Proceedings of the Thirty-fifth (2025) International Ocean and Polar Engineering Conference **www.isope.org**
Seoul/Goyang, Korea, June 1-6, 2025

ISBN 978-1-880653-74-6; ISSN 1098-6189

Numerical Simulation of Large-Scale Multiphase Flows Based on MPS and DCU

*Fengze Xie[1], Guohua Pan[2], Yuan Zhuang[1], Decheng Wan[1]**

[1] Computational Marine Hydrodynamics Lab (CMHL), School of Naval Architecture, Ocean and Civil Engineering, Shanghai Jiao Tong University, Shanghai, China
[2] Ningbo Pilot Station, Ningbo Dagang Pilotage Co., Ltd., Ningbo, China

*Corresponding author

ABSTRACT

In this work, the Moving Particle Semi-Implicit method (MPS) is combined with a GPU-like processor, the Deep Computing Unit (DCU). An efficient numerical solver for multiphase flow problems characterized by strong nonlinearity and significant interface deformation is developed. Our in-house GPU-accelerated MPS algorithm is transported to the HIP programming environment-based DCU platform and the MPSDCU-SJTU solver is developed. In the simulation of 3D bubbly flow involving a set of bubbles, the numerical results of DCU solver are in good agreement with those of the mature GPU solver, validating the accuracy of the DCU solver. Performance testing indicates that the solver's computational efficiency is significantly enhanced and its potential application in large-scale multiphase flow problems is expanded.

KEY WORDS: Moving Particle Semi-Implicit method; Deep Computing Unit; multiphase flows; Large scale simulation.

INTRODUCTION

MPS, with its Lagrangian description, has advantages in simulating complex flow problems, especially in high-precision simulation of multiphase flows. However, the high computational cost associated with neighbor particle search and solution of the pressure Poisson equation limits its practicality in large-scale applications.

In recent years, several multi-resolution techniques have been proposed to optimize the MPS algorithm, improving computational efficiency and reducing computing time. Particles with higher resolution are arranged near the key areas of concern, such as near structures (Zhong et al., 2023; Huang et al., 2022; Zhang et al., 2022), bubbles (Jiang et al., 2024; Yang et al., 2023) and free surface (Chen et al., 2023). While in the parts far away from the key areas of concern, particles with lower resolution are used.

With the development of computer hardware, the simulation speed of the particle method has been greatly improved. Parallel acceleration technique with CPUs is introduced to the particle-based methods. A particle method task is decomposed into multiple subtasks and assigned to multiple CPUs for simultaneous processing. Parallel particle-based methods have been used to investigate several problems (Xie et al., 2021), such as wave breaking (Marrone et al., 2012), liquid sloshing (Jiao et al., 2024), liquid-solid two-phase flows (Xie et al., 2025) and wave–body interactions (Xie et al., 2023).

The GPU was originally designed to accelerate the rendering process of computer graphics. Thanks to its large-scale data processing capabilities, it has been widely used to accelerate numerical simulations in recent years. Researchers have introduced the GPU technique to the mesh-based methods, such as finite volume method (Ma et al., 2024), finite difference method (Pekkila et al., 2017), etc. With the acceleration of GPU, particle-based methods have also been applied to solve large-scale problems. Zhao et al. (2023) studied the high-speed water entry of objects using a GPU-accelerated SPH (Smoothed Particle Hydrodynamics) solver. Zhao et al. (2023) investigated landslide-induced surge waves using the GPU-accelerated SPH-DEM coupling method. Lyu et al. (2023) employed the SPH-GPU solver to simulate water entry process of a vessel section, waves impact offshore structures, vessels advance in calm water, etc. Due to the solution of pressure Poisson equation, the MPS (Moving Particle Semi-implicit) method may require greater computational cost. Therefore, it is of greater importance to introduce GPU acceleration technique into the MPS method. Xie et al. (2020) studied the sloshing in a square liquid tank based on the GPU-accelerated MPS method. Zhang et al. (2022) simulated atomization by jet impact using MPS with GPU acceleration. Zhan et al. (2025) developed a enhanced SPH solver - DualSPHysics+ with GPU acceleration technique. However, as the artificial intelligence arms race unfolds, it's difficult to obtain high - end GPU from NVIDIA.

The DCU (Deep Computing Unit), a newly developed domestic accelerator in China, features high acceleration performance and demonstrates strong adaptability in migrating CFD-GPU programs (Hua et al., 2023; Shang et al., 2023).

In this work, the in-house GPU-accelerated multi-phase flow MPS

algorithm program written in CUDA language is migrated to the DCU based on the HIP programming environment. In the first section, the multi-phase MPS method and porting scheme for moving GPU - based algorithms to DCU platforms are presented briefly. In the second section, the case of 3D bubbly flow involving a set of bubbles is simulated and the results of DCU are compared with those of GPU. Finally, conclusions and future work are presented.

NUMERICAL METHOD

Governing Equations

The fluid governing equations, including the continuity equation and momentum equation, can be written as follows:

$$\frac{D\rho}{Dt} + \rho\nabla\cdot\boldsymbol{u} = 0 \quad (1)$$

$$\rho\frac{D\boldsymbol{u}}{Dt} = -\nabla p + \boldsymbol{F}^V + \boldsymbol{F}^B + \boldsymbol{F}^S \quad (2)$$

where $\boldsymbol{u}$ is fluid velocity vector, ρ is fluid density, t is physical time, ∇ represents the Hamiltonian operator, p represents fluid pressure, $\boldsymbol{F}^V$, , $\boldsymbol{F}^B$ and $\boldsymbol{F}^S$ denote the viscous force, body force, and surface tension force, respectively.

In the multi-density model, an improved density smoothing scheme (Wen et al. 2021) is ultised, which realizes the smooth transition of density field by performing the spatial weighted averaging of density for particles near the phase interface. With the improved scheme, the smoothed density field can be obtained by:

$$<\rho>_i = \frac{\rho_i W_{\text{self}} + \sum_{j\in I}\rho_j W(|\boldsymbol{r}_j - \boldsymbol{r}_i|)}{W_{\text{self}} + \sum_{j\in I} W(|\boldsymbol{r}_j - \boldsymbol{r}_i|)} \quad (3)$$

where I includes the target particle i and all its neighboring particles. W_{self} is a weight function to amplify the effect of the target particle itself on the smoothed density, through which the sharpness of density variation across the phase interface can be better preserved.

The multi-viscosity model (Shakibaeinia and Jin, 2012) is used to calculate the viscous force, given by,

$$\boldsymbol{F}^V = \mu\nabla^2\boldsymbol{u} = \frac{2d}{n^0\lambda}\sum_{j\neq i}\frac{2\mu_i\mu_j}{\mu_i+\mu_j}(\boldsymbol{u}_j - \boldsymbol{u}_i)\cdot W(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \quad (4)$$

Particle interaction models

The interaction between MPS particles is controlled by the Kernel Function (KF). A KF proposed by Zhang et al. (2014) is adopted in this paper, which is written as follows:

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

where r represents the distance between MPS neighbor particles, r_e represents the effective radius.

The inter-particle interaction models including the gradient model, divergence model, and Laplacian model are written as,

$$\langle\nabla\phi\rangle_i = \frac{d}{n^0}\sum_{j\neq i}\frac{\phi_j+\phi_i}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}(\boldsymbol{r}_j - \boldsymbol{r}_i)w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \quad (5)$$

$$\langle\nabla\cdot\boldsymbol{\Phi}\rangle_i = \frac{d}{n^0}\sum_{j\neq i}\frac{(\boldsymbol{\Phi}_j - \boldsymbol{\Phi}_i)\cdot(\boldsymbol{r}_j - \boldsymbol{r}_i)}{|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \quad (6)$$

$$\langle\nabla^2\phi\rangle_i = \frac{2d}{n^0\lambda}\sum_{j\neq i}(\phi_j - \phi_i)\,w(|\boldsymbol{r}_j - \boldsymbol{r}_i|) \quad (7)$$

where subscripts i and j represent the number of fluid particles, $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ are the position vectors of fluid particles i and j, ϕ is the physical scalar carried by the MPS particles, $\boldsymbol{\Phi}$ represents the physical vector carried by the MPS particles, d is the spatial dimension of the computational domain, n^0 is the particle number density under the initial distribution, λ is a correction parameter, which is a compensation for the error caused by using a finite range kernel function to approximate an infinite range Gaussian function in the derivation process of the Laplace model, written as,

$$\lambda = \frac{\sum_{j\neq i} w(|\boldsymbol{r}_j - \boldsymbol{r}_i|)|\boldsymbol{r}_j - \boldsymbol{r}_i|^2}{\sum_{j\neq i} w(|\boldsymbol{r}_j - \boldsymbol{r}_i|)} \quad (8)$$

Model of incompressibility

For the incompressible fluid, pressure information is obtained by solving the Pressure Poisson Equation (PPE). In order to balance between stability and accuracy, a mixed source method (Tanaka et al., 2010; Khayyer and Gotoh, 2011) is adopted, defined by,

$$\langle\nabla^2 p^{m+1}\rangle_i = (1-\gamma)\frac{\rho}{\Delta t}\nabla\cdot\boldsymbol{u}_i^* - \gamma\frac{\rho}{\Delta t^2}\frac{\langle n^m\rangle_i - n^0}{n^0} \quad (9)$$

For the compressible fluid, an incompressible-compressible model (Khayyer and Gotoh, 2016; Duan el al. 2017) is employed, given by,

$$\langle\nabla^2 p^{m+1}\rangle_i = (1-\gamma)\frac{\rho}{\Delta t}\nabla\cdot\boldsymbol{u}_i^* - \gamma\frac{\rho}{\Delta t^2}\frac{\langle n^m\rangle_i - n^0}{n^0} + \frac{1}{\Delta t^2 C_s^2}P_i^{m+1} \quad (10)$$

where C_S is the speed of sound.

Surface tension model

The deformation of bubbles is significantly influenced by surface tension force. the continuum surface force (CSF) model proposed by Brackbill et al. (1992) is employed in this work, given by,

$$\boldsymbol{F}^S = -\sigma\kappa\nabla C \quad (11)$$

where σ is the surface tension coefficient, κ is the interface curvature, ∇C is the gradient of color function.

The density-weighted color function (Zhang et al., 2015) is employed to keep the continuity of surface tension force, which is defined as,

$$C_{ij} = \begin{cases} 0 & \text{if particle } i \text{ and } j \text{ belong to the same phase} \\ \dfrac{2\rho_i}{\rho_i+\rho_j} & \text{if particle } i \text{ and } j \text{ belong to different phase} \end{cases} \quad (12)$$

Regarding the calculation of the interface curvature κ, the analytical

method put forward in the contoured continuum surface force (CCSF) model, as proposed by Duan et al. (2015) is utilized. In the first step, the smoothed color function *f* at an arbitrary location (x, y) can be obtained by performing a spatial weighted averaging of the above color function, given by,

$$f(x,y)=\frac{\sum_j C_j G(r_{ij},r_s)}{\sum_j G(r_{ij},r_s)}, G(r_{ij},r_s)=\frac{9}{\pi r_s^2}\exp\left(-\frac{9{r_{ij}}^2}{{r_s}^2}\right) \quad (13)$$

the interface curvature at particle i can be analytically calculated as

$$\kappa_i=\frac{f_{xx,i}\left(f_{y,i}^2+f_{z,i}^2\right)+f_{yy,i}\left(f_{x,i}^2+f_{z,i}^2\right)+f_{zz,i}\left(f_{x,i}^2+f_{y,i}^2\right)}{\left(f_{x,i}^2+f_{y,i}^2+f_{z,i}^2\right)^{3/2}} -\frac{2\left(f_{x,i}f_{y,i}f_{xy,i}+f_{x,i}f_{z,i}f_{xz,i}+f_{y,i}f_{z,i}f_{yz,i}\right)}{\left(f_{x,i}^2+f_{y,i}^2+f_{z,i}^2\right)^{3/2}} \quad (14)$$

Porting strategies for GPU programs to DCU

The compilation environment HIP C++ of DCU is quite similar to CUDA C/C++ of GPUs in terms of the overall programming model, thread structure, and memory structure. Meanwhile, the HIP runtime implements most of the APIs that correspond one - to - one with CUDA, which facilitates the porting of CUDA applications to HIP.

The kernel function syntax of HIP C++ is exactly the same as that of CUDA C++. That is, the source code of kernel functions in CUDA programming can be directly used for compilation on the HIP platform. In addition, HIP and CUDA have the same kernel function invocation syntax. HIP C++ and CUDA C++ have similar runtime API call interfaces. Currently, HIP also includes most of the runtime API function calls that are compatible with CUDA. During the transcoding and porting process, the corresponding CUDA API function call names only need to be replaced with HIP ones.

The specific program flow of the MMPSDCU - SJTU solver is shown in Figure 1. According to the functional characteristics of the execution instructions, the solver's code is divided into CPU code and DCU code. The CPU code is mainly used for inputting and outputting data and invoking DCU settings for accelerated computing, while the DCU code is mainly used for accelerating and optimizing computationally intensive modules.

In the original MPS solver developed based on GPU accelerators, NVIDIA's cuBLAS and cuSparse libraries were used to perform matrix operations, and the Preconditioned Biconjugate Gradient Stabilized Method (PBICG) was used to solve linear equations. These two libraries provide highly optimized functions, which greatly improve the efficiency of matrix operations and the solving process. In this paper, to achieve cross - platform porting of the code from GPUs to DCUs, the CUDA - based libraries are replaced with ROCm - based libraries, namely hipBLAS/rocBLAS and rocSPARSE, and they are applied to perform low - level operations on matrices. The Thrust library in the CUDA library for solving sparse matrix systems is also replaced with rocThrust in the ROCm library. Thus, the migration of GPU code to DCUs can be realized.

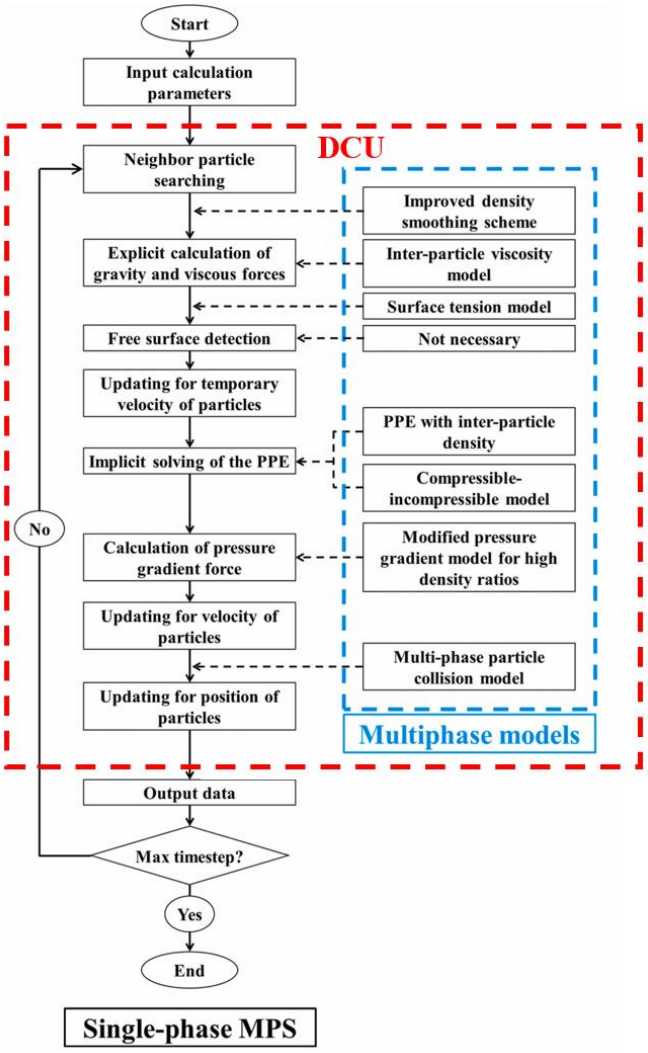


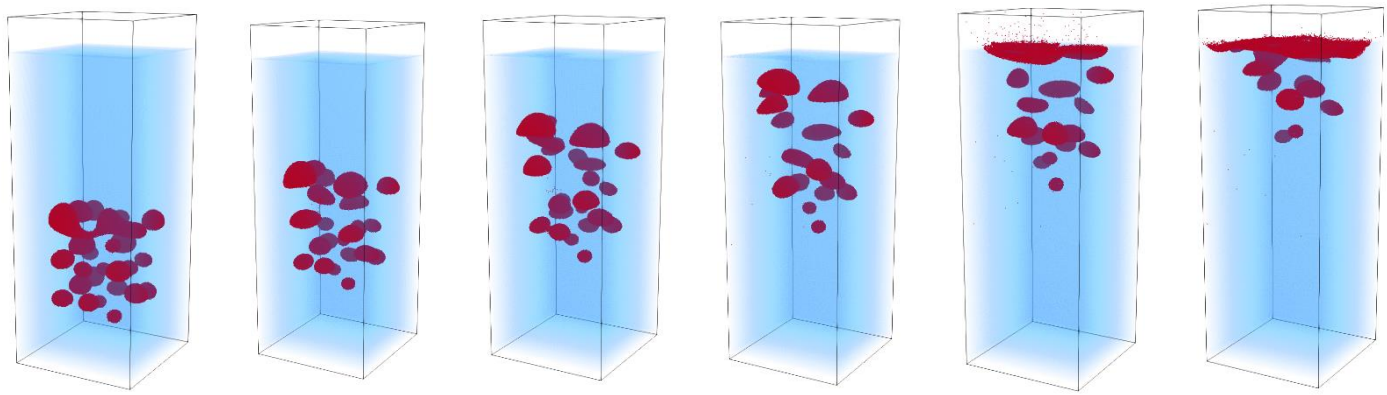
Fig.1 Flow chart of MMPSDCU-SJTU solver

NUMERICAL RESULTS

To verify the computational accuracy and efficiency of the MPSDCU - SJTU solver, a 3-D bubble rising process is simulated.

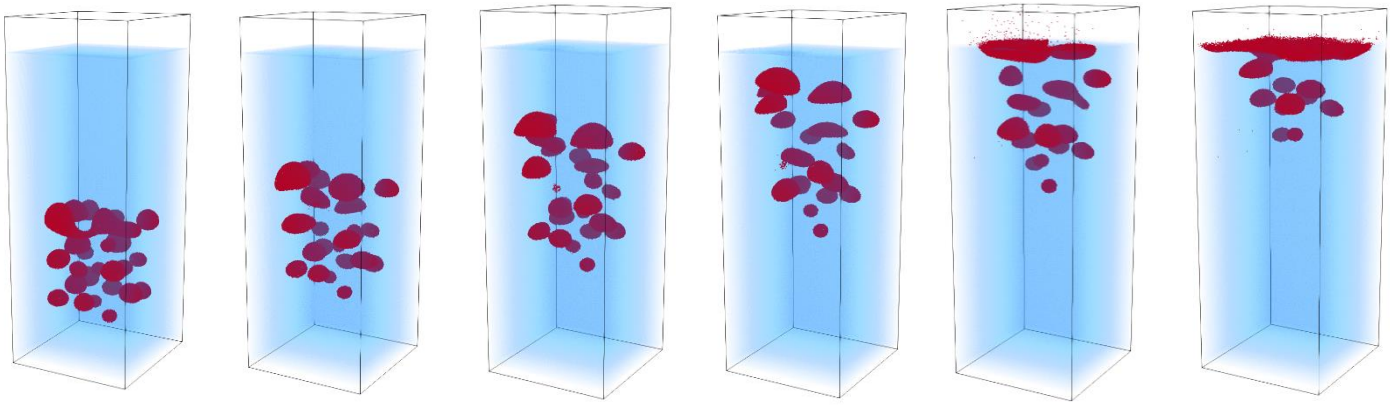
Numerical model

Fig.2 shows the Initial configuration of bubbles. The fluid domain is set to be $19.2R \times 19.2R \times 45R$, where R = 1.5 mm, and a total of 27 bubbles are initially generated on a 3 × 3 × 3 lattice, thus a total of more than two million MPS particles are used in the 3D simulations. In order to save the computation cost, there is no lighter fluid arranged above the heavier fluid in the 3D model. The densities of liquid and bubble phases are 1000 kg/m^3 and 100 kg/m^3, thus the density ratio is equal to 10 in this case. The viscosities of liquid and bubble phases are 0.156 kg/(m.s) and 0.078 kg/(m.s), respectively. The Reynolds number is 40 and Bond number is 1.25, respectively.



(a1) t = 0.1 s (a2) t = 0.2 s (a3) t = 0.3 s (a4) t = 0.4 s (a5) t = 0.5 s (a6) t = 0.6 s

(a) DCU



(b1) t = 0.1 s (b2) t = 0.2 s (b3) t = 0.3 s (b4) t = 0.4 s (b5) t = 0.5 s (b6) t = 0.6 s

(b) GPU (NVIDA A100)

Fig. 3 Comparison of numerical simulation results between DCU and GPU

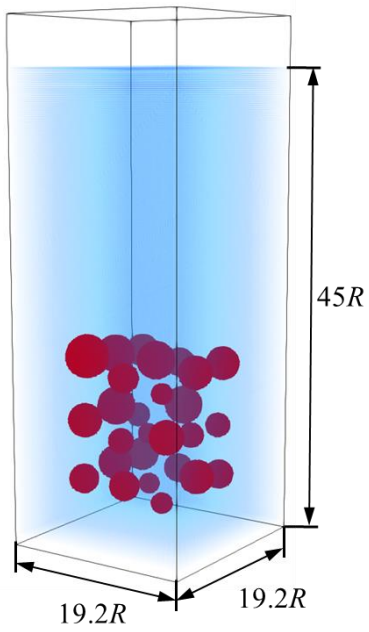


Fig.2 Initial configuration of bubbles

Model verification

Fig. 3 shows a comparison of the numerical results between the NVIDIA A100 GPU and the DCU. It is worth noting that when t < 0.3 s, the morphological features and position of each bubble simulated by the DCU solver are consistent with the numerical results simulated by the GPU. When t > 0.3 s, the overall distribution characteristics of the bubble flow simulated by DCU solver are in good agreement with the numerical results simulated by GPU solver. However, there are slight differences in the morphological features and positions of some individual peculiar bubbles simulated by the DCU compared with the numerical results of the GPU simulation. This might be due to differences in the number of decimal places retained by the different solvers.

In general, both solvers are capable of capturing the deformation, upward floating, coalescence, and rupture of bubbles. The accuracy of the DCU multiphase flow solver has been proven

Performance Test

Fig. 4 presents the comparison of computational efficiency between DCU and GPU. Besides, the computational time of the DCU and the GPU is listed in Tab. 5. In the PPE part, the simulation time of the DCU is approximately 2.24 times that of the GPU, and the overall computational time is about 2.05 times that of the GPU simulation time. This indicates that in the process of solving multiphase - flow problems based on the MPS method, the performance of the DCU can almost reach

50% of that of the NVIDIA A100 GPU.

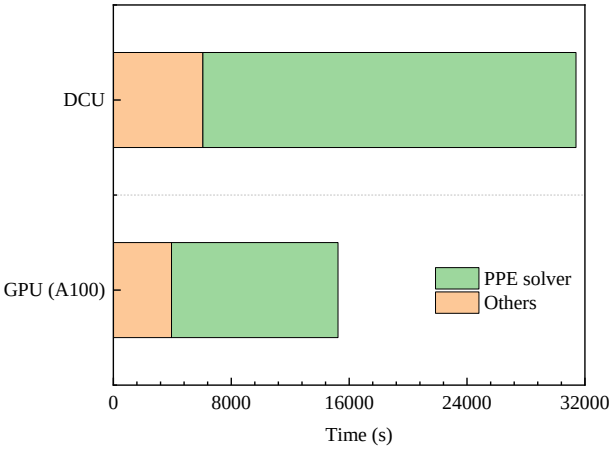


Fig. 4 Comparison of computational efficiency between DCU and GPU

Tab. 1 Computational time of DCU and GPU for different parts

Module	DCU	GPU NVIDA A100
PPE solver	25319.46 s	11294.41 s
Others	6077.80 s	3948.51 s
Total	31397.26 s	15242.92 s

CONCLUSIONS

In this work, the MPS is combined with DCU acceleration technology. A multiphase flow solver MMPSDCU-SJTU is developed using the HIP programming model. The problem of multiple bubbles rising is simulated.

It can be concluded that the simulation results obtained by MMPSDCU - SJTU are in good agreement with the numerical results of the GPU solver, which verifies the accuracy of the solver. In addition, the numerical simulation time of the DCU is approximately 2.05 times that of the GPU, indicating that the computational efficiency of the DCU solver is approximately 50% of that of the GPU solver.

Generally, in numerical simulations, DCU can serve as a partial substitute for GPU.

ACKNOWLEDGEMENTS

This work was supported by the GHfund A（202407018053）and the National Natural Science Foundation of China (52401334, 52131102), to which the authors are most grateful.

REFERENCES

Brackbill, JU, Kothe, DB, Zemach, C (1992). “A continuum method for modeling surface tension,” *J Comput Phys* 100, 335–354.

Bouscasse, B, Marrone, S, Colagrossi, A, and Antuono, M (2012). "Study of ship wave breaking patterns using 3D parallel SPH simulations." *Comput Fluids*, 69, 54 - 66.

Chen, D., Huang, W, Huang, D, and Liang, C (2023). "An adaptive multi - resolution SPH approach for three - dimensional free - surface flow with fluid impacting," *Eng Anal Bound Elem*, 155, 642 - 651.

Duan, G, Koshizuka, S, Chen, B, Xiang, H (2017). “Stable multiphase moving particle semi - implicit method for incompressible interfacial flow,” *Comput Methods Appl Mech Eng* 318, 636–666

Hua, HB, Song, ZL, Xiong, W, Shang, JD, Zhang, LT, and Han, L (2023). "Parallel Immersed Boundary Method for Two - Phase Flows on DCU Clusters." *Int J Comput Fluid Dyn*, 37, 711 - 728.

Huang, XT, Sun, PN, Lyu, HG, & Zhong, SY (2022) "Study of 3D self - propulsive fish swimming using the δ^+ - SPH model," *Acta Mechanica Sinica*, 39, 722053.

Jiang, T, Liu, YH, Meng, ZF, Sun, PN, Wei, XY, and Wang, DS (2024). "Embedment of WENO - Z Reconstruction in Lagrangian WLS Scheme Implemented on GPU for Strongly - Compressible Multi - Phase Flows," *Comput Methods Appl Mech Eng*, 430, 117209.

Jiao, J., Zhao, M., Jia, G. and Ding, S (2024). SPH simulation of two side - by - side LNG ships’ motions coupled with tank sloshing in regular waves. *Ocean Eng*, 297, 117022.

Khayyer, A, and Gotoh, H (2011). “Enhancement of stability and accuracy of the moving particle semi - implicit method,” *J Comput Phy*, 230(8): 3093 - 3118.

Khayyer, A, and Gotoh, H (2016). “A multiphase compressible - incompressible particle method for water slamming,” *int j offshore polar eng*, 26(01), 20 - 25.

Lyu, HG, Sun, PN, Huang, XT, Peng, YX, Liu, NN, Zhang, X, Xu, Y, and Zhang, AM (2023). "SPHHydro: Promoting smoothed particle hydrodynamics method toward extensive applications in ocean engineering." *Phys Fluids*, 35, 017116.

Ma, YH, Xiao, XY, Li, W, Desbrun, M, and Liu, XP (2024). "Hybrid LBM - FVM solver for two - phase flow simulation." *J Comput Phys*, 506, 112920.

Marrone, S, Bouscasse, B, Colagrossi, A, and Antuono, M (2012). "Study of ship wave breaking patterns using 3D parallel SPH simulations." *Comput Fluids*, 69, 54 - 66.

Pekkila, J. Vaisala, MS, Kapyla, MJ, Kapyla, PJ, and Anjum, O (2017). "Methods for compressible fluid simulation on GPUs using high - order finite differences." *Comp Phys Commun*, 217, 11 - 22.

Shakibaeinia, A, and Jin, Y. (2012). MPS mesh - free particle method for multiphase flow. Comput Methods Appl Mech Eng, 229–232, 13–26.

Shang, JD, Xiong, W, Hua, HB, Song, ZL, Guo, HL, and Zhang, J. (2023). "Heterogeneous Implementation of Fluid - Structure Interaction Immersed Boundary Method for DCU." *Comput Eng*.

Tanaka, M, and Masunaga, T (2020). “Stabilization and smoothing of pressure in MPS method by Quasi - Compressibility,” *J Comput Phy*, 229(11): 4279 - 4290.

Wen, X, Zhao, WW, Wan, DC (2021). A multiphase MPS method for bubbly flows with complex interfaces. Ocean Eng, 238, 109743.

Xie, CM, Yang, JC, Sun, PN, Lyu, HG, Yu, J, and Ye, YL (2023). "An accurate and efficient HOS - meshfree CFD coupling method for simulating strong nonlinear wave–body interactions." *Ocean Eng*, 287(Part 2), 115889.

Xie, FZ, Zhao, WW, and Wan, DC (2021). "MPS - DEM coupling method for interaction between fluid and thin elastic structures," *Ocean Eng*, 236, 109449.

Xie, FZ, Zhao, WW, and Wan, DC (2020). "CFD simulations of three - dimensional violent sloshing flows in tanks based on MPS and GPU." *J Hydrodyn*, 32, 672 - 683.

Xie, FZ, Pan, GH, Zhao, WW, and Wan, DC (2025). "Unresolved MPS - DEM coupling method for three - dimensional liquid - solid dam - break flows impacting on rigid structures." *Ocean Eng*, 323, 120601.

Yang, Q, Xu, F, Yang, Y, Dai, Z, and Wang, J (2023). "A GPU - accelerated adaptive particle refinement for multi - phase flow and fluid - structure coupling SPH," *Ocean Eng*, 279, 114514.

Zhan, Y, Luo, M, and Khayyer, A (2025). “DualSPHysics++: An enhanced DualSPHysics with improvements in accuracy, energy conservation and resolution of the continuity equation.” *Comput Phys Commun*, 306, 109389.

Zhang, K, Sun, YJ, Sun, ZG, Wang, F, Chen, X, and Xi, G (2022). "An efficient MPS refined technique with adaptive variable - size particles, " *Eng Anal Bound Elem*, 143, 663 - 676.

Zhang, S, Gou, W, Wang, Y, Zhang, J, and Zheng, Y (2022). "Direct numerical simulation of atomization by jet impact using moving particle semi - implicit method with GPU acceleration." Comput Particle Mech, 9, 499 - 512.

Zhang, Y, Hou, SH, Di, SJ, Liu, ZB, and Xu, YF (2023). "DEM - SPH Coupling Method for Landslide Surge Based on a GPU Parallel Acceleration Technique." *Comput Geotech*, 164, 105821.

Zhang, Y, Wan, D, and Hino, T (2014). “Comparative study of MPS method and level - set method for sloshing flows,” *J Hydrodyn*, 26(4): 577 - 585.

Zhao, ZX, Bilotta, G, Yuan, QE, Gong, ZX, Liu, H (2023). "Multi - GPU multi - resolution SPH framework towards massive hydrodynamics simulations and its applications in high - speed water entry." *J Comput Phys*, 490, 112339.

Zhong, SY, Sun PN, Peng YX, Liu NN, Lyu HG, and Huang XT (2023) "An SPH study of slamming and splashing at the bow of SYSU vessel," *Ocean Eng*, 269, 113581.