GPU Parallelization of solving pressure Poisson Equation in MPS Method

†Zhe Sun[1,3], *Zi-kai Xu[1], Xi Zhang[2], Bi-ye Yang[1],†Gui-yong Zhang[1,3,4], Zhi-fan Zhang[1]

[1]School of Naval Architecture and Ocean Engineering, Dalian University of Technology, Dalian, 116024, P. R. China
[2]National Supercomputing Center , Guangzhou, 511400, P.R.China
[3]State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian, 116024, PR China
[4]Collaborative Innovation Center for Advanced Ship and Deep-Sea Exploration, Shanghai, 200240, P.R. China

*Presenting author: a417851468@dlut.edu.cn
†Corresponding author: zsun@dlut.edu.cn gyzhang@dlut.edu.cn

Abstract

In this paper, an improved form of explicit solution of pressure Poisson equation is introduced. On the basis of the existing program, the numerical simulation of 2D dam-break problem is carried out. The numerical results agree well with experiments and GMRES method,computing efficiency is greatly improved; At the same time, the MPS method is combined with the GPU parallel acceleration technique. Based on the CUDA programming language, the GPU is parallelized to solve the pressure Poisson equation. Compared with the CPU solver, the developed program greatly reduces the solution time of the pressure Poisson equation and improves the calculation efficiency. The maximum acceleration ratios of 11.486 can be obtained by numerical simulation for 2-D dam-break problems with different particle numbers. And the developed program has better reliability and good adaptability.

Keywords: Meshless particle method, MPS method, CUDA, parallel computing

1. Introduction

The meshless particle method has been developed rapidly in recent years. It has shown great advantages in dealing with large deformation of free surface and phase interface or strong nonlinear deformation and large motion of boundary. The basic idea of the method is to disperse the computational domain into a series of fluid particles with flow field information, and the transfer of flow field information is expressed by the interaction of particles. Smoothed Particle Hydrodynamics (SPH) and Moving Particle Semi-implicit (MPS) are two typical meshless particle class methods, of which the MPS method has attracted much attention as a new computational method. In recent years, some scholars have used MPS method to simulate problems such as dam break, slogging, fluid-structure coupling, and floating body motion on waves, and obtained good results, indicating that MPS method has great applicability.

However, there are still some problems in the MPS method, one of which is the computational efficiency. Because the MPS method is based on the incompressible condition, the pressure is obtained by solving the pressure Poisson equation, which needs to solve a sparse matrix. The

dimension of the matrix is positively correlated with the particle numbers. Therefore, how to improve computational efficiency has been a problem worthy of attention.

In recent years, the appearance of a parallel technology called GPU (Graphic Processing Unit) has aroused many people's attention. With the rapid development of GPU hardware and the development of related programming technology, GPU begins to play an important role in some general computing fields because of its powerful floating point computing ability and high efficiency. Because of the difference between GPU and CPU in design goal, there is a great difference between them in logic structure of hardware. As shown in figure 1 [1], GPUs are divided into more execution units (ALUs) and have more memory bandwidth than CPUs, which have more control and cache units. This difference, fundamentally, determines that CPU has more complex computing power, while GPU has more floating point computing power and parallel computing power. This natural multi-core architecture mode makes GPU suitable for large-scale parallel scientific computing.

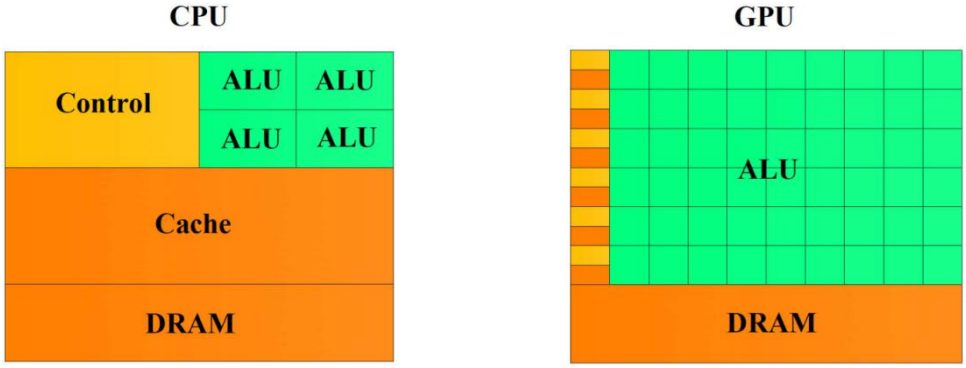


Figure 1. Different hardware architectures for GPU and CPU

The numerical model of meshless particle method is relatively simple. Except for Poisson equation, the calculation of each particle is relatively independent, and it is easy to combine the whole method with GPU. The application of GPU parallel acceleration technology in SPH method has been relatively mature. Harada et al. [2] and Zhang et al. [3] first applied GPU technology to meshless particle method SPH. Then the application of GPU in SPH gradually increased. Crespo et al. [4] reviewed the theory of SPH and the different processing methods of some key problems, and compared the similarities and differences of SPH method on CPU and GPU. Herault et al. [5] used CUDA library to accelerate the SPH method, and compared the acceleration ratio ratios of neighbor particle search, integral calculation and particle movement through the dam-break problem in detail, which showed that the GPU parallel acceleration ratio technology has great potential in the SPH method. Wei et al. [6]used the GPUSPH program to simulate the impact of a tsunami on a vertical column. The effects of pier shape and upstream angle on free surface evolution and hydrodynamic load were studied. The DualSP Hysics solver using ISPH method, such as Chow [7], has carried out the research of 3D focusing wave slamming on vertical cylinder on a single GPU. However, compared with the explicit calculation process of SPH, the MPS method adopts semi-implicit calculation process, which brings difficulties to the combination of MPS method and GPU acceleration technology, so the related research work much less than SPH method. Hori et al.[8]developed the MPS program of GPU parallel acceleration by CUDA language. The

pressure Poisson equation and pressure gradient are solved by GPU parallel acceleration technique. Kakuda et al.[9]applied the MPS method based on GPU acceleration to the 3D simulation problem, and the acceleration ratio obtained by the improved MPS method was 17.33 when the number of particles was 200,000. Gou et al.[10]implemented the parallel optimization of MPS method on GPU, simulated the interaction of isothermal and multiphase fuel coolant, and the numerical results were in agreement with the experimental results, and achieved a higher acceleration ratio. Vieira-e-Silva et al.[11] used the improved MPS method to simulate the dam-break flow on GPU, but the number of particles in the numerical model was less. Taniguchi et al.[12] have developed a WCMPS algorithm program for multiple GPUs, and tested the reliability of the program through a standard 3D dam-break example. Kawamura et al.[13] and Hashimoto et al.[14] applied the GPU parallel technology to the WCMPS method, developed the GPGPU program, and simulated the 3D sloshing of oil storage tanks under earthquake excitation with 6 million particles.

In this paper, an improved explicit solution form of pressure Poisson equation is presented and compared with the implicit solution form. Secondly, the computational efficiency of pressure Poisson equation is discussed by developing GPU program and CPU solver. The second part introduces the basic theory of MPS and the improved method of explicit solution of Poisson equation, and compares it with the result of implicit solution of Poisson equation, which verifies its feasibility. In the third part, based on the CUDA programming language and the existing 2-D dam-break program, a meshless particle method program is developed to solve the pressure Poisson equation on the GPU equipment,and the computational efficiency of solving the pressure Poisson equation is compared. The reliability and adaptability of the program are verified by numerical simulation of 2-D dam break problems with different particle numbers, and the acceleration ratio under different particle numbers is discussed. The fourth part is the conclusion.

2. Numerical method

2.1 Basics of MPS method

The MPS method is a meshless method based on the Lagrange method, and the computational domain is represented by discrete particles. The particles are not connected by grids or nodes, but each carries its own physical information, such as mass, velocity, and acceleration. The numerical model of MPS method used in this paper is introduced below.

2.1.1 Governing equation

For incompressible fluids, the MPS method uses the Navier-Stokes equation and the continuity equation as the governing equations. The forms are as follows:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1}$$

$$\rho \frac{D\boldsymbol{u}}{Dt} = -\nabla P + \mu \nabla^2 \boldsymbol{u} + \rho g \tag{2}$$

D/Dt denotes the derivative of matter, t is time, ρ is fluid density, P is pressure, μ is dynamic viscosity, V is velocity vector, and g is gravitational acceleration vector.

2.1.2 Particle interaction model

In the MPS method, the computational domain is composed of a series of discrete particles whose interactions are realized by kernel functions. The kernel functions used in this article are as follows:

$$w(r_{ij}) = \begin{cases} \frac{r_e}{r_{ij}} - 1, 0 < r_{ij} \le r_e \\ \\ 0, r_{ij} > r_e \end{cases} \quad (3)$$

In the formula, $r_{ij} = | r_i - r_j |$ is the distance between particles I and J, r_e is the radius of the particle's support domain. r_{e_lap} and r_{e_grad} adopted in this paper are 4.0 l_0 and 2.1 l_0 respectively[15], where is the initial particle spacing of l_0.

2.1.3 Boundary conditions

In the MPS method, when solving the pressure Poisson equation, we usually assign the pressure of the free surface particle and the second kind of boundary particle to 0 as the boundary condition.

2.1.3.1 Pressure Neumann condition on solid boundaries

In this study, the innermost solid boundary is treated by Newman boundary condition, and the pressure gradient between the current boundary particle and the closest fluid particle is calculated to avoid the deficiency of the particle in the support region. The formula is as follows:

$$\mathbf{n} \cdot \nabla p^{(k+1)} = \rho_0 (\mathbf{n} \cdot \mathbf{g} - \mathbf{n} \cdot \boldsymbol{u}_b^{(k+1)}) \quad (4)$$

Where $\boldsymbol{u}_b$ is the acceleration of the boundary and $\mathbf{n}$ is the normal vector of the boundary.

2.1.3.2 Laplacian operator compensation near solid boundary

For fluid particles close to the solid boundary, Laplace operator needs to be modified to meet the Neumann condition on the solid boundary and make up for the shortcomings of the adjacent particles.

The formula is as follows:

$$p_v = p_s + \rho_0(\mathbf{n}\cdot\mathbf{g} - \mathbf{n}\cdot\boldsymbol{u}_b^{(k+1)})r_0 \tag{5}$$

As shown in Figure 2[16], p_v is the pressure of the virtual particle and p_s is the pressure of the corresponding solid particle.

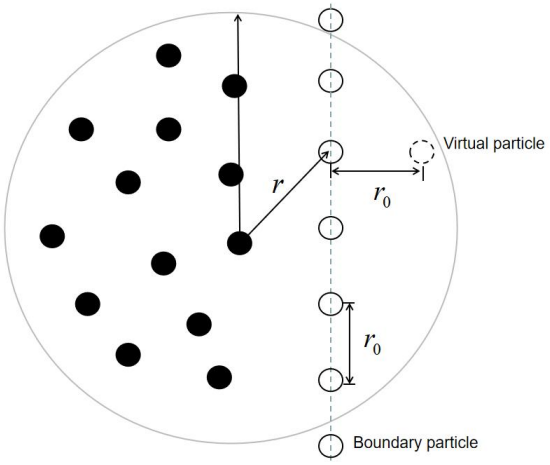


Fig 2. virtual particle for compensating the Lasplace operator near solid boundary

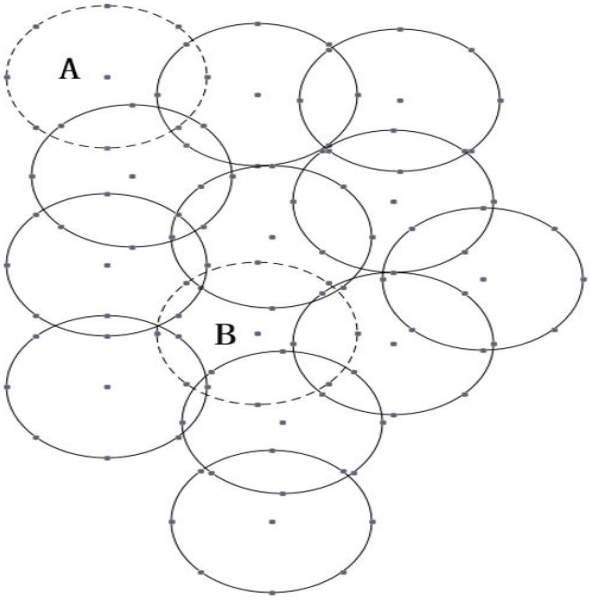


Fig 3. Demonstration of free surface particle identification

2.1.3.3 Free surface particle identification

A simplified version of the method used by Koh et al. [17]. Each self-centered particle is assigned a circle with a radius of $1.05r_0$, and the circle is discretized to 360 points and placed

evenly along the circle. If these points are completely covered by the circle of its neighbor particle, then it is considered an internal fluid particle, otherwise it is a free surface particle.

As shown in Figure 3, particle A is identified as a free-surface particle because it is not covered by a neighbor particle; accordingly, particle B is identified as an internal fluid particle because it is covered by a neighbor particle.

2.1.4 Basic flow of the MPS approach

In the MPS method, the control equation is solved by a predictor-corrector semi-implicit method. The general flow of a single time step is as follows:

(1) After completing the initial arrangement and entering the time step cycle, the first step is to correct the position and velocity of particles. The first correction is to calculate the intermediate velocity without considering the pressure and then move the particle to the intermediate position based on the increment of the velocity:

$$\mathbf{u}^{(*)} = \mathbf{u}^{(k)} + \Delta t g \tag{6}$$

$$\mathbf{r}^{(*)} = \mathbf{r}^{(k)} + \Delta t \mathbf{u}^{(*)} \tag{7}$$

(2)Calculate the particle numbers density n^*

Particle number density is the denseness of neighbouring particles around a particle, specifically refers to the accumulation of the nuclear function of the particle i and its neighbouring particles within the scope of the nuclear function. The formula is as follows:

$$\langle n \rangle_i = \sum_{j \neq i} W(|r_j - r_i|) \tag{8}$$

In order to provide the newest particle number density field in the next solution of pressure Poisson equation, we need to search the neighbor particle of each particle again, and calculate the middle step density of each fluid particle.

(3)Solving pressure Poisson equation

In the MPS method, the particle pressure is obtained by solving the pressure Poisson equation. The pressure Poisson equation used in this study is as follows[18][19][20]:

$$\nabla^2 p^{(k+1)} = \rho_0 \frac{\nabla \cdot u^{(*)}}{\Delta t} + \alpha \rho_0 \frac{n_0 - n^{(k)}}{n_0 \Delta t^2} \tag{9}$$

k+1 is the time step, and α is chosen by the following formula:

$$\alpha = \begin{cases} \left|\frac{n_0 - n^{(k)}}{n_0}\right| + \Delta t \left|\nabla \cdot u^{(k)}\right|, (n_0 - n^{(k)}) \nabla \cdot u^{(k)} \geq 0 \\ \left|\frac{n_0 - n^{(k)}}{n_0}\right|, (n_0 - n^{(k)}) \nabla \cdot u^{(k)} < 0 \end{cases} \tag{10}$$

The Laplacian model is used to discretize the second derivative terms in the governing equations. The Laplace operator is described as follows:

$$\left\langle \nabla^2 \phi \right\rangle_i = \frac{2d}{n_0 \lambda} \sum_{j \neq i} \left(\phi_j - \phi_i \right) w\left(\left| \boldsymbol{r}_j - \boldsymbol{r}_i \right| \right) \tag{11}$$

The λ in the formula is obtained by using the following formula:

$$\lambda = \frac{1}{n_0} \sum_{j \neq i}^{N} w(r_{ij}) r_{ij}^2 \tag{12}$$

(4)Pressure gradient
the following formulas are used to solve pressure gradients:

$$\left\langle \nabla p \right\rangle_i = \frac{d}{n_0} \sum_{j \neq i}^{N} \frac{p(r_j) - \tilde{p}(r_i)}{r_{ij}^2} (r_j - r_i) w(r_{ij}) \tag{13}$$

d is the dimension of the problem, n_0 is the initial particle number density, and $\tilde{p}(r_i)$ is the minimum pressure among all particles in the support domain.
Similarly, the divergence model is used to discretize the divergence of velocity in the governing equation. The expression is:

$$\left\langle \nabla \cdot V \right\rangle_i = \frac{d}{n_0} \sum_{j \neq i} \frac{(V_j - V_i) \cdot (r_j - r_i)}{| r_j - r_i |^2} W(| r_j - r_i |) \tag{14}$$

After the pressure gradient of each particle is calculated, the position and velocity of the second particle can be corrected according to the following formula:

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(*)} - \Delta t \frac{\nabla p^{(k+1)}}{\rho_0} \tag{15}$$

$$\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} + \Delta t \mathbf{u}^{(k+1)} \tag{16}$$

2.2 Explicit solution of pressure Poisson equation

Because the MPS method adopts the semi-implicit method to solve the Poisson equation, we need to solve the linear equations in the process of solving, the calculation efficiency is not very high, so this paper adopts the explicit solution method to solve the pressure Poisson equation, that is, do not solve the equations, directly solve the pressure of particles through the formula iteration; and the MPS method adopts the implicit method to solve the Poisson equation brings difficulties for GPU parallelism, so we use the explicit solution form to speed up the process of GPU parallelism.

2.2.1 Relaxed Jacobi

As we all know, when we bring the Laplace model in this paper into the Poisson equation, it will form a linear equation system about pressure, and can be discretized into the form of linear equation system $Ax = b$. The original CPU solver is solved implicitly through the GMRES method. Here we use the Jacobian relaxation iteration method to solve the pressure Poisson equation, which is described below:

According to the Relaxed Jacobi method, we can iteratively solve for the particle pressure value:

$$p_i^{l+1} = (1-\omega)p_i^l + \omega\frac{b_i - \sum_{j\neq i} a_{ij}p_j^l}{a_{ii}} \tag{17}$$

Where l is the number of iterations, ω is called the relaxation factor, b_i is the right-hand term of the pressure Poisson equation.

When the time step is over, we get the pressure of each particle p^{l+1}. When the next time step is done, we take the pressure of the last time step p^{l+1} as the initial pressure p^{l+1} of the current time step.

Iterations and relaxation factor are selected using the following formula：

$$R = \frac{\sum_{i=1}^{N}(Ap_i - b_i)}{N} \tag{18}$$

In the formula, A is the coefficient matrix of the pressure Poisson equation, p is the pressure value solved by Jocabi method, N is the total number of particles involved in solving the pressure Poisson equation, i is the particle sequence number, and R is called the residual value. When R < 10, the selection of iteration times and relaxation coefficient is reasonable. In this paper, the selection of relaxation factor and iterations follow this formula,no further details are given below.

2.2.2 2-D dam-break simulation

In order to verify the feasibility of explicit solution of the pressure Poisson equation, in this paper, a 2-D dam-break example is used.

As shown in Figure 4, the liquid portion is 0.6m in length and height, the boundary portion is 1.61m in length and 0.6m in height, and a layer of solid particles is arranged outside to prevent particles from passing through. The particle spacing is 0.005m, the time step is dynamically arranged and the maximum is 0.001s. The total number of particles is 15530, among which liquid particles are 14400, internal solid particles are 563 and external solid

particles are 567. The acceleration of gravity g=9.81m/s, and the density of water is 1000kg/m^3.After testing, we chose 0.5 relaxation factor and 150 iterations to get the best results.

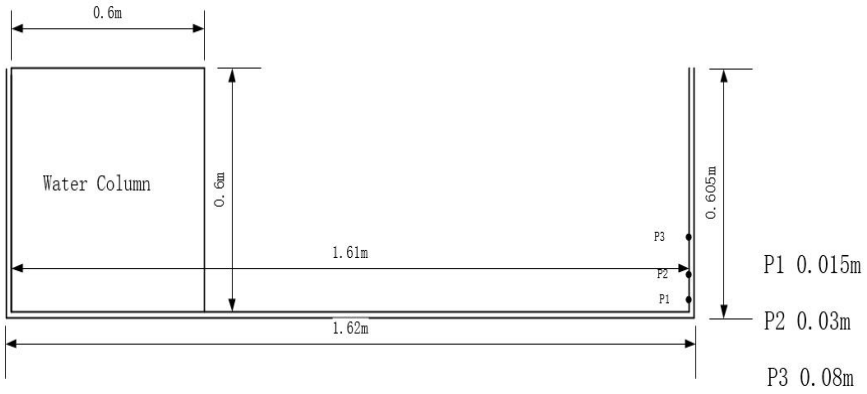


Fig 4. Sketch of 2-D dam-break

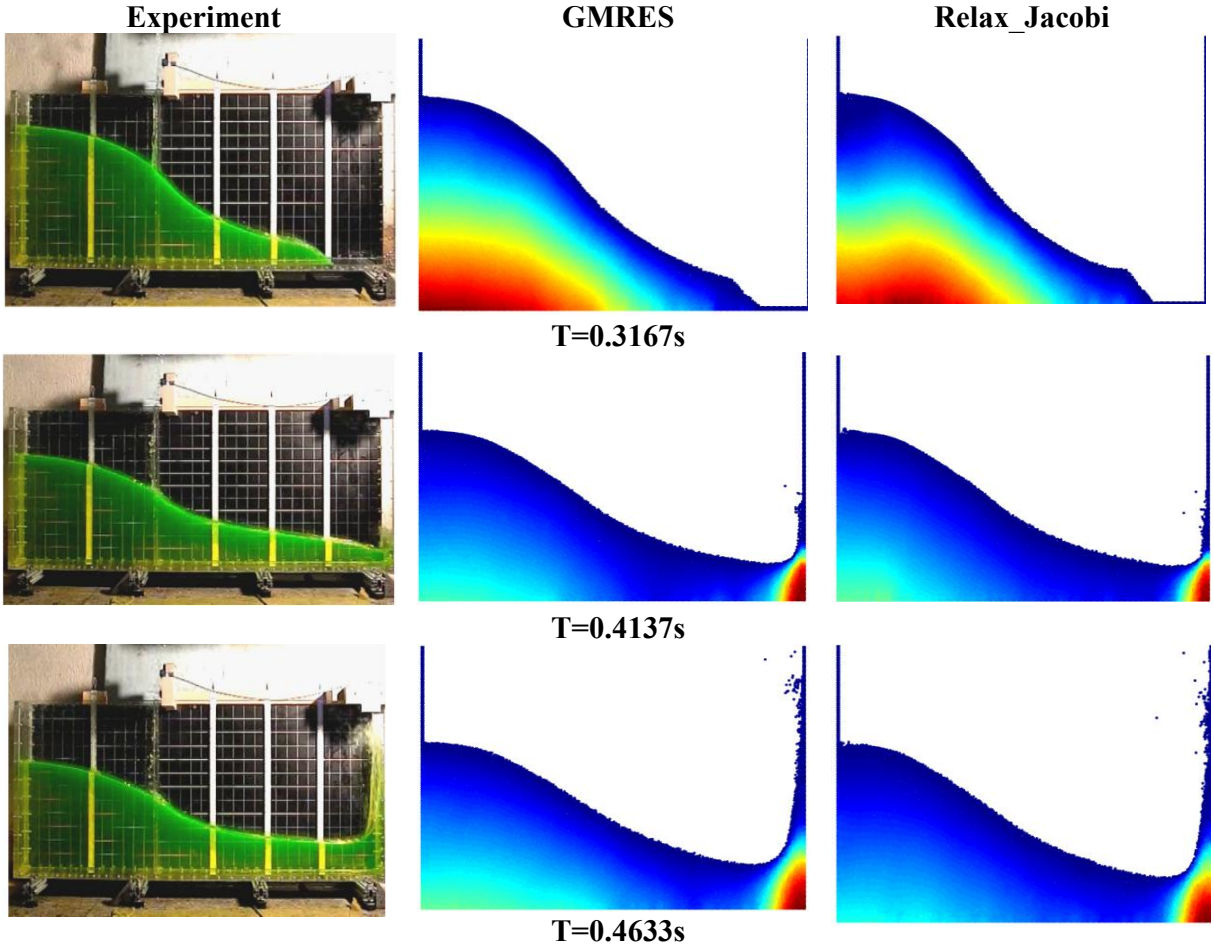


Fig 5. Comparison of numerical simulation Results at T = 0.3167s, T= 0.4137s and T = 0.4633s

In order to verify the accuracy of the explicit solution, we selected three pressure measurement points on the right side of the wall, and compared with the experimental results of Lobovsky et al.[21], as well as the pressure time curve, pressure cloud diagram and free

surface diagram of the implicit GMRES for solving the pressure Poisson equation, to prove the feasibility of the explicit solution of the pressure Poisson equation. The calculation results are as follows:

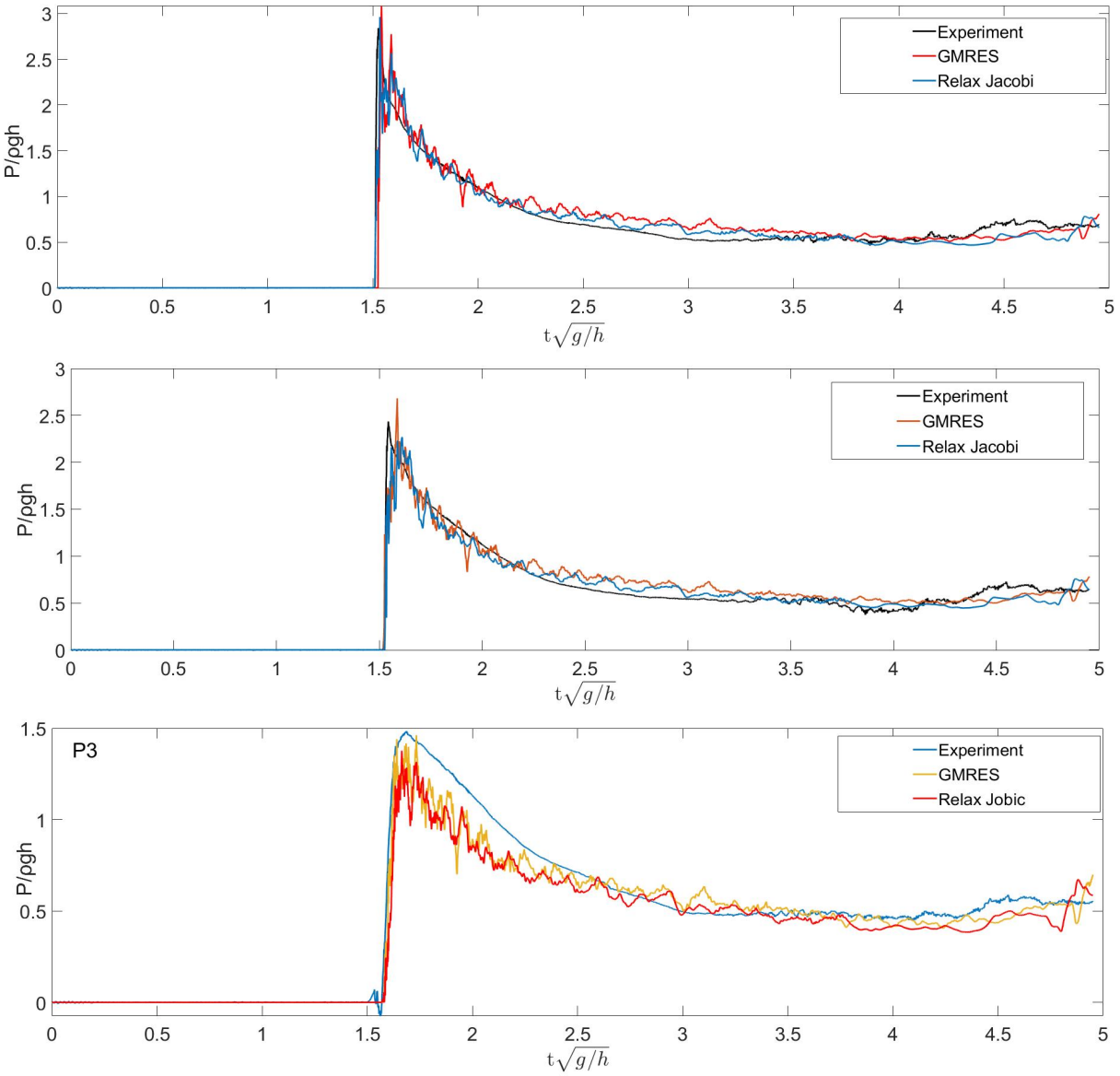


Fig 6. Comparison of pressure duration curves at P1, P2 and P3

As shown in Figure 5, according to Lobovsky et al. 's experiment, we selected three moments for analysis, t = 0.3167s, t = 0.4137s and t = 0.4633s, with a time error of no more than 5ms. According to the results, the explicit solution of pressure Poisson equation is consistent with the implicit solution and experiment, and the smoothness of pressure field is better.

As shown in Figure 6, the experimental, GMRES method, Relaxed Jacobi method under the P1, P2, P3 three measurement points under the pressure curve, horizontal and vertical coordinates are through dimensionless, respectively, $t\sqrt{(g/h)}$ and $P/\rho gh$. After comparison, Relaxed Jacobi method is in good agreement with experiment and GMRES method.

In conclusion, the Relaxed Jacobi method is feasible to solve the pressure Poisson equation. Then, the time of solving the pressure Poisson equation by GMRES method and Relaxed Jacobi method is calculated. The results are shown in Figure 7:

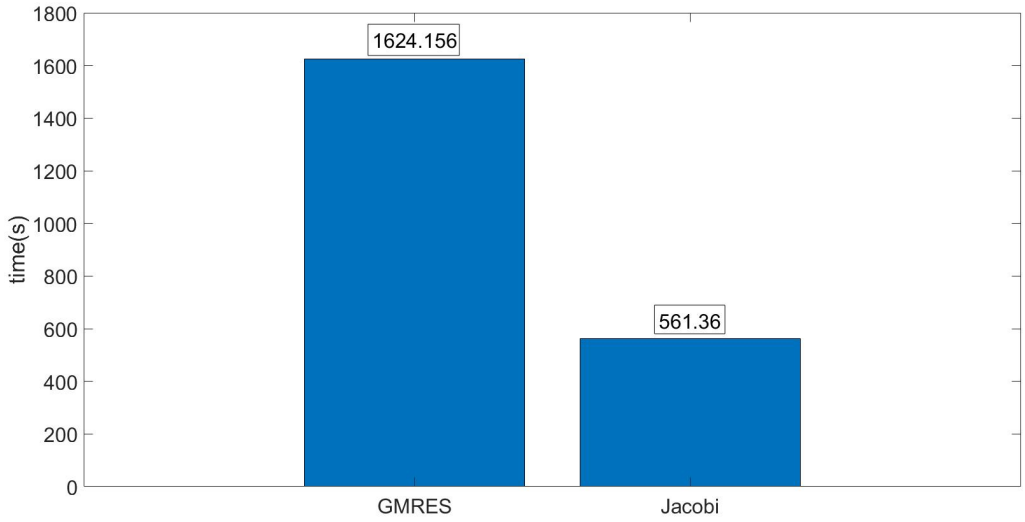


Figure 7. Solution time of pressure Poisson equation

Compared with the GMRES method, the Relaxed Jacobi method reduces the time of solving the pressure Poisson equation and improves the efficiency by 65.4%.

3. GPU parallelision for MPS

In 2.2, we verify the feasibility of explicitly solving Poisson's equations. In this section, we parallelize the explicit solution of Poisson's equations into GPU parallelization.

3.1 GPU parallel computing

3.1.1 CUDA architecture

CUDA (Compute Unified Device Architecture) is a hardware and software architecture released by NVIDIA to manipulate GPU computing. It is a general parallel computing platform and programming model based on NVIDIA's GPUs. It provides a simple interface for GPU programming. CUDA -based programming allows you to build GPU -based applications and use the GPUs parallel computing engine to solve more complex computing challenges more efficiently. It treats the GPU as a data-parallel computing device and does not need to map these calculations to a graphics API. The operating system's multitasking mechanism allows CUDA to access both the GPU and graphical runtime, and its computational features enable CUDA to visually write GPU core programs.

CUDA consists of a CUDA library, an application programming interface (API) and its runtime, and two high-level general-purpose mathematical libraries, CUFFT and CUBLAS. CUDA improves the read-write flexibility of DRAM so that the GPU fits the mechanism of

CPU. On the other hand, CUDA provides on-chip shared memory that allows threads to share data. Applications can utilize shared memory to reduce DRAM data transfers and rely less on DRAM memory bandwidth.

3.1.2 CUDA programming mode

The CUDA architecture introduces the concepts of host and device. CUDA programs contain both host and device programs. At the same time, host and device can communicate so that data can be copied between them. Among them, the memory of the CPU and the system (memory bar) is called the host, and the display memory of the GPU and the GPU itself is called the device.

A typical CUDA program executes as follows:
1. Allocating host memory and initializing data;
2.Allocating device memory and copying data from host to the device;
3.Call the kernel function of CUDA to complete the specified operation on the device;
4. Copy the operation result on the device to the host;
5. Free memory allocated on device and host.

3.2 GPU algorithm for solving pressure Poisson equation

3.2.1 Row compression method (CSR)

The pressure Poisson equation and the Laplace model are combined into a linear system. The coefficient matrix A of the Poisson equation is a typical sparse matrix. We cannot store all the elements in matrix A, otherwise the number of particles that can be simulated will be greatly reduced, which is not ideal for GPUs with smaller memory sizes. Therefore, this article will use the Compressed Sparse Row method to store sparse matrices. The matrix is represented by three one-dimensional arrays, A _ temp, JA _ temp, and IA _ temp. On the one hand, it can save storage space. On the other hand, three arrays are used for storage, which is easy to use. The array A _ temp stores all nonzero elements of the coefficient matrix; the JA _ temp array stores the column index of all nonzero elements of the coefficient matrix A; and the array IA _ temp stores the cumulative number of nonzero elements in row I, excluding row I. We found the number of non-zero elements on line I by looking at IA _ temp [I-1] - IA _ temp [I].
Using row compression (CSR) format to store matrix - related information facilitates the transfer of matrix information in the GPU parallel algorithm in this paper.

3.2.2 Introduction of GPU parallel algorithm for solving pressure Poisson equation

In Chapter 2, we introduce the explicit solution of the pressure Poisson equation by Relaxed Jacobi. In this section, we parallelize the pressure Poisson equation with GPU. The process is shown in Figure 8:

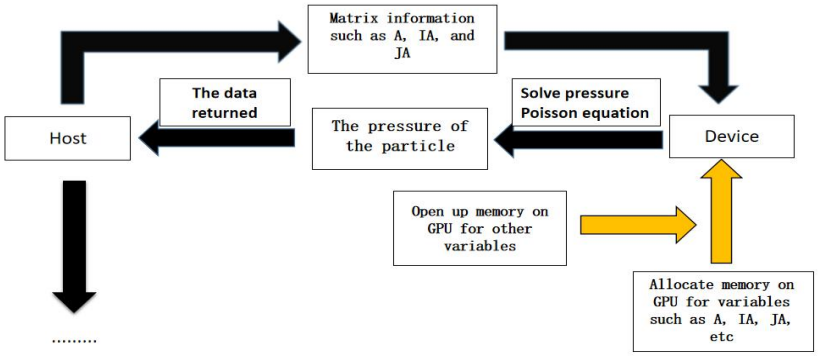


Fig 8. GPU parallel process for pressure Poisson equation

In the previous introduction, in the process of solving the pressure Poisson equation, we can get the linear equations about pressure by combining the Laplace model with the pressure Poisson equation, and then disperse them into the system of linear equations. Matrix information is already stored in A _ temp, JA _ temp, and IA _ temp through the CSR method, so we need to open memory for arrays A _ temp, JA _ temp, IA _ temp, and BM on the GPU and pass the Matrix information from the CPU to the GPU.

After the matrix information is transferred, we need to open up space in the GPU for the other variables in the Relaxed Jacobi method on the CPU solver to convert the host variable into the corresponding device variable.

To determine the accuracy of our calculations, we need to test the data transfer from the CPU to the GPU, and after each GPU invocation, we need to test the results, not verbose here.
After solving the pressure Poisson equation, we need to get the pressure back to facilitate the CPU solver to continue to calculate, and finally get the results.

3.3 2-D dam-break simulation

3.3.1 Computational verification

The numerical model still adopts the model in 2.2.2, The calculation configuration is as follows：

CPU	AMD Ryzen 9 3900X 12-Core Processor 3.97GHz 64GB
GPU	NVIDIA GeForce RTX 2060 6GB
compiler	CUDA 11.2 GCC 7.5.0

And other conditions remain unchanged, to verify the suitability of the GPU code, the CPU and GPU computations are as follows:

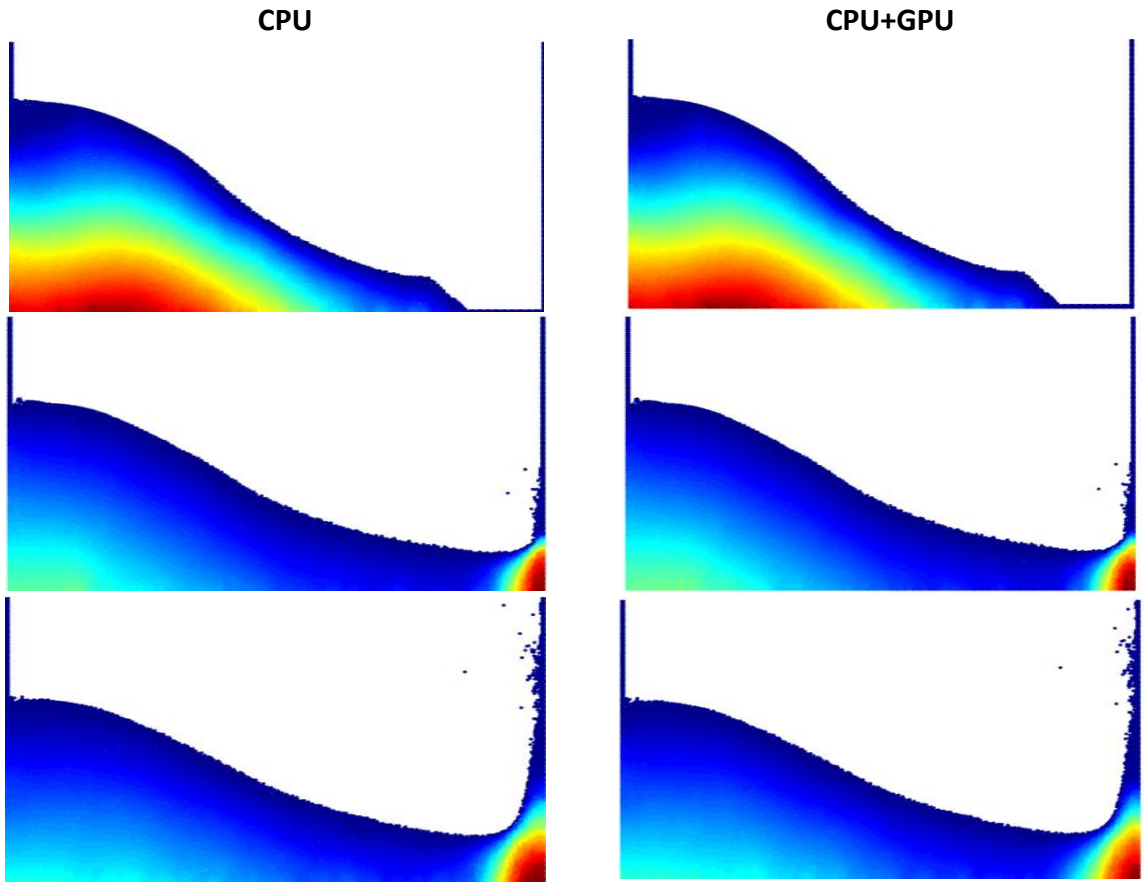
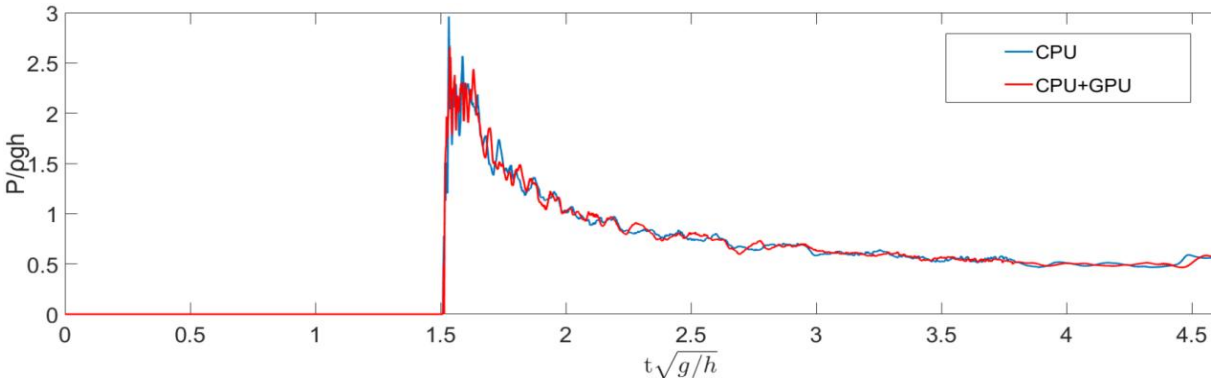


Fig 9. Comparison chart of calculation results at t = 0.3167s, t = 0.4137s and t = 0.4633s

As shown in Figure 9 , the results under CPU are basically consistent with those under GPU+ CPU, the outlines of flow field are basically the same, and the smoothness of pressure field is better.

In theory, the CPU and CPU+GPU calculations should be the same, but the data transmitted by the CPU and GPU, there will be data loss of the last digit, although the gap is small, but will still have an impact on the results of the calculation, so the pressure curve in Figure 140has a certain difference is normal, still can prove the adaptability of GPU code.



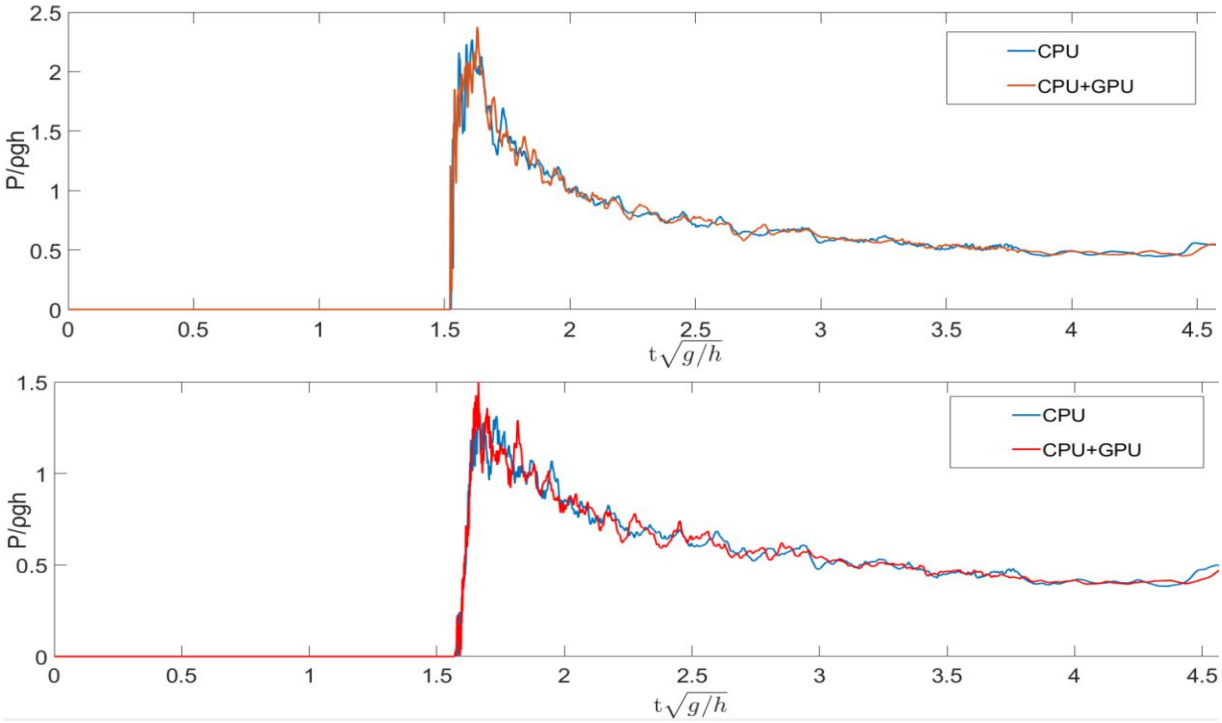


Figure. 10 Comparison of pressure duration curves at P1, P2 and P3

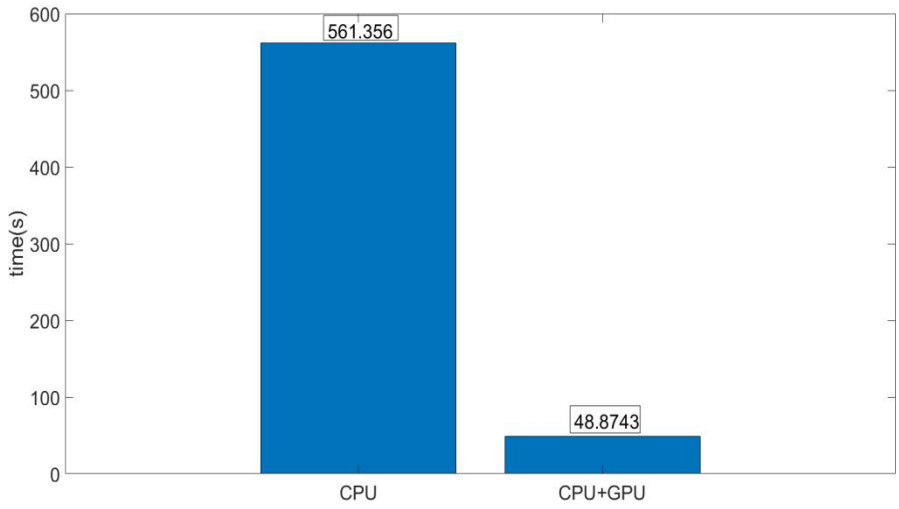


Figure 11. Pressure Poisson Equation Solution Time

As shown in Figure 11, By statistics, compared with the previous CPU serial code, the efficiency of GPU parallelization is improved by 91.3%.

3.3.2 The solution acceleration ratio of different particle numbers

After verifying the accuracy of the GPU parallel program, we discuss the influence of different particle numbers on the acceleration ratio. Table 1 lists the basic information of different 2-D dam-break examples.

Table 1. Basic Parameters of the Calculation Example

dr	Particle numbers	iterations	Relax factors
0.005	15k	150	0.5
0.0075	7.1k	80	0.5
0.01	4.1k	45	0.5

The number of iterations here refers to the number of iterations in Relaxed Jacobi. Calculation and verification show that the number of iterations needs to be further increased with the increase of the number of particles in the example. Too small number of iterations will lead to the occurrence of calculation irregularities and particle erasure.

Through calculation, Fig. 12 shows the acceleration ratio of Poisson's equation under different particle numbers(Here the acceleration ratio is the ratio of the solution time of the pressure Poisson equation under CPU to the solution time of the pressure Poisson equation under GPU. The greater the acceleration, the better). It can be seen that the acceleration ratio increases with the increase of particle number,the maximum acceleration ratio can reach 11.486.

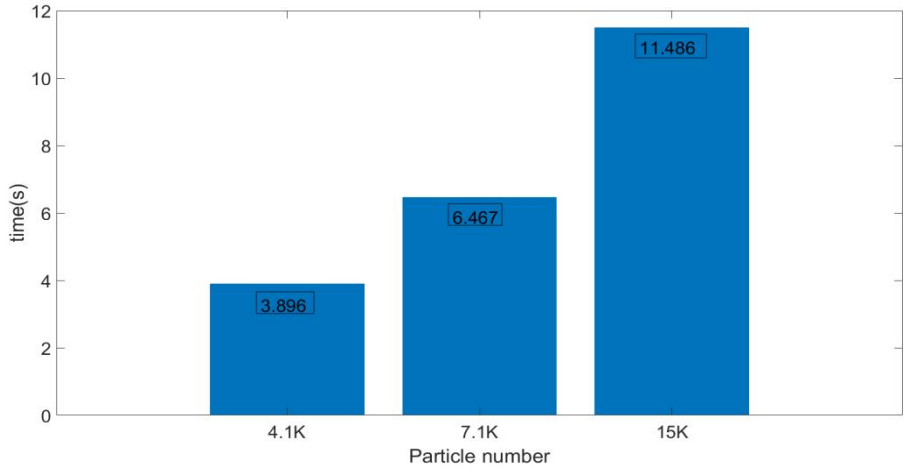


Figure 12. Acceleration ratio at different particle numbers

4. Conclusion

Based on the MPS — CPU solver, the Relaxed Jacobi method is used to solve the pressure Poisson equation explicitly instead of the GMRES implicit method. The simulation results are in good agreement with the GMRES implicit method, the efficiency of solving pressure Poisson equation is greatly improved. Based on the above research, the GPU parallelization of the Relaxed Jacobi method for the explicit solution of the pressure Poisson equation is completed, and the simulation results agree well with each other. The efficiency of the solution of the pressure Poisson equation is greatly improved. The maximum acceleration ratios of 11.486 can be obtained by simulating 2-D dam-break with different particle numbers.

On the whole, the overall computational efficiency has not been improved effectively due to the time consuming of matrix data transmission. Parallelization of a portion of the GPU does improve that portion of the computation. Therefore, on the basis of the current part of GPU, all the computations are given to GPU processing, so the potential of GPU high-speed computing can be more developed.

ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (No. 52171295 and 52192692), Open Project of State Key Laboratory of Deep Sea Mineral Resources Development and Utilization Technology (Grant No. SH-2020-KF-A01), Young Scholar Supporting Project of Dalian City (Grant No. 2020RQ006), Liao Ning Revitalization Talents Program（No：XLYC1908027）, Fundamental Research Funds for the Central Universities (No. DUT20TD108) to which the authors are most grateful.

References:

[1] CUDA_C_Programming_Guide.NVIDIA,CUDA Toolkit Doc,2015.
[2] Harada T, Koshizuka S, Kawaguchi Y. Smoothed particle hydroddynamics on GPUs[C]. Computer Graphics International.Petropolis: SBC,2007: 63-70.
[3] Zhang,Y X, Solenthaler B, Pajarola R. GPU accelerated SPH particle simulation and rendering[C]. ACM SIGGRAPH 2007 posters. ACM, 2007.
[4] Crespo A J C, Dominguez J M, Barreiro A, et al. GPUs, a new tool of acceleration in CFD: efficiency and reliability on smoothed particle hydrodynamics methods[J]. PLoS One, 2011, 6(6): e20685.
[5] Herault, A., Bilotta, G., Dalrymple, R.A. SPH on GPU with CUDA[J]. Journal of Hydraulic Research, 2010, 48(Extra): 74-79.
[6] Wei, Z.P., Dalrymple, R.A., Hérault, A., Bilotta, G., Rustico, E., Yeh, H. SPH modeling of dynamic impact of tsunami bore on bridge piers[J]. Coastal Engineering, 2015, 104: 26-42.
[7] Chow, A.D., Rogers, B.D., Lind, S.J., Stansby, P.K. Numerical wave basin using incompressible smoothed particle hydrodynamics (ISPH) on a single GPU with vertical cylinder test cases[J]. Computers & Fluids, 2019, 179: 543-562.
[8] Hori, C., et al., GPU-acceleration for moving particle semi-implicit method[J]. Computers & Fluids, 2011. 51(1): p. 174-183.
[9] Kakuda, K., et al., Particle-based fluid flow simulations on GPGPU using CUDA[J]. CMES: Computer Modeling in Engineering and Sciences, 2012. 88(1): p. 17-28.
[10] Gou, W., S. Zhang and Y. Zheng, Simulation of isothermal multi-phase fuel-coolant interaction using MPS method with GPU acceleration[J]. Kerntechnik, 2016. 81(3): p. 330-336.
[11] Vieira-e-Silva, A.L.B., et al., Improved meshless method for simulating incompressible fluids on GPU[C]. 2017 19th Symposium on Virtual and Augmented Reality (SVR). Curitiba, Brazil.November 1-4,2017, p. 297-308.
[12] Taniguchi, D., Sato, L.M., Cheng, L.Y. Explicit moving particle simulation method on GPU cluster[C]. 10th World Congress on Computational Mechanics, Brazil, July 8-13, 2012: 1155-1168.
[13] Kawamura, K., Hashimoto, H., Onodera, N., Munesue, T. GPGPU simulation of oil tank sloshing based on explicit MPS method[C]. Proceedings of 3rd International Conference on Violent Flows, Osaka, Japan, March 9-11, 2016: Paper No.56.
[14] Hashimoto, H., Hata, Y., Kawamura, K. Estimation of oil overflow due to sloshing from oil storage tanks subjected to a possible Nankai Trough earthquake in Osaka bay area[J]. Journal of Loss Prevention in the Process Industries, 2017, 50: 337-346.
[15] Sun, Z., Djidjeli, K. & Xing, J. T.,A Mixed Particle-Mode function Method for Nonlinear Marine Fluid-Structure Interaction Problems with Free Surface[D],University of Southampton,2016.
[16] Koshizuka, S. and Oka, Y. (1996). Moving-particle semi-implicit method for fragmentation of incompressible flfluid.[J] *Nuclear Science and Engineering*, 123:421–434.

[17] Koh, C. G., Gao, M. & Luo, C., A new particle method for simulation of incompressible free surface flow problems[J]. *INT J NUMER METH ENG***89** 1582 (2012).
[18] Khayyer, A. and Gotoh, H. (2013). Enhancement of performance and stability of MPS mesh-free particle method for multiphase flflows characterized by high density ratios.[J] *Journal of Computational Physics*, 242:211–233.
[19] Kondo, M. and Koshizuka, S. (2011). Improvement of stability in moving particle semi-implicit method.[J] *International Journal for Numerical Methods in Fluids*, 65(6):638–654.
[20] 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.[J] *Computer Methods in Applied Mechanics and Engineering*, 200(9-12):1113–1125.
[21] Lobovsky, L., Botia-Vera, E., Castellana, F., Mas-Soler, J., and Souto-Iglesias, A. (2014). Experimental investigation of dynamic pressure loads during dambreak. *Journal of Fluids and Structures*, 48:407–434.