Time-based dynamic load balancing algorithm for domain decomposition with particle method adopting three-dimensional polygon-wall boundary model

Yoshiki Mizuno1, *, Naoto Mitsume2, Tomonori Yamada1, Shinobu Yoshimura1

1Department of Systems Innovations, School of Engineering, The University of Tokyo
2Department of Engineering Mechanics and Energy, Faculty of Engineering, Information and Systems, University of Tsukuba

*mizuno@mps.q.t.u-tokyo.ac.jp

Received: April 8, 2019; Accepted: August 6, 2019; Published: August 31, 2019

Abstract. This paper presents a time-based dynamic load balancing algorithm for efficient domain decomposition by the Explicit Moving Particle Simulation (EMPS) method with the Explicitly Represented Polygon (ERP) wall boundary model in three-dimensional computational domains. This proposed algorithm utilizes calculation time required for the EMPS and ERP algorithms to decompose analysis domains. Compared to a distributed memory parallel algorithm that partitions computational domains based on the number of particles, the newly designed algorithm simulates free-surface flow problems with less computational cost. This paper also demonstrates the parallel efficiency of the time-based dynamic load balancing algorithm by dynamic load balancing and strong scaling efficiency tests.

Keywords: Distributed memory parallel computing, Domain decomposition, Dynamic load balancing, Free-surface flow analysis, Particle method, Polygon wall boundary model

1. Introduction

Japan always confronts tsunami problems and needs to minimize damage caused by these disasters. For instance, the Great East Japan Earthquake of 2011 occurred off the pacific coast of Japan. This earthquake was recorded as the 9.0-9.1 magnitude tremor and caused tsunami waves that impaired structures such as private houses and public facilities.

To investigate tsunami and related issues, numerical simulation is considered as one of the most appropriate substitutes for flume experiments. Tsunami behaviors can be categorized as free-surface flow problems and described by the Navier-Stokes and continuity equations. Although there are various numerical methods to discretize these governing equations, particle methods or mesh-free methods are regarded as one of the most effective techniques. Compared to mesh-based methods, meshfree methods such as the Smoothed Particle Hydrodynamics (SPH) method, Moving Particle Semi-Implicit (MPS) method, and Explicit Moving Particle Simulation (EMPS) method can express those equations by particles without node connectivity information [1, 2, 3]. Furthermore, particle methods adopt the Lagrangian description but not the Eulerian description, and consequently it can solve free-surface flow problems without calculating a convective term.

Among these techniques, EMPS is considered to be one of the most appropriate methods in terms of solving tsunami problems on parallel computers. It can calculate all the terms of the
Navier-Stokes equations explicitly [4]. Also, it can conduct parallel computing more easily than the MPS method [5], which requires solving the pressure term implicitly. In fact, given the total number of particles is represented as $n$, it costs $O(n^{1.0})$ to analyze free-surface flow problems in the EMPS method but $O(n^{1.5})$ in the MPS method.

The EMPS method generally adopts free boundaries and fixed boundaries to satisfy wall boundary conditions [6]. Free boundaries are suitable for problems with moving boundaries but not the fixed ones. In this study, we focus on problems with fixed boundaries used in free-surface flow problems, such as hydrostatic pressure and dam break problems. In the EMPS method, fixed boundaries include the conventional wall-particle boundary model [2] and polygon-wall boundary models. Each of these models has advantages with representing wall boundary conditions, yet this study applies a polygon-wall boundary model for simulating free-surface flow problems with less computational cost on parallel computers.

In the previous studies, for instance, Harada et al., Yamada et al., and Mitsume et al. proposed polygon-wall boundary models specifically for the MPS method and/or EMPS method [7, 8, 9]. Compared to the former models, the Explicitly Represented Polygon (ERP) wall boundary model developed by Mitsume et al. demonstrates improvements in decreasing pressure distribution for hydrostatic pressure, dam break, Poiseuille flow, and Couette flow problems. In addition, the ERP algorithm is simply designed for the EMPS method, and thus we apply the polygon-wall boundary model in this study. However, it was not extended to three-dimensional free-surface flow problems on parallel computers.

Murotani et al. developed a parallel free-surface flow analysis system, HDDM_EMPS, based on the distributed parallel SPH method [10, 11]. It can conduct tsunami-scale simulations by the EMPS method on parallel computers. In the previous study, they developed a distributed memory parallel algorithm for decomposing analysis domains into processing elements for parallel computing [12]. However, HDDM_EMPS adopted the conventional wall-particle boundary model but not a polygon-wall boundary model. Hence, this paper illustrates the following improvements from the previous studies.

(a) Developing the three-dimensional ERP wall boundary model for HDDM_EMPS

(b) Designing another distributed memory parallel algorithm suitable for this model

Regarding (b), Mizuno et al. reported that the distributed memory parallel algorithm developed by Murotani et al. is not necessarily appropriate for the system with the ERP wall boundary model [13, 14]. The polygon wall boundary model necessitates pressure values from particles located close to respective polygons, and consequently this algorithm may cause dynamic load balancing problems if the number of polygons is asymmetrically allocated in analysis domains. To solve these problems, we design a time-based dynamic load balancing algorithm for HDDM_EMPS with the three dimensional ERP wall boundary model. Although the former algorithm cuts analysis domains by distributing mostly the same number of particles into processing elements, the latter algorithm divides computational domains based on calculation time required for the EMPS and ERP algorithms.

The remaining sections are composed as follows. Section 2 overviews the EMPS method, and Section 3 describes the ERP wall boundary model. Section 4 is the most significant part of this paper; it explains more details about (a) and (b) listed above. Section 5 presents the parallel efficiency of the time-based dynamic load balancing algorithm in the HDDM_EMPS system with the three-dimensional ERP wall boundary model. Section 6 states the concluding remarks about this study.
2. Explicit Moving Particle Simulation method

Oochi et al. proposed the Explicit Moving Particle Simulation (EMPS) method [3, 4]. The EMPS method discretizes the following Navier-Stokes and continuity equations:

\[
\frac{Dv}{Dt} = -\frac{1}{\rho} \nabla p + \nu \nabla^2 v + g, \tag{1}
\]

\[
\frac{Dp}{Dt} + \rho \nabla \cdot v = 0, \tag{2}
\]

where \( \frac{D}{Dt} \), \( v \), \( \rho \), \( p \), \( \nu \), and \( g \) signify the Lagrange derivative, the velocity vector, the fluid density, the pressure, the kinetic viscosity, and the gravitational acceleration vector, respectively.

EMPS differentiates MPS in that it assumes fluids are weakly compressible and calculates all the terms of the Navier-Stokes equations explicitly but not semi-implicitly. The EMPS method expresses the derivatives for the particles \( i \) as

\[
\langle \nabla p \rangle_i = \frac{d}{n^0} \sum_{j=1}^N \left( \frac{(p_j + p_i)(x_j - x_i)}{|x_j - x_i|^2} \right) w(|x_j - x_i|), \tag{3}
\]

\[
\langle \nabla^2 v \rangle_i = \frac{2d}{n^0} \sum_{j=1}^N \left( w(|x_j - x_i|) \right), \tag{4}
\]

where \( d \), \( x \), \( x^0 \), and \( w \) indicate the number of dimensions, the position of particles, the initial value of a correction parameter, and weight function, respectively. \( n^0 \) is the particle number density in the initial particle configuration. The particle number density for particle \( i \), \( n_i \), is defined as follows:

\[
n_i = \sum_{j=1}^N w(|x_j - x_i|). \tag{5}
\]

In the EMPS method, the pressure of particle \( i \) is explicitly computed as follows:

\[
p_i = c^2 \rho \left( \frac{n_i}{n^0} - 1 \right), \tag{6}
\]

where \( c \) is a sound speed parameter adjusted for numerical stability. We use the Euler method for the time integration.

3. Explicitly Represented Polygon wall boundary model

Mitsume et al. proposed the Explicitly Represented Polygon (ERP) wall boundary model [9]. It improves the accuracy of pressure values compared to the former polygon wall boundary models and does not need to allocate virtual particles. Besides, the ERP wall boundary model has a potential to treat arbitrarily-shaped triangular polygons for wall boundaries and reduce computational cost for planar walls more than the wall-particle boundary model. It adopts the mirror particle approach [15] and fulfills the pressure Neumann boundary and the slip/no-slip boundary conditions on walls.

Let \( x^\text{near}_{i,k} \) denote the \( k \)th node of a polygon located the closest to the particle \( i \). As shown in Fig. 1, the ERP wall boundary model assumes the acting force from the particle \( i \) exerts only on the nearest point \( x^\text{wall}_i \) of a polygon wall. If \( x_i \) signifies the position of \( i \), then \( x^\text{wall}_i \) is defined as

\[
x^\text{wall}_i \equiv \arg\min_{x^\text{near}_{i,k}} |x_i - x^\text{near}_{i,k}|, \quad (k = 1, 2, \ldots, N), \tag{7}
\]
where $N$ is the number of polygons.

If $n_i^{\text{wall}}$ denotes the outward unit normal vector from $x_i^{\text{wall}}$, then it is expressed as

$$n_i^{\text{wall}} = \frac{x_i - x_i^{\text{wall}}}{|x_i - x_i^{\text{wall}}|}. \quad (8)$$

The EMPS method with the ERP wall boundary model does not require virtual particles, and thus it has a possibility to reduce computational cost.

The particle number density for particle $i$, $n_i^{\text{ERP}}$, is computed using the wall weight function, $z(r)$, which is the function with respect to the distance between particle $i$ and its closest polygon, as follows:

$$n_i^{\text{ERP}} = n_i + z(|x - x_i^{\text{wall}}|). \quad (9)$$

Details of the wall weight function, $z(r)$, is explained in [9].

Let $R_i$ be the reflectional transformation matrix for the particle $i$ defined as follows:

$$R_i = I - 2n_i^{\text{wall}} \otimes n_i^{\text{wall}}, \quad (10)$$

where $I$ denotes the identity matrix. In the ERP wall boundary model, the pressure gradient term for particle $i$ is computed as follows:

$$\langle \nabla p \rangle_i^{\text{ERP}} = \langle \nabla p \rangle_i + R_i \langle \nabla p \rangle_{i'} + f_i, \quad (11)$$

where $i'$ indicates the mirror particle of $i$. The pressure value of particle $i'$, $p_{i'}$, is given as

$$p_{i'} = p_i, \quad (12)$$

to impose the pressure Neumann boundary condition. $f_i$ is the repulsive force defined as follows:

$$f_i = \begin{cases} -\alpha\left(\frac{0.5l_0}{r} - 1\right)n_i^{\text{wall}} & (0 \leq r < 0.5l_0) \\ 0 & (0.5l_0 \leq r) \end{cases}, \quad (13)$$

where $\alpha$ denotes a repulsive coefficient that enhances stability of computation, $r$ is the distance between particle $i$ and its closest polygon, and $l_0$ is the initial particle spacing.

For the slip/no-slip boundary conditions in the case with fixed boundaries, the viscosity term is computed as follows:

$$\langle \nabla^2 v \rangle_i^{\text{ERP}} = \begin{cases} \langle \nabla^2 v \rangle_i + R_i \langle \nabla^2 v \rangle_{i'} & \text{(slip condition)} \\ \langle \nabla^2 v \rangle_i - I \langle \nabla^2 v \rangle_{i'} & \text{(no-slip condition)} \end{cases}, \quad (14)$$

$$v_{i'} = \begin{cases} R_i v_i & \text{(slip condition)} \\ -I v_i & \text{(no-slip condition)} \end{cases}. \quad (15)$$
4. Domain decomposition methods

4.1. Parallel free-surface flow analysis system

Murotani et al. developed HDDM_EMPS that simulates free-surface flow problems by the EMPS method with the wall-particle boundary model on parallel computers [10, 12]. HDDM signifies Hierarchical Domain Decomposition Method, and the system is based on the distributed parallel SPH algorithm that utilizes the single bucket-based domain decomposition method [11]. However, they differentiated this bucket-based method in that it adopts a domain decomposition of two hierarchical levels, “Parts” and “Sub-Domains.” HDDM_EMPS decomposes three-dimensional analysis domains into these two hierarchical components by ParMETIS, an MPI-based library that partitions unstructured meshes and graphs [16, 17, 18, 19]. “Parts” are the first hierarchical level assigned to processing elements. On the other hand, “Sub-Domains” are the second hierarchical level decomposed from the first ones. In this study, we make HDDM_EMPS decompose analysis domains into “Parts” only. The following steps exemplifies how to decompose a two-dimensional analysis domain with three processing elements.

(a) A bounding box encapsulates an analysis domain as illustrated in Fig. 2.
(b) Buckets are placed in this bounding box as shown in Fig. 3.
(c) Particles are stored in these buckets as depicted in Fig. 4.
(d) The domain is divided by the number of processing elements as illuminated in Fig. 5 (more details are explained in 4.3.).
(e) The steps (a) through (d) are repeated if dynamic load balancing problems occur (more details are described in 4.4.).

In (b), nine buckets are set up in the bounding box. In (c), 30 particles are allocated in the computational domain. In (d), these particles are sent to the three processing elements, “PE0” colored with green, “PE1” colored with red, and “PE2” colored with brown.
4.2. Three-dimensional Explicitly Represented Polygon wall boundary model

The ERP wall boundary model was available only for two-dimensional analysis domains [9], but we extend it to three-dimensional analysis domains for the HDDM_EMPS system in this study. Compared to HDDM_EMPS with the wall-particle boundary model, HDDM_EMPS with ERP has advantages with expressing complicated-shaped boundaries and simulating free-surface flow problems with less computational time. The latest parallel free-surface flow analysis system conducts the ERP algorithm in the following steps.

(a) Locate a global bounding box for an arbitrary number of polygons as illustrated in Fig. 6.

(b) Allocate a local bounding box inside a global bounding box as depicted in Fig. 7.

(c) Filter whether particles are utilized for the ERP calculation as shown in Fig. 8.

(d) Repeat the steps (b) and (c) for each node and particle.

In (a), the pink line with dashes, the gray dots, and the blue particles indicate a global bounding box, nodes, and fluid particles, respectively. In this study, a polygon is defined as a line connected with two nodes. Figure 7 exemplifies 12 polygons are encapsulated in a global bounding box. In (b), the orange line with dashes signifies a local bounding box. The length of this bounding box is defined as that of an effective radius in the three-dimensional directions used for neighboring search for particles in the EMPS algorithm. This step prepares for the calculation of (7). As expressed in the previous section, this computation uses the particle $i$ to find the nearest point of a polygon, $x_{i,wall}^w$. In (c), the yellow particle $i$ inside the local bounding box is utilized for (7) but not the blue particles. Note that (b) is skipped if there is no fluid particle inside a global bounding box.

These steps in the ERP algorithm increases computational time required for free-surface flow simulations as the number of polygons becomes greater. Furthermore, if the number of polygons is set up asymmetrically in analysis domains, HDDM_EMPS with ERP may cause dynamic load balancing problems [13, 14].

4.3. Weight values for domain decompositions

HDDM_EMPS utilizes the specific routine of ParMETIS, ParMETIS_V3_PartGeomKway, for domain decompositions [16, 17, 18, 19]. ParMETIS_V3_PartGeomKway conducts a k-way partitioning of analysis domains by weight values of vertices and edges. The parallel free-surface flow analysis system modifies weight values of vertices but not edges. It regards particles as vertices and partitions analysis domains based on the allocation of particles in “Parts” or processing
elements. Without modulating these values appropriately, HDDM_EMPS causes dynamic load balancing problems and influences the efficiency of parallel computing in an unfavorable manner.

![Figure 9: Original algorithm with wall-particle boundary model](image1)

**Figure 9: Original algorithm with wall-particle boundary model**

![Figure 10: Original algorithm with ERP wall boundary model](image2)

**Figure 10: Original algorithm with ERP wall boundary model**

![Figure 11: Proposed algorithm with ERP wall boundary model](image3)

**Figure 11: Proposed algorithm with ERP wall boundary model**

### 4.3.1. Distributed memory parallel algorithm

The distributed memory parallel algorithm distributes particles so that respective processing elements hold mostly the same number of particles as shown in Fig. 9. This diagram illustrates how the algorithm partitions an analysis domain in the case of the three processing elements, “PE0” colored with green, “PE1” colored with red, and “PE2” colored with brown. The total number of particles is 45, and 15 particles located vertically on the right edge of the analysis domain represent a wall boundary. The algorithm designed by Murotani et al. sends 15 particles to each processing element and decomposes the domain into the three parts.

On the other hand, the algorithm developed by Murotani et al. does not function efficiently for the system adopting the ERP wall boundary model because it cannot cope with the ERP algorithm. As explained in the previous section, the polygon-wall boundary model needs to detect particles located close to each polygon wall and obtain a pressure value from the nearest particle among them. Figure 10 depicts the same analysis domain with the polygon-based wall boundary model, yet the ERP wall boundary model requires no particle for expressing wall boundaries. In fact, the 15 particles in Fig. 9 are not located on the right edge of this domain as opposed to the particle-based boundary model. In Fig. 10, the four yellow particles signify the particles incorporated in the ERP algorithm. However, the distributed memory parallel algorithm simply partitions the analysis domain based on the number of particles without differentiating the yellow particles from the others. Because the total number of particles becomes 30, it distributes 10 particles to respective processing elements, “PE0,” “PE1,” and “PE2” regardless of whether the ERP algorithm uses particles as illustrated in Fig. 10. Accordingly, there is a possibility that this method causes dynamic load balancing problems; it takes more computational cost to execute the yellow particles...
than the blue ones, and a certain processing element holds more computational burden than the other processing elements.

### 4.3.2. Time-based dynamic load balancing algorithm

The time-based dynamic load balancing algorithm designed in this study can differentiate the yellow particles from the blue ones based on their calculation time necessary for the EMPS and ERP algorithms. Since additional computation is required to fulfill the polygon-wall boundary condition, it gives respective particles corresponding weight values and decomposes the analysis domain into the three “Parts” based on the computational time. Figure 11 depicts the same analysis domain with the polygon-wall boundary as shown in Fig. 10. However, the allocation of particles are modulated by the proposed algorithm but not the conventional one. In this figure, “$T_0$,” “$T_1$,” and “$T_2$” indicate the time taken in “PE0,” “PE1,” and “PE2,” respectively. Generally, the calculation time in the $p$th processing element is measured every time step as

$$T_p = \sum_{i=0}^{n_{\text{fluid}}^p} t_{\text{fluid}}^i + \sum_{i=0}^{n_{\text{ERP}}^p} t_{\text{ERP}}^i,$$

where $n_{\text{fluid}}^p$, $n_{\text{ERP}}^p$, $t_{\text{fluid}}^i$, and $t_{\text{ERP}}^i$ denote the number of particles that is not used in the ERP algorithm (blue particles) in the $p$th processing element, the number of particles incorporated in the ERP algorithm (yellow particles) in the $p$th processing element, the computational time stored in each of blue particles, and the one in each of yellow particles, respectively.

The newly designed algorithm allocates particles so that that “$T_0$,” “$T_1$,” and “$T_2$” become mostly the same among the three processing elements. The total number of particles is 30 as in Fig. 10, and there are the same yellow particles influenced by the ERP algorithm. However, Fig. 11 indicates each processing element holds a different number of particles; “PE0,” “PE1,” and “PE2” have 13 particles, 8 particles, and 9 particles, respectively. The number of particles in each processing element is not the same because particles are allocated based on the calculation time required for EMPS and ERP algorithms.

Note that this algorithm utilizes the number of particles as weight values exceptionally at the very first step because it needs to obtain the computational time stored in each particle from the previous time step, and a domain repartition always occur in the 2nd step.

### 4.4. Thresholds for domain repartitions

During simulations, it is required to repartition calculation domains if the amount of computation becomes unstable among processing elements. Static problems such as hydrostatic pressure do not usually necessitate repartitions because their analysis domains do not move from the first configuration. Conversely, dynamic problems such as dam break may need to conduct domain repartitions, for their analysis domains change dynamically and affect dynamic load balancing.

#### 4.4.1. Distributed memory parallel algorithm

The distributed memory parallel algorithm conducts domain repartitions based on the number of particles if the following equation is satisfied:

$$c < \frac{N_p \times \max_{0 \leq p \leq N_p-1} n_p}{\sum_{p=0}^{N_p-1} n_p},$$

where $c$, $N_p$, and $n_p$ indicate a threshold greater than 1.0, the total number of processing elements, and the total number of particles in the $p$th processing element, respectively.
Figure 12: Transition in the number of particles

Figure 13: Transition in calculation time

The right hand side of (17) always becomes larger than 1.0, so the timing of repartitions varies based on the value \( c \). Figure 12 exemplifies the timing of domain repartitions for a dam break problem (refer to the next section) where \( c \) is set to be 1.05. This figure shows that the right hand side of (17) exceeds the threshold at the 65th and 66th steps, and consequently a domain repartition is executed at each of these time steps.

4.4.2. Time-based dynamic load balancing algorithm

The time-based dynamic load balancing algorithm repartitions analysis domains based on the computational time stored in particles if the following equation is fulfilled:

\[
c < \frac{N_p \times \max_{0 \leq p \leq N_p - 1} T_p}{\sum_{p=0}^{N_p-1} T_p}.
\]

Figure 13 illustrates the timing of domain repartitions for the same dam break problem where \( c \) is set to be 1.5 this time. It shows that the right hand side of (18) exceeds this threshold in the 21st, 27th, and 71st steps. Figure 13 depicts zigzag lines as opposed to Fig. 12 since respective processing elements have a variety of computational time every time step. Note that \( c \) is set to be 1.5 for both (17) and (18) in the rest of this study.

Figure 14 and Fig. 15 demonstrate the relationship between the number of polygons and the calculation time required for the EMPS and ERP computation. We solve a hydrostatic pressure problem (refer to the next section), while changing the number of polygons. In this problem, only the proposed algorithm repartitions an analysis domain once at the 2nd time step as the exceptional execution. First, Fig. 14 reveals the distributed memory parallel algorithm increases computational burden only in “PE3” as the number of polygons becomes greater. Conversely, Fig. 15 indicates the time-based dynamic load balancing algorithm solves this dynamic load balancing problem and decreases the calculation time necessary for the EMPS and ERP algorithms. These results clarify the proposed algorithm corresponds to the ERP algorithm in more desirable ways.
5. Verification of parallel performance

5.1. Problem settings

In this study, we utilize a hydrostatic pressure problem and a dam break problem for confirming their dynamic load balancing and strong scaling efficiency. Figure 16 and Fig. 17 illustrate this static problem and dynamic problem, respectively. Note the number of triangular polygons are allocated in an asymmetric manner to cause dynamic load balancing problems intentionally.

In Fig. 16, static fluid particles are allocated in a cuboid container. The volume of this container is $5.0 \times 0.4 \times 1.0$ [m$^3$], and that of the fluid particles is $5.0 \times 0.4 \times 0.6$ [m$^3$]. A surface represented by 16,200 triangular polygons is set up on the right side of the container. Except this surface, all the others have two triangular polygons. In Fig. 17, conversely, dam-shaped fluid particles are placed in the same cuboid container used in the hydrostatic pressure problem. The volume of the fluid particles is $1.0 \times 0.4 \times 1.0$ [m$^3$]. These particles eventually impacts on a rigid
structure whose volume is $0.5 \times 0.2 \times 0.6$ [m$^3$]. 16,200 triangular polygons are set up on the surface of the structure facing the particles, but the remaining surfaces hold two polygons.

5.2. Dynamic load balancing tests

Dynamic load balancing tests measure how the time-based dynamic load balancing algorithm can reduce computational burden in certain processing elements compared to the distributed memory parallel algorithm. They are conducted with four processing elements on a local machine. It has Intel Core i5-650 with four processing elements. On this environment, the problems are simulated under the conditions listed in Table 1.

Table 1: Conditions for hydrostatic pressure and dam break problems

| Conditions                  | Hydrostatic pressure | Dam break |
|-----------------------------|----------------------|-----------|
| Number of particles         | 76,800               | 25,600    |
| Simulation time             | $1.5 \times 10^{-1}$ [s] | $4.0 \times 10^{-1}$ [s] |
| Time step                   | $1.0 \times 10^{-4}$ [s] | $1.0 \times 10^{-4}$ [s] |
| Particle distance           | $2.5 \times 10^{-2}$ [m] | $2.5 \times 10^{-2}$ [m] |
| Kinematic viscosity         | $1.0 \times 10^{-6}$ [m$^2$/s] | $1.0 \times 10^{-6}$ [m$^2$/s] |
| Fluid density               | $1.0 \times 10^3$ [kg/m$^3$] | $1.0 \times 10^3$ [kg/m$^3$] |
| Effective radius            | $3.1 \times 10^{-2}$ [m] | $3.1 \times 10^{-2}$ [m] |
| Gravity                     | -9.81 [m/s$^2$]      | -9.81 [m/s$^2$] |

5.2.1. Hydrostatic pressure problem

Figure 18 indicates the results of the dynamic load balancing tests for the hydrostatic pressure problem. In this figure, “Original” and “Proposed” denote the original algorithm and proposed algorithm, respectively. In addition, “Calculation Time” indicate the total time required for the EMPS and ERP algorithms. In this problem, the number of domain repartitions is 0 and 1 by the
distributed memory parallel algorithm and time-based dynamic load balancing algorithm, respectively.

The result of the original algorithm indicates the calculation time in “PE3” is the greatest among the other processing elements since the ERP algorithm increases the calculation cost for the analysis domain in charge of “PE3.” However, the conventional algorithm does not influence the other domains in charge of “PE0,” “PE1,” and “PE2.” Because of the calculation burden only in “PE3,” the distributed memory parallel algorithm requires 2,470 [s] for completing the whole execution, including the EMPS algorithm, ERP algorithm, communication between processing elements, input/output, and so forth. On the other hand, the figure shows that the proposed algorithm reduces the calculation time taken in “PE3,” and consequently all the processing elements hold mostly the same amount of calculation time. As a matter of fact, it requires only 1,753 [s] for finishing the whole execution and saves 717 [s] compared to the conventional algorithm.

That is, these results derive from the fact that only the proposed algorithm can cope with the EMPS and ERP computation simultaneously. Figure 20 depicts the initial configuration of the hydrostatic pressure problem with four processing elements. In this diagram, “PE0,” “PE1,” “PE2,” and “PE3” are colored with green, blue, yellow, and red, respectively. Since the distributed memory parallel algorithm does not repartition the analysis domain, the configuration remains the same during the whole simulation. Conversely, Fig. 21 visualizes the results of a domain repartition conducted by the time-based dynamic load balancing algorithm in the 2nd step. It is clear the domain colored with red or the one in charge of “PE3” becomes smaller. The calculation cost necessary for the ERP algorithm becomes much larger than the other domains, and thus the newly developed algorithm decreases the number of particles from the red domain of “PE3.”

5.2.2. Dam break problem

Figure 19 illuminates the results of the dynamic load balancing tests for dam break problem. In this problem, the number of domain repartitions becomes 0 and 4 by the original algorithm and proposed algorithm, respectively. As in the hydrostatic problem, “PE3” requires the longest amount of calculation time among the other processing elements. Due to the imbalance of calculation burden among the processing elements, it takes 3,508 [s] to execute the dam break simulation by the distributed memory parallel algorithm. Conversely, the proposed algorithm decreases the computational cost in “PE3,” and thus it distributes the EMPS and ERP calculation to the remaining processing elements. Compared to the original algorithm, the time-based dynamic load balancing
algorithm solves the problem for 2,874 [s] and saves 634 [s]. Note that the effect of the proposed algorithm becomes smaller in this problem because the dam-shaped fluid particles allocated asymmetrically in the container move forward to the rigid structure as opposed to the static fluid particles of the hydrostatic pressure problem.

5.3. Strong scaling efficiency tests

Strong scaling measures how computational time varies in a fixed problem size according to the number of processing elements. Strong scaling efficiency with \( p \) processing elements, \( E_p \), is calculated as

\[
E_p = \frac{2t_2}{pt_p},
\]

where \( t_p \) denotes the computational time with \( p \) processing elements. Note that HDDM_EMPS needs to execute programs with at least two processing elements. In this study, strong scaling efficiency tests are conducted with 2, 4, 16, 32, and 48 processing elements on the supercomputing system, PRIMEHPC FX100 (FX100) [20]. FX100 has SPARC64 XIfx with 32 processing elements per node, and the total number of nodes is over 100,000. On FX100, the problems are analyzed under the conditions described in Table 2.

Table 2: Conditions for hydrostatic pressure and dam break problems

| Conditions          | Hydrostatic pressure | Dam break       |
|---------------------|----------------------|-----------------|
| Number of particles | 392,196              | 400,000         |
| Simulation time     | \( 1.0 \times 10^{-4} \) [s] | \( 2.5 \times 10^{-4} \) [s] |
| Time step           | \( 1.0 \times 10^{-4} \) [s] | \( 1.0 \times 10^{-4} \) [s] |
| Particle distance   | \( 2.5 \times 10^{-2} \) [m] | \( 2.5 \times 10^{-2} \) [m] |
| Kinematic viscosity | \( 1.0 \times 10^{-6} \) [m²/s] | \( 1.0 \times 10^{-6} \) [m²/s] |
| Fluid density       | \( 1.0 \times 10^{7} \) [kg/m³] | \( 1.0 \times 10^{5} \) [kg/m³] |
| Effective radius    | \( 3.1 \times 10^{-2} \) [m] | \( 3.1 \times 10^{-2} \) [m] |
| Gravity             | -9.81 [m/s²]         | -9.81 [m/s²]    |

5.3.1. Hydrostatic pressure problem

Figure 22 exhibits the higher efficiency of the proposed algorithm. The proposed algorithm keeps 78.0% efficiency of strong scaling with 48 processing elements, but the original algorithm has only 14.0% efficiency with the same number of processing elements. On average, the strong scaling efficiency is approximately 88.2% by the newly developed algorithm with the 2, 4, 16, 32, and 48 processing elements. Conversely, it is about 43.6% by the conventional algorithm. Hence, the strong scaling efficiency tests for the static problem demonstrate more preferable parallel efficiency of the time-based dynamic load balancing algorithm than the distributed memory parallel algorithm.

5.3.2. Dam break problem

Figure 23 clarifies the strong scaling efficiency for the dam break problem. The proposed algorithm keeps 78.0% strong scaling efficiency with 48 processing elements, but the original algorithm has only 36.0% efficiency with the same number of processing elements. On average, the
Figure 22: Results of hydrostatic pressure

Figure 23: Results of dam break

strong scaling efficiency becomes approximately 88.2% by the algorithm designed in this study with the 2, 4, 16, 32, and 48 processing elements. Diversely, it becomes about 68.0% by the algorithm developed by Murotani et al. Therefore, the strong scaling efficiency tests for the dynamic problem also verify the higher parallel efficiency of the time-based dynamic load balancing algorithm than the distributed memory parallel algorithm.

6. Conclusion

This paper presents a time-based dynamic load balancing algorithm for the parallel free-surface flow analysis system, HDDM_EMPS, with the three-dimensional ERP wall boundary model. It first explains the ERP algorithm may cause dynamic load balancing problems if the number of polygons becomes greater and polygons are asymmetrically allocated in analysis domains.

Next, the paper illustrates how the proposed algorithm differentiates from the conventional algorithm. To decompose computational domains into processing elements, the time-based dynamic load balancing algorithm can give a variety of weight values based on the computational time necessary not only for the EMPS algorithm but also for the ERP algorithm. Besides, this algorithm repartitions analysis domains if the amount of computation exceeds a threshold based on this time. Conversely, the distributed memory parallel algorithm allocates mostly the same number particles to each processing element regardless of whether particles are incorporated in the ERP algorithm. Also, this algorithm repartitions analysis domains if the number of particles becomes unbalanced in a certain processing element.

Finally, the paper describes the time-based dynamic load balancing algorithm has a more potential to enhance the efficiency of parallel computing. It is confirmed by dynamic load balancing tests and strong scaling efficiency tests for hydrostatic pressure and dam break problems. In terms of dynamic load balancing tests, both the static and dynamic problems are simulated with four processing elements on a local computer. As a result, the proposed algorithm calculates them with less computational time than the original algorithm. The newly developed algorithm spends 1,753 [s] and 2,874 [s] for the total calculation of hydrostatic pressure problem and dam break problem, respectively. However, it takes 2470 [s] and 3508 [s] for the conventional algorithm to simulate these problems, respectively. Regarding strong scaling efficiency tests, those problems are utilized with 2, 4, 16, 32, and 48 processing elements on the supercomputing environment, FX100. The
newly developed algorithm illustrates 88.2% of strong scaling efficiency for the hydrostatic pressure problem on average, but the conventional algorithm demonstrates 43.6% efficiency. Besides, the proposed algorithm keeps approximately 88.2% of the strong scaling efficiency for the dam break problem on average, yet the original algorithm has about 68.0% of the efficiency.

Hence, compared to the conventional distributed memory parallel algorithm, the time-based dynamic load balancing algorithm has more potential to solve dynamic load balancing problems and improves the strong scaling efficiency in HDDM_EMPS with the three-dimensional ERP wall boundary model. In the future, our research groups plan to simulate tsunami behaviors on supercomputing systems. Although we utilize approximately 400,000 particles and 48 processing elements at maximum in this study, it is estimated tsunami simulations require more particles and processing elements for more accurate and time-efficient analysis.

Acknowledgement

This study was supported by JSPS KAKENHI Grant Number 18K18062. Also, we would like to appreciate Dr. Kohei Murotani at Railway Technical Research Institute for giving us insightful opinions about it.

References

[1] J.J. Monaghan: Smoothed Particle Hydrodynamics, *Annual Review of Astronomy and Astrophysics*, 30 (1992), 543–574.

[2] S. Koshizuka, Y. Oka: Moving-Particle Semi-Implicit method for fragmentation of incompressible fluid, *Nuclear Science and Engineering*, 123 (1996), 421–434.

[3] M. Oochi, S. Koshizuka, M. Sakai: Explicit MPS algorithm for free surface flow analysis, *Transactions of the Japan Society for Computational Engineering and Science*, 2010 (2010), 20100013. (in Japanese)

[4] M. Oochi, Y. Yamada, S. Koshizuka, M. Sakai: Validation of pressure calculation in Explicit MPS method, *Transactions of the Japan Society for Computational Engineering and Science*, 2011 (2011), 20110002. (in Japanese)

[5] S. Koshizuka, A. Nobe, Y. Oka: Numerical analysis of breaking waves using the Moving Particle Semi-implicit method, *International Journal for Numerical Method in Fluids*, 26 (1998), 751–769.

[6] T. Matsunaga, K. Shibata, K. Murotani, S. Koshizuka: Fluid flow simulation using MPS method with mirror particle boundary representation, *Transactions of the Japan Society for Computational Engineering and Science*, 2016 (2016), 20160002. (in Japanese)

[7] T. Harada, S. Koshizuka, K. Shimazaki: Improvement of wall boundary calculation model for MPS method, *Transactions of the Japan Society for Computational Engineering and Science*, 2008 (2008), 20080006. (in Japanese)

[8] Y. Yamada, M. Sakai, S. Mizutani, S. Koshizuka, M. Oochi, K. Murozono: Numerical simulation of three-dimensional free-surface flows with Explicit Moving Particle Simulation method, *Transactions of the Atomic Energy Society of Japan*, 10:3 (2011), 185–193. (in Japanese)
N. Mitsume, S. Yoshimura, K. Murotani, T. Yamada: Explicitly Represented Polygon wall boundary model for the Explicit MPS method, *Computational Particle Mechanics*, 2:1 (2015), 73–89.

K. Murotani, S. Koshizuka, T. Tamai, K. Shibata, N. Mitsume, S. Yoshimura, S. Tanaka, K. Hasegawa, E. Nagai, T. Fujisawa: Development of hierarchical domain decomposition Explicit MPS method and application to large-scale tsunami analysis with floating objects, *Journal of Advanced Simulation in Science and Engineering*, 1:1 (2014), 16–35.

A. Ferrari, M. Dumbser, E.F. Toro: A new 3D parallel SPH scheme for free surface flows, *Computers and Fluids*, 38 (2009), 1203–1217.

K. Murotani, M. Oochi, T. Fujisawa, S. Koshizuka, S. Yoshimura: Distributed memory parallel algorithm for Explicit MPS using ParMETIS, *Transactions of the Japan Society for Computational Engineering and Science*, 2012 (2012), 20120012. (in Japanese)

Y. Mizuno, N. Mitsume, T. Yamada, S. Yoshimura: Dynamic load balancing for parallel MPS analysis adopting polygon wall boundary model, in *Proceedings of Computational Engineering Conference 23rd*, Aichi, 2018.

Y. Mizuno, N. Mitsume, T. Yamada, S. Yoshimura: Dynamic load balancing algorithm for particle methods based on computational cost of respective particles, in *The Japan Society of Mechanical Engineers 31st Computational Mechanics Division Conference*, Tokushima, 2018. (in Japanese)

J.P. Morris, P.J. Fox, Y. Zhu: Modeling low Reynolds number incompressible flows using SPH, *Journal of Computational Physics*, 136:1 (1997), 214–226.

G. Karypis, V. Kumar: Multilevel k-way partitioning scheme for irregular graphs, in *Technical Report*, University of Minnesota, 1995, TR 95-064.

G. Karypis, V. Kumar: Parallel multilevel series k-way partitioning scheme for irregular graphs, *Siam Review*, 41:2 (1996), 278–300.

G. Karypis, V. Kumar: A fast and highly quality multilevel scheme for partitioning irregular graphs, *SIAM Journal on Scientific Computing*, 20:1 (1999), 359–392.

G. Karypis, K. Schloegel: ParMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library Version 4.0, in *Technical Report*, University of Minnesota, 2013.

FUJITSU Supercomputer PRIMEHPC FX100. Retrieved March 14, 2019, from http://www.fujitsu.com/global/products/computing/servers/supercomputer/primehpc-fx100/