Computational Particle Mechanics (2024) 11:511–527
https://doi.org/10.1007/s40571-023-00636-4

A scalable physically consistent particle method for high-viscous incompressible flows

Masahiro Kondo[1] · Junichi Matsumoto[1] · Tomohiro Sawada[1]

Received: 8 February 2023 / Revised: 1 June 2023 / Accepted: 2 July 2023 / Published online: 19 July 2023
© The Author(s) 2023

Abstract
A scalable matrix solver was developed for the moving particle hydrodynamics for incompressible flows (MPH-I) method. Since the MPH-I method can calculate both incompressible and highly viscous flows while ensuring stability through physical consistency, a wide range of industrial applications is expected. However, in its implicit calculation, both the pressure and velocity must be solved simultaneously via a linear equation with a nondefinite symmetric coefficient matrix. In this study, this nondefinite linear system was converted into a symmetric positive definite (SPD) system where only the velocity is unknown. This conversion enabled us to solve the system with well-known solvers such as the conjugated gradient (CG) and conjugated residual (CR) methods. For scalability, bucket-based multigrid preconditioned CG and CR solvers were developed for the SPD system. To handle multidimensionality during preconditioning, an extended Jacobi smoother that is even applicable in a nondiagonally dominant matrix system was proposed. The numerical efficiency was confirmed via a simple high-viscosity incompressible dam break calculation, and the scalability within the presented case was confirmed. In addition, the performance under shared memory parallel computations was studied.

Keywords Particle methods · Physical consistency · High-viscosity flows · Incompressible flows · Pressure–velocity coupled approach · Multigrid method

1 Introduction

Particle methods can easily handle large deformations of free surface flows compared to mesh methods such as the finite volume method (FVM) because the motion of a continuum, i.e., a fluid and a solid, can be directly expressed by particle movement. The representative particle methods are the smoothed particle hydrodynamics (SPH) method proposed by Monaghan [1] and the moving particle semi-implicit (MPS) method proposed by Koshizuka et al. [2]. Although they have been adopted in various applications, they require empirical relaxations to obtain stable results (e.g., artificial viscosity [1], density smoothing [3–6], background pressure [7], relaxation of the pressure Poisson equation (PPE) [2, 8–11], and particle regularization [12–16]). On the other hand, the moving particle hydrodynamics (MPH) method [17–22], which inherits the concepts of the SPH and MPS methods, can conduct calculations while avoiding unphysical instability, such as particle scattering, even without empirical treatments. This is because the stability with respect to particle motion is ensured through physical consistency in the MPH method. When the discrete particle motion equations can be fit into the analytical mechanical framework [23], the system will be physically consistent. The MPH method currently has two types, i.e., MPH for weakly compressible flows (MPH-WC) [18] and MPH for incompressible flows (MPH-I) [17]. In previous studies [17–20], the MPH method was validated with various calculations, e.g., static pressure [17, 18], dam break [17, 18], Taylor Couette flow [19], high-viscosity free surface flow [19], droplet oscillation [20], liquid bridge [20], and Plateau–Rayleigh instability [20]. Although it appears straightforward owing to its simplicity, calculating the static pressure using particle methods is not easy, as the static pressure calculation is easily affected

✉ Masahiro Kondo
kondo.masahiro@aist.go.jp

Junichi Matsumoto
matsumoto-junichi@aist.go.jp

Tomohiro Sawada
tomohiro-sawada@aist.go.jp

1 National Institute of Advanced Industrial Science and Technology (AIST), Central 2, 1-1-1 Umezono, Tsukuba, Ibaraki 305-8568, Japan

by unphysical fluctuation. In fact, unrealistic results were observed with the SPH and MPS methods [17]. However, even in such cases, the physically consistent MPH method could obtain the reasonable results [17]. In addition, Other physically consistent particle methods, e.g., the elastic body models proposed by Suzuki et al. [24] and Kondo et al. [25], thin plate model by Kondo et al. [26], constraint-based incompressible SPH model by Ellero et al. [27] and Hamiltonian MPS model by Suzuki et al. [28], have also been proposed.

To simulate practically incompressible fluids such as water, a very large bulk modulus must be employed. In addition to the physical consistency, the constraint-based incompressible SPH method proposed by Ellero et al. [27] and the Hamiltonian MPS method proposed by Suzuki et al. [28] could treat the incompressibility. However, to strictly satisfy the geometric incompressible constraint, they must solve nonlinear equations in symplectic algorithms [29], i.e., RATTLE and SHAKE. In contrast, the MPH-I method [17] can practically simulate incompressible flows only by solving linear equations because it adopts a very large bulk modulus and bulk viscosity instead of directly applying incompressible constraints. Since it can treat not only incompressibility but also high viscosity, it can be applied in various industrial problems with respect to complex flows [30–32].

However, the arising linear equation in the MPH-I method [17] has a nonpositive definite coefficient matrix when the pressure and velocity are both treated as unknowns. Therefore, convergence in solving the linear equation is not assured when well-known solvers such as the conjugated gradient (CG) and conjugate residual (CR) methods [33] are adopted. In fact, the convergence is sometimes unstable in the MPH-I method [17] when the CR solver is adopted. When the coefficient matrix is symmetric positive definite (SPD), convergence is ensured using the CG and CR solvers adopting short recurrence for iteration. Therefore, the SPD linear system is favorable in terms of both calculation efficiency and stability. In the MPH-I method, a linear equation can be converted to an SPD equation, whose unknowns are only velocity, via pressure substitution [21]. Since the equations before and after the conversion are mathematically the same, the calculation will be faster without changing the results. However, scalability is not achieved because the number of iterations increases with the system size even when solving the SPD system.

For large computations, it is important to develop a numerical method whose calculation cost is linear to the problem size. To obtain such a scalable feature, matrix solvers such as the multigrid method [34–36] are needed. In particle methods, there are fewer studies adopting multigrid solvers than the finite element method (FEM) or finite volume method (FVM). Cummins and Rudman's work [37] is a pioneering study. They applied a bucket-based geometric multigrid (BMG) solver in an isolated manner to solve the pressure Poisson equation in their incompressible SPH method. In recent studies, multigrid methods were used as preconditioners in Krylov subspace methods, e.g., the CG, CR and GMRES methods. Algebraic multigrid (AMG) methods were adopted by Trask et al. [38], Chow et al. [39] and Guo et al. [40] in the incompressible SPH method and by Matsunaga et al. [41] in the MPS method. Geometric bucket-based multigrid (BMG) methods were adopted by Sodersten et al. [42] in the MPS method and by Takahashi and Lin [43] in the incompressible SPH method. In addition, Sodersten et al. [42] reported that the BMG solver was more efficient in particle methods because the AMG solver needs to be set up at every time step due to the dynamic change in connectivity. In these multigrid solvers for particle methods [37–43], only smoothers demanding diagonally dominant matrix equations, e.g., Jacobi and Gauss–Seidel smoothers, were adopted. Therefore, their applications were limited to diagonally dominant systems, which are obtained when difference-based Laplacian models are applied to the Poisson equation or the Helmholtz equation. In the finite point method (FPM), which is a meshless method, Seibold [44] and Metsch et al. [45] adopted the AMG solver. Metsche et al. [45] applied a multigrid solver not only for the simple pressure Poisson equation but also for the pressure–velocity coupled equation, where pressure and velocity are both treated implicitly. Although they successfully obtained solutions in nonsymmetric, nondiagonally dominant, and nonpositive definite systems with the combination of the GMRES method and AMG preconditioning with a Uzawa smoother [46], they noted that convergence was not assured due to the nonsymmetric matrix. In addition, they reported that the calculation time was dominated by the AMG setup time because the point cloud in FPM dynamically changes. This implies that the AMG setup time is not negligible when the connectivity dynamically changes, as in the particle methods and meshless methods.

Fortunately, the linear equation in the MPH-I method [21] can be converted and will have the SPD feature even in the pressure–velocity coupled approach. Therefore, the classic multigrid preconditioned CG solver [47] is applicable. However, the coefficient matrix is not diagonally dominant due to the complex connectivity and multidimensionality, has many nonzero elements corresponding to the neighboring particles, and has a large condition number due to incompressibility and large viscosity. To handle multidimensionality and heterogeneity [48, 49], a damped Jacobi smoother [34–36] is often applied to relax the convergence. However, it is difficult to set the damped parameter to ensure asymptotic convergence in the problem where the connectivity dynamically changes.

In this study, a scalable MPH-I method was developed. It was shown that the SPD matrix equation can be derived via pressure substitution [21] and that the SPD feature generally appeared in physically consistent systems. For the SPD matrix equation, geometric bucket-based multigrid (BMG) preconditioned CG/CR solvers were constructed, and the preconditioner was designed to satisfy the condition for theoretical convergence in a finite number of iterations. To handle multidimensionality, the Jacobi smoother was extended to the one that is applicable for the nondiagonally dominant matrix. To confirm the validity of the multigrid solver, the CG, CR, multigrid preconditioned CG (MGCG) and multigrid preconditioned CR (MGCR) solvers were compared with respect to the number of solver iterations and computation time. Specifically, high-viscosity incompressible dam break calculations were conducted with various resolutions. Furthermore, the performance in the shared memory parallel computation with CPU and GPU were also investigated.

2 Moving particle hydrodynamics for incompressible flows (MPH-I)

2.1 Governing equations and physical consistency

The governing equations in the MPH methods [17–22] are the Navier–Stokes equation with a Lagrangian description

$$\rho_0 \frac{d\mathbf{u}}{dt} = -\nabla\Psi + \mu\nabla^2\mathbf{u} + \rho\mathbf{g} \tag{1}$$

and the equation for pressure

$$\Psi = -\lambda\nabla\cdot\mathbf{u} + \kappa\frac{\rho-\rho_0}{\rho_0}, \tag{2}$$

where $\rho, \mathbf{u}, \Psi, \mu, \mathbf{g}, \lambda$ and κ are the density, velocity, pressure, shear viscosity, gravity, bulk viscosity and bulk modulus, respectively. Although these governing equations do not directly include the incompressible condition, incompressible flows can practically be expressed by setting λ and μ to sufficiently large values. The expression in Eq. (2) enables the arising matrix equation to be SPD, which is discussed later. In the MPH method, the governing equations are discretized using particle interaction models, which is conceptually the same as that in the SPH and MPS methods. Simultaneously, for physical consistency, the interaction models are to be chosen such that they can be fit into an analytical mechanical framework [23]. In this study, the normalized weight function

$$w_{ij} = w(|\mathbf{r}_{ij}|)$$
$$w(r) = \begin{cases} \frac{1}{S}\frac{1}{h^d}\left(1-\frac{r}{h}\right)^2 & (r \le h) \\ 0 & (r > h) \end{cases}$$

$$S = \int_{r<h} \frac{1}{h^d}\left(1-\frac{r}{h}\right)^2 dv \tag{3}$$

was used for discretization, where $\mathbf{r}_{\mathrm{ij}}$ is the relative position between particles i and j, h is the effective radius, i.e., the cutoff radius, and$d$ is the number of dimensions. Here, the effective radius for pressure term h_{p} and the radius for shear viscosity h_{v} are separately given, and the corresponding weight functions are denoted by w_{ij}^p and w_{ij}^v, respectively. The Navier–Stokes equation (Eq. (1)) is discretized as [20]

$$\rho_0\frac{d\mathbf{u}_i}{dt} = \sum_j(\Psi_j+\Psi_i)\mathbf{e}_{ij}w_{ij}^{p\prime} - 2\mu(d+2)\sum_j\left(\mathbf{u}_{ij}\cdot\mathbf{e}_{ij}\right)\mathbf{e}_{ij}\frac{w_{ij}^{v\prime}}{|\mathbf{r}_{ij}|} + \rho_0\mathbf{g}, \tag{4}$$

where $\mathbf{e}_{ij}$ is the unit vector in the $\mathbf{r}_{ij}$ direction, Ψ_i and $\mathbf{u}_{\mathrm{i}}$ are the pressure and velocity of particle i, and $\mathbf{u}_{ij} = \mathbf{u}_j$-$\mathbf{u}_i$ is the relative velocity between particles i and j, respectively. The right shoulder prime in $w_{ij}^{p\prime}$ and $w_{ij}^{v\prime}$ indicates the differential of the weight function

$$w'_{ij} = \frac{\partial w(|\mathbf{r}_{ij}|)}{\partial r}, \tag{5}$$

which yields negative values. On the other hand, the equation for pressure (Eq. (2)) is discretized as

$$\Psi_i = \lambda\sum_j\left(\mathbf{u}_{ij}\cdot\mathbf{e}_{ij}\right)w_{ij}^{p\prime} + \kappa(n_i - n_0), \tag{6}$$

where n_{i} is a particle number density given by the summation of the weight function $w_{\mathrm{ij}}{}^{\mathrm{p}}$ as

$$n_i = \sum_j w_{ij}^p, \tag{7}$$

and n_0 is a base value of the particle number density.

In addition, Eqs. (4) and (6) can be fit into the extended Lagrangian mechanics framework with dissipation [23]

$$\frac{d}{dt}\left(\frac{\partial\mathcal{L}}{\partial\mathbf{u}_i}\right) - \left(\frac{\partial\mathcal{L}}{\partial\mathbf{x}_i}\right) + \left(\frac{\partial\mathcal{D}}{\partial\mathbf{u}_i}\right) = \mathbf{0}, \tag{8}$$
$$\mathcal{L} = \mathcal{T} - \mathcal{V}$$

where $\mathcal{L}$, $\mathcal{T}$, $\mathcal{V}$ and $\mathcal{D}$ are the Lagrangian, kinetic energy, potential energy and Rayleigh's dissipative function, respectively. Therefore, the system in the MPH-I method is physically consistent [17, 20]. Specifically, when Lagrangian $\mathcal{L}$

and dissipative function $\mathcal{D}$ are given as

$$\mathcal{L} = \sum_i \left(\frac{1}{2} m |\mathbf{u}_i|^2 - \frac{\kappa}{2} (n_i - n_0)^2 \Delta V + m\mathbf{g} \cdot \mathbf{x}_i \right) \tag{9}$$

and

$$\mathcal{D} = \sum_i \left(\frac{\lambda}{2} \left(\sum_j \left(\mathbf{u}_{ij} \cdot \mathbf{e}_{ij} \right) w_{ij}^{p'} \right)^2 \Delta V - \sum_j \frac{\mu}{2} (d+2) \left(\mathbf{u}_{ij} \cdot \mathbf{e}_{ij} \right)^2 \frac{w_{ij}^{v'}}{|\mathbf{r}_{ij}|} \Delta V \right), \tag{10}$$

the discretized governing equations (Eqs. (4) and (6)) are derived using Eq. (8). In Eqs. (9) and (10), m and ΔV are the mass and volume of the particles, respectively, which are given as constants

$$\begin{aligned} \Delta V &= l^d \\ m &= \rho_0 \Delta V \end{aligned} \tag{11}$$

using initial particle spacing l.

Although the particle interaction models appearing in the physically consistent formulations (Eqs. (4) and (6)) are zeroth-order accurate, the calculation model, i.e., the MPH method, was validated in previous studies [17–21], for example, via the static pressure calculation [17, 18], dam break calculation [17, 18] and Taylor Couette calculation [19]. The static pressure calculation is not always easy for particle methods because it is sensitive to the unphysical fluctuation that often appears in particle methods. Even so, the MPH method could obtain reasonable results without any empirical relaxations in such a problem, where the classical particle methods, i.e., the SPH and MPS methods, fail [17]. In addition, since the MPH method is purely a multibody system defined via analytical mechanics [23], no special treatment is needed for giving boundary conditions. Specifically, the wall boundary can be easily expressed by putting fixed particles in the calculation domain, and the free surface boundary is naturally given via the vacant space where particles do not exist.

2.2 Linear matrix equation in the implicit calculation

For practically simulating incompressible flows, the bulk viscosity λ and bulk modulus κ are set very large. For setting κ, numerical stability is assured when the condition

$$\kappa \leq \frac{\lambda}{\Delta t} \tag{12}$$

is satisfied [17], where Δt is the time step width. This implies that the bulk modulus κ can be set large when the bulk viscosity λ is large. Therefore, the way to handle a large bulk viscosity λ must be considered in the incompressible calculation. In addition, to simulating high-viscosity flows, a large shear viscosity μ must be calculated stably. For stability with a large bulk viscosity λ and a large shear viscosity μ, the velocity in Eqs. (4) and (6) is implicitly treated as

$$\rho_0 \frac{\mathbf{u}_i^{k+1} - \mathbf{u}_i^k}{\Delta t} = \sum_j (\Psi_j^{k+1} + \Psi_i^{k+1}) \mathbf{e}_{ij} w_{ij}^{p'} - 2\mu(d+2) \sum_j \left(\mathbf{u}_{ij}^{k+1} \cdot \mathbf{e}_{ij} \right) \mathbf{e}_{ij} \frac{w_{ij}^{v'}}{|\mathbf{r}_{ij}|} + \rho_0 \mathbf{g} \tag{13}$$

$$\Psi_i^{k+1} = \lambda \sum_j \left(\mathbf{u}_{ij}^{k+1} \cdot \mathbf{e}_{ij} \right) w_{ij}^{p'} + \kappa (n_i - n_0), \tag{14}$$

where the upper shoulder index k attached to the velocity $\mathbf{u}_\mathrm{i}$ indicates the time step. Equations (13) and (14) form a linear matrix equation whose unknowns are the velocity $\mathbf{u}^{\mathrm{k+1}}$ and pressure $\Psi^{\mathrm{k+1}}$. By solving this matrix equation, the velocity at the next step $\mathbf{u}^{\mathrm{k+1}}$ is obtained, and by updating the particle position $\mathbf{x}$ as

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{u}_i^{k+1} \Delta t, \tag{15}$$

the particle movement can be calculated. However, since the linear system (Eqs. (13) and (14)) has a nondefinite coefficient matrix, the convergence is not assured when the well-known CG and CR solvers are applied [17]. In this study, the system is converted to a system with a symmetric positive-definite (SPD) feature by substituting Eq. (14) into the pressure $\Psi^{\mathrm{k+1}}$ in Eq. (13) as

$$\begin{aligned} &\rho_0 \frac{\mathbf{u}_i^{k+1}}{\Delta t} + 2\mu(d+2) \sum_j \left((\mathbf{u}_j^{k+1} - \mathbf{u}_i^{k+1}) \cdot \mathbf{e}_{ij} \right) \mathbf{e}_{ij} \frac{w_{ij}^{v'}}{|\mathbf{r}_{ij}|} \\ &- \sum_j \left(\lambda \sum_n \left(\mathbf{u}_n^{k+1} - \mathbf{u}_j^{k+1} \right) \cdot \mathbf{e}_{jn} w_{jn}^{p'} \right. \\ &\left. + \lambda \sum_m \left(\mathbf{u}_m^{k+1} - \mathbf{u}_i^{k+1} \right) \cdot \mathbf{e}_{im} w_{im}^{p'} \right) \mathbf{e}_{ij} w_{ij}^{p'}, \\ &= \rho_0 \frac{\mathbf{u}_i^k}{\Delta t} + \sum_j \left(\kappa (n_j - n_0) + \kappa (n_i - n_0) \right) \mathbf{e}_{ij} w_{ij}^{p'} + \rho_0 \mathbf{g} \end{aligned} \tag{16}$$

where the unknowns are only the velocity $\mathbf{u}^{\mathrm{k+1}}$ [21]. Since this conversion mathematically retains the same equation, it enables justifiably solving the system with CG and CR solvers without affecting the calculation results. The partial differential version of Eq. (16) is obtained by substituting Eq. (2) into Eq. (1) as

$$\rho_0 \frac{d\mathbf{u}}{dt} - \nabla \cdot \mu \nabla \mathbf{u} - \nabla(\lambda \nabla \cdot \mathbf{u}) = -\nabla\left(\kappa \frac{\rho - \rho_0}{\rho_0}\right) + \rho_0 \mathbf{g}. \tag{17}$$

The partial differential equation to be solved (Eq. (17)) is more complex than the Poisson equation or Helmholtz equation that were solved in previous studies [34–40], but it is a kind of diffusion equation with respect to velocity. Therefore, the discretized version is expected to be basically diagonally dominant.

To obtain the SPD feature in Eq. (16), the formulation of Eq. (2) is a key, and this approach is analogous to the penalty method in the finite element method [50] which was originally developed for structural calculations. In structural calculations, the Lagrangian is given with elastic strain energy, and the motion equation is derived by minimizing the potential energy. On the other hand, in fluid calculations with Lagrangian specifications, i.e., particle methods, the dissipative function is given, and the corresponding force in the motion equation is derived by minimizing the function. In structural calculations, the incompressible constraint can practically be posed using a very large bulk modulus. This is the penalty method with which the SPD feature can be obtained. Analogically, in the particle methods, the practically incompressible calculation can be conducted with very large bulk viscosity, λ in Eq. (2), and similarly, the SPD feature can be maintained with this approach.

Moreover, the SPD feature appeared in Eq. (16) generally arises in physically consistent systems. In the extended Lagrangian mechanics with dissipation (Eq. (8)), Rayleigh's dissipative function must be positive definite. This allows the dissipative function to be expressed as

$$\mathcal{D} = \frac{1}{2}\{\mathbf{u}, \mathbf{C}\mathbf{u}\} \tag{18}$$

using an SPD matrix $\mathbf{C}$, where the vector $\mathbf{u}$ without a lower index indicates a large vector unifying the velocity of all particles, and the bracket $\{,\}$ indicates the dot product of the unified vectors. Using the matrix $\mathbf{C}$, the motion equation of the particles is expressed as

$$m\frac{d\mathbf{u}}{dt} = \frac{d\mathcal{L}}{d\mathbf{x}} - \mathbf{C}\mathbf{u}. \tag{19}$$

When the velocity is treated implicitly,

$$\left(\frac{m}{\Delta t}\mathbf{I} + \mathbf{C}\right)\mathbf{u}^{k+1} = m\frac{\mathbf{u}^k}{\Delta t} + \frac{d\mathcal{L}}{d\mathbf{x}} \tag{20}$$

is obtained. Since the coefficient matrix appears on the left-hand side of Eq. (20) is SPD, it is proven that the SPD feature will arise in arbitrary physically consistent systems that can be fit into the analytical mechanical framework of Eq. (8). Therefore, it is interpreted that the SPD feature in Eq. (16) emerged owing to the physical consistency in the MPH-I method.

3 Multigrid preconditioned CG/CR method

3.1 Generalized CG/CR algorithm

The CG and CR solvers, where the convergence is assured with short recursive iterations whose number is smaller than the degree of freedom, can be generalized as follows. Here, let the weight matrix $\mathbf{M}$ and the preconditioning matrix $\mathbf{K}$. Using the initial solution $\mathbf{x}_0$, the residual $\mathbf{r}$ and search direction $\mathbf{p}$ are initially given as

$$\begin{aligned}\mathbf{r}_0 &= \mathbf{b} - \mathbf{A}\mathbf{x}_0\\ \mathbf{p}_0 &= \mathbf{K}\mathbf{r}_0,\end{aligned} \tag{21}$$

where the lower index indicates the iteration. Then, the solution $\mathbf{x}$ and residual $\mathbf{r}$ are updated

$$\begin{aligned}\mathbf{x}_{k+1} &= \mathbf{x}_k + \alpha_k \mathbf{p}_k\\ \mathbf{r}_{k+1} &= \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k\end{aligned} \tag{22}$$

with a parameter α_k

$$\alpha_k = \frac{\{\mathbf{r}_k, \mathbf{M}\mathbf{A}\mathbf{p}_k\}}{\{\mathbf{A}\mathbf{p}_k, \mathbf{M}\mathbf{A}\mathbf{p}_k\}}, \tag{23}$$

which is determined such that

$$\{\mathbf{r}_{k+1}, \mathbf{M}\mathbf{A}\mathbf{p}_k\} = 0, \tag{24}$$

Here, the bracket $\{\mathbf{a},\mathbf{b}\}$ indicates the dot product of unified vectors $\mathbf{a}$ and $\mathbf{b}$. Then, the conjugate vector $\mathbf{p}$ is updated as

$$\mathbf{p}_{k+1} = \mathbf{K}\mathbf{r}_{k+1} + \beta_k \mathbf{p}_k \tag{25}$$

with a parameter β_k

$$\beta_k = -\frac{\{\mathbf{A}\mathbf{K}\mathbf{r}_{k+1}, \mathbf{M}\mathbf{A}\mathbf{p}_k\}}{\{\mathbf{A}\mathbf{p}_k, \mathbf{M}\mathbf{A}\mathbf{p}_k\}}, \tag{26}$$

which is determined such that

$$\{\mathbf{A}\mathbf{p}_{k+1}, \mathbf{M}\mathbf{A}\mathbf{p}_k\} = 0. \tag{27}$$

The iteration given by Eqs. (22)–(27) is repeated until the L2 norm of the residual $|\mathbf{r}_k|^2$ becomes small enough, i.e., the convergence. When the matrices **M** and **MAK** are symmetric, the orthogonalities

$$\{\mathbf{r}_i, \mathbf{MAp}_j\} = 0 \quad (j < i) \tag{28}$$

$$\{\mathbf{r}_i, \mathbf{MAKr}_j\} = 0 \quad (j \neq i) \tag{29}$$

$$\{\mathbf{Ap}_i, \mathbf{MAp}_j\} = 0 \quad (j \neq i) \tag{30}$$

are provided (see "Appendix A"). Using Eqs. (28)–(30), α_k and β_k are rewritten as

$$\alpha_k = \frac{\{\mathbf{r}_k, \mathbf{MAp}_k\}}{\{\mathbf{MAp}_k, \mathbf{Ap}_k\}} = \frac{\{\mathbf{r}_k, \mathbf{MAKr}_k\}}{\{\mathbf{Ap}_k, \mathbf{MAp}_k\}} \tag{31}$$

and

$$\beta_k = -\frac{\{\mathbf{MAKr}_{k+1}, \mathbf{Ap}_k\}}{\{\mathbf{MAp}_k, \mathbf{Ap}_k\}} = \frac{\{\mathbf{r}_{k+1}, \mathbf{MAKr}_{k+1}\}}{\{\mathbf{r}_k, \mathbf{MAKr}_k)\}}, \tag{32}$$

which are often used in implementation. When the matrices **M** and **MAK** are symmetric positive definite (SPD), theoretical convergence is assured (see "Appendix A"). This generalized algorithm (Eqs. (21)–(27)) can generate specific algorithms such as the CG and CR methods. For example, the CG method is derived from $\mathbf{M} = \mathbf{A}^{-1}$ and $\mathbf{K} = \mathbf{I}$, the CR method is from $\mathbf{M} = \mathbf{I}$ and $\mathbf{K} = \mathbf{I}$, the CGNR method [33] is from $\mathbf{M} = \mathbf{I}$ and $\mathbf{K} = \mathbf{A}^{\mathrm{T}}$, and the CGNE method [33] is from $\mathbf{M} = (\mathbf{AA}^{\mathrm{T}})^{-1}$ and $\mathbf{K} = \mathbf{A}^{\mathrm{T}}$. Furthermore, the preconditioned CG method is equivalent to the case with $\mathbf{M} = \mathbf{A}^{-1}$ and $\mathbf{K} = \mathbf{K}_{\mathrm{CG}}$, and the preconditioned CR method is the case with $\mathbf{M} = \mathbf{K} = \mathbf{K}_{\mathrm{CR}}$. Therefore, theoretical convergence is obtained when $\mathbf{K}_{\mathrm{CG}}$ and $\mathbf{K}_{\mathrm{CR}}$ are SPD because **M** and **MAK** will be SPD in such cases.

It is noteworthy that the residual in the CG and preconditioned CG methods is weighted by matrix $\mathbf{A}^{-1}$. In fact, the weight matrix is specified as $\mathbf{M} = \mathbf{A}^{-1}$ when deriving the CG method. Consequently, the solution in the CG iteration is updated such that $\{\mathbf{r}, \mathbf{A}^{-1}\mathbf{r}\}$ is minimized using the search direction **p**. This unintended weight may affect the convergence. In fact, the convergence degrades, especially when the condition number of **A** is large. On the other hand, in the CR and preconditioned CR methods, the objective functions to be minimized are straightforwardly expressed as $\{\mathbf{r}, \mathbf{r}\}$ and $\{\mathbf{r}, \mathbf{K}_{\mathrm{CR}}\mathbf{r}\}$, respectively, which results in good convergence properties, such as a monotonic decrease in the residual.

3.2 Bucket-based geometric multigrid preconditioner

In this study, background bucket cells are utilized for constructing a geometric multigrid preconditioner for the CG and CR methods. The algorithm of the multigrid preconditioned conjugated residual (MGCR) solver is shown in Fig. 1. To construct the multigrid structure, the bucket size is set the same as the effective radius such that the range of the interaction is limited to the next buckets. The linear equation in the MPH-I method (Eq. (16)) has multidimensionality because the unknowns are the velocities of particles. To restrict the multidimensional vector of the particles to the finest grid, i.e., buckets, the vectors of particles in the buckets are simply summed as

$$\mathbf{u}_l^0 = \sum_{i \in l} \mathbf{u}_i, \tag{33}$$

where the i and l on the lower right of **u** show the indices of the particles and the buckets, respectively, and the upper index 0 of $\mathbf{u}_{\mathrm{l}}$ indicates that the parameter belongs to the finest grid (level 0 in Fig. 1). Here, the restriction matrix corresponding to Eq. (33) is denoted by **R**, and the prolongation matrix from the buckets to the particles **P** is given by its transpose as

$$\mathbf{P} = \mathbf{R}^T. \tag{34}$$

Then, the coefficient matrix in the finest grid scale $\mathbf{A}^0$ is defined as

$$\mathbf{A}^0 = \mathbf{RAP}, \tag{35}$$

where**A** is the coefficient matrix at the original particle scale, which is expressed in Eq. (16). Furthermore, the coarser grids are recursively created from the finer grids. In this study, the size of the coarser grids is double that of the finer grids. Specifically, the coarser grid consists of 4 finer grids in 2D and 8 finer grids in 3D. The level of the grid is incremented as the grid size is doubled (level 0: grid size = h_v, level 1: grid size = $2h_v$, level 2: grid size = $4h_v$, and so on). The restriction from the finer grid (level r) to the coarser grid (level $r + 1$) is simply given by the summation

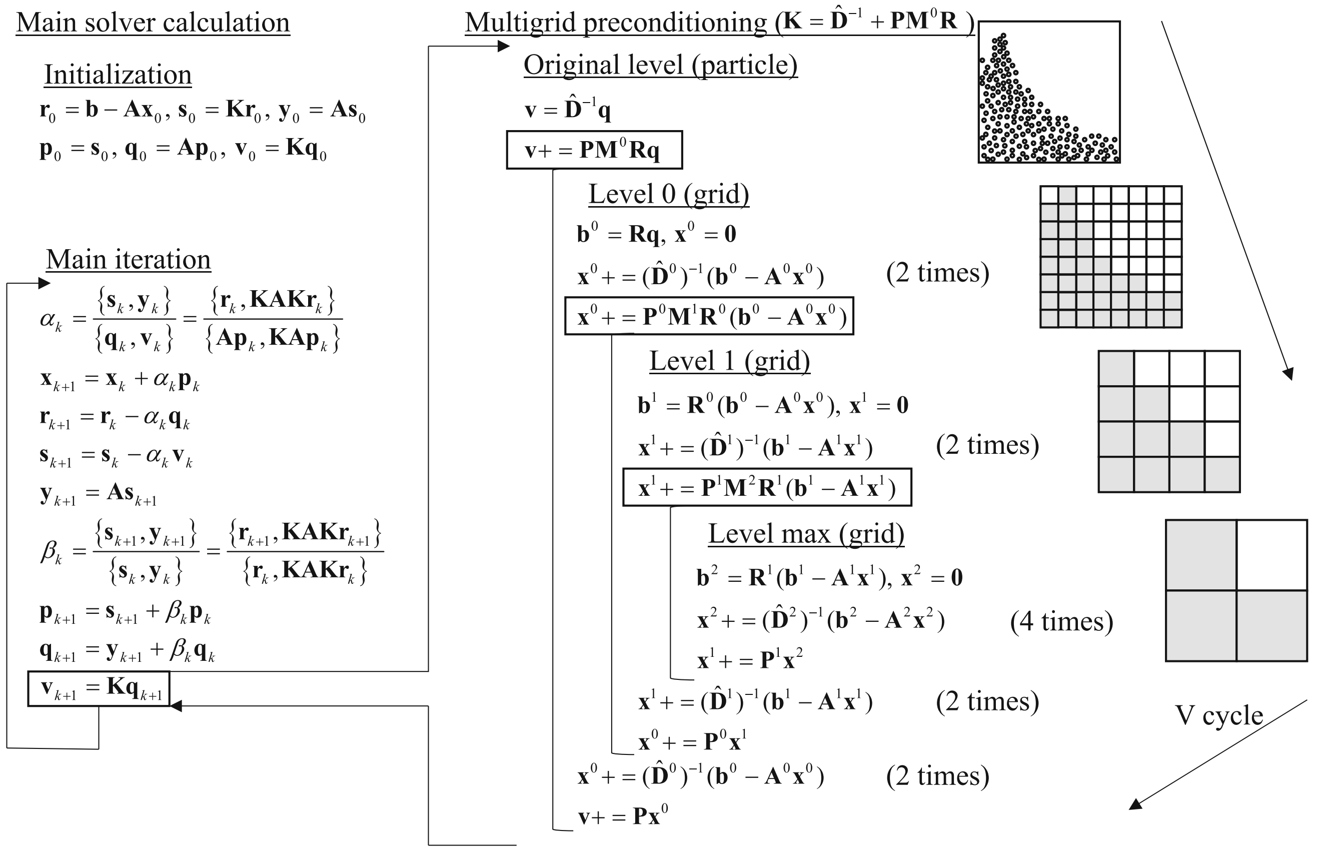
$$\mathbf{u}_l^{r+1} = \sum_{s \in l} \mathbf{u}_s^r, \tag{36}$$

where l and s on the lower right of **u** are the indices of the coarser and finer grids, respectively. Using the restriction matrix $\mathbf{R}^{\mathrm{r}}$, which corresponds to Eq. (36), the prolongation matrix from level r + 1 to level r is given by

$$\mathbf{P}^r = \mathbf{R}^{rT}. \tag{37}$$

Then, the coefficient matrix $\mathbf{A}^{\mathrm{r}}$ is recursively provided as

$$\mathbf{A}^{r+1} = \mathbf{R}^r\mathbf{A}^r\mathbf{P}^r. \tag{38}$$



```
Main solver calculation
  Initialization
    $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0,\ \mathbf{s}_0 = \mathbf{K}\mathbf{r}_0,\ \mathbf{y}_0 = \mathbf{A}\mathbf{s}_0$
    $\mathbf{p}_0 = \mathbf{s}_0,\ \mathbf{q}_0 = \mathbf{A}\mathbf{p}_0,\ \mathbf{v}_0 = \mathbf{K}\mathbf{q}_0$
  Main iteration
    $\alpha_k = \frac{\{\mathbf{s}_k, \mathbf{y}_k\}}{\{\mathbf{q}_k, \mathbf{v}_k\}} = \frac{\{\mathbf{r}_k, \mathbf{KAKr}_k\}}{\{\mathbf{Ap}_k, \mathbf{KAp}_k\}}$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
    $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k$
    $\mathbf{s}_{k+1} = \mathbf{s}_k - \alpha_k \mathbf{v}_k$
    $\mathbf{y}_{k+1} = \mathbf{A}\mathbf{s}_{k+1}$
    $\beta_k = \frac{\{\mathbf{s}_{k+1}, \mathbf{y}_{k+1}\}}{\{\mathbf{s}_k, \mathbf{y}_k\}} = \frac{\{\mathbf{r}_{k+1}, \mathbf{KAKr}_{k+1}\}}{\{\mathbf{r}_k, \mathbf{KAKr}_k\}}$
    $\mathbf{p}_{k+1} = \mathbf{s}_{k+1} + \beta_k \mathbf{p}_k$
    $\mathbf{q}_{k+1} = \mathbf{y}_{k+1} + \beta_k \mathbf{q}_k$
    $\boxed{\mathbf{v}_{k+1} = \mathbf{K}\mathbf{q}_{k+1}}$

Multigrid preconditioning ($\mathbf{K} = \hat{\mathbf{D}}^{-1} + \mathbf{PM}^0\mathbf{R}$)
  Original level (particle)
    $\mathbf{v} = \hat{\mathbf{D}}^{-1}\mathbf{q}$
    $\boxed{\mathbf{v} += \mathbf{PM}^0\mathbf{Rq}}$
      Level 0 (grid)
        $\mathbf{b}^0 = \mathbf{Rq},\ \mathbf{x}^0 = \mathbf{0}$
        $\mathbf{x}^0 += (\hat{\mathbf{D}}^0)^{-1}(\mathbf{b}^0 - \mathbf{A}^0\mathbf{x}^0)$   (2 times)
        $\boxed{\mathbf{x}^0 += \mathbf{P}^0\mathbf{M}^1\mathbf{R}^0(\mathbf{b}^0 - \mathbf{A}^0\mathbf{x}^0)}$
          Level 1 (grid)
            $\mathbf{b}^1 = \mathbf{R}^0(\mathbf{b}^0 - \mathbf{A}^0\mathbf{x}^0),\ \mathbf{x}^1 = \mathbf{0}$
            $\mathbf{x}^1 += (\hat{\mathbf{D}}^1)^{-1}(\mathbf{b}^1 - \mathbf{A}^1\mathbf{x}^1)$   (2 times)
            $\boxed{\mathbf{x}^1 += \mathbf{P}^1\mathbf{M}^2\mathbf{R}^1(\mathbf{b}^1 - \mathbf{A}^1\mathbf{x}^1)}$
              Level max (grid)
                $\mathbf{b}^2 = \mathbf{R}^1(\mathbf{b}^1 - \mathbf{A}^1\mathbf{x}^1),\ \mathbf{x}^2 = \mathbf{0}$
                $\mathbf{x}^2 += (\hat{\mathbf{D}}^2)^{-1}(\mathbf{b}^2 - \mathbf{A}^2\mathbf{x}^2)$   (4 times)
                $\mathbf{x}^1 += \mathbf{P}^1\mathbf{x}^2$
            $\mathbf{x}^1 += (\hat{\mathbf{D}}^1)^{-1}(\mathbf{b}^1 - \mathbf{A}^1\mathbf{x}^1)$   (2 times)
            $\mathbf{x}^0 += \mathbf{P}^0\mathbf{x}^1$
        $\mathbf{x}^0 += (\hat{\mathbf{D}}^0)^{-1}(\mathbf{b}^0 - \mathbf{A}^0\mathbf{x}^0)$   (2 times)
        $\mathbf{v} += \mathbf{P}\mathbf{x}^0$
```

Fig. 1 Multigrid preconditioned conjugated residual algorithm

Using the geometric multigrid expressed by Eqs. (33)–(38), a preconditioner for the CG and CR methods satisfying the SPD condition is constructed. However, the linear system to be solved in the MPH-I method is not diagonally dominant even at the coarse grid scale because of multidimensionality. Therefore, the widely used smoothers, e.g., the Jacobi and Gauss–Seidel smoothers, cannot directly be applied because they demand diagonal dominancy. To address this issue, the Jacobi smoother is extended to be applicable even for nondiagonally dominant systems. In solving the linear equation

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{39}$$

the Jacobi iteration is expressed as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{D}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_k), \tag{40}$$

where $\mathbf{D}$ is the diagonal component of the coefficient matrix $\mathbf{A}$. In the extended Jacobi iteration, the diagonal matrix $\mathbf{D}$ is replaced by another diagonal matrix $\hat{\mathbf{D}}$, whose elements satisfy

$$\hat{D}_{ii} > \frac{1}{2}\sum_j |A_{ij}|. \tag{41}$$

Then, the iteration is given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \hat{\mathbf{D}}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}_k). \tag{42}$$

With this simple extension, asymptotic convergence will be obtained even in a nondiagonally dominant system (see "Appendix B"). In this study, the right-hand side of Eq. (41) was simply adopted for calculating the elements $\hat{\mathbf{D}}$ because it works when $2\hat{\mathbf{D}} - \mathbf{A}$ is not singular, which is satisfied in most cases. Since the matrix equation in the MPH-I method is a discretized version of the diffusion equation, it is close to diagonally dominant. In such cases, this extension is useful.

In this study, the V cycle multigrid calculation was included as a preconditioner of the CG and CR methods (Fig. 1), and the extended Jacobi iteration was adopted as a smoother in each level. It is better for the preconditioner to skip smoothing at the original particle level, where matrix–vector multiplication requires a large computational cost. This is because the CG and CR solvers already have the main iterations in the original level, which are more efficient than the smoothing iterations in the multigrid calculation, i.e., the extended Jacobi iterations. Therefore, in this study, the preconditioning matrix $\mathbf{K}$ was designed not to include particle level smoothing as

$$\mathbf{K} = \hat{\mathbf{D}}^{-1} + \mathbf{PM}^0\mathbf{R}, \tag{43}$$

where $\hat{\mathbf{D}}$ is the extended diagonal matrix defined in Eq. (41) corresponding to the coefficient matrix $\mathbf{A}$ in Eq. (16), and $\mathbf{M}^0$ is the matrix corresponding to the calculation at level 0. When the extended Jacobi iterations are conducted twice in both pre- and postsmoothing in each level (Fig. 1), the recursive relation between the matrices $\mathbf{M}^{\mathrm{r}}$ and $\mathbf{M}^{\mathrm{r+1}}$

$$\begin{aligned}\mathbf{M}^r &= (\mathbf{A}^r)^{-1} - \left(\mathbf{I}^r - (\hat{\mathbf{D}}^r)^{-1}\mathbf{A}^r\right)^2 (\mathbf{A}^r)^{-1}\left(\mathbf{I}^r - \mathbf{A}^r(\hat{\mathbf{D}}^r)^{-1}\right)^2 \\ &\quad + \left(\mathbf{I}^r - (\hat{\mathbf{D}}^r)^{-1}\mathbf{A}^r\right)^2 \mathbf{P}^r\mathbf{M}^{r+1}\mathbf{R}^r\left(\mathbf{I}^r - \mathbf{A}^r(\hat{\mathbf{D}}^r)^{-1}\right)^2 \end{aligned} \tag{44}$$

holds, where the upper right index of $\mathbf{M}$ indicates the level (see "Appendix C"). In the maximum level, the extended Jacobi iterations are conducted 4 times, and the matrix $\mathbf{M}^{\max}$ is expressed as

$$\begin{aligned}\mathbf{M}^{\max} &= (\mathbf{A}^{\max})^{-1} - \left(\mathbf{I}^{\max} - (\hat{\mathbf{D}}^{\max})^{-1}\mathbf{A}^{\max}\right)^2 (\mathbf{A}^{\max})^{-1} \\ &\quad \left(\mathbf{I}^{\max} - \mathbf{A}^{\max}(\hat{\mathbf{D}}^{\max})^{-1}\right)^2. \end{aligned} \tag{45}$$

Here, the matrix $\mathbf{M}^{\max}$ is SPD (see "Appendix B"). In the same way, the sum of the first and second terms on the right-hand side of Eq. (44) is SPD. When $\mathbf{M}^{\mathrm{r+1}}$ is SPD, the third term in Eq. (44) is symmetric nonnegative definite. Therefore, $\mathbf{M}^{\mathrm{r}}$ is recursively SPD, and $\mathbf{K}$ is also SPD. Therefore, $\mathbf{K}$ satisfies the condition to be a preconditioner for the CG and CR solvers. Note that $\mathbf{PM}^0\mathbf{R}$ in the second term on the right-hand side of Eq. (43) cannot solely be a preconditioner because it is nonnegative definite but singular. Therefore, it is combined with $\hat{\mathbf{D}}^{-1}$ to construct a preconditioner.

4 Benchmark calculations

4.1 Number of iterations

The presented geometric bucket-based multigrid preconditioner was implemented in the MPH-I methods, where open-source code [22] was used. The calculations were conducted using the multigrid preconditioned conjugated residual (MGCR), multigrid preconditioned conjugated gradient (MGCG), nonpreconditioned conjugated residual (CR) and nonpreconditioned conjugated gradient (CG) solvers. Specifically, the simple high-viscosity incompressible dam

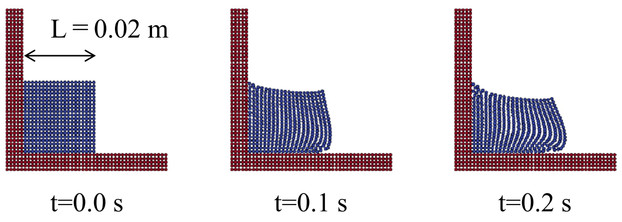


Fig. 2 Dam-break calculation of a highly viscous incompressible flow

Table 1 Calculation conditions for the base case

	Base case
Particle spacing (m) l	$1/1 \times 10^{-3}$
Time step width (s) Δt	$1/1 \times 10^{-3}$
Gravity (m/s^2) g	10
Shear viscosity (Pas) μ	10
Bulk viscosity (Pas) λ	104
Bulk modulus (Pa) κ	106
Density (kg/m^3) ρ_0	103
Radius ratio (h_v/l)	1.75
Radius ratio (h_v/l)	3.5
Number of particles	400

break shown in Fig. 2 was calculated for a phenomenon time of 0.2 s. The calculation condition for the base case is shown in Table 1, and the scaled cases, whose particle spacing l and time step width Δt are 1/2, 1/4, 1/8,…,1/64 times the base case, are shown in Table 2. In Table 2, only the difference from the base case (Table 1) is shown. In addition, the diffusion numbers d_λ and d_μ, degrees of freedom $\mathrm{DoF}_{\mathbf{A}}$, approximated number of nonzero elements $\mathrm{nnz}_{\mathbf{A}}$ and approximated condition numbers $\mathrm{K}_{\mathbf{A}}$ are displayed in Table 2. The $\mathrm{DoF}_{\mathbf{A}}$ was calculated as a product of the number of fluid particles and the number of dimensions. The $\mathrm{nnz}_{\mathrm{A}}$ was estimated as

$$\mathrm{nnz}_{\mathbf{A}} \approx \begin{cases} \mathrm{DoF}_{\mathbf{A}} \times 2\pi(h_v/l)^2 \ (d=2) \\ \mathrm{DoF}_{\mathbf{A}} \times 4\pi(h_v/l)^3 \ (d=3) \end{cases} \tag{46}$$

using the approximated number of neighboring particles, which is $\pi(h_v/l)^2$ in 2D and $4/3\pi(h_v/l)^3$ in 3D. For the rough estimation of $\mathrm{K}_{\mathbf{A}}$, maximum and minimum eigen values $\Lambda_{\max}$ and $\Lambda_{\min}$ were predicted using

$$f(u) = \frac{\rho_0}{\Delta t}u - (\mu + \lambda)\frac{d^2u}{dx^2}, \tag{47}$$

which is the 1D version of the left-hand side of Eq. (17). In estimating $\Lambda_{\max}$, it was assumed that the discretization in the MPH-I method is analogous to the finite difference

Table 2 Calculation conditions for the scaled cases with various resolutions

	Case x1 (base case)	Case x1/2	Case x1/4	Case x1/8	Case x1/16	Case x1/32	Case x1/64
Particle spacing (m) l	1×10^{-3}	$1/2 \times 10^{-3}$	$1/4 \times 10^{-3}$	$1/8 \times 10^{-3}$	$1/16 \times 10^{-3}$	$1/32 \times 10^{-3}$	$1/64 \times 10^{-3}$
Time step width (s) Δt	1×10^{-3}	$1/2 \times 10^{-3}$	$1/4 \times 10^{-3}$	$1/8 \times 10^{-3}$	$1/16 \times 10^{-3}$	$1/32 \times 10^{-3}$	$1/64 \times 10^{-3}$
Number of particles	400	1600	6400	25,600	102,400	409,600	1,638,400
$d_\mu = \mu \Delta t / \rho_0 l^2$	1×10	2×10	4×10	8×10	16×10	32×10	64×10
$d_\lambda = \mu \Delta t / \rho_0 l^2$	1×10^3	2×10^3	4×10^3	8×10^3	16×10^3	32×10^3	64×10^3
DoF	800	3200	12,800	51,200	204,800	819,200	3,276,800
Nonzero count nnz (approx.)	6.16×10^4	2.46×10^5	9.85×10^5	3.94×10^6	1.58×10^7	6.31×10^7	2.52×10^8
Condition number K_A (approx.)	5.21×10^1	2.05×10^2	7.95×10^2	3.00×10^3	1.08×10^4	3.57×10^4	1.06×10^5
Maximum level in multigrid calculation	2	3	4	5	6	7	8

discretization with a mesh size of $h_v (= 2h_p)$. Equation (47) was discretized as

$$f(u_l) \approx \frac{\rho_0}{\Delta t} u_l - (\mu + \lambda) \frac{u_{l+1} - 2u_l + u_{l-1}}{h_v^2}, \tag{48}$$

and the maximum eigen value $\Lambda_{\max}$ was approximated as

$$\Lambda_{\max} \approx \frac{\rho}{\Delta t} + (\mu + \lambda) \frac{4}{h_v^2}. \tag{49}$$

On the other hand, for predicting $\Lambda_{\min}$, the maximum wavelength $4L$ which was determined from the calculation geometry (Fig. 2), was focused on. By substituting a sine wave with a wavelength of $4L$

$$u(x) = \sin \frac{\pi x}{2L} \tag{50}$$

into Eq. (47),

$$f(u) = \left[\frac{\rho_0}{\Delta t} + (\mu + \lambda) \left(\frac{\pi}{2L} \right)^2 \right] u(x) \tag{51}$$

was obtained. Then, the minimum eigen value $\Lambda_{\min}$ was approximated as

$$\Lambda_{\min} \approx \frac{\rho_0}{\Delta t} + (\mu + \lambda) \left(\frac{\pi}{2L} \right)^2, \tag{52}$$

and the condition number was predicted as

$$K_A = \Lambda_{\max} / \Lambda_{\min}. \tag{53}$$

The number of solver iterations at t = 0.2 s with respect to the scaled cases (Table 2) is presented in Fig. 3. Here, only

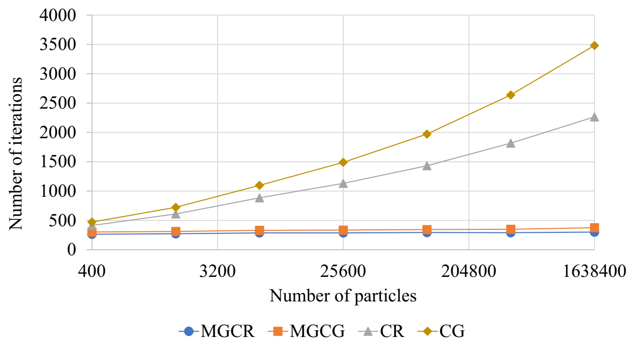


Fig. 3 Number of iterations in the scaled cases (Table 2)

the main iterations (Fig. 1) are counted with the convergence threshold of $|\mathbf{r}|^2/|\mathbf{b}|^2 < 10^{-12}$. When the CR and CG solvers were adopted, the number of iterations drastically increased with the problem size. On the other hand, with the MGCR and MGCG solvers, the increase in the number of iterations was limited. Additionally, the number of iterations was smaller with the MGCR and CR solvers than with the MGCG and CG solvers, respectively. The objective function in the MGCG and CG was unintendedly weighted by $\mathbf{A}^{-1}$ as $\{\mathbf{r}, \mathbf{A}^{-1}\mathbf{r}\}$, and it possibly affected the convergence.

The difficulty in solving the scaled cases shown in Table 2 is not only because of the problem size but also because of the large diffusion numbers. Although the Courant numbers in the scaled cases are the same as those in the base case, the diffusion numbers d_λ and d_μ are larger than those in the base case. For the comparison, the small cases shown in Table 3 were studied, and the problem size was set the same as that of the base case, but the diffusion numbers varied to follow each scaled case in Table 2. In Table 3, only the conditions that are different from those of the base case (Table 1) are

Table 3 Calculation conditions for the small cases with various diffusion numbers

	d_μ,d_λ x1 (base case)	d_μ,d_λ x2	d_μ,d_λ x4	d_μ,d_λ x8	d_μ,d_λ x16	d_μ,d_λ x32	d_μ,d_λ x64
Gravity (m/s2) g	1×10	2×10	4×10	8×10	16×10	32×10	64×10
Shear viscosity (Pas) μ	1×10	2×10	4×10	8×10	16×10	32×10	64×10
Bulk viscosity (Pas) λ	1×10^4	2×10^4	4×10^4	8×10^4	16×10^4	32×10^4	64×10^4
$d_\mu = \mu \Delta t/\rho_0 l^2$	1×10	2×10	4×10	8×10	16×10	32×10	64×10
$d_\lambda = \mu \Delta t/\rho_0 l^2$	1×10^3	2×10^3	4×10^3	8×10^3	16×10^3	32×10^3	64×10^3

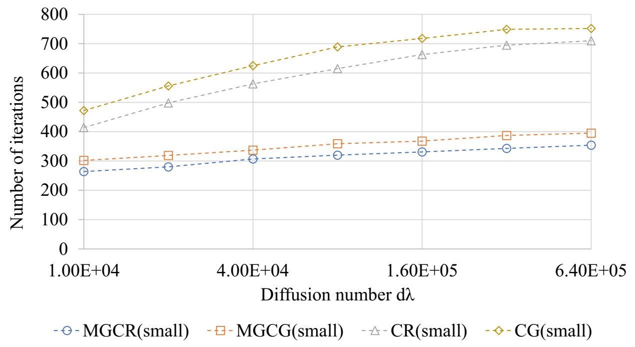


Fig. 4 Number of iterations in the small cases with various diffusion numbers (Table 3)

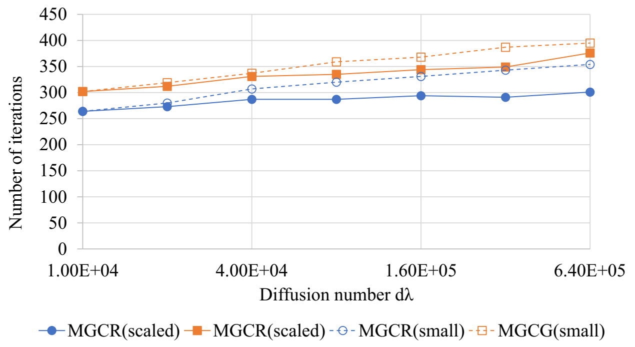


Fig. 5 Comparison between the scaled cases (Table 2) and small cases (Table 3)

presented. By setting the gravity g and viscosities μ and λ, which were 2, 4, 8,…64 times larger those of the base case, only the diffusion numbers d_λ and d_μ were enlarged, while almost the same flow as that of the base case was maintained. The number of solver iterations at t = 0.2 s with respect to the small cases (Table 3) is presented in Fig. 4. The number of iterations is smaller with the MGCR and MGCG solvers than with the CR and CG solvers, respectively. This implies that multigrid preconditioning also suppresses the number of iterations associated with large diffusion numbers. The scaled cases (Table 2) and the small cases with the same diffusion numbers (Table 3) are compared in Fig. 5. When the problem size is large, the multigrid preconditioner contribution is large, and the number of iterations in the scaled cases is kept smaller than that in the corresponding small cases. In addition, it is confirmed that the MGCR solver shows better scalability than the MGCG solver. Compared to the previous studies [38, 42, 43], the numbers of iterations in this study (Figs. 3 and 4) are relatively large. This is because the linear matrix equation in this study (Eq. (16)) is difficult to be solve due to the nondiagonally dominancy, large number of nonzero elements and large condition number (Table 2).

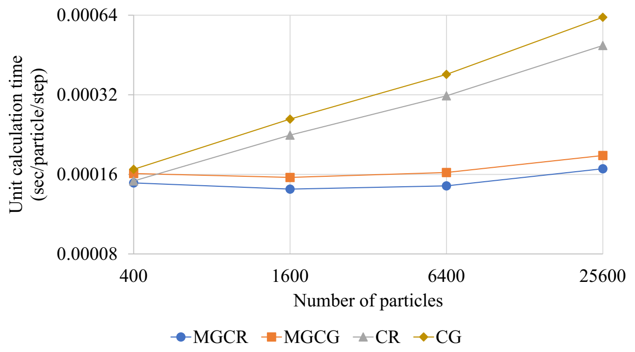


Fig. 6 Calculation time of a single thread computation with a CPU

4.2 Single CPU calculation

The calculation times for a single thread CPU (Xeon Gold 6252 (24 core)) computation are shown in Fig. 6, where the results for Case x1~x1/8 in Table 2 are presented. Hereinafter, the calculation times are shown with dividing by the number of particles and number of time steps for comparison, and such calculati on times are referred to “unit calculation

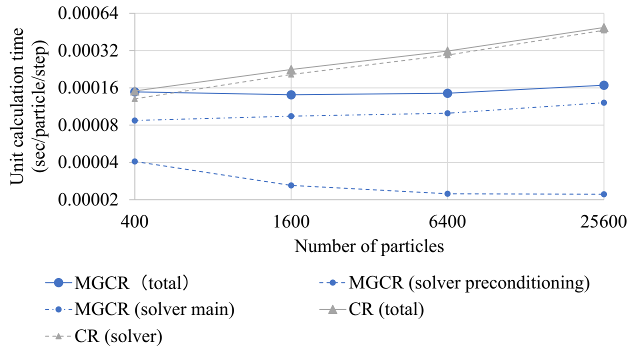


Fig. 7 Breakdown of the calculation time of a single thread computation with a CPU

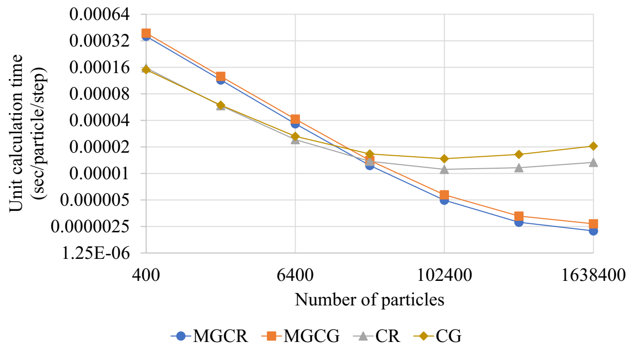


Fig. 8 Calculation time of a parallel computation with OpenACC and a GPU

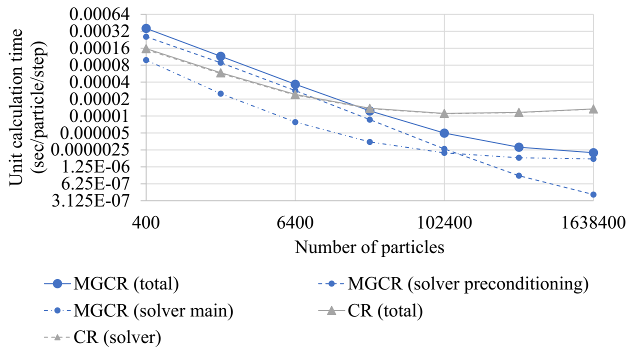


Fig. 9 Breakdown of the calculation time of a parallel computation with OpenACC and a GPU

times”. The unit calculation times using the CR and CG methods become larger as the problem size becomes larger. On the other hand, when using the MGCR and MGCG methods, the unit calculation time is almost constant even for large problems. This implies that the computational time is proportional to the problem size and that the multigrid methods are scalable. The breakdowns of the unit calculation times with the MGCR and CR solvers are shown in Fig. 7. For each calculation, the whole computational time is labeled “total”. For the MGCR solver, the time spent for preconditioning is labeled “solver preconditioning”, and the other time spent by the solver, which is mostly for the main iteration, is labeled “solver main”. For the CR solver, the time spent by the solver is labeled “solver” because there is no preconditioning. Most of the calculation time was spent by the solvers. In the single CPU calculation, the preconditioning time was not dominant in all the time spent by the MGCR solver. In the preconditioning stage, the most computationally expensive matrix–vector product calculations in the original particle level are avoided as in Eq. (43), and the V-cycle only includes the product calculations in the coarser level. Therefore, the amount of computation required for the preconditioning is basically smaller than that for the main iteration.

4.3 Parallel CPU and GPU calculations

The calculation times of the OpenACC parallel computation on the GPU (A100 (80 GB)) are shown in Fig. 8, where the results for Cases x1~x1/64 in Table 2 are presented. In the relatively small cases with 400–6400 particles (Cases x1 ~x1/4), the CR and CG methods were faster than the MGCR and MGCG methods, but in the larger cases with over 25,600 particles (Cases x1/16 ~), the multigrid methods were faster. The breakdowns of the calculation times with the MGCR and CR methods are shown in Fig. 9, where the legends are the same as in Fig. 7. With both methods, the time spent by the solvers dominated the total calculation time. Using the CR solver, the unit calculation time against the problem size (CR (total)) decreased once in the range of 400–102,400 particles (Cases x1 ~x1/16) and increased again in the range over 102,400 particles (Cases x1/16 ~). When the problem size was large, the number of iterations dominated the computation time, and the increasing trend in the large cases reflects the large number of iterations in the CR solver. When the problem size was small, the overhead cost for parallelization dominated the computation time. The straight decreasing trends in the small cases were due to the overhead cost. Since the overhead cost can be assumed to be almost constant, the unit calculation time will be close to inversely proportional to the problem size when the overhead cost is dominant. With the MGCR solver, the unit calculation time (MGCR (total)) linearly decreased in the range of 400–102,400 particles (Cases x1 ~ x1/16), and the decrease slowed down in the range over 102,400 particles (Cases x1/16 ~). This indicates that the overhead cost was dominant with 400–102,400 particles (Cases x1 ~ x1/16). According to the breakdown of the total computational time with the MGCR solver, the preconditioning time was larger with 400 –102,400 particles (Case x1 ~ 1/16), and the main iteration time was larger with more than 102,400 particles (Case x1/32 ~). The unit calculation time with respect to the main iteration (MGCR

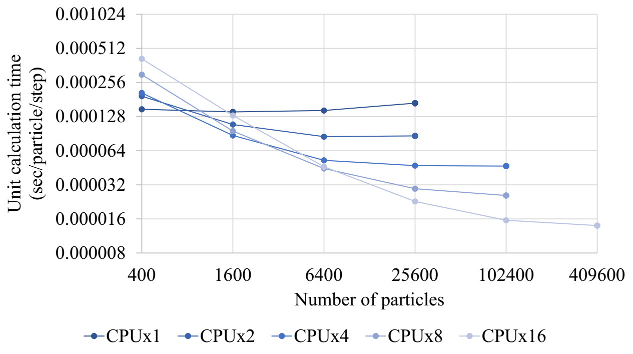


Fig. 10 Calculation time of a parallel computation with OpenMP and a CPU using MGCR solver

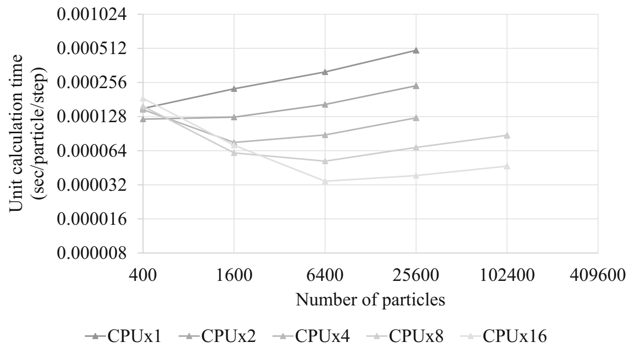


Fig. 11 Calculation time of a parallel computation with OpenMP and a CPU using CR solver

(solver main)) linearly decreased in the range of 400–6400 particles (Case x1~1/4), and after the transition range of 6400 –102,400 particles (Case x1/4~x1/16), it became almost constant in the range over 102,400 particles (Case x1/16 ~). On the other hand, the unit calculation time with respect to the preconditioning (MGCR (solver preconditioning)) linearly decreased in the whole range, showing that it was dominated by the overhead cost within all the presented cases (Cases x1~ x1/64). In addition, the unit calculation time of the MGCR preconditioning was larger than that of the MGCR main iteration in the small cases where the overhead cost is thought to be dominant. This implies that the overhead cost of the preconditioning was larger than that of the main iteration. This large overhead cost is the main reason why the MGCR method showed lower performance than the CR method in the small cases. However, the scalability of the MGCR solver is expected when considering the extrapolations in Fig. 9 toward larger cases with more than 1,638,400 particles (Case x1/64 ~). It was previously confirmed via the single CPU cases that the preconditioning stage does not need as much computation compared to that of the main iteration. Therefore, the unit calculation time with respect to the MGCR preconditioning stage decreases in larger cases where the parallel efficiency is expected to be improved. In addition, the unit calculation time of the MGCR main iteration is already almost constant with over 102,400 particles (Case x1/16 ~). Therefore, the MGCR solver is expected to be scalable in larger cases, where the MGCR main iteration time will dominate the total computational time.

A larger overhead cost is required when the number of parallel threads is larger. Since the parallel computation on GPU (A100 (80 GB)) is highly parallelized, the overhead cost was thought to be relatively large. To confirm the dependency on the number of parallel threads, the cases in Table 2 are studied with OpenMP parallel computations on a CPU (Xeon Gold 6252 (24 core)). The unit calculation times obtained with the MGCR solver are shown in Fig. 10. The overhead cost was dominant in the range where the straight decreasing trends were found. With a larger number of CPU threads, the overhead cost was larger, and it dominated the total computational time in a wider range. In addition, the unit calculation time was mostly constant in the sufficiently large cases where the overhead cost was not dominant. This implies that the computational time is proportional to the problem size and that the numerical method is scalable when the problem size is sufficiently large. In comparison, the unit calculation time with the CR solver is shown in Fig. 11. A straight decreasing trend was also observed in Fig. 11, but the overhead cost and its dominant range were smaller than those in Fig. 10. This is because the CR solver does not include preconditioning where the large overhead cost is needed. In contrast to Fig. 10, the unit calculation time in Fig. 11 increased in the sufficiently large cases where the overhead cost was not dominant. This is because the number of iterations in the CR solver increases against the problem size.

Overall, in parallel computation, the multigrid solvers (MGCR and MGCG) did not perform well in the small cases due to the large overhead cost with respect to the preconditioning stage, but they were efficient in the large cases where the number of iterations mainly determines the total computational time.

4.4 Three-dimensional calculations

The trends observed in the above sections were almost the same as those in the 3D calculations. Here, a simple 3D high-viscosity incompressible dam break problem (Fig. 12) was taken as an example. Based on the conditions in Table 1, the calculation conditions in the 3D scaled cases are given in Table 4. The calculations were conducted on a GPU (A100 (80 GB)) with OpenACC applying the MGCR and CR solvers. The number of solver iterations at t = 0.2 s is shown in Fig. 13. While the CR method suffered from a large number of iterations in the large cases, the MGCR method could suppress the iterations even in the large cases. The breakdowns of the unit calculation times are shown in

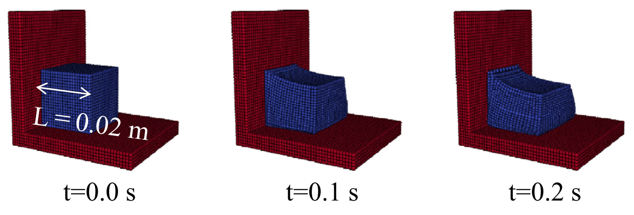


Fig. 12 Dam-break calculation of a highly viscous incompressible flow in 3D

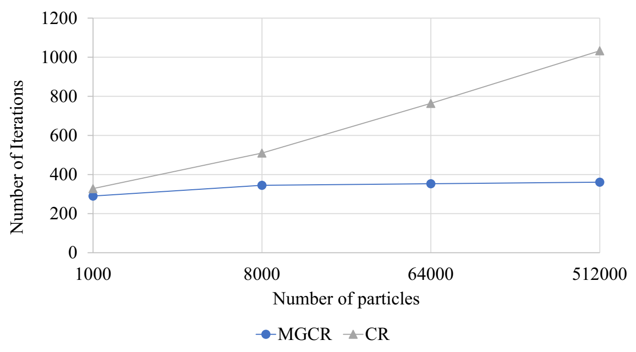


Fig. 13 Number of iterations in the 3D scaled cases (Table 4)

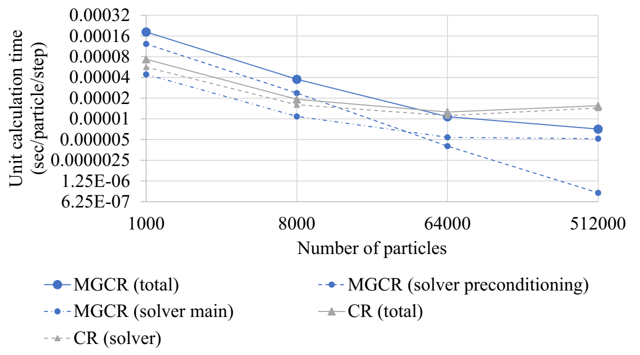


Fig. 14 Calculation time in the 3D scaled cases (Table 4) computed with OpenACC and a GPU

Fig. 14. Since the parallel efficiency of the preconditioning was not good, the MGCR method was slower than the CR method in the small cases. In contrast, the MGCR method was faster in the large cases where the number of iterations mainly determines the total computational time. This indicates that the multigrid technique is useful for both 2D and 3D parallel calculations when the problem size is sufficiently large.

5 Conclusion

In this study, a scalable MPH-I method was developed. A derivation of the SPD matrix equation through pressure substitution [21] was presented, and additionally, it was shown that the SPD feature generally appears in a physically consistent system by deriving it from the analytical mechanical equation [23]. To solve the SPD matrix equation, a bucket-based multigrid preconditioner was constructed such that it satisfies the condition for application to the CG and CR solvers. Moreover, to handle the complexity due to multi-dimensionality, the extended Jacobi iteration, which is also applicable for nondiagonally dominant matrix equations, was proposed. In the benchmark calculations, the simple high-viscosity incompressible dam break problems were calculated with the MGCR, MGCG, CR and CG solvers in both 2D and 3D. The number of iterations could be suppressed by the multigrid solvers, and it was smaller with the CR solvers than with the CG solvers regardless of the preconditioning steps. Consequently, the MGCR solver showed the best performance of the four, and the number of iterations hardly depended on the problem size. In fact, the computational time in the single CPU calculation was almost proportional to the numbers of particles and time steps, and the scalability was presented within the tested cases. The performance of the multigrid solvers was also tested in parallel computations on a CPU and GPU. For the small problems, the MGCR and MGCG solvers were slower than the CR and CG solvers

Table 4 Calculation conditions for the scaled cases in 3D

	Case x2	Case x1 (base case)	Case x1/2	Case x1/4
Particle spacing (m) l	2×10^{-3}	1×10^{-3}	$1/2 \times 10^{-3}$	$1/4 \times 10^{-3}$
Time step width (s) Δt	2×10^{-3}	1×10^{-3}	$1/2 \times 10^{-3}$	$1/4 \times 10^{-3}$
Number of particles	1000	8000	64,000	512,000
$d_\mu = \mu\Delta t/\rho_0 l^2$	$1/2 \times 10$	1×10	2×10	4×10
$d_\lambda = \mu\Delta t/\rho_0 l^2$	$1/2 \times 10^3$	1×10^3	2×10^3	4×10^3
DoF	3000	24,000	192,000	1,536,000
Nonzero count nnz (approx.)	1.62×10^6	1.29×10^7	1.03×10^8	8.28×10^8
Condition number K_A (approx.)	1.31×10^1	5.21×10^1	2.05×10^2	7.95×10^2
Maximum level in multigrid calculation	2	3	4	5

because of the large overhead cost in the preconditioning process. However, they showed better performance than conventional solvers for large problems, where the number of solver iterations mainly determines the calculation time.

Funding Not applicable.

Data availability The data that support the findings of this study are available from the corresponding author upon reasonable request.

Code availability The code that supports the findings of this study is available from the corresponding author upon reasonable request and licensing.

Declarations

Conflict of interest The author reports no conflict of interest in this study.

Appendix A: Convergence of the generalized CG/CR solvers

The algorithm expressed in Eqs. (21)–(27) will theoretically converge within N iterations when solving a linear matrix equation with N degrees of freedom when the matrices $\mathbf{M}$ and $\mathbf{MAK}$ are symmetric positive definite (SPD). First, the orthogonalities (Eqs. (28)–(30)) will be proven. Assume that Eqs. (28)–(30) hold when$i$,$j < k$. From Eq. (30),

$$\{\mathbf{Ap}_i, \mathbf{MAp}_j\} = \left\{-\frac{1}{\alpha_i}(\mathbf{r}_{i+1} - \mathbf{r}_i), \mathbf{MAp}_j\right\} = 0$$
$$\{\mathbf{r}_{i+1}, \mathbf{MAp}_j\} = \{\mathbf{r}_i, \mathbf{MAp}_j\} \tag{A.1}$$

is derived with Eq. (22). Using the condition (Eq. (24)) for determining α_k,

$$\begin{aligned}\{\mathbf{r}_{k+1}, \mathbf{MAp}_j\} &= \{\mathbf{r}_k, \mathbf{MAp}_j\} = \{\mathbf{r}_{k-1}, \mathbf{MAp}_j\}\\ &= \cdots = \{\mathbf{r}_{j+2}, \mathbf{MAp}_j\} = \{\mathbf{r}_{j+1}, \mathbf{MAp}_j\} = 0\end{aligned} \tag{A.2}$$

holds in $j < k + 1$. Therefore, Eq. (28) is also satisfied when i,$j < k + 1$. Because Eq. (25),

$$\{\mathbf{r}_{k+1}, \mathbf{MAKr}_j\} = \{\mathbf{r}_{k+1}, \mathbf{MAp}_j - \beta_{j-1}\mathbf{MAp}_{j-1}\} = 0 \tag{A.3}$$

under $j < k + 1$. Since $\mathbf{MAK}$ is symmetric,

$$\{\mathbf{r}_{k+1}, \mathbf{MAKr}_j\} = \{\mathbf{r}_j, \mathbf{MAKr}_{k+1}\} = 0. \tag{A.4}$$

Therefore,

$$\{\mathbf{r}_i, \mathbf{MAKr}_j\} = 0 \tag{A.5}$$

under Conditions $i \neq j$ and $i, j < k + 1$, and Eq. (29) is also satisfied when i, $j < k + 1$. Using Eqs. (22) and (25), Eq. (30) is deformed as

$$\begin{aligned}&\{\mathbf{Ap}_j, \mathbf{MAp}_{k+1}\}\\ &= \left\{-\frac{1}{\alpha_j}(\mathbf{r}_{j+1} - \mathbf{r}_j), \mathbf{MAKr}_{k+1} - \beta_k\mathbf{MAp}_k\right\}\\ &= \left\{-\frac{1}{\alpha_j}(\mathbf{r}_{j+1} - \mathbf{r}_j), -\beta_k\mathbf{MAp}_k\right\}\\ &= \{\mathbf{Ap}_j, -\beta_k\mathbf{MAp}_k\} = 0.\end{aligned} \tag{A.6}$$

With the condition (Eq. (27)) used for determining β_k,

$$\{\mathbf{Ap}_j, \mathbf{MAp}_{k+1}\} = 0 \tag{A.7}$$

holds in $j < k + 1$, and with the symmetry of $\mathbf{M}$,

$$\{\mathbf{Ap}_i, \mathbf{MAp}_j\} = 0 \tag{A.8}$$

is derived under $i \neq j$ and i,$j < k + 1$. Therefore, Eq. (30) is also satisfied when i,$j < k + 1$. Then, Eqs. (28)–(30) are satisfied recursively. Moreover, since$\mathbf{M}$ is SPD, it is expressed as

$$\mathbf{M} = \mathbf{W}^T\mathbf{W} \tag{A.9}$$

with a nonsingular matrix $\mathbf{W}$, and Eq. (30) will be

$$\{\mathbf{WAp}_i, \mathbf{WAp}_j\} = 0 \quad (j \neq i). \tag{A.10}$$

Here, $\{\mathbf{WAp}_0, \mathbf{WAp}_1, \ldots, \mathbf{WAp}_k\}$ form an orthogonal basis for a k dimensional space, and the search vectors $\{\mathbf{p}_0, \mathbf{p}_1, \ldots \mathbf{p}_k\}$ are linearly independent of each other. Therefore, k linearly independent search directions are produced by k iterations, but the number of linearly independent vectors cannot exceed N in an N dimensional space. This implies that the N dimensional space is all explored with the N search directions, and the solver will find the exact solution at most after N iterations.

Appendix B: Extended Jacobi iteration

It is shown here that the extended Jacobi iteration (Eq. (42)) can obtain asymptotic convergence even in a nondiagonally dominant matrix equation. Assume that the initial solution is $\mathbf{x}_0 = \mathbf{0}$ in solving the linear matrix equation (Eq. (39)) having a symmetric coefficient matrix $\mathbf{A}$. With a matrix $\mathbf{Q}$ given by

$$\mathbf{Q} = \hat{\mathbf{D}} - \mathbf{A} \tag{B.1}$$

Equation (42) is written as

$$\mathbf{x}^{k+1} = \hat{\mathbf{D}}^{-1}\mathbf{b} + \hat{\mathbf{D}}^{-1}\mathbf{Q}\mathbf{x}^{k}, \tag{B.2}$$

and the matrix $\mathbf{M}_{\mathrm{exJ}}$ corresponding to the iterative calculation is expressed as

$$\begin{aligned}\mathbf{M}_{exJ} &= \hat{\mathbf{D}}^{-1} + \hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1} + \hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1}\\ &\quad + \hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1} + \ldots\\ &= \hat{\mathbf{D}}^{-1}(\hat{\mathbf{D}} + \mathbf{Q})\hat{\mathbf{D}}^{-1} + \hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1}(\hat{\mathbf{D}} + \mathbf{Q})\hat{\mathbf{D}}^{-1}\mathbf{Q}\hat{\mathbf{D}}^{-1} + \ldots\end{aligned} \tag{B.3}$$

Since the matrix $\hat{\mathbf{D}} + \mathbf{Q} = 2\hat{\mathbf{D}} - \mathbf{A}$ is symmetric and diagonally dominant as

$$\hat{D}_{ii} + Q_{ii} > \sum_{j} |A_{ij}| - A_{ii} = \sum_{j \neq i} |A_{ij}| = \sum_{j \neq i} |\hat{D}_{ii} + Q_{ii}|, \tag{B.4}$$

it is symmetric positive definite (SPD). Consequently, $\mathbf{M}_{\mathrm{exJ}}$ is also SPD. On the other hand, matrix $\mathbf{M}_{\mathrm{exJ}}$ can also be written as

$$\mathbf{M}_{exJ} = \mathbf{A}^{-1}(\mathbf{I} - (\mathbf{I} - \mathbf{A}\hat{\mathbf{D}}^{-1})^{\mathrm{k}}) \tag{B.5}$$

using $\mathbf{A}^{-1}$ (see Appendix C). For Eq. (B.5) to be SPD, the eigenvalues of $(\mathbf{I} - (\mathbf{I} - \mathbf{A}\hat{\mathbf{D}}^{-1})^{\mathrm{k}})$ need to be positive, and those of $\mathbf{I} - \mathbf{A}\hat{\mathbf{D}}^{-1}$ need to be less than 1. Let $\mathbf{q}_{\mathrm{i}}$ and Λ_{i} be the eigenvectors and eigenvalues of $\mathbf{I} - \mathbf{A}\hat{\mathbf{D}}^{-1}$, respectively. An arbitrary vector $\mathbf{z}$ is expressed as

$$\mathbf{z} = \sum_{i} \alpha_i \Lambda_i \mathbf{q}_i, \tag{B.6}$$

Because

$$\{\mathbf{z}, \mathbf{A}^{-1}(\mathbf{I} - (\mathbf{I} - \mathbf{A}\hat{\mathbf{D}}^{-1})^{k})\mathbf{z}\} = \sum_{i} \alpha_i (1 - \Lambda_i^{k}) \{\mathbf{q}_i, \mathbf{A}^{-1}\mathbf{q}_i\} > 0, \tag{B.7}$$

the eigenvalues $\Lambda_{\mathrm{i}} < 1$. This implies that the convergence $(\mathbf{I} - \mathbf{A}\hat{\mathbf{D}}^{-1})^{\mathrm{k}} \to \mathbf{O}$ and $\mathbf{M}_{\mathrm{ex}} \to \mathbf{A}^{-1}$ is obtained when $k \to \infty$.

Appendix C: Matrix corresponding to iterative calculation

The Jacob iteration, extended Jacobi iteration (Eq. (42)) and multigrid calculation are generally expressed as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{L}_k(\mathbf{b} - \mathbf{A}\mathbf{x}_k). \tag{C.1}$$

For convergence, $\mathbf{L}_{\mathrm{k}}$ must be a good approximation of $\mathbf{A}^{-1}$. Specifically, in the Jacobi iteration, extended Jacobi iteration and multigrid calculation, $\mathbf{L}_{\mathrm{k}}$ s are given as

$$\begin{aligned}\mathbf{L}_k &= \mathbf{D}^{-1}\\ \mathbf{L}_k &= \hat{\mathbf{D}}^{-1}\\ \mathbf{L}_k &= \mathbf{PMR},\end{aligned} \tag{C.2}$$

respectively, where $\mathbf{P}$, $\mathbf{R}$ and $\mathbf{M}$ are the prolongation matrix, restriction matrix and the matrix corresponding to the upper-level calculation. Since the recursive formula (Eq. (C.1)) is deformed as

$$\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1} = (\mathbf{I} - \mathbf{A}\mathbf{L}_k)(\mathbf{b} - \mathbf{A}\mathbf{x}_k), \tag{C.3}$$

the iterative calculation with the initial solution of $\mathbf{x}_0 = \mathbf{0}$ is expressed as

$$\begin{aligned}\mathbf{b} - \mathbf{A}\mathbf{x}_k &= (\mathbf{I} - \mathbf{A}\mathbf{L}_{k-1})(\mathbf{I} - \mathbf{A}\mathbf{L}_{k-2})\\ &\qquad \cdots (\mathbf{I} - \mathbf{A}\mathbf{L}_1)(\mathbf{I} - \mathbf{A}\mathbf{L}_0)(\mathbf{b} - \mathbf{A}\mathbf{x}_0)\\ &= (\mathbf{I} - \mathbf{A}\mathbf{L}_{k-1})(\mathbf{I} - \mathbf{A}\mathbf{L}_{k-2})\\ &\qquad \cdots (\mathbf{I} - \mathbf{A}\mathbf{L}_1)(\mathbf{I} - \mathbf{A}\mathbf{L}_0)\mathbf{b}\\ \mathbf{x}_k &= \{\mathbf{A}^{-1} - \mathbf{A}^{-1}(\mathbf{I} - \mathbf{A}\mathbf{L}_{k-1})(\mathbf{I} - \mathbf{A}\mathbf{L}_{k-2})\\ &\qquad \cdots (\mathbf{I} - \mathbf{A}\mathbf{L}_1)(\mathbf{I} - \mathbf{A}\mathbf{L}_0)\}\mathbf{b},\end{aligned} \tag{C.4}$$

and the corresponding matrix is

$$\begin{aligned}\mathbf{M}_{gen} &= \mathbf{A}^{-1} - \mathbf{A}^{-1}(\mathbf{I} - \mathbf{A}\mathbf{L}_{k-1})(\mathbf{I} - \mathbf{A}\mathbf{L}_{k-2})\\ &\qquad \cdots (\mathbf{I} - \mathbf{A}\mathbf{L}_1)(\mathbf{I} - \mathbf{A}\mathbf{L}_0)\\ &= \mathbf{A}^{-1} - \mathbf{Z}.\end{aligned} \tag{C.5}$$

Moreover, the second term $\mathbf{Z}$ in Eq. (C.5) is deformed as

$$\begin{aligned}\mathbf{Z} &= \mathbf{A}^{-1}(\mathbf{I} - \mathbf{A}\mathbf{L}_{k-1}) \cdots (\mathbf{I} - \mathbf{A}\mathbf{L}_p)(\mathbf{I} - \mathbf{A}\mathbf{L}_{p-1}) \ldots (\mathbf{I} - \mathbf{A}\mathbf{L}_0)\\ &= (\mathbf{I} - \mathbf{L}_{k-1}\mathbf{A}) \cdots (\mathbf{I} - \mathbf{L}_p\mathbf{A})\mathbf{A}^{-1}(\mathbf{I} - \mathbf{A}\mathbf{L}_{p-1}) \ldots (\mathbf{I} - \mathbf{A}\mathbf{L}_0)\\ &= (\mathbf{I} - \mathbf{L}_{k-1}\mathbf{A}) \cdots (\mathbf{I} - \mathbf{L}_p\mathbf{A})(\mathbf{I} - \mathbf{L}_{p-1}\mathbf{A}) \ldots (\mathbf{I} - \mathbf{L}_0\mathbf{A})\mathbf{A}^{-1}.\end{aligned} \tag{C.6}$$

When $\mathbf{A}$ is symmetric and

$$\mathbf{L}_p = \mathbf{L}_{k-p-1} \tag{C.7}$$

for every $p = 0,1,2,\ldots k - 1,$

$$\mathbf{Z} = \mathbf{Z}^T \tag{C.8}$$

holds, and $\mathbf{M}_{gen}$ will be symmetric.

References

1. Monaghan JJ (1994) Simulating free surface flows with SPH. J Comput Phys 110:399–406. https://doi.org/10.1006/jcph.1994.1034
2. Koshizuka S, Oka Y (1996) Moving-Particle Semi-Implicit methods for fragmentation of incompressible fluid. Nucl Sci Eng 123:421–434. https://doi.org/10.13182/NSE96-A24205
3. Colagrossi A, Landrini M (2003) Numerical simulation of interfacial flows by smoothed particle hydrodynamics. J Comput Phys 191:448–475. https://doi.org/10.1016/S0021-9991(03)00324-3
4. Molteni D, Colagrossi A (2009) A simple procedure to improve the pressure evaluation in hydrodynamic context using the SPH. Comput Phys Commun 180:861–872. https://doi.org/10.1016/j.cpc.2008.12.004
5. Antuono M, Colagrossi A, Marrone S, Molteni D (2010) Free-surface flows sloved by means of SPH schemes with numerical diffusive terms. Comput Phys Commun 181:532–549. https://doi.org/10.1016/j.cpc.2009.11.002
6. Marrone S, Antuono M, Colagrossi A, Colicchio G, Le Touze D, Graziani G (2011) δ-SPH model for simulating violent impact flows. Comput Methods Appl Mech Engrg 200:1526–1542. https://doi.org/10.1016/j.cma.2010.12.016
7. Hu XY, Adams NA (2006) A multi-phase SPH method for macroscopic and mesoscopic flows. J Comput Phys 213:844–861. https://doi.org/10.1016/j.jcp.2005.09.001
8. Khayyer A, Gotoh H (2011) Enhancement of stability and accuracy of the moving particle semi-implicit method. J Comput Phys 230:3093–3118. https://doi.org/10.1016/j.jcp.2011.01.009
9. Tanaka M, Masunaga T (2010) Stabilization and smoothing of pressure in MPS method by quasi-compressiblility. J Comput Phys 229:4279–4290. https://doi.org/10.1016/j.jcp.2010.02.011
10. Kondo M, Koshizuka S (2011) Improvement of stability in moving particle semi-implicit method. Int J Numer Meth Fluids 65:638–654. https://doi.org/10.1002/fld.2207
11. Asai M, Aly AM, Sonoda Y, Sakai Y (2012) A stabilized incompressible SPH method by relaxing the density invariance condition. J Appl Math 2012:139583. https://doi.org/10.1155/2012/139583
12. Xu R, Stansby P, Laurence D (2009) Accuracy and stability in incompressible SPH (ISPH) based on projection method and a new approach. J Comput Phys 228:6703–6725. https://doi.org/10.1016/j.jcp.2009.05.032
13. Hosseini SM, Feng JJ (2011) Pressure boundary conditions for computing incompressible flows with SPH. J Comput Phys 230:7473–7487. https://doi.org/10.1016/j.jcp.2011.06.013
14. Lind SJ, Xu R, Stansby PK, Rogers BD (2012) Incompressible smoothed particle hydrodynamics for free-surface flows: a general diffusion-based algorithm for stability and validations for impulsive flows and propagating waves. J Comput Phys 231:1499–1523. https://doi.org/10.1016/j.jcp.2011.10.027
15. Shadloo MS, Zainali A, Sadek SH, Yildiz M (2011) Improved incompressible smoothed particle hydrodynamics method for simulating flow around bluff bodies. Comput Methods Appl Mech Eng 200:1008–1020. https://doi.org/10.1016/j.cma.2010.12.002
16. Tsuruta N, Khayyer A, Gotoh H (2013) A short note on dynamic stabilization of moving particle semi-implicit method. Comput Fluids 82:158–164. https://doi.org/10.1016/j.compfluid.2013.05.001
17. Kondo M (2021) A physically consistent particle method for incompressible fluid flow calculation. Comput Part Mech 8:69–86. https://doi.org/10.1007/s40571-020-00313-w
18. Kondo M, Matsumoto J (2021) Weakly compressible particle method with physical consistency for spatially discretized system, Transactions of JSCES (2021) Paper No. 20210006 (in Japanese). https://doi.org/10.11421/jsces.2021.20210006
19. Kondo M, Fujiwara T, Masaie I, Matsumoto J (2021) A physically consistent particle method for high-viscous free-surface flow calculation. Comput Part Mech. https://doi.org/10.1007/s40571-021-00408-y
20. Kondo M, Matsumoto J (2021) Surface tension and wettability calculation using density gradient potential in a physically consistent particle method. Comput Methods Appl Mech Eng 385:114072. https://doi.org/10.1016/j.cma.2021.114072
21. Kondo M, Matsumoto J (2021) Pressure substituting implicit solver to speed-up moving particle hydrodynamics method for high-viscous incompressible flows, Transactions of JSCES (2021) Paper No. 20210016. (in Japanese). https://doi.org/10.11421/jsces.2021.20210016
22. MphImplicit (GPLv3 license). https://github.com/Masahiro-Kondo-AIST/MphImplicit
23. Goldstein H, Poole CP, Safko JL (2013) Clasical mechanics, Pearson New International Edition
24. Suzuki Y, Koshizuka S (2008) A Hamiltonian particle method for non-linear elastodynamics. Int J Numer Meth Eng 74:1344–1373. https://doi.org/10.1002/nme.2222
25. Kondo M, Suzuki Y, Koshizuka S (2010) Suppressing local particle oscillations in the Hamiltonian particle method for elasticity. Int J Numer Meth Eng 81:1514–1528. https://doi.org/10.1002/nme.2744
26. Kondo M, Koshizuka S (2010) Development of thin plate model using Hamiltonian particle method, Transactions of JSCES, Paper No. 20100016 (in Japanese). https://doi.org/10.11421/jsces.2010.20100016
27. Ellero M, Serrano M, Español P (2007) Incompressible smoothed particle hydrodynamics. J Comput Phys 226:1731–1752. https://doi.org/10.1016/j.jcp.2007.06.019
28. Suzuki Y, Koshizuka S, Oka Y (2007) Hamiltonian moving-particle semi-implicit (HMPS) method for incompressible fluid flows. Comput Methods Appl Mech Eng 196:2876–2894. https://doi.org/10.1016/j.cma.2006.12.006
29. Leimkuhler B, Reich S (2004) Simulating Hamiltonian dynamics. Cambridge University Press, Cambridge
30. Yokoyama R, Kondo M, Suzuki S, Okamoto K (2021) Analysis of molten metal spreading and solidification behaviors utilizing moving particle full-implicit method. Front Energy 15:959–973. https://doi.org/10.1007/s11708-021-0753-0
31. Yokoyama R, Kondo M, Suzuki S, Okamoto K (2022) Simulating melt spreading into shallow water using moving particle hydrodynamics with turbulence model. Comput Part Mech. https://doi.org/10.1007/s40571-022-00520-7
32. Negishi H, Kondo M, Amakawa H, Obara S, Kurose R (2023) A fluid lubrication analysis including negative pressure using a physically consistent particle method. Comput Part Mech. https://doi.org/10.1007/s40571-023-00584-z
33. Saad Y (2003) Iterative methods for sparse linear systems, 2nd edn. SIAM Press, Philadelphia
34. Trottenberg U, Oosterlee CW, Schuller A (2000) Multigrid. Elsevier, Amsterdam
35. Briggs WL, Henson VE (2000) S, 2nd edn. F. McCormick, A multigrid tutorial
36. Wesseling P, Oosterlee CW (2001) Geometric multigrid with applications to computational fluid dynamics. J Comput Appl Math 128:311–334. https://doi.org/10.1016/S0377-0427(00)00517-3

37. Cummins SJ, Rudman M (1999) An SPH projection method. J Comput Phys 152:584–607. https://doi.org/10.1006/jcph.1999.6246
38. Trask N, Maxey M, Kim K, Perego M, Parks ML, Yang K, Xu J (2015) A scalable consistent second-order SPH solver for unsteady low Reynolds number flows. Comput Methods Appl Mech Eng 289:155–178. https://doi.org/10.1016/j.cma.2014.12.027
39. Chow AD, Rogers BD, Lind SJ, Stansby PK (2018) Incompressible SPH (ISPH) with fast Poisson solver on a GPU. Comput Phys Commun 226:81–103. https://doi.org/10.1016/j.cpc.2018.01.005
40. Guo X, Rogers BD, Lind S, Stansby PK (2018) New massively parallel scheme for Incompressible Smoothed Particle Hydrodynamics (ISPH) for highly nonlinear and distorted flow. Comput Phys Commun 233:16–28. https://doi.org/10.1016/j.cpc.2018.06.006
41. Matsunaga T, Shibata K, Murotani K, Koshizuka S (2016) Solution of pressure Poisson equation in particle method using algebraic multigrid method, Transactions of JSCES Paper No. 20160012 (in Japanese). https://doi.org/10.11421/jsces.2016.20160012
42. Södersten A, Matsunaga T, Koshizuka S (2019) Bucket-based multigrid preconditioner for solving pressure Poisson equation using a particle method. Comput Fluids. https://doi.org/10.1016/j.compfluid.2019.104242
43. Takahashi T, Lin MC (2006) A multilevel SPH solver with unified solid boundary handling. Comput Graph Forum 35:517–512. https://doi.org/10.1111/cgf.13048
44. Seibold B (2010) Performance of algebraic multigrid methods for non-symmetric matrices arising in particle methods. Numer Linear Algebra Appl 17:433–451. https://doi.org/10.48550/arXiv.0905.3005
45. Metsch B, Nick F, Kuhnert J (2020) Algebraic multigrid for the finite pointset method. Comput Vis Sci 23:3. https://doi.org/10.1007/s00791-020-00324-3
46. Schöberl J, Zulehner W (2003) On Schwarz-type smoothers for saddle point problems. Numer Math 95:377–399. https://doi.org/10.1007/s00211-002-0448-3
47. Tatebe O (1993) The multigrid preconditioned conjugate gradient method. In: Proceedings of sixth copper mountain conference on multigrid methods, NASA conference publication, vol 3224, pp 621–634. https://www.hpcs.cs.tsukuba.ac.jp/~tatebe/research/paper/CM93-tatebe.pdf
48. Fish J, Belsky V (1995) Multigrid method for periodic heterogeneous media Part 1: convergence studies for one-dimensional case. Comput Methods Appl Mech Eng 126:1–16. https://doi.org/10.1016/0045-7825(95)00811-E
49. Fish J, Belsky V (1995) Multi-grid method for periodic heterogeneous media Part 2: Multiscale modeling and quality control in multidimensional case. Comput Methods Appl Mech Eng 126:17–38. https://doi.org/10.1016/0045-7825(95)00812-F
50. Zienkiewicz OC, Taylor RL (2002) The finite element method, 5th edn. Butterworth-Heinemann, Oxford