Chapter 5

Development of a Large-Scale 2D-3D Hybrid Model

5.1 Introduction

In the Chapter 4, the 2D-3D hybrid model has been developed. However, in order to simulate large-scale tsunami travel over a real terrain, the parallel computing is needed for the 2D-3D hybrid model. Therefore, the purpose of this chapter is to develop a parallel computing 2D-3D hybrid model using the Message Passing Interface (MPI) method [82].

5.2 Parallel Computing Method

5.2.1 Domain Decomposition Method

For the domain decomposition method, the analysis domain is divided into plurality of subdomains, then processors are assigned to each subdomain to the computation in parallel. There are two main kinds of domain decomposition method, the first is node-based method that equally divides the number of nodes (see **Figure 5.1**(1)), the second is element-based method that equally divides the number the elements (see **Figure 5.1**(2)). In the case of using the finite element method, most of the computation are based on element, so the element-based method is generally used. In this study, The METIS software [83] using the element-based method is applied for domain decomposition.

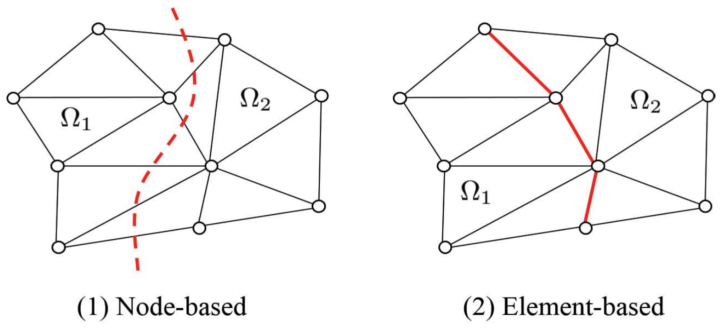


Figure 5.1 **Domain decomposition method**

5.2.2 Classification of Parallel Computing Method

Parallel computer can be classified by the use of memory (see **Figure 5.2**). A shared memory type (**Figure 5.2**(a)) that all the processors (CPUs) share a same memory. A distributed memory type (**Figure 5.2**(b)) that processors can communicate with other processors by calling library routines to send and receive messages. A shared-distributed memomory type (**Figure 5.2**(c)) is a combination of both the former two types.

MPI, OpenMP (Open Multi-processing) [84],[85] and MPI-OpenMP hybrid method are the gerneral methods for the parallel computing. One of the advantage for OpenMP is that it can be parallelized easily by placing OpenMP directives in time loops. But its drawback is that it is not easy for optimizing workflow and memory access. [86] The OpenMP is suitable for the shared memory type computer. For the MPI, it is necessary to reduce the communication load in order to obain a high efficiency. However, one of the advantage for MPI is that it can deal with much larger scale problem than OpenMP by increasing the number of processors. The MPI is suitable for all the three types of parallel computer mentioned above. MPI-OpenMP hybrid method is suitable for the shared-distributed memomory type computer. However, the algorithm is complicate. Therefore, in order to deal with large-scale tsunami simulation, the MPI is used to parallel computing in this study.

The computer used in this study is the supercomputer system CRAY XC40 [87] owned by Kyoto University. The performance of CRAY XC40 is shown in **Table 5.1**. The CRAY XC40 is a shared-distributed memomory type.

5.2.3 Parallel Programing Using MPI

The data communication between processes in MPI is by inserting appropriate subroutines into the source code. These subroutines can be roughly divided into four classes [82]:

1. Environment management subroutines.

   This class of subroutines are used for initializing the MPI environment, identifying the number of processors being used, terminating the MPI environment, etc.

2. Point to point communication subroutines.

   This class of subroutines are used for sending and receiving message between two processors.

3. Collective communication subroutines.

   This class of subroutines are used for communications among processors.

4. Others.

   The others are such as the evaluting subroutine for computational time.

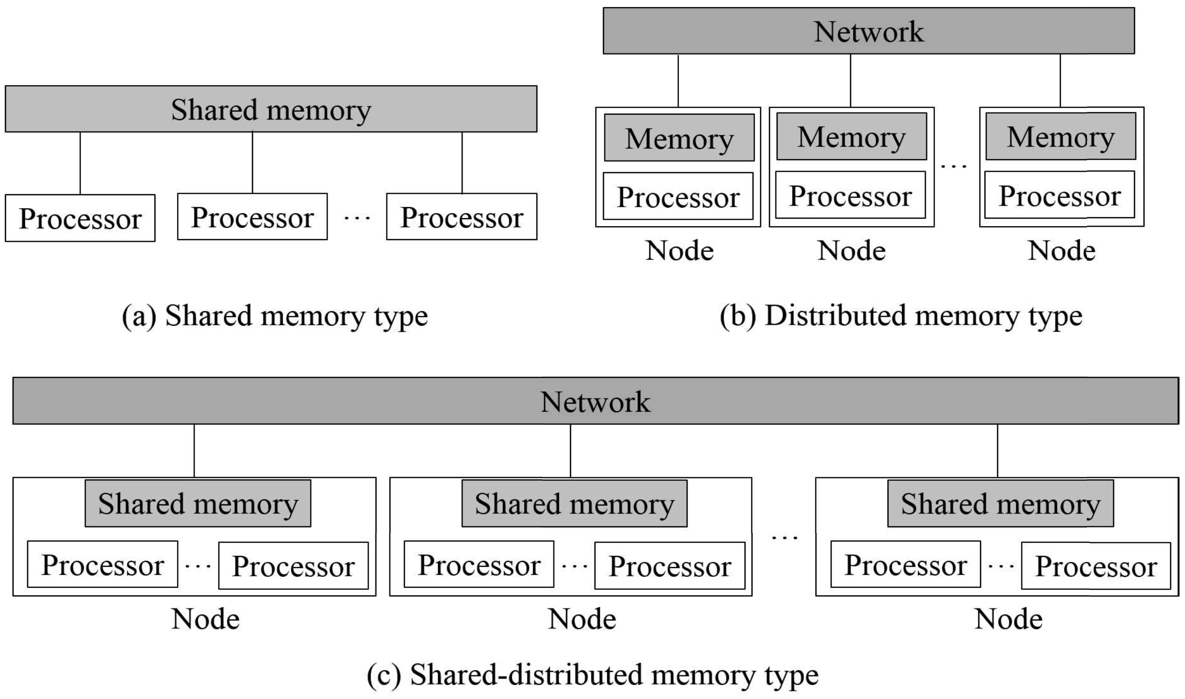


Figure 5.2 **Structures of computer system**

Table 5.1 **Computer performance**

CRAY XC40	
CPU	Intel Xeon Phi (KNL) 1.4GHz
Total cores	68 cores × 1,800 nodes
Memory capacity (1 node)	112 GB
Peak performance (1 node)	3.05 TFops
OS	Cray Compute Node Linux

5.2.4 Parallel Implementation of the Bi-CGSTAB Method

In the finite element method, after the spatial discretization, temperal discretization and the preconditioning (diagonal scaling) [60] of the governing equatins, the finite element equations can be written into the following form,

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{5.1}$$

This is a set of simultaneous linear equations. In this study, the Bi-Conjugate Gradient Stabilized (Bi-CGSTAB) [62],[63] method which can solve nonsymmetric linear systems is applied to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$. The element-by-element treatment [61] is applied to save memory and to solve large-scale problem. The algorithm for the parallel implementation of the Bi-CGSTAB method is shown below,

1. Select an $\mathbf{x}$ and $\mathbf{r}_0^*$, e.g., $\mathbf{x}$=$\mathbf{0}$ and $\mathbf{r}_0^*$ is set to random vector.
2. Compute $\mathbf{r}_0$=$\mathbf{b}$-$\underline{\mathbf{A}\mathbf{x}}_{(1)}$, $\mathbf{p}_0$=$\mathbf{r}_0$
3. do $k = 0$, $kmax$
4. $$\alpha_k = \frac{\utilde{(\mathbf{r}_0^*, \mathbf{r}_k)}_{(2)}}{\utilde{(\mathbf{r}_0^*, \underline{\mathbf{A}\mathbf{p}_k)}_{(1)}}_{(2)}} \tag{5.2}$$
5. $$\mathbf{t}_k = \mathbf{r}_k - \alpha_k \underline{\mathbf{A}\mathbf{p}_k}_{(1)} \tag{5.3}$$
6. $$\zeta_k = \frac{\utilde{(\underline{\mathbf{A}\mathbf{t}_k}_{(1)}, \mathbf{t}_k)}_{(2)}}{\utilde{(\underline{\mathbf{A}\mathbf{t}_k}_{(1)}, \underline{\mathbf{A}\mathbf{t}_k}_{(1)})}_{(2)}} \tag{5.4}$$
7. $$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k + \zeta_k \mathbf{t}_k \tag{5.5}$$
8. $$\mathbf{r}_{k+1} = \mathbf{t}_k - \zeta_k \underline{\mathbf{A}\mathbf{t}_k}_{(1)} \tag{5.6}$$

```
 9. if $||\mathbf{r}_{k+1}||/||\mathbf{b}|| \leq stop\_tol$ then
10. write $\mathbf{x}_{k+1}$
11. else
```

$$\beta_k = \frac{\alpha_k}{\zeta_k} \cdot \frac{\underset{\sim\sim\sim}{(\mathbf{r}_0^*, \mathbf{r}_{k+1})}_{(2)}}{\underset{\sim\sim\sim}{(\mathbf{r}_0^*, \mathbf{r}_k)}_{(2)}} \tag{5.7}$$

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \beta_k(\mathbf{p}_k - \zeta_k \underline{\mathbf{A}\mathbf{p}_k}_{(1)}) \tag{5.8}$$

```
12. end if
13. end do
```

where *kmax* is the maximum number of allowed iterations, *stop_tol* is the relative convergence tolerance. The underlines of (1),(2) indicate the computation with communication. (1) uses the communication between neighboring subdomains (point to point communication). (2) is the inner product part of vector, the sum of them is computed by the communication for all the subdomains (collective communication). For the nodes belong to multiple subdomains, the number N of multiple subdomains should be counted, and the sum of the inner product part of vector is multplied by $1/N$ to exclude duplication.

5.3 Parallelization of the 2D-3D Overlapping Method

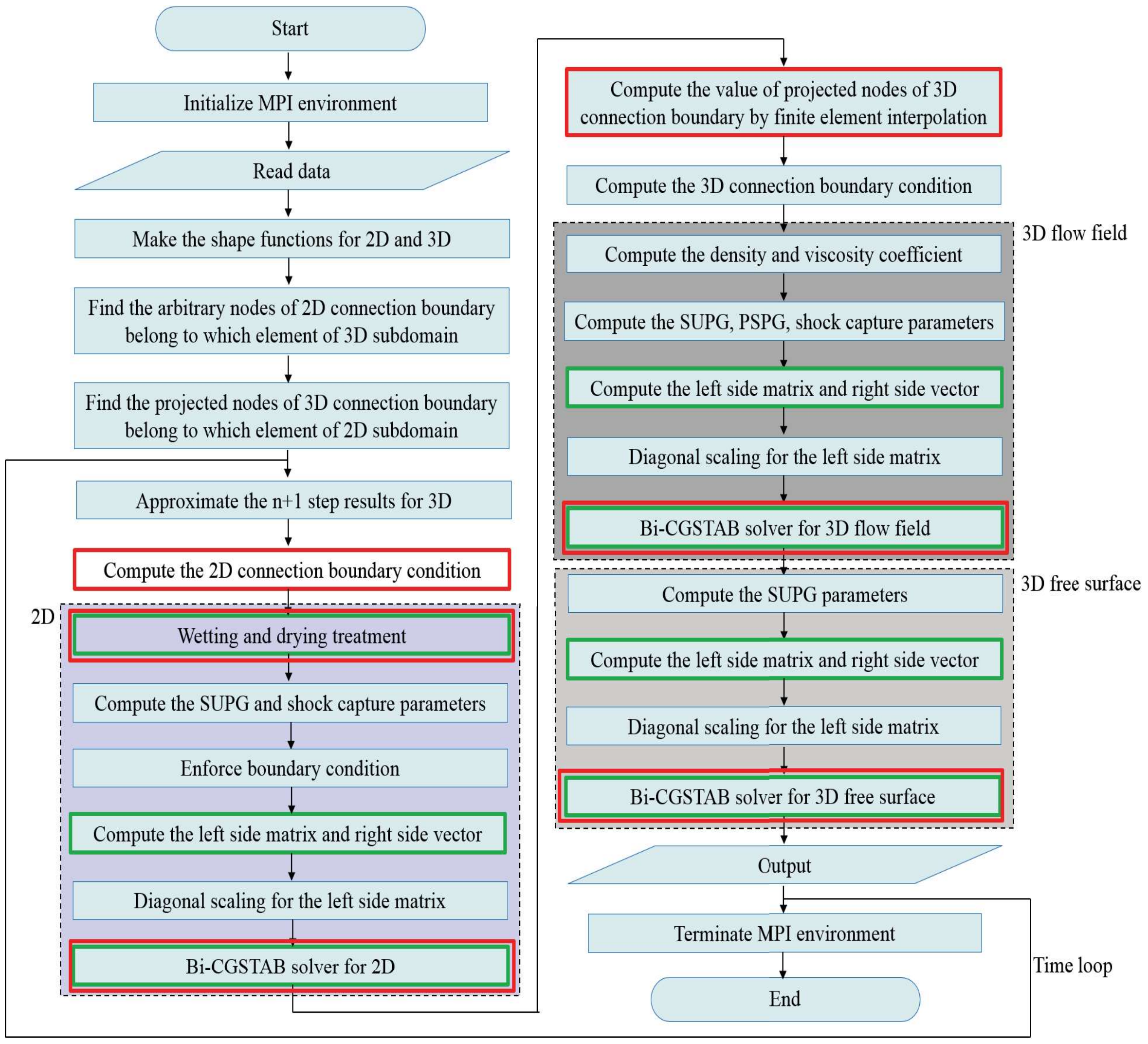

Start
Initialize MPI environment
Read data
Make the shape functions for 2D and 3D
Find the arbitrary nodes of 2D connection boundary belong to which element of 3D subdomain
Find the projected nodes of 3D connection boundary belong to which element of 2D subdomain
Approximate the n+1 step results for 3D
Compute the 2D connection boundary condition
2D
Wetting and drying treatment
Compute the SUPG and shock capture parameters
Enforce boundary condition
Compute the left side matrix and right side vector
Diagonal scaling for the left side matrix
Bi-CGSTAB solver for 2D
Compute the value of projected nodes of 3D connection boundary by finite element interpolation
Compute the 3D connection boundary condition
3D flow field
Compute the density and viscosity coefficient
Compute the SUPG, PSPG, shock capture parameters
Compute the left side matrix and right side vector
Diagonal scaling for the left side matrix
Bi-CGSTAB solver for 3D flow field
3D free surface
Compute the SUPG parameters
Compute the left side matrix and right side vector
Diagonal scaling for the left side matrix
Bi-CGSTAB solver for 3D free surface
Output
Terminate MPI environment
End
Time loop

Figure 5.3 **Flowchart with parallel implementation (green frame: communication with neighboring subdomains; red frame: communication for all the subdomains)**

Figure 5.3 shows the flowchart with parallel implementation for the 2D-3D overlapping method. In the figure, the green frame indicates the step needs to do communication with the neighboring subdomain (point to point communication), the red frame indicates that the step needs to do communication with all the subdomains (collective communication). About this flowchart, the computation of the 2D/3D connection boundary conditions with parallel implementation and the parallelization for the wetting and drying treatment are introduced in the following sections.

5.4 Parallel Implementation of the 2D/3D Connection Boundary

Firstly, to compute the 2D connection boundary condition. **Figure 5.4** shows the decomposed meshes around the overlap domain for parallel computing. **Figure 5.4**(A) shows the bottom mesh, **Figure 5.4**(B) shows the bird's eye view of 2D-3D mesh. In the **Figure 5.4**, ⓘ (i=1,2,3) denotes the number of subdomains. **Figure 5.5** shows the conceptual diagram for computing the flow velocity and the water depth of an arbitrary node P at the 2D connection boundary. For the preprocess, auxiliary nodes in the same distance above the nodes of the 2D connection boundary are set. Then to search and to memorize which elements of 3D domain the auxiliary nodes belong to. **Figure 5.6** shows the positions of an auxiliary node and a tetrahedron element, to judge whether the auxiliary node n belong to the tetrahedron element $ijkl$, the volumes of the new tetrahedron $ijkn, njkl, inkl, ijnl$ are computed by the following equations.

$$\begin{aligned} V_{ijkn} = \frac{1}{6} \times [&(x_j - x_i)(y_k - y_i)(z_n - z_i) + (y_j - y_i)(z_k - z_i)(x_n - x_i) \\ &+(z_j - z_i)(x_k - x_i)(y_n - y_i) - (x_j - x_i)(z_k - z_i)(y_n - y_i) \\ &-(z_j - z_i)(y_k - y_i)(x_n - x_i) - (y_j - y_i)(x_k - x_i)(z_n - z_i)] \end{aligned} \tag{5.9}$$

$$\begin{aligned} V_{njkl} = \frac{1}{6} \times [&(x_j - x_n)(y_k - y_n)(z_l - z_n) + (y_j - y_n)(z_k - z_n)(x_l - x_n) \\ &+(z_j - z_n)(x_k - x_n)(y_l - y_n) - (x_j - x_n)(z_k - z_n)(y_l - y_n) \\ &-(z_j - z_n)(y_k - y_n)(x_l - x_n) - (y_j - y_n)(x_k - x_n)(z_l - z_n)] \end{aligned} \tag{5.10}$$

$$\begin{aligned} V_{inkl} = \frac{1}{6} \times [&(x_n - x_i)(y_k - y_i)(z_l - z_i) + (y_n - y_i)(z_k - z_i)(x_l - x_i) \\ &+(z_j - z_i)(x_k - x_i)(y_l - y_i) - (x_n - x_i)(z_k - z_i)(y_l - y_i) \\ &-(z_j - z_i)(y_k - y_i)(x_l - x_i) - (y_n - y_i)(x_k - x_i)(z_l - z_i)] \end{aligned} \tag{5.11}$$

$$\begin{aligned} V_{ijnl} = \frac{1}{6} \times [&(x_j - x_i)(y_n - y_i)(z_l - z_i) + (y_j - y_i)(z_n - z_i)(x_l - x_i) \\ &+(z_j - z_i)(x_n - x_i)(y_l - y_i) - (x_j - x_i)(z_n - z_i)(y_l - y_i) \\ &-(z_j - z_i)(y_n - y_i)(x_l - x_i) - (y_j - y_i)(x_n - x_i)(z_l - z_i)] \end{aligned} \tag{5.12}$$

If $V_{ijkn} \geq 0.0$, $V_{njkl} \geq 0.0$, $V_{inkl} \geq 0.0$ and $V_{ijnl} \geq 0.0$, the node n belongs to the the tetrahedron element $ijkl$, Then the flow velocity and the water depth of the auxiliary node n are computed by making linear interpolation of the belonging element $ijkl$ (refer to Chapter 4). In this study, the distance of auxiliary nodes is decided by the size of mesh. The physical quantities of the node P are computed by integrating the value of the auxiliary nodes in the vertical direction. In addition, as shown in the **Figure 5.5**, when the auxiliary nodes in the vertical direction exist in multiple regions, the value obained from each small region is communitcated to the small region which the node P belongs (collective communication). In the case when the auxiliary node is shared by multiple regions, the duplication is excluded by diving by the number of multiple regions.

Secondly, to compute the physical quantities of the arbitrary node Q at the 3D connection boundary (see **Figure 5.4**(B)), the first step is to collect all the informations of 2D subdomains to every 3D subdomain by collective communication, then to seach the projected node Q' on the 2D domain belong to which element. To jugde whether the node Q' belongs to an triangle element ABC (see **Figure 5.7**), the areas of the new triangle ABQ', BCQ', CAQ' are computed by the following equations,

$$A_{\mathrm{ABQ'}} = \frac{1}{2} \times [(x_{\mathrm{A}} - x_{\mathrm{B}})(y_{\mathrm{A}} - y_{\mathrm{Q'}}) - (x_{\mathrm{A}} - x_{\mathrm{Q'}})(y_{\mathrm{A}} - y_{\mathrm{B}})] \tag{5.13}$$

$$A_{\mathrm{BCQ'}} = \frac{1}{2} \times [(x_{\mathrm{B}} - x_{\mathrm{C}})(y_{\mathrm{B}} - y_{\mathrm{Q'}}) - (x_{\mathrm{B}} - x_{\mathrm{Q'}})(y_{\mathrm{B}} - y_{\mathrm{C}})] \tag{5.14}$$

$$A_{\mathrm{CAQ'}} = \frac{1}{2} \times [(x_{\mathrm{C}} - x_{\mathrm{A}})(y_{\mathrm{C}} - y_{\mathrm{Q'}}) - (x_{\mathrm{C}} - x_{\mathrm{Q'}})(y_{\mathrm{C}} - y_{\mathrm{A}})] \tag{5.15}$$

If $A_{\mathrm{ABQ'}} \geq 0.0$, $A_{\mathrm{BCQ'}} \geq 0.0$, $A_{\mathrm{CAQ'}} \geq 0.0$, the node Q' belongs to the triangle element ABC. Then the physical quantities of the node Q' are computed by making linear interpolation of the belonging element. Finally, the value of node Q' is given to the node Q.

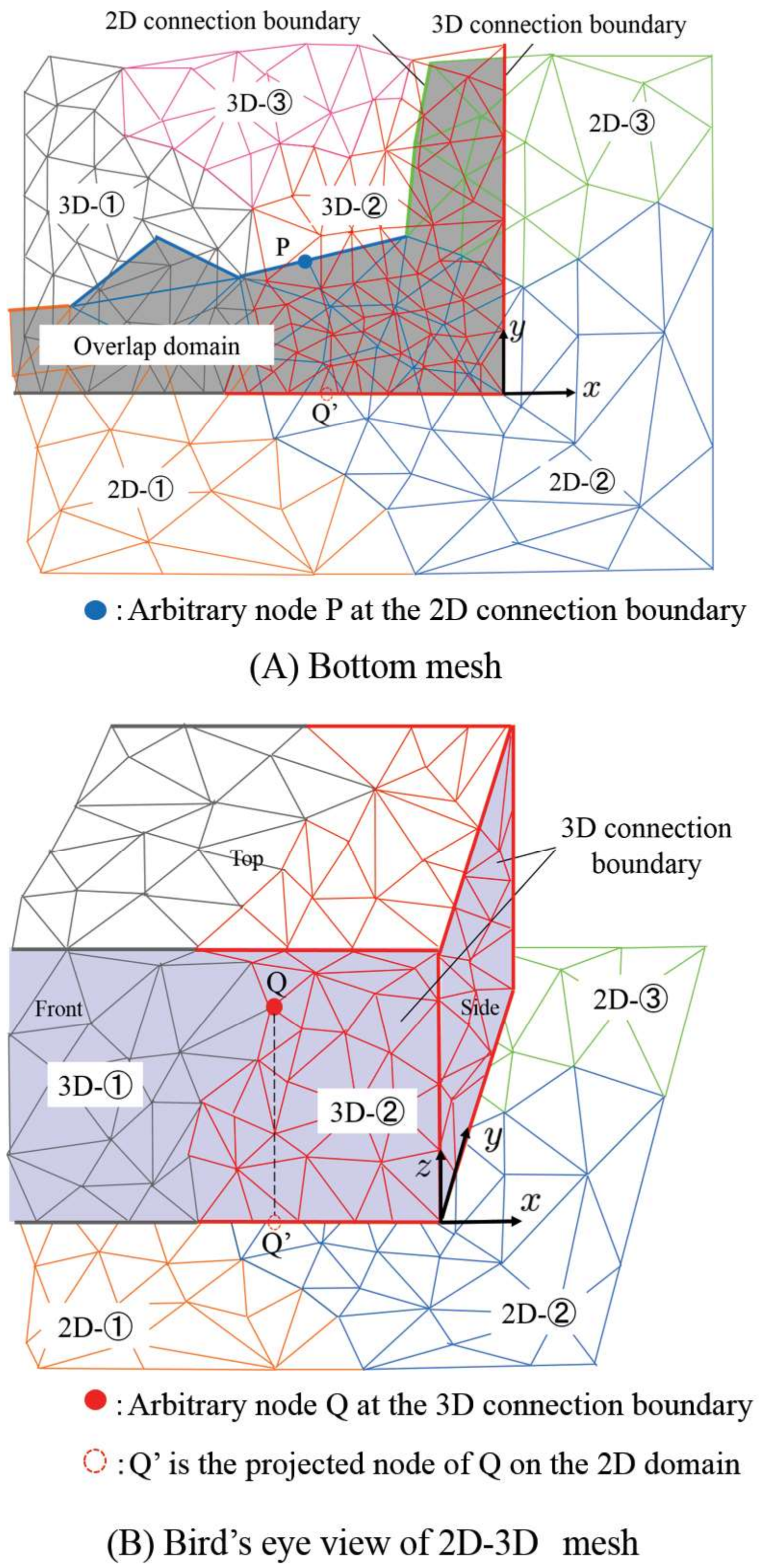

2D connection boundary
3D connection boundary
3D-③
2D-③
3D-①
3D-②
P
Overlap domain
y
x
Q'
2D-①
2D-②
● : Arbitrary node P at the 2D connection boundary
(A) Bottom mesh
Top
3D connection boundary
Front
Q
Side
2D-③
3D-①
3D-②
y
z
x
Q'
2D-①
2D-②
● : Arbitrary node Q at the 3D connection boundary
○ : Q' is the projected node of Q on the 2D domain
(B) Bird's eye view of 2D-3D mesh

Figure 5.4 **Separated region for the mesh around the 2D-3D overlap domain**

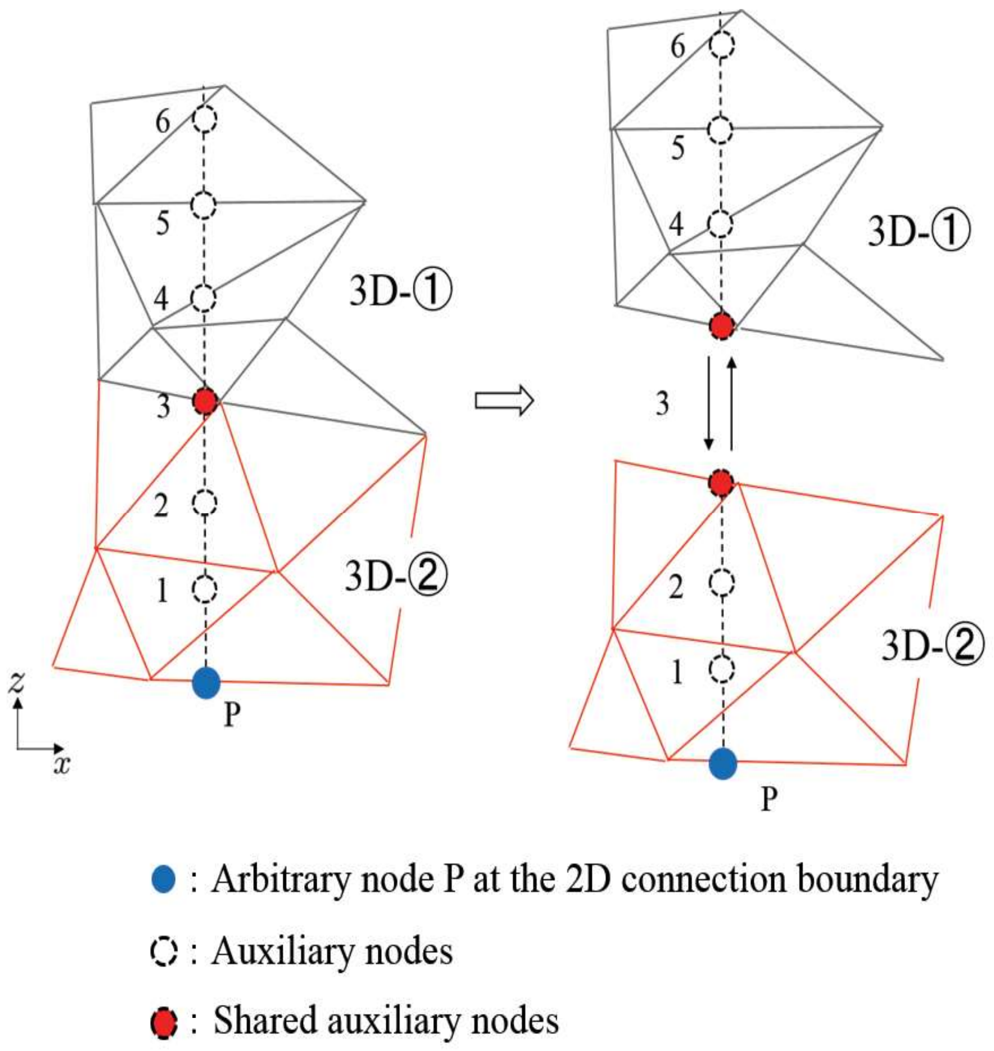
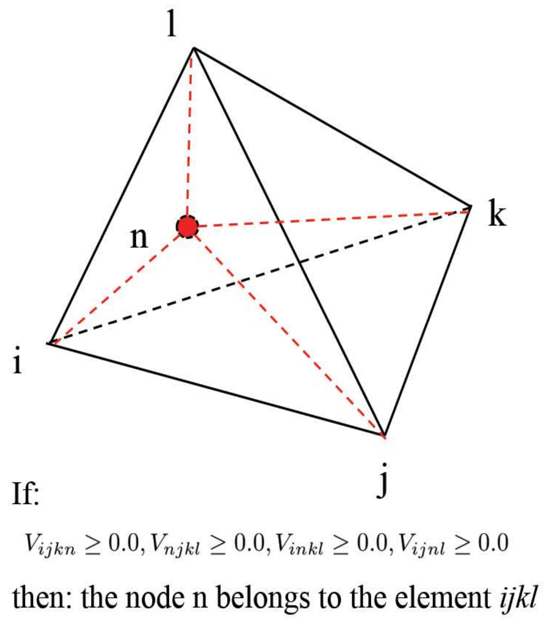

6
5
4
3
2
1
3D-①
3D-②
P
z
x
3
● : Arbitrary node P at the 2D connection boundary
◌ : Auxiliary nodes
● : Shared auxiliary nodes

Figure 5.5 **Computation for the physical quantities of the arbitrary node P at the 2D connection boundary**

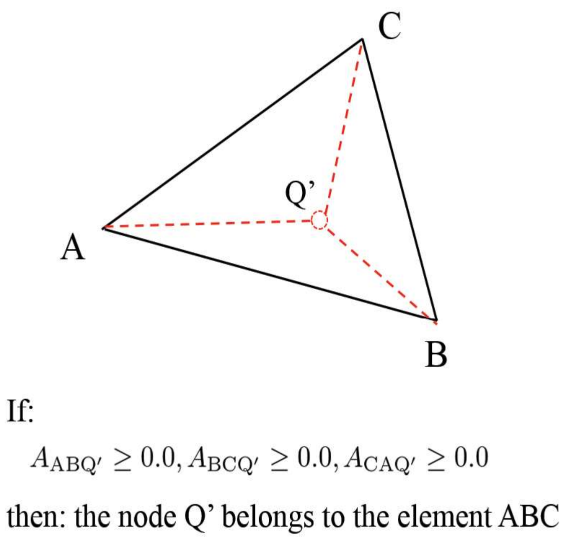


If:

$V_{ijkn} \geq 0.0, V_{njkl} \geq 0.0, V_{inkl} \geq 0.0, V_{ijnl} \geq 0.0$

then: the node n belongs to the element *ijkl*

Figure 5.6 **Judgement for whether an node belongs to a tetrahedron element**



If:

$A_{\mathrm{ABQ'}} \geq 0.0, A_{\mathrm{BCQ'}} \geq 0.0, A_{\mathrm{CAQ'}} \geq 0.0$

then: the node Q' belongs to the element ABC

Figure 5.7 **Judgement for whether an node belongs to a triangle element**

5.5 Parallel Implementation of the Wetting and Drying Treatment

In this study, the parallelization for the wetting and drying treatment is based on the Eulerian method which we mentioned in Chapter 2. For the preprocessing, auxiliary nodes in the same distance are set at the bottom of the 3D connection boundary, and to remember the belonging elements and small regions. Then to do the following steps by every time step:

(1) To check all the elements of the 2D domain to decide the dry elements, the wet elements and the shore elements (refer to Chapter 2, section 2.6). **Figure 5.8**(A) shows the case that when the wave is only in the 2D domain. In this case, we can just compute all the computational domain by 2D model. In the left figure, the blue region denotes the wetting region. In the right figure, the blue line denotes the front of the wave, and the red region denotes the computational elements. Attention should be payed to the black nodes which are shared by the neighboring subdomains, it may be the node of a dry element in a subdomain but it may be the node of a wet element in the other neighboring subdomain, thus the communication with the neighboring subdomain should be done.

(2) To check the water depth of the auxiliary nodes at the 3D connection boundary. If the water depth of any auxiliary node is higher or lower than the initial depth, such as **Figure 5.8**(B), then using the collective communication to let the other processors to start the 2D-3D hybrid computation. In addiction, for this case the gray region is excluded from the 2D computational domain.

(3) Attention should be payed to the duplication of the black nodes in the **Figure 5.8** in the inner product calculation of the simultaneous linear equations (collective communication).

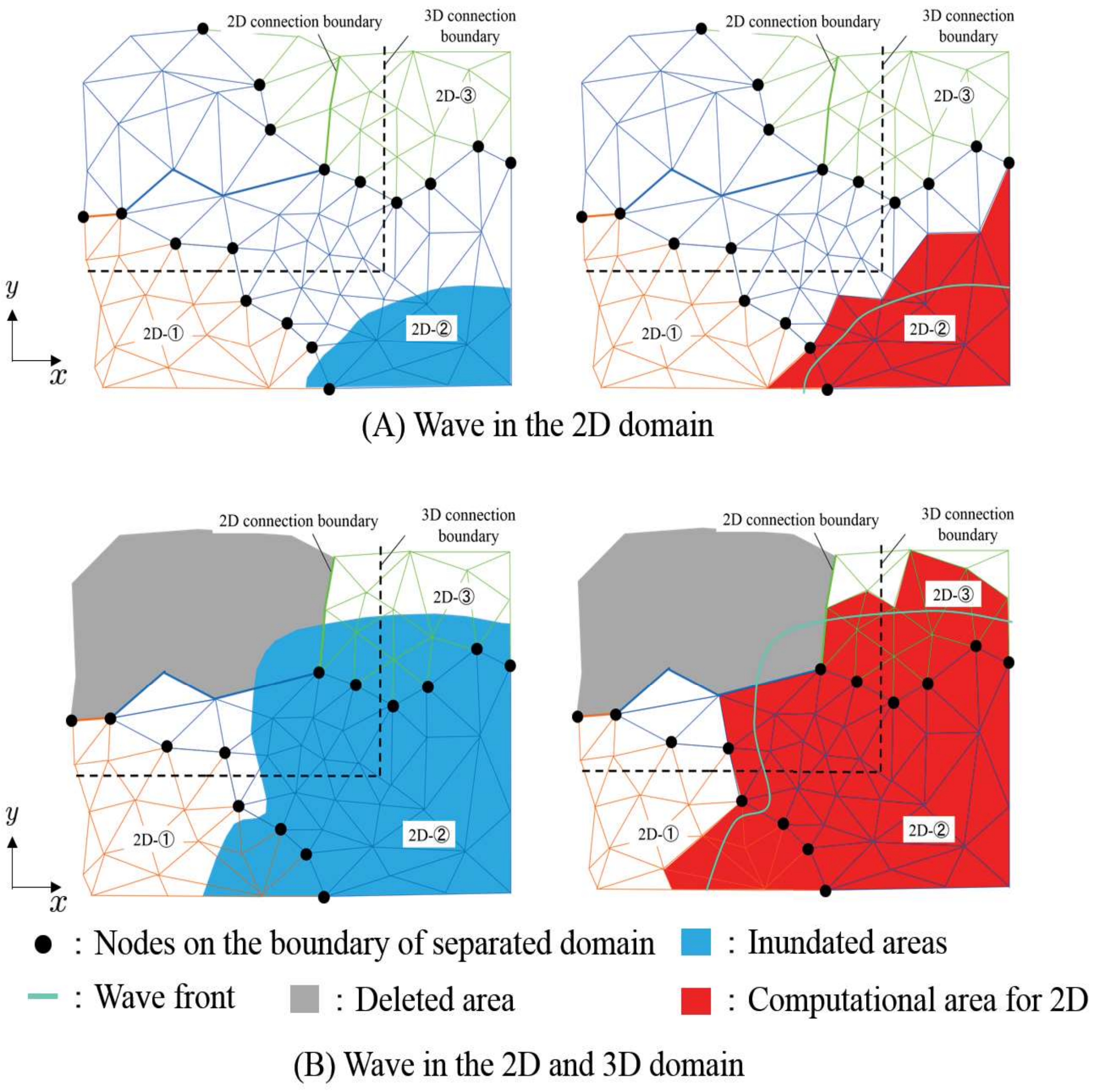

2D connection boundary
3D connection boundary
2D-①
2D-②
2D-③
y
x
(A) Wave in the 2D domain
● : Nodes on the boundary of separated domain
: Inundated areas
: Wave front
: Deleted area
: Computational area for 2D
(B) Wave in the 2D and 3D domain

Figure 5.8 **Wetting and drying treatment**

5.6 Numerical Example

5.6.1 Tsunami Simulation on Real Terrain

In order to test the applicability of the large-scale 2D-3D hybrid tsunami numerical model to the real terrain, the computational model showed in **Figure 5.9** is simulated. For this example, the area around the Onagawa town is computed by 3D while the other area by 2D (see **Figure 5.10**, an observer is set on the hill for the view of visualization).

The initial conditions and the boundary conditions for the 2D domain are the same with the numerical example shown in the Chapter 2. The mesh information of the 2D domain and 3D domain is shown in **Table 5.2**. The computational domain is separated into 512 small subdomains by METIS (See **Figure 5.11**). The time increment is 0.1s. The supercomputer system (Cray XC40) of Kyoto University is used for computing.

Table 5.2 **Mesh information**

	2D	3D
number of nodes	557,252	23,232,416
number of elements	1,103,938	134,768,640

Figure 5.12 shows the results at t=720s, 2000s of 2D domain, we can see the wave propagating from source area to shore area stably. **Figure 5.13** shows the computational results from the observer, the green denotes the results of 2D domain and the blue denotes the results of 3D domain. In **Figure 5.13**(a), the tsunami wave hasn't reached the 3D connection boundary, all the domain is computed by 2D model. After 1,920s, the tsunami wave reaches the 3D connection boundary and the hybrid analysis is began, we can see the building are flooded in **Figure 5.13**(b)(c). About the computational time, for the hybrid model introduced in the Chapter 4 that the 2D domain is sequential computing and the 3D domain is parallel computing, it takes 184,895s (about 9.67s for 1 step) for the wave propagate to the 3D domain. And when the hybrid model begins to work, the proportion of computational time for 2D and 3D is about 35% for 2D and about 65% for 3D. The computational time for 1 step of 2D is about 27s and 3D is about 49s (the average value of 10 steps). On the other

hand, by the parallel method introduced in this Chapter that both the 2D domain and 3D domain are parallel computing, it takes 2,247s (about 0.12s for 1 step which is about 1/80 of the sequential computing) for the wave reaches the 3D domain. And when the hybrid model begins to work, the proportion of computational time for 2D and 3D is about 22% for 2D and about 78% for 3D. The computational time for 1 step of 2D is about 15s and 3D is about 52s (the average value of 10 steps).

The efficiency and the applicability of the present method to the real terrain has been confirmed by this numerical example.

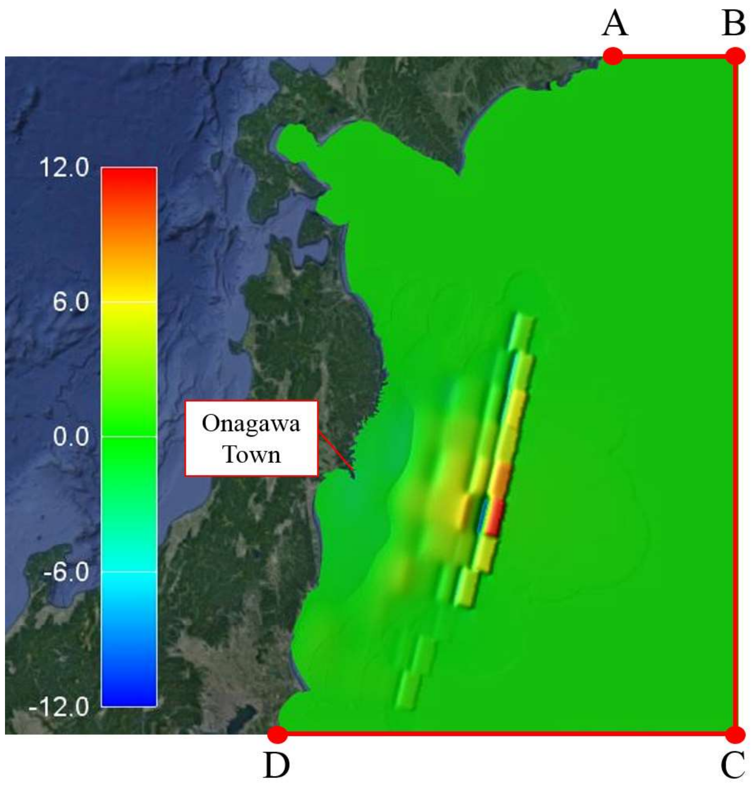


Figure 5.9 **Computational model (initial condition)**

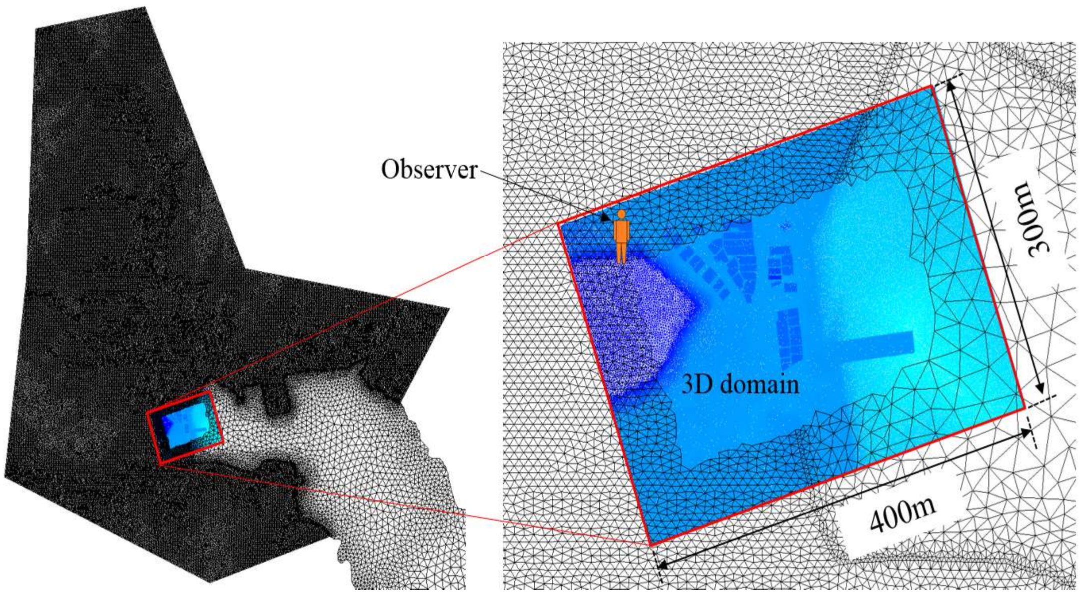
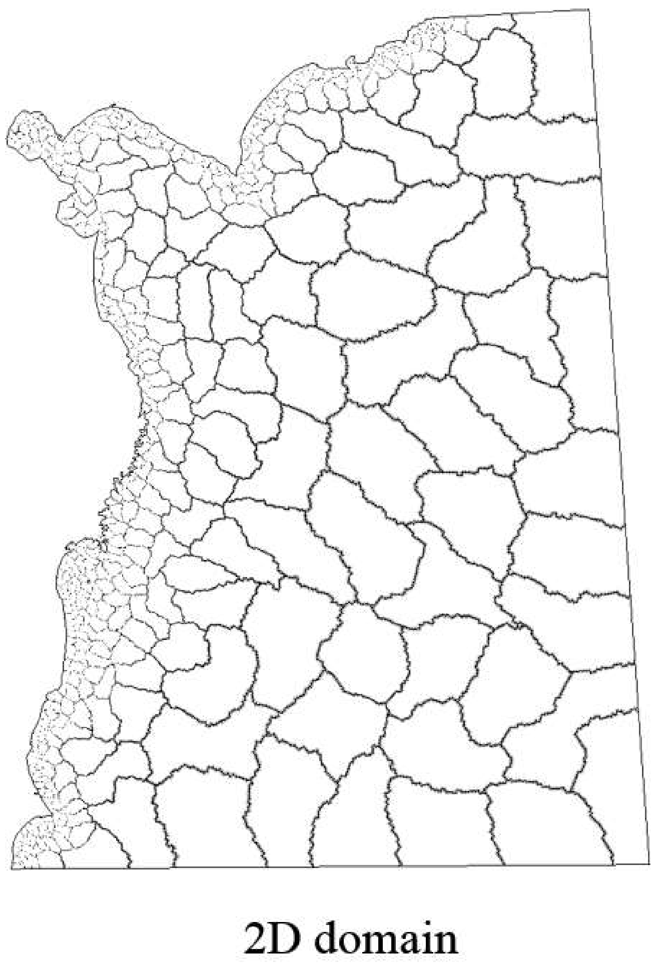


Figure 5.10 **Meshes around 3D domain (Onagawa town) and the position of observer**



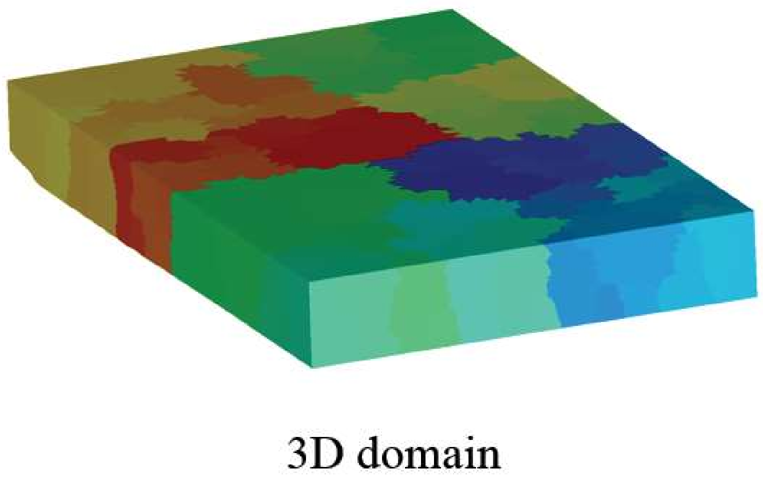
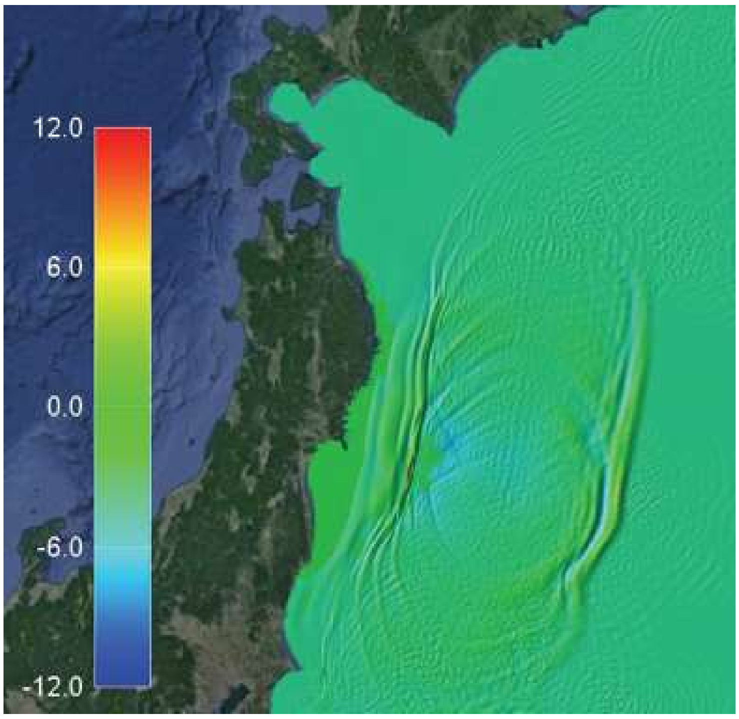
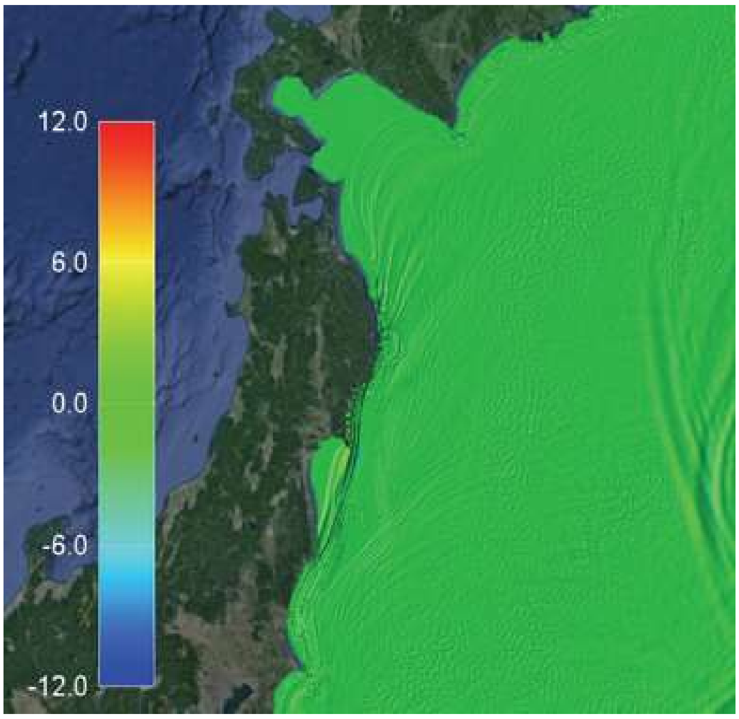


Figure 5.11 **Region segmentation (512)**

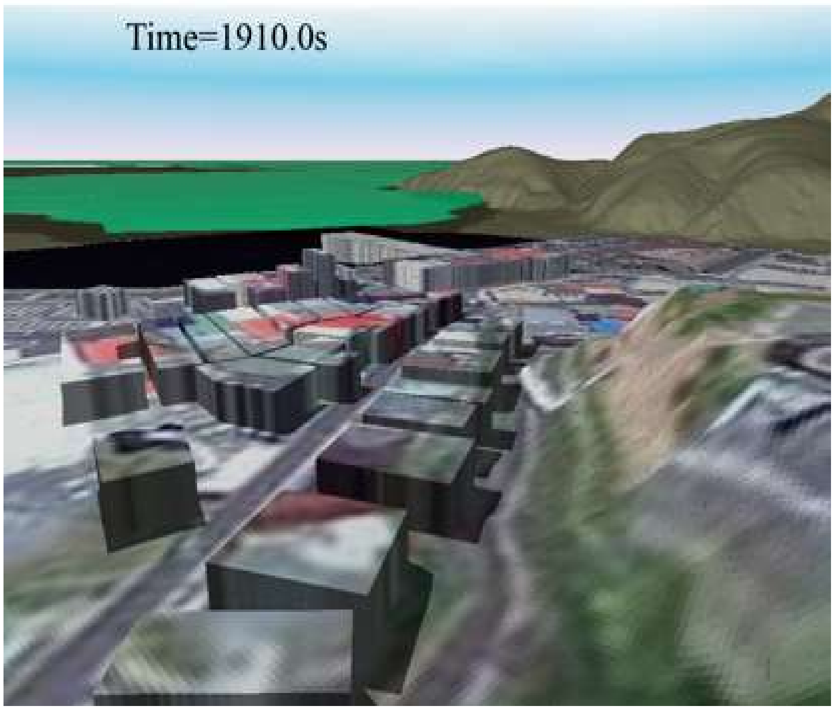


(a) $t = 720s$

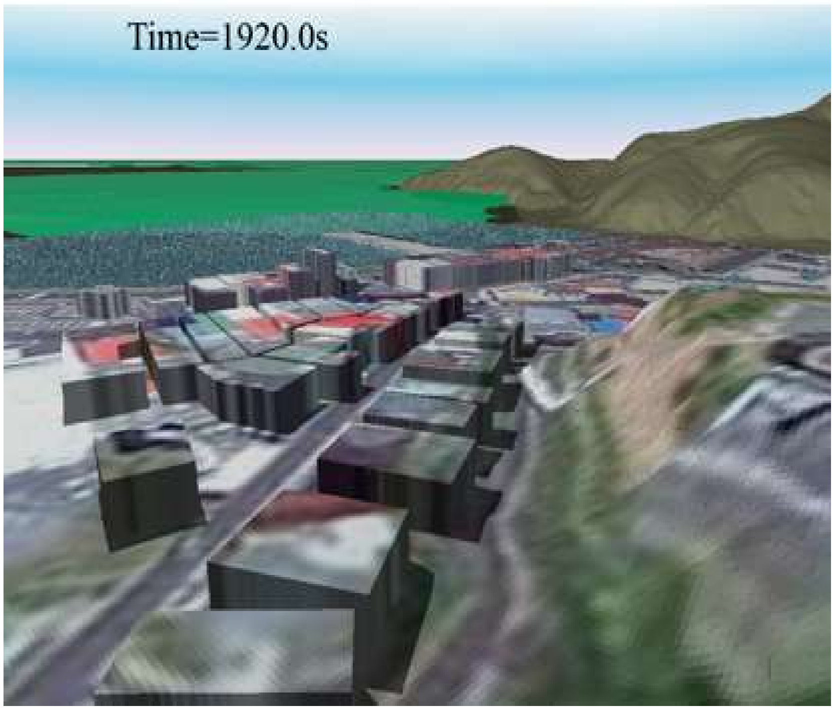


(b) $t = 2000s$

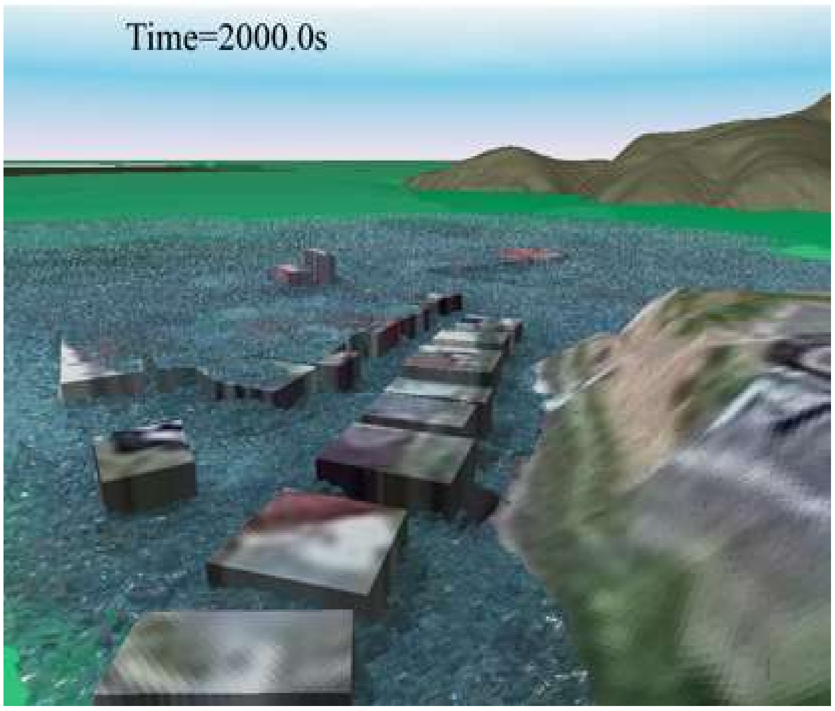
Figure 5.12 **Computational results**



(a) $t = 1910s$



(b) $t = 1920s$



(c) $t = 2000s$

Figure 5.13 **Computational results from the observer**

5.7 Chapter Summary

In this chapter, the domain decomposition method and the parallel computing method have been introduced. The parallel 2D-3D overlapping method, the parallel wetting and drying treatment for the 2D analysis model have been proposed. As a result, the large-scale 2D-3D hybrid tsunami numerical model has been developed.

From the application to the tsunami caused by the Great East Japan Earthquake, the efficiency and the applicability of the present method to the real terrain has been confirmed, the computational time has been saved significantly by paralleling the whole 2D-3D hybrid model.