU calculation, experimental research, Smooth Particle Hydrodynamics (SPH) and Boundary Element Method (BEM) simulations. And the comparison of fluid field and impact pressure among GPU, CPU and experiment is made in sloshing flow. The accuracy of GPU solver is verified by these comparisons. Moreover, the computation time of every part in each calculation step is compared between GPU and CPU solvers. The results show that computational efficiency is improved dramatically by employing GPU acceleration technique.

Keywords: Modified MPS method; GPU acceleration technique; MPSGPU-SJTU solver; 3D violent free surface flows.

1. Introduction

Moving particle semi-implicit (MPS) method is one Lagrangian meshless method for incompressible fluid, which was introduced by Koshizuka and Oka [1996]. Similar to other meshless methods like SPH, many randomly distributed particles are

*Corresponding author.

used to represent the fluid domain in MPS. These particles contain the information of mass, momentum, pressure and so on. The pattern of solving Navier–Stokes equation is semi-implicit, which is a distinctive feature of MPS. The stable pressure field of fluid can be obtained from solving pressure Poisson equation. Because of the particles representation, MPS can easily track free surfaces and moving boundaries and remove many numerical difficulties due to the nonlinear surface. In the recent years, more and more researchers have used MPS to simulate the violent flow problems with large deformation or nonlinear fragmentation of free surface, such as dam break [Zhang *et al.* (2011)], sloshing [Yang *et al.* (2015)], water entry [Chen *et al.* (2017)], fluid-structure interaction [Zhang *et al.* (2016)] and so on.

In order to obtain more accurate and stable results, many researchers have devoted themselves to improving the calculation accuracy and suppressing the pressure oscillation. Many numerical models of MPS are modified such as kernel function [Koshizuka *et al.* (1998); Ataie-Ashtiani and Farhadi (2006)], gradient model [Koshizuka *et al.* (1998); Khayyer and Gotoh (2008); Tsuruta *et al.* (2013)], Laplacian model [Khayyer and Gotoh (2012); Ikari *et al.* (2015)], pressure Poisson equation [Khayyer and Gotoh (2009); Tanaka and Masunaga (2010)], Kondo and Koshizuka (2011) and free surface detection [Khayyer and Gotoh (2009); Tanaka and Masunaga (2010)]. But in the past years, MPS is usually applied to simulate the two-dimensional (2D) problems because of the low computational efficiency. The refined particles methods like multi-resolution [Tang *et al.* 2016a, 2016b] and overlapping [Shibata *et al.* (2012); Tang *et al.* (2016)] are typical numerical acceleration technologies to reduce the computational cost. In addition, many researchers use CPU parallel technique to accelerate the calculation of MPS [Ikari and Gotoh (2008); Iribe *et al.* (2010)].

By using CPU parallel technique, it is found that the computation time of MPS is reduced with the increase of calculation cores. The graphics processing unit (GPU) whose remarkable feature is multi-cores have been produced with the development of industry. Based on GPU acceleration technique, many meshless methods are applied to simulate large-scale problems. The application of GPU technique in SPH is more mature than MPS. Harada *et al.* [2007] developed one search method for neighboring particles in order to implement the SPH entirely on GPU. By the limit of GPU card capacity, the maximum particle number is four million and the maximum speedup is 28. Crespo *et al.* [2011] developed DualSPHysics solver based on GPU acceleration technique. They used this solver to simulate three-dimensional (3D) dam break problem with one million particles and achieved a speedup of 64 by comparing to one CPU core. Then, Domínguez *et al.* [2013] optimized DualSPHysics solver based on the characters of GPU and accelerated the SPH codes with a maximum speedup of 56.2. Mokos *et al.* [2015] developed two-phase GPU code to simulate 3D dam break with obstacle and obtain high acceleration ratio on different GPU card. Because pressure Poisson equation is solved implicitly, the acceleration effect of GPU for MPS is not remarkable and the research in this field is rare. Zhang *et al.* [2011] developed different versions of MPS code based on

different GPU memories. Hori *et al.* [2011] used CUDA language to develop a GPU-accelerated MPS code and only acquired about 3–7 acceleration ratio by simulating 2D dam break. Li *et al.* [2015] applied GPU acceleration technique to two parts of MPS, neighbor particle list and pressure Poisson equation. By simulating 3D dam break and sloshing, the speedup of these two parts is about 1.5 and 10, respectively. Gou *et al.* [2016] used GPU accelerated MPS to simulate the isothermal multi-phase fuel-coolant interaction.

In this work, the GPU acceleration technique is introduced in modified MPS to simulate 3D violent free surface flows. The brief introduction of modified MPS and GPU implementation in this paper is presented. Then, the GPU solver is used to simulate 3D dam break and sloshing problems. The calculated results of GPU code such as fluid field, impact pressure, wave height and water front are compared with the results of CPU solver, experiment and other numerical methods. In addition, the comparison of computation time between GPU solver and CPU solver is conducted.

2. Numerical Method

In this paper, the violent free surface flows are calculated by our in-house particle solver MPSGPU-SJTU based on modified MPS method. The applied numerical models are introduced briefly in this section.

2.1. *Governing equations*

The governing equations for viscous incompressible fluid contain continuity equation and Navier–Stokes equation.

$$\frac{1}{\rho}\frac{D\rho}{Dt} = \nabla \cdot \mathbf{V} = 0, \tag{1}$$

$$\frac{D\mathbf{V}}{Dt} = -\frac{1}{\rho}\nabla P + \nu\nabla^2\mathbf{V} + \mathbf{g}, \tag{2}$$

where ρ is the fluid density, t is the time, $\mathbf{V}$ is the velocity vector, P is the pressure, ν is the kinematic viscosity and $\mathbf{g}$ is the gravitational acceleration vector.

2.2. *Kernel function*

In MPS method, the particle interaction is described by a kernel function. Zhang and Wan [2012] developed a modified kernel function in order to avoid the singularity at $r = 0$ in original version.

$$W(r) = \begin{cases} \dfrac{r_e}{0.85r + 0.15r_e} - 1 & 0 \le r < r_e \\ 0 & r_e \le r \end{cases}, \tag{3}$$

where r is the distance between two particles and r_e is the radius of the particle interaction. The particle number density and gradient model is $r_e = 2.1l_0$, while $r_e = 4.0l_0$ is used for the Laplacian model, where l_0 is the initial distance between two adjacent particles.

2.3. *Particle interaction models*

The models of particle interaction include gradient model, divergence model and Laplacian model in MPS. These models can be written as

$$\langle \nabla \phi \rangle_i = \frac{D}{n^0} \sum_{j \neq i} \frac{\phi_j + \phi_i}{|\mathbf{r}_j - \mathbf{r}_i|^2} (\mathbf{r}_j - \mathbf{r}_i) \cdot W(|\mathbf{r}_j - \mathbf{r}_i|), \tag{4}$$

$$\langle \nabla \mathbf{V} \rangle_i = \frac{D}{n^0} \sum_{j \neq i} \frac{(\mathbf{V}_j - \mathbf{V}_i) \cdot (\mathbf{r}_j - \mathbf{r}_i)}{|\mathbf{r}_j - \mathbf{r}_i|^2} \cdot W(|\mathbf{r}_j - \mathbf{r}_i|), \tag{5}$$

$$\langle \nabla^2 \phi \rangle_i = \frac{2D}{n^0 \lambda} \sum_{j \neq i} (\phi_j - \phi_i) \cdot W(|\mathbf{r}_j - \mathbf{r}_i|), \tag{6}$$

$$\lambda = \frac{\sum_{j \neq i} W(|\mathbf{r}_j - \mathbf{r}_i|) \cdot |\mathbf{r}_j - \mathbf{r}_i|^2}{\sum_{j \neq i} W(|\mathbf{r}_j - \mathbf{r}_i|)}, \tag{7}$$

where D is the space dimension, n^0 is the initial particle number density, $\mathbf{r}$ is coordinate vector of particle, ϕ is any physical quantity and λ is applied to make sure that the increase of variance is equal to the analytical solution.

2.4. *Model of incompressibility*

Lee *et al.* [2011] improved a mixed source term method [Tanaka and Masunaga (2010)] combined with the velocity divergence-free condition and constant particle number density condition.

$$\langle \nabla^2 P^{k+1} \rangle_i = (1 - \gamma) \frac{\rho}{\Delta t} \nabla \cdot \mathbf{V}_i^* - \gamma \frac{\rho}{\Delta t^2} \frac{\langle n^* \rangle_i - n^0}{n^0}, \tag{8}$$

where γ is a variable parameter from 0 to 1, Δt is the time step, $n*$ is the temporal particle number density and defined as

$$\langle n \rangle_i = \sum_{j \neq i} W(|\mathbf{r}_j - \mathbf{r}_i|). \tag{9}$$

2.5. *Free surface detection*

In MPS method, the Dirichlet boundary condition is imposed by assigning zero pressure for surface particles. Zhang and Wan [2012] developed a modified surface particle detection method, which is based on the asymmetry arrangement of neighboring particles.

$$\langle \mathbf{F} \rangle_i = \frac{D}{n^0} \sum_{j \neq i} \frac{1}{|\mathbf{r}_i - \mathbf{r}_j|} (\mathbf{r}_i - \mathbf{r}_j) W(r_{ij}), \tag{10}$$

where $\mathbf{F}$ is a vector which represents the asymmetry of arrangements of neighbor particles. If one particle is on the free surface, the value of vector will be large.

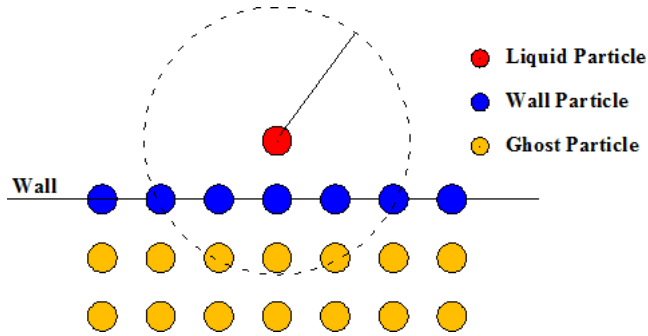


Fig. 1. Schematic of boundary particles.

Thus, particle satisfying:

$$\langle |\mathbf{F}| \rangle_i > 0.9|\mathbf{F}|^0, \tag{11}$$

is considered as surface particle, where $|\mathbf{F}|^0$ is the initial value of $|\mathbf{F}|$.

2.6. *Boundary condition*

In this work, multilayer particles are used to present the wall boundary as shown in Fig. 1. The wall particles are arranged at the boundary and their pressures are solved by PPE. Two layers of ghost particles are configured to fulfill the particle number density near the boundary so that the particle interaction can be properly simulated near the boundary. The pressure of ghost particle is obtained by interpolation.

3. GPU Acceleration

Based on the above brief introduction, one feature of MPS method can be found that the calculation of each particle is independent of the synchronous results of other particles except solving pressure Poisson equation. This feature determines that the calculation flow of MPS can be effectively parallelized. Comparing with CPU, GPU is designed to possess more arithmetic logic units (ALU) in the same chip area. This hardware design makes GPU to own high floating point operations per second (FLOPS) and the ability to process multi-objects simultaneously.

CUDA [2017] is a parallel computing platform and programming model created by NVIDIA and implemented by GPU. A CUDA program is divided into a host part and a device part. The host part runs on CPU while the device part runs on GPU. The host code includes instructions for setting parallelism and communicating data between host and device. The differences between CPU and GPU codes are described in Appendix A.

In order to accelerate the iteration of pressure Poisson equation, the open source library CUSP [2017] is applied in GPU solver. CUSP is a library for sparse linear

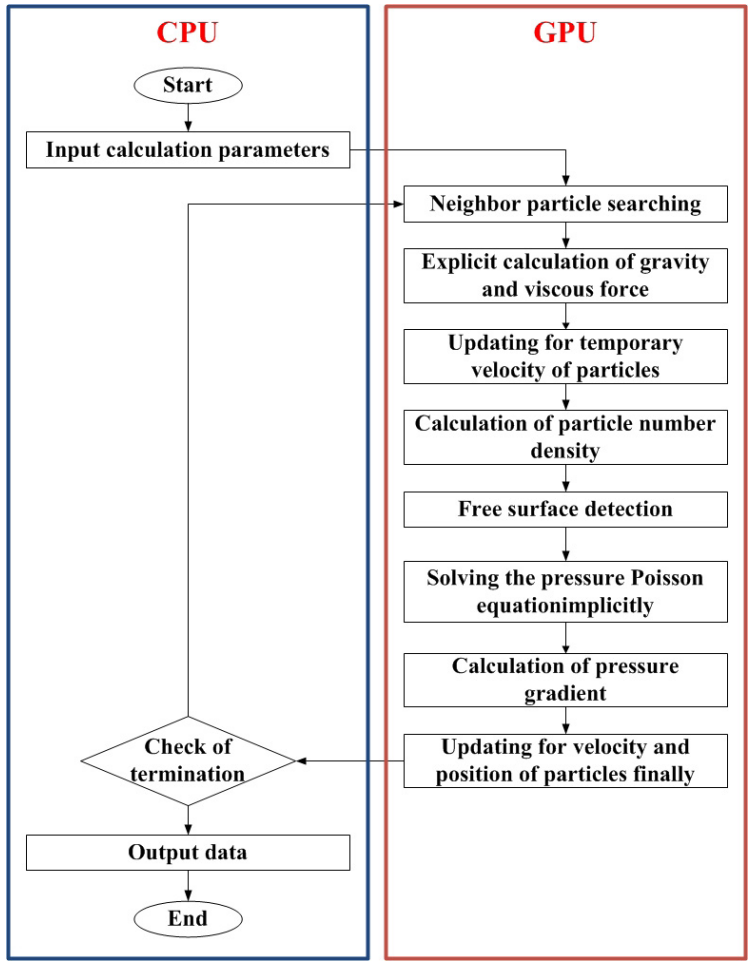


Fig. 2. The flow chart of GPU implementation.

algebra and graph computations based on Thrust. CUSP provides a flexible, high-level interface for manipulating sparse matrices and solving sparse linear systems.

The computational flow chart of MPS method is shown in Fig. 2. One time integration of MPS method is mainly composed of two steps. The first step corresponds to an explicit calculation considering the gravity and viscosity terms. The second step is an implicit calculation accounting for the pressure term. The pressure field of particles is obtained by solving pressure Poisson equation which is discretized into a linear system. The GPU implementation mainly consists of eight steps except the data exchange between GPU and CPU.

Step 1. Neighbor particle searching.

Step 2. Explicit calculation of gravity and viscous force.

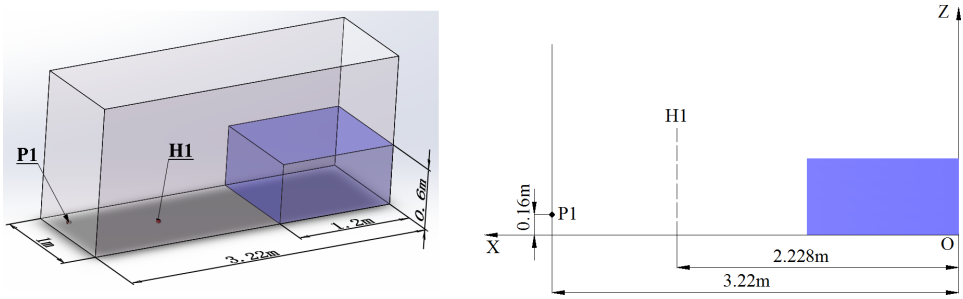


Fig. 3. The sketch of model.

Step 3. Updating for temporary velocity of particles.

Step 4. Calculation of particle number.

Step 5. Free surface detection.

Step 6. Solving the pressure Poisson equation implicitly.

Step 7. Calculation of pressure gradient.

Step 8. Updating for velocity and position of particles finally.

4. Numerical Simulation

In this section, the results of GPU simulations are obtained by running MPSGPU-SJTU solver. In addition, another in-house CPU solver MLParticle-SJTU is used to compare with GPU solver. The reliability of MLParticle-SJTU solver was validated by many violent flow cases in previous papers [Zhang *et al.* (2011, 2016); Yang *et al.* (2015); Chen *et al.* (2017)]. The comparisons between CPU and GPU include fluid field, monitoring data, computation time and so on. In this paper, all simulations are performed on high-performance computing (HPC) cluster "π" in our university equipped with Intel(R) Xeon(R) E5-2670 (8 Cores, 2.6 GHz, 20 MB Cache, 8.0 GT). The GPU card is NVIDIA Tesla K40M, which has 2880 CUDA cores with 12GB graphics memory. Table 1 shows the parameters of computing devices. All data are saved by double precision floating point in both CPU and GPU solvers.

Table 1. Computational environment of CPU and GPU.

	HPC	GPU
Card	Intel(R) Xeon(R) E5-2670	Tesla K40M
Memory	DDR3 1600, 16GB	12GB
Max cores	8	2880
Programming language	C++	CUDA C/C++
Compiler	gcc, MVAPICH	CUDA 7.0, Cusp v0.5.1

4.1. *Dam break flow*

Dam break flow is a typical violent free surface flow with complex phenomena such as the overturning of free surface, splashing and jet flow. In this sub-section, a 3D dam break flow is numerically simulated by MPSGPU-SJTU solver and MLParticle-SJTU solver, respectively. The numerical model is the same as the experimental facility given by Zhou *et al.* [1999] and Buchner [2002]. Figure 3 shows the sketch of computational domain. For fluid domain, the height of water column is 0.6 m and the length is 1.2 m. One pressure probe and one wave gauge are placed in the tank to measure the impact pressure on lateral wall and wave height. The arrangements of monitoring points are listed in Table 2. In this case, the initial particle space is 0.01 m. In total, 1,199,205 particles with 712,800 fluid particles are used to model. The time step is 2.5×10^{-4} s and the density of liquid is 1,000 kg/m^3.

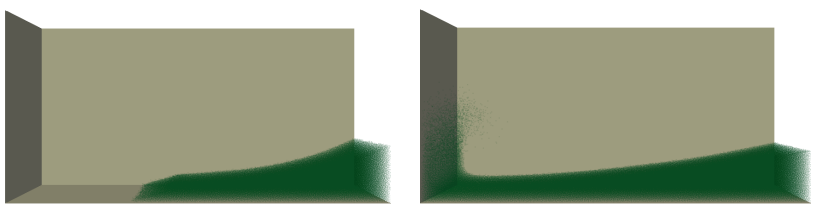
The comparisons of flow fields by experiment, GPU and CPU solvers are shown in Fig. 4. The numerical flow fields of GPU are in good agreement with those of

Table 2. Arrangements of probes.

	X (m)	Y (m)	Z (m)
H1	2.228	0	0
P1	3.22	0.5	0.16

(a1) $t(g/H)^{1/2} = 1.82$ (a2) $t(g/H)^{1/2} = 2.83$

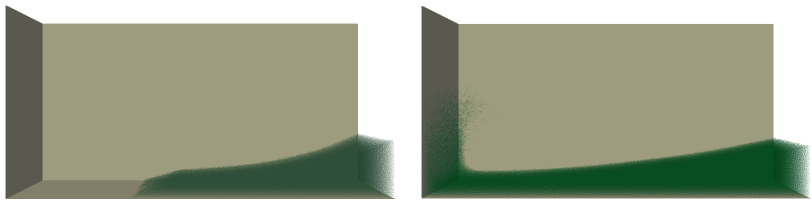
(a) Exp. [Buchner (2002)]



(b1) $t(g/H)^{1/2} = 1.82$ (b2) $t(g/H)^{1/2} = 2.83$

(b) GPU (MPSGPU-SJTU solver)

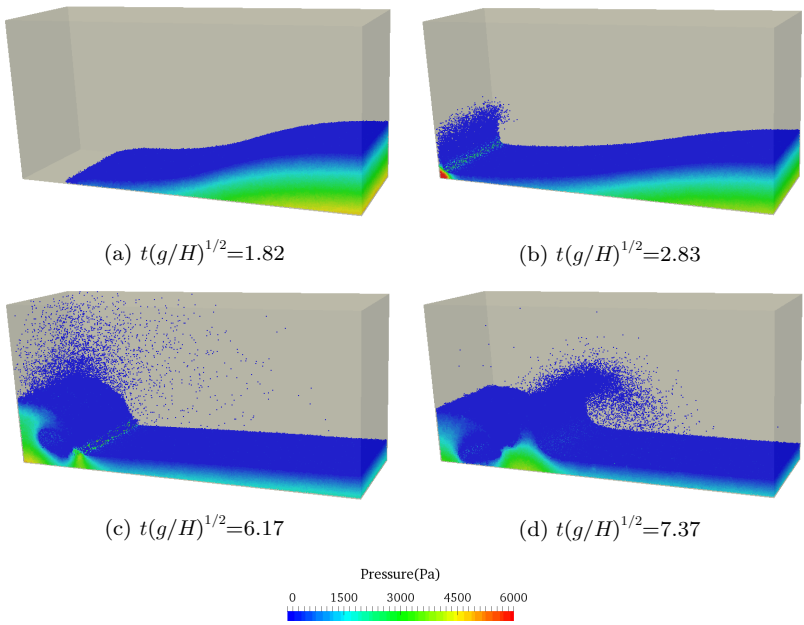
Fig. 4. Comparisons of flow fields by experiment, GPU and CPU solvers.



(c1) $t(g/H)^{1/2} = 1.82$ (c2) $t(g/H)^{1/2} = 2.83$

(c) CPU (MLParticle-SJTU solver)

Fig. 4. (*Continued*)



(a) $t(g/H)^{1/2}$=1.82 (b) $t(g/H)^{1/2}$=2.83

(c) $t(g/H)^{1/2}$=6.17 (d) $t(g/H)^{1/2}$=7.37

Fig. 5. The flow fields by GPU simulation (MPSGPU-SJTU solver).

CPU and experiment. Figure 5 shows some snapshots of dam break flow by GPU simulation. After the water column is released, the water front firstly moves along the dry bottom of tank. The free surface is smooth and the pressure of fluid field is equal to hydrostatic pressure. Then the water front impacts the corner of tank and the pressure around the corner increases suddenly. The water front runs up along the lateral wall and a part of fluid splashes. Under the action of gravity, the subsequent fluid forms the overturn of free surface and falls into the lower fluid domain. The fallen water impacts the surface and generates the second successive overturning wave.

In addition, some calculated results of GPU (MPSGPU-SJTU solver) are also compared with the data of CPU (MLParticle-SJTU solver), experiment

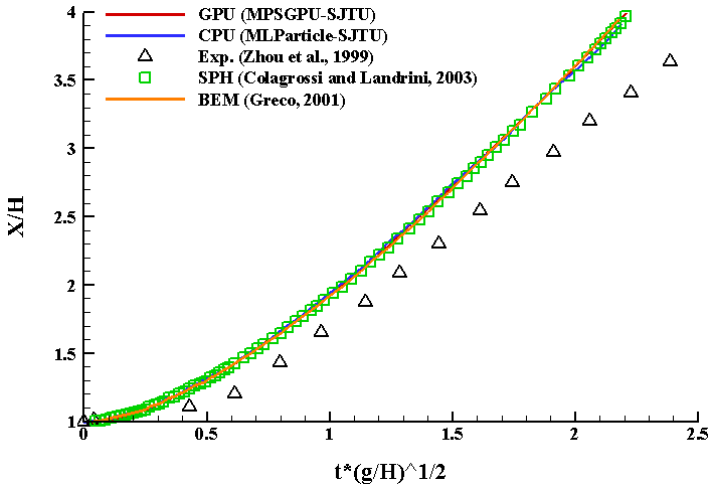


Fig. 6. The water-front of different researches.

[Zhou *et al.* (1999)], SPH [Colagrossi and Landrini (2003)] and BEM [Greco (2001)]. Figure 6 gives the propagation of water front. All the results show that the fluid accelerates smoothly and reaches a stable velocity. The wave front propagation along the dry bottom of tank by GPU simulation is similar to that of CPU and also in good agreement with the results of SPH and BEM. However, the experimental propagation speed of water front is slower than the results of numerical simulation.

Figure 7 shows the comparison of impact pressure on the right wall among numerical methods and experiment. The overall tendency of pressure history by MPSGPU-SJTU solver shows a good agreement with CPU solver, experiment, SPH

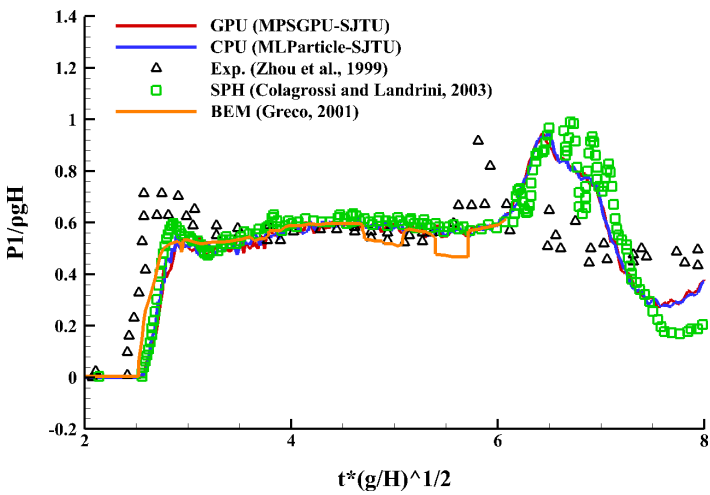


Fig. 7. The impact pressure of different researches.

and BEM. There are two peak values in the pressure history. After the process of water front moves along the bottom of tank, the first peak value of pressure is suddenly caused by the impact of fluid on the side wall. And the second pressure peak results from the impact of fallen overturning water on the free surface of lower fluid domain. There is a clear phase difference and value difference of the second pressure peak between numerical results and experimental data.

The variation of wave height by numerical simulations and experiment is shown in Fig. 8. Because of the transition from dry-deck condition to wet-deck condition, the wave height increases suddenly. Then the wave height rises slowly with the process of fluid movement. Due to the overturn of water front, the wave height reaches to the maximum value in whole process. The peak value of wave height by numerical simulations is obviously higher than that of experiment. Two phase model mentioned by Colagrossi and Landrini [2003] is the possible approach to deal with this difference between numerical methods and experiment. In addition, BEM can't predict the impact pressure and wave height at the nonlinear stage. However, these results can be recorded by MPS and SPH methods.

In this paper, one thousand time steps from 1.0 s to 1.25 s by GPU and CPU simulations are selected to compare. Figure 9 shows the average computation time of every part in one step by GPU and CPU solvers. For MPS, solving pressure Poisson equation has the most amount of time which limits the computational efficiency. The computation time of every part decreases with the increase of CPU cores.

The speedup ratio is used to evaluate the acceleration effect by using GPU technique. The speedup ratio can be calculated as:

$$\text{speedup ratio} = \frac{\text{TotalTime}_{\text{CPU}}}{\text{TotalTime}_{\text{GPU}}}, \tag{12}$$

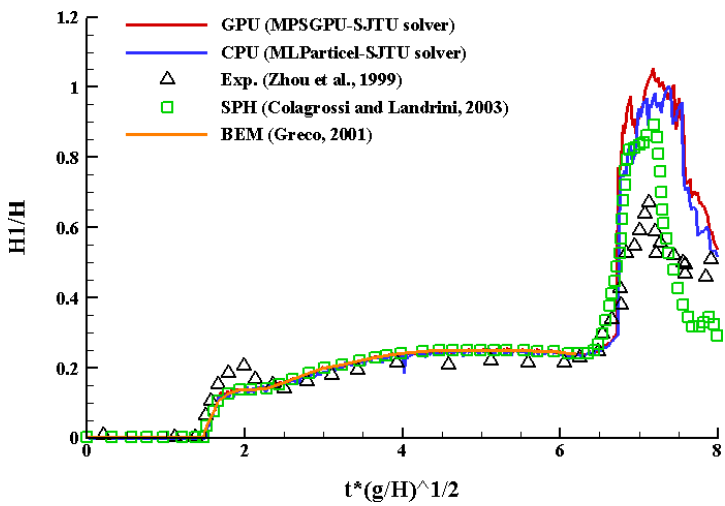


Fig. 8. The wave height of different researches.

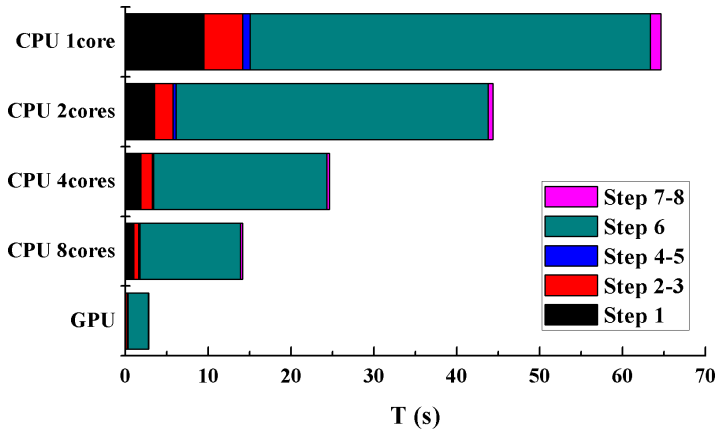


Fig. 9. The computation times of GPU and CPU.

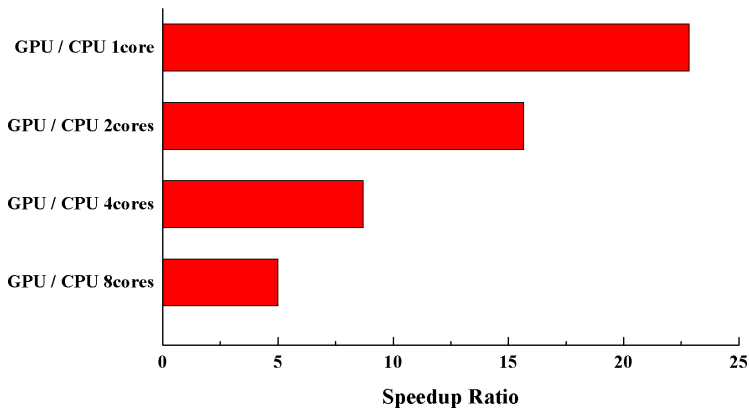


Fig. 10. The speedup ration of GPU.

where $\text{TotalTime}_{\text{CPU}}$ and $\text{TotalTime}_{\text{GPU}}$ correspond to the total computation time in one step on CPU and GPU, respectively. The computational efficiency of GPU is remarkable and the speedup ratio is up to 22.84 by comparing to one CPU core in Fig. 10.

4.2. *Sloshing flow*

A 3D liquid tank under horizontal excitation which is the same as the experimental model given by Song *et al.* [2013] is selected as numerical model to simulate in this sub-section. The sketch and geometric parameters of the liquid tank are shown in Fig. 11. The length of tank is 0.79 m, the width and the height are 0.48 m. The filling level is 30% and the corresponding depth of water is 0.144 m. One pressure probe whose arrangement is listed in Table 3 is placed at the lateral wall to measure

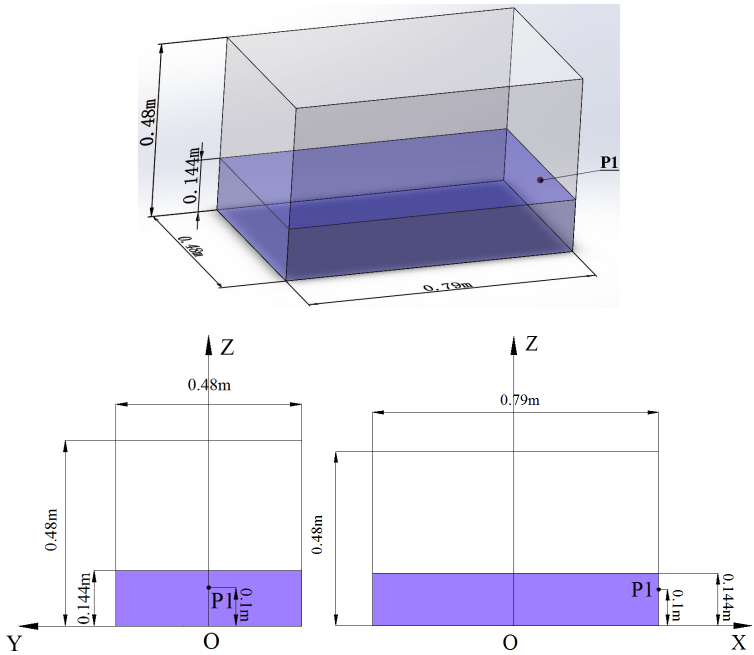


Fig. 11. The sketch of model.

Table 3. Arrangement of the pressure probe.

	X (m)	Y (m)	Z (m)
P1	0.395	0	0.1

the variation of impact pressure. The whole sloshing system is subject to move by the external surge excitation:

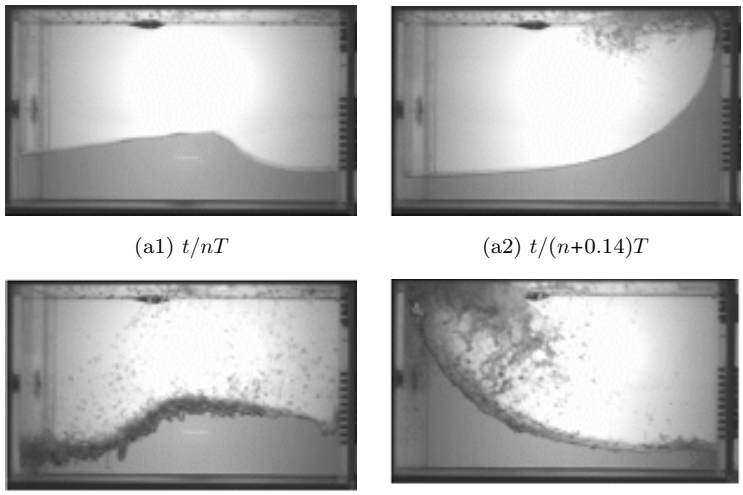
$$x = A \cdot \sin(\omega \cdot t), \tag{13}$$

where A is the amplitude of excitation with the value of 0.0575 m and ω is the excitation frequency which is set to 4.49 rad/s. The initial particle space is 0.005 m, the time step is 2×10^{-4} s and the density of liquid is 1,000 kg/m^3. In this case, total 678373 particles including 432535 fluid particles are used to simulate this model.

Some snapshots of experimental and numerical flow fields are shown in Fig. 12. The free surfaces of GPU simulation are in good agreement with those of CPU calculation and experiment. The fluid field is forced to move by the movement of tank. The sloshing wave travels to the right walls when the tank moves right. Then the water front impacts and climbs along the lateral wall. And the front of fluid hits the corner and forms jet flow which still spreads along the ceiling of tank. Under the action of gravity, the jet flow falls into the lower fluid field and moves to the left wall.

From Fig. 12, the obvious nonlinear phenomena such as overturning water and liquid splash can be observed. These large deformation and nonlinear fragmentation of free surface are simulated by both MPSGPU-SJTU and MLParticle-SJTU solvers.

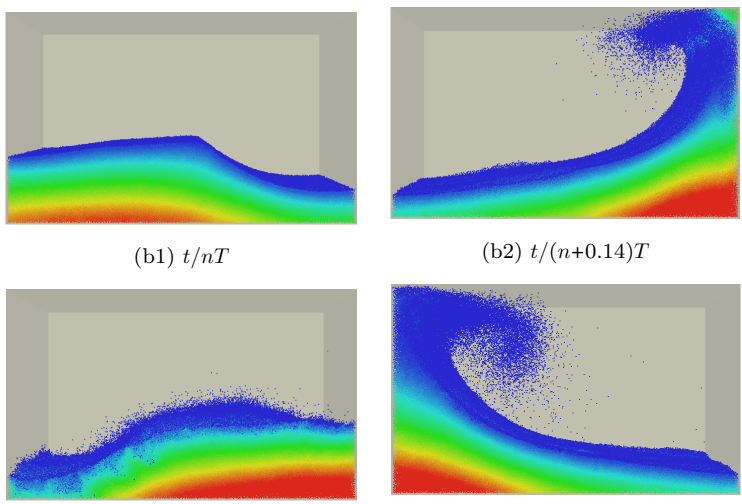
In addition, the comparison of pressure histories among GPU, CPU and experiment is shown in Fig. 13. A typical impact pressure pattern "church roof" can be



(a1) t/nT (a2) $t/(n{+}0.14)T$

(a3) $t/(n{+}0.42)T$ (a4) $t/(n{+}0.7)T$

(a) Exp. (Song et al., 2013)



(b1) t/nT (b2) $t/(n{+}0.14)T$

(b3) $t/(n{+}0.42)T$ (b4) $t/(n{+}0.7)T$

(b) GPU (MPSGPU-SJTU solver)

Fig. 12. The flow fields of different researches.

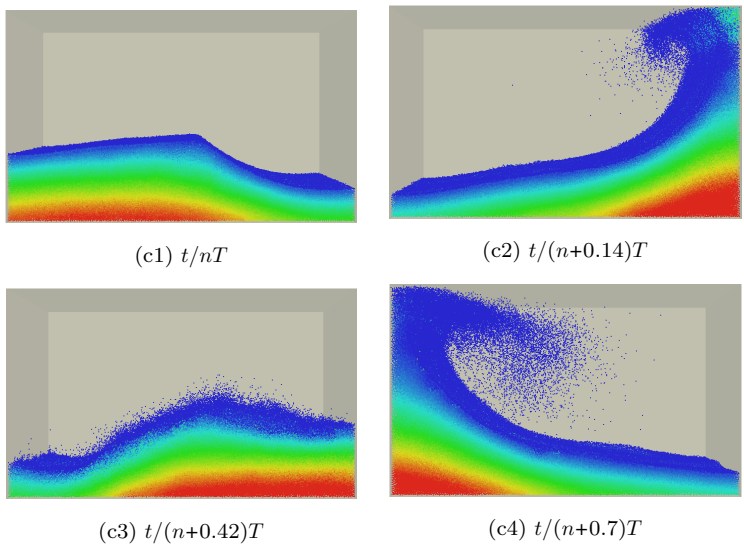


Fig. 12. (*Continued*)

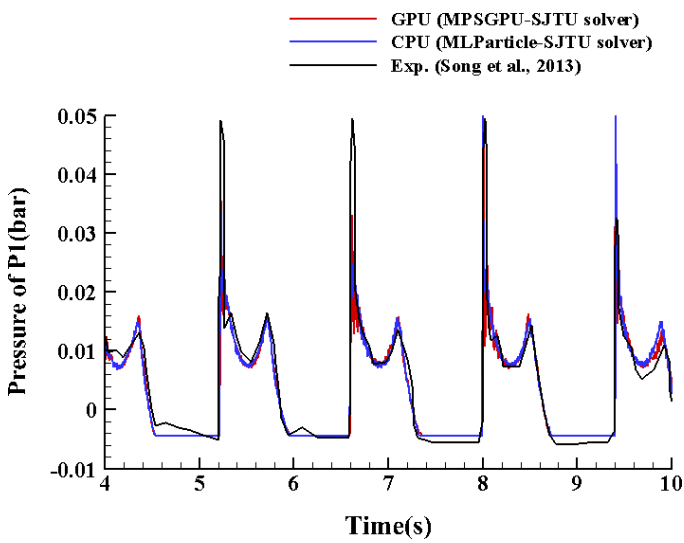


Fig. 13. The impact pressure of different researches.

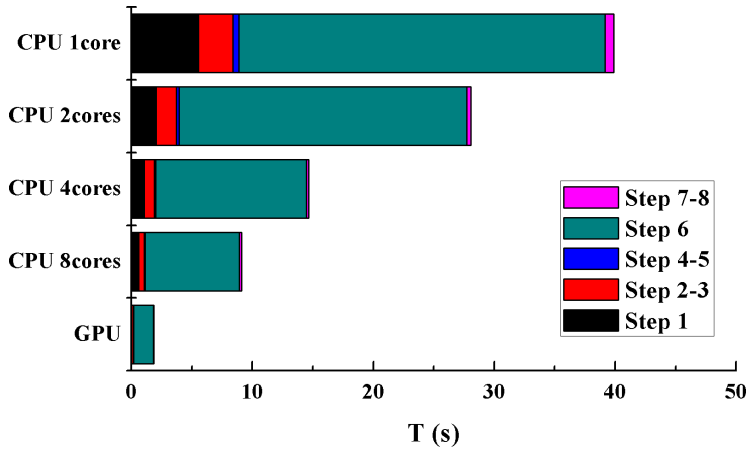


Fig. 14. The computation times of GPU and CPU.

observed in the figure. When the liquid tank reaches its maximum position on the right and starts to move left, the fluid field still moves to and impacts the right wall, which results in an instant pressure peak. The sloshing flow runs up along the right wall with the decrease of pressure. The fallen fluid which drops down on the free surface under the gravity causes the second pressure peak. Finally, the pressure is less than zero, because the depth of water near right-side wall is below pressure probes while the fluid field moves to the left-side wall.

Figure 14 also gives the comparison of average computation time in one step between GPU and CPU through 1,000 selected time steps. The performance of GPU is outstanding in the conditions of guaranteeing the accuracy. The GPU solver can reduce the computation time of every part. In addition, the acceleration ratio between GPU and one CPU core is up to 21.39 from Fig. 15.

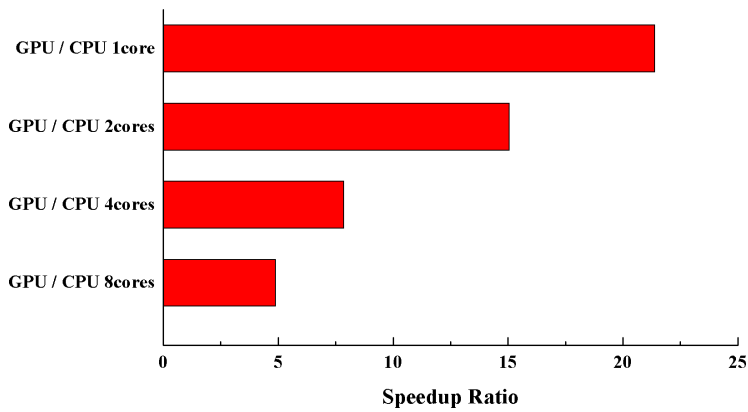


Fig. 15. The speedup ration of GPU.

5. Conclusions

In this work, the GPU acceleration technique is applied to improve the computational efficiency of MPS method. Based on modified MPS and GPU technique, an in-house solver MPSGPU-SJTU is developed to simulate the 3D large-scale violent flows such as dam break and sloshing. The large deformation and nonlinear fragmentation of free surface, like overturning wave, jet flow, splashing and so on, can be observed clearly in these numerical simulations. The numerical results such as impact pressure on the lateral wall, wave height and water front by GPU simulation shows a good agreement with those by CPU calculation, experiment, SPH and BEM. These comparisons demonstrate the validity of MPSGPU-SJTU solver. In addition, the GPU acceleration technique can dramatically improve the computational efficiency by comparing the computation time between GPU and CPU solvers. The speedup ratio of every calculation step between GPU and CPU solvers is up to 22.84.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (51879159, 51490675, 11432009, 51579145), Chang Jiang Scholars Program (T2014099), Shanghai Excellent Academic Leaders Program (17XD1402300), Program for Professor of Special Appointment (Eastern Scholar) at Shanghai Institutions of Higher Learning (2013022), Innovative Special Project of Numerical Tank of Ministry of Industry and Information Technology of China (2016-23/09) and Lloyd's Register Foundation for doctoral student, to which the authors are most grateful.

Appendix A

The different codes of particle position update between CPU and GPU are shown in Figs. A.1 and A.2. In CPU code, the particle position is orderly updated according to the particle ID. All calculation steps are executed on CPU. However, the GPU code includes two parts executed on CPU and GPU, respectively. Firstly, the data

```
void PositionUpdate(double* position, double* velocity)
{
    for(int i = 0; i < N; ++i) //i: particle ID; N: total particle number
    {
        position[i] += velocity[i]*dt;
    }
}
```

Fig. A.1. Code of particle position update on CPU.

device code

```
__global__ void PositionUpdate (double* position_g, double* velocity_g)
{
    int i = threadIdx.x; //i: not only particle ID but also thread ID
    if (i < N) //N: total particle number
    {
        position_g[i] += velocity_g[i]*dt;
    }
}
```

host code

```
int main()
{
    //copy CPU data to GPU
    cudaMemcpy(position_g, position_c, N*sizeof(double), cudaMemcpyHostToDevice);
    cudaMemcpy(velocity_g, velocity_c, N*sizeof(double), cudaMemcpyHostToDevice);

    //run device code
    PositionUpdate <<<1,1>>>(position_g, velocity_g);

    //copy GPU data to CPU
    cudaMemcpy(position_c, position_g, N*sizeof(double), cudaMemcpyDeviceToHost);
    cudaMemcpy(velocity_c, velocity_g, N*sizeof(double), cudaMemcpyDeviceToHost);

    return 0;
}
```

Fig. A.2. Code of particle position update on GPU.

are copied from CPU to GPU, which is executed by the host code. Then, the particle position is updated by the device code. Here, GPU can update the positions of many particles based on the number of threads, which is the biggest difference from CPU. For example, if there are 1,000 threads on GPU, the positions of 1,000 particles can be updated simultaneously. The index i not only represents particle ID, but also represents thread ID of GPU. This index ensures one-to-one correspondence between particle and thread. Finally, the updated particle position is transferred from GPU to CPU.

References

Ataie-Ashtiani, B. and Farhadi, L. [2006] "A stable moving-particle semi-implicit method for free surface flows," *Fluid Dyn. Res.* **38**, 241–256.

Buchner, B. [2002] "Green water on ship-type offshore structures," Ph.D. Thesis, Delft University of Technology.

Chen, X., Rao, C. P. and Wan, D. C. [2017] "Numerical simulation of water entry for two-dimensional wedge by MPS," *Chin. J. Comput. Mech.* **34**, 356–362.

Colagrossi, A. and Landrini, M. [2003] "Numerical simulation of interfacial flows by smoothed particle hydrodynamics," *J. Comput. Phys.* **191**, 448–475.

Crespo, A. J. C., Domínguez, J. M., Barreiro, A., Gómez-Gesteira, M. and Rogers, B. D. [2011] "GPUs, a new tool of acceleration in CFD: Efficiency and reliability on smoothed particle hydrodynamics methods," *PLoS One* **6**, e20685.

CUDA Toolkit Documentation v8.0.61, http://docs.nvidia.com/cuda/. 2017.

CUSP, https://developer.nvidia.com/cusp. 2017.

Domínguez, J. M., Crespo, A. J. C. and Gómez-Gesteira, M. [2013] "Optimization strategies for CPU and GPU implementations of a smoothed particle hydrodynamics method," *Comput. Phys. Commun.* **184**, 617–627.

Gou, W., Zhang, S. and Zheng, Y. [2016] "Simulation of isothermal multi-phase fuel-coolant interaction using MPS method with GPU acceleration," *Kerntechnik* **81**, 330–336.

Greco, M. [2001] *A Two-Dimensional Study of Green-Water Loading*, Ph.D. Thesis, Norwegian University of Science and Technology.

Harada, T., Koshizuka, S. and Kawaguchi, Y. [2007] "Smoothed particle hydrodynamics on GPUs," *Structure* **4**, 671–691.

Hori, C., Gotoh, H., Ikari, H. and Khayyer, A. [2011] "GPU-acceleration for moving particle semi-implicit method," *Comput. Fluids* **51**, 174–183.

Ikari, H. and Gotoh, H. [2008] "Parallelization of MPS method for 3D wave analysis," Advances in Hydro-science and Engineering, *8th Int. Conf. Hydro-Science and Engineering* (*ICHE*), Nagoya, Japan.

Ikari, H., Khayyer, A. and 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**, 361–376.

Iribe, T., Fujisawa, T. and Koshizuka, S. [2010] "Reduction of communication in parallel computing of particle method for flow simulation of seaside areas," *Coastal Eng. J.* **52**, 287–304.

Khayyer, A. and Gotoh, H. [2012] "A 3D higher-order Laplacian model for enhancement and stabilization of pressure calculation in 3D MPS-based simulations," *Appl. Ocean Res.* **37**, 120–126.

Khayyer, A. and Gotoh, H. [2008] "Development of CMPS method for accurate water-surface tracking in breaking waves," *Coastal Eng. J.* **50**, 179–207.

Khayyer, A. and Gotoh, H. [2009] "Modified moving particle semi-implicit methods for the prediction of 2D wave impact pressure," *Coastal Eng.* **56**, 419–440.

Kondo, M. and Koshizuka, S. [2011] "Improvement of stability in moving particle semi-implicit method," *Int. J. Numer. Meth. Fluids* **65**, 638–654.

Koshizuka, S. and Oka, Y. [1996] "Moving-particle semi-implicit method for fragmentation of incompressible fluid," *Nucl. Sci. Eng.* **123**(3), 421–434.

Koshizuka, S., Nobe, A. and Oka, Y. [1998] "Numerical analysis of breaking waves using the moving particle semi-implicit method," *Int. J. Numer. Meth. Fluids* **26**, 751–769.

Lee, B. H., Park, J. C., Kim, M. H. and Hwang, S. C. [2011] "Step-by-step improvement of MPS method in simulating violent free-surface motions and impact-loads," *Comput. Meth. Appl. Mech. Eng.* **200**, 1113–1125.

Li, H. Z., Zhang, Y. L. and Wan, D. C. [2015] "GPU based acceleration of MPS for 3D free surface flows," *Proc. 9th Int. Workshop on Ship and Marine Hydrodynamics*, 1–7.

Mokos, A., Rogers, B. D., Stansby, P. K. and Domínguez, J. M. [2015] "Multi-phase SPH modelling of violent hydrodynamics on GPUs," *Comput. Phys. Commun.* **196**, 304–316.

Shibata, K., Koshizuka, S., Sakai, M. and Tanizawa, K. [2012] "Lagrangian simulations of ship-wave interactions in rough seas," *Ocean Eng.* **42**, 13–25.

Song, Y. K., Chang, K. A., Ryu, Y. and Kwon, S. H. [2013] "Experimental study on flow kinematics and impact pressure in liquid sloshing," *Exp. Fluids* **54**, 1–20.

Tanaka, M. and Masunaga, T. [2010] "Stabilization and smoothing of pressure in MPS method by quasi-compressibility," *J. Comput. Phys.* **229**, 4279–4290.

Tang, Z. Y., Wan, D. C., Chen, G. and Xiao, Q. [2016b] "Numerical simulation of 3D violent free-surface flows by multi-resolution MPS method," *J. Ocean Eng. Marine Energy* **2**, 355–364.

Tang, Z. Y., Zhang, Y. L. and Wan, D. C. [2016] "Numerical simulation of 3D free surface flows by overlapping MPS," *J. Hydrodyn.* **28**, 306–312.

Tang, Z. Y., Zhang, Y. L. and Wan, D. C. [2016a] "Multi-resolution MPS method for free surface flows," *Int. J. Comput. Meth.* **13**, 1641018.

Tsuruta, N., Khayyer, A. and Gotoh, H. [2013] "A short note on dynamic stabilization of moving particle semi-implicit method," *Comput. Fluids* **82**, 158–164.

Yang, Y. Q., Tang, Z. Y., Zhang, Y. L. and Wan, D. C. [2015] "Investigation of excitation period effects on 2D liquid sloshing by MPS method," *Proc. Twenty-fifth Int. Ocean and Polar Eng. Conf.* pp. 937–944.

Zhang, Y. X. and Wan, D. C. [2012] "Numerical simulation of liquid sloshing in low-filling tank by MPS," *J. Hydrodyn.* **27**, 101–107.

Zhang, Y. L., Tang, Z. Y. and Wan, D. C. [2016] "Numerical investigations of waves interacting with free rolling body by modified MPS method," *Int. J. Comput. Meth.* **13**, 1641013-1–1641013-14.

Zhang, C., Zhang, Y. X. and Wan, D. C. [2011] "Comparative study of SPH and MPS methods for numerical simulations of dam breaking problems," *Chin. J. Hydrodyn.* **26**, 736–746.

Zhou, Z. Q., De Kat, J. O. and Buchner, B. [1999] "A nonlinear 3D approach to simulate green water dynamics on deck," *7th Int. Conf. Num. Ship Hydrod.* Nantes, France.

Zhu, X. S., Cheng, L., Lu, L. and Teng, B. [2011] "Implementation of the moving particle semi-implicit method on GPU," *Sci. China* **54**, 523–532.