VIII International Conference on Particle-Based Methods
PARTICLES 2023
U. Perego, M. Cremonesi, A. Franci (Eds)

ISPH WITH A GEOMETRIC MULTIGRID PRECONDITIONING SOLVER USING BACKGROUND CELLS IN GPU ENVIRONMENT

H. OSAKI[1], D. MORIKAWA[2] AND M. ASAI[3]

[1] Department of Civil Engineering, Graduate School of Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka 819-0395, JAPAN
e-mail: h-osaki@doc.kyushu-u.ac.jp, https://kyushu-u.wixsite.com/structural-analysis

[2] Ph.D., Center for Mathematical Science and Advanced Technology, JAMSTEC
3173-25 Shōwamachi, Kanazawa Ward, Yokohama, Kanagawa, JAPAN
e-mail: morikawad@jamstec.go.jp

[2] Ph.D., Associate Prof., Department of Civil Engineering, Graduate School of Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka 819-0395, JAPAN
e-mail: asai@doc.kyushu-u.ac.jp, https://kyushu-u.wixsite.com/structural-analysis

Abstract. Incompressible fluid analysis using the ISPH or MPS methods requires the solution of the pressure Poisson equation, which takes up most of the overall computation time. In addition, the iteration number for solving pressure Poisson equations may increase as the simulation model scale increases. This is a common problem in particle methods and the other implicit time integration solvers. In different methods, FEM, etc., good quality preconditioning, such as multigrid preconditioning, can significantly improve the convergence of iterative solution methods. There are two types of multigrid preconditioners, algebraic multigrid and geometric multigrid methods, but there are few examples of their application in particle methods.

In this study, we attempted to develop a framework for a geometric multigrid preconditioner for solving the pressure Poisson equation in the ISPH. First, we focused on the geometric multigrid preconditioner using background cells, which are used for neighboring particle search, and implemented it on a GPU environment. Through a simple dam-break problem, we compared the computation time between the Conjugate gradient (CG) solver with a traditional preconditioner and the CG solver with a geometric multigrid preconditioner. We confirmed that the background cell-based geometric multigrid preconditioner is practical for the ISPH method.

Keywords: SPH, ISPH, Multigrid Method, Preconditioner, GPU.

1 INTRODUCTION

In Japan, large-scale disasters have occurred frequently in recent years, including the Great East Japan Earthquake of 2011. The importance of disaster impact assessment based on numerical analysis is increasing to prepare for such large-scale disasters. Particle methods such as the SPH (Smoothed Particle Hydrodynamics) method[1] are used for disaster simulations such as tsunami run-up analysis[2] and slope disaster[3]. Water and ground are modeled in these disaster simulations as incompressible or highly viscous fluids. When solving incompressible viscous fluids with the SPH method, the semi-implicit ISPH (Incompressible SPH) method[2] and the implicit IISPH (Implicit ISPH) method[3] are used to relax the CFL condition and the diffusion number condition and to solve the fluid stably. In this method, the pressure and velocity are obtained by solving a simultaneous linear equation using an iterative

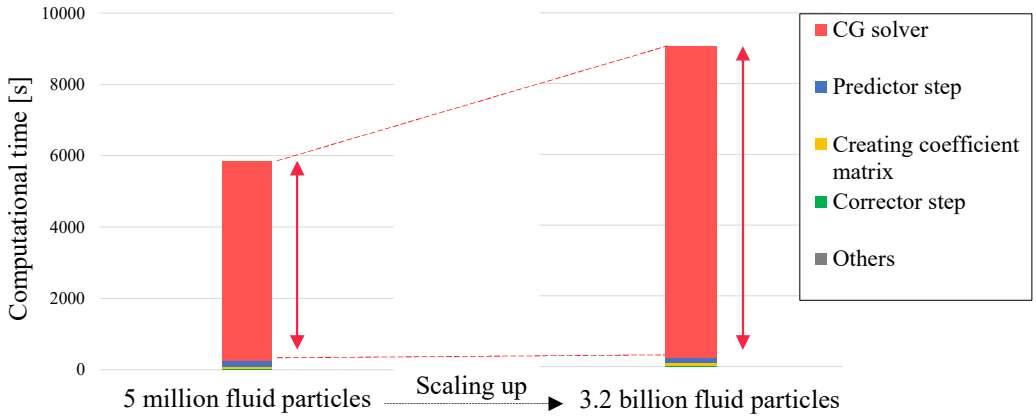

Figure 1. Breakdown of computation time for dam-break calculations using the ISPH method (left: 5 million particles (1 GPU), right: 320 million particles (64 GPUs))

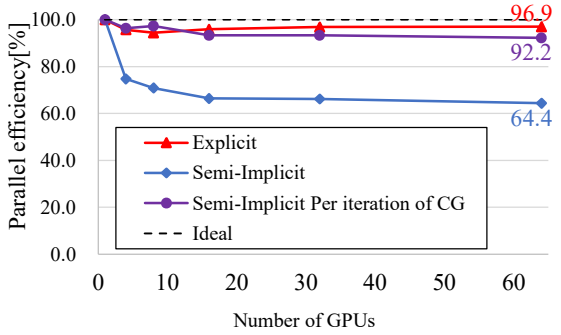

Figure 2. Results of weak scaling

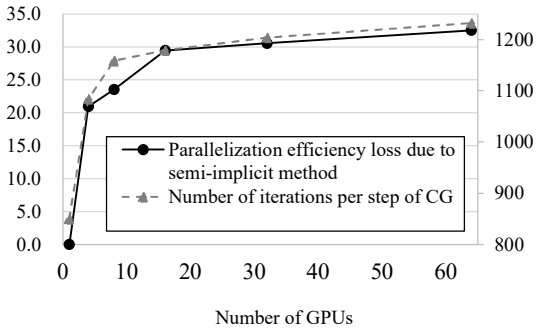

Figure 3. Efficiency loss of the semi-implicit method and number of iterations of the CG method

solver such as the conjugate gradient (CG) method.

Since the SPH method must be solved with uniform particle spacing, the computational model is inevitably large for wide-area disaster simulations. In general, it is difficult to maintain scalability (efficiency of parallel computation) in parallel computation when solving large-scale simultaneous linear equations.

Fig. 1 shows a Breakdown of computation time for dam-break calculations using ISPH. The iterative matrix solver accounts for most of the computation time, and increasing the computation time increases the number of solver iterations, which increases the computation time. The computational efficiency is also reduced accordingly. Fig. 2 shows the weak scalability of the SPH method when up to 64 GPUs are used to analyze the dam-break problem. It can be seen that the explicit method maintains high parallelization efficiency even when the number of GPUs and the problem size are increased. In contrast, the semi-implicit method significantly decreases parallelization efficiency as the problem size increases, indicating that the scalability is not maintained. To investigate the cause of this problem, Fig. 3 shows the number of iterations required to solve a simultaneous linear equation using the conjugate gradient method. As shown in the figure, the number of iterations required until convergence

increases as the problem size increases. In other words, it is impossible to maintain the high parallelization efficiency of the implicit solver unless there is a good preconditioning technique that prevents the number of iterations in the iterative solver from increasing as the problem size increases. There is a multigrid method for grid-like computational models such as the difference method [4,5] where the number of iterations does not depend on the problem size. There are many examples of application of multigrid methods to numerical methods that use grids or meshes, such as the finite difference method (FDM)[4] and the finite element method (FEM) [6]. However, there are few examples of application of the multigrid method to particle methods[7-9], and none to the SPH method used in this study. In this study, we develop a multigrid preconditioning solver for the linear equations of the ISPH method. Specifically, following previous studies[8,9], we develop a geometric multigrid preconditioning solver that constructs an auxiliary grid based on the background cell used to search for neighboring particles. Then, we compare the number of iterations between the no-preconditioning CG method and the multigrid preconditioning CG method using a dam-breaking problem for incompressible fluids and confirm the effectiveness of the multigrid preconditioning method.

2 INCOMPRESSIBLE SMOOTHED PARTICLE HYDRODYMANICS (ISPH)

2.1 Governing equations

The governing equations in ISPH methods are the incompressible Navier-Stokes equation and continuity equation, which, in vector notation, can be written as

$$\frac{D\mathbf{v}}{Dt} = -\frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{v} + \mathbf{g} \tag{1}$$

and

$$\nabla \cdot \mathbf{v} = 0, \tag{2}$$

where $\boldsymbol{v}$ is the velocity vector, D/Dt the time derivative, p the pressure, ρ the density of the fluid, ν the kinematic viscosity, $\boldsymbol{g}$ the acceleration of gravity, and t the time, respectively.

2.2 SPH Approximations

The SPH is a Lagrangian numerical technique that discretizes the space as individual particles and approximates the different attributes of each particle using a smoothing kernel function W. In this manner, the equations for approximating a genetic function ϕ (either scalar or vector) is

$$\langle\phi\rangle_i = \sum_j \frac{m_j}{\rho_j}\phi_j W(\mathbf{r}_{ij}, h), \tag{3}$$

where the subscripted indexes i and j represent the target and neighboring particles, respectively, h is the smoothing length, m_j is the mass of the fluid particle , ρ_j is the density of the fluid particle, $\boldsymbol{r}_{ij} = \boldsymbol{x}_i - \boldsymbol{x}_j$ is the relative position vector between particles i and j , and symbol $\langle\cdot\rangle$ signifies the application of the SPH approximation. In this study, we selected the cubic spline function as the kernel function and selected $h = 1.2d$.

The derivative $\nabla\phi$ and the Laplacian $\nabla^2\phi$ can be assumed by using the above-defined SPH approximation as follows:

$$\langle\nabla\phi\rangle_i = \sum_j \frac{m_j}{\rho_j}(\phi_j - \phi_i)\nabla W(\boldsymbol{r}_{ij}, h), \tag{4}$$

$$\langle\nabla\phi\rangle_i = \sum_j \frac{m_j}{\rho_j}\left(\frac{\phi_j}{\rho_j^2} + \frac{\phi_i}{\rho_i^2}\right)\nabla W(\mathbf{r}_{ij}, h), \tag{5}$$

$$\langle\nabla^2\phi\rangle_i = 2\sum_j \frac{m_j}{\rho_j}\frac{\mathbf{r}_{ij}\cdot\nabla W(\mathbf{r}_{ij}, h)}{{\mathbf{r}_{ij}}^2}(\phi_i - \phi_j), \tag{6}$$

where $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$ is the nabla operator.

In this study, Eq. (5) is used for the velocity divergence, while an alternative form for the SPH approximation of the derivative of a function, Eq. (6), is used for the pressure gradient.

2.3 ISPH formulation and linear matrix equation in the ISPH

The ISPH method is based on a fractional step method. The main advantage of the stabilized ISPH method is that the pressure is calculated through a theoretically rigorous pressure Poisson equation (PPE), as opposed to weakly compressible SPH (WCSPH), where it is calculated from an empirical equation of state. That is, ISPH is not as sensitive as WCSPH to empirical coefficients. However, the pressure distribution on ISPH is obtained by solving a $N_f \times N_f$ matrix equation from the PPE, where N_f is the number of fluid particles.

In this section, we describe ISPH formulation based on the fractional step method. First, the Navier-Stokes equation (Eq. (1)) is divided into two projection steps.

Predictor step:

$$\mathbf{v}^* = \mathbf{v}^n + \Delta t(\nu\nabla^2\mathbf{v}^n + \mathbf{g}) \tag{7}$$

and corrector step:

$$\mathbf{v}^{n+1} = \mathbf{v}^* + \Delta t\left(-\frac{\nabla p^{n+1}}{\rho}\right), \tag{8}$$

where Δt is the time increment, n and $n+1$ superscript indexes refer to the current and next time steps, and the superscript $*$ indicates the predictor step.

Then, multiplying both sides of Eq. (8) by ∇ leads to

$$\nabla\cdot(\mathbf{v}^{n+1} - \mathbf{v}^*) = \nabla\cdot\left[\Delta t\left(-\frac{\nabla p^{n+1}}{\rho}\right)\right]. \tag{9}$$

$\nabla\cdot\mathbf{v}^{n+1} = 0$ because of Eq. (2), so rearranging Eq. (9), the pressure can be calculated from the PPE

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t}\nabla\cdot\mathbf{v}^*. \tag{10}$$

[10] proposed a stabilized ISPH method to stabilize the pressure distribution. It combines density invariance and divergence-free conditions, which we also adopted in this study.

According to [10], PPE (Eq. (10)) reformulated as

$$\nabla^2 p^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{v}^* + \alpha \frac{\rho - \langle \rho^n \rangle}{\Delta t^2}, \tag{10}$$

where α is called the relaxation coefficient and is generally set to be much less than 1.0.

To summarize the time integration method in the ISPH method, first, the predictor velocity $\mathbf{v}^*$ is obtained by Eq. (8), then the pressure field p is obtained by Eq. (10). Finally, velocity is corrected by Eq. (9).

Using the SPH approximation for the Laplacian operator (Eq. (7)), the PPE (Eq. (10)) can be rewritten as

$$\sum_{j=1}^{N_f} 2\frac{m_j}{\rho_j}\frac{\mathbf{r}_{ij}\cdot\nabla W(\mathbf{r}_{ij},h)}{{\mathbf{r}_{ij}}^2}\left(p_i^{n+1} - p_j^{n+1}\right) = \frac{\rho}{\Delta t}\nabla\cdot\mathbf{v}^* + \alpha\frac{\rho - \langle\rho^n\rangle}{\Delta t^2}. \tag{11}$$

This is a $N_f \times N_f$ matrix equation with pressure p as the unknown variable.

3 MULTIGRID METHOD

As shown in section 2.3, matrix equation from the PPE needs to be solved in the ISPH method. This procedure is time-consuming and requires a large amount of memory, so we developed a background cell-based geometric multigrid preconditioner following previous studies[7-9].

3.1 General iterative solvers and multigrid method

In general, iterative matrix solvers such as Successive Over Relaxation (SOR), Conjugate gradient (CG), and so on, errors in the high-frequency component of the same frequency as the grid size converge quickly, and convergence of low-frequency components is delayed[4].

On the other hand, in the multigrid method, Errors in all frequency components converge uniformly. To understand the effect of the multigrid method, we describe it shortly.

The computational scheme of the multigrid method is as follows. First, pre-smoothing is performed on the fine grid to some extent, and the residuals that cannot be converged are restricted to the fine grid (restriction). Based on this, the solution is corrected on a coarse grid (coarse grid correction), extended to a fine grid (prolongation), and finally smoothed again on a fine grid (post-smoothing). The geometric multigrid method uses geometric information to perform the above calculations, whereas the algebraic one uses only the algebraic information of the coefficient matrix[7]. For example, numerical analysis methods that use grids or meshes, such as FDM[4] and FEM[6], can compute restriction and prolongation operations and coefficient matrices on coarse grids using their geometrical information. This is the geometric multigrid method. On the other hand, regardless of the type of numerical analysis method, only the coefficient matrix and the right-hand side vector are used as input information for their computation in the algebraic multigrid method.

Comparing the two methods in particle methods, [8] reported that the cell-based geometric multigrid method solver was more efficient because the algebraic multigrid solver needs to be set up at every time step due to the dynamic change in connectivity. Therefore, we developed a geometric multigrid method for ISPH with background cells in this study.

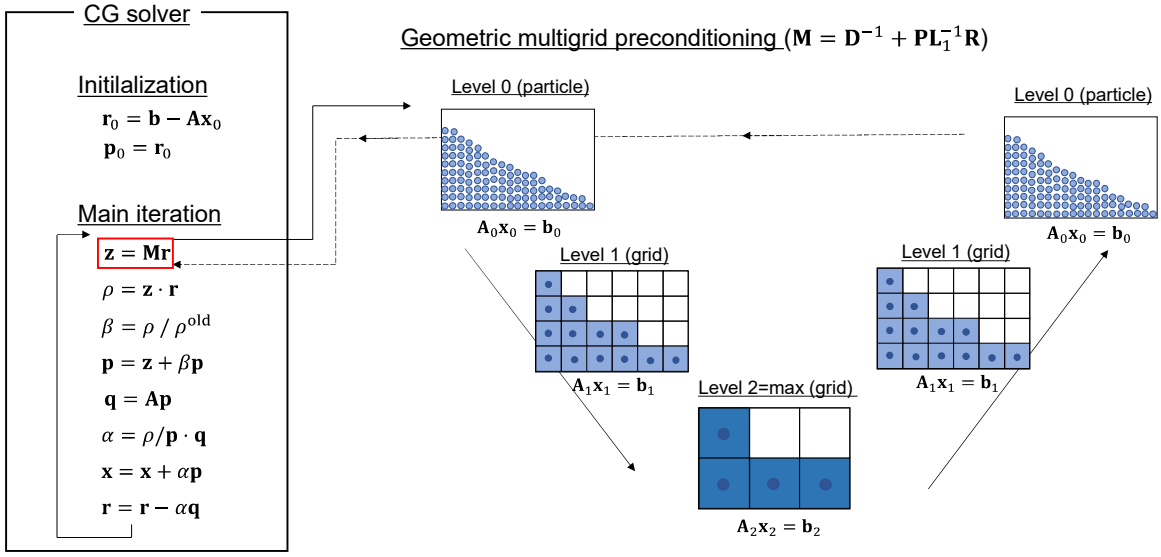

Figure 4. Algorithm for background cell-based geometric multigrid preconditioned conjugate gradient method (MGCG)

3.2 Background cell-based geometric multigrid preconditioner

As shown in section 3.1, the construction of the geometric multigrid method requires the definition of restriction and prolongation in a hierarchical manner. However, the grid is not used explicitly in the particle method, instead, background cells are used for neighboring particle search in particle methods in general.

Therefore, we thought of utilizing background cells to geometrically construct a multigrid and use them as preconditioning for the CG method following previous studies[7-9]. Fig 4. shows the algorithm for the background cell-based geometric multigrid preconditioned conjugate gradient method (MGCG). As shown in Fig. 4, the grid levels are numbered from the finer cells, and the grid with the width of one cell is set as level 1.

We define the restriction from particles to cells to take the sum of the variables of the particles in the cell:

$$p_1^c = \sum_{i \in c} p_i, \tag{12}$$

where the superscript c means the coarse grid, and in contrast, f means the fine grid. The subscript 1 means that the grid level is 1. On the other hand, the prolongation from cells to particles is defined as copying the variables of the cell containing the particle so that the restriction matrix $\mathbf{R}$ and the prolongation matrix $\mathbf{P}$ are in a transpose relation:

$$\mathbf{P} = \mathbf{R}^T. \tag{13}$$

As shown in Fig. 4, the geometric structure is defined so that there are 4 fine cells (8 in 3-dimensions) in one coarse cell. The restriction from the fine cell to the coarse cell is defined as

$$p_{l+1}^c = \sum_{l \in l+1} p_l^f \tag{14}$$

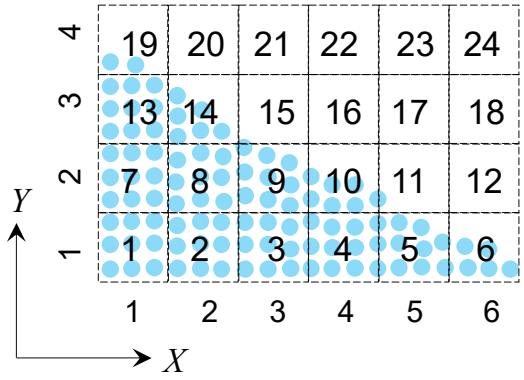

Figure 5. Schematic illustration of the cell grid

so that it is the sum of the variables of the fine grid as well as Eq. (12), and the prolongation from a coarse cell to the fine cell is defined so that $\mathbf{R}$ and $\mathbf{P}$ are transpose relations. As defined above, the coefficient matrix $\boldsymbol{A}_{l+1}$ in coarse cells of level $l+1$ can be defined recursively from fine cells of level l :

$$\mathbf{A}_{l+1} = \mathbf{R}_{l+1}\mathbf{A}_l\mathbf{P}_{l+1}. \tag{15}$$

In this study, following a previous study[9], we defined preconditioning matrix $\mathbf{M}$ as

$$\mathbf{M} = \mathbf{D}^{-1} + \mathbf{P}\mathbf{L}_1^{-1}\mathbf{R}, \tag{16}$$

where $\boldsymbol{D}^{-1}$ is the diagonal scaling matrix and $\mathbf{L}_1^{-1}$ is an iterative matrix representing the multigrid computation at grid level 1. $\mathbf{L}_1^{-1}$ defines recursively the multigrid calculation in the coarser cell hierarchy:

$$\begin{aligned}\mathbf{L}_l^{-1} = (\mathbf{A}_l)^{-1} - (\mathbf{I}_l - \mathbf{D}_l^{-\mathbf{1}}\mathbf{A}_l)^{\nu}(\mathbf{A}_l)^{-1}(\mathbf{I}_l - \mathbf{A}_l\mathbf{D}_l^{-\mathbf{1}})^{\nu} \\ +(\mathbf{I}_l - \mathbf{D}_l^{-\mathbf{1}}\mathbf{A}_l)^{\nu}\mathbf{P}_l\mathbf{M}_{l+1}\mathbf{R}_l(\mathbf{I}_l - \mathbf{A}_l\mathbf{D}_l^{-1})^{\nu},\end{aligned} \tag{17}$$

where ν is the number of smoothing iterations at level l. The calculations are performed by V-cycles with several iterations in each grid hierarchy using the Jacobi method as a smoother.

3.3 GPU implementation

We wrote the GPU part of the code entirely on CUDA Fortran using some cuSPARSE and Thrust libraries. Generally, the SPH equations described here will be calculated for every fluid or wall particle. Therefore, we associate one GPU thread for each particle and evoke them simultaneously in a CUDA kernel.

Here, in this section, we describe the background cells for searching neighboring particles and their use in constructing a multigrid. We leave a detailed explanation of the former to [2], and explain the latter in detail.

We do not use a neighbor list to store the neighbor information of each particle. Instead, we search for neighboring particles located in neighboring cells in every step.

First, a cell grid is defined at the beginning of the simulation in which each cell is a cube, the size of the influence radius r_e. Using the cubic spline kernel function, the influence radius yields $r_e = 2h$. As schematically illustrated in Fig. 5, each cell has a specific ID iC defined by its grid position (iC_X, iC_Y, iC_Z) as

$$iC = N_X N_Y (iC_X - 1) + N_X (iC_Y - 1) + iC_X, \tag{18}$$

where N_X and N_Y are the number of cell divisions in the *X* and *Y* directions. Notice that Eq. (18) enumerates the cells prioritizing the $X > Y > Z$.

The cell position for each particle is retrieved from

$$iC_X = \mathrm{int}\left(\frac{x - x_{\min}}{r_e}\right) + 1, \tag{19}$$

$$iC_Y = \mathrm{int}\left(\frac{y - y_{\min}}{r_e}\right) + 1, \tag{20}$$

$$iC_Z = \mathrm{int}\left(\frac{z - z_{\min}}{r_e}\right) + 1, \tag{21}$$

where int() is the operator that returns the largest integer number lower or equal to its input value, (x, y, z) are the each particle position and $x_{\min}$, $y_{\min}$, and $z_{\min}$ are the minimum values of (x, y, z) in the calculation domain, respectively. During these calculations, no additional arrays are necessary to find the cell ID that each particle belongs to. We sort particle ID using sort_by_key[11] in the Thrust library every time step as the particles move. And then, we store only the ID information of each cell's first and last particle in the array partCell.

On the other hand, the cell position from each cell ID iC is retrieved from

$$iC_X = \mathrm{mod}\big((iC - 1), N_X\big) + 1, \tag{22}$$

$$iC_Y = \mathrm{mod}\left(\mathrm{int}\left(\frac{iC - 1}{N_X}\right), N_Y\right) + 1, \tag{23}$$

$$iC_Z = \mathrm{int}\left(\frac{iC - 1}{N_X N_Y}\right) + 1. \tag{24}$$

where mod(*A,B*) is the operator that returns t remainder of *A* divided by *B*.

Using these formulas and the array partCell, we can compute the restriction and prolongation operations between cells and particles (between grid levels 0-1). The restriction and prolongation operations are computed by defining matrices $\mathbf{R}$ and $\mathbf{P}$. Since these are sparse matrices, they are stored in CSR format to reduce memory usage. Also, the coefficient matrices at the coarse grid level in Eq. (15) are computed using the sparse matrix – sparse matrix product library cusparseSpGEMM[12] in cuSPARSE libraries.

Cells located within $[iC_{Xf\min}, iC_{Xf\max}] \times [iC_{Yf\min}, iC_{Yf\max}] \times [iC_{Zf\min}, iC_{Zf\max}]$ are used as the grids, where $iC_{Xf\min}, iC_{Xf\max}$ are minimum and maximum positions in the *X* direction of the cell where fluid particles are present, the *Y* and *Z* directions are the same. For grid level 2, in other words, when multiple cells are combined into one coarse cell, cells located within $[iC_{Xf\min}/2, iC_{Xf\max}/2] \times [iC_{Yf\min}/2, iC_{Yf\max}/2] \times [iC_{Zf\min}/2, iC_{Zf\max}/2]$ are used as the grids and it can be calculated by changing N_X, N_Y, N_Z in Eqs. (19)-(21) to $N_X/2, N_Y/2, N_Z/2$ basically. For grid level 3 and above, the range is further divided by 2.

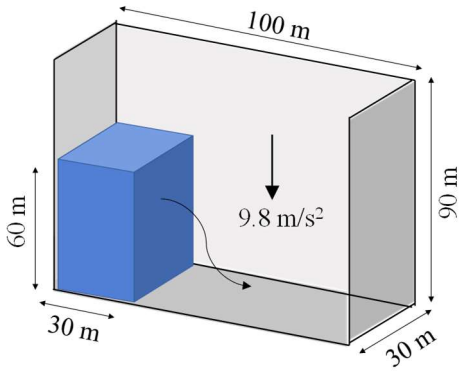

Figure 6. Model diagram of the dam break problem

Table 1. Calculation conditions

Parameters	Value
Particle diameter d_0	1.0 m
Time increment Δt	10^{-2} s
Stabilized parameter α	10^{-2}
Density ρ	10^3 kg/m^3
Kinematic viscosity ν	10^6 m^2/s
Number of total steps	4,000
Number of fluid particles	58,621

4 NUMERICAL EXPERIMENT

4.1 Problem

To confirm the background cell-based geometric multigrid preconditioned solver, we calculated the 3-dimensional dam-break problem for the incompressible fluid shown in Fig. 6. The calculation are performed using the conditions shown in Table 1.

4.2 Number of iterations

Fig. 7 shows convergence histories at t=3.2 s for the no preconditioning CG solver and cell-based multigrid preconditioned CG (MGCG) solver, where MGCG(m,n) means that m grids of the multigrid method are added and the number of smoothing iterations in each grid hierarchy is n. Compared to the CG solver, the MGCG solver improves the decreasing trend of residuals to convergence and reduces the number of iterations. Also, comparing MGCG(1,4) and MGCG(2,4), it was confirmed that the number of iterations can be significantly reduced by increasing the number of grids, even though the number of iterations in each grid hierarchy is the same.

Next, we compare the number of iterations when the model in Fig. 6 is run at different widths and with a different number of particles and show the result in Fig. 8. The number of iterations increases in the CG solver. In contrast, the number of iterations is suppressed in the MGCG solver, which confirms the effectiveness of multigrid preconditioning.

5 CONCLUSION

In this study, we tried to accelerate the pressure Poisson solver in ISPH by incorporating preconditioning using a background cell-based geometric multigrid method. As a result, the number of main iterations of the iterative solver was greatly reduced.

In the future, we will further tune the pre-processing matrix calculation part to further speed up the entire matrix solver. We also expect to develop a solver that can solve landslide problems at high speed by extending it to the IISPH method [3], which also solves the velocity vector implicitly.

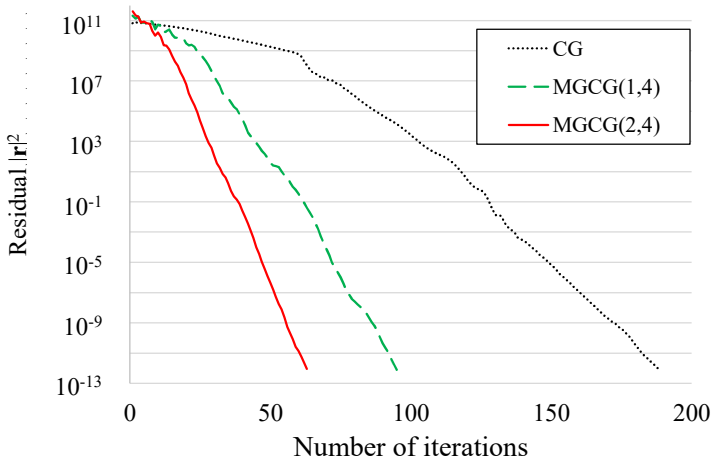

Figure 7. Convergence history in t=3.2[s] for the dam break problem

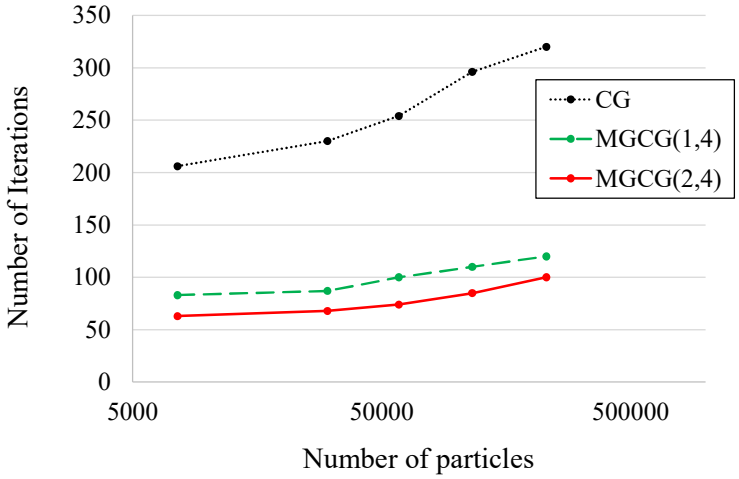

Figure 8. Number of iterations in the scaled cases (Horizontal axis is a logarithmic scale)

REFERENCES

[1] J.J. Monaghan, "Simulating free surface flows with SPH", *Journal of Computational Physics*, **110**, pp. 399-406, 1994.

[2] D.S. Morikawa, H. Senadheera and M. Asai, Explicit Incompressible Smoothed Particle Hydrodynamics in a multi-GPU environment for large scale simulations, *Computatinal Particle Mechanics*, **8(3)**, pp. 493-510, 2020.

[3] D.S. Morikawa and M. Asai, A phase-change approach to landslide simulations: Coupling finite strain elastoplastic TLSPH with non-Newtonian IISPH, *Computers and Geotechnics*, **148**, pp. 104815, 2022.

[4] U. Trottenberg, C.W. Oosterlee and A. Schuller, Multigrid, Elsevier, Amsterdam ,2000.

[5] O. Tatebe and Y. Oyanagi, Efficient implementation of the multigrid preconditioned conjugate gradient method on distributed memory machines, *Proceedings of SC'94*, pp. 194-203, 1994.

[6] T. Ichimura, K. Fujita, P. E. B. Quinay, L. Maddegedara, M. Hori, S. Tanaka, Y.Shizawa, H. Kobayashi, and K. Minami, Implicit NonlinearWave Simulation with 1.08T DOF and 0.270T

Unstructured Finite Elements to Enhance Comprehensive Earthquake Simulation, *Proceedings of SC' 15*, pp. 1–12, 2015.
[7] T. Matsunaga, K. Shibata, K. Murotani and S. Koshizuka, Solution of pressure Poisson equation in particle method using algebraic multigrid method, *Transactions of JSCES*, **2016**, 20160012, 2016. (Japanese).
[8] A. Södersten, T. Matsunaga and S. Koshizuka, Bucket-based multigrid preconditioner for solving pressure Poisson equation using a particle method, *Computers and Fluids*, **191**, 104242, 2019.
[9] M. Kondo, J. Matsumoto and T. Sawada, A scalable physically consistent particle method for high-viscous incompressible flows. *Computational Particle Mechanics*, 2023.
[10] M. Asai, AM. Aly, Y. Sonoda and Y. Sakai, A stabilized incompressible SPH method by relaxing the density invariance condition, *Int. J. for Applied Mathematics*, **2012**, Article ID 139583, 2012.
[11] NVIDIA, The API Reference Guide for Thrust. Last updated July 25, 2023. Retrieved from https://docs.nvidia.com/cuda/thrust/index.html
[12] NVIDIA, The API Reference Guide for cuSPARSE. Last updated December, 2022. Retrieved from https://docs.nvidia.com/cuda/cusparse/index.html