GPU Acceleration of MPS for Three-Dimensional Sloshing

Xiang Chen[*], Youlin Zhang, 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, China

[*]Presenting author: just_chenxiang@163.com

[‡]Corresponding author: dcwan@sjtu.edu.cn

http://dcwan.sjtu.edu.cn

Abstract

In this paper, the application of a GPU-based particle method to three-dimensional sloshing problem is presented. Moving particle semi-implicit (MPS) method is a Lagrangian method which can be used to simulate nonlinear flow effectively. But one of its drawbacks is the high computation cost with the increase of particle number. Based on modified MPS, the MPS-GPU-SJTU solver is developed to simulate a large sum of particles by using GPU which supports large-scale scientific computations. In addition, one optimization strategy is applied to reduce the storage and computation cost of Poisson equation of pressure (PPE). Then the convergent validation is carried out to verify the accuracy of present solver. And the accuracy and performance of GPU-based solver are investigated by comparing the results with those by CPU. As a summary of results, the GPU-based solver shows a good agreement with CPU solver (MLParticle-SJTU). And the computation efficiency of GPU is much higher than CPU.

Keywords: Moving particle semi-implicit (MPS); GPU acceleration; MPS-GPU-SJTU solver; sloshing

Introduction

With the development of economy, the demand of energy is increasing. Many countries which lack energy import liquefied natural gas, oil and liquefied petroleum gas by the transportation of vessels such as LNG, LPG, VLCC and so on. Because of the variable sea conditions, the liquid in a partially filled tank will be prone to complicated and nonlinear sloshing phenomenon. The instantaneous impact pressure induced by liquid sloshing may destroy the structure of tank walls and the stability of the ship. Therefore, many researchers have devoted themselves to investigating the characters and mechanisms of sloshing.

Faltisnen (1978) developed the boundary element method (BEM) to study sloshing in a two-dimensional (2-D) rectangular tank under translational excitation [1]. Nakayama and Washizu (1980) used the finite element method (FEM) to study 2-D problem of nonlinear liquid sloshing in a rectangular tank under pitch excitation [2]. Arai et al. (1992) used the Marker-and-cell (MAC) method to simulate numerically a 3-D sloshing phenomenon in liquid tank with vertical baffle [3]. Koh et al. (1998) developed a coupled BEM-FEM method for analyzing 3-D liquid sloshing in rectangular tanks [4]. Kim (2001) studied 2-D and 3-D sloshing in a rectangular tank with a large vertical baffle and three horizontal baffles by using the finite difference method (FDM) [5]. Liu and Lin (2009) modeled the vertical baffle in 2-D and 3-D tank by the virtual boundary force (VBF) method [6]. Yin et al. (2012) employed the volume of fluid (VOF) method by utilizing their inhouse solver naoe-FOAM-SJTU to study the effect of filling rate on sloshing [7].

In addition to the above methods, the fully Lagrangian particle methods are used to research the nonlinear sloshing problem. Delorme et al. (2009) applied the smoothed particle hydrodynamics (SPH) method to research the impact pressure of sloshing in shallow filled tank under roll excitation [8]. Shao et al. (2012) simulated 2-D liquid sloshing by using SPH [9]. Koh et al. (2013) used the consistent particle method (CPM) to investigate the effect of a constrained floating baffle in prismatic tank [10]. Kim et al. (2014) developed MPS method to research multiliquid-layer sloshing problems and investigated the elevation of interface under different excited frequencies and amplitudes [11]. Yang et al. (2015) studied the effects of excitation period on 2-D liquid sloshing by MPS Method [12]. Hashimoto et al. (2016) estimated oil overflow from an oil storage tank subjected to a possible Nankai trough earthquake in Osaka bay area based explicit MPS method [13]. Zhou et al. (2016) modified the pressure correction algorithm and simulated the sloshing of multiphase flows with large density ratio [14]. Chen et al. (2016) used modified MPS method to study the effect of the location of horizontal baffle on liquid sloshing in 3-D tank [15].

Though mesh-free methods can easily track free surface and effectively simulate sloshing problem, they still suffer from high computation cost with the increase of particle number. The GPU (Graphics Processing Unit) is a multi-processor designed to optimize for the execution of massive number of threads. Because of the explicit algorithm, it is easy to apply GPU technology to SPH. Harada et al. (2007) applied the acceleration technique of GPU to SPH and the computation speed of GPU is up to 28 times faster than CPU [16]. Hérault et al. (2010) used CUDA to implement SPH on GPU to simulate the problems of dam break and paddle-generated waves [17]. Crespo et al. (2011) developed a CPU-GPU solver DualSPHysics to deal with free-surface flow problems and achieved a speedup of 64 in the simulation of 3-D dam break by using one million particles [18]. Domínguez et al. (2013) improved DualSPHysics by using several optimizations for the GPU implementations and accelerated serial SPH codes with a speedup of 56.2 [19]. Fourtakas and Rogers (2016) applied a two-phase model based on SPH to simulate the problem of two-phase liquid-sediments flows [20]. However, it is difficult to implement a GPU-based MPS calculation due to the semi-implicit algorithm adopted to obtain the pressure field. Hori et al. (2011) developed a GPU-accelerated MPS code by using CUDA language and simulated 2-D elliptical drop evolution and dam break [21]. Kakuda et al. (2012, 2013) presented a GPU-based MPS to calculate 2-D and 3-D dam break problems compared with CPU simulations and the speed-up is 12 times and 17 times respectively [22]. Li et al. (2015) applied GPU acceleration in the solution of PPE and search of neighboring particles and achieved speedups of 10 and 6 respectively [23]. Gou et al. (2016) simulated the isothermal multi-phase fuel-coolant interaction by using MPS method with GPU acceleration [24].

In this study, the MPS-GPU-SJTU solver based on modified MPS method is employed for numerical simulation of 3-D liquid sloshing by using GPU acceleration. In the first section, a brief description of MPS method is presented. In the second section, the flow chart of implementations on GPU follows. Then one optimization strategy to reduce the storage and computation time of PPE is applied and improves the computation efficiency. In addition, the convergent validation is carried out to verify the accuracy of present solver. And the 3-D sloshing problems are simulated by GPU and CPU solvers at the same time. It is shown that the results of GPU solver show a good agreement with CPU and a large amount of computation time is reduced by GPU.

MPS Method

Koshizuka and Oka (1996) have explained the MPS method in detail [25]. In this section, the numerical models adopted in this paper are introduced briefly.

Governing Equations

The governing equations for incompressible and viscous fluid include conservation equations of mass and momentum.

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

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

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

Particle Interaction Models

Kernel Function

A kernel function is used for all interaction models to describe the particle interaction in MPS method.

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

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.

Gradient Model

The gradient operator is modeled as a local weighted average of the gradient vectors between particle i and its neighboring particle j.

$$<\nabla P>_i = \frac{D}{n^0}\sum_{j\neq i}\frac{P_j + P_i}{|\vec{r}_j - \vec{r}_i|^2}(\vec{r}_j - \vec{r}_i)\cdot W(|\vec{r}_j - \vec{r}_i|) \tag{4}$$

where D is the number of space dimension, n^0 is the initial particle number density and $\vec{r}$ is coordinate vector of fluid particle.

Laplacian Model

The Laplacian operator is modeled by weighted average of the distribution of a quantity ϕ from particle i to its neighboring particle j.

$$<\nabla^2\phi>_i = \frac{2D}{n^0\lambda}\sum_{j\neq i}(\phi_j - \phi_i)\cdot W(|\vec{r}_j - \vec{r}_i|) \tag{5}$$

$$\lambda = \frac{\sum_{j\neq i} W(|\vec{r}_j - \vec{r}_i|)\cdot|\vec{r}_j - \vec{r}_i|^2}{\sum_{j\neq i} W(|\vec{r}_j - \vec{r}_i|)} \tag{6}$$

where λ is applied to make sure that the increase of variance is equal to the analytical solution.

Model of Incompressibility

In this paper, PPE is solved by using a mixed source term method which is developed by Lee et al. (2011) [26].

$$< \nabla^2 P^{k+1} >_i = (1-\gamma)\frac{\rho}{\Delta t}\nabla \cdot \vec{V}_i^* - \gamma \frac{\rho}{\Delta t^2}\frac{< n^* >_i - n^0}{n^0} \tag{7}$$

where γ is a blending parameter which varies from 0 to 1, n^* is the temporal particle number density and Δt is the time step. In this paper, γ =0.01 is employed for all numerical simulations.

Free Surface Detection

Zhang (2012) developed a modified surface particle detection method, which is based on the asymmetry arrangement of neighboring particles [27].

$$< \vec{F} >_i = \frac{D}{n^0}\sum_{j \neq i}\frac{1}{|\vec{r}_i - \vec{r}_j|}(\vec{r}_i - \vec{r}_j)W(r_{ij}) \tag{8}$$

$$< |\vec{F}| >_i \; > \; \alpha \tag{9}$$

$$\alpha = 0.9\left|\vec{F}\right|^0 \tag{10}$$

where $\vec{F}$ is a vector which represents the asymmetry of arrangements of neighbor particles, $\left|\vec{F}\right|^0$ is the initial value of $\left|\vec{F}\right|$.

Boundary Condition

For MPS, multilayer particles are used to present the wall boundary. The wall particles are arranged at the boundary and the pressures of them 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.

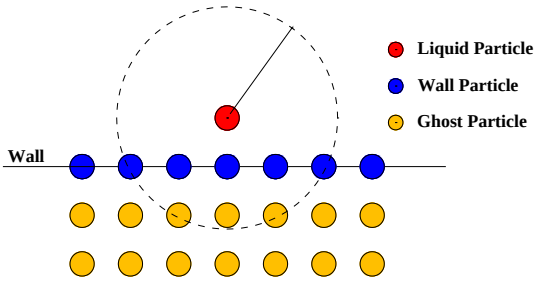


Figure 1. Schematic of boundary particles

Implementations on GPU

It is very important to reduce the CPU cost of MPS with the high-efficient parallelization. As shown in Figure 2, GPU is designed to possess more calculation threads to process data

simultaneously. For example, when one thousand particles are simulated, one thousand threads are launched to calculate. The function of CPU is only to conduct GPU codes to implement and communicate data between GPU. Therefore, GPU is a better choice for high parallel MPS method.

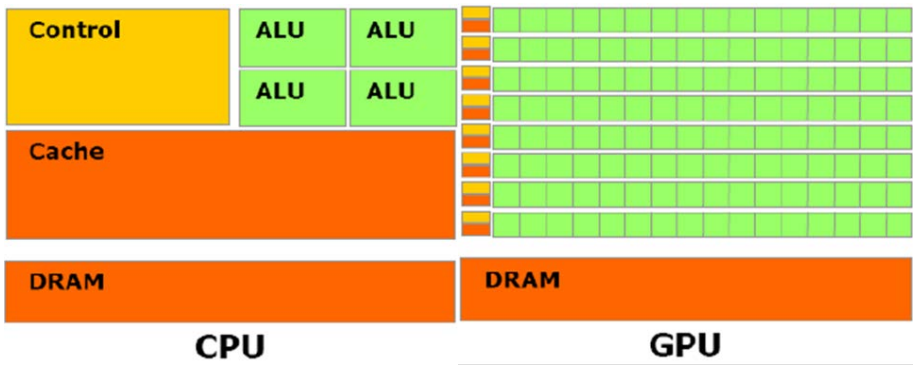


Figure 2. The frameworks of CPU and GPU

For MPS, the time integration is mainly composed of two steps. One step is an explicit calculation considering the gravity and viscosity terms. Another step corresponds to an implicit calculation of PPE. The computational flow chart of MPS on GPU is shown in Figure 3. Except the exchange of data between GPU and CPU, the GPU implementation mainly consists of six steps:

1) Search neighboring particles and create neighbor list
2) Explicit calculation (gravity and viscosity terms)
3) Calculate particle number density and detect free surface
4) Solve PPE to obtain pressure field
5) Calculate pressure gradient
6) Update velocities and positions of particles

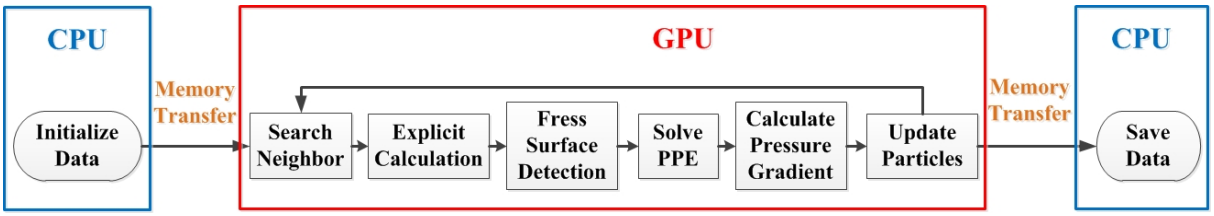


Figure 3. The flow chart of MPS on GPU

Simulation and Result

In this section, a 3-D liquid tank same as the experimental model given by Kim (2001) is selected as the numerical model to simulate. The geometry of liquid tank is shown in Figure 4. The dimensions of tank are 0.8 m (L), 0.35 m (B) and 0.5 m (H), respectively. The depth of water (D) is 0.25 m, corresponding filling level is 50%. The liquid tank is subject to move by the external surge excitation:

$$x = A \cdot \sin(\omega \cdot t) \tag{11}$$

where A is the amplitude of excitation set to be 0.02 m and the excitation frequency ω is 5.39 rad/s which is same as the first natural period of fluid motion in the tank. In addition, two pressure probes are arranged on lateral wall to measure the variation of pressure. The arrangements of pressure probes are listed in Table 1.

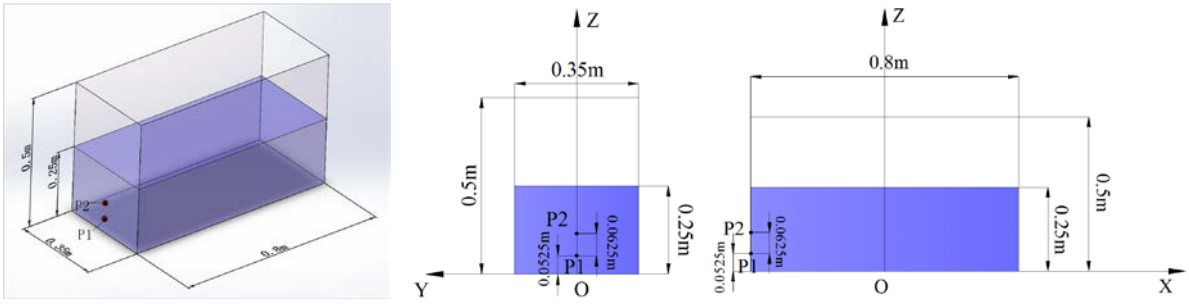


Figure 4. The sketch of numerical model

Table 1. Arrangements of pressure probe

	X/m	Y/m	Z/m
P1	-0.4	0	0.0525
P2	-0.4	0	0.115

All simulations are performed on parallel high performance computing (HPC) with multi cores of Intel(R) Xeon(R) E5-2680 v2, 2.80 GHz. And the GPU device is NVIDIA graphics card Tesla K40M, which has 2880 CUDA cores with 12GB graphics memory. Table 2 shows the parameters of computing devices. In this paper, the double precision floating point computation is only used in both CPU and GPU codes.

Table 2. Computational environment of CPU and CPU

	HPC	GPU
Card	Intel(R) Xeon(R) E5-2680 v2, 2.80 GHz	Tesla K40M
Memory	DDR3 1600, 16GB	12GB
Core	10	2880
Compiler	gcc, MVAPICH	CUDA 7.0, CULA Sparse S6

PPE Optimization

It is well known that the most computation time of MPS is consumed to solve PPE which can be discretized into a linear system **Ax**=**b**. Eq. 12 shows the discretization of linear system. The coefficient matrix of PPE, **A** is a typical sparse matrix. In order to reduce the storage of matrix **A**, the compressed sparse row (CSR) data format is employed. Moreover, the Biconjugate gradient stabilized (BiCGSTAB) method is applied to solve this linear system based on CSR format. Nevertheless, the pressure of ghost particles and free surface particles is no need to solve in PPE and set to zero as the boundary condition. For example, if particle *i* is ghost particle or surface particle, the relevant row of matrix A and array B will be set to constant in Eq. 13. Here, Thrust is a C++ template library for CUDA which provides a rich collection of data parallel primitives such as scan, sort, and reduce [28]. In order to reduce the iteration time and storage of PPE, the relevant rows and columns of ghost and free surface particles are removed from the linear system by using Thrust. Therefore, the matrix dimensions of **A**, **x** and **b** can be reduced in Eq. 14 and the iteration speed of PPE will be faster. The sparse linear algebra library of CULA-Sparse is utilized for accelerating the iteration of PPE [29].

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,i-1} & a_{1,i} & a_{1,i+1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{i-1,1} & \cdots & a_{i-1,i-1} & a_{i-1,i} & a_{i-1,i+1} & \cdots & a_{i-1,n} \\ a_{i,1} & \cdots & a_{i,i-1} & a_{i,i} & a_{i,i+1} & \cdots & a_{i,n} \\ a_{i+1,1} & \cdots & a_{i+1,i-1} & a_{i+1,i} & a_{i+1,i+1} & \cdots & a_{i+1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,i-1} & a_{n,i} & a_{n,i+1} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_i \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{i-1} \\ b_i \\ b_{i+1} \\ \vdots \\ b_n \end{pmatrix} \tag{12}$$

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,i-1} & a_{1,i} & a_{1,i+1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{i-1,1} & \cdots & a_{i-1,i-1} & a_{i-1,i} & a_{i-1,i+1} & \cdots & a_{i-1,n} \\ 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ a_{i+1,1} & \cdots & a_{i+1,i-1} & a_{i+1,i} & a_{i+1,i+1} & \cdots & a_{i+1,n} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,i-1} & a_{n,i} & a_{n,i+1} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_i \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{i-1} \\ 0 \\ b_{i+1} \\ \vdots \\ b_n \end{pmatrix} \tag{13}$$

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,i-1} & a_{1,i+1} & \cdots & a_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{i-1,1} & \cdots & a_{i-1,i-1} & a_{i-1,i+1} & \cdots & a_{i-1,n} \\ a_{i+1,1} & \cdots & a_{i+1,i-1} & a_{i+1,i+1} & \cdots & a_{i+1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,i-1} & a_{n,i+1} & \cdots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_{i-1} \\ x_{i+1} \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{i-1} \\ b_{i+1} \\ \vdots \\ b_n \end{pmatrix} \tag{14}$$

The comparison of computation time between original and optimized PPE is conducted in this sub-section. Total 750793 particles are used to model the liquid tank. The simulation model includes 144812 ghost particles and uncertain free surface particles which are filtered after free surface detection at each step. Figure 5 shows the comparison of pressure history between original and optimized PPE. The overall trends of pressure are almost same for two cases. The relative errors of average pressure peak on both P1 and P2 are small. The calculation time of every step is shown in Figure 6. For optimized PPE, the time cost of solving PPE is about 7% smaller than original PPE. With the increase of ghost and free surface particles, the optimized PPE may reduce more computation time. The concrete results are listed in Table 3.

Table 3. The results of original and optimized PPE

PPE Type	Pressure Peak of P1 (Pa)	Relative error	Pressure Peak of P2 (Pa)	Relative error	Computation time of PPE (s)
Original	3166	-	2630	-	3.101
Optimized	3180	0.44%	2639	0.34%	2.877

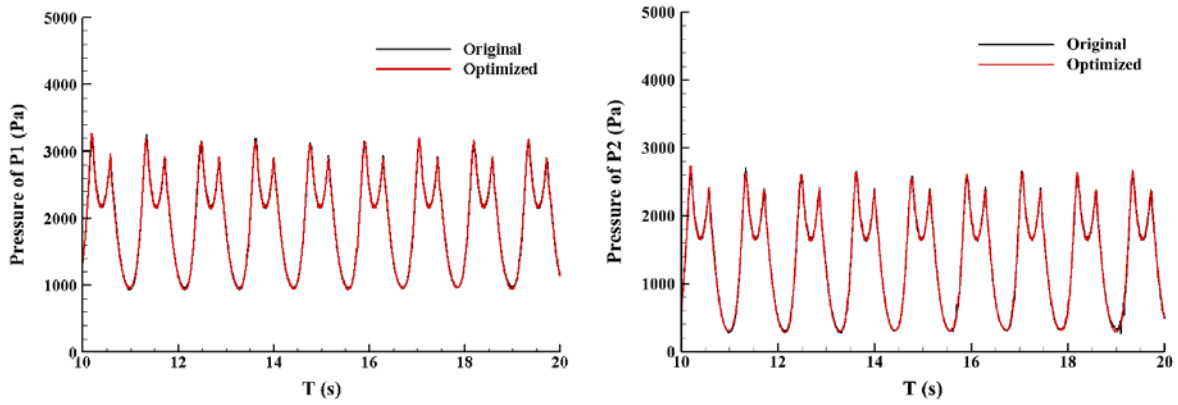


Figure 5. The pressure histories of original and optimized PPE

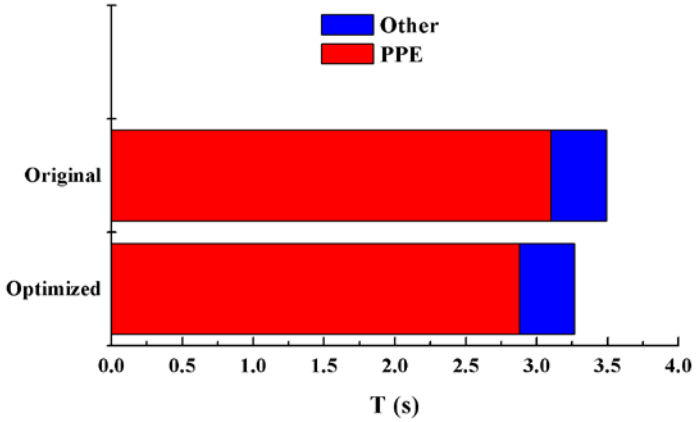


Figure 6. The computation times of original and optimized PPE

Convergence verification

In this sub-section, the convergence verification is conducted to investigate the effect of particle spacing on the numerical results. Three different spatial resolutions (0.006 m, 0.005 m, 0.004 m) are employed to check the convergence of numerical results. The pressure variations of different spatial resolutions are shown in Figure 7. The average values of pressure peak are listed to check the convergence of three cases quantitatively in Table 4. For three spatial resolutions, the variation tendency of pressure is almost same. Furthermore, the relative error is so tiny that the results are convergent with respect to the spatial resolution. In addition, the computation times of different spatial resolution are compared in Figure 8. For every step, the computation time of medium resolution is 1.888 times than coarse resolution. And every step of fine resolution is about 7.086 s which is almost twice than medium resolution. Considering the computation time and accuracy, the medium spatial resolution is selected in following sections.

Table 4. The pressure peaks of different spatial resolutions

Spatial resolution	Particle spacing (m)	Particle number	Pressure Peak of P1 (Pa)	Relative error	Pressure Peak of P2 (Pa)	Relative error
Coarse	0.006	456588	3156	-	2630	-
Medium	0.005	750793	3180	0.76%	2639	0.34%
Fine	0.004	1390732	3224	1.35%	2647	0.30%

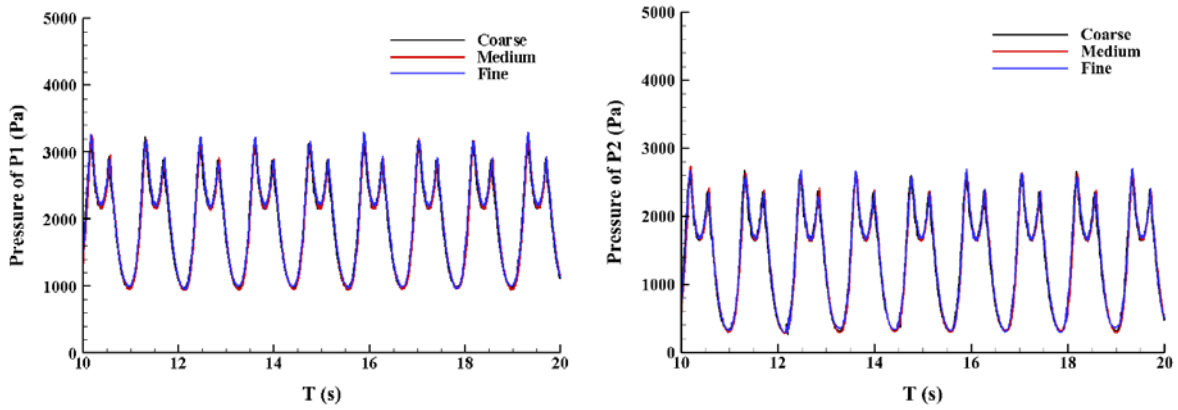


Figure 7. The pressure histories of different spatial resolutions

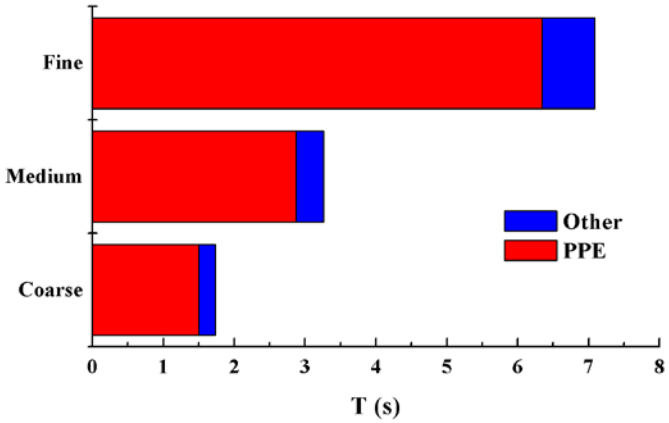
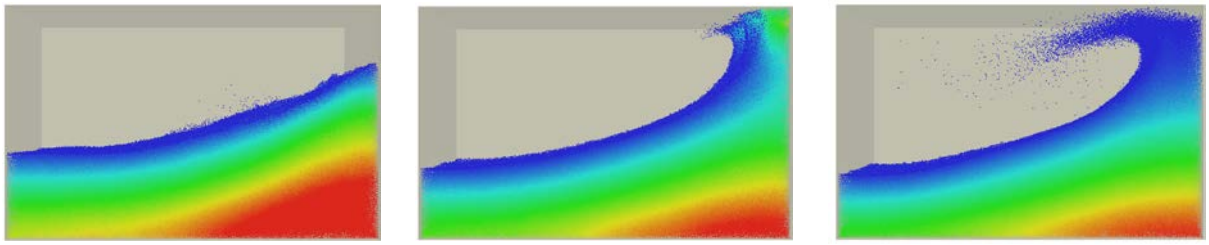


Figure 8. The computation times of different spatial resolutions

3-D Sloshing

In this sub-section, there are some comparisons between CPU and GPU including simulation results, computation time and so on. Figure 9 shows some snapshots of numerical flow field. The nonlinear deformation and large fragmentation of free surface can be observed. The numerical flow field of GPU is in good agreement with CPU simulation. Many details of flow field such as overturning wave, local splashing and travelling wave are clearly captured. The flow of fluid is opposite to the movement of liquid tank. The tank moves to the left lateral wall while fluid flows to right. Therefore, the fluid runs up along lateral wall and impacts the ceiling of the tank. Then fluid spreads along the roof and drops down under the action of gravity. In this process, a part of water even splashes on the right side wall. Then the sloshing wave travels to the left side wall.

CPU 10cores



$t \approx 15.35$ s $t \approx 15.45$ s $t \approx 15.55$ s

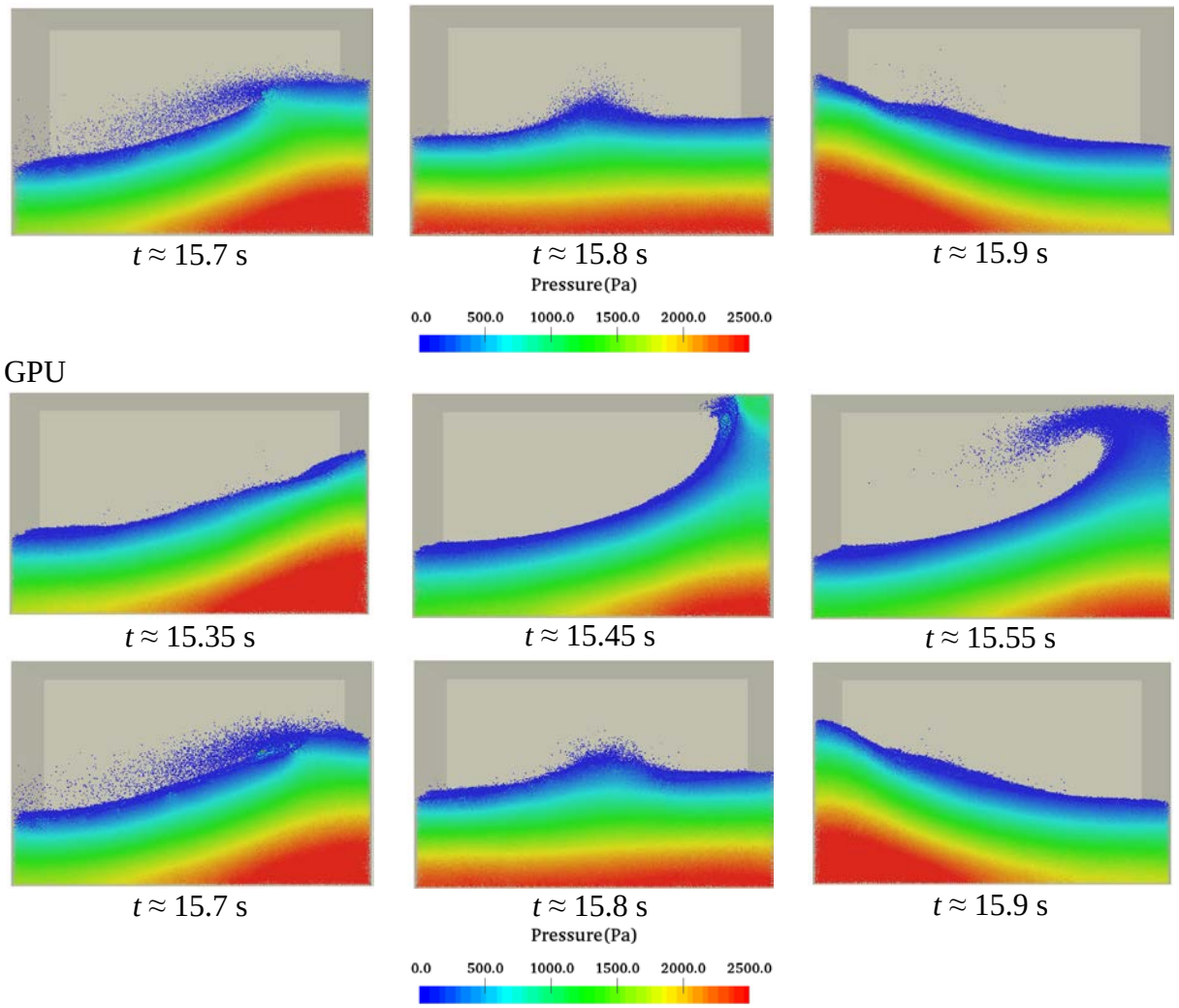


Figure 9. The flow fields of CPU and GPU

In addition, the numerical pressure histories of CPU and GPU are shown in Figure 10. The pressures of GPU simulation on two probes show a good congruency with CPU results. Two successive pressure peaks in each period can be observed. Because of the phase difference between fluid and tank, the sloshing wave that impacts on lateral wall induces the first pressure peak. Then the pressure decreases when fluid runs up along lateral wall. And the second pressure peak results from the fallen water which spreads along the roof and drops down on the free surface. Figure 11 shows the spectrum analysis of GPU results. When frequency is 0 Hz, the spectrum amplitudes of P1 and P2 are similar to hydrostatic pressure. When the frequency is same as excitation frequency, a peak of spectrum amplitude explains the first successive pressure peak. As frequency is twice as excitation frequency, there is the second peak of spectrum amplitude corresponding to the second pressure peak in pressure history.

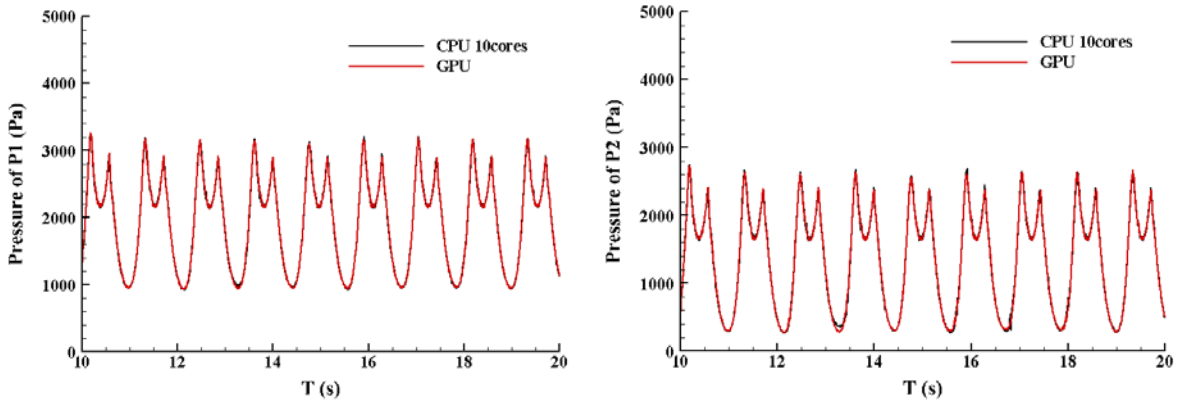


Figure 10. The pressure histories of CPU and GPU

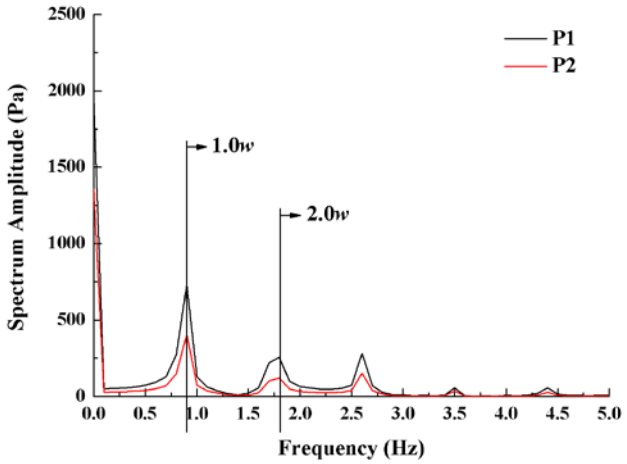


Figure 11. Spectrum of numerical pressures

In this paper, various schemes of multi cores are implemented to run CPU code on HPC. Figure 12 shows the computation times on CPU and GPU devices. The specific computation time of every step is listed in Table 5. No matter CPU and GPU, solving PPE is the most cost for calculation. From Figure 12, the total time decreases with the increase of CPU cores. Comparing GPU and CPU one core, the speedups of PPE and total time are 25.22 and 24.05, respectively. Therefore, how to solve PPE quickly is the greatest problem for researchers.

Table 5. The computation times of CPU and GPU

	CPU 1core	CPU 2cores	CPU 4cores	CPU 6cores	CPU 8cores	CPU 10cores	GPU
PPE	72.562	40.167	25.398	20.667	17.892	16.607	2.877
Other	6.011	3.416	1.872	1.592	1.404	1.424	0.390
Total	78.573	43.584	27.270	22.259	19.295	18.032	3.267

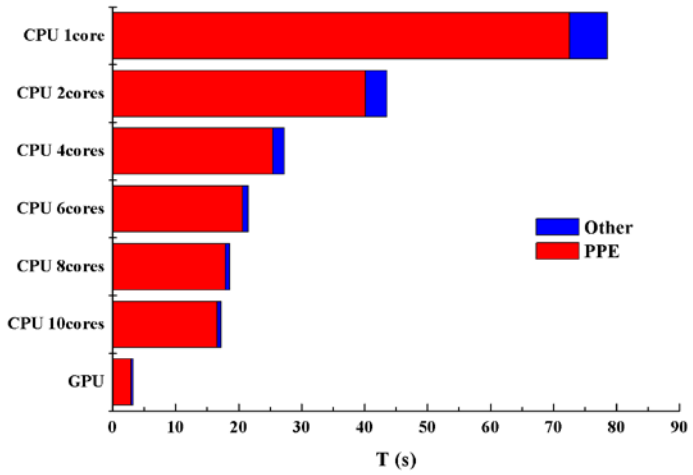


Figure 12. The computation times of CPU and GPU

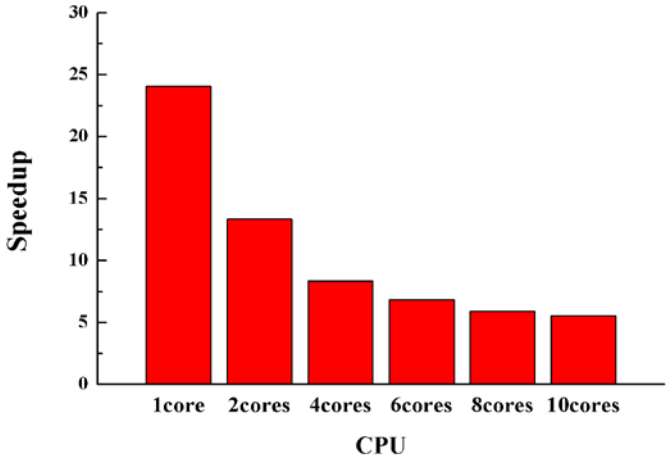


Figure 13. The speedup by GPU

Conclusions

In this paper, the MPS-GPU-SJTU solver based on modified MPS is developed to simulate 3-D sloshing problem by applying GPU acceleration technique. By reducing the dimensions of matric and arrays, the computation efficiency of solving PPE is improved. The convergence verification is conducted to prove the stability of GPU solver. In addition, the results of GPU show a good agreement with CPU. The computation time of GPU solver is much smaller than CPU and the speedup of every time iteration is up to 24.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (51379125, 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.

References

[1] Faltinsen, O. M. (1978) A Numerical Nonlinear Method of Sloshing in Tanks with Two Dimensional Flow, *Journal of Ship Research* **22** (3), 193-202.
[2] Nakayama, T., and Washizu, K. (1980) Nonlinear Analysis of Liquid Motion in a Container Subjected to Forced Pitching Oscillation, *International Journal for Numerical Methods in Engineering* **15** (8), 1207-1220.
[3] Arai, M., Cheng, L. Y., and Inoue, Y. (1992) 3D Numerical Simulation of Impact Load due to Liquid Cargo Sloshing, *Journal of the Society of Naval Architects of Japan* **171**, 177-184.
[4] Koh, H. M., Kim, J. K., and Park, J. H. (1998) Fluid-structure Interaction Analysis of 3-D Rectangular Tanks by a Variationally Coupled BEM-FEM and Comparison with Test Eesults, *Earthquake Engineering and Structural Dynamics* **27** (27), 109-124.
[5] Kim, Y. (2001) Numerical Simulation of Sloshing Flow with Impact Load, *Applied Ocean Research* **23**, 53-62.
[6] Liu, D. M., and Lin, P. Z. (2009) Three-dimensional Liquid Sloshing in a Tank with Baffles, *Ocean Engineering* **36**, 202-212.
[7] Yin C. H., Wu J. W., Wan D. C. (2014) *Numerical study on liquid sloshing in three-dimensional rectangular tanks with different filling rates and fixed baffle*, Proceedings of the 24th International Offshore and Polar Engineering Conference, Hawaii, United States: 381-391.
[8] Delorme, L., Colagrossi, A., Iglesias, A. S., Rodriguez, R. Z., and Vera, E. B. (2009) A Set of Canonical Problems in Sloshing, Part I: Pressure Field in Forced Roll-comparison between Experimental Results and SPH, *Ocean Engineering* **36** (2), 168-178.
[9] Shao, J. R., Li, H. Q., Liu, G. R., and Liu, M. B. (2012) An Improved SPH Method for Modeling Liquid Sloshing Dynamics, *Computers and Structures* **100-101**, 18-26.
[10] Koh, C. G., Luo, M., and Bai, W. (2013) Modelling of Liquid Sloshing with Constrained Floating Baffle, *Computers and Structures* **122**, 270-279.
[11] Kim, K. S., Kim M. H., and Park, J. C. (2014) Development of Moving Particle Simulation Method for Multiliquid-Layer Sloshing, *Mathematical Problems in Engineering* **137**(5), 282-290.
[12] 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*, Proceedings of the Twenty-fifth (2015) International Ocean and Polar Engineering Conference, Hawaii, USA, 937-944.
[13] Hashimoto, H., Hata, Y., and Kawamura, K. (2016) Estimation of oil overflow due to sloshing from oil storage tanks subjected to a possible Nankai Trough earthquake in Osaka bay area, *Journal of Loss Prevention in the Process Industries*, 1-10.
[14] Zhou, L., Cai, Z. W., Zong, Z., and Chen, Z. (2016) An SPH pressure correction algorithm for multiphase flows with large density ratio, *Int. J. Numer. Meth. Fluids* **81**, 765-788.
[15] Chen, X., Zhang, Y. L., and Wan, D. C. (2016) *Effects of the location of horizontal baffle on liquid sloshing by MPS method*, Proceedings of the Second Conference of Global Chinese Scholars on Hydrodynamics, Wuxi, China, 448-454
[16] Harada, T., Koshizuka, S., and Kawaguchi, Y. (2007) Smoothed particle hydrodynamics on GPUs, *Structure* **4** (4), 671-691.
[17] Crespo, A. J. C., Dominguez, J. M., Barreiro, A., Gómez-Gesteira, M. and Rogers, B. D. (2011) GPUs, a new tool of acceleration in CFD: efficiency and reliability on smoothed particle hydrodynamics methods, *PLoS One* **6** (6): e20685.
[18] 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, *Computer Physics Communications*, **184**(3), 617-627.
[19] Fourtakas, G., and Rogers, B. D. (2016) Modelling multi-phase liquid-sediment scour and resuspension induced by rapid flows using Smoothed Particle Hydrodynamics (SPH) accelerated with a Graphics Processing Unit (GPU), *Advances in Water Resources* **92**, 186-199.
[20] Hori, C., Gotoh, H., Ikari, H., and Khayyer, A. (2011) GPU-acceleration for moving particle semi-implicit method, *Computers & Fluids* **51**, 174-183.
[21] Kakuda, K., Nagashima, T., Hayashi, Y., Obara, S., Toyotani, J., Katsurada, N., Higuchi, S., and Matsuda, S. (2012) Particle-based fluid flow simulations on GPGPU using CUDA, *Computer Modeling in Engineering & Sciences* **88**(1), 17-28.
[22] Kakuda, K., Nagashima, T., Hayashi, Y., Obara, S., Toyotani, J., Miura, S., Katsurada, N., Higuchi, S., and Matsuda, S. (2013) Three dimensional fluid flow simulations using GPU-based particle method, *Computer Modeling in Engineering & Sciences* **93**(5), 363-376.
[23] Li, H. Z., Zhang, Y. L., and Wan, D. C. (2015) *GPU Based Acceleration of MPS for 3D Free Surface Flows*, Proceedings of the 9th International Workshop on Ship and Marine Hydrodynamics, Glasgow, UK.

[24] Gou, W., Zhang S., and Zheng Y. (2016) Simulation of isothermal multi-phase fuel-coolant interaction using MPS method with GPU acceleration, *Kerntechnik* **81**(3), 330-336.
[25] Koshizuka, S., and Oka, Y. (1996) Moving-particle semi-implicit method for fragmentation of incompressible fluid, *Nuclear Science and Engineering* **123**(3), 421-434.
[26] 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, *Computer Methods in Applied Mechanics and Engineering* **200**(9-12), 1113-1125.
[27] Zhang, Y. X., and Wan, D. C. (2012) Numerical simulation of liquid sloshing in low-filling tank by MPS, *Journal of Hydrodynamics* **27**(1), 101-107.
[28] CUDA Toolkit Documentation v8.0.61. <http://docs.nvidia.com/cuda/>.
[29] CULA (GPU-Accelerated LAPACK). <http://www.culatools.com/>.