TITech-WARM

A Simulator for Water Reservoirs

User Manual

Takashi Nakamura
Department of Environmental Science & Technology
Tokyo Institute of Technology
March, 2012

Contents

Introduction

This manual explains an outline of Tokyo Institute of Technology WAter Reservoir Model (TITech-WARM), a simulator for water reservoirs. TITech-WARM is a non-hydrostatic-pressure and free-surface model, which enables analyses of thermal stratifications, 3-dimensional gravity currents in brackish lakes, and so on.

TITech-WARM is based on the combination of CIP method, a highly accurate solution for advection equation, and the Soroban meshes. The Soroban meshes enable precise descriptions of water surface and the interface of density stratifications by adjusting vertical mesh resolution arbitrarily and dynamically. Besides, the Soroban meshes also enable various computational meshes with arbitrary arrangements of cross sections and mesh point locations on them.

Parallel computations with TITech-WARM are available on a multi-core computer or a cluster of network-connected computers by using MPI libraries.

The following contents explain the outline of TITech-WARM and how to configure it to use. Please see a separate volume (not translated into English as of February, 2017) to refer the precise specifications of each variable or function.

Computational Space

Coordinate System

In the computational space of TITech-WARM, the x-axis is defined along the water route (main stream).

Each y-axis is perpendicular to the x-axis, and its origin ($y = 0$) is the intersection with the x-axis. As shown in **Fig. 1**, each y-axis of the computational space goes across the water route (main stream) perpendicularly in the real space. Then, y-coordinate of the computational space represents the distance from the water route on each cross section of the real space.

For the vertical direction, z-coordinate is defined by making upper direction positive.

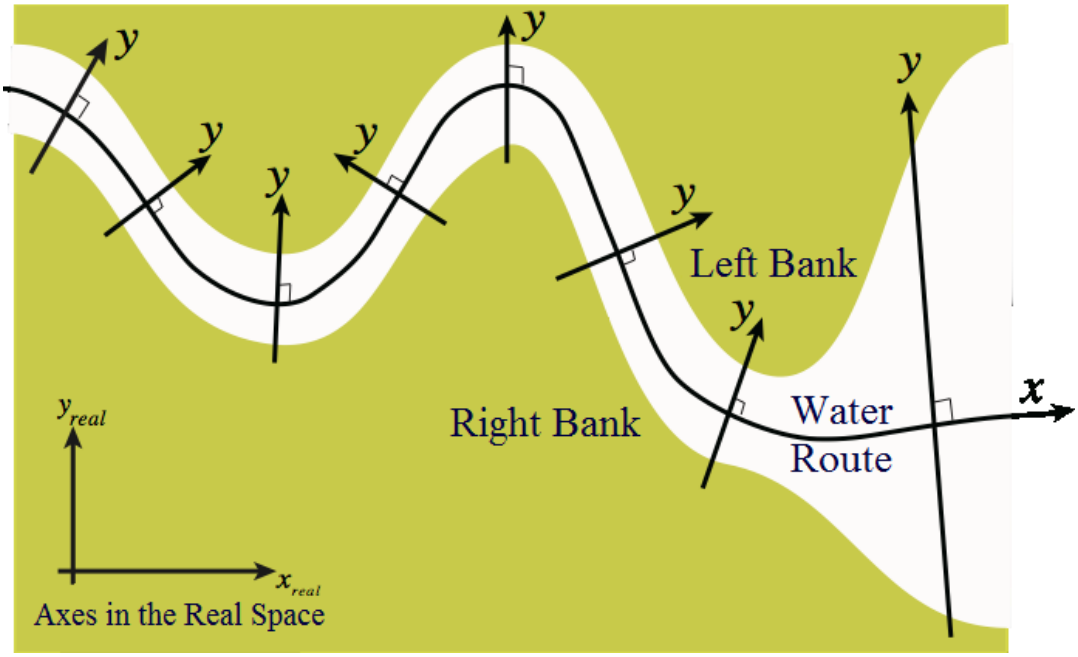

Fig. 1 Axes in the computational space

Computational Domain

The computational domain is fixed with the topographic data files when the computation begins. As shown in **Fig. 2**, the -x-side and +x-side boundary of the computational domain are written as $X_{\min}$ and $X_{\max}$ respectively.

The y-directional boundaries are located on the edges of riverbanks at each *x*-coordinate. Then, *y*-coordinates of the boundaries are the functions of x, and the -y-side and +y-side boundary of the computational domain are written as $Y_{\min}(x)$ and $Y_{\max}(x)$ respectively.

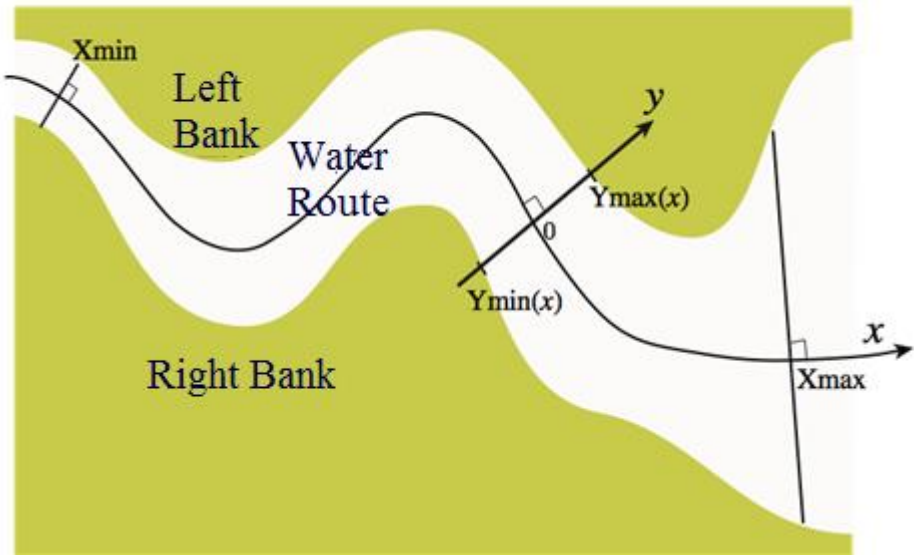

Fig. 2 Coordinates in the computational space

Straightened Topography (in Compt. Space)

A computation with TITech-WARM is conducted in the coordinate system shown in the Fig. 1, which contains the x- (main-stream direction) and y- (transverse direction) axes. These axes cross each other perpendicularly, and it means TITech-WARM assumes the water route (main stream) to be a straight line, as shown in **Fig. 3**. The influence of the bends in actual topography is added to the equation of motion as the centrifugal force term (function of curvature, see pp. 15~).

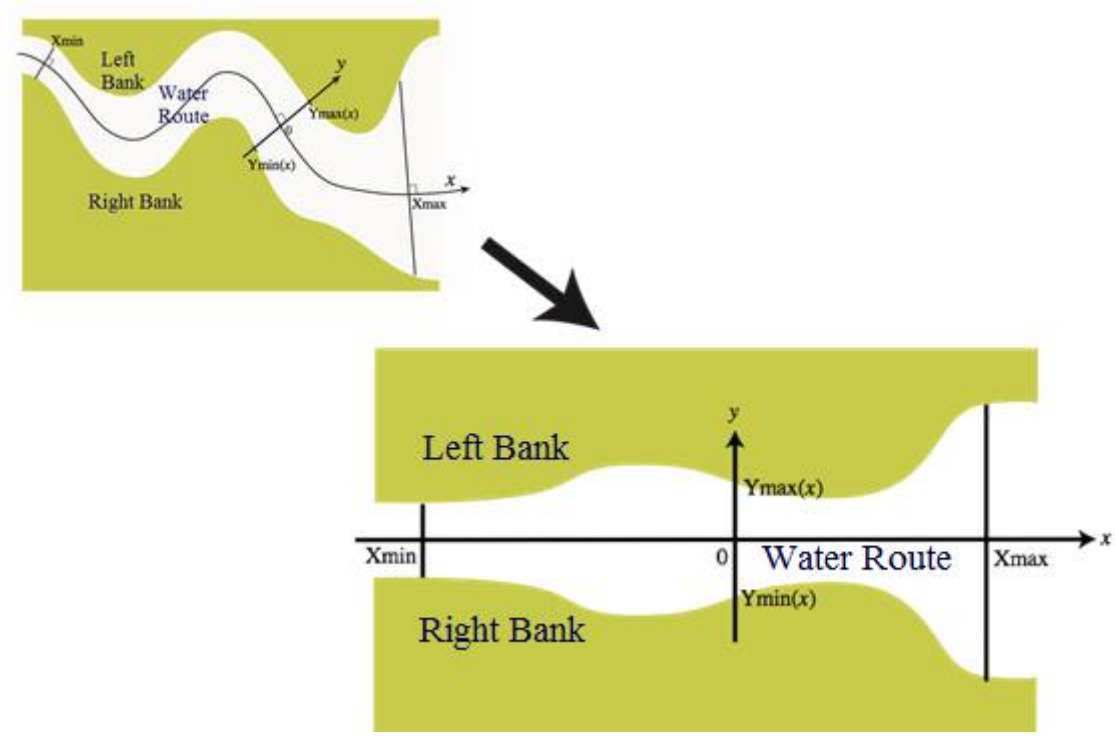

Fig. 3 Assumption on the topography

Soroban Meshes

As shown in **Fig. 3**, 3-dimensional computational space of TITech-WARM is composed of the straightened horizontal coordinates (*x*, *y*) and the vertical coordinate *z*. TITech-WARM discretizes this computational space with the Soroban meshes. **Fig. 4 (a) ~ (d)** show the general idea of discretization with the Soroban meshes. In x-y-z space, the Soroban meshes are composed of some mesh planes, some mesh axes on each mesh plain, and some mesh points on each mesh axis.

First, mesh planes are arranged perpendicularly to the main-stream direction (i.e. x-axis of computational space) like **Fig. 4 (a)**. Actual configurations of them are given in the input topographic data files (see pp. 79 ~).Each mesh plane is to be located in the computational domain, but never located on the *x*-directional boundary: $X_{\min}$or$X_{\max}$. These arrangements represent the perpendicular cross sections to the water route in the real space, as shown in **Fig. 4 (b)**.

All mesh points on a mesh plane have the same x-coordinate: x_i, because each mesh plain is perpendicular to the x-axis and parallel to the y-axis. This manual prints the index of mesh planes as subscript "*i*", unless otherwise noted.

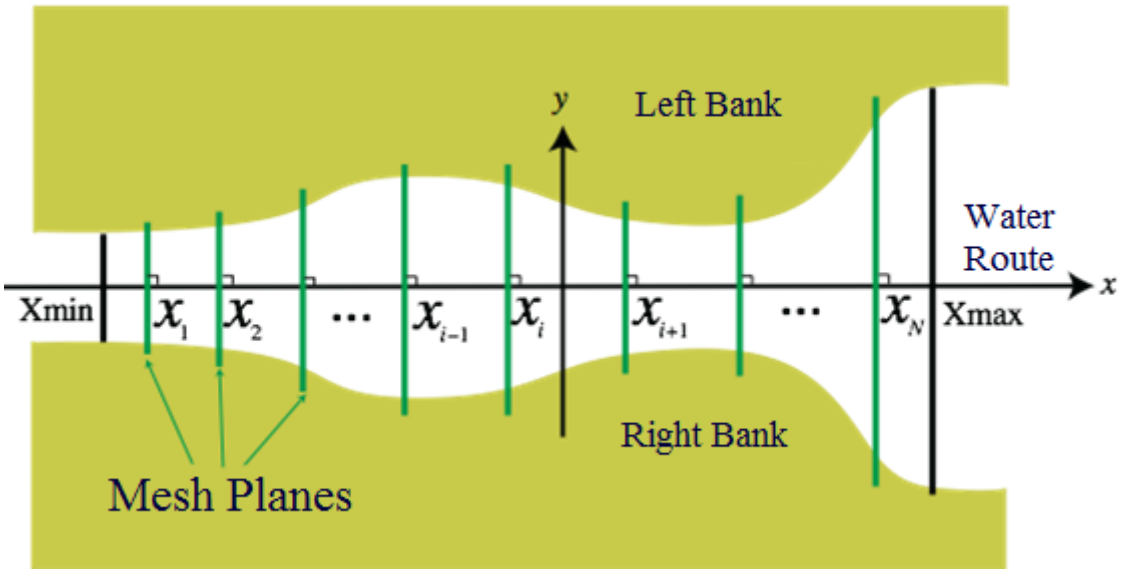

Fig. 4 (a) Mesh planes in the computational space

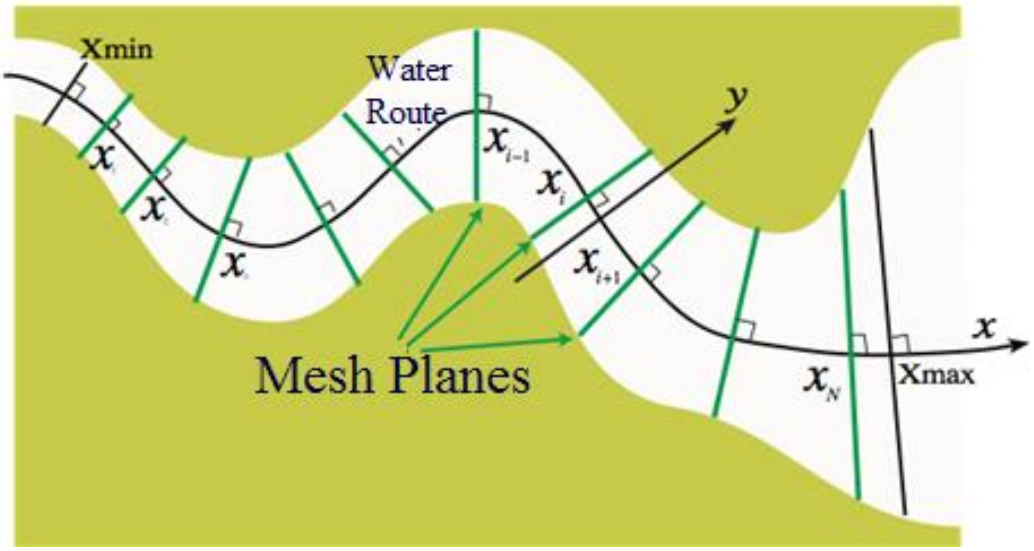

Fig. 4 (b) Mesh planes in the real space

Second, mesh axes are arranged on the mesh planes. Some parallel mesh axes to the vertical direction (i.e. z-axis) are arranged on each mesh plane like **Fig. 4 (c)**. The y-coordinates of mesh axes are given in the input topographic data files. On every mesh plane, mesh axes are to be arranged with arbitrary locations and intervals (between each other), but never located on the edges of riverbanks: $Y_{\min}(x)$or$Y_{\max}(x)$.

This manual prints the index of mesh axes as subscript "*j*", unless otherwise noted. Thus, the y-coordinate of *j*-th mesh axis on the *i*-thmesh plane (x_i) is written as y_{ij},and the horizontal coordinates of this axis are (x_i, y_{ij}). The 2-dimansional variables (ex. the heights of water surface and river bed surface) are defined on every combination of (x_i, y_{ij}).

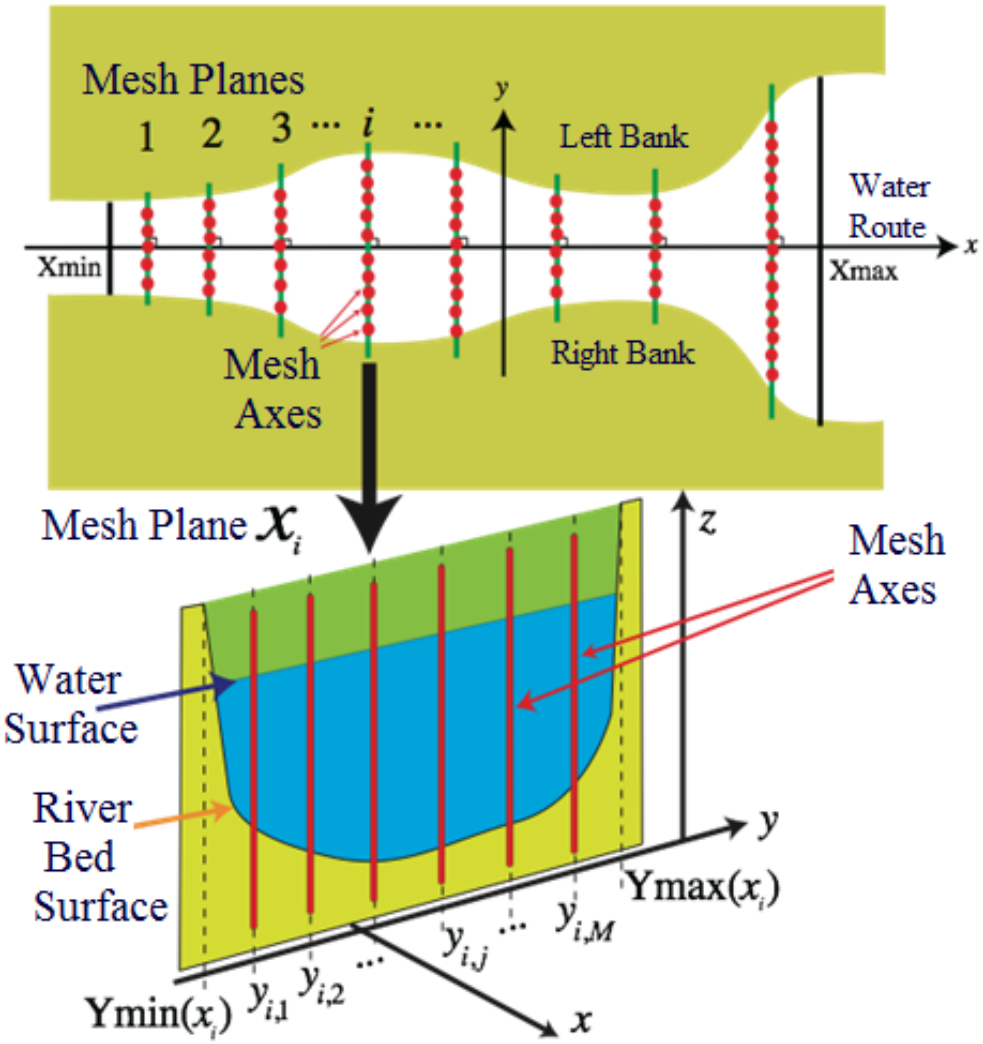

Fig. 4 (c) Mesh axes in the computational space

At last, mesh points are arranged on each mesh axis, like **Fig. 4 (d)**, to compute 3-dimensional variables (ex. velocities, water temperature, and salinity). The number of mesh points is given in the input topographic data files. On each mesh axis, the locations of mesh points are adjusted in every computational step. They are concentrated into the area which has the large vertical fluctuation of certain physical quantity (ex. density), based on its vertical gradient (see pp. 51~).

However, 1 mesh point each is always located on the water surface and river bed surface respectively to introduce the kinematic and mechanical conditions. Besides, 2 mesh points each are always contained above the water surface and below the river bed surface respectively, to compute the velocities of these areas. The velocity computations of these areas are based on the idea of inner boundary conditions, and adopt the same method with the water area by assuming these areas to be incompressible.

This manual prints the index of mesh points as subscript "*k*", unless otherwise noted. Thus, the z-coordinate of *k*-th mesh point on the *j*-th mesh axis of the *i*-th mesh plane (y_{ij}) is written as z_{ijk}, and the 3-dimensional coordinates of this point are (x_i, y_{ij}, z_{ijk}).

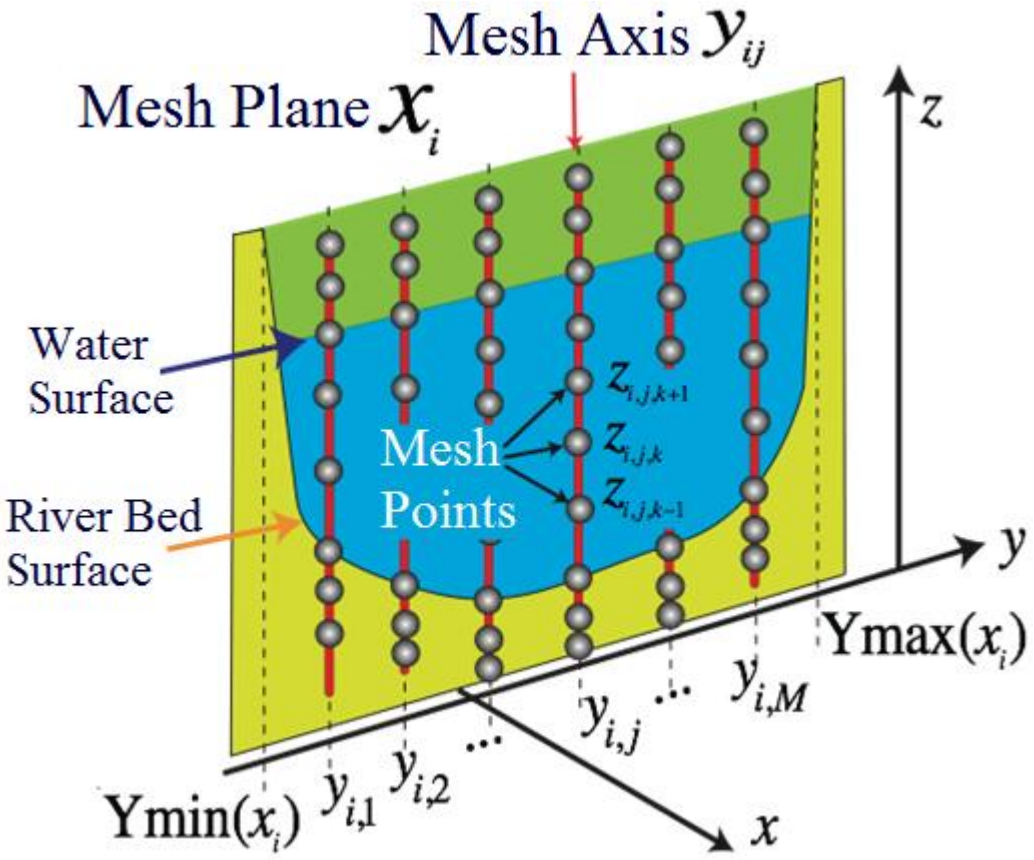

Fig. 4 (d) Mesh points in the computational space

Parallel Computations

TITech-WARM supports parallel computations with the Message Passing Interface (MPI) libraries. This chapter explains an outline of parallel computations of TITech-WARM, which is based on the decomposition of computational domain.

Decomposition of Computational Domain

The main aim of parallel computations is to make computations faster by dividing up the computational loads among several cores (computers or CPUs). The decomposition of computational domain method divides computational loads based on the computational space.

Fig. 5 shows the general idea of decomposition of computational domain. First, the locations of mesh planes, the locations of mesh axes, and the number of mesh points contained in each mesh axis are read and stored from the input topographic data files. Then, the computational domain is divided into some sub-domains by the *x*-coordinates of mesh planes. The number of sub-domains is given by the `-np` option when you start computations with the `mpirun` command. The extent of each sub-domain is adjusted so that every sub-domain will contain almost same number of mesh points. 1 sub-domain each is assigned to CPUs. (To be precise, it's assigned to the processes on CPUs. A process is also called "rank".) MPI gives unique integers (`myid` = 0, 1, 2…) to sub-domains (processes, ranks) so that a process can identify the sub-domain which it's in charge of. The sub-domain with `myid` = 0 is located the $X_{\min}$-side end of the row, then the sub-domain with `myid` = 1, `myid` = 2… are lined up toward +x-direction in order.

Incidentally, a sub-domain needs to contain at least 3 mesh planes because of the computational algorithm. Then if the number of sub-domain is too many to satisfy this requirement, the computation will not begin.

A sub-domain requires the values on the mesh points contained in the next sub-domains to compute. Then the MPI functions forward the values on the boundaries to next sub-domains as the need arises. (This is conducted by calling `update_boundary_value...` function.)

The indexes of mesh planes are given from $X_{\min}$ side in order. Each sub-domain stores the first and last index of contained mesh planes into the valuables `plane_start` and `plane_end`. In the same way, each mesh plane stores the first and last mesh axis index into `line_start[i]` and `line_end[i]`, and each mesh axis stores the first and last mesh point index into `point_start[i][j]` and `point_end[i][j]`.

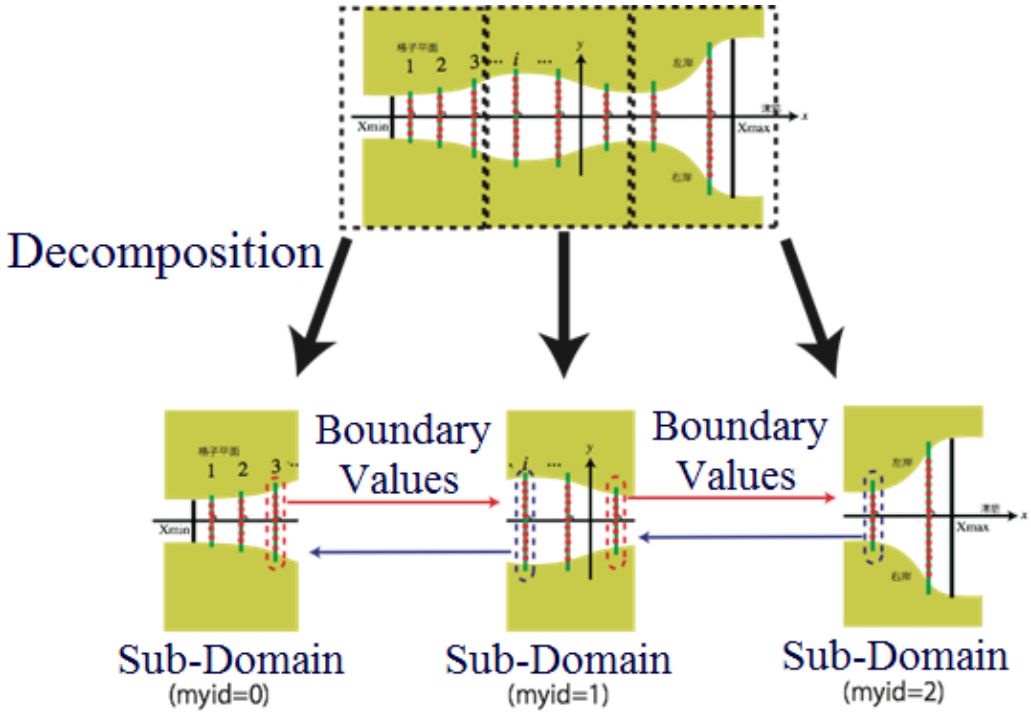

Fig. 5 Parallel computations with decomposition of computational domain (the case of 3 sub-domains)

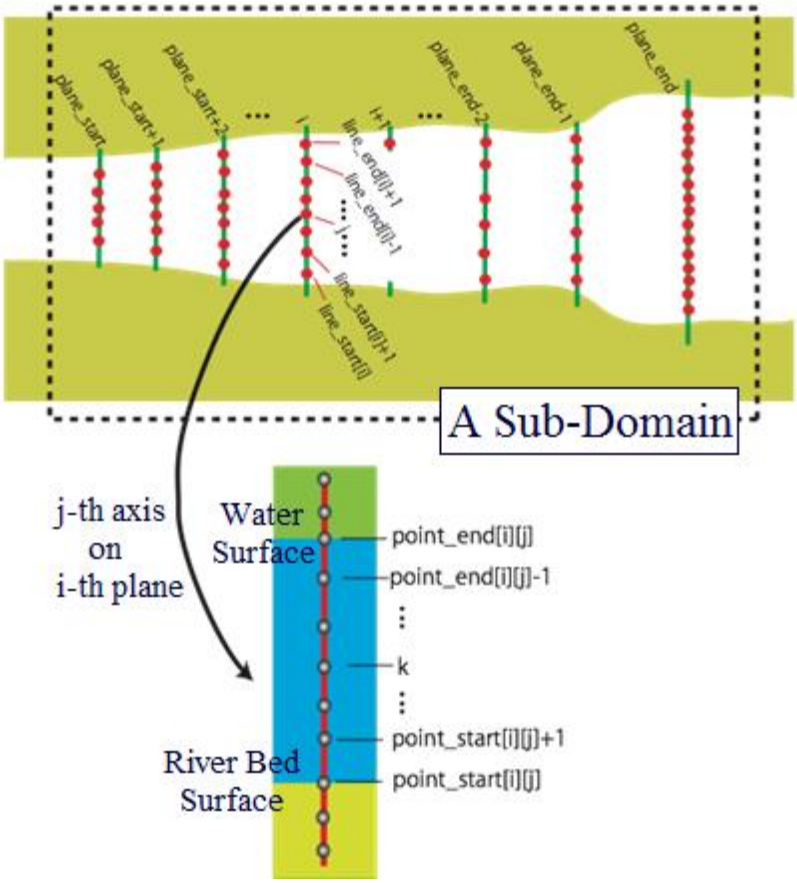

Fig. 6 First and last index in each sub-domain

Fundamental Equations

This chapter explains the fundamental equations of TITech-WARM.

3D k-ε Turbulence Model

TITech-WARM computes following variables on 3-dimensional Soroban meshes(x_i, y_{ij}, z_{ijk}) with the 3-dimensional*k-ε* turbulence model; flow velocities ($u(t,x,y,z)$, $v(t,x,y,z)$, and$w(t,x,y,z)$ represent x, y, and z-directional component respectively), water temperature $T(t,x,y,z)$, salinity $s(t,x,y,z)$, turbulent kinetic energy $k(t,x,y,z)$, turbulent dispersion rate $\varepsilon(t,x,y,z)$, and user-defined scalar $\phi_{\mathrm{sp}}(t,x,y,z)$. The fundamental equations of these 3-dimensional variables are as follows.

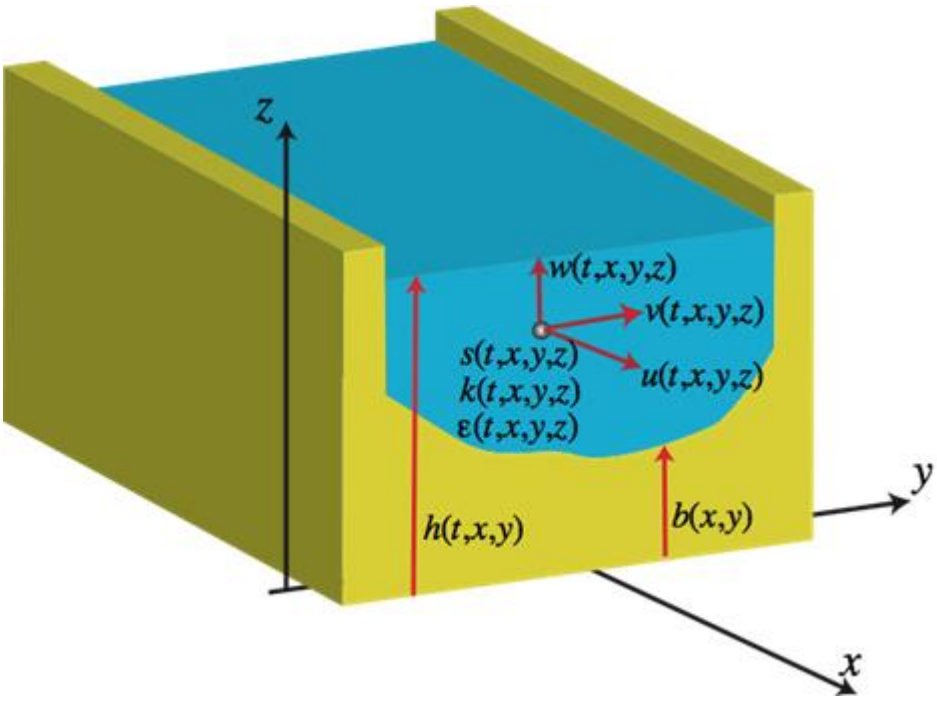

Fig. 7 Variables to compute

Equations of Motion

Flow velocities of water are computed with following equations of motions;

$$\frac{\mathrm{D}u}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial x} - \frac{2}{3}\frac{\partial k}{\partial x} + \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial u}{\partial z}\right) + \frac{uv}{R} + f_{\mathrm{co}}v + \frac{\tau_{\mathrm{x}}}{\rho} \quad \textbf{(1.1)}$$

$$\frac{\mathrm{D}v}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial y} - \frac{2}{3}\frac{\partial k}{\partial y} + \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial v}{\partial z}\right) + \frac{u^2}{R} + f_{\mathrm{co}}u + \frac{\tau_{\mathrm{y}}}{\rho} \quad \textbf{(1.2)}$$

$$\frac{\mathrm{D}w}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial z} - \frac{2}{3}\frac{\partial k}{\partial z} + \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial w}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial w}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial w}{\partial z}\right) + \frac{\tau_{\mathrm{z}}}{\rho} - g \quad \textbf{(1.3)}$$

$$\frac{\mathrm{D}}{\mathrm{D}t} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}$$

Equations **(1.1)** ~ **(1.16)** are defined in the computational space (x, y, z), which is composed of the main-stream direction x, the transverse direction y, and the vertical direction z. Equations **(1.1)** and **(1.2)** contain the effect of centrifugal force in the real space, where $1/R$ is the curvature of main-stream direction. Each variable's meaning is as follows.

t: Computational Time [sec]

x: Coordinate of computational space (main-stream direction) [m]

y: Coordinate of computational space (transverse direction) [m]

z: Coordinate of computational space (vertical direction) [m]

$u(t, x, y, z)$: Flow velocity (x direction, main-stream direction) [m/sec]

$v(t, x, y, z)$: Flow velocity (y direction, transverse direction) [m/sec]

$w(t, x, y, z)$: Flow velocity (z direction, vertical direction) [m/sec]

$p(t, x, y, z)$: Pressure [Pa]

$\rho(t, x, y, z)$: Density [kg/m^3]

$k(t, x, y, z)$: Turbulent kinetic energy [J]

$\nu_{\mathrm{lic\_M}}(x, y)$: Horizontal turbulent kinematic viscosity [m^2/sec]

$\nu_{\mathrm{eff}}(t, x, y, z)$: Vertical effective diffusivity [m^2/sec]

$R(x)$: Radius of curvature [m]

f_{co}: Coriolis parameter ($f = 2 \times 7.27 \times 10^{-5} \times \sin(\text{latitude})$)

g: Gravitational acceleration [m/sec^2]

$\tau_{\mathrm{x}}(t, x, y, z), \tau_{\mathrm{y}}(t, x, y, z), \tau_{\mathrm{z}}(t, x, y, z)$: Stress of each direction (on the water surface, the river bed surface, and the edges of riverbanks)

Incompressivity

The time evolutions of flow velocities are computed to satisfy following condition. This is based on the assumption that the fluid is incompressible.

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0 \quad \textbf{(1.4)}$$

Transportation Equations（of Water Temperature, Salinity, and Other Scalars）

Water temperature, salinity, and user-defined scalars (dissolved oxygen, suspended solid, etc.) are computed with following transportation equations;

$$\frac{\mathrm{D}T}{\mathrm{D}t}=\frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial z}\right)+S_T \quad \textbf{(1.5)}$$

$$\frac{\mathrm{D}s}{\mathrm{D}t}=\frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial z}\right) \quad \textbf{(1.6)}$$

$$\frac{\mathrm{D}\phi_{\mathrm{sp}}}{\mathrm{D}t}=w_{\mathrm{settlement}}\frac{\partial \phi_{\mathrm{sp}}}{\partial z}+\frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial x}\right)+\frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial y}\right)+\frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial z}\right)+S_{\phi\mathrm{sp}} \quad \textbf{(1.7)}$$

Each variable's meaning is as follows.

$T(t,x,y,z)$: Water temperature [°C]

$s(t,x,y,z)$: Salinity [psu]

$\phi_{\mathrm{sp}}(t,x,y,z)$: Concentration of "sp" scalar

$\nu_{\mathrm{lic}}(x,y)$: Horizontal turbulent diffusivity [m^2/sec]

σ_{t}: Constant of standard *k-ε* turbulence model (for water temperature and salinity)

$\sigma_{\phi\mathrm{sp}}$: Constant (for user-defined scalar)

$S_T(t,x,y,z)$: Source term of water temperature (heat) by solar radiation etc.

$S_{\phi\mathrm{sp}}(t,x,y,z)$: Source term of user-defined scalar

$w_{\mathrm{settlement}}$: Settlement (vertical) velocity of user-defined scalar [m/sec]

Turbulence Equations

Following fundamental equations are based on the *k-ε* turbulence model.

$$\frac{\mathrm{D}k}{\mathrm{D}t} = P_k - \varepsilon + G_k + \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_k}\frac{\partial k}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_k}\frac{\partial k}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_k}\frac{\partial k}{\partial z}\right) \quad \textbf{(1.8)}$$

$$\frac{\mathrm{D}\varepsilon}{\mathrm{D}t} = (C_1 P_k - C_2\varepsilon)\frac{\varepsilon}{k} + C_1(1 - C_3)\frac{\varepsilon}{k}G_k + \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial z}\right) \quad \textbf{(1.9)}$$

Each variable's meaning is as follows.

$k(t, x, y, z)$: Turbulent kinetic energy [J/kg]

$\varepsilon(t, x, y, z)$: Turbulent dispersion rate [J/kg sec]

σ_k: Constant of standard *k-ε* turbulence model (for turbulent kinetic energy)

σ_ε: Constant of standard *k-ε* turbulence model (for turbulent dispersion rate)

P_k: Production term of turbulent kinetic energy [J/kg sec]

G_k: Restraint term of turbulence by density stratifications [J/kg sec]

C_1, C_2, C_3: Constants of standard *k-ε* turbulence model

The production term of turbulent kinetic energy is based on the standard *k-ε* turbulence model and the assumption that the vertical shears of horizontal velocities are dominant.

$$P_k = \nu_{\mathrm{turb}}\left\{\left(\frac{\partial u}{\partial z}\right)^2 + \left(\frac{\partial v}{\partial z}\right)^2\right\} \quad \textbf{(1.10)}$$

The restraint term of turbulence by density stratifications is determined as follows;

$$G_k = \frac{g}{\rho}\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\min\left(\frac{\partial \rho}{\partial z}, 0\right) \quad \textbf{(1.11)}$$

The function $\min(a, b)$gives the smaller value of the parameters *a* and *b*. Then the equation **(1.11)** recognizes the extinction of turbulent energy (restraint of turbulence) alone in density stratifications.

Diffusivities have quite different spatial scale between horizontal directions and vertical direction. Then turbulent diffusivity is determined with the *k-ε* turbulence model in vertical direction alone. Those of horizontal directions are given from Richardson's 4/3 law;

$$\nu_{\mathrm{lic}} = \nu_0 D^{\frac{4}{3}} \quad \textbf{(1.12)}$$

D is the representative length of horizontal turbulences, which is assumed to be the largest horizontal interval of computational meshes.

$$D = \max(\Delta x_i, \Delta y_{ij}) \quad \textbf{(1.13)}$$

According to the observations in ocean, standard value of $\nu_0$is 0.01.

Horizontal turbulent viscosity is determined as

$$\nu_{\text{lic\_M}} = S_c \times \nu_{\text{lic}} \quad \textbf{(1.14)}$$

with the horizontal turbulent diffusivity (calculated in the equation**(1.12)**) and the turbulent Schmidt number S_c. Arbitrary S_c is available in the computational conditions, although commonly it is larger than 1.

The vertical turbulent diffusivity (viscosity) is given as

$$\nu_{\text{eff}} = \nu_{\text{mol}} \times \nu_{\text{turb}} \quad \textbf{(1.15)}$$

ν_{mol} is the molecular viscosity (diffusivity) of water in certain momentum, salinity, and water temperature. The turbulent viscosity $\nu_{\text{turb}}$is calculated as

$$\nu_{\text{turb}} = C_\mu \frac{k^2}{\varepsilon} \quad \textbf{(1.16)}$$

although it is independent of actual turbulent viscosity or diffusivity.

Water Surface Equation

The vertical location of water surface $h(t,x,y)$is assumed to be a single-valued function of horizontal coordinates: (x,y). This means that no rolling up or overtaking of waves is recognized. The time evolution of $h(t,x,y)$is computed with following equation;

$$\frac{\partial h}{\partial t} + \frac{\partial m}{\partial x} + \frac{\partial n}{\partial y} = 0 \quad \textbf{(1.17)}$$

Here, *m* and *n* are the x-directional and y-directional flow rate per a unit of width at (x,y). They are estimated with *u* and *v* which were given in the equations **(1.1)** and **(1.2)** as follows;

$$m(t,x,y) \equiv \int_{b(x,y)}^{h(t,x,y)} u(t,x,y,z)\mathrm{d}z \quad \textbf{(1.18)}$$

$$n(t,x,y) \equiv \int_{b(x,y)}^{h(t,x,y)} v(t,x,y,z)\mathrm{d}z \quad \textbf{(1.19)}$$

where $b(x,y)$ is the vertical location of river bed surface.

Boundary Conditions

The time evolutions of these fundamental equations are computed with boundary conditions on the water surface, the river bed surface, the edges of riverbanks, and the upstream and downstream end of the computational domain.

On the Water Surface

The time evolutions of areas above the water surface and below the river bed surface are computed by the same method with the water area, as if these areas are also filled with water. Meanwhile, a following mechanical condition is applied to the pressure on the water surface to ensure the validity of flows in the water area;

$$p\big(t,x,y,h(t,x,y)\big)=0 \tag{1.20}$$

There is no transportation of salinity through the water surface. Then the boundary condition of salinity is

$$\left(\frac{\partial s}{\partial z}\right)_{(t,x,y,h(t,x,y))}=0 \tag{1.21}$$

The change of water temperature caused by meteorological conditions (solar radiation etc.) is considered in the source term computation. Then no heat transportation is recognized in the advection and diffusion computations;

$$\left(\frac{\partial T}{\partial z}\right)_{(t,x,y,h(t,x,y))}=0 \tag{1.22}$$

If no input data of wind field is prepared, the influence of wind stress on the flow velocity will not be recognized;

$$\left(\frac{\partial u}{\partial z}\right)_{(t,x,y,h(t,x,y))}=0 \tag{1.23}$$

$$\left(\frac{\partial v}{\partial z}\right)_{(t,x,y,h(t,x,y))}=0 \tag{1.24}$$

In contrast, if some input data of wind field are prepared, the influence of wind stresses on the flow velocity will be applied as follows;

$$\nu_{\mathrm{turb}}\left(\frac{\partial u}{\partial z}\right)_{(t,x,y,h(t,x,y))}=-\frac{\tau_{\mathrm{wind}}}{\rho_{\mathrm{water}}}\frac{U_{\mathrm{x}}}{\sqrt{U_{\mathrm{x}}^2+U_{\mathrm{y}}^2}} \tag{1.25}$$

$$\nu_{\mathrm{turb}}\left(\frac{\partial v}{\partial z}\right)_{(t,x,y,h(t,x,y))}=-\frac{\tau_{\mathrm{wind}}}{\rho_{\mathrm{water}}}\frac{U_{\mathrm{y}}}{\sqrt{U_{\mathrm{x}}^2+U_{\mathrm{y}}^2}} \tag{1.26}$$

Here, U_{x} and U_{y} are the x-directional (main-stream-direction) and y-directional (transverse direction) components of the wind velocity at altitudes above 10 m.ρ_{water}[kg/m^3] is the density of water τ_{wind} is the frictional stress of wind;

$$\tau_{\mathrm{wind}} = \rho_{\mathrm{air}} C_{\mathrm{D}} U^2 \quad \textbf{(1.27)}$$

$\rho_{\mathrm{air}} = 1.3\ \mathrm{kg/m^3}$ is the density of air. $U = \sqrt{U_{\mathrm{x}}^2 + U_{\mathrm{y}}^2}$ is the speed of wind. The drag coefficient C_{D} is determined with the bulk equation by Kondo et al.[i];

$$10^3 \times C_{\mathrm{D}} = \begin{cases} 1.08U^{-0.15} & \text{for } U < 2.2\ \mathrm{m/s} \\ 0.771 + 0.0858U & \text{for } 2.2 < U < 5\ \mathrm{m/s} \\ 0.867 + 0.0667U & \text{for } 5 < U < 8\ \mathrm{m/s} \\ 1.2 + 0.025U & \text{for } 8 < U < 25\ \mathrm{m/s} \\ 0.073U & \text{for } 25\ \mathrm{m/s} < U \end{cases} \quad \textbf{(1.28)}$$

The turbulent kinetic energy and the turbulent dispersion rate are given as constants based on the wall boundary conditions;

$$k = \frac{1}{\sqrt{C_\mu}} \frac{\tau_{\mathrm{wind}}}{\rho_{\mathrm{water}}} \quad \textbf{(1.29)}$$

$$\varepsilon = \frac{\left(\sqrt{C_\mu} k\right)^{\frac{3}{2}}}{\kappa z_{\mathrm{s}}/2} \quad \textbf{(1.30)}$$

This is based on an assumption that production (by the stress) and dispersion of turbulent kinetic energy are balanced in a close area to the water surface. The coefficient $\kappa = 0.4$ is the von Karman constant, and z_{s} is the distance from the water surface. The actual value of z_{s} is always 0 because TITech-WARM locates mesh points on the water surface. However, vertical mesh intervals on the water surface are adopted here to ensure stable computations.

Several boundary conditions for the user-defined scalars $\phi_{\mathrm{sp}}(t, x, y, z)$ are available on the water surface (see pp. 105 ~).

On the River Bed Surface

The pressure is estimated with the following mechanical condition:

$$u_{\mathrm{n}}\big(t, x, y, b(t, x, y)\big) = 0 \quad \textbf{(1.31)}$$

Here, u_{n} is the perpendicular component of velocity to the river bed surface. Besides, the influence of friction against the river bed surface is applied to the parallel components of velocity to the river bed surface;

$$\nu_{\mathrm{turb}} \left(\frac{\partial u}{\partial z}\right)_{(t,x,y,b(t,x,y))} = -f_{\mathrm{b}} u_{\mathrm{b}} u \quad \textbf{(1.32)}$$

$$\nu_{\mathrm{turb}}\left(\frac{\partial v}{\partial z}\right)_{(t,x,y,b(t,x,y))} = -f_{\mathrm{b}}u_{\mathrm{b}}v \quad \textbf{(1.33)}$$

$$\nu_{\mathrm{turb}}\left(\frac{\partial w}{\partial z}\right)_{(t,x,y,b(t,x,y))} = -f_{\mathrm{b}}u_{\mathrm{b}}w \quad \textbf{(1.34)}$$

f_{b} is the friction coefficient, for which arbitrary value is available in computational conditions. u_{b} is the parallel component of velocity to the river bed surface.

There is no transportation of salinity and heat (water temperature) through the river bed surface. Then the boundary conditions of them are

$$\left(\frac{\partial s}{\partial z}\right)_{(t,x,y,b(t,x,y))} = \left(\frac{\partial T}{\partial z}\right)_{(t,x,y,b(t,x,y))} = 0 \quad \textbf{(1.35)}$$

Two ways of boundary conditions are available for the turbulent kinetic energy and turbulent dispersion rate. You can select one of them with the value of KE_BED_BOUNDRY_MODEL in a fundamental configuration file "*.inc":

If KE_BED_BOUNDARY_MODEL=0

The turbulent kinetic energy and the turbulent dispersion rate are given as constants based on the wall boundary conditions;

$$k = \frac{1}{\sqrt{C_\mu}}\frac{\tau_{\mathrm{btm}}}{\rho_{\mathrm{water}}} \quad \textbf{(1.36)}$$

$$\varepsilon = \frac{\left(\sqrt{C_\mu}k\right)^{\frac{3}{2}}}{\kappa z_{\mathrm{b}}/2} \quad \textbf{(1.37)}$$

This is based on an assumption that production (by the stress) and dispersion of turbulent kinetic energy are balanced in a close area to the river bed surface, just like the water surface. The stress from the river bed surface is assumed to be

$$\tau_{\mathrm{btm}} = \rho_{\mathrm{water}}f_{\mathrm{b}}(u_{\mathrm{b}}^2 + v_{\mathrm{b}}^2 + w_{\mathrm{b}}^2) \quad \textbf{(1.38)}$$

z_{b} is the distance from the river bed surface. The actual value of z_{b} is always 0 because TITech-WARM locates mesh points on the river bed surface. However, vertical mesh intervals on the river bed surface are adopted here to ensure stable computations.

If KE_BED_BOUNDARY_MODEL = 1

Ishikawa and Moribayashi et al[ii].analyzed density plumes which are running down a slope. These analyses were based on the similarity of plumes, and formularized a set of supply-flux model of the turbulent kinetic energy and the turbulent dispersion rate;

$$\nu_{\mathrm{turb}}\left(\frac{\partial k}{\partial z}\right)_{(t,x,y,b(t,x,y))} = \alpha f_{\mathrm{b}}\left(u_{\mathrm{b}}^2 + v_{\mathrm{b}}^2 + w_{\mathrm{b}}^2\right)^{\frac{3}{2}} \quad \textbf{(1.39)}$$

$$\nu_{\mathrm{turb}}\left(\frac{\partial \varepsilon}{\partial z}\right)_{(t,x,y,b(t,x,y))} = \beta f_{\mathrm{b}}\left(u_{\mathrm{b}}^2 + v_{\mathrm{b}}^2 + w_{\mathrm{b}}^2\right)^{\frac{3}{2}}\varepsilon/k \quad \textbf{(1.40)}$$

In the case of KE_BED_BOUNDARY_MODEL = 1, the turbulent kinetic energy and the turbulent dispersion rate are determined by this model. Here, α and $\beta$are the constants which are determined by the frictional coefficient f_{b} and the constants of *k*-ε model C_1 and C_2. The actual values of α and $\beta$are determined automatically in the set_stress_and_flux_on_surface function.

Several boundary conditions for the user-defined scalars $\phi_{\mathrm{sp}}(t,x,y,z)$ are available on the river bed surface (see pp. 105 ~).

On the Edges of Riverbanks

If a mesh point comes in contact with the riverbanks: $y = Y_{\min}$ or $Y_{\max}$, following boundary conditions are applied on it, just like the river bed surface;

$$\left(\frac{\partial u}{\partial n}\right) = -f_{\mathrm{b}}u_{\mathrm{s}}u \quad \textbf{(1.41)}$$

$$\left(\frac{\partial v}{\partial n}\right) = -f_{\mathrm{b}}u_{\mathrm{s}}v \quad \textbf{(1.42)}$$

$$\left(\frac{\partial w}{\partial n}\right) = -f_{\mathrm{b}}u_{\mathrm{s}}w \quad \textbf{(1.43)}$$

Here, *n* is a perpendicular vector to the edge of touching riverbank. u_{s} is flow velocity on this mesh point. Besides, these boundary conditions are also applied on the mesh points which come in contact with the “land areas” (see pp. 48~).

No transportation of salinity, heat (water temperature), turbulent kinetic energy, and turbulent dispersion rate is recognized through the edges of riverbanks;

$$\left(\frac{\partial s}{\partial z}\right) = \left(\frac{\partial T}{\partial z}\right) = \left(\frac{\partial k}{\partial z}\right) = \left(\frac{\partial \varepsilon}{\partial z}\right) = 0 \quad \textbf{(1.44)}$$

On the Upstream and Downstream Boundaries

Arbitrary boundary conditions of x-directional velocity, salinity, water temperature, and user-defined scalars are available on the upstream and downstream boundaries of the computational domain: $x = X_{\min}$ or $X_{\max}$. They are prepared as time-series input data files on these boundaries.

TITech-WARM supports branching of rivers (as shown in **Fig. 8**) with several "boundary domains" on the $X_{\min}$- and $X_{\max}$-boundaries. Each boundary can contain up to 10 boundary domains. The extent of each boundary domain is determined in the fundamental configuration file "*.inc" with the minimum and maximum y-coordinates (of the computational space): $Y_{\min_?}$and $Y_{\max\_?}$(? is the index of boundary domains).

Fig. 9 shows a y-z cross section at the boundary $X_{\min}$ or $X_{\max}$. Different boundary conditions are available on the boundary domain #1 and #2 by preparing different time-series input data files for the values of flow rate, salinity, water temperature, and so on. If an input data file of flow rates is prepared on a boundary domain, the velocity of main-stream direction is given equally throughout this boundary domain (i.e. its value is the flow rate divided by the cross sectional area). In the same way, if an input data file of salinity, water temperature, or a user-defined scalar is prepared, its value is given equally throughout the boundary domain. If an input data file of water level is prepared, it's also given equally throughout the boundary domain. If no input data file of these physical quantities is prepared, or in the case of turbulent kinetic energy and turbulent dispersion rate, an *x*-directional free boundary condition$\partial f/\partial x = 0$is applied.

You can restrict inlets or outlets within some partial extents by applying "gate conditions" on each boundary domain. **Fig. 10** shows an example of gate conditions. Several rectangular extent where inlets or outlets are allowed (gates, passage domains) are available. Gate conditions are activated by writing the name of an input data file of passage domains (gates) on the fundamental configuration file. This data file needs to content the number of passage domains and their corners' y- and z-coordinates in the computational space: $\left(y_{\min}^{\mathrm{G?}}, z_{\min}^{\mathrm{G?}}\right)$ and $(y_{\max}^{\mathrm{G?}}, z_{\max}^{\mathrm{G?}})$. The extents of passage domains can overlap with the area above the water surface (like Gate #1 in **Fig. 10**) or below the river bed surface (like Gate #2 in **Fig. 10**) to represent upper or lower layer discharges. If a gate condition is applied, every area except the passage domains is assumed to be a wall, and no inlet or outlet is allowed through it. If an input data file of flow velocity, salinity, water temperature etc. is prepared on a boundary domain with gate conditions, they are applied to the passage areas alone. If an input data file of flow rate is prepared on such a boundary domain, the value of flow velocity is given as the flow rate divided by the total cross sectional area of passage domains.

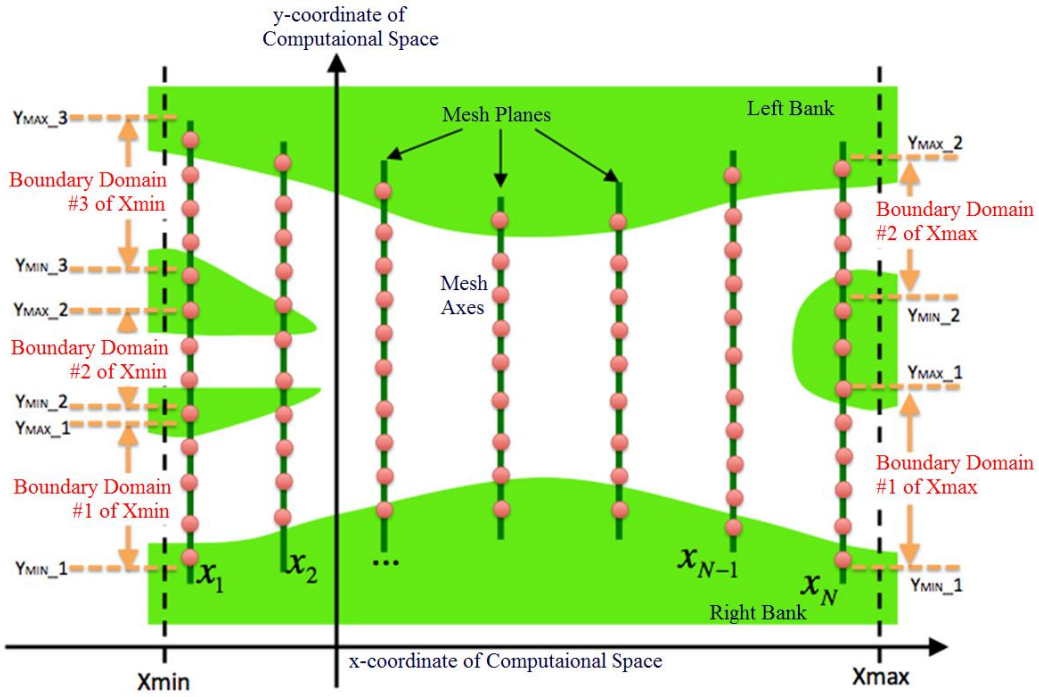

Fig. 8 Boundary domains on the x boundaries

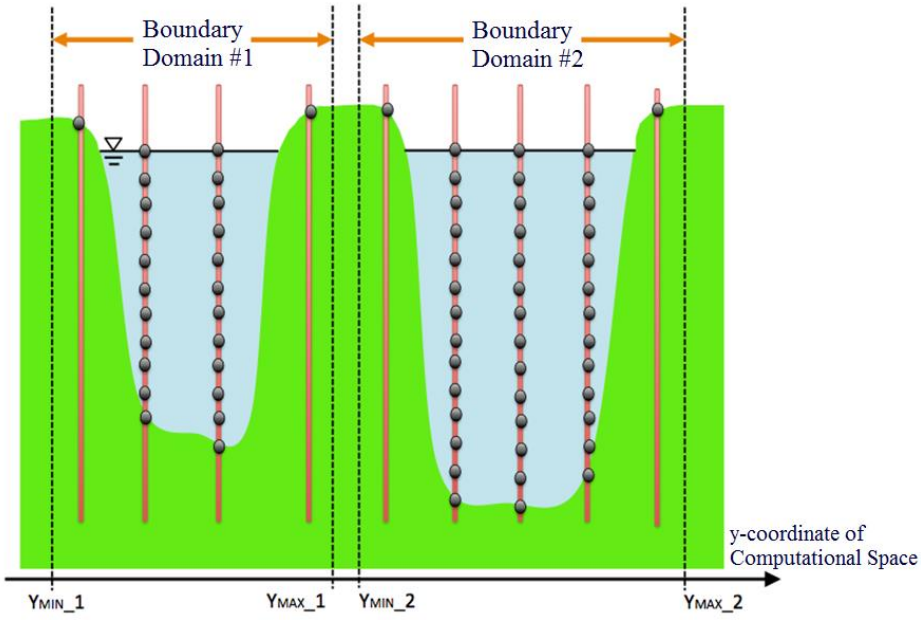

Fig. 9 Boundary domains on a x-boundary cross section

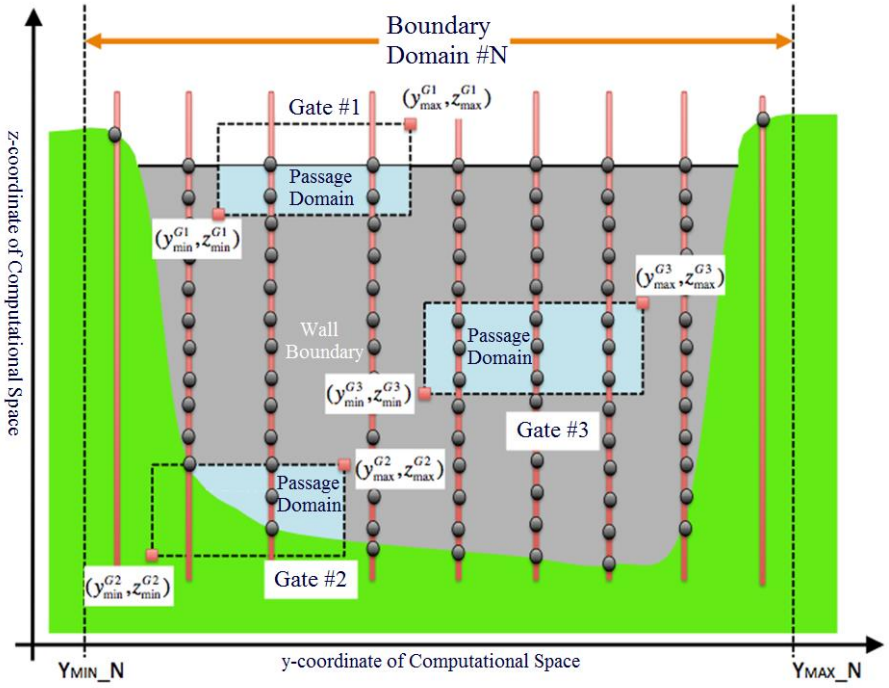

Fig. 10 Gate conditions on a boundary domain

---

[i] Kondo, J.（近藤純正） et al., 水環境の気象学 (Meteorology of Water Environment), Asakura Publishing Co. Ltd., (1994), pp.169-170.

[ii] Moribayashi, T.（盛林哲）,傾斜プルームの連行則に関する解析的研究 (An analytical research on the entrainment law of slope plumes), *The master thesis of Tokyo Institute of Technology, Interdisciplinary Graduate School of Science and Engineering*, (1999).

Heat Transportation Model

This section explains the modeling of heat transportation terms from meteorological conditions (solar radiation, air temperature, wind etc.) to represent the ups and downs of thermal stratifications. The transportation equation of water temperature is

$$\frac{\mathrm{D}T}{\mathrm{D}t} = \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial z}\right) + S_T \quad \textbf{(1.5)}$$

$S_T$in the right side is given as

$$S_T = \frac{\phi}{\rho C_{\mathrm{w}}} \quad \textbf{(1.45)}$$

ϕ[kcal/m^3 sec] is the heat flux which outer atmosphere supplies through the water surface. C_{w} is the specific heat capacity of water. The heat transportation from/to the outside of water (with the heat flux ϕ) is composed of longwave and shortwave solar radiation and the latent and sensible through the water surface. Incidentally, the ways of temperature computation are different between above and below the water surface. This is because the shortwave solar radiation alone affects the temperature below the water surface, although other factors also affect above.

Above the Water Surface

Fig. 11 shows the heat transportations around and below the water surface of water reservoirs. The water surface receives heat from the shortwave solar radiation (global solar irradiance): ϕ_{I} and the longwave radiation from atmosphere and clouds: ϕ_{Ldn}. Meanwhile, it also emits the heat of longwave radiation ϕ_{Lup} to atmosphere. Besides, latent and sensible heat is transported between the water surface and the atmosphere, which is called latent/sensible heat flux. Here, latent heat means the heat transported by water evaporation, and sensible heat means the heat transported by the heat conduction and convection. The amounts of them are called latent heat flux ϕ_{c} and sensible heat flux ϕ_{e} respectively.

The water surface reflects a part of shortwave solar radiation (ϕ_{I}), so $(1-\alpha)\phi_{\mathrm{I}}$is transported into the water. Here, around half of $(1-\alpha)\phi_{\mathrm{I}}$ is absorbed by the water surface, and the rest reaches the depths of the water reservoir while it's getting weak exponentially. Therefore, the heat flux through the water surface $\phi_{\mathrm{surf}}$is modeled as follows;

$$\phi_{\mathrm{surf}} = \beta(1-\alpha)\phi_{\mathrm{I}} - C_{\mathrm{l}}\left\{\left(\phi_{\mathrm{Lup}} - \phi_{\mathrm{Ldn}}\right) + \left(\phi_{\mathrm{c}} + \phi_{\mathrm{e}}\right)\right\} \quad \textbf{(1.46)}$$

α is the reflection ratio (albedo) of the water surface, and $\beta$is the absorption ratio of shortwave solar radiation on the water surface. They are constants which are determined by the transparency of water and so on. Generally speaking, α is around 0.03 ~ 0.04 and β is around 0.4 ~ 0.6[iii].C_{l} is a parameter to adjust the amount of heat flux except shortwave. It's used to represent stronger cooling around the water surface (in night times, strong wind etc.), and its standard is 1.

There are several empirical models to estimate the total amount of heat from longwave (net longwave radiation) $\phi_{\mathrm{L}} = \phi_{\mathrm{Lup}} - \phi_{\mathrm{Ldn}}$ or the sum of latent heat and sensible heat $\phi_{\mathrm{c}} + \phi_{\mathrm{e}}$ based on experiments and observations. TITech-WARM adopts the Swinbank's formula[iv]for the net longwave radiation $\phi_{\mathrm{L}} = \phi_{\mathrm{Lup}} - \phi_{\mathrm{Ldn}}$:

$$\phi_{\mathrm{L}} = 0.97\sigma_{\mathrm{s}}\left\{T_{\mathrm{W}}^4 - 0.937 \times 10^{-5} T_{\mathrm{A}}^6 \left(1.0 + 0.17\left(\frac{C}{10}\right)^2\right)\right\} \quad \textbf{(1.47)}$$

$\sigma_{\mathrm{s}} = 1.171 \times 10^{-6}$ kcal/m^2 day K is the Stefan-Boltzmann constant. T_{W} [K] and T_{A} [K] arethe temperature of the water surface and the atmosphere respectively.C is the cloudiness (cloud cover; 0 ~ 10).

Rowher's formula[v]is adopted for the latent and sensible heat: $\phi_{\mathrm{c}} + \phi_{\mathrm{e}}$:

$$\phi_{\mathrm{c}} + \phi_{\mathrm{e}} = (0.000308 + 0.000185 U_{15\mathrm{cm}})\rho(e_{\mathrm{s}} - \psi e_{\mathrm{a}})\left\{L_{\mathrm{v}} + C_{\mathrm{w}} T_{\mathrm{s}} + \frac{269.1(T_{\mathrm{s}} - T_{\mathrm{a}})}{\mathrm{e_s} - \psi e_{\mathrm{a}}}\right\} \quad \textbf{(1.48)}$$

$U_{15\mathrm{cm}}$[m/s] is the wind speed at altitude above 15 cm.e_{s} [mmHg] and e_{a} [mmHg] are the saturated vapor pressure at the temperature of the water surface and the atmosphere respectively.ψ[%] is the relative humidity. L_{v} [kcal/kg] is the latent heat of water evaporation.T_{s}[°C] and T_{a}[°C] are the temperature of the water surface and the atmosphere respectively. Here, saturated vapor pressure is determined by the Tetens' formula:

$$e_{\mathrm{s}} = 6.112 \times 10^{\frac{7.5T_{\mathrm{s}}}{T_{\mathrm{s}}+273.3}} \quad \textbf{(1.49)}$$

$$e_{\mathrm{a}} = 6.112 \times 10^{\frac{7.5T_{\mathrm{a}}}{T_{\mathrm{a}}+273.3}} \quad \textbf{(1.50)}$$

This formula is practically correct on surfaces of water or ice in usual temperature (-30 to +40 °C). The latent heat of water evaporation L_{v} is determined by the following approximation:

$$L_{\mathrm{v}} = -3.766 \times 10^{-4} T_{\mathrm{s}}^2 - 5.440 \times 10^{-1} T_{\mathrm{s}} + 5.97 \times 10^2 \quad \textbf{(1.51)}$$

The wind speed at altitude above 15 cm $U_{15\mathrm{cm}}$ is derived from that of 10 m altitude:

$$U_{15\mathrm{cm}} = U_{10\mathrm{m}} + \frac{u^*}{\kappa}\log\left(\frac{15\ cm}{10\ m}\right)$$

This derivation is based on an assumption that the wind speed has a logarithmic distribution near the water surface:

$$U(z) = \frac{u^*}{\kappa}\log\left(\frac{Z}{z_0}\right)$$

The frictional velocity $u^*$is estimated with the drag coefficient C_{D} of equation **(1.28)** as follows:

$$u^* = \sqrt{C_{\mathrm{D}}} U_{10\mathrm{m}}$$

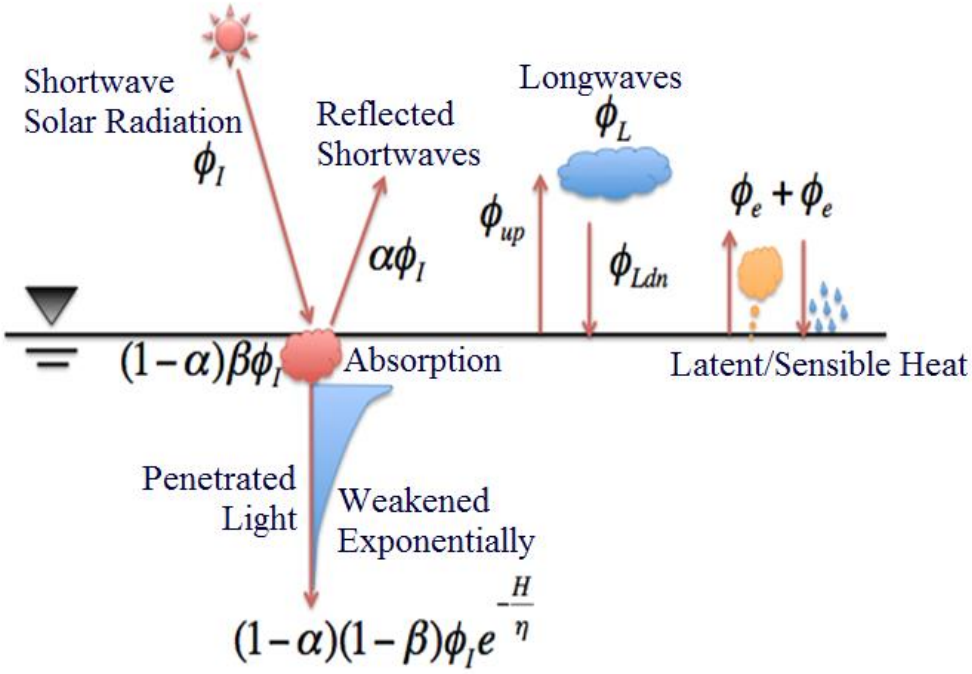

Fig. 11 Fluxes of heat transportation

Below the Water Surface

There is no latent or sensible heat below the water surface. The amount of heat is mainly varied by the absorption of solar radiation there. The Lambert-Beer law shows that penetrated light gets weak exponentially by the depth *H* of transparent matters (water etc.). As shown in **Fig. 11**, the penetrated light is assumed to be the part of shortwave solar radiation (ϕ_I) which was neither reflected nor absorbed by the water surface: $(1-\alpha)(1-\beta)\phi_I$. Therefore, the heat flux of penetrated light $\phi_d$at the depth $H = h - z$is estimated as follows:

$$\phi_d = (1-\alpha)(1-\beta)\phi_I \exp\left(-\frac{H}{\eta}\right) \quad \textbf{(1.52)}$$

Here, η is the attenuation length; the strength of penetrated light becomes 1/e while it travelling an attenuation length. $\eta$is assumed to be a constant in TITech-WARM, although it depends on the thickness of suspended solid etc. actually. h [m] is the heights of the water surface.

[iii]For example: Ikegami, J. and Umeda, J., Analysis on Thermal Stratification in a Reservoir Using a Vertical 2-Dimensional Model （ダム貯水池の水温成層に関する鉛直二次元数値解析）, *Proceedings of Hydraulic Engineering （水工学論文集）*, Vol. 51, (2007), pp. 1349-1354.

[iv]Swinbank, W.C.: Long-wave radiation from clear skies. *Quart. J. Roy. Meteor. Soc.*, 89, (1963) pp.330-348.

[v]Rohwer, C.: Evaporation from Free Water Surface, U.S. Department of Agriculture, *Technical Bulletin* No.271 (1931).

Aeration Model

This section explains how an aeration model is introduced in TITech-WARM. This model assumes air aerations, which send out micro bubbles continuously with diffuser pipes and so on. As shown in **Fig. 12**, the bubbles are sent out from diffuser pipes, rise up, and compose a bubble core. The bubble core entrains surrounding water and generates a rising "inner plume" around itself. The inner plume rises up while it's being diluted, stops when it loses all momentum by buoyancy or entrainment, and generates a sinking "outer plume" in the outside of itself. Then, the outer plume sinks (because of negative buoyancy) until surrounding water becomes the same density with it, and intrudes into the surrounding water horizontally.

The speed of rising bubbles is around several tens cm/sec, and it's shorter than the time scale of main flows in a water reservoir. Besides, the size (diameter) of each plume is around 10 m at most, and it's smaller than the spatial (mesh) scale of the main flows. Then, this aeration model separates the computation of plume generation from that of main flow, and conduct the former with a 1-dimensional model. In other words, at every time step of main flow computation, this model (1) estimates the vertical distribution of density above aeration devices, (2) make a vertical 1D plume analysis with this density distribution (as input condition) and an assumption that the plume is in a steady state, and then (3) gets relative source or sink at every depth and applies it to the main flow computation. The following contents explain a plume model which is adopted by TITech-WARM and its coupling with the model for the main flows.

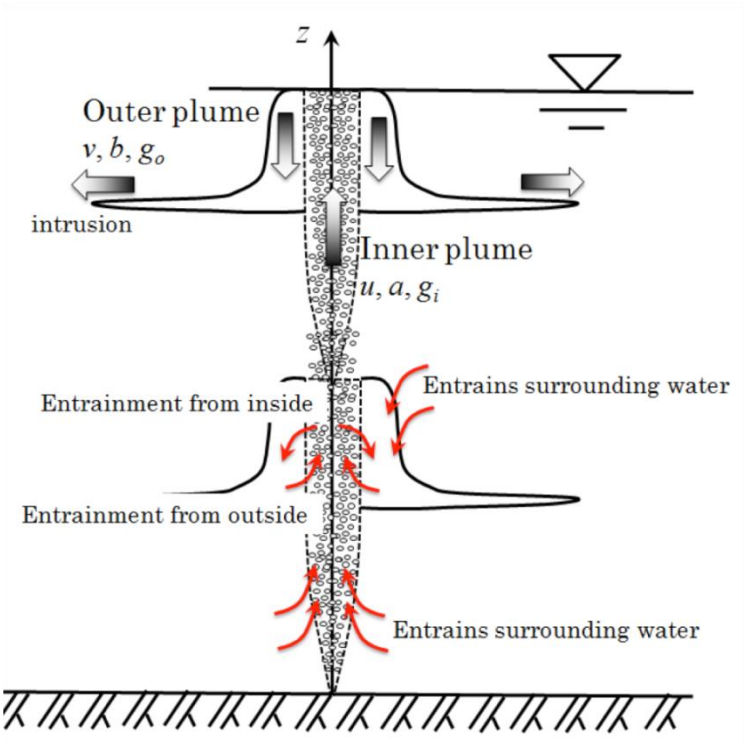

Fig. 12 Double plume model of an air aeration

Aeration Process Described in the Vertical 1D Double Plume Model

TITech-WARM adopts the double plume model developed by Asaeda &Imberger[vi] to describe plumes in a vertical gradient of density (caused by the difference of water temperature etc.). Its fundamental equations are explained in the next sub-section. As shown in **Fig. 12**, this model assumes a rising inner plume and a sinking outer plume. The speed of rising plume is generally several tens cm/sec, so it takes several tens seconds to reach the water surface in a water reservoir of several tens meters depth. This is the representative time scale of the plume model, and it's much shorter than that of main flows of actual water reservoirs. Hence, this model analyses the plumes in a steady state based on an assumption that density distribution etc. of surrounding water never vary within the time scale of the plumes.

When the *z*-axis is the vertical axis, inner and outer plume are assumed to be symmetric with respect to the vertical axis, and be shaped into cylinders which have radiuses of $a(z)$ and $b(z)$ respectively. Surrounding water is assumed to be stratified vertically by its density$\rho_a(z)$. The inner plume and the outer plume respectively have vertical velocities $u(z)$ and $v(z)$, plume radiuses $a(z), b(z)$, and relative buoyancies $g'(z), g''(z)$ in the fundamental equations. All upward momentum of rising water is assumed to be included in the inner plume, and all downward momentum of sinking water is assumed to be included in the outer plume.

As shown in **Fig. 12**, the inner plume is entrained by the sent out bubbles and rises up. When the outer plume does not exist in the same depth, the surrounding water is assumed to be stationary and the amount of entrainment from the surrounding water into the inner plume$\beta u(z)$is proportional to the velocity of the rising inner plume $u(z)$ (β is its entrainment coefficient).

When the outer plume appears in the same depth, the inner plume is assumed to begin exchanges of water with it through entrainments. Some experiments show that the turbulences in the inner plume are generated by the difference of velocity between the inner and the outer plume. In contrast, the turbulences in the outer plume are shown to be generated by the difference between the velocity of sinking outer plume $v(z)$ and the stationary surrounding water. Therefore, the amount of entrainment from the outer plume into the inner plume$\beta(u(z) - v(z))$and that of the opposite direction $\gamma v(z)$ are assumed to be proportional to the velocity difference between the plumes $u(z) - v(z)$ and the velocity of the outer plume $v(z)$ respectively.

The inner plume loses its relative buoyancy and momentum while it's rising up. Then, when its momentum $u(z)$ becomes 0, it's assumed to leave from the bubbles and begin to sink as the outer plume. The relative buoyancy of the plume is transported from the bottom of the water reservoir, so the outer plume is heavier than the surrounding water and has negative buoyancy at the beginning. This is why the outer plume sinks. While the outer plume is sinking, it entrains some water not only from the inner plume but from the surrounding water which is lighter than itself (α is its entrainment coefficient). As a result, these entrainments deprive the outer plume of its relative buoyancy. Then the outer plume intrudes into the surrounding water horizontally when its relative buoyancy becomes 0 and the density of the surrounding water becomes the same with its.

Fundamental Equations of the Vertical 1D Double Plume Model

Following equations are derived from the previous assumptions by considering the conservation of the volume flux, the momentum flux, and the buoyancy flux in the inner plume;

For the Inner Plume

$$\frac{\mathrm{d}(a^2u)}{\mathrm{d}z} = 2\beta a(u-v) - 2\gamma av \quad \textbf{(1.53)}$$

$$\frac{\mathrm{d}(a^2u^2)}{\mathrm{d}z} = a^2g' + 2\beta av(u-v) - 2\gamma auv \quad \textbf{(1.54)}$$

$$\frac{\mathrm{d}(a^2ug')}{\mathrm{d}z} = a^2u\frac{g}{\bar{\rho}}\frac{\mathrm{d}\rho_\mathrm{a}}{\mathrm{d}z} - 2\beta a(u-v)g'' + 2\gamma av\tilde{g} + \frac{\mathrm{d}(a^2ugF)}{\mathrm{d}z} \quad \textbf{(1.55)}$$

For the Outer Plume

$$\frac{\mathrm{d}}{\mathrm{d}z}\{(b^2-a^2)v\} = -2\beta a(u-v) - 2\gamma av - 2\alpha bv \quad \textbf{(1.56)}$$

$$\frac{\mathrm{d}}{\mathrm{d}z}\{(b^2-a^2)v^2\} = -(b^2-a^2)g'' - 2\beta av(u-v) - 2\gamma auv \quad \textbf{(1.57)}$$

$$\frac{\mathrm{d}}{\mathrm{d}z}\{(b^2-a^2)vg''\} = (b^2-a^2)v\frac{g}{\bar{\rho}}\frac{\mathrm{d}\rho_\mathrm{a}}{\mathrm{d}z} - 2\beta a(u-v)g'' + 2\gamma av\tilde{g} \quad \textbf{(1.58)}$$

Here, z[m] is the height from the bubble outlet of the aeration device. $u(z), v(z)$ [m/s] are the average velocity of the rising and sinking plume, and $g'(z), g''(z)$ [m/s^2] are the buoyancy to the surrounding water of them, respectively.$\alpha, \beta,$ and γ are the entrainment coefficients of from the surrounding water to the outer plume, from the outer plume to the inner plume, and from the inner plume to the outer plume.$g'(z), g''(z)$ are defined with following equations:

$$g' = g(\rho_\mathrm{a} - \rho_\mathrm{i})\bar{\rho} \quad \textbf{(1.59)}$$

$$g'' = g(\rho_\mathrm{o} - \rho_\mathrm{a})\bar{\rho} \quad \textbf{(1.60)}$$

$\rho_\mathrm{a}, \rho_\mathrm{i},$ andρ_o [kg/m^3] are the density of the surrounding water, the inner plume, and the outer plume. $\bar{\rho}$ [kg/m^3] is the criterion density, and the vertical average of density above the aeration device is adopted for it.

$\tilde{g}$ is the part of g' which has been generated by water, and it's determined by the following equation:

$$\tilde{g} = g' - gF = g' - g\frac{Q_H}{\pi(a\lambda)^2(u+u_\mathrm{s})} \quad \textbf{(1.61)}$$

Here, F is the gas content of the inner plume's cross sections. Q_H is the gas flow rate at the depth H. λ is the ratio of the inner plume's radius to that of the central bubble plume. u_s [m/s] is the slip velocity.

The standard values of parameters are given by Asaeda & Imberger as follows:

$$\alpha = 0.083, \quad \beta = 0.042, \quad \gamma = 0.083, \quad u_s = 0.3, \quad \lambda = 0.72 \tag{1.62}$$

Computational Procedures of the Vertical 1D Double Plume Model

The radiuses, velocities, and buoyancies of assumed inner and outer plume are derived from previous equations of the double plume model. When the TITech-WARM solves these equations practically, each variable is converted into a dimensionless number as follows:

$$A = \frac{a}{2\alpha H}, \quad B = \frac{b}{2\alpha H}, \quad U = \frac{u}{u_s M_H^{-1/3}}, \quad V = \frac{v}{u_s M_H^{-1/3}}$$

$$G' = \frac{g'H}{u_s^2 M_H^{2/3}}, \quad G'' = \frac{g''H}{u_s^2 M_H^{2/3}}, \quad S^2 = B^2 - A^2, \quad Z = \frac{z}{H}$$

$$B' = \frac{\beta}{\alpha}, \quad \Gamma = \frac{\gamma}{\alpha}, \quad \tilde{G} = G' - \frac{1}{\tilde{A}^2(1 - xH_R)\left(U + M_H^{-1/3}\right)}$$

$$C_H = (4\pi)^{2/3}\alpha^{4/3}P_N^{2/3} \tag{1.63}$$

$$H_R = H/(H_A + H) \tag{1.64}$$

$$P_N = N^3H^4/Q_B g \tag{1.65}$$

$$M_H = Q_B g/4\pi\alpha^3 H u_s^3 \tag{1.66}$$

Here, H_A [m] is the hydraulic head of the atmospheric pressure.P_N and M_H are dimensionless numbers which represent the relationships of aeration with the strength of stratification and the amount of sent out bubbles respectively. N is the buoyancy frequency. H [m] is the total depth. Q_B [m^3/s] is the amount of sent out bubbles (air). The equations **(1.53)** ~ **(1.58)** are retold with these variables as follows:

For the Inner Plume

$$\frac{d(A^2u)}{dZ} = B'A(U - V) + \Gamma AV \tag{1.67}$$

$$\frac{d(A^2U^2)}{dZ} = A^2G' + B'AV(U - V) + \Gamma AUV \tag{1.68}$$

$$\frac{d(A^2UG')}{dZ} = -A^2UC_H - B'A(U - V)G'' + \Gamma AV\tilde{G} + \frac{d}{dZ}\left\{\frac{A^2U}{\tilde{A}^2(1 - xH_R)\left(U + M_H^{-1/3}\right)}\right\} \tag{1.69}$$

For the Outer Plume

$$\frac{d(S^2V)}{dZ} = -B'A(U - V) - \Gamma AV - B'V \tag{1.70}$$

$$\frac{\mathrm{d}(S^2V^2)}{\mathrm{d}Z} = -S^2G'' - B'AV(U-V) - \Gamma AUV \tag{1.71}$$

$$\frac{\mathrm{d}(S^2VG'')}{\mathrm{d}Z} = S^2VC_H - B'A(U-V)G'' + \Gamma AV\tilde{G} \tag{1.72}$$

Then, the plume model integrates simultaneous equations **(1.67)** ~ **(1.72)**with the forth-order accuracy Runge-Kutta method (RK4). First, the plume model assumes the initial conditions of the inner plume $A, AU,$ and $A^2UG'$to be following series solutions (like Asaeda & Imberger):

$$A = Z\left\{0.6 + 0.01719M_H^{-1/3}Z^{1/3} - 0.002527M_H^{-2/3}Z^{2/3} + Z(-0.04609 + 0.000031M_H^{-1}) + \cdots\right\} \tag{1.73}$$

$$U = Z^{-1/3}\left\{1.609 - 0.3195M_H^{-1/3}Z^{1/3} + 0.06693M_H^{-2/3}Z^{2/3} + Z(0.4536 - 0.0105M_H^{-1}) + \cdots\right\} \tag{1.74}$$

$$A^2UG' = -ZA^2UC_H + \frac{A^2U}{\tilde{A}^2(1 - ZH_\mathrm{R})\left(U + M_H^{-1/3}\right)} \tag{1.75}$$

Z is the dimensionless distance from the bubble outlet of an aeration device, and it's assumed to be the same with vertical mesh intervals used in the plume model. Then, with the initial conditions (on the bottom of the water reservoir) **(1.73)** ~ **(1.75)**, the plume model integrates equations **(1.67)** ~ **(1.69)**upward until the speed of rising inner plume becomes 0.

Next, the plume model assumes that the inner plume turns into the sinking outer plume there, and then begins computations of the outer plume. Its initial condition is the conservation of volume flow rate between the plumes:

$$S^2V = A^2U \tag{1.76}$$

If the inner plume reaches the water surface, it spreads horizontally before sinks. In this case, the volume flux of sinking plume becomes larger because the plume entrains a lot of water around the water surface. Thus, according to Asaeda & Imberger, the following volume flux needs to be added if the plume begins to sink from the water surface:

$$Q_\mathrm{e} = \frac{C_\mathrm{e}Q_0^{3/4}}{gN^{-5/4}} \tag{1.77}$$

Here, Q_0 is the amount of air which released into the atmosphere (of the atmospheric pressure). C_e is an arbitrary constant (around 1). The initial condition of the outer plume's momentum flux is

$$S^2V^2 = S^2G''(\tilde{Z} - Z) \tag{1.78}$$

This is an integration of the equation **(1.71)**from the height where the plume model begins the computation of the outer plume Z to the height where the outer plume begins$\tilde{Z}$. However, the second and third term on the right side of the equation **(1.71)** is assumed to be 0 here. This is because the outer plume entrains little water at the begging of sinking.

At last, the initial condition of the sinking outer plume's relative buoyancy is assumed to be that of the inner plume except what comes from the bubbles. Then it's determined by subtracting **(1.72)** from **(1.70)**:

$$S^2VG'' = A^2UG' - \frac{A^2U}{\tilde{A}^2(1 - ZH_{\mathrm{R}})\left(U + M_H^{-1/3}\right)} \quad \textbf{(1.79)}$$

With this initial condition, the plume model integrates equations **(1.70)** ~ **(1.72)** downwards until the buoyancy of outer plume G'' becomes 0.

The plume model repeats the integrations of **(1.67)** ~ **(1.72)** in this way until all variables converge. Incidentally, when an inner plume turns into an outer plume, the bubbles entrain water again and generate a new inner plume (because bubbles never vanish until they reach the water surface). Thus the plume model also computes the equations **(1.67)** ~ **(1.72)** for the new plume with an assumption that the height of its beginning is the bubble outlet of the aeration device.

Introduction into the Main Flow Computaions

The radiuses of plumes generated by an aeration device are generally around 10 m at most. In most cases, they are smaller than the spatial scale of main flows and horizontal intervals of computational meshes. Thus, as shown in **Fig. 13**, we can assume an aeration plume to be a vertically long and horizontally narrow domain (I). Here, the vertical distribution of vertical volume flux is derived from the computations of the double plume model as follows:

$$Q_{\mathrm{total}}(z) = \pi a^2 u - \pi(b^2 - a^2)v \quad \textbf{(1.80)}$$

z-directional derivative of **(1.78)**$\mathrm{d}Q_{\mathrm{totoal}}(z)/\mathrm{d}z$ is an increase of the plume's volume, and it represents the amount of water sink (drainage) per unit length (1 m) by entrainments. Then, as shown in **Fig. 13**, we can assume an aeration plume in a computational mesh to be imaginary sources and sinks of water at certain depth. Therefore TITech-WARM introduces the plume model into main flow computations with the following procedures:

(1): Specify a computational mesh which contains an aeration device.

(2): Locate computational meshes equally between the outlet of aeration device and the water surface. Then get density distribution $\rho(z)$ by interpolations of computational results which conducted on surrounding Soroban meshes.

(3): Get $Q_{\mathrm{total}}(z)$ by computing the double plume model with $\rho(z)$. Then get the amount of water sources/sinks $\mathrm{d}Q_{\mathrm{totoal}}(z)/\mathrm{d}z$.

(4): Distribute water source / sink $\mathrm{d}Q_{\mathrm{totoal}}(z)/\mathrm{d}z$ to surrounding mesh axes with the distance to each as the weighing coefficient. Then apply the effect of water sources/sinks to the pressure (velocity) computations by modifying conservation equations as follows (A and B are the distances from the center of plume to the neighboring mesh axis (x_i, y_{ij}) and (x_l, y_{lm}) respectively):

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=-\frac{B}{A+B}\frac{\mathrm{d}Q_{\mathrm{totoal}}}{\mathrm{d}z} \quad \text{on mesh points of the mesh axis } (x_i, y_{ij})$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=-\frac{A}{A+B}\frac{\mathrm{d}Q_{\mathrm{totoal}}}{\mathrm{d}z} \quad \text{on mesh points of the mesh axis } (x_l, y_{lm})$$

(5): Repeat these procedures on the next computational step (after the computations of advection etc.).

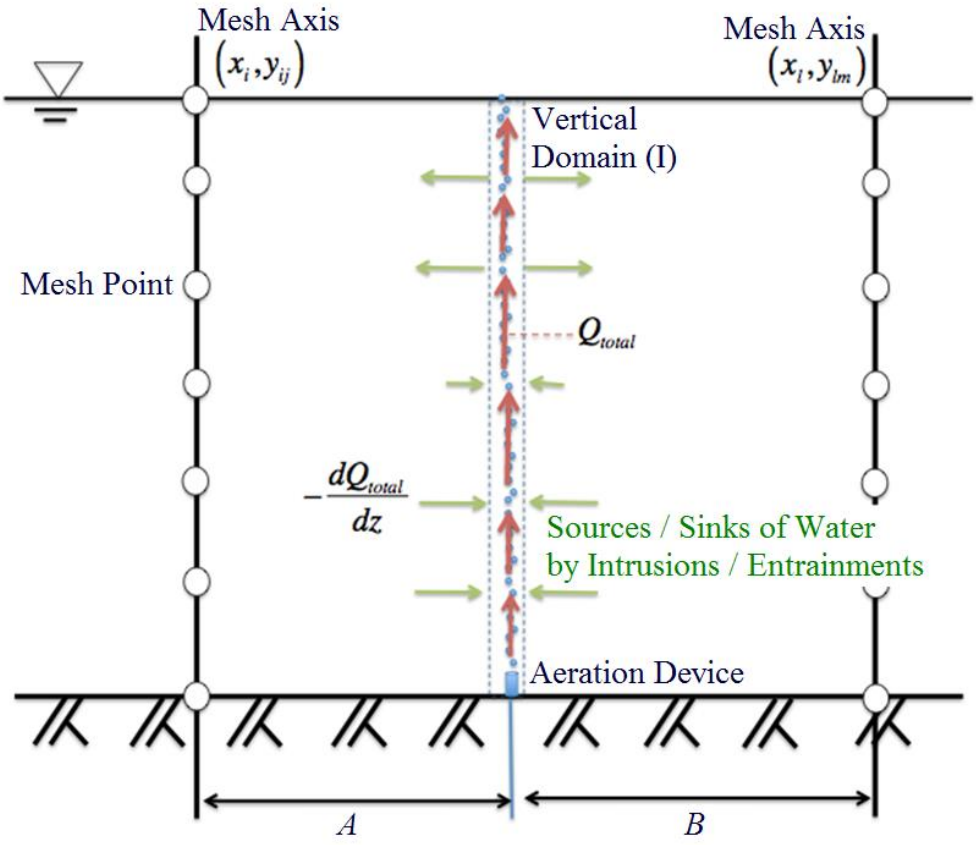

Fig. 13 Assumption of aeration plumes in a computational mesh

[vi] Asaeda, T. and Imberger, J.: Structure of bubble plumes in linearly stratified environments, *J. Fluid Mech.*, vol.249, (1993), pp.35-57.

Computational Methods

This chapter explains the outlines of computational methods used in TITech-WARM.

Locations of Variables

The following physical quantities are arranged on 3-dimansional Soroban meshes by the collocated arrangement: velocity components u, v, w, water temperature T, salinity s, density ρ, pressure p, turbulent kinetic energy k, turbulent dispersion rate ε, and user-defined scalar ϕ_{sp}. In other words, variables $u_{ijk}^n, v_{ijk}^n, w_{ijk}^n, T_{ijk}^n, s_{ijk}^n, \rho_{ijk}^n, p_{ijk}^n, k_{ijk}^n, \varepsilon_{ijk}^n,$ and $(\phi_{\mathrm{sp}})_{ijk}^n$ are located on the *k*-th mesh point of the *j*-th mesh axis of the *i*-th mesh plane: (x_i, y_{ij}, z_{ijk}). In the source codes of TITech-WARM (written in C language), these 3-dimensional physical quantities are described as members of a ValueOnVolume-type structure. For example, you can get the x-directional velocity u on the mesh point (x_i, y_{ij}, z_{ijk}) by referring uu_p->ff[i][j][k].

On the other hand, the water level $h(t, x, y)$ and the height of river bed $b(x, y)$ are the functions of x and y, and they are arranged on Soroban mesh axes. In other words, variables h_{ij}^n and b_{ij} are located on the *j*-th mesh axis of *i*-th mesh plane: $(x_i, y_{ij},)$.In the source codes, these 2-dimensional physical quantities are described as members of a ValueOnSurface-type structure. For example, you can get the water level h on the mesh axis (x_i, y_{ij}) by referring water_level_p->ff[i][j].

TITech-WARM computes advections of the following 3-dimensional physical quantities with CIP interpolations on 3-simensional Soroban meshes: velocity components u, v, w, water temperature T, salinity s, turbulent kinetic energy k, turbulent dispersion rate ε, and user-defined scalar ϕ_{sp}. Thus, not only these physical quantities but their x, y, and z-directional spatial gradient $g_{\mathrm{x}}, g_{\mathrm{y}},$ and g_{z} are also arranged and computed on each mesh point. In the source codes, these gradients are also the members of a ValueOnVolume-type structure, so you can get them by referring uu_p->gx[i][j][k], uu_p->gy[i][j][k], uu_p->gz[i][j][k] etc..

CIP Spatial Interpolations on Soroban Meshes

TITech-WARM uses CIP interpolations to compute advection terms. This section explains the procedures of CIP interpolations on 3-dimensional Soroban meshes. Here, a physical quantity to be interpolated and its spatial gradients are described as $f(x,y,z)$, $g_x(x,y,z)$, $g_y(x,y,z)$, and $g_z(x,y,z)$. The physical quantity of $(X_\mathrm{q}, Y_\mathrm{q}, Z_\mathrm{q})$ is to be estimated through interpolations of surrounding mesh points. TITech-WARM adopts a kind of CIP interpolation called Type-M CIP, in which repetitions of 1-dimensional CIP interpolations compose a 3-dimensional interpolation as follows.

As shown in **Fig. 14**, a pair of neighboring mesh plains which contains the coordinates $(X_\mathrm{q}, Y_\mathrm{q}, Z_\mathrm{q})$is detected by the bisectional method or a like. The indexes of these mesh planes *iup* and *idn* have the following relationship:

$$x_{idn} \le X_\mathrm{q} < x_{iup}, \qquad iup = idn + 1 \tag{2.1}$$

The points where perpendicular lines from $(X_\mathrm{q}, Y_\mathrm{q}, Z_\mathrm{q})$ intersect the mesh plains are named A_{iup} and A_{idn}, and their coordinates are given as $(x_{iup}, Y_\mathrm{q}, Z_\mathrm{q})$ and $(x_{idn}, Y_\mathrm{q}, Z_\mathrm{q})$.

Next, a pair of neighboring mesh axes which contains A_{iup} is detected on the plane *iup*, and each of itis named *jup_on_iup* and *jup_on_idn*. The points where perpendicular lines from A_{iup} intersect *jup_on_iup* and *jup_on_idn* are respectively named B and C. Then 2 pairs of mesh points which respectively contain B or C are detected, and each of them is named *kup_on_jup_on_iup*, *kdn_on_jup_on_iup*, *kup_on_jdn_on_iup*, and *kdn_on_jdn_on_iup*.

Thus, physical quantity and its gradients of A_{iup} is estimated from those of previous 4 mesh points with following 1-dimensional CIP interpolations;The physical quantity and its z-directional gradient of B:$f^\mathrm{B}$and $g_\mathrm{z}^\mathrm{B}$are computed by a z-directionalCIP interpolation with those of *kup_on_jdn_on_iup*and*kdn_on_jdn_on_iup*. The other gradients of B: g_x^B and g_y^B are computed with linear interpolations with those of *kup_on_jdn_on_iup*, *kdn_on_jdn_on_iup*. In the same way, the physical quantity and its gradients of C: $(f^\mathrm{C}, g_\mathrm{x}^\mathrm{C}, g_\mathrm{y}^\mathrm{C}, g_\mathrm{z}^\mathrm{C})$ are also computed with CIP and linear interpolations between *kup_on_jup_on_iup*, *kdn_on_jup_on_iup*. Then the physical quantity and its gradients of A_{iup} are computed y-directional interpolations between B and C. The physical quantity and its y-directional gradient: $f^{\mathrm{A}iup}$and $g_\mathrm{y}^{\mathrm{A}iup}$ are computed by a y-directional CIP interpolation, and other gradients: $g_\mathrm{x}^{\mathrm{A}iup}$ and $g_\mathrm{z}^{\mathrm{A}iup}$ are computed with linear interpolations.

In the same way, the physical quantity and its gradients of A_{idn}: $(f^{\mathrm{A}idn}, g_\mathrm{x}^{\mathrm{A}idn}, g_\mathrm{y}^{\mathrm{A}idn}, g_\mathrm{z}^{\mathrm{A}idn})$ are computed on the plane idn. Then at last, the physical quantity (and its x-directional gradient) of $(X_\mathrm{q}, Y_\mathrm{q}, Z_\mathrm{q})$ is computed with a x-directional CIP interpolation betweenA_{iup} and A_{idn}.

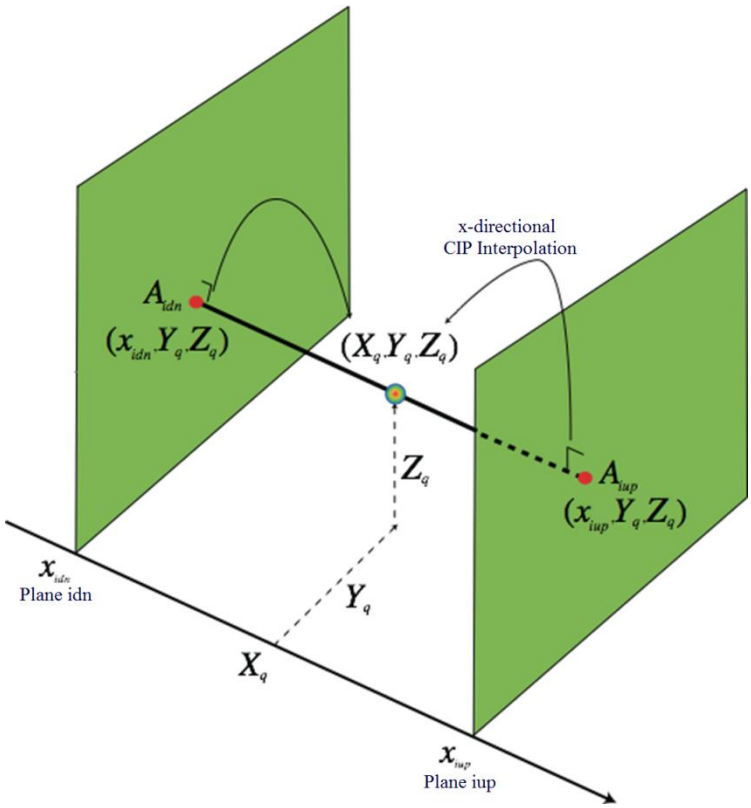

Fig. 14 CIP interpolation between mesh planes

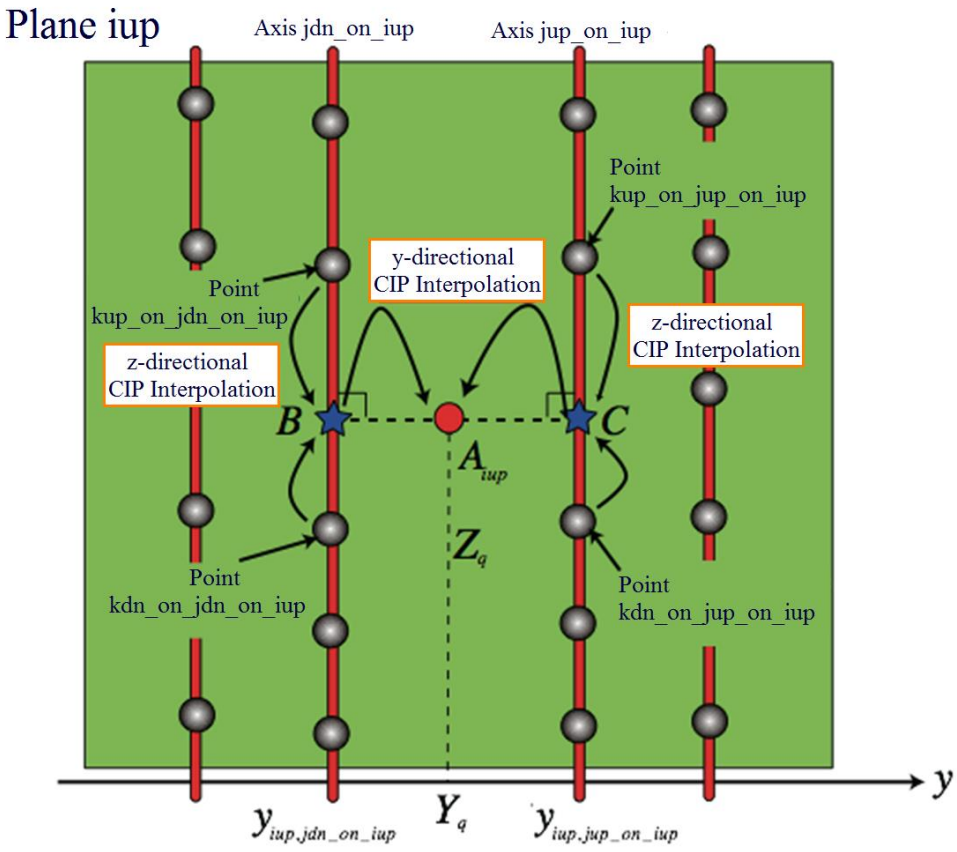

Fig. 15 CIP interpolations in a mesh plane

Spatial Difference Approximations on Soroban Meshes

The computations of diffusions, estimations of turbulent source terms, and so on are conducted with finite difference approximations on the 3-dimansional Soroban meshes. This section explains the derivations of finite difference approximation equations on the Soroban mesh. Here, the gradients on the *k*-th mesh point of the *j*-th mesh axis of the *i*-th mesh plane: (*i*, *j*, *k*)-th mesh point are to be estimated.

x-directional spatial differences

As shown in **Fig. 16**, the planes which are +x-side or–x-side next to plane *i* respectively have an index:$iup = i + 1 \text{or} idn = i - 1$. The points where perpendicular lines from the (*i*, *j*, *k*)-th mesh point respectively intersect x-directionally neighboring planes are named A_{iup} and A_{idn}, and their coordinates are given as $(x_{iup}, y_{ij}, z_{ijk})$ and $(x_{idn}, y_{ij}, z_{ijk})$. Then the x-directional spatial difference of physical quantity f on (*i*, *j*, *k*)-th mesh point is estimated with that of A_{iup} and A_{idn}: $f^{\mathrm{A}iup}$and $f^{\mathrm{A}idn}$as follows;

$$\left(\frac{\partial f}{\partial x}\right)_{ijk} = \frac{f^{\mathrm{A}iup} - f^{\mathrm{A}idn}}{x_{iup} - x_{idn}} \tag{2.2}$$

$$\begin{aligned}\left(\frac{\partial^2 f}{\partial x^2}\right)_{ijk} = {} & \frac{f^{\mathrm{A}iup}}{(x_{iup} - x_i)(x_{iup} - x_{idn})/2} \\ & - \left(\frac{1}{(x_{iup} - x_i)(x_{iup} - x_{idn})/2} + \frac{1}{(x_i - x_{idn})(x_{iup} - x_{idn})/2}\right) f_{ijk} \\ & + \frac{f^{\mathrm{A}idn}}{(x_i - x_{idn})(x_{iup} - x_{idn})/2}\end{aligned} \tag{2.3}$$

$f^{\mathrm{A}iup}$ and $f^{\mathrm{A}idn}$ are respectively computed with liner interpolations on the plane *iup* and *idn*. For example, $f^{\mathrm{A}iup}$ is computed with a y-directional linear interpolation by using the physical quantities of B and C:f^{B} and f^{C}as follows;

$$f^{\mathrm{A}iup} = \left(1 - \alpha_{iup}\right) f^{\mathrm{B}} + \alpha_{iup} f^{\mathrm{C}} \tag{2.4}$$

$$\alpha_{iup} = \frac{y_{ij} - y_{iup,jdn\_on\_iup}}{y_{iup,jup\_on\_iup} - y_{iup,jdn\_on\_iup}} \tag{2.5}$$

As shown in **Fig. 17**, f^{B} and $f^{\mathrm{C}}$are computed with z-directional linear interpolation on the mesh axes on which B or C are located: *jdn_on_iup* and *jup_on_iup* as follows;

$$\begin{aligned}f^{\mathrm{B}} = {} & \left(1 - \beta_{jdn\_on\_iup}\right) f_{iup,jdn\_on\_iup,kdn\_on\_jdn\_on\_iup} \\ & + \beta_{jdn\_on\_iup} f_{iup,jdn\_on\_iup,kup\_on\_jdn\_on\_iup}\end{aligned} \tag{2.6}$$

$$\beta_{jdn\_on\_iup} = \frac{z_{ijk} - z_{iup,jdn\_on\_iup,kdn\_on\_jdn\_on\_iup}}{z_{iup,jdn\_on\_iup,kup\_on\_jdn\_on\_iup} - z_{iup,jdn\_on\_iup,kdn\_on\_jdn\_on\_iup}} \tag{2.7}$$

$$f^{\mathrm{C}} = \left(1 - \beta_{jup\_on\_iup}\right) f_{iup,jup\_on\_iup,kdn\_on\_jup\_on\_iup} + \beta_{jup\_on\_iup} f_{iup,jup\_on\_iup,kup\_on\_jup\_on\_iup} \quad \textbf{(2.8)}$$

$$\beta_{jup\_on\_iup} = \frac{z_{ijk} - z_{iup,jup\_on\_iup,kdn\_on\_jup\_on\_iup}}{z_{iup,jup\_on\_iup,kup\_on\_jup\_on\_iup} - z_{iup,jup\_on\_iup,kdn\_on\_jup\_on\_iup}} \quad \textbf{(2.9)}$$

Then $f^{\mathrm{A}iup}$ is computed by substituting **(2.6)** ~ **(2.9)** for **(2.4)** as follows;

$$\begin{aligned} f^{\mathrm{A}iup} = {} & \left(1 - \alpha_{iup}\right)\left(1 - \beta_{jdn\_on\_iup}\right) f_{iup,jdn\_on\_iup,kdn\_on\_jdn\_on\_iup} \\ & + \left(1 - \alpha_{iup}\right)\beta_{jdn\_on\_iup} f_{iup,jdn\_on\_iup,kup\_on\_jdn\_on\_iup} \\ & + \alpha_{iup}\left(1 - \beta_{jup\_on\_iup}\right) f_{iup,jup\_on\_iup,kdn\_on\_jup\_on\_iup} \\ & + \alpha_{iup}\beta_{jup\_on\_iup} f_{iup,jup\_on\_iup,kup\_on\_jup\_on\_iup} \end{aligned} \quad \textbf{(2.10)}$$

In the same way, $f^{\mathrm{A}idn}$ is estimated with linear interpolations on the plane idn as follows;

$$\begin{aligned} f^{\mathrm{A}idn} = {} & \left(1 - \alpha_{idn}\right)\left(1 - \beta_{jdn\_on\_idn}\right) f_{idn,jdn\_on\_idn,kdn\_on\_jdn\_on\_idn} \\ & + \left(1 - \alpha_{idn}\right)\beta_{jdn\_on\_idn} f_{idn,jdn\_on\_idn,kup\_on\_jdn\_on\_idn} \\ & + \alpha_{idn}\left(1 - \beta_{jup\_on\_idn}\right) f_{idn,jup\_on\_idn,kdn\_on\_jup\_on\_idn} \\ & + \alpha_{idn}\beta_{jup\_on\_idn} f_{idn,jup\_on\_idn,kup\_on\_jup\_on\_idn} \end{aligned} \quad \textbf{(2.11)}$$

$$\alpha_{idn} = \frac{y_{ij} - y_{idn,jdn\_on\_iup}}{y_{idn,jup\_on\_idn} - y_{idn,jdn\_on\_idn}}$$

$$\beta_{jdn\_on\_idn} = \frac{z_{ijk} - z_{idn,jdn\_on\_idn,kdn\_on\_jdn\_on\_idn}}{z_{idn,jdn\_on\_idn,kup\_on\_jdn\_on\_idn} - z_{idn,jdn\_on\_idn,kdn\_on\_jdn\_on\_idn}}$$

$$\beta_{jup\_on\_idn} = \frac{z_{ijk} - z_{idn,jup\_on\_idn,kdn\_on\_jup\_on\_idn}}{z_{idn,jup\_on\_idn,kup\_on\_jup\_on\_idn} - z_{idn,jup\_on\_idn,kdn\_on\_jup\_on\_idn}}$$

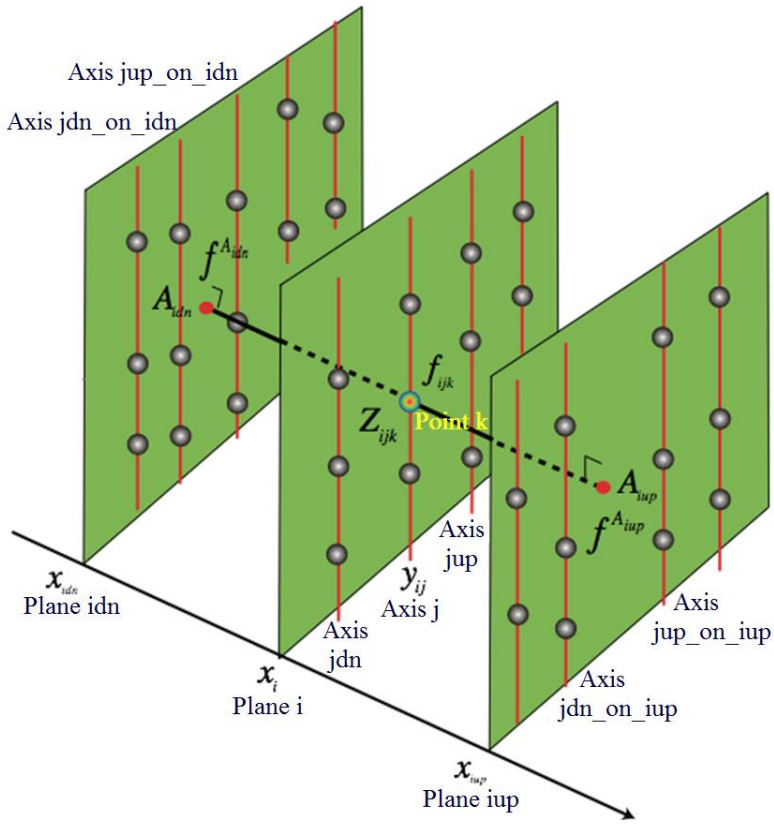

Fig. 16 x-directional differences between mesh planes

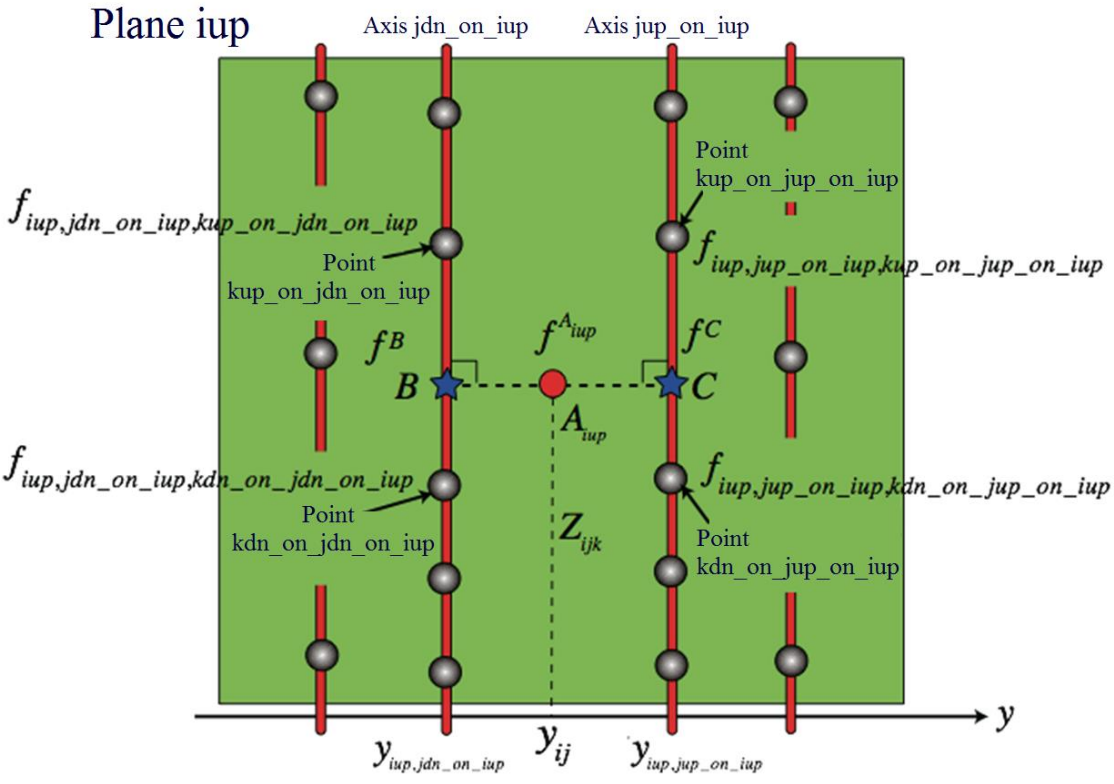

Fig. 17 x-directional differences between mesh points

y-directional spatial differences

As shown in **Fig. 18**, on the *i*-th plane, the axes which are +y-side or –y-side next to axis *j* respectively have an index:$jup = j + 1$ and $jdn = j - 1$. The points where perpendicular lines from the (*i*, *j*, *k*)-th mesh point respectively intersect y-directionally neighboring axes are named A_{jup} and A_{jdn}, and their coordinates are given as (x_i, y_{jup}, z_{ijk}) and (x_i, y_{jdn}, z_{ijk}). Then the y-directional spatial difference of physical quantity f on (*i*, *j*, *k*)-th mesh point is estimated with that of A_{jup} and A_{jdn}: $f^{\mathrm{A}jup}$ and $f^{\mathrm{A}jdn}$ as follows;

$$\left(\frac{\partial f}{\partial y}\right)_{ijk} = \frac{f^{\mathrm{A}jup} - f^{\mathrm{A}jdn}}{y_{i,jup} - y_{i,jdn}} \tag{2.12}$$

$$\left(\frac{\partial^2 f}{\partial y^2}\right)_{ijk} = \frac{f^{\mathrm{A}jup}}{(y_{i,jup} - y_{i,j})(y_{i,jup} - y_{i,jdn})/2} - \left(\frac{1}{(y_{i,jup} - y_{i,j})(y_{i,jup} - y_{i,jdn})/2} + \frac{1}{(y_{i,j} - y_{i,jdn})(y_{i,jup} - y_{i,jdn})/2}\right) f_{ijk} + \frac{f^{\mathrm{A}jdn}}{(y_{i,j} - y_{i,jdn})(y_{i,jup} - y_{i,jdn})/2} \tag{2.13}$$

$f^{\mathrm{A}jup}$ and $f^{\mathrm{A}jdn}$ are respectively computed with liner interpolations on the axis *jup* and *jdn*;

$$f^{\mathrm{A}jdn} = \left(1 - \alpha_{jdn}\right) f_{i,jdn,kdn\_on\_jdn} + \alpha_{jdn} f_{i,jdn,kup\_on\_jdn} \tag{2.14}$$

$$\alpha_{jdn} = \frac{z_{ijk} - z_{i,jdn,kdn\_on\_jdn}}{z_{i,jdn,kup\_on\_jdn} - z_{i,jdn,kdn\_on\_jdn}} \tag{2.15}$$

$$f^{\mathrm{A}jup} = \left(1 - \alpha_{jup}\right) f_{i,jup,kdn\_on\_jup} + \alpha_{jup} f_{i,jup,kup\_on\_jup} \tag{2.16}$$

$$\alpha_{jup} = \frac{z_{ijk} - z_{i,jup,kdn\_on\_jup}}{z_{i,jup,kup\_on\_jup} - z_{i,jup,kdn\_on\_jup}} \tag{2.17}$$

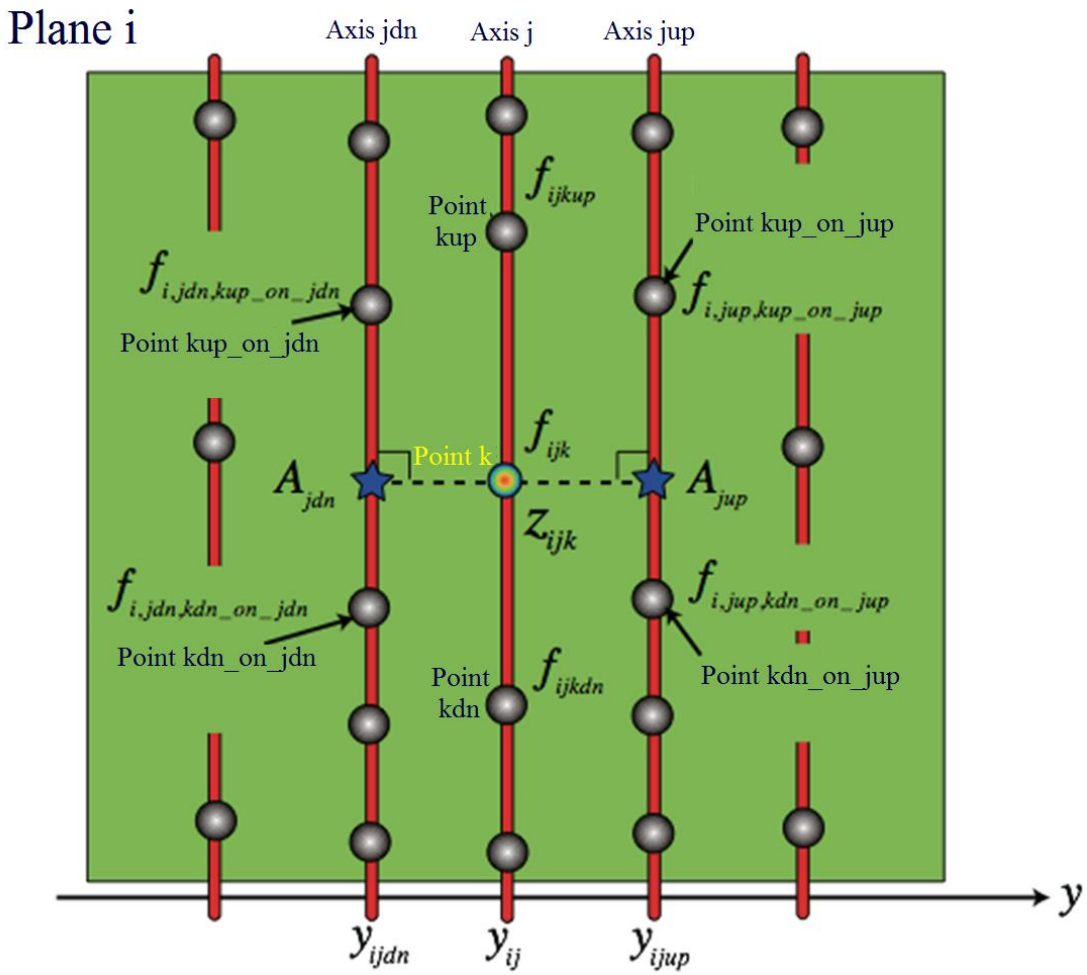

Fig. 18 y-directional differences between mesh points

z-directional spatial differences

On the *j*-th axis of *i*-th plane, the points which are +z-side or –z-side next to point *k* respectively have an index:$kup = k + 1$ and $kdn = k - 1$.Every mesh axis is parallel to vertical direction, so the z-directional spatial difference of physical quantity f on (*i*, *j*, *k*)-th mesh point is estimated as follows;

$$\left(\frac{\partial f}{\partial z}\right)_{ijk} = \frac{f_{i,j,kup} - f_{i,j,kup}}{z_{i,j,kup} - z_{i,j,kdn}} \tag{2.18}$$

$$\left(\frac{\partial^2 f}{\partial z^2}\right)_{ijk} = \frac{f_{i,j,kup}}{(z_{i,j,kup} - z_{i,j,k})(z_{i,j,kup} - z_{i,j,kdn})/2} - \left(\frac{1}{(z_{i,j,kup} - z_{i,j,k})(z_{i,j,kup} - z_{i,j,kdn})/2} + \frac{1}{(z_{i,j,k} - z_{i,j,kdn})(z_{i,j,kup} - z_{i,j,kdn})/2}\right) f_{ijk} + \frac{f_{i,j,kdn}}{(z_{i,j,k} - z_{i,j,kdn})(z_{i,j,kup} - z_{i,j,kdn})/2} \tag{2.19}$$

Computational Flow

Time Splitting Method

TITech-WARM is based on the time splitting method. It "splits" fundamental equations into several physical phases and computes them one by one to compute time evolutions. The fundamental equations of TITech-WARM are the follows;

$$\frac{\mathrm{D}u}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial x} - \frac{2}{3}\frac{\partial k}{\partial x} + \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial u}{\partial z}\right) + \frac{uv}{R} + f_{\mathrm{co}}v + \frac{\tau_{\mathrm{x}}}{\rho} \quad \textbf{(2.20)}$$

$$\frac{\mathrm{D}v}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial y} - \frac{2}{3}\frac{\partial k}{\partial y} + \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial v}{\partial z}\right) + \frac{u^2}{R} + f_{\mathrm{co}}u + \frac{\tau_{\mathrm{y}}}{\rho} \quad \textbf{(2.21)}$$

$$\frac{\mathrm{D}w}{\mathrm{D}t} = -\frac{1}{\rho}\frac{\partial p}{\partial z} - \frac{2}{3}\frac{\partial k}{\partial z} + \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial w}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial w}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial w}{\partial z}\right) + \frac{\tau_{\mathrm{z}}}{\rho} - g \quad \textbf{(2.22)}$$

$$\frac{\mathrm{D}k}{\mathrm{D}t} = P_k - \varepsilon + G_k + \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_k}\frac{\partial k}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_k}\frac{\partial k}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_k}\frac{\partial k}{\partial z}\right) \quad \textbf{(2.23)}$$

$$\frac{\mathrm{D}\varepsilon}{\mathrm{D}t} = (C_1 P_k - C_2\varepsilon)\frac{\varepsilon}{k} + C_1(1 - C_3)\frac{\varepsilon}{k}G_k + \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial z}\right) \quad \textbf{(2.24)}$$

$$\frac{\mathrm{D}s}{\mathrm{D}t} = \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial z}\right) \quad \textbf{(2.25)}$$

$$\frac{\mathrm{D}T}{\mathrm{D}t} = \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial z}\right) + S_T \quad \textbf{(2.26)}$$

$$\frac{\mathrm{D}\phi_{\mathrm{sp}}}{\mathrm{D}t} = w_{\mathrm{settlement}}\frac{\partial \phi_{\mathrm{sp}}}{\partial z} + \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial z}\right) + S_{\phi\mathrm{sp}} \quad \textbf{(2.27)}$$

$$\frac{\partial h}{\partial t} + \frac{\partial m}{\partial x} + \frac{\partial n}{\partial y} = 0 \quad \textbf{(2.28)}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = \begin{cases} 0 & \text{without aeration} \\ -\dfrac{\mathrm{d}Q_{\mathrm{total}}}{\mathrm{d}z} & \text{with aeration} \end{cases} \quad \textbf{(2.29)}$$

$$\rho(t, x, y, z) = \rho(s, T, \phi_{\mathrm{sp}})$$

$$\frac{\mathrm{D}}{\mathrm{D}t} = \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + w\frac{\partial}{\partial z}$$

Fig. 19 shows the computational flow of TITech-WARM. The following contents explain the outline of each part.

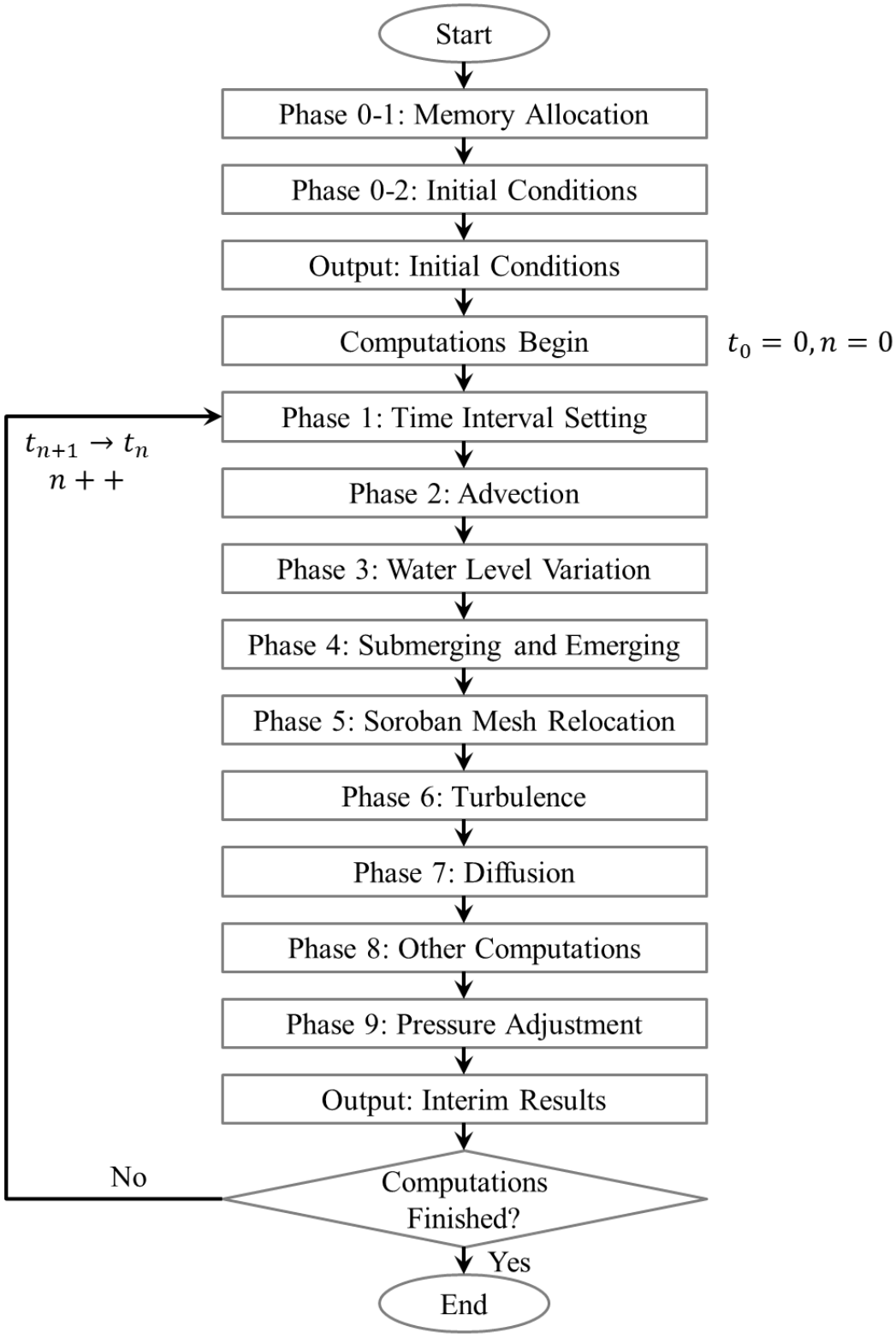

Fig. 19 Computational flow of TITech-WARM

Phase 0-1:Memory Allocation

The variables which will be used in computations are allocated on the memories of your computer(s). Initialization of MPI libraries is also conducted here. This phase is conducted by the memory_allocation_phase function.

Phase 0-2:Configurations of Initial Conditions

Topographic data files are read configurations of Soroban meshes are conducted. Arrangement of sub-domains (for parallel computations) and initial conditions of variables are also determined here. This phase is conducted by the initial_condition_phase function. See pp. 95 ~ to refer how to configure initial conditions.

Phase 1:Determination of Time Interval

The computational time interval to the next computational time step is determined by the CFL condition and so on. See the time_control function to refer precise procedures for determination.

Phase 2:Computations of Advection

The left sides of equations **(2.20)** ~ **(2.27)** corresponds to the following equations:**(2.30)** ~ **(2.37)**. They are solved with the values at$t_n$forvelocity components u, v, w, water temperature T, salinity s, turbulent kinetic energy k, turbulent dispersion rate ε, and user-defined scalar ϕ_{sp} to get those of after advections: u^{adv}, v^{adv}, w^{adv}, T^{adv}, s^{adv}, k^{adv}, $\varepsilon^{\mathrm{adv}}$, and $(\phi_{\mathrm{sp}})^{\mathrm{adv}}$.

$$\frac{\partial u}{\partial t} + u^n \frac{\partial u}{\partial x} + v^n \frac{\partial u}{\partial y} + w^n \frac{\partial u}{\partial z} = 0 \quad \textbf{(2.30)}$$

$$\frac{\partial v}{\partial t} + u^n \frac{\partial v}{\partial x} + v^n \frac{\partial v}{\partial y} + w^n \frac{\partial v}{\partial z} = 0 \quad \textbf{(2.31)}$$

$$\frac{\partial w}{\partial t} + u^n \frac{\partial w}{\partial x} + v^n \frac{\partial w}{\partial y} + w^n \frac{\partial w}{\partial z} = 0 \quad \textbf{(2.32)}$$

$$\frac{\partial T}{\partial t} + u^n \frac{\partial T}{\partial x} + v^n \frac{\partial T}{\partial y} + w^n \frac{\partial T}{\partial z} = 0 \quad \textbf{(2.33)}$$

$$\frac{\partial s}{\partial t} + u^n \frac{\partial s}{\partial x} + v^n \frac{\partial s}{\partial y} + w^n \frac{\partial s}{\partial z} = 0 \quad \textbf{(2.34)}$$

$$\frac{\partial k}{\partial t} + u^n \frac{\partial k}{\partial x} + v^n \frac{\partial k}{\partial y} + w^n \frac{\partial k}{\partial z} = 0 \quad \textbf{(2.35)}$$

$$\frac{\partial \varepsilon}{\partial t} + u^n \frac{\partial \varepsilon}{\partial x} + v^n \frac{\partial \varepsilon}{\partial y} + w^n \frac{\partial \varepsilon}{\partial z} = 0 \quad \textbf{(2.36)}$$

$$\frac{\partial \phi_{\mathrm{sp}}}{\partial t} + u^n \frac{\partial \phi_{\mathrm{sp}}}{\partial x} + v^n \frac{\partial \phi_{\mathrm{sp}}}{\partial y} + w^n \frac{\partial \phi_{\mathrm{sp}}}{\partial z} = 0 \quad \textbf{(2.37)}$$

The local theoretical solutions of these advection equations are given as follows;

$$u_{ijk}^{\mathrm{adv}} = u(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) = u(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t) \quad \textbf{(2.38)}$$

$$v_{ijk}^{\mathrm{adv}} = v(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) = v(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t) \quad \textbf{(2.39)}$$

$$w_{ijk}^{\mathrm{adv}} = w(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) = w(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t) \quad \textbf{(2.40)}$$

$$s_{ijk}^{\mathrm{adv}} = s(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) = s(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t) \quad \textbf{(2.41)}$$

$$T_{ijk}^{\mathrm{adv}} = T(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) = T(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t) \quad \textbf{(2.42)}$$

$$k_{ijk}^{\mathrm{adv}} = k(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) = k(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t) \quad \textbf{(2.43)}$$

$$\varepsilon_{ijk}^{\mathrm{adv}} = \varepsilon(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) = \varepsilon(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t) \quad \textbf{(2.44)}$$

$$\begin{aligned} \left(\phi_{\mathrm{sp}}\right)_{ijk}^{\mathrm{adv}} &= \phi_{\mathrm{sp}}(t_n + \Delta t, x_i, y_{ij}, z_{ijk}) \\ &= \phi_{\mathrm{sp}}\left(t_n, x_i - \left(u_{\mathrm{sp}}\right)_{ijk}^n \Delta t, y_{ij} - \left(v_{\mathrm{sp}}\right)_{ijk}^n \Delta t, z_{ijk} - \left(w_{\mathrm{sp}}\right)_{ijk}^n \Delta t\right) \end{aligned} \quad \textbf{(2.45)}$$

Thus, the equations **(2.30)** ~ **(2.37)** are solved by computing the right sides of **(2.38)** ~ **(2.45)**at advection origin:$(t_n, x_i - u_{ijk}^n \Delta t, y_{ij} - v_{ijk}^n \Delta t, z_{ijk} - w_{ijk}^n \Delta t)$ with CIP interpolations. This phase is conducted by the advection_phase function.

Phase 3: Computations of Water Level Variation

The change of water level is computed. Its fundamental equations are the following, which uses flow rate per a unit of width;

$$\frac{\partial h}{\partial t} + \frac{\partial m}{\partial x} + \frac{\partial n}{\partial y} = 0 \quad \textbf{(2.46)}$$

$$m(t, x, y) \equiv \int_{b(x,y)}^{h(t_n,x,y)} u^{\mathrm{other}} \mathrm{d}z \quad \textbf{(2.47)}$$

$$n(t, x, y) \equiv \int_{b(x,y)}^{h(t_n,x,y)} v^{\mathrm{other}} \mathrm{d}z \quad \textbf{(2.48)}$$

The water level at the time of next computational step (t_{n+1}): $h_{ij}^{n+1} = h(t_{n+1}, x_i, y_{ij})$ is computed by solving equation **(2.46)**. Here, an implicit solution is used to solve it, because velocities of water waves are generally faster than those of main flow: $u, v,$ and w. This phase is con-

ducted by the water_propagation_phase function. See the water_surface_propagation function to refer precise procedures of the solution.

Phase 4: Detecting Submerged/Emerged Mesh Axes

Whether each mesh axis is in a water area or not (i.e. in a land area) is judged by comparing the water level of next step: $h_{ij}^{n+1} = h(t_{n+1}, x_i, y_{ij})$ with certain lower limit. This phase is conducted by the submerge_and_emerge_phase function. See also the update_land_mask function to refer precise procedures.

If a mesh axis is judged to be in a land area, time evolutions of physical quantities (velocities etc.) are never computed on it. As shown in **Fig. 20**, land areas are distinguished with the land mask function: $land\_mask_{ij}^{n} = land\_mask(t_n, x_i, y_{ij})$. If a mesh axis is in a land area, it becomes$land_mask_{ij}^{n} = 1$, otherwise it becomes $land\_mask_{ij}^{n} = 0$ (To be precise, its value is stored in the array grid_p->land_mask[i][j]).

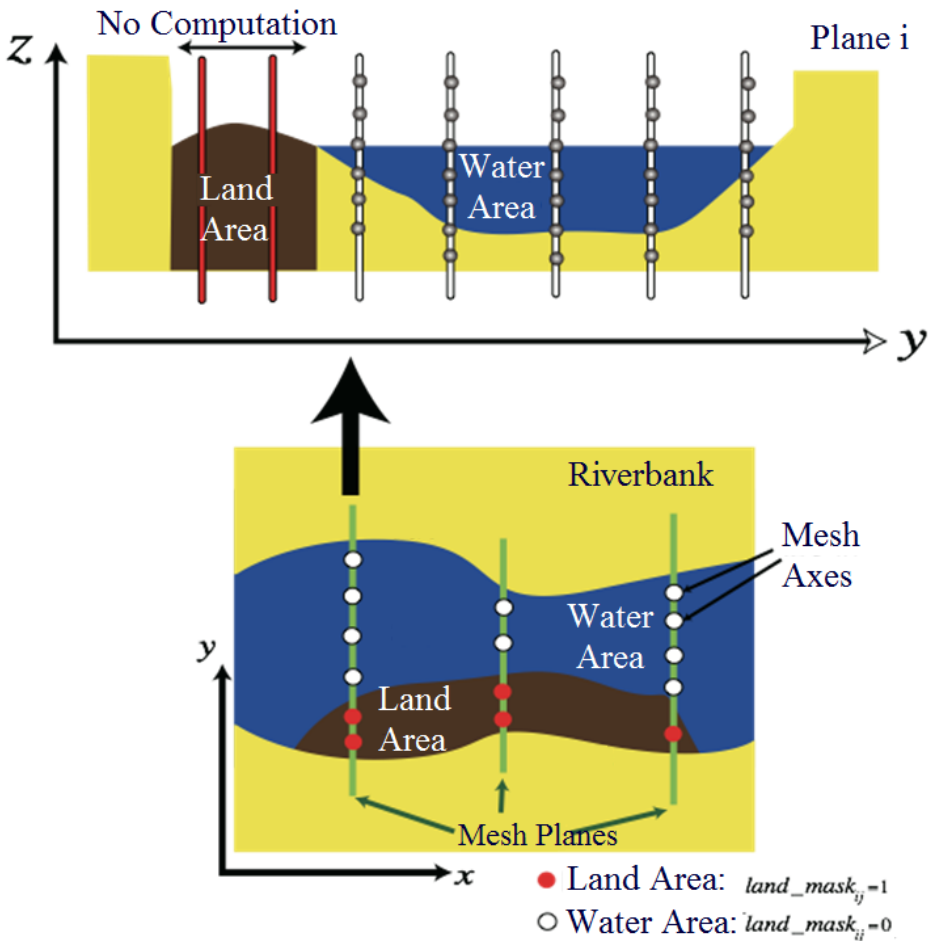

Fig. 20 Land area masking

The variation of boundary between water and land area is handled as follows;

Emerging (Water Area to Land Area)

When the water level $d = h_{ij}^{n+1} - b_{ij}$ becomes lower than its lower limit $d_{\min}$(prepared as an input data), the mesh axis (x_i, y_{ij}) is judged to be emerged. Then$land_mask_{ij} = 1$ is set and the axis is omitted from computations. As shown in **Fig. 21**, the water areas are computed with the assumption that a wall appears and the boundary conditions for riverbanks are applied between the water and the land area. Although **Fig. 21** shows this procedure in the y-z plane, it's also applied to the x-direction.

Submerging (Land Area to Water Area)

The submerging of mesh axes (in land areas) is judged with the relationships of water level among neighboring mesh axes. As shown in **Fig. 22**, when the difference between the height of river bed at the mesh axis b_{ij} and the water level of a neighboring mesh axis h_{lm}^{n+1} becomes larger than the lower limit of water level (i.e. when it becomes$d^* = h_{lm}^{n+1} - b_{ij} > d_{\min}$), the mesh axis is judged to be submerged. Then$land_mask_{ij} = 0$ is set and the axis is included into computations. Although **Fig. 22** shows this procedure in the y-z plane, it's also applied to the x-direction.

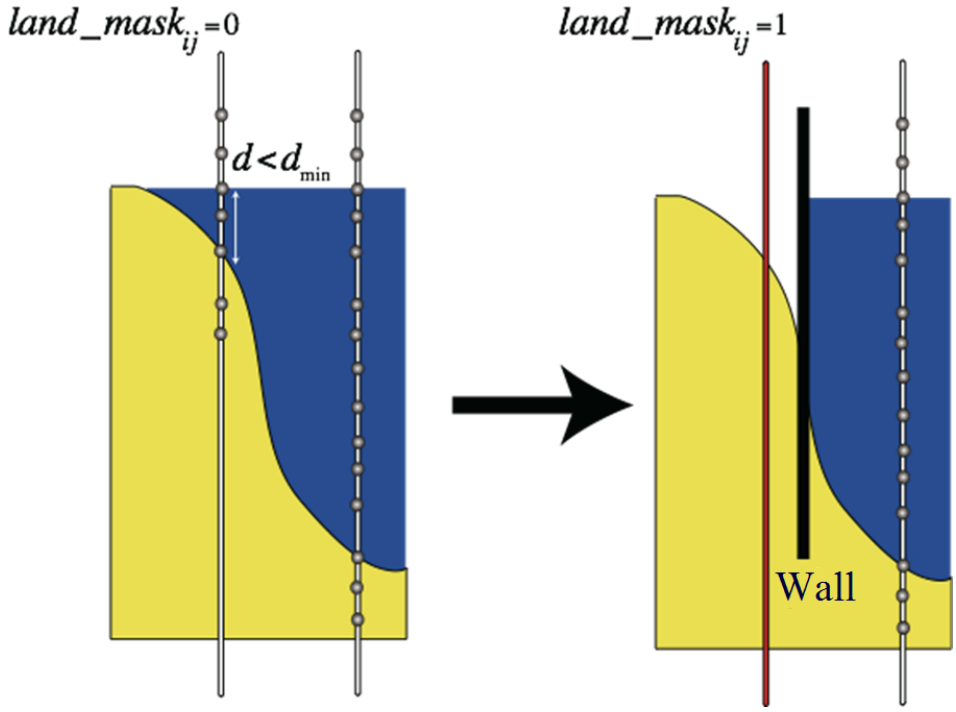

Fig. 21 Emerging (into a land area)

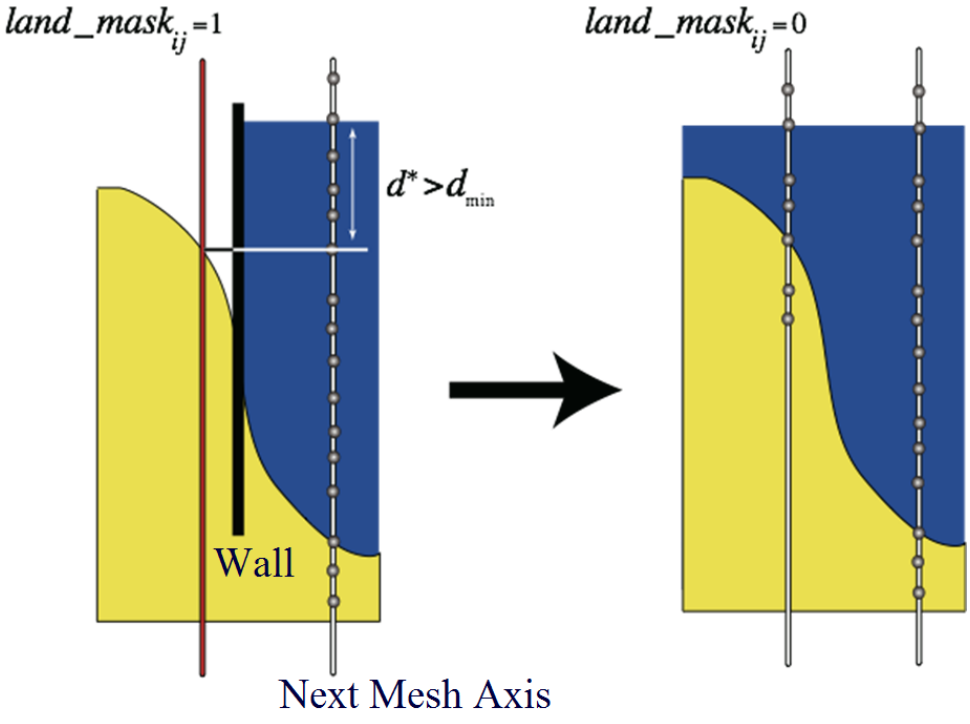

Fig. 22 Submerging (into a water area)

Phase 5:Relocations of Soroban Meshes

The vertical locations of Soroban mesh points (of every mesh axis) are updated based on the computed physical quantities' distributions and water level $h_{ij}^{n+1} = h(t_{n+1}, x_i, y_{ij})$. This phase is conducted by the soroban_update_phase function. As shown in **Fig. 23**, the new vertical locations: z_{ijk} are determined automatically with the monitor function M. Here, the old vertical locations (before relocation) are written as z_{ijk}^{n}, and the new vertical locations (after relocation) are written as z_{ijk}^{n+1}.

Assume that the vertical variation of a physical quantity $f^{n+1}(x, y, z)$ is needed to be represented precisely. Thus, the monitor function is computed on every mesh axis with the old vertical locations of mesh points $f^{n+1}(x, y, z)$ as follows;

$$M(x_i, y_{ij}, z) = \sqrt{1 + w_z \left| \frac{\partial f^{n+1}(x_i, y_{ij}, z^n)}{\partial z} \right|} \qquad \textbf{(2.49)}$$

Next, this function is divided into $N_i - 1$ domains (on every mesh axis) so that every domain has the same areas. Here, N_i is the number of mesh points of each mesh axis which was specified as an input topographic data at the beginning of computations. The boundaries of domains become the new vertical locations of mesh points. Then, a mesh point is always located on computed water surface: $h_{ij}^{n+1} = h(t_{n+1}, x_i, y_{ij})$. Besides, mesh points are concentrated around the area where the physical quantity $f^{n+1}(x, y, z)$ changes rapidly in this new distribution.

The coefficient w_z in the equation **(2.49)** adjusts the concentrations of mesh points; the larger it becomes, the sharper concentration of mesh points gets. If w_z is too large, or the water area is too shallow, the vertical intervals of mesh points might become too small. If an interval among relocated meshes is smaller than the lower limit (determined in advance), the number of mesh points N_i is reduced by 1 and mesh points are relocated over again (these procedures are repeated until all mesh intervals exceed the lower limit).

The 3-dimensional physical quantities (u, v, w, T, s, k, ε, and ϕ_{sp}) on the new locations z_{ijk}^{n+1} are obtained by interpolating those of old locations z_{ijk}^{n} on each mesh axis with vertical 1D CIP.

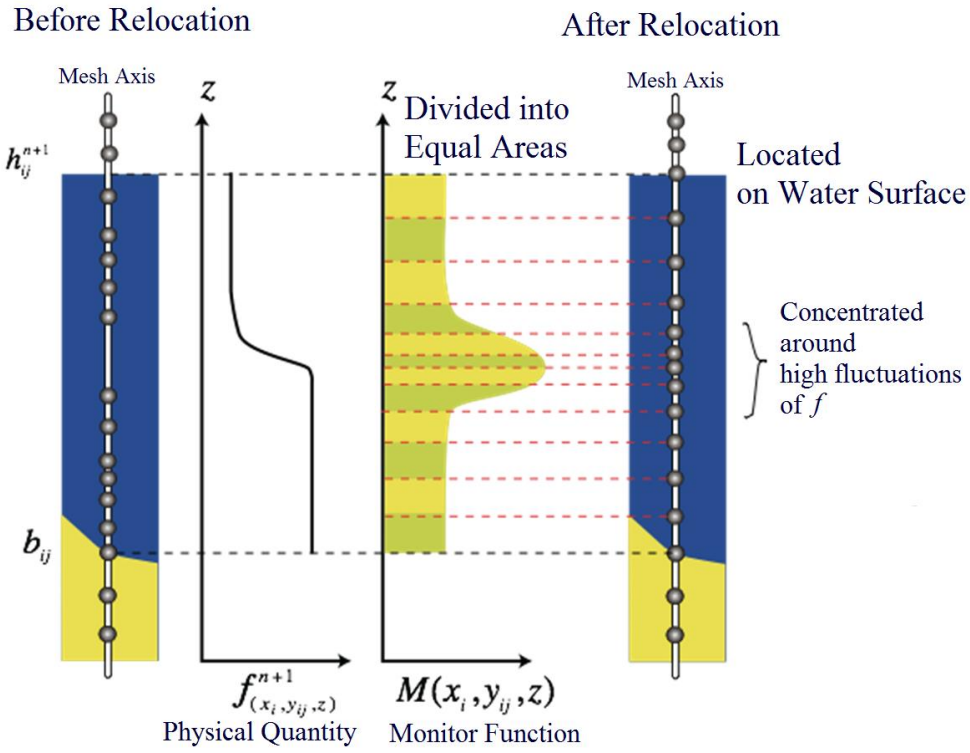

Fig. 23 Vertical relocation of Soroban mesh points

Phase 6:Computations of Turbulence

The turbulence production terms in the right sides of equations **(2.20)** ~ **(2.27)** corresponds to the following equations: **(2.50)** ~ **(2.57)**. They are solved with the physical quantities after advection computations: u^{adv}, v^{adv}, w^{adv}, T^{adv}, s^{adv}, k^{adv}, $\varepsilon^{\mathrm{adv}}$, and $(\phi_{\mathrm{sp}})^{\mathrm{adv}}$ to get those of after turbulence production: u^{tur}, v^{tur}, w^{tur}, T^{tur}, s^{tur}, k^{tur}, $\varepsilon^{\mathrm{tur}}$, and $(\phi_{\mathrm{sp}})^{\mathrm{tur}}$.

$$\frac{\partial u}{\partial t} = 0 \qquad \textbf{(2.50)}$$

$$\frac{\partial v}{\partial t} = 0 \qquad \textbf{(2.51)}$$

$$\frac{\partial w}{\partial t} = 0 \qquad \textbf{(2.52)}$$

$$\frac{\partial T}{\partial t} = 0 \qquad \textbf{(2.53)}$$

$$\frac{\partial \mathrm{s}}{\partial t} = 0 \qquad \textbf{(2.54)}$$

$$\frac{\partial k}{\partial t} = P_k + G_k - \varepsilon \qquad \textbf{(2.55)}$$

$$\frac{\partial \varepsilon}{\partial t} = C_1 \frac{\varepsilon}{k} P_k + C_1 (1 - C_3) \frac{\varepsilon}{k} G_k - C_2 \frac{\varepsilon^2}{k} \qquad \textbf{(2.56)}$$

$$\frac{\partial \phi_{\mathrm{sp}}}{\partial t} = 0 \qquad \textbf{(2.57)}$$

First, turbulence production terms P_k,G_k, and turbulence from wind stress are obtained with spatial differences on Soroban meshes. Then these equations are computed by discretizing k and ε with the Euler explicit method. This phase is conducted by the turblence_phase function.

Phase 7:Computations of Diffusion

The diffusion terms in the right sides of equations **(2.20)** ~ **(2.27)** corresponds to the following equations: **(2.58)** ~ **(2.65)**. They are solved with the physical quantities after turbulence computations: u^{tur}, v^{tur}, w^{tur}, T^{tur}, s^{tur}, k^{tur}, $\varepsilon^{\mathrm{tur}}$, and $(\phi_{\mathrm{sp}})^{\mathrm{tur}}$to get those of after diffusion: u^{dif}, v^{dif}, w^{dif}, T^{dif}, s^{dif}, k^{dif}, $\varepsilon^{\mathrm{dif}}$, and $(\phi_{\mathrm{sp}})^{\mathrm{dif}}$.

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial u}{\partial z}\right) + \frac{\tau_{\mathrm{x}}}{\rho} \qquad \textbf{(2.58)}$$

$$\frac{\partial v}{\partial t} = \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial v}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial v}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial v}{\partial z}\right) + \frac{\tau_{\mathrm{y}}}{\rho} \qquad \textbf{(2.59)}$$

$$\frac{\partial w}{\partial t} = \frac{\partial}{\partial x}\left(\nu_{\mathrm{lic\_M}}\frac{\partial w}{\partial x}\right) + \frac{\partial}{\partial y}\left(\nu_{\mathrm{lic\_M}}\frac{\partial w}{\partial y}\right) + \frac{\partial}{\partial z}\left(\nu_{\mathrm{eff}}\frac{\partial w}{\partial z}\right) + \frac{\tau_{\mathrm{z}}}{\rho} \qquad \textbf{(2.60)}$$

$$\frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\frac{\partial T}{\partial z}\right) \qquad \textbf{(2.61)}$$

$$\frac{\partial s}{\partial t} = \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\mathrm{t}}}\frac{\partial s}{\partial z}\right) \qquad \textbf{(2.62)}$$

$$\frac{\partial k}{\partial t} = \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{k}}\frac{\partial k}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{k}}\frac{\partial k}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{k}}\frac{\partial k}{\partial z}\right) \qquad \textbf{(2.63)}$$

$$\frac{\partial \varepsilon}{\partial t} = \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\varepsilon}}\frac{\partial \varepsilon}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\varepsilon}}\frac{\partial \varepsilon}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{eff}}}{\sigma_{\varepsilon}}\frac{\partial \varepsilon}{\partial z}\right) \qquad \textbf{(2.64)}$$

$$\frac{\partial \phi_{\mathrm{sp}}}{\partial t} = w_{\mathrm{settlement}}\frac{\partial \phi_{\mathrm{sp}}}{\partial z} + \frac{\partial}{\partial x}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{\nu_{\mathrm{lic}}}{\sigma_{\phi\mathrm{sp}}}\frac{\partial \phi_{\mathrm{sp}}}{\partial z}\right) \qquad \textbf{(2.65)}$$

These equations are divided into vertical diffusion and horizontal diffusion to be computed one by one. This is because spatial scales are quite different between vertical and horizontal directions. This phase is conducted by the diffusion_phase function. When the vertical diffusion is computed, effects of settlement speed (constant) are also computed for user-defined scalars.

Phase 8: The Other Computations

The heat transportation term, the centrifugal force term, the Coriolis force term, and the source terms of user-defined scalars in the right sides of equations **(2.20)** ~ **(2.27)** corresponds to the following equations: **(2.66)** ~ **(2.69)**. They are solved with the physical quantities after diffusion computations: u^{tur}, v^{tur}, w^{tur}, T^{tur}, s^{tur}, k^{tur}, $\varepsilon^{\mathrm{tur}}$, and $(\phi_{\mathrm{sp}})^{\mathrm{tur}}$ to get those of after computations of these terms: u^{other}, v^{other}, w^{other}, T^{other}, s^{other}, k^{other}, $\varepsilon^{\mathrm{other}}$, and $(\phi_{\mathrm{sp}})^{\mathrm{other}}$.

$$\frac{\partial u}{\partial t} = \frac{1}{R} u^{\mathrm{dif}} v^{\mathrm{dif}} + f_{\mathrm{co}} v^{\mathrm{dif}} \tag{2.66}$$

$$\frac{\partial v}{\partial t} = -\frac{1}{R} u^{\mathrm{dif}} u^{\mathrm{dif}} - f_{\mathrm{co}} u^{\mathrm{dif}} \tag{2.67}$$

$$\frac{\partial T}{\partial t} = S_T \tag{2.68}$$

$$\frac{\partial (\phi_{\mathrm{sp}})}{\partial t} = S_{\phi \mathrm{sp}} \tag{2.69}$$

The equations **(2.66)** and **(2.67)** are computed with the Euler explicit method. The heat transportation term **(2.68)** is integrated by introducing unique minute time intervals. The source terms of user-defined scalars **(2.69)** are computed in the source_term_of_user_defined_scalars function, however its procedures need to be written there in person.

Phase 9: Adjustments of Pressure

The pressure is derived from flow velocities on relocated Soroban meshes. Then flow velocities are accelerated and adjusted with this derived pressure. Besides, before the computations of pressure, apparent sinks (divergences) of aeration plumes are added to the mesh axes which contain aeration devices. Their amount is determined with 1-dimensional vertical plume analyses for each aeration device. The fundamental equations of this phase are the following;

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho} \frac{\partial p^{n+1}}{\partial x} \tag{2.70}$$

$$\frac{\partial v}{\partial t} = -\frac{1}{\rho} \frac{\partial p^{n+1}}{\partial y} \tag{2.71}$$

$$\frac{\partial \mathrm{w}}{\partial t} = -\frac{1}{\rho} \frac{\partial p^{n+1}}{\partial z} - g \tag{2.72}$$

$$\frac{\partial u^{n+1}}{\partial x} + \frac{\partial v^{n+1}}{\partial y} + \frac{\partial w^{n+1}}{\partial z} = \begin{cases} 0 & \text{without aeration} \\ -\dfrac{\mathrm{d}Q_{\mathrm{total}}}{\mathrm{d}z} & \text{with aeration} \end{cases} \tag{2.73}$$

Here, $-\mathrm{d}Q_{\mathrm{total}}/\mathrm{d}z$ is the amount of apparent sinks determined by the analyses of aeration plumes. If a mesh axis contains no aeration device, the right side of **(2.73)** becomes 0 there, so that the conservation law will be satisfied. To determine the pressure, first of all, a Poisson equation is derived

from the incompressibility. Next, it's discretized into a matrix equation on Soroban meshes, so that boundary conditions **(1.20)** and **(1.29)** will be satisfied. Then the pressure is determined by solving the matrix equation with the incomplete LU factorization and the preconditioned BiCGstab method. This phase is conducted by the pressure_phase function.

Configurations of Computational Conditions

Computational conditions are configured by the input data files of following categories:

1. Fundamental configurations
2. Topography
3. Wind field
4. Boundary conditions
5. Initial conditions
6. Time series of user-defined scalars

Besides, you need to edit certain functions of TITech-WARM in person to configure source terms of user-defined scalars (see pp. 103 ~).

Fundamental Configurations (define.inc)

A data file for fundamental configurations specifies the names of input data files and fix various computational conditions: the CFL condition (time interval), Schmidt number, output interval etc. It's prepared as define.inc, but you can use arbitrary name for it. You need to specify its name in a command to start computations like

```
#>mpirun -np 8 titech_warm define.inc
```

If a line in the file begins with the character "#", it's recognized as a comment and never read into the computations. Thus section lists the items which need to be specified in the file.

Items on Time Conditions

MAXIMUM_TIME

Set a floating point number [sec].It's the computational time of the end of computational period. The computations end when its computational time t_n exceeds MAXIMUM_TIME or its computational step number n exceeds MAXIMUM_TIME_STEP.

MAXIMUM_TIME_STEP

Set an integer. It's the computational step number of the end of computational period. The computations end when its computational time t_n exceeds MAXIMUM_TIME or its computational step number n exceeds MAXIMUM_TIME_STEP.

DT_AT_INITIAL_STEP

Set a decimal [sec]. It's the computational time interval on the first computational step of computations.

MAXIMUM_DT

Set a decimal [sec]. It's the upper limit of computational time interval Δt. MAXIMUM_DT has priority over the computational time interval which determined by the CLF condition, even this interval is larger than it.

MAXIMUM_CFL

Set a decimal **less than 0.5**. It's the upper limit of CFL number determined by flow velocities on each computational step. The computational time interval Δt(to the next computational step) is determined so that the CFL number will be less than MAXIMUM_CFL on all mesh points.

MAXIMUM_CFL_FOR_WAVE

Set a decimal (1 or more is also OK). It's the upper limit of CFL number determined by wave velocities on the water surface of each computational step. The computational time valΔt (to the next computational step) is determined so that the CFL number will be less than MAXIMUM_CFL_FOR_WAVE on all mesh points. Its value can exceed 1 because transformations of the water surface are computed with an implicit method.

SAFETY_COEFFICIENT_FOR_EXPLICIT_VISCOSITY_CALCULATION

Set a decimal. It's a safety coefficient$\alpha$of the computational time intervals. It's applied when TITech-WARM computes horizontal diffusions with a time explicit method. This computation is conducted with the multi time step method, and its imaginary time interval $\Delta t_{\mathrm{virtual}}$ is determined to satisfy following condition:

$$\Delta t_{\mathrm{virtual}} < \alpha \min\left(\frac{\Delta x^2}{2\nu_x}, \frac{\Delta y^2}{2\nu_y}\right)$$

Items on Output

OUTPUT_INTERVAL_SWITCH

Set 0 or 1. It decides how to determine the computational time interval to the next output of computational results. If OUTPUT_INTERVAL_SWITCH equals to 1, TITech-WARM outputs interim results at every OUTPUT_TIME_INTERVAL of computational time. If it equals to 0, TITech-WARM outputs interim results on every OUTPUT_STEP_INTERVAL of computational steps.

OUTPUT_TIME_INTERVAL

Set a decimal [sec]. It's the computational time interval of outputs in the case of OUTPUT_INTERVAL_SWITCH = 1.

OUTPUT_TIME_INTERVAL

Set an integer. It's the computational-step-number interval of outputs in the case of OUTPUT_INTERVAL_SWITCH = 0.

RUN_ONLY_FOR_OUTPUT_GRID_POSITION_SWITCH

Set 0 or 1 (usually set 0). If it's equals to 1, the computations finish when the topographic data files are read and the locations of mesh axes are fixed. Then no time evolution will be computed. This is used to get the locations of meshes when you prepared some input data files (of wind field etc.).

OUTPUT_COMPRESSION_SWITCH

Set 0 or 1. If it's equals to 1, each interim result (a TecPlot format file in TecData directory) is compressed into a bzip archive.

OUTPUT_GROUND_LEVEL

Set a decimal [m] which is larger than the maximum water level you suppose. It's the heights of banks (i.e. the part of topographic surface data: ZONE = Bottom Surface which is out of water area) in 3D interim results (Tec_VolumeData_…).

Items on Soroban Meshes

MAX_ITER_NUM_OF_INITIAL_CONDITION

Set an integer. It's the maximum iteration (repetition) number of computations to generate Soroban meshes from the initial conditions.

WEIGHTING_PARAMETER_TO_GATHER_GRID_POINT

Set a dimensionless decimal. It's the weighting coefficient which is multiplied by the vertical gradient of physical quantity to estimate the monitor function. In other words, it's w_z in the equation **(2.49)**and used to determine vertical locations of Soroban meshes. The larger it becomes, the sharper concentration of Soroban mesh points (around vertical fluctuations)gets. If it equals to 0, Soroban mesh points are located with equal vertical intervals between the water and river bed surfaces.

MININUM_VERTICAL_GRID_WIDTH

Set a decimal [m]. It's the lower limit of vertical intervals of Soroban mesh points. Soroban mesh points are located vertically not to be closer than it.

MAX_ITER_NUM_OF_SMOOTHING_MONITORING_FUNCTION_IN_X

Set an integer. When the monitor function is estimated from vertical distributions of a physical quantity, x-directional 3-point moving average is applied to it for the same number of times with this constant (before it's used to determine the vertical locations of Soroban mesh

points). The larger this constant becomes, the smoother the locations of mesh points get in x-direction.

MAX_ITER_NUM_OF_SMOOTHING_MONITORING_FUNCTION_IN_Y

Set an integer. When the monitor function is estimated from vertical distributions of a physical quantity, y-directional 3-point moving average is applied to it for the same number of times with this constant (before it's used to determine the vertical locations of Soroban mesh points). The larger this constant becomes, the smoother the locations of mesh points get in y-direction.

MAX_ITER_NUM_OF_SMOOTHING_MONITORING_FUNCTION_IN_Z

Set an integer. When the monitor function is estimated from the equation **(2.63)**, z-directional 3-point moving average is applied to it for the same number of times with this constant (before it's used to determine the vertical locations of Soroban mesh points). The larger this constant becomes, the more moderate the variations of vertical mesh intervals get.

MININUM_WATER_DEPTH

Set a decimal [m]. It's the lower limit of water level to recognize water areas. When the water level becomes lower than it on a mesh axis, the axis is judged to be in a land area and omitted from computations.

Items on Turbulence Computations

VISCOSITY_MOLECULAR

Set a decimal [m^2/sec]. It's the molecular viscosity of water.

VISCOSITY_MOLECULAR_SALINITY

Set a decimal [m^2/sec]. It's the molecular diffusivity of salinity.

HEAT_CONDUCTIVITY

Set a decimal [m^2/sec]. It's the thermal diffusivity of water: α. The relationship between the thermal (heat) conductivity k [J/s m K] is

$$\alpha = \frac{k}{\rho c_p}$$

Here, ρ [kg/m^3] is the density of water, and c_p [J/kg K] is the specific heat capacity.

KE_RICHARDSON_RAW_COEFFICIENT

Set a decimal. It's the coefficient of Richardson's 4/3 law (i.e. the coefficientν_0 in the equation **(1.13)**).

KE_SCHMIDT_NUMBER_FOR_HORIZONTAL

Set a decimal. It's the turbulent Schmidt number to determine the horizontal ty$\nu_{\mathrm{lic_M}}$ (i.e. the coefficient S_c in the equation **(1.14)**).

KE_CMYU

Set a decimal. It's the coefficient C_μ in the *k*-*ε* turbulent model.

KE_C1

Set a decimal. It's the coefficient C_1 in the *k*-*ε* turbulent model.

KE_C2

Set a decimal. It's the coefficient C_2 in the *k*-*ε* turbulent model.

KE_C3

Set a decimal. It's the coefficient C_3 in the *k*-*ε* turbulent model.

KE_SIGMA_K

Set a decimal. It's the turbulent Schmidt number σ_k for the turbulent kinetic energy k in the *k*-ε turbulent model.

KE_SIGMA_E

Set a decimal. It's the turbulent Prandtl number σ_ε for the turbulent dispersion rate ε in the *k*-ε turbulent model.

KE_SIGMA_T

Set a decimal. It's the turbulent Prandtl number σ_s for the (density of) salinity in the *k*-ε turbulent model.

KE_BED_BOUNDARY_MODEL

Set an integer. It's used to choose a boundary condition model of turbulence in the *k*-ε turbulent model. If it equals to 0, constants are given as the boundary value of *k* and ε based on the wall boundary condition. In contrast, if it equals to 1, the supply-flux model of turbulence (based on the analyses by Ishikawa and Moribayashi et al.) is given as the boundary condition (see pp. 20~ to refer the details).

Items on Water Surface Variations

MAX_ITER_NUM_OF_BICGSTAB_SOLUTION_IN_WATER_SURFACE_PROPAGATION

Set an integer. It's the maximum iteration (repetition) number of ILU-BiCGstab matrix solutions to compute water surface variations implicitly.

EPSIRON_OF_BICGSTAB_SOLUTION_IN_WATER_SURFACE_PROPAGATION

Set a decimal. It's the threshold of ILU-BiCGstab matrix solutions' convergence to compute water surface variations implicitly.

Items on Pressure Adjustments

MAX_ITER_NUM_OF_BICGSTAB_SOLUTION_IN_PRESSURE_PHASE

Set an integer. It's the maximum iteration (repetition) number of ILU-BiCGstab matrix solutions to estimate the pressure.

EPSIRON_OF_BICGSTAB_SOLUTION_IN_PRESSURE_PHASE

Set a decimal. It's the threshold of ILU-BiCGstab matrix solutions' convergence to estimate the pressure.

Items on Various Physical Quantities

GRAVITY_ACCE

Set a decimal [m/s^2]. It's the gravity acceleration.

FRICTION_COEFFICIENT

Set a decimal. It's the friction coefficient to compute friction against river bed or riverbanks (i.e. the coefficient f_b in the equations **(1.30)** ~ **(1.32)** and **(1.34)** ~ **(1.36)**).

SALINITY_ENABLE_SWITCH

Set an integer. It determines whether TITech-WARM computes salinity or not. If it equals to 0, the computations of salinity will be never conducted. Thus set 1 to compute it.

WATER_TEMPERATURE_ENABLE_SWITCH

Set an integer. It determines whether TITech-WARM computes water temperature or not. If it equals to 0, the computations of water temperature will be never conducted. Thus set 1 to compute it.

Items on Input Topographic Data

NAME_OF_LIST_FILE_OF_TOPOGRAPHY_CROSS_SECTION

Specify a filename. It's a topographic data file which determines the height of river bed and mesh conditions.

LATITUDE

Set a decimal [˚]. It's the latitude which is used to determine the Coriolis parameter. A positive value means the north latitude, and negative one means the south latitude.

Items on Wind Field

NAME_OF_LIST_FILE_OF_2D_WIND_FIELD

Specify a filename. It contains the time series of wind velocities on each mesh axis to represent 2-dimensional distributions of wind. If you want use a data file for it, **do not** specify any filename for the next item: FILE_OF_UNIFORM_WIND_FIELD.

FILE_OF_UNIFORM_WIND_FIELD

Specify a filename. It contains the time series of wind velocity to represent uniform wind distributions on every mesh axis. If you want use a data file for it, **do not** specify any filename for the previous item: NAME_OF_LIST_FILE_OF_2D_WIND_FIELD.

Items on Heat Transportation

HEAT_BUDGET_ENABLE_SW

Set an integer. It determines whether TITech-WARM computes heat inlet/outlet with the heat_budget_water_temperature function or not. If it equals to 0, its computations will not be conducted. Thus set 1 to compute it.

HEAT_BUDGET_DT

Set a decimal [sec]. It's computational time interval of time integration to compute heat transportation with the heat_budget_water_temperature function. If HEAT_BUDGET_DT exceeds the time interval used in other advection computations, the time interval of advection computations is used for the time integration instead of it.

HEAT_BUDGET_ALBEDO

Set a decimal. It's the reflection ratio (albedo) of shortwave solar radiation on the water surface.

HEAT_BUDGET_ABSORPTION_RATE_AT_WATER_SURFACE

Set a decimal. It's the ratio of light which is absorbed by the water surface to the shortwave solar radiation which was not reflected by the water surface.

HEAT_BUDGET_BUDGET_ATTENUATION_LENGTH_OF_SHORT_WAVE

Set a decimal [m]. It's the attenuation length; the strength of light which penetrated into the water becomes 1/e while it travelling an attenuation length of water.

HEAT_BUDGET_BUDGET_BUDGET_EMPIRICAL_COEFFICIENT_OF_LONG_WAVE

Set a decimal (usually set 1). It adjusts the amount of heat transportation from the water surface by the fluxes of longwave radiation, latent heat, and sensible heat. It corresponds to the coefficient C_l in the equation **(1.46)** (see pp. 26~).

FILE_OF_AIR_TEMPERATURE

Specify a filename. It's the time-series data file of air temperature [˚C] for the computations of heat transportation. Heat transportation will not be computed without it.

FILE_OF_AIR_CLOUDINESS

Specify a filename. It's the time-series data file of cloudiness (0 ~ 10) for the computations of heat transportation. Heat transportation will not be computed without it.

FILE_OF_AIR_HUMIDITY

Specify a filename. It's the time-series data file of air humidity [%] for the computations of heat transportation. Heat transportation will not be computed without it.

FILE_OF_AIR_SOLOR_RADIATION

Specify a filename. It's the time-series data file of the amount of shortwave solar radiation [MJ/m^2 day] for the computations of heat transportation. Heat transportation will not be computed without it.

Items on Aeration Model

The following items determine various parameters of the aeration model and aeration devices (locations, amount of sending air etc.). The parameters of more than one aeration devices can be written in the same format. Up to 25 aeration devices are available in TITech-WARM.

AERATION_ATM_WATER_HEAD

Set a decimal [m]. It's the hydraulic head of the atmospheric pressure.

AERATION_SURFACE_TENSION_COFF

Set a decimal. It's the surface tension coefficient of water.

AERATION_DZ_FOR_PLUME_MODEL

Set a decimal [m]. It's the mesh interval used in 1-dimensional vertical analyses of aeration plumes. It should be smaller than the intervals of Soroban meshes, so set a smaller value than MININUM_VERTICAL_GRID_WIDTH.

AERATION_ENTRAINMENT_COFF_ALPHA

Set a decimal. It's the entrainment coefficient between surrounding water and aeration plumes: α (see pp. 30~).

AERATION_ENTRAINMENT_COFF_BETA

Set a decimal. It's the entrainment coefficient from surrounding water and the outer plume to the inner plume: β (see pp. 30~).

AERATION_ENTRAINMENT_COFF_GAMMA

Set a decimal. It's the entrainment coefficient from the inner plume to the outer plume: γ (see pp. 30~).

AERATION_EMPIRICAL_COFF_FOR_WATER_SURFACE

Set a decimal (1.0 is a standard). It affects the increase of entrainment when a plume begins to sink from the water surface. It corresponds to the coefficientC_{e} in the equation **(1.77)** (see pp. 32~)

AERATION_BUBBLE_PLUME_SLIP_VELOCITY

Set a decimal [m/s]. It's the slip velocity of aeration (bubble) cores.

AERATION_MAX_ITER_NUM

Set an integer. It's the maximum iteration (repetition) number of computations in the double plume model.

AERATION_EPSIRON_OF_ITER

Set a decimal. It's the threshold to finish computations of double plume model.

AERATION_MAX_ITER_NUM_OF_NEWTON_FOR_INITIAL_PLUME

Set an integer. It's the maximum iteration (repetition) number of Newton-Raphson method to determine initial values of inner plumes.

AERATION_EPSIRON_OF_ITER

Set a decimal. It's the threshold of Newton-Raphson method to determine initial values of inner plumes.

AERATION_DEVICE_1_SWITCH

Set an integer. It determines whether TITech-WARM computes aeration device #1 or not. If it equals to 0, its computations will not be conducted. Thus set 1 to compute it.

AERATION_DEVICE_1_X

Set a decimal [m]. It's the x-coordinate of aeration device #1 in the computational space.

AERATION_DEVICE_1_Y

Set a decimal [m]. It's the y-coordinate of aeration device #1 in the computational space.

AERATION_DEVICE_1_Z

Set a decimal [m]. It's the z-coordinate of aeration device #1 in the computational space.

AERATION_DEVICE_1_BUBBLE_VOLUME_FLUX

Set a decimal [m^3/sec]. It's the amount of air which aeration device #1 sends out.

If you want compute more than one aeration device, set AERATION_DEVICE_??_SWITCH, AERATION_DEVICE_??_X,AERATION_DEVICE_??_Y, AERATION_DEVICE_??_Z, and AERATION_DEVICE_??_BUBBLE_VOLUME_FLUXin the same way for each device. Here, "??" represents the index of aeration devices (2 ~ 25).

Items on User-Defined scalars

Computations of a user-defined scalar are activated by specifying its information here. Up to 25 user-defined scalars are available in TITech-WARM.

NAME_OF_USER_DEFINED_SCALAR_1

Set a character string. It's the name of user-defined scalar #1.The computations of its time evolutions are activated by specifying this name here.

KE_SIGMA_USER_DEFINED_SCALAR_1

Set a decimal. It's the turbulent Schmidt number of user-defined scalar #1: $\sigma_{\phi 1}$. If it equals to a negative number, no diffusion computation will be conducted.

VISCOSITY_MOLECULAR_USER_DEFINED_SCALAR_1

Set a decimal [m^2/sec]. It's the diffusivity of user-defined scalar #1 for the molecular diffusion (non-turbulent diffusion).

SETTLING_SPEED_USER_DEFINED_SCALAR_1

Set a decimal [m/sec]. It's the settlement speed of user-defined scalar #1 (positive value means downward). If no value is set, it becomes 0.

If you want compute more than one user-defined scalar, set NAME_OF_USER_DEFINED_SCALAR_??, KE_SIGMA_USER_DEFINED_SCALAR_??, VISCOSITY_MOLECULAR_USER_DEFINED_SCALAR_??, and SETTLING_SPEED_USER_DEFINED_SCALAR_?? in the same way for each physical quantity. Here, "??" represents the index of user-defined scalar (2 ~ 25).

Items on User-Input Data (1D Time-Series Data)

NAME_OF_USER_INPUT_DATA_1

Set a character string. It's the name of 1-dimensional time-series data #1 which you can arbitrarily add to the computations. The values in the file which is specified as FILE_OF_USER_INPUT_DATA_1 will be read by specifying this name here.

FILE_OF_USER_INPUT_DATA_1

Specify a filename. It's the data file of 1-dimensional time-series data #1.

NAME_OF_USER_INPUT_DATA_2

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_2

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_3

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_3

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_4

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_4

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_5

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_5

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_6

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_6

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_7

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_7

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_8

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_8

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_9

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_9

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

NAME_OF_USER_INPUT_DATA_10

Set it in the same way with NAME_OF_USER_INPUT_DATA_1, if needed.

FILE_OF_USER_INPUT_DATA_10

Set it in the same way with FILE_OF_USER_INPUT_DATA_1, if needed.

Items on User-Input Data (2D Time-Series Data)

NAME_OF_USER_INPUT_2D_DATA_1

Set a character string. It's the name of 2-dimensional time-series data #1 which you can arbitrarily add to the computations. The values in the files which are specified as NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1 will be read by specifying this name here.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1

Specify a filename. It's a list which contains the filenames of 2-dimensional time-series data #1.

NAME_OF_USER_INPUT_2D_DATA_2

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_2

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_3

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_3

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_4

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_4

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_5

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_5

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_6

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_6

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_7

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_7

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_8

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_8

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_9

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_9

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_USER_INPUT_2D_DATA_10

Set it in the same way with NAME_OF_USER_INPUT_2D_DATA_1, if needed.

NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_10

Set it in the same way with NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA_1, if needed.

Items on Initial Conditions

FILE_OF_INITIAL_WATER_LEVEL

Specify a filename. It contains the initial distributions of water level. If this item is commented out (with the character "#"), the values determined by the water_level_initializition function are set as the initial condition.

NAME_OF_LIST_FILE_OF_INITIAL_VELOCITY_X

Specify a filename. It's a list of filenames which contain the initial vertical distributions of main flow velocity (x-directional velocity component u in the computational space). If this item is commented out, the values determined by the set_initial_condition_of_ValueOnVolume function are set as the initial condition.

NAME_OF_LIST_FILE_OF_INITIAL_VELOCITY_Y

Specify a filename. It's a list of filenames which contain the initial vertical distributions of transverse velocity (y-directional velocity component v in the computational space). If this item is commented out, the values determined by the set_initial_condition_of_ValueOnVolume function are set as the initial condition.

NAME_OF_LIST_FILE_OF_INITIAL_VELOCITY_Z

Specify a filename. It's a list of filenames which contain the initial vertical distributions of vertical velocity (z-directional velocity component w in the computational space). If this item is commented out, the values determined by the set_initial_condition_of_ValueOnVolume function are set as the initial condition.

NAME_OF_LIST_FILE_OF_INITIAL_SALINITY

Specify a filename. It's a list of filenames which contain the initial vertical distributions of salinity s. If this item is commented out, the values determined by the set_initial_condition_of_ValueOnVolume function are set as the initial condition.

NAME_OF_LIST_FILE_OF_INITIAL_WATER_TEMPERATURE

Specify a filename. It's a list of filenames which contain the initial vertical distributions of water temperature T. If this item is commented out, the values determined by the set_initial_condition_of_ValueOnVolume function are set as the initial condition.

NAME_OF_LIST_FILE_OF_INITIAL_USER_DEFINED_SCALAR_1

Specify a filename. It's a list of filenames which contain the initial vertical distributions of user-defined scalar #1. If this item is commented out, the values determined by the set_initial_condition_of_ValueOnVolume function are set as the initial condition.

You can set initial conditions for other user-defined scalars by specifying data files as NAME_OF_LIST_FILE_OF_INITIAL_USER_DEFINED_SCALAR_?? in the same way. Here, "??" represents the index of user-defined scalar (2 ~ 25).

Items on Boundary Conditions

The –-side boundary: $X_{\min}$ and +x-side boundary $X_{\max}$ are configured in the same way. First, to support branching rivers, the domains which have different boundary conditions (boundary domains) need to be specified on each boundary with their extents of y-coordinates. Then boundary conditions of various physical quantities are determined for each boundary domains. The inlets or outlets of each boundary domains can be restricted into certain areas with a data file of gate conditions.

Configuration of Conditions on $X_{\max}$ Boundary

(1) Configuration of Boundary Domains

BOUNDARY_POSITION_YMIN_AT_XMAX_1

Set a decimal [m]. It's the minimum y-coordinate of the boundary domain #1 (of $X_{\max}$ boundary). The extent between this value and the next item BOUNDARY_POSITION_YMAX_AT_XMAX_1 becomes the boundary domain #1.

BOUNDARY_POSITION_YMAX_AT_XMAX_1

Set a decimal [m]. It's the maximum y-coordinate of the boundary domain #1 (of $X_{\max}$ boundary). The extent between this value and the next item BOUNDARY_POSITION_YMIN_AT_XMAX_1 becomes the boundary domain #1.

Boundary domains #2 ~ #10 are also available on the $X_{\max}$ boundary in the same way.

(2) Gate Conditions of Each Boundary Domain

FILE_OF_BOUNDARY_CONDITION_FOR_GATE_AT_XMAX_1

Specify a filename. It contains time-series extents of passage domains (rectangular areas where inlets or outlets are allowed) in the boundary domain #1 of $X_{\max}$ boundary. If no filename is specified, boundary conditions are applied to this boundary domain equally.

Gate conditions for boundary domains #2 ~ #10 are also available in the same way.

(3) Boundary Conditions of Each Boundary Domain

FILE_OF_BOUNDARY_CONDITION_FOR_WATER_LEVEL_AT_XMAX_1

Specify a filename. It contains time-series data of water level in the boundary domain #1 of $X_{\max}$ boundary. Given water level is equally applied to the boundary domain as a boundary condition. If this item is commented out, a free boundary condition is applied instead of it.

FILE_OF_BOUNDARY_CONDITION_FOR_Q_AT_XMAX_1

Specify a filename. It contains time-series data of incoming flow rate in the boundary domain #1 of $X_{\max}$ boundary. Given flow rate divided by total cross sectional area of the boundary domain (or the passage domains) is equally applied to the boundary domain as a boundary condition of flow velocity. If this item is commented out, a free boundary condition is applied instead of it.

FILE_OF_BOUNDARY_CONDITION_FOR_SALINITY_AT_XMAX_1

Specify a filename. It contains time-series data of salinity in the boundary domain #1 of $X_{\max}$ boundary. Given salinity is equally applied to the boundary domain as a boundary condition. If this item is commented out, a free boundary condition is applied instead of it.

FILE_OF_BOUNDARY_CONDITION_FOR_WATER_TEMPERATURE_AT_XMAX_1

Specify a filename. It contains time-series data of water temperature in the boundary domain #1 of $X_{\max}$ boundary. Given water temperature is equally applied to the boundary domain as a boundary condition. If this item is commented out, a free boundary condition is applied instead of it.

FILE_OF_BOUNDARY_CONDITION_FOR_USER_DEFINED_SCALAR_??_AT_XMAX_1

Specify a filename. It contains values of user-defined scalar #?? (?? is 1 ~ 25) in the boundary domain #1 of $X_{\max}$ boundary. If this item is commented out, a free boundary condition is applied instead of it.

Boundary conditions for boundary domains #2 ~ #10 are also available in the same way.

Configuration of Conditions on $X_{\min}$ Boundary

They are configured in the same way with the $X_{\min}$ boundary.

Input Data Files of Topographic Conditions

This section explains the input data files which specify height of river bed, the number and locations of Soroban mesh points, the extent of computational domain, and so on. They are composed of some "transverse topographic files" and their list: "cross-sectional list file". A transverse topographic file contains heights of river bed etc. on the x-coordinate of certain mesh plain.

Cross-Sectional List File

It's a list of transverse topographic files, and its name needs to be specified by NAME_OF_LIST_FILE_OF_TOPOGRAPHY_CROSS_SECTION. **Fig. 24** shows an example of cross-sectional list file. A line which begins with the character "#" is a comment. The meanings of other lines are the followings;

Line 2: Xmin

It's the x-coordinate [m] of $X_{\min}$ boundary (in the computational space).

Line 4: Xmax

It's the x-coordinate [m] of $X_{\max}$ boundary (in the computational space).

Line 6: NUMBER_OF_TOPOGRAPHY_CROSS_SECTION

It's the number of transverse topographic files, and needs to be same with the value of NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE. Soroban mesh planes are located at the x-coordinates which were specified in the transverse topographic files, thus the value of NUMBER_OF_TOPOGRAPHY_CROSS_SECTION also corresponds to the number of mesh planes.

Line 8 ~: NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE

The names of transverse topographic files are specified in order from the –x-side to the +x-side. Soroban mesh planes are located at the x-coordinates which were specified in the transverse topographic files, and a transverse topographic file contains heights of river bed etc. on certain mesh plain.

```
#Boundary Position At Xminimum Along Main Stream Gut
Xmin=-100.
#Boundary Position At Xmaximum Along Main Stream Gut
Xmax=2000.
#Number of Data File of Cross Section
NUMBER_OF_TOPOGRAPHY_CROSS_SECTION=100
#Name of Cross Section Data File
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_1.geo
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_2.geo
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_3.geo
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_4p.geo
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_5.geo
...
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_river_98.geo
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_river_99.geo
NAME_OF_TOPOGRAPHY_CROSS_SECTION_FILE=cross_section_river_100.geo
```

Fig. 24 An Example of Cross-Sectional List File

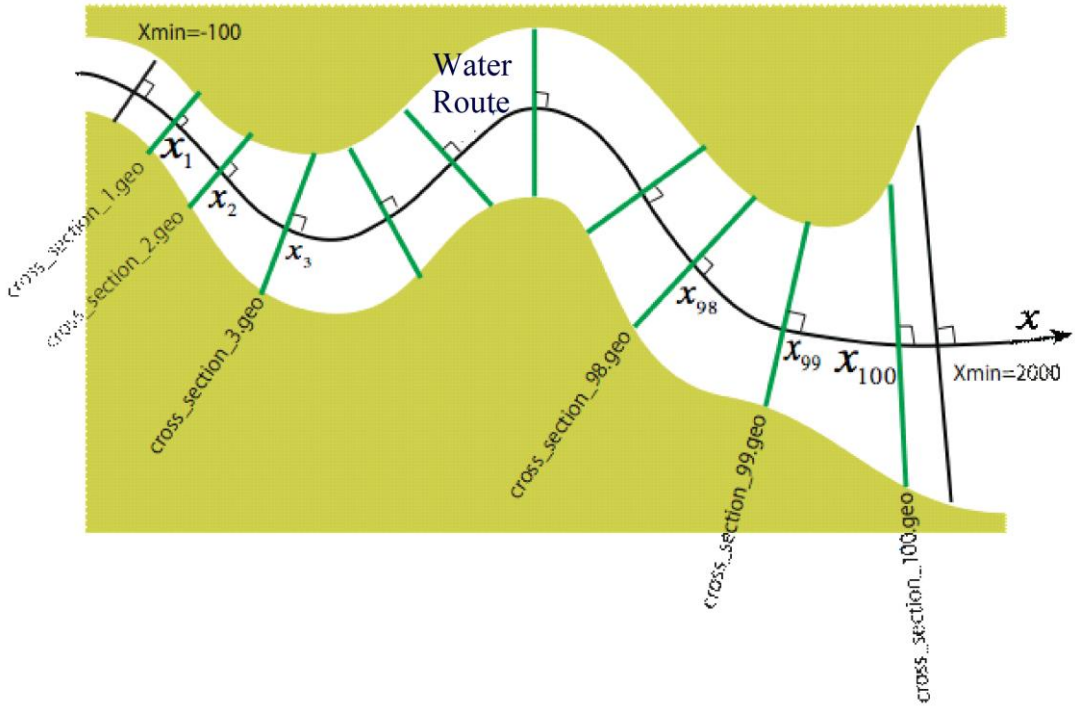

Fig. 25 Relationship between the cross-sectional list file and the computational domain

```
CROSS_SECTION_NUMBER
26
POSITION_ALONG_MAIN_STREAM_GUT[m]
-1.200000e+04
POSITION_OF_MAIN_STREAM_GUT_(X_Y)_IN_REAL_SPACE[m]
4.432400e+04 -1.199000e+03
COUNTERCLOCK-
WARD_ANGLE_FROM_Y_AXIS_IN_REALSPACE_TO_Y_AXIS_IN_COMPTSPACE[radian]
8.024614e-01
CURVATURE(STREAM_TOWARD_+X_CURVING_LEFTWARD_IS_POSITIVE)[1/m]
2.644414e-04
DISTANCE_FROM_MAIN_STREAM_GUT_TO_YMIN_SIDEWALL[m]
-1.425400e+02
DISTANCE_FROM_MAIN_STREAM_GUT_TO_YMAX_SIDEWALL[m]
1.495400e+02
NUMBER_OF_SOROBAN_GRID_AXES
12
DISTANCE_FROM_MAIN_STREAM_GUT[m] MAX_NUMBER_OF_GRID_POINTS RIV-
ER_BED_LEVEL[m]
-1.303700e+02 26 -1.651400e+00
-1.060300e+02 32 -2.481400e+00
-8.169000e+01 30 -2.394200e+00
-5.735000e+01 24 -1.444000e+00
-3.301000e+01 38 -3.355000e+00
-8.670001e+00 38 -3.545000e+00
1.567000e+01 40 -3.740200e+00
4.001000e+01 38 -3.426200e+00
6.435000e+01 40 -3.687400e+00
8.869000e+01 40 -3.745000e+00
1.130300e+02 36 -3.189800e+00
1.373700e+02 34 -2.813400e+00
```

Fig. 26 An Example of Cross-River Topographic File (cross_section_26.geo)

Cross-River Topographic File

Fig. 26 shows an example of transverse topographic file (data on a cross section). The meaning of each item is the following;

Note: Do not edit any character string; just edit numbers.

Line 1 ~ 2

```
CROSS_SECTION_NUMBER
26
```

The order of this file in the cross-sectional list file is specified. This example shows that its cross section is the 26th from the –x-side.

Line 3 ~ 4

```
POSITION_ALONG_MAIN_STREAM_GUT[m]
-1.200000e+04
```

Its cross section's x-coordinate x_i is specified (main-stream direction in the computational space, see **Fig. 25**).

Line 5 ~ 6

```
POSITION_OF_MAIN_STREAM_GUT_(X_Y)_IN_REAL_SPACE[m]
4.432400e+04 -1.199000e+03
```

Xc Yc

Coordinates of a point (X_c, Y_c) where its cross section intersects x-axis of computational space (i.e. water route, main-stream axis) are specified. The X_c and Y_c in the real space need to be specified (not in the computational space, see **Fig. 27**).

Line 7 ~ 8

```
COUNTERCLOCKWARD_ANGLE_FROM_Y_AXIS_IN_REALSPACE_TO_Y_AXIS_IN_COMPTSPACE[radian]
8.024614e-01
```

An angle θ_i [rad] between its cross section (i.e. the y-axis in computational space) and the y-axis in the real space is specified. Positive value means counterclockwise from the y-axis in the real space (see **Fig. 28**).

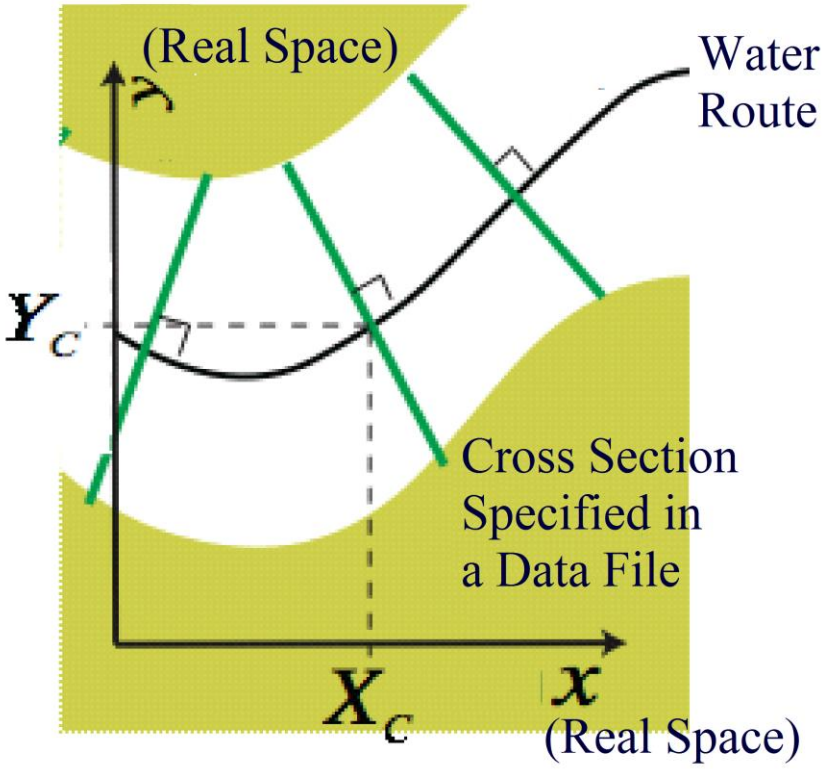

Fig. 27 Definitions of *X*c and *Y*c

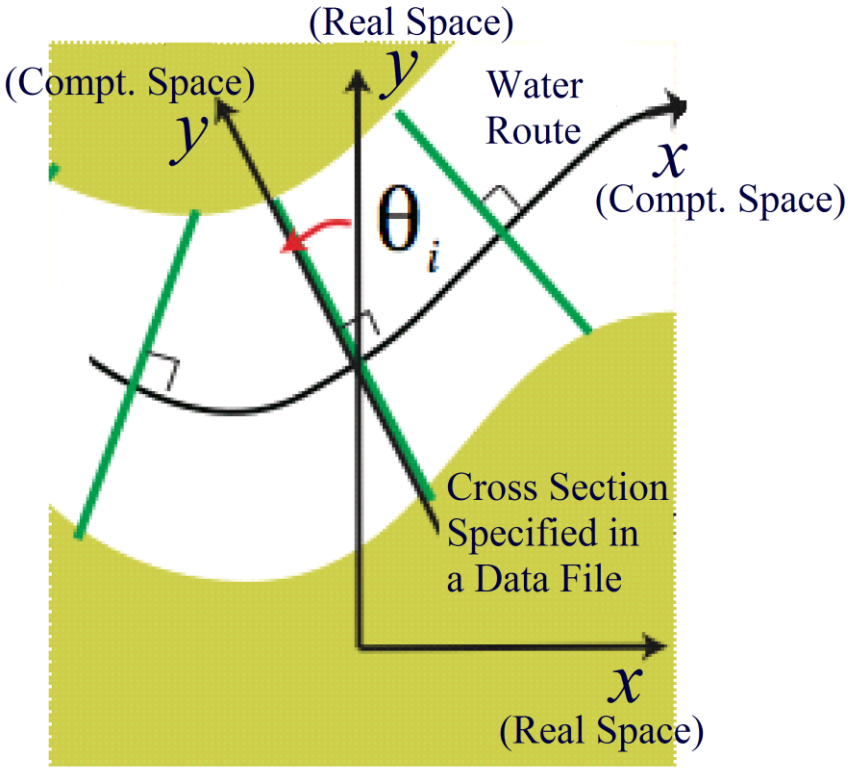

Fig. 28 Definition of θ

Line 9 ~ 10

```
CURVATURE(STREAM_TOWARD_+X_CURVING_LEFTWARD_IS_POSITIVE)[1/m]
2.644414e-04
```

Curvature around its cross section $1/R$ is specified. Positive value means the main stream (+x direction) is turning to the +y-side (see **Fig. 29**).

Line 11 ~ 14

```
DISTANCE_FROM_MAIN_STREAM_GUT_TO_YMIN_SIDEWALL[m]
-1.425400e+02
DISTANCE_FROM_MAIN_STREAM_GUT_TO_YMAX_SIDEWALL[m]
1.495400e+02
```

The y-coordinates (in the computational space) of riverbanks' edges (i.e. values of $Y_{\min}$ and $Y_{\max}$) on the cross section are specified. They correspond to the y-directional distance from the point where the cross section intersects the water route (i.e. the origin of y-axis: $y = 0$, see **Fig. 30**).

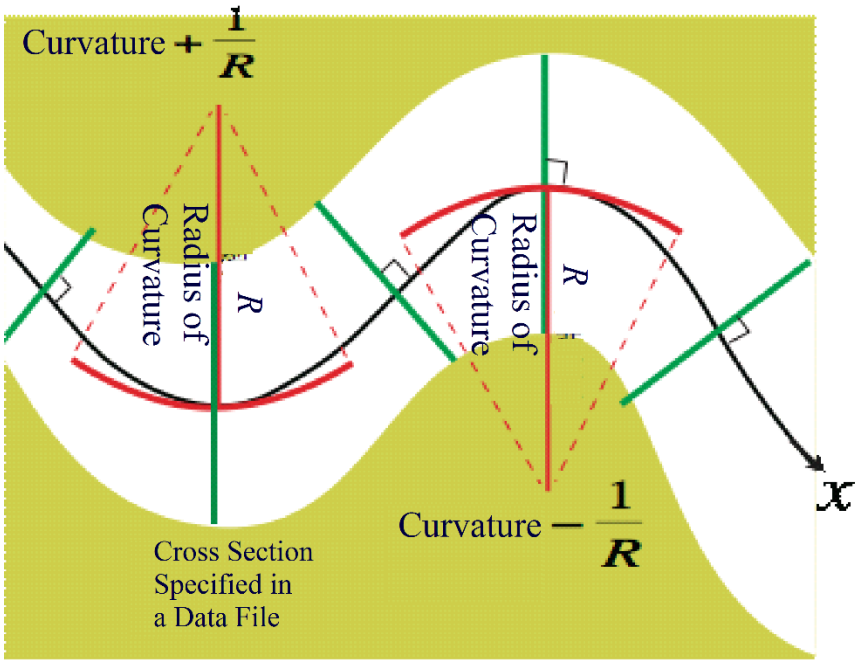

Fig. 29 Definition of Curvature 1/*R*

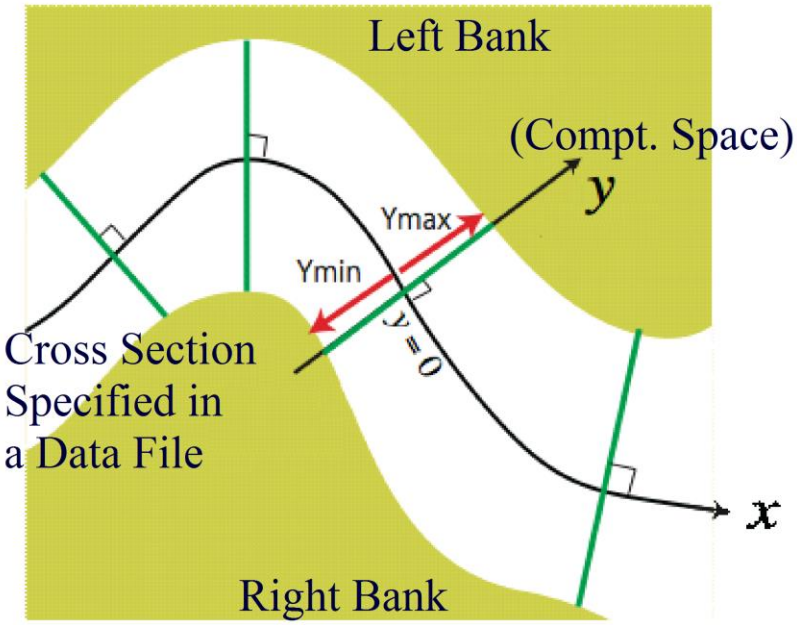

Fig. 30 Definitions of *Y*min, *Y*max

Line 15 ~ 16

```
NUMBER_OF_SOROBAN_GRID_AXES
12
```

The number of mesh axes $(N_y)_i$ in the cross section is specified (see **Fig. 31**).

Line 17 ~

```
DISTANCE_FROM_MAIN_STREAM_GUT[m] MAX_NUMBER_OF_GRID_POINTS RIVER_BED_LEVEL[m]
-1.303700e+02 26 -1.651400e+00
-1.060300e+02 32 -2.481400e+00
-8.169000e+01 30 -2.394200e+00
-5.735000e+01 24 -1.444000e+00
-3.301000e+01 38 -3.355000e+00
-8.670001e+00 38 -3.545000e+00
1.567000e+01 40 -3.740200e+00
4.001000e+01 38 -3.426200e+00
6.435000e+01 40 -3.687400e+00
8.869000e+01 40 -3.745000e+00
1.130300e+02 36 -3.189800e+00
1.373700e+02 34 -2.813400e+00
```

A line contains data of each mesh axis in the cross section: the y-coordinate in the computational space y_{ij}, the number of mesh points $(N_y)_i$, and the height of river bed b_{ij}. Then the lines are listed in order from the –y-side. The number of listed lines (line 18~) needs to be same with the number specified in the line 16 (see **Fig. 31**).

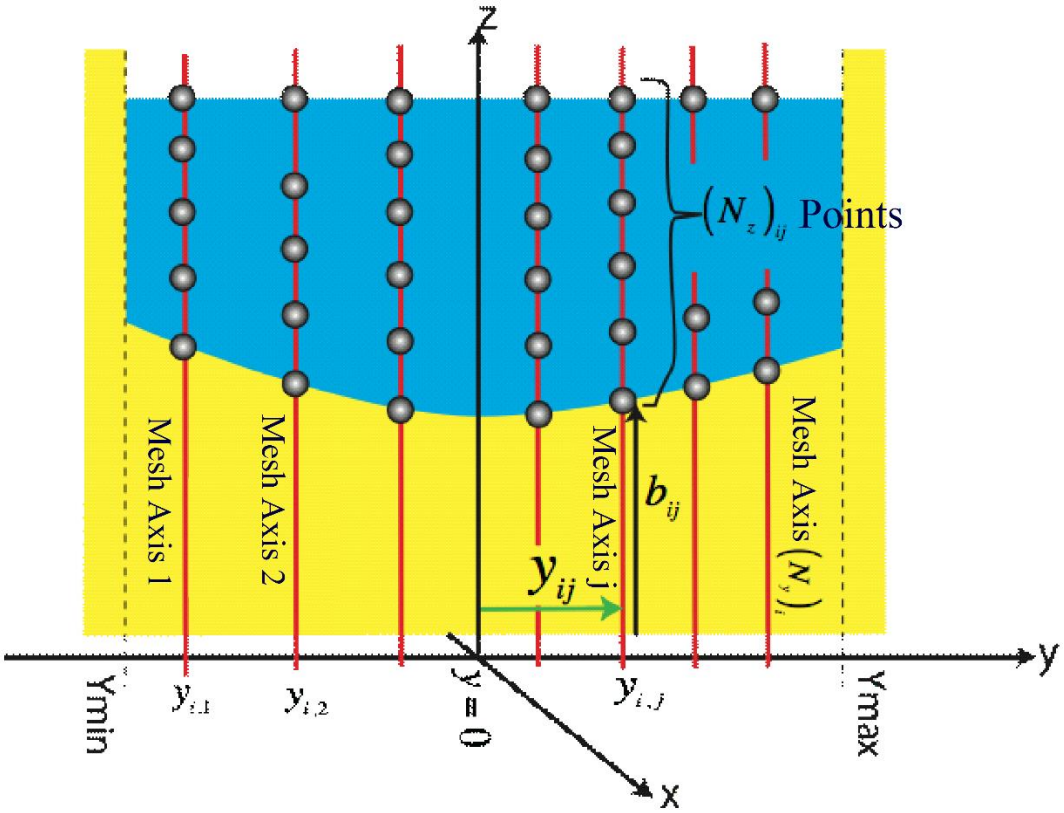

Fig. 31 Definitions of mesh axes' numbers

Input Data Files of Blowing Wind

The effect of wind field is added to the computations by specifying a filename as the value of NAME_OF_LIST_FILE_OF_2D_WIND_FIELD or FILE_OF_UNIFORM_WIND_FIELD. Different kinds of wind will be added to the computations by the configurations of these items. In any cases, the x-directional and y-directional components (in the real space) of wind velocity at an altitude above 10 m $(U_{10})_x$ and $(U_{10})_y$ are required as input data (see **Fig. 32**).

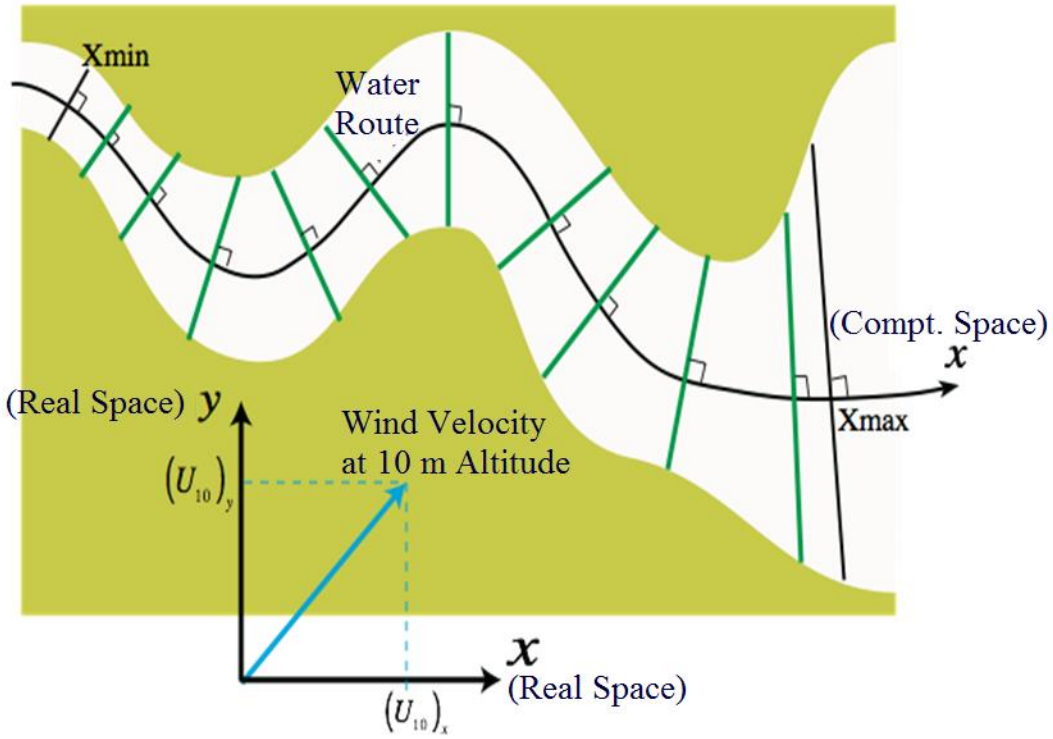

Fig. 32 Components of wind field in the real space coordinates

Uniform Wind (throughout the Compt. Domain)

The wind field is assumed to be spatially equal by specifying a data file for FILE_OF_UNIFORM_WIND_FIELD and commenting out NAME_OF_LIST_FILE_OF_2D_WIND_FIELD with the character #. A data file of the following format is needed to be specified for FILE_OF_UNIFORM_WIND_FIELD. Each line in the data file contains the computational time and the components of wind velocity (of 10 m altitude): $(U_{10})_x$ and $(U_{10})_y$. The wind velocity of certain computational time t_n is obtained with time linear interpolations of given wind data. If t_n exceeds the computational time of last wind data t_N, afterward the wind velocity of t_n is assumed to be same with that of t_N.

```
0 -1.18463e-08 -2.2
3600 -1.18463e-08 -2.2
7200 -0.91844 -2.21731
10800 -0.707107 -0.707107
...
[Compt. time(sec)] [x-component of wind vel.(m/s)] [y-component(m/s)]
```

Fig. 33 Time-series 1D data of uniform wind

2D Wind Distribution (of Non-Uniform Wind)

It is assumed that the wind field is not spatially equal and has 2-dimensional distributions by specifying a data file for NAME_OF_LIST_FILE_OF_2D_WIND_FIELD and commenting out FILE_OF_UNIFORM_WIND_FIELD with the character #. In this case, wind data is composed of some "wind distribution data files" and their list: "wind distribution data list". A wind distribution data file contains 2-dimensional distribution of wind velocity at certain computational time.

The wind distribution data list needs to be specified for NAME_OF_LIST_FILE_OF_2D_WIND_FIELD in the fundamental configuration file "*.inc". As shown in **Fig. 34**, each line of wind distribution data list contains computational time and a file-name of wind distribution data file (Wind_on_Grid_1.geo, Wind_on_Grid_2.geo, ...). As shown in **Fig. 35**, a wind distribution data file contains wind velocities of all mesh axes at certain computational time.

No Wind

The computations will be conducted without any wind stress by commenting out both of NAME_OF_LIST_FILE_OF_2D_WIND_FIELD and FILE_OF_UNIFORM_WIND_FIELD.

```
0 Wind_on_Grid_0.geo
10800 Wind_on_Grid_1.geo
21600 Wind_on_Grid_2.geo
32400 Wind_on_Grid_3.geo
43200 Wind_on_Grid_4.geo
54000 Wind_on_Grid_5.geo
64800 Wind_on_Grid_6.geo
75600 Wind_on_Grid_7.geo
…
[Compt. time(sec)] [Filename of a wind distribution data file]
```

Fig. 34 Example of wind distribution data list

```
1 1 6.466936e-02 5.804900e-01
1 2 7.192096e-02 1.391544e+00
1 3 6.803382e-02 1.281178e+00
1 4 6.803382e-02 1.281178e+00
1 5 6.803382e-02 1.281178e+00
2 1 1.957665e-01 1.185633e+00
2 2 1.957665e-01 1.185633e+00
2 3 1.957665e-01 1.185633e+00
2 4 3.859721e-02 1.156766e+00
2 5 3.859721e-02 1.156766e+00
2 6 3.859721e-02 1.156766e+00
3 1 9.430469e-02 6.400581e-01
…
[Plane index i] [Axis index j] [x-component of wind vel.(m/s)] [y-component(m/s)]
```

Fig. 35 Example of wind distribution data file

Input Data Files of Boundary Conditions

In default, boundary conditions are determined by the update_boundary_value_for_... function. However, you can also specify some of water level, salinity, flow rate (flow velocity), and user-defined scalars on the $X_{\min}$ and $X_{\max}$ boundaries as time-series data. It is realized by specifying filenames as some of following items in the fundamental configuration file;

FILE_OF_BOUNDARY_CONDITION_FOR_WATER_LEVEL_AT_XMAX _??

FILE_OF_BOUNDARY_CONDITION_FOR_WATER_LEVEL_AT_XMIN_??

FILE_OF_BOUNDARY_CONDITION_FOR_Q_AT_XMAX_??

FILE_OF_BOUNDARY_CONDITION_FOR_Q_AT_XMIN_??

FILE_OF_BOUNDARY_CONDITION_FOR_SALINITY_AT_XMAX_??

FILE_OF_BOUNDARY_CONDITION_FOR_SALINITY_AT_XMIN_??

FILE_OF_BOUNDARY_CONDITION_FOR_USER_DEFINED_SCALAR_1_AT_XMAX_??

FILE_OF_BOUNDARY_CONDITION_FOR_USER_DEFINED_SCALAR_1_AT_XMIN_??

...

("??" is an index of boundary domain)

Values in specified data files are equally applied to the cross section (or the passage domains) on each boundary. If one of previous items is commented out, a free boundary condition $\partial f/\partial x = 0$ is applied instead of it. Every data file has the same format like **Fig. 36**. The unit of flow rate is [t/sec] (metric ton per second). In the case of flow rate, positive value means the amount of incoming flow on the $X_{\min}$ boundary, and it means that of outgoing flow on the $X_{\max}$ boundary. A physical quantity of certain computational time t_n is obtained with time linear interpolations of given data. If t_n exceeds the computational time of last data t_N, afterward the physical quantity of t_n is assumed to be same with that of t_N.

```
0 -0.053
3600 -0.193
7200 -0.233
10800 -0.173
...
[Compt. time(sec)] [Value on boundary]
```

Fig. 36 Example of boundary condition data file

Input Data Files of Gate Conditions

The extents of "boundary domains" are specified on the on the $X_{\min}$ and $X_{\max}$ boundaries by setting values for the following items in the fundamental configuration file ("?" means an index of boundary domain, see pp. 23~ and 76~);

BOUNDARY_POSITION_YMIN_AT_XMAX_?

BOUNDARY_POSITION_YMAX_AT_XMAX_?

BOUNDARY_POSITION_YMIN_AT_XMIN_?

BOUNDARY_POSITION_YMAX_AT_XMIN_?

Besides, you can set some passage domains and wall domains in the y-z cross section of each boundary by specifying filenames of gate conditions for the following items ("?" means an index of boundary domain);

FILE_OF_BOUNDARY_CONDITION_FOR_GATE_AT_XMAX_?

FILE_OF_BOUNDARY_CONDITION_FOR_GATE_AT_XMIN_?

The file format of gate conditions is like **Fig. 37**. The first column shows the computational time [sec], and the second column shows the number of gates (passage domains). From the third column, coordinates of 2 corners on diagonal line: $(y_{\min}^{G1}, z_{\min}^{G1})$ and $(y_{\max}^{G1}, z_{\max}^{G1})$ are lined up in this order to represent the extent of each gate. Up to 10 gates are available. In the case of t_3 in Fig. **37**, the number of gate equals to 0. It means that whole extent is a wall and the wall boundary condition is applied to there instead of any specified conditions.

Operations of opening or closing gates are assumed to be conducted at each specified computational time t_i. In other words, a gate condition which is specified at t_i will be kept until t_{i+1}. For example, when the current computational time t_n is between t_1 and t_2 of **Fig. 37** (i.e. if $t_1 \leq t_n < t_2$), the gate condition of t_1 (1 gate) is applied. If t_n exceeds the computational time of last data t_{Max}, afterward the gate condition of t_n is assumed to be same with that of t_{Max}.

t_1 1 $y_{\min}^{G1}$ $z_{\min}^{G1}$ $y_{\max}^{G1}$ $z_{\max}^{G1}$

t_2 3 $y_{\min}^{G1}$ $z_{\min}^{G1}$ $y_{\max}^{G1}$ $z_{\max}^{G1}$ $y_{\min}^{G2}$ $z_{\min}^{G2}$ $y_{\max}^{G2}$ $z_{\max}^{G2}$ $y_{\min}^{G3}$ $z_{\min}^{G3}$ $y_{\max}^{G3}$ $z_{\max}^{G3}$

t_3 0

t_4 2 $y_{\min}^{G1}$ $z_{\min}^{G1}$ $y_{\max}^{G1}$ $z_{\max}^{G1}$ $y_{\min}^{G2}$ $z_{\min}^{G2}$ $y_{\max}^{G2}$ $z_{\max}^{G2}$

...

t_i N $y_{\min}^{G1}$ $z_{\min}^{G1}$ $y_{\max}^{G1}$ $z_{\max}^{G1}$ $y_{\min}^{G2}$ $z_{\min}^{G2}$ $y_{\max}^{G2}$ $z_{\max}^{G2}$ ... $y_{\min}^{GN}$ $z_{\min}^{GN}$ $y_{\max}^{GN}$ $z_{\max}^{GN}$

...

t_{Max} 1 $y_{\min}^{G1}$ $z_{\min}^{G1}$ $y_{\max}^{G1}$ $z_{\max}^{G1}$

Fig. 37 File format of gate condition data file

Input Data Files of Initial Conditions

The initial conditions of water level are available in the water_level_initilizition function, and those of the other 3-dimensional physical quantities are available in the set_initial_condition_of_ValueOnVolume function. However, you can also set some of them by specifying filenames of input data. The following contents explain configurations of initial water level and other physical quantities with some data files.

Input Data File of Initial Water Level

A filename of initial water level distributions needs to be specified for FILE_OF_INITIAL_WATER_LEVEL in the fundamental configuration file. Its format is like **Fig. 38**. Do not edit the first line. Data of initial water level are listed from the second line. Each line contains a set of coordinates in the computational space: (X_l, Y_l), water level of there: H_l, and x-directional and y-directional distance for interpolations: $w_{x,l}$ and $w_{y,l}$. You can set initial water level at arbitrary number of points. The initial distribution of water level is determined with input values as follows;

$$h_{ij}^0 = h(t_0, x_i, y_{ij}) = \frac{\sum_{l=1}^{N} H_l \exp\left(-\frac{|x_i - X_l|}{w_{x,l}}\right) \exp\left(-\frac{|y_{ij} - Y_l|}{w_{y,l}}\right)}{\sum_{l=1}^{N} \exp\left(-\frac{|x_i - X_l|}{w_{x,l}}\right) \exp\left(-\frac{|y_{ij} - Y_l|}{w_{y,l}}\right)} \quad \textbf{(3.1)}$$

```
X_in_Compt Y_in_Compt WaterLevel Weight_in_X Weight_in_Y
-17000 0 -0.053 500 100
-16800 0 -0.046024 500 100
-16600 0 -0.039047 500 100
-16400 0 -0.032071 500 100
...
[x-coordinate] [y-coordinate] [Water level(m)] [x-directional distance] [y-directional distance]
```

Fig. 38 Example of initial water-level condition data file

Input Data Files of Initial 3D Physical Quantities

Initial conditions for velocity components, salinity, and user-defined scalar #1 ~ #5 are available by specifying data files of vertical distributions at some points. Every data file has the same format, and its filename needs to be specified for a corresponding item in the fundamental configuration file. See also the explanation of each item. Initial condition data for a 3D physical quantity are composed of some "initial vertical distribution data files" and their list: "initial distribution data list". An initial vertical distribution data file contains a vertical distribution of certain physical quantity at certain point.

An initial distribution data list for each physical quantity needs to be specified for one of the following items in the fundamental configuration file;

NAME_OF_LIST_FILE_OF_INITIAL_VELOCITY_X
NAME_OF_LIST_FILE_OF_INITIAL_VELOCITY_Y
NAME_OF_LIST_FILE_OF_INITIAL_VELOCITY_Z
NAME_OF_LIST_FILE_OF_INITIAL_SALINITY
NAME_OF_LIST_FILE_OF_WATER_TEMPERATURE
NAME_OF_LIST_FILE_OF_INITIAL_USER_DEFINED_SCALAR_1

Each initial distribution data list has a format like **Fig. 39**. Do not edit the first line. Data of initial vertical distributions are listed from the second line. Each line contains a set of coordinates in the computational space: (X_l, Y_l), a data filename of vertical distribution at there, and x-directional and y-directional distance for interpolations: $w_{x,l}$ and $w_{y,l}$. You can set vertical distributions at arbitrary number of points. On the other hand, each initial vertical distribution data file has a format like **Fig. 40**. Values of certain physical quantity are listed from the shallower depth to the diaper depth. You can set arbitrary number of values with arbitrary intervals of depth.

Initial distributions of physical quantities on every mesh point (x_i, y_{ij}, z_{ijk}) are determined with these data files as follows;

$$f_{ijk}^0 = f(t_0, x_i, y_{ij}, z_{ijk}) = \frac{\sum_{l=1}^{N} F_l(d_{ijk}) \exp\left(-\frac{|x_i - X_l|}{w_{x,l}}\right) \exp\left(-\frac{|y_{ij} - Y_l|}{w_{y,l}}\right)}{\sum_{l=1}^{N} \exp\left(-\frac{|x_i - X_l|}{w_{x,l}}\right) \exp\left(-\frac{|y_{ij} - Y_l|}{w_{y,l}}\right)} \quad \textbf{(3.2)}$$

Here, $F(d_{ijk})$ is the physical quantity at the depth $d_{ijk} = h_{ij}^0 - z_{ijk}$. It's obtained with vertical interpolations of data in the *l*-th initial vertical distribution data file.

```
X_in_Compt Y_in_Compt Filename Weight_in_X Weight_in_Y
1.5e3 0. vertical_profile_of_salinity_at_0kp.geo 0.5e3 1.e2
1.0e3 0. vertical_profile_of_salinity_at_0kp.geo 0.5e3 1.e2
0.e3 0. vertical_profile_of_salinity_at_10kp.geo 0.5e3 1.e2
-10.e3 0. vertical_profile_of_salinity_at_10kp.geo 0.5e3 1.e2
...
[x-coordinate] [y-coordinate] [Data filename] [x-directional distance] [y-directional distance]
```

Fig. 39 Example of initial distribution data list

```
0.1 15.03
0.5 15.81
1.0 18.35
1.5 20.44
2.0 24.27
2.5 25.52
3.0 26.78
3.5 26.78
4.0 26.78
...
[Depth(m)] [Physical Quantity]
```

Fig. 40 Example of initial vertical distribution data file

User-Input Time-Series Data Files

In addition to the boundary and initial conditions etc. from data files, you can input arbitrary time-series data to the computations. 1-dimensional time series and 2-dimensional time-series distribution on the x-y plane are available as data formats (both of them are available at the same time). A user-input time-series data can be referred in the programs of TITech-WARM by just specifying its filename in the fundamental configuration file. This section explains the formats of 1D and 2D user-input time-series data.

1D User-Input Time-Series Data

Up to 10 1-dimensional user-input time-series data can be read into TITech-WARM. A 1D user-input time-series data will be read by respectively specifying its name and a name of its data file for NAME_OF_USER_INPUT_DATA_?? and FILE_OF_USER_INPUT_DATA_??. Here, "??" represents an index (1 ~ 10) of the data. The format of data files is like **Fig. 41**. Each line contains computational time and the value of data at the time. The value of certain computational time: t_n is obtained with time linear interpolations of given data. The value of certain computational step can be referred in the programs of TITech-WARM with a Parameters-type structure: pram_p as follows;

```
param_p->user_input_data??->current_value
```

Here, "??" represents an index (1 ~ 10) of the data. If t_n exceeds the computational time of last wind data: t_N, afterward the value of t_n is assumed to be same with that of t_N.

```
0 10.
3600. 20.
4200. 40.
...
[Compt. time(sec)] [Value]
```

Fig. 41 Example of user-input 1D data file

2D User-Input Time-Series Data

Up to 10 1-dimensional user-input time-series data can be read into TITech-WARM. A 1D user-input time-series data will be read by respectively specifying its name and a name of its data files' list: "user-input 2D distribution data list" for NAME_OF_USER_INPUT_2D_DATA_?? and NAME_OF_LIST_FILE_OF_USER_INPUT_2D_DATA _??. Here, "??" represents an index (1 ~ 10) of the data. The format of user-input 2D distribution data list is like **Fig. 42**. Each line contains computational time and the name of "user-input 2D distribution data file" at the time. A user-input 2D distribution data file has the format like **Fig. 43** and contains values of the data for all mesh axes.

The value of certain computational step can be referred in the programs of TITech-WARM with a Parameters-type structure: user_input_2d_data?? as follows;

```
param_p->user_input_2d_dataxx->current_value[i][j]
```

Here, "??" means the index (1 ~ 10) of the data. "[i][j]" means the *j*-th mesh axis on the *i*-th mesh plane. If certain computational time t_n exceeds the computational time of last wind data: t_N, afterward the value of t_n is assumed to be same with that of t_N.

```
0. user_input_2d_data_0.geo
300. user_input_2d_data_1.geo
600. user_input_2d_data_2.geo
…
[Compt. time(sec)] [Name of 2D data file]
```

Fig. 42 Example of user-input 2D distribution data list

```
1 1 6.466936e-02
1 2 7.192096e-02
1 3 6.803382e-02
1 4 6.803382e-02
1 5 6.803382e-02
2 1 1.957665e-01
2 2 1.957665e-01
2 3 1.957665e-01
2 4 3.859721e-02
2 5 3.859721e-02
2 6 3.859721e-02
3 1 9.430469e-02
...
[Plane index i] [Axis index j] [Value on ij]
```

Fig. 43 Example of user-input 2D distribution data file

Meteorological Time-Series Data Files (Temperature, Solar Radiation, Humidity, and Cloudiness)

The effects of outer meteorological conditions to the water temperature are computed with the heat transportation model. To compute them, some 1-dimensional time-series data need to be input for all of the air temperature, the solar radiation, the humidity, and the cloudiness.

1D Time-Series Data of Air Temperature

It's read from a 1-dimensional time-series data file which is specified for FILE_OF_AIR_TEMPERATURE. The data file has the same format with that of 1-dimensional user-input time-series data (**Fig. 41**). The first and second columns respectively represent the computational time [sec] and the air temperature [˚C]. The value of certain computational step can be referred in the programs of TITech-WARM with a Parameters-type structure: param_p as follows;

```
param_p->air_temperature->current_value
```

If certain computational time t_n exceeds the computational time of last wind data: t_N, afterward the value of t_n is assumed to be same with that of t_N.

1D Time-Series Data of Solar Radiation

It's read from a 1-dimensional time-series data file which is specified for FILE_OF_SOLOR_RADIATION. The data file has the same format with that of 1-dimensional user-input time-series data (**Fig. 41**). The first and second columns respectively represent the computational time [sec] and the amount of solar radiation [MJ/m^2 day]. The value of certain computational step can be referred in the programs of TITech-WARM with a Parameters-type structure: param_p as follows;

```
param_p->solor_radiation->current_value
```

If certain computational time t_n exceeds the computational time of last wind data: t_N, afterward the value of t_n is assumed to be same with that of t_N.

1D Time-Series Data of Humidity

It's read from a 1-dimensional time-series data file which is specified for FILE_OF_SOLOR_HUMIDITY. The data file has the same format with that of 1-dimensional user-input time-series data (**Fig. 41**). The first and second columns respectively represent the computational time [sec] and the humidity [%]. The value of certain computational step can be referred in the programs of TITech-WARM with a Parameters-type structure: param_p as follows;

```
param_p->humidity->current_value
```

If certain computational time t_n exceeds the computational time of last wind data: t_N, afterward the value of t_n is assumed to be same with that of t_N.

1D Time-Series Data of Cloudiness

It's read from a 1-dimensional time-series data file which is specified for FILE_OF_SOLOR_CLOUDINESS. The data file has the same format with that of 1-dimensional user-input time-series data (**Fig. 41**). The first and second columns respectively represent the computational time [sec] and the cloudiness (0 ~ 10). The value of certain computational step can be referred in the programs of TITech-WARM with a Parameters-type structure: param_p as follows;

```
param_p->cloudiness->current_value
```

If certain computational time t_n exceeds the computational time of last wind data: t_N, afterward the value of t_n is assumed to be same with that of t_N.

Configurations with Program Modifications

In addition to the boundary and initial conditions etc. from data files, you can set precise configurations of the following items for computations of user-defined scalars etc. They are realized by modifying some parts of the programs of TITech-WARM, thus do not forget to compile programs over again with the make command before an execution.

Source Terms of User-Defined Scalars

The source terms of user-defined scalars are computed in the "Phase 8: The Other Computations". Its fundamental equation is the following;

$$\frac{\partial(\phi_{\mathrm{sp}})}{\partial t} = S_{\phi \mathrm{sp}} \tag{2.69}$$

Its computations are realized by representing the equation in the source_term_of_user_defined_scalars function in person. **Fig. 44** shows an example of it. Here, a source term of user-defined scalar #2: ϕ_2 is computed only on the water surface (k=grid_p->end_point[i][j]) by discretizing the following equation with an explicit method;

$$\frac{\partial \phi_2}{\partial t} = (user\_1d\_data_1(t_n) - \phi_2) \times 10^{-5}$$

The term $user\_1d\_data_1(t_n)$ represents the current value of 1-dimensional user-input time-series data #1.

```
void source_term_of_user_defined_scalars(SorobanGrid *grid_p,
                                         Parameters *param_p,
                                         MPIParameters *mpi_param_p,
                                         MiscData *data_p,
                                         ComputationalTime *compt_time_p,
                                         WorkSpace *work_space_p,
                                         double ***uu,
                                         double ***vv,
                                         double ***ww,
                                         double ***density,
                                         double ***salinity,
                                         double ***turb_k,
                                         double ***turb_e,
                                         double ***user_defined_scalar1,
                                         double ***user_defined_scalar2,
                                         double ***user_defined_scalar3,
                                         double ***user_defined_scalar4,
                                         double....)
{
    static int i,j,k;
    static double X,Y,Z;

    // User Defined scalar1
    if(param_p->user_defined_scalar1_active_switch){
      for(i=grid_p->plane_start;i<=grid_p->plane_end;i++){
        for(j=grid_p->line_start[i];j<=grid_p->line_end[i];j++){
          for(k=grid_p->point_start[i][j];k<=grid_p->point_end[i][j];k++){
            X=grid_p->X[i];
            Y=grid_p->Y[i][j];
            Z=grid_p->Z[i][j][k];
            // user_defined_scalar1[i][j][k]=.......;
      }}}
    }

    // User Defined scalar2
    if(param_p->user_defined_scalar2_active_switch){
       for(i=grid_p->plane_start;i<=grid_p->plane_end;i++){
         for(j=grid_p->line_start[i];j<=grid_p->line_end[i];j++){
           for(k=grid_p->point_start[i][j];k<=grid_p->point_end[i][j];k++){
             X=grid_p->X[i];
             Y=grid_p->Y[i][j];
             Z=grid_p->Z[i][j][k];
             if(k==grid_p->point_end[i][j]){
               user_defined_scalar2[i][j][k]+=
                    (param_p->user_input_data1->current_value
                    -user_defined_scalar2[i][j][k])*1.e-5*compt_time_p->dt;
             }
      }}}
    }

   <omitted>

    return;
}
```

Fig. 44 Example of source term computations for a user-defined scalar (source_term_of_user_defined_scalars.c)

Boundary Conditions of User-Defined Scalars on the Water Surface and the River Bed Surface

The boundary conditions of user-defined scalars on the water surface and the river bed surface can be configured in the set_stress_and_flux_on_surfaces function. Various boundary conditions are available by setting a MiscData-type structure: data_p as follows (see also the explanation of this function);

1. If a user-defined scalar has a proportional transportation flux on the water (river bed) surface

In this case, transportation flux of a user-defined scalar is given as follows;

$$\frac{\partial \phi_{\mathrm{sp}}}{\partial t} = \alpha \phi_{\mathrm{sp}}$$

To represent it, substitute the value "0" for

data_p->kind_of_flux_user_defined_scalar??_on_water_surface or

data_p->kind_of_flux_user_defined_scalar??_on_river_bed

and substitute a coefficient α on the *j*-th mesh axis of *i*-th mesh plane for

data_p->flux_user_defined_scalar??_on_water_surface[i][j] or

data_p->flux_user_defined_scalar??_on_river_bed[i][j]

Fig. 45 shows an example of this case. Here, a proportional flux is applied to the user-defined scalar #1 with a coefficient $\alpha = -1.\mathrm{e}-4$ on the river bed surface.

```
<omitted>

   // User Defined scalar1
   if(param_p->user_defined_scalar1_active_switch){
      for(i=grid_p->plane_all_start;i<=grid_p->plane_all_end;i++){
        for(j=grid_p->line_all_start[i];j<=grid_p->line_all_end[i];j++){
            data_p->kind_of_flux_user_defined_scalar1_on_water_surface=1;
            data_p->flux_user_defined_scalar1_on_water_surface[i][j]=0.;
            data_p->kind_of_flux_user_defined_scalar1_on_river_bed=0;
            data_p->flux_user_defined_scalar1_on_river_bed[i][j]=-1.e-4;
      }}
   }
<omitted>
```

Fig. 45 Example of configuration for a user-defined scalar on river bed surface 1

2. If a user-defined scalar has a constant transportation flux on the water (river bed) surface

In this case, transportation flux of a user-defined scalar is given as follows;

$$\frac{\partial \phi_{\mathrm{sp}}}{\partial t} = \alpha$$

To represent it, substitute the value "1" for

data_p->kind_of_flux_user_defined_scalar??_on_water_surface or

data_p->kind_of_flux_user_defined_scalar??_on_river_bed

and substitute the amount of flux α on the *j*-th mesh axis of *i*-th mesh plane for

data_p->flux_user_defined_scalar??_on_water_surface[i][j] or

data_p->flux_user_defined_scalar??_on_river_bed[i][j]

Fig. 46 shows an example of this case. Here, a constant flux $\alpha = 0$ is applied to the user-defined scalar #1 on the water surface.

```
<omitted>
   // User Defined scalar1
   if(param_p->user_defined_scalar1_active_switch){
      for(i=grid_p->plane_all_start;i<=grid_p->plane_all_end;i++){
        for(j=grid_p->line_all_start[i];j<=grid_p->line_all_end[i];j++){
            data_p->kind_of_flux_user_defined_scalar1_on_water_surface=1;
            data_p->flux_user_defined_scalar1_on_water_surface[i][j]=0.;
            data_p->kind_of_flux_user_defined_scalar1_on_river_bed=0;
            data_p->flux_user_defined_scalar1_on_river_bed[i][j]=-1.e-4;
      }}
   }
<omitted>
```

Fig. 46 Example of configuration for a user-defined scalar on river bed surface 2

3. If a user-defined scalar is constant on the water (river bed) surface

In this case, transportation flux of a user-defined scalar is given as follows;

$\phi_{sp} = \alpha$

To represent it, substitute the value "2" for

data_p->kind_of_flux_user_defined_scalar??_on_water_surface or

data_p->kind_of_flux_user_defined_scalar??_on_river_bed

and substitute value of the user-defined scalar α on the *j*-th mesh axis of *i*-th mesh plane for

data_p->flux_user_defined_scalar??_on_water_surface[i][j] or

data_p->flux_user_defined_scalar??_on_river_bed[i][j]

Fig. 47 shows an example of this case. Here, a constant value $\alpha = 10$ is set as the user-defined scalar #1 on the water surface.

```
<omitted>

   // User Defined scalar1
   if(param_p->user_defined_scalar1_active_switch){
      for(i=grid_p->plane_all_start;i<=grid_p->plane_all_end;i++){
        for(j=grid_p->line_all_start[i];j<=grid_p->line_all_end[i];j++){
            data_p->kind_of_flux_user_defined_scalar1_on_water_surface=2;
            data_p->flux_user_defined_scalar1_on_water_surface[i][j]=10.;
            data_p->kind_of_flux_user_defined_scalar1_on_river_bed=0;
            data_p->flux_user_defined_scalar1_on_river_bed[i][j]=-1.e-4;
      }}
   }
<omitted>
```

Fig. 47 Example of configuration for a user-defined scalar on river bed surface 3

Output Data Files

This chapter explains the output data files as the results of computations.

timestep.dat

It's to be output into your workspace (the directory in which you execute the computations). The information about computational time is output on every computational step. It contains the following items;

```
[Step No.] [Time] [Time Interval] [CFL No.] [Week] [Day] [Hour] [Min.] [Sec.] [Exec. Time per Step] [Avg. Exec. Time per Step] [Total Exec. Time]
.
.
.
```

Week, day, hour, min. (minute), and sec. (second) represent the computational time converted into the unit [Day]. The execution (exec.) time represents the actual time to conduct the computations in the unit [sec].

Grid_Position.dat

It's to be output into your workspace. The coordinates of Soroban mesh axes in the real space are output. 2-dimensional input data (wind field etc.) need to be prepared based on it.

Mesh Plane Index i	Mesh Axis Index j	X-coordinate in Real Space [m]	Y-coordinate in Real Space [m]
1	1	6.466936e-02	5.804900e-01
1	2	7.192096e-02	1.391544e+00
1	3	6.803382e-02	1.281178e+00
1	4	6.803382e-02	1.281178e+00
1	5	6.803382e-02	1.281178e+00
2	1	1.957665e-01	1.185633e+00
2	2	1.957665e-01	1.185633e+00
2	3	1.957665e-01	1.185633e+00
2	4	3.859721e-02	1.156766e+00
2	5	3.859721e-02	1.156766e+00
2	6	3.859721e-02	1.156766e+00
3	1	9.430469e-02	6.400581e-01
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.

Tec_TimeDependentData.plt

It's to be output into your workspace. It's a 1-dimensional time series data file written in the TecPlot format. The following variables are output in every time interval determined in the fundamental configuration file (variables of a step are written in one line);

Variables	Names in the TecPlot file
Computational time [s]	Time
Step number	Steps
Output file index	Output_flame_number
Water level on $X_{\min}$boundary domain #1 [m]	WaterLevel_At_Xmin_1[m]
Flow rate on $X_{\min}$ boundarydomain #1 [m^3/s]	Q_At_Xmin_1[m^3/s]
Main flow velocity on $X_{\min}$ boundarydomain #1 [m/s]	Velocity_At_Xmin_1[m/s]
Salinity on $X_{\min}$ boundary domain #1 [psu]	Salinity_At_Xmin_1[psu]
Water temperature on $X_{\min}$ boundarydomain #1 [˚C]	Water_Temperature_At_Xmin_1[degree_Celcius]
User-defined scalar #1 on $X_{\min}$ boundarydomain #1	
……	
Water level on $X_{\max}$ boundary domain #1 [m]	WaterLevel_At_Xmax_1[m]
Flow rate on $X_{\max}$ boundary domain #1 [m^3/s]	Q_At_Xmax_1[m^3/s]
Main flow velocity on $X_{\max}$ boundary domain #1 [m/s]	Velocity_At_Xmax_1[m/s]
Salinity on $X_{\max}$ boundary domain #1 [psu]	Salinity_At_Xmax_1[psu]
Water temperature on $X_{\max}$ boundary domain #1 [˚C]	Water_Temperature_At_Xmax_1[degree_Celcius]

User-defined scalar #1 on $X_{\max}$ boundary domain #1	
……	
User-input 1D time-series data #1	
User-input 1D time-series data #2	
……	

Computational time, step number, and output file index are always output. The output file index is a serial number which is added on the filename of TecPlot-format interim results (Tec_VolumeData_xxx.plt, Tec_SurfaceData_xxx.plt). Each of the other boundary values is output if its input data file is specified in the fundamental configuration file. If a user-input 1D time-series data is specified in the fundamental configuration file, its value of certain time is also output.

Tec_SurfaceData_time_xxx.plt / Tec_SurfaceData_step_xxx.plt

It's to be output into the "TecData" directory in your workspace. The variables which have 2-dimensional distributions (water level, height of river bet, wind velocities etc.) of every mesh axis are output. If you set OUTPUT_INTERVAL_SWITCH = 1 in the fundamental configuration file, the filename becomes Tec_SurfaceData_time_xxx.plt. If you set OUTPUT_INTERVAL_SWITCH = 0, the filename becomes Tec_SurfaceData_step_xxx.plt. Here, "xxx" in these filenames represents a serial number which increasesinevery output.Thenthe relationship between this index and computational time (step number) is as follows;

Filename	Computational time (step no.)
Tec_SurfaceData_time_1.plt (Tec_SurfaceData_step_1.plt)	Initial conditions
Tec_SurfaceData_time_2.plt (Tec_SurfaceData_step_2.plt)plt	$t = \mathrm{OUTPUT\_TIME\_INTERVAL}$ $(n = \mathrm{OUTPUT\_STEP\_INTERVAL})$
Tec_SurfaceData_time_3.plt (Tec_SurfaceData_step_3.plt)	$t = \mathrm{OUTPUT\_TIME\_INTERVAL} \times 2$ $(n = \mathrm{OUTPUT\_STEP\_INTERVAL} \times 2)$
…….	

The variables to be output are the following;

Variables	Names in the TecPlot file
x-coordinate (in computational space)	X_comp_space
y-coordinate (in computational space)	Y_comp_space
x-coordinate (in real space)	X_real_space
y-coordinate (in real space)	Y_real_space
Water level [m]	Water_Level[m]

Height of river bed [m]	River_Bed[m]
Depth [m]	Depth[m]
Wind velocity at altitudes above 10 m (x-component in computational space) [m/s]	Wind_at_10m_height(X_comp_Component)
Wind velocity at altitudes above 10 m (y-component in computational space) [m/s]	Wind_at_10m_height(Y_comp_Component)
Wind velocity at altitudes above 10 m (x-component in real space) [m/s]	Wind_at_10m_height(X_real_Component)
Wind velocity at altitudes above 10 m (y-component in real space) [m/s]	Wind_at_10m_height(Y_real_Component)
User-input 2D time-series data #1	
User-input 2D time-series data #2	
……	

Wind field data and user-input 2D time-series data are output only if their input data files are prepared. No data is output for land areas.

Tec_VolumeData_time_xxx.plt / Tec_VolumeData_step_xxx.plt

It's to be output into the "TecData" directory in your workspace. The variables which have 3-dimensional distributions (salinity, flow velocities etc.) of every mesh point are output. If you set OUTPUT_INTERVAL_SWITCH = 1 in the fundamental configuration file, the filename becomes Tec_VolumeData_time_xxx.plt. If you set OUTPUT_INTERVAL_SWITCH = 0, the filename becomes Tec_VolumeData_step_xxx.plt. Here, "xxx" in these filenames represents a serial number which increases in every output. Then the relationship between this index and computational time (step number) is as follows;

Filename	Computational time (step no.)
Tec_VolumeData_time_1.plt (Tec_VolumeData_step_1.plt)	Initial conditions
Tec_VolumeData_time_2.plt (Tec_VolumeData_step_2.plt)	$t = \mathrm{OUTPUT\_TIME\_INTERVAL}$ $(n = \mathrm{OUTPUT\_STEP\_INTERVAL})$
Tec_VolumeData_time_3.plt (Tec_VolumeData_step_3.plt)	$t = \mathrm{OUTPUT\_TIME\_INTERVAL} \times 2$ $(n = \mathrm{OUTPUT\_STEP\_INTERVAL} \times 2)$
…….	

Not only variables computed on Soroban meshes but topographic data (river bed, riverbanks etc.) are to be output with the Zone format of TecPlot. The variables to be output into the Zone name "Volume Data" are the following;

Variables	Names in the TecPlot file
x-coordinate (in computational space)	X_comp_space
y-coordinate (in computational space)	Y_comp_space
z-coordinate (in computational space)	Z
x-coordinate (in real space)	X_real_space

y-coordinate (in real space)	Y_real_space
Depth [m]	Depth
Flow velocity (x-component in computational space) [m/s]	U_comp_space[m/s]
Flow velocity (y-component in computational space) [m/s]	V_comp_space[m/s]
Flow velocity (vertical component) [m/s]	W[m/s]
Wind velocity at altitudes above 10 m (x-component in real space) [m]	U_real_space[m/s]
Wind velocity at altitudes above 10 m (y-component in real space) [m]	V_real_space[m/s]
Salinity [psu]	Salinity[psu]
Water temperature [˚C]	Water_Temperature[degree_Celcius]
Pressure [Pa]	Pressure[Pa]
Density [kg/m^3]	Density[kg/m^3]
Turbulent kinetic energy	Turbulent_K
Turbulent dispersion rate	Turbulent_E
Turbulent viscosity	Turblent_viscosity_vt
Horizontal turbulent viscosity (x-component)	Horizontal_Turblent_viscosity_X[m^2/s]
Horizontal turbulent viscosity (y-component)	Horizontal_Turblent_viscosity_Y[m^2/s]
User-defined scalar #1	
User-defined scalar #2	
……	

The user-defined scalar is output only if it's specified in the fundamental configuration file. No data is output into the "Volume Data" Zone for land areas. However, the topography of river bed and riverbanks is output into the "Bottom Surface" Zone. Thus visualize them (with shade description etc.) to see the topography.

File Compression

If you set OUTPUT_COMPRESSION_SWITCH=1 in the fundamental configuration file, each output files in the "TecData" directory (Tec_SurfaceData, Tec_VolumeData) is compressed into bzip archive automatically. They have the .bz2 extension, so use the following command to extract them;

```
> bunzip2 –fv<filename>.bz2
```

Execution of Comuputations

Required Environment

TITech-WARM is written in C language and uses MPI libraries. Thus the following environment needs to be set up in your computer;

- MPI librariy（OpenMPI, MPICH, LAM）
- C compiler（GNU C gcc, Intel Compiler icc）
- bzip2 command（if you want to compress output files）

We confirmed that TITech-WARM works normally on the Linux (Fedora, SUSE) and Mac OS X (10.5). When you use it on a cluster of network-connected computers, not only the previous conditions (on every computer) but the following environment is required;

- Storage sharing with NFS
- Giga Bit Ethernet or faster network
- Permission configurations of rlogin and rsh

Refer some books or consult us (tnakamur@depe.titech.ac.jp) to learn how to set up environment to use MPI programs on a cluster of network-connected computers. Here we show some examples of environment on which we confirmed normal working of TITech-WARM;

MacPro

- Built by Apple Inc.
- Xenon 2.8 GHz * 2 CPUs (12 cores = 12 threads in total)
- 6 GB memory
- Required environment (OpenMPI, gcc, bzip2 etc.) is available just by downloading and installing the Development Tools
- A computer cluster needs to be built to execute computations with more than 12 threads
- Around 400,000 JPY per a computer
- Around 7 times of computational time is required to execute enclosed case of Kamafusa water reservoir

Core i7 Culster（Network-Connected Computers）

- Built by ourselves
- Core i7 3690K 3.4 GHz * 6 CPUs (6 cores = 6 threads)
- OpenMPI, Intel Compiler, and gcc are installed
- NFS, rlogin, and rsh are configured in person
- Around 200,000 JPY per a computer
- OS: Fedora 5
- 3 computers are connected through network (18 cores in total)
- 8 GB memory
- Around 20 times of computational time is required to execute enclosed case of Kamafusa water reservoir

Installration of TITech-WARM

No special procedure is necessary to install TITech-WARM; just copy and paste the TITech-WARM directory from the CD-ROM into your workspace. C source files, header files, a makefile, and a set of configuration files (an example of Kamafusa water reservoir) are enclosed. A C source file with the same name is prepared for every function in TITech-WARM except the set_pressure function. Only an object file is prepared for the set_pressure function. However, it's not compatible with other OS or environment, thus we'll visit you to install it after your computational environment is built.

Compiling

When you execute TITech-WARM for the first time, or if you modified a part of source codes, you need to compile the codes. A makefile is prepared, so you can compile the source codes and get an executable file "titech_warm" by just putting

```
#> make
```

The mpicc command in a MPI library is used for the compiling, so a MPi library is need to be properly installed in your computer.

Put the following commands to compile all functions over again;

```
#> make clean

#> make
```

All object files (*.o) in the directory and the executable "titech_warm" are deleted with the make clean command.

Execution

A set of parallel computations is executed with the mpirun command. How to descript options of the command depends on your MPI library. Here we show an example of execution with OpenMPI;

```
#>mpirun –np 8 titech_warm define.inc
```

This is the example of an execution with 8 parallel threads and the fundamental configuration file "define.inc".

Deletion of Output Data Files

You can delete the results of computations with the following command;

```
#> make clean_data
```

All data files (*.dat, *.plt, and *.log) in the TITech-WARM directory and all data files (*.plt and *.plt.bz2) in the TecData directory are deleted with this command.

Contact Information

(As of February, 2017)

Tokyo Institute of Technology, Dept. of Transdisciplinary Science and Engineering

Assoc. Prof. Takashi Nakamura

4259-G5-3, Nagatsutacho, Midori-ku, Yokohama, Kanagawa, 226-8502, Japan

Tel: +81-45-924-5548

Fax: +81-45-924-5549

mail: tnakamur@depe.titech.ac.jp