PCOMCOT User Manual
(version 2.1)

by

Yifan Zhu (zyftop@sjtu.edu.cn)

Chao An (anchao@sjtu.edu.cn)

School of Ocean and Civil Engineering, Shanghai Jiao Tong University
Shanghai 200240, China

June 13, 2025

Abstract

PCOCMOT is a parallel computer program for simulating nonlinear dispersive tsunami waves. It is developed based on the shallow water model COMCOT (Cornell Multi-grid Coupled Tsunami) and a depth-integrated non-hydrostatic model, with the motivation to enhance computational efficiency and accuracy of large-scale tsunami simulations. The main features of this 2.1 version include: i) accounting for wave dispersion by correcting the shallow water equations with non-hydrostatic pressure terms; ii) moving boundary technique for inundation; iii) eddy-viscosity scheme for wave breaking; iv) nested grids for cross-scale tsunami modeling; v) parallel implementation on both CPU and GPU. This manual provides a detailed description of the governing equations and numerical schemes, as well as instructions on how to use the program. Besides, various numerical examples are presented for model validation and performance analysis.

Source code can be downloaded from https://github.com/yifanzhu-fluid/PCOMCOT2.1

Contents

1 Introduction

In the past two decades, various tsunami models have been developed for tsunami prediction and warning, including the 2D depth-integrated models and the 3D models directly solving the Navier-Stokes equations. The 2D models are adopted much more widely than 3D models, as they have sufficient accuracy and much lower computational cost. There are basically two types of 2D models, i.e., the shallow water models which treat tsunamis as ideal long waves, and the Boussinesq-type models considering wave dispersion. Most operational tsunami codes are based on the shallow water equations (SWEs), such as COMCOT (*Liu et al.*, 1998; *Wang and Liu*, 2006), GEOCLAW (*LeVeque et al.*, 2011), MOST (*Titov and González*, 1997; *Titov and Synolakis*, 1998), TUNAMI (*Imamura*, 1996), etc. These codes are commonly applied to tsunamis generated by large earthquakes, and have been validated in many real tsunami events (e.g., *Wang and Liu*, 2006; *Harig et al.*, 2008; *Arcos and LeVeque*, 2015; *Heidarzadeh et al.*, 2016; *Oishi et al.*, 2015). For landslide tsunamis and trans-oceanic earthquake tsunamis, where the wavelength is relatively short or the propagation distance is as long as thousands of kilometers, the dispersive effects may be significant (*Glimsdal et al.*, 2013). Under such circumstances, the Boussinesq-type models can provide more accurate results. Different Boussinesq-type models such as FUNWAVE (*Shi et al.*, 2012), JAGURS (*Baba et al.*, 2015, 2017), and COULWAVE (*Lynett and Liu*, 2004; *Lynett et al.*, 2002) are now available, and are mainly used by the scientific community in a research context.

Although the Boussinesq-type models satisfactorily describe the dynamics of tsunami waves, they are quite difficult to solve. As the higher-order derivatives in the dispersive terms can easily cause numerical instability, it is generally necessary to either simplify the governing equations or implement complicated numerical treatment. *Baba et al.* (2015) rewrite the dispersive terms for constant water depth with conserved variables (i.e., volume flux), and solve the equations with the finite-difference scheme. For non-uniform water depth, the higher-order derivatives of non-conserved variables are unavoidable, and more advanced schemes are needed to improve stability. For example, *Shi et al.* (2012) utilize the MUSCL-TVD finite-volume scheme and shock-capturing technique to solve the fully nonlinear Boussinesq equations. These complex methods are much more time-consuming than solving the SWEs. As both accuracy and efficiency are important in tsunami modeling, we aim to seek a dispersive model which can be stably solved with relatively simple schemes.

The non-hydrostatic model, which decomposes the water pressure into hydrostatic and non-hydrostatic components, are commonly used in 3D modeling of free surface flows (e.g., *Casulli*, 1999; *Koçyigit et al.*, 2002; *Ma et al.*, 2012; *Stelling and Zijlema*, 2003; *Zijlema et al.*, 2011). Such model effectively describes wave dispersion, and just 2∼3 vertical grid cells can yield good results for highly dispersive waves (*Stelling and Zijlema*, 2003). When using a single vertical layer, the 2D version (i.e., depth-integrated non-hydrostatic model) shows accuracy comparable to the classical Boussinesq equations, but is much simpler without higher-order derivatives (*Stelling and Zijlema*, 2003; *Yamazaki et al.*, 2009; *Zijlema and Stelling*, 2008). The depth-integrated non-hydrostatic model accounts for wave dispersion by correcting the SWEs with non-hydrostatic pressure terms, and can be solved efficiently with a semi-implicit scheme (e.g., NEOWAVE, *Yamazaki et al.*, 2009, 2011). This scheme is directly applicable to existing shallow water models, that is, adding the implicitly solved correction terms to the explicit solution of SWEs.

Beside improvement of governing equations and numerical schemes, parallel computing is another way to increase the efficiency of tsunami modeling. Most aforementioned tsunami models have been parallelized for running on multiple CPU cores of HPC clusters (e.g., *An et al.*, 2014; *Baba et al.*, 2015; *Shi et al.*, 2012), which enables large-scale computation at reasonable time cost. However, HPC clusters are highly expensive, and satisfactory acceleration may not be achieved sometimes due to the time used by inter-core communication. These days, graphics processing units (GPUs) are playing a more and more important role in numerical computation. Different from CPUs, GPUs are highly parallel architectures with thousands of cores, and thus can provide much higher throughput. Some dispersive tsunami models have recently been ported to GPUs. For example, *Yuan et al.* (2020) developed a GPU version of FUNWAVE, and reported a speedup ratio of > 4 compared with the CPU version running on a 36-core HPC node. Considering the variety of computational environments, both CPU- and GPU-parallelization are needed to achieve satisfactory performance on different platforms.

In this study, we develop a depth-integrated non-hydrostatic model and the corresponding tsunami simulation package named PCOMCOT (Parallelized COMCOT). We theoretically derive the depth-integrated non-hydrostatic model based on the Boussinesq equations. Our governing equations are shown to have slightly better accuracy for wave dispersion than those in previous studies (*Stelling and Zijlema*, 2003; *Yamazaki et al.*, 2009; *Zijlema and Stelling*, 2008). This non-hydrostatic model is added to the widely used tsunami package COMCOT (Cornell Multi-grid

Coupled Tsunami, *An et al.*, 2014; *Wang and Liu*, 2006; *Liu et al.*, 1998). First, the SWEs are explicitly solved with the similar method as COMCOT. Then, the non-hydrostatic pressure is calculated implicitly and used to correct the shallow water solution, which gives the dispersive result. For near-shore processes, a moving-boundary technique is used to track wave run-up and run-down, and an eddy-viscosity scheme is employed to handle wave breaking. A nested grid system is adopted for tsunami modeling across different scales. For parallel implementation of PCOMCOT, both a CPU version using the MPI library, and a CUDA-based GPU version are provided.

In summary, PCOMCOT is capable of simulating the whole life span of tsunamis — generation, propagation and inundation. It calculates nonlinear, dispersive, and breaking tsunami waves stably and efficiently. Some important features of PCOMCOT are listed below.

- Non-hydrostatic pressure added to shallow water equations for wave dispersion
- Moving boundary technique for run-up and run-down.
- Eddy-viscosity scheme for wave breaking.
- One- and two-way nesting grids.
- Parallel implementation on both CPU and GPU.

2 Governing Equations

2.1 Depth-integrated Non-hydrostatic Model

In this section, we describe the governing equations of depth-integrated non-hydrostatic free surface flows. These equations are the ones that PCOMCOT solves to simulate tsunamis. Here, for the first time, the depth-integrated non-hydrostatic model is theoretically derived based on the Boussinesq equations. By adopting a more realistic approximation of non-hydrostatic pressure, the accuracy in dispersion is slightly improved compared with the previous models (*Yamazaki et al.*, 2009; *Stelling and Zijlema*, 2003; *Zijlema and Stelling*, 2008), without changing the simple form.

In the non-hydrostatic model, the water pressure p is decomposed into a hydrostatic part p_{sta} and a non-hydrostatic part p_{dyn}, as follows.

$$p = \rho\left(p_{\text{sta}} + p_{\text{dyn}}\right) = \rho\left[g(\eta - z) + p_{\text{dyn}}\right], \tag{2-1}$$

in which η represents the surface elevation, g is the gravitational acceleration, and the pressure is normalized by water density. Thus, neglecting viscosity, the 3D continuity equation and incompressible Navier-Stokes equations are expressed as

$$\begin{cases} \nabla \cdot \mathbf{u} + \dfrac{\partial w}{\partial z} = 0, & \text{(2-2a)} \\ \dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + w\dfrac{\partial \mathbf{u}}{\partial z} + g\nabla\eta + \nabla p_{\text{dyn}} = 0, & \text{(2-2b)} \\ \dfrac{\partial w}{\partial t} + \mathbf{u} \cdot \nabla w + w\dfrac{\partial w}{\partial z} + \dfrac{\partial p_{\text{dyn}}}{\partial z} = 0, & \text{(2-2c)} \end{cases}$$

in which $\mathbf{u}$ and w denote the horizontal and vertical velocity components, respectively. Note that the gradient operator ∇ works only in horizontal direction. The kinematic and dynamic boundary conditions at the free surface and the seabed are

$$\begin{cases} \dfrac{\partial \eta}{\partial t} + \mathbf{u} \cdot \nabla\eta - w = 0, & z = \eta, & \text{(2-3a)} \\ p_{\text{dyn}} = 0, & z = \eta, & \text{(2-3b)} \\ \dfrac{\partial h}{\partial t} + \mathbf{u} \cdot \nabla h + w = 0, & z = -h = -(h_1 + h_b), & \text{(2-3c)} \end{cases}$$

where the water depth h is divided into h_1 and h_b, representing the initial water depth which does

not change with time, and the depth change caused by seafloor motion, respectively.

For the sake of clarity, we will manipulate these equations in their dimensionless form. The wave amplitude A, the characteristic water depth h_0, the wavenumber k, and the characteristic phase speed of linear long wave $\sqrt{gh_0}$ are used for the non-dimensionalization. Following *Chiang et al.* (2005), the dimensionless variables are

$$\begin{aligned} &(x', y') = k(x, y), \quad z' = \frac{z}{h_0}, \quad t' = k\sqrt{gh_0}t, \quad h' = \frac{h}{h_0}, \quad h_1' = \frac{h_1}{h_0}, \\ &\eta' = \frac{\eta}{A}, \quad h_b' = \frac{h_b}{A}, \quad \mathbf{u}' = \frac{\mathbf{u}}{\varepsilon\sqrt{gh_0}}, \quad w' = \frac{\mu}{\varepsilon\sqrt{gh_0}}w, \quad p_{\mathrm{dyn}}' = \frac{p_{\mathrm{dyn}}}{gA}. \end{aligned} \tag{2-4}$$

Here we have introduced two important small parameters $\varepsilon = A/h_0$ and $\mu^2 = (kh_0)^2$, which represent wave nonlinearity and frequency dispersion, respectively. They are assumed to be in the same magnitude for weakly nonlinear and moderately dispersive waves, that is,

$$O(\varepsilon) \approx O(\mu^2) \ll 1. \tag{2-5}$$

The governing equations (2-2a) to (2-2c) and boundary conditions (2-3a) to (2-3c) are nondimensionalized to be

$$\begin{cases} \mu^2 \nabla \cdot \mathbf{u} + \dfrac{\partial w}{\partial z} = 0, & \text{(2-6a)} \\ \dfrac{\partial \mathbf{u}}{\partial t} + \varepsilon\left(\mathbf{u}\cdot\nabla\mathbf{u} + \dfrac{1}{\mu^2}w\dfrac{\partial \mathbf{u}}{\partial z}\right) + \nabla\eta + \nabla p_{\mathrm{dyn}} = 0, & \text{(2-6b)} \\ \dfrac{\partial w}{\partial t} + \varepsilon\left(\mathbf{u}\cdot\nabla w + \dfrac{1}{\mu^2}w\dfrac{\partial w}{\partial z}\right) + \dfrac{\partial p_{\mathrm{dyn}}}{\partial z} = 0, & \text{(2-6c)} \end{cases}$$

$$\begin{cases} \mu^2\left(\dfrac{\partial \eta}{\partial t} + \varepsilon\mathbf{u}\cdot\nabla\eta\right) - w = 0, \quad z = \varepsilon\eta, & \text{(2-7a)} \\ p_{\mathrm{dyn}} = 0, \quad z = \varepsilon\eta, & \text{(2-7b)} \\ \mu^2\left(\dfrac{\partial h_b}{\partial t} + \mathbf{u}\cdot\nabla h\right) + w = 0, \quad z = -h = -(h_1 + \varepsilon h_b). & \text{(2-7c)} \end{cases}$$

Note that the primes of dimensionless variables are omitted for convenience, and $\frac{\partial h}{\partial t} = \varepsilon\frac{\partial h_b}{\partial t}$ is used. By integrating the continuity equation (2-6a) along the z direction from $-h$ to $\varepsilon\eta$ and applying the

kinematic boundary conditions (2-7a, 2-7c), we obtain the 2D continuity equation as

$$\frac{\partial}{\partial t}\big(\eta + h_b\big) + \nabla \cdot \big[\big(h + \varepsilon\eta\big)\overline{\mathbf{u}}\big] = 0, \tag{2-8}$$

where $\overline{\mathbf{u}}$ is the depth-averaged horizontal velocity defined as $\overline{\mathbf{u}} = \frac{1}{h+\varepsilon\eta}\int_{-h}^{\varepsilon\eta}\mathbf{u}\mathrm{d}z$. Note that the Leibniz rule is used in the integration, that is,

$$\nabla \cdot \int_{-h}^{\varepsilon\eta}\mathbf{u}\mathrm{d}z = \int_{-h}^{\varepsilon\eta}\nabla\cdot\mathbf{u}\mathrm{d}z + \mathbf{u}\Big|_{z=\varepsilon\eta}\cdot\varepsilon\nabla\eta + \mathbf{u}\Big|_{z=-h}\cdot\nabla h. \tag{2-9}$$

Since we are seeking equations with similar accuracy as the Boussinesq equations , we will directly use the solutions of the Boussinesq equations to simplify the governing equations. For the classical Boussinesq equations with terms up to $O(\varepsilon,\mu^2)$(e.g., *Chiang et al.*, 2005; *Peregrine*, 1967), the horizontal and vertical velocities $\mathbf{u}$ and w are written as

$$\begin{cases} \mathbf{u} = \overline{\mathbf{u}} + \mu^2\mathbf{u}^*, & \text{(2-10a)} \\ w = -\mu^2\left[\big(z+h\big)\nabla\cdot\overline{\mathbf{u}} + \overline{\mathbf{u}}\cdot\nabla h\right]. & \text{(2-10b)} \end{cases}$$

Here, $\mathbf{u}^*(x,y,z,t)$ is the dispersive component of horizontal velocity, and $\int_{-h}^{\varepsilon\eta}\mathbf{u}^*\mathrm{d}z = 0$. By substituting (2-10a, 2-10b) into equations (2-6b, 2-6c), and ignoring the higher-order terms, the N-S equations are approximated as

$$\begin{cases} \dfrac{\partial}{\partial t}\big(\overline{\mathbf{u}} + \mu^2\mathbf{u}^*\big) + \varepsilon\overline{\mathbf{u}}\cdot\nabla\overline{\mathbf{u}} + \nabla\eta + \nabla p_{_{\mathrm{dyn}}} = 0, & \text{(2-11a)} \\ \dfrac{\partial w}{\partial t} + \dfrac{\partial p_{_{\mathrm{dyn}}}}{\partial z} = 0. & \text{(2-11b)} \end{cases}$$

Integrating equations (2-11a, 2-11b) over the total water depth, and neglecting $O(\varepsilon\mu^2)$ terms, we obtain the following depth-integrated momentum equations after some manipulations.

$$\begin{cases} \dfrac{\partial\mathbf{F}}{\partial t} + \varepsilon\nabla\cdot\left(\dfrac{\mathbf{FF}}{D}\right) + D\nabla\eta + \nabla\left(\displaystyle\int_{-h}^{\varepsilon\eta} p_{_{\mathrm{dyn}}}\mathrm{d}z\right) - q\nabla h = 0, & \text{(2-12a)} \\ \dfrac{\partial W}{\partial t} - \dfrac{q}{D} = 0. & \text{(2-12b)} \end{cases}$$

Here $\mathbf{F}$ is the horizontal volume flux, $\mathbf{FF}$ is a dyadic tensor (i.e., the outer product of $\mathbf{F}$ and itself), D is the total water depth, q is the non-hydrostatic pressure at the bottom, and W is the

depth-averaged vertical velocity.

$$\mathbf{F}=\big(h+\varepsilon\eta\big)\overline{\mathbf{u}},\qquad D=\big(h+\varepsilon\eta\big),\qquad q=p_{_{\rm dyn}}\Big|_{z=-h},\qquad W=\frac{1}{h+\varepsilon\eta}\int_{-h}^{\varepsilon\eta}w\,\mathrm{d}z. \tag{2-13}$$

Because the vertical velocity varies linearly in z direction as indicated by equation (2-10b), W is simply the average of w at the free surface and the bottom. Again, please note that the Leibniz rule is applied when integrating $\overline{\mathbf{u}}_t$ and $\nabla p_{_{\rm dyn}}$ along z direction, which is

$$\begin{cases}\dfrac{\partial}{\partial t}\displaystyle\int_{-h}^{\varepsilon\eta}\overline{\mathbf{u}}\mathrm{d}z=\int_{-h}^{\varepsilon\eta}\frac{\partial\overline{\mathbf{u}}}{\partial t}\mathrm{d}z+\varepsilon\frac{\partial\eta}{\partial t}\overline{\mathbf{u}}+\varepsilon\frac{\partial h_b}{\partial t}\overline{\mathbf{u}}, & \text{(2-14a)}\\[2ex] \nabla\displaystyle\int_{-h}^{\varepsilon\eta}p_{_{\rm dyn}}\mathrm{d}z=\int_{-h}^{\varepsilon\eta}\nabla p_{_{\rm dyn}}\mathrm{d}z+p_{_{\rm dyn}}\Big|_{z=\varepsilon\eta}\varepsilon\nabla\eta+p_{_{\rm dyn}}\Big|_{z=-h}\nabla h. & \text{(2-14b)}\end{cases}$$

And the relation below is used to derive the conservative form in equation (2-12a).

$$\big(h+\varepsilon\eta\big)\overline{\mathbf{u}}\cdot\nabla\overline{\mathbf{u}}=\nabla\cdot\big[\big(h+\varepsilon\eta\big)\overline{\mathbf{u}}\,\overline{\mathbf{u}}\big]-\nabla\cdot\big[\big(h+\varepsilon\eta\big)\overline{\mathbf{u}}\big]\,\overline{\mathbf{u}}. \tag{2-15}$$

The above depth-integrated continuity and momentum equations (2-8, 2-12) are expressed in dimensional form as

$$\begin{cases}\dfrac{\partial}{\partial t}\big(\eta+h\big)+\nabla\cdot\mathbf{F}=0, & \text{(2-16a)}\\[2ex] \dfrac{\partial\mathbf{F}}{\partial t}+\nabla\cdot\left(\dfrac{\mathbf{F}\,\mathbf{F}}{D}\right)+gD\nabla\eta+\nabla\left(\displaystyle\int_{-h}^{\eta}p_{_{\rm dyn}}\mathrm{d}z\right)-q\nabla h=0, & \text{(2-16b)}\\[2ex] \dfrac{\partial W}{\partial t}=\dfrac{q}{D}, & \text{(2-16c)}\end{cases}$$

where $\mathbf{F}=\int_{-h}^{\eta}\mathbf{u}\mathrm{d}z=\big(h+\eta\big)\overline{\mathbf{u}}$, $D=h+\eta$, and W is simply the average value of w at the free surface and the bottom given by (2-3a, 2-3c). These equations have the same accuracy in nonlinearity and dispersion as the classical Boussinesq equations, but cannot be solved yet, because the integration of $p_{_{\rm dyn}}$ is not given. A reasonable approximation of $p_{_{\rm dyn}}$ is needed for evaluation of this integration.

The expression of $p_{_{\rm dyn}}$ corresponding to the classical Boussinesq equations is given by *Peregrine* (1967) as

$$p_{_{\rm dyn}}=z\frac{\partial}{\partial t}\nabla\cdot(h\overline{\mathbf{u}})+\frac{z^2}{2}\frac{\partial}{\partial t}\nabla\cdot\overline{\mathbf{u}}, \tag{2-17}$$

in which $p_{_{\rm dyn}}$ varies quadratically in the vertical direction. This expression cannot be directly adopted in the depth-integrated non-hydrostatic model, because depth-integration of (2-17) would

lead to complex coefficients of q in the horizontal momentum equation. These coefficients contain higher-order derivatives, making it highly difficult to stably solve q. To simplify the depth-integration of p_{dyn} , we rewrite (2-17) as

$$p_{\mathrm{dyn}} = \left\{ \frac{\left(z+h\right)^2 - D^2}{2} + \tau D \left[\left(z+h\right) - D \right] \right\} \nabla \cdot \overline{\mathbf{u}}_t. \tag{2-18}$$

Here, we assume that $h \approx D$ and neglect the terms containing $\partial h/\partial t$, which only causes difference in the order of $O(\varepsilon\mu^2)$. And the dimensionless parameter τ in (2-18) is expressed as

$$\tau = \frac{\overline{\mathbf{u}}_t \cdot \nabla h}{h \nabla \cdot \overline{\mathbf{u}}_t} \sim \frac{\nabla h}{kh} = \frac{1}{2\pi} \frac{\lambda}{h/\nabla h}. \tag{2-19}$$

Though τ is a variable with respect to time and space, its order can be roughly estimated to be $\nabla h/(kh)$, which represents the contribution of bottom slope to wave dispersion. Since p_{dyn} mainly exists for short waves in the deep ocean with a smooth bottom, where $|\nabla h|$ is much less than kh, it is reasonable to assume that $\tau \approx 0$. Correspondingly, the vertical distribution of p_{dyn} is approximated as a pure quadratic function of $z+h$ (i.e., vertical distance from the seabed), and the integration of p_{dyn} becomes

$$\int_{-h}^{\eta} p_{\mathrm{dyn}} \mathrm{d}z \approx \int_{-h}^{\eta} \frac{\left(z+h\right)^2 - D^2}{2} \nabla \cdot \overline{\mathbf{u}}_t \mathrm{d}z = -\frac{D^3}{3} \nabla \cdot \overline{\mathbf{u}}_t = \frac{2}{3} qD. \tag{2-20}$$

In fact, *Zhang et al.* (2021) have shown that even on a relatively steep bottom (e.g., $\theta = 30°$), the effect of bottom slope on the vertical profile of p_{dyn} is negligible. Thus, equation (2-20) is generally an appropriate approximation to the depth-integration of non-hydrostatic pressure.

Substituting (2-20) into (2-16b), we finally obtain the governing equations of the depth-integrated non-hydrostatic model, that is

$$\begin{cases} \dfrac{\partial}{\partial t}\left(\eta + h\right) + \nabla \cdot \mathbf{F} = 0, & \text{(2-21a)} \\ \dfrac{\partial \mathbf{F}}{\partial t} + \nabla \cdot \left(\dfrac{\mathbf{F}\,\mathbf{F}}{D} \right) + gD\nabla\eta + \alpha \left[D\nabla q + q\nabla\left(\eta - \beta h\right) \right] = 0, & \text{(2-21b)} \\ \dfrac{\partial W}{\partial t} = \dfrac{q}{D}, & \text{(2-21c)} \end{cases}$$

in which the values of coefficients α and β are $\alpha = 2/3$, $\beta = 0.5$. The equations (2-21) are the same as the conservative shallow water equations, except the addition of a vertical momentum equation

and the non-hydrostatic terms in the horizontal momentum equation. Thus, the whole system is closed with three unknowns $\{\eta, \mathbf{F}, q\}$ being solved by three equations. In the above derivation, we approximate the complex vertical distribution of p_{dyn} as a simple quadratic function. As a result, the horizontal momentum equation is significantly simplified to (2-21b). By avoiding higher-order derivatives, we can obtain similar dispersive results as the classical Boussinesq equations with much simpler schemes, which is the main advantage of the depth-integrated non-hydrostatic model.

In previous studies about the depth-integrated non-hydrostatic model (e.g., *Yamazaki et al.*, 2009; *Stelling and Zijlema*, 2003), a linear vertical distribution of p_{dyn} is simply assumed, which leads to a horizontal momentum equation with $\alpha = 0.5$, $\beta = 1.0$. In fact, non-hydrostatic pressure of moderately dispersive waves varies quadratically instead of linearly along z direction, as is shown by the Boussinesq equations. In this study, we propose a more realistic approximation of p_{dyn} (i.e., a pure quadratic function of $z + h$), and derive a different horizontal momentum equation with $\alpha = 2/3$, $\beta = 0.5$. Such modification slightly improves the accuracy in wave dispersion, and brings it closer the classical Boussinesq equations. In numerical tests, our model performs better for smooth bathymetry than the previous one, though the difference in the leading and the first trailing waves is not significant.

2.2 Governing Equations in Cartesian Coordinates

The governing equations (2-21) together with bottom friction effects in Cartesian coordinates are written as

$$\begin{cases} \dfrac{\partial}{\partial t}(\eta + h) + \dfrac{\partial M}{\partial x} + \dfrac{\partial N}{\partial y} = 0, & \text{(2-22a)} \\ \dfrac{\partial M}{\partial t} + \dfrac{\partial}{\partial x}\left(\dfrac{M^2}{D}\right) + \dfrac{\partial}{\partial y}\left(\dfrac{MN}{D}\right) + gD\dfrac{\partial \eta}{\partial x} = -\alpha\left[D\dfrac{\partial q}{\partial x} + q\dfrac{\partial(\eta - \beta h)}{\partial x}\right] - f_x, & \text{(2-22b)} \\ \dfrac{\partial N}{\partial t} + \dfrac{\partial}{\partial x}\left(\dfrac{MN}{D}\right) + \dfrac{\partial}{\partial y}\left(\dfrac{N^2}{D}\right) + gD\dfrac{\partial \eta}{\partial y} = -\alpha\left[D\dfrac{\partial q}{\partial y} + q\dfrac{\partial(\eta - \beta h)}{\partial y}\right] - f_y, & \text{(2-22c)} \\ \dfrac{\partial W}{\partial t} = \dfrac{q}{D}, & \text{(2-22d)} \end{cases}$$

in which η is the free surface elevation, (M, N) denote the volume fluxes in x and y directions respectively, $D = \eta + h$ is the total water depth, q is defined as the non-hydrostatic pressure at the bottom, W is the depth-averaged vertical velocity equal to the average of w at the surface and

the seabed given by (2-3a, 2-3c), and (f_x, f_y) are the bottom friction in x and y directions. The bottom friction is evaluated via Manning's formula, that is

$$\begin{cases} f_x = \dfrac{gn^2}{D^{7/3}} M(M^2+N^2)^{1/2}, \\ f_y = \dfrac{gn^2}{D^{7/3}} N(M^2+N^2)^{1/2}, \end{cases} \tag{2-23}$$

where n is the Manning's roughness coefficient which is usually set to ~ 0.03.

2.3 Governing Equations in Earth Spherical Coordinates

On the surface of Earth, all the differential operators should be expressed in the spherical coordinates defined by (R, x, y), where R is the constant Earth radius, x and y are the longitude and latitude, respectively. By rewriting all the terms in spherical coordinates and considering the influence of Earth rotation, the governing equations on Earth surface are

$$\begin{cases} \dfrac{\partial}{\partial t}(\eta + h) + \dfrac{1}{R\cos y}\left[\dfrac{\partial M}{\partial x} + \dfrac{\partial}{\partial y}(N\cos y)\right] = 0, & \text{(2-24a)} \\ \dfrac{\partial M}{\partial t} + \dfrac{1}{R\cos y}\dfrac{\partial}{\partial x}\left(\dfrac{M^2}{D}\right) + \dfrac{1}{R}\dfrac{\partial}{\partial y}\left(\dfrac{MN}{D}\right) - \dfrac{\sin y}{R\cos y}\dfrac{2MN}{D} + \dfrac{gD}{R\cos y}\dfrac{\partial \eta}{\partial x} \\ \quad = -\alpha\left[\dfrac{D}{R\cos y}\dfrac{\partial q}{\partial x} + \dfrac{q}{R\cos y}\dfrac{\partial(\eta - \beta h)}{\partial x}\right] - f_x + 2N\Omega\sin y, & \text{(2-24b)} \\ \dfrac{\partial N}{\partial t} + \dfrac{1}{R\cos y}\dfrac{\partial}{\partial x}\left(\dfrac{MN}{D}\right) + \dfrac{1}{R}\dfrac{\partial}{\partial y}\left(\dfrac{N^2}{D}\right) + \dfrac{\sin y}{R\cos y}\dfrac{M^2 - N^2}{D} + \dfrac{gD}{R}\dfrac{\partial \eta}{\partial y} \\ \quad = -\alpha\left[\dfrac{D}{R}\dfrac{\partial q}{\partial y} + \dfrac{q}{R}\dfrac{\partial(\eta - \beta h)}{\partial y}\right] - f_y - 2M\Omega\sin y, & \text{(2-24c)} \\ \dfrac{\partial W}{\partial t} = \dfrac{q}{D}, & \text{(2-24d)} \end{cases}$$

where Ω is the angular velocity of Earth rotation, and the corresponding terms represent Coriolis forces.

3 Computational Method

The governing equations of our depth-integrated non-hydrostatic model are solved with a semi-implicit staggered finite-difference scheme. In such method, the shallow water equations without the non-hydrostatic pressure are firstly solved explicitly to give an intermediate result. Then, the non-hydrostatic pressure is implicitly solved from a Poisson-type equation. Finally, the shallow water solution is corrected by the non-hydrostatic pressure, and the dispersive result is obtained. Besides, a moving boundary technique is adopted to compute inundation, and an empirical eddy-viscosity scheme is used for wave breaking. Nested grids are employed to simulate tsunamis in different scales, and both one-way and two-way nesting algorithms are provided. All the numerical schemes are parallelized for a CPU cluster and a GPU. For the CPU version, we assign the computational task to multiple CPU cores using space domain decomposition, with inter-core communication facilitated by the MPI library. For the GPU version, we adopt the CUDA FORTRAN programming model and map the CUDA threads to the entire simulation domain.

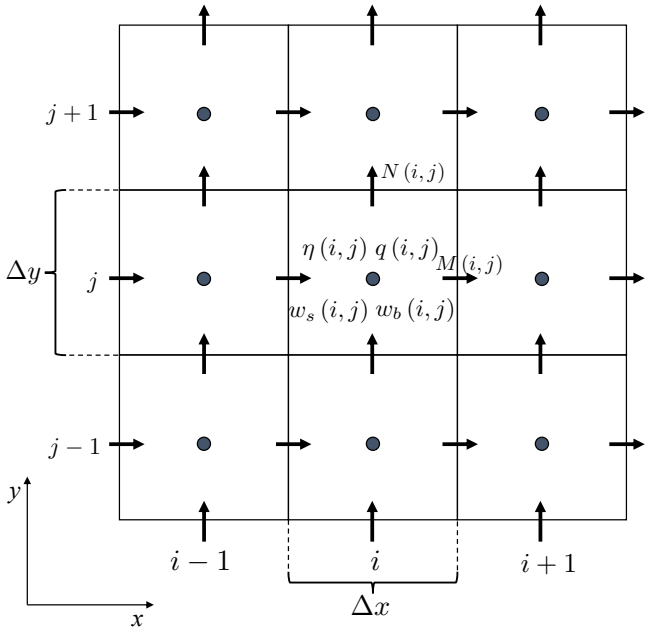

Figure 3.1 Staggered grid setup in PCOMCOT. The surface elevation η, non-hydrostatic pressure q, the vertical velocity at the free surface w_s and that at the bottom w_b are defined at the center of each grid cell. While the volume flux components M and N are defined at the cell interfaces.

3.1 Solution of Shallow Water Equations

The shallow water equations in the Cartesian or Earth coordinates comprise the continuity equation (2-22a) or (2-24a), and the horizontal momentum equations (2-22b, 2-22c) or (2-24b, 2-24c), with the non-hydrostatic terms neglected. By further ignoring the nonlinear terms in the momentum equations, the linear shallow water equations are obtained. Both the linear and nonlinear shallow water equations are solved explicitly in the staggered grids shown in Figure 3.1, where the water depth $h_{i,j}$ and surface elevation $\eta_{i,j}$ are defined at the center of grid cell (i,j), while the volume fluxes $M_{i,j}$ and $N_{i,j}$ are evaluated at the cell interfaces $(i+1/2,j)$ and $(i,j+1/2)$, respectively.

Most tsunami models including COMCOT (*Wang and Liu*, 2006; *Liu et al.*, 1998), solve the linear shallow water equations with a forward time-centered space (FTCS) scheme. The FTCS scheme is simple and efficient, but has some stability problems at low friction, which usually causes spurious oscillations near coastlines (*de Almeida et al.*, 2012; *Bates et al.*, 2010). *de Almeida et al.* (2012) have demonstrated that these instabilities originate from the lack of diffusive terms in the modified equations for FTCS scheme, and proposed a "q-centered" scheme (referred to as "flux-centered" in this paper) to address this problem. The flux-centered scheme introduces artificial diffusive terms without changing the equations to be solved, and thus is adopted in PCOMCOT for improving stability when solving the SWEs. This scheme is the same as the FTCS method, except that each flux variable at the previous time step is replaced by the weighted average of itself and the neighboring ones.

In Cartesian coordinates, the linear shallow water equations are discretized as

$$\begin{cases} \eta_{i,j}^{n+1} = \eta_{i,j}^{n} - \left(h_{i,j}^{n+1} - h_{i,j}^{n}\right) - \dfrac{\Delta t}{\Delta x}\left(M_{i,j}^{n} - M_{i-1,j}^{n}\right) - \dfrac{\Delta t}{\Delta y}\left(N_{i,j}^{n} - N_{i,j-1}^{n}\right), \\ M_{i,j}^{n+1} = \theta M_{i,j}^{n} + \dfrac{1-\theta}{2}\left(M_{i-1,j}^{n} + M_{i+1,j}^{n}\right) - gD_{i+1/2,j}^{n+1}\dfrac{\Delta t}{\Delta x}\left(\eta_{i+1,j}^{n+1} - \eta_{i,j}^{n+1}\right) - f_x \Delta t, \\ N_{i,j}^{n+1} = \theta N_{i,j}^{n} + \dfrac{1-\theta}{2}\left(N_{i,j-1}^{n} + N_{i,j+1}^{n}\right) - gD_{i,j+1/2}^{n+1}\dfrac{\Delta t}{\Delta y}\left(\eta_{i,j+1}^{n+1} - \eta_{i,j}^{n+1}\right) - f_y \Delta t. \end{cases} \tag{3-1}$$

In Earth coordinates, the discretization is given as

$$
\begin{cases}
\eta_{i,j}^{n+1} = \eta_{i,j}^{n} - \left(h_{i,j}^{n+1} - h_{i,j}^{n}\right) \\
\qquad - \dfrac{1}{R\cos y_j}\left[\dfrac{\Delta t}{\Delta x}\left(M_{i,j}^{n} - M_{i-1,j}^{n}\right) - \dfrac{\Delta t}{\Delta y}\left(N_{i,j}^{n}\cos y_{j+1/2} - N_{i,j-1}^{n}\cos y_{j-1/2}\right)\right], \\
M_{i,j}^{n+1} = \theta M_{i,j}^{n} + \dfrac{1-\theta}{2}\left(M_{i-1,j}^{n} + M_{i+1,j}^{n}\right) - gD_{i+1/2,j}^{n+1}\dfrac{\Delta t}{R\Delta x\cos y_j}\left(\eta_{i+1,j}^{n+1} - \eta_{i,j}^{n+1}\right) \\
\qquad + 2N_{i+1/2,j-1/2}^{n}\Delta t\Omega\sin y_j - f_x\Delta t, \\
N_{i,j}^{n+1} = \theta N_{i,j}^{n} + \dfrac{1-\theta}{2}\left(N_{i,j-1}^{n} + N_{i,j+1}^{n}\right) - gD_{i,j+1/2}^{n+1}\dfrac{\Delta t}{R\Delta y}\left(\eta_{i,j+1}^{n+1} - \eta_{i,j}^{n+1}\right) \\
\qquad - 2M_{i-1/2,j+1/2}^{n}\Delta t\Omega\sin y_{j+1/2} - f_y\Delta t.
\end{cases}
\tag{3-2}
$$

In the above finite difference formulation, Δt denotes the time step size, Δx and Δy the grid sizes in x and y directions, respectively. The parameter θ is the weighting factor of the flux-centered scheme, and can be adjusted between 0 and 1. With $\theta = 0$, the Lax-Wendroff diffusive scheme (*Lax and Wendroff*, 1960) is obtained, while $\theta = 1$ restores the FTCS formulation without numerical diffusion. In PCOMCOT, the value of θ is calculated by an adaptive formulation (*Sridharan et al.*, 2020), which varies in time and space depending on the local velocity and water depth. We find that the flux-centered scheme, together with the adaptive weighting factor, effectively eliminate unphysical oscillations without changing the tsunami waveforms.

The flow depth at cell interfaces is evaluated with the upwind scheme of *Kowalik et al.* (2005), which is

$$
D_{i+1/2,j}^{n+1} = \begin{cases} \eta_{i,j}^{n+1} + \dfrac{h_{i,j}^{n+1} + h_{i+1,j}^{n+1}}{2}, & M_{i,j}^{n} \geq 0, \\ \eta_{i+1,j}^{n+1} + \dfrac{h_{i,j}^{n+1} + h_{i+1,j}^{n+1}}{2}, & M_{i,j}^{n} < 0, \end{cases}
\tag{3-3a}
$$

$$
D_{i,j+1/2}^{n+1} = \begin{cases} \eta_{i,j}^{n+1} + \dfrac{h_{i,j}^{n+1} + h_{i,j+1}^{n+1}}{2}, & N_{i,j}^{n} \geq 0, \\ \eta_{i,j+1}^{n+1} + \dfrac{h_{i,j}^{n+1} + h_{i,j+1}^{n+1}}{2}, & N_{i,j}^{n} < 0. \end{cases}
\tag{3-3b}
$$

The bottom friction terms are calculated as

$$\begin{cases} f_x = \dfrac{gn^2}{\left(D^{n+1}_{i+1/2,j}\right)^{7/3}} M^n_{i,j} \left[\left(M^n_{i,j}\right)^2 + \left(N^n_{i+1/2,j-1/2}\right)^2\right]^{1/2}, \\ f_y = \dfrac{gn^2}{\left(D^{n+1}_{i,j+1/2}\right)^{7/3}} N^n_{i,j} \left[\left(M^n_{i-1/2,j+1/2}\right)^2 + \left(N^n_{i,j}\right)^2\right]^{1/2}. \end{cases} \tag{3-4}$$

And the fluxes not originally defined in the staggered grids are estimated with the average of existing ones around them.

$$\begin{cases} M_{i-1/2,j+1/2} = \dfrac{1}{4}\Big(M_{i-1,j} + M_{i,j} + M_{i-1,j+1} + M_{i,j+1}\Big), \\ N_{i+1/2,j-1/2} = \dfrac{1}{4}\Big(N_{i,j-1} + N_{i+1,j-1} + N_{i,j} + N_{i+1,j}\Big). \end{cases} \tag{3-5}$$

At each time step, the surface elevation η is firstly calculated from the continuity equation, and then the fluxes M and N are updated by the momentum equation.

For nonlinear shallow water equations, the nonlinear convection terms are discretized with an upwind scheme and then added to the solution of linear momentum equations.

In Cartesian coordinates, the nonlinear momentum equations are discretized as

$$\begin{cases} M^{n+1}_{i,j} = linear\ terms - \dfrac{\Delta t}{\Delta x}\Big(MU_2 - MU_1\Big) - \dfrac{\Delta t}{\Delta y}\Big(NU_2 - NU_1\Big), \\ N^{n+1}_{i,j} = linear\ terms - \dfrac{\Delta t}{\Delta x}\Big(MV_2 - MV_1\Big) - \dfrac{\Delta t}{\Delta y}\Big(NV_2 - NV_1\Big). \end{cases} \tag{3-6}$$

In Earth coordinates, the discretization is expressed as

$$\begin{cases} M^{n+1}_{i,j} = linear\ terms - \dfrac{\Delta t}{R\Delta x \cos y_j}\Big(MU_2 - MU_1\Big) - \dfrac{\Delta t}{R\Delta y}\Big(NU_2 - NU_1\Big) \\ \qquad + \dfrac{\Delta t \sin y_j}{R\cos y_j} \dfrac{2M^n_{i,j}N^n_{i+1/2,j-1/2}}{D^{n+1}_{i+1/2,j}}, \\ N^{n+1}_{i,j} = linear\ terms - \dfrac{\Delta t}{R\Delta x \cos y_{j+1/2}}\Big(MV_2 - MV_1\Big) - \dfrac{\Delta t}{R\Delta y}\Big(NV_2 - NV_1\Big) \\ \qquad - \dfrac{\Delta t \sin y_{j+1/2}}{R\cos y_{j+1/2}} \dfrac{\left(M^n_{i-1/2,j+1/2}\right)^2 - \left(N^n_{i,j}\right)^2}{D^{n+1}_{i,j+1/2}}. \end{cases} \tag{3-7}$$

Here *linear terms* represent the terms on the right-hand side of (3-1) or (3-2), and the nonlinear

terms in parentheses are evaluated as follows.

$$
\begin{cases}
MU_2 - MU_1 = \begin{cases} \dfrac{\left(M^n_{i,j}\right)^2}{D^{n+1}_{i+1/2,j}} - \dfrac{\left(M^n_{i-1,j}\right)^2}{D^{n+1}_{i-1/2,j}}, & M^n_{i,j} \geq 0, \\ \dfrac{\left(M^n_{i+1,j}\right)^2}{D^{n+1}_{i+3/2,j}} - \dfrac{\left(M^n_{i,j}\right)^2}{D^{n+1}_{i+1/2,j}}, & M^n_{i,j} < 0, \end{cases} & \text{(3-8a)} \\
NU_2 - NU_1 = \begin{cases} \dfrac{M^n_{i,j}N^n_{i+1/2,j-1/2}}{D^{n+1}_{i+1/2,j}} - \dfrac{M^n_{i,j-1}N^n_{i+1/2,j-3/2}}{D^{n+1}_{i+1/2,j-1}}, & N^n_{i+1/2,j-1/2} \geq 0, \\ \dfrac{M^n_{i,j+1}N^n_{i+1/2,j+1/2}}{D^{n+1}_{i+1/2,j+1}} - \dfrac{M^n_{i,j}N^n_{i+1/2,j-1/2}}{D^{n+1}_{i+1/2,j}}, & N^n_{i+1/2,j-1/2} < 0, \end{cases} & \text{(3-8b)} \\
MV_2 - MV_1 = \begin{cases} \dfrac{M^n_{i-1/2,j+1/2}N^n_{i,j}}{D^{n+1}_{i,j+1/2}} - \dfrac{M^n_{i-3/2,j+1/2}N^n_{i-1,j}}{D^{n+1}_{i-1,j+1/2}}, & M^n_{i-1/2,j+1/2} \geq 0, \\ \dfrac{M^n_{i+1/2,j+1/2}N^n_{i+1,j}}{D^{n+1}_{i+1,j+1/2}} - \dfrac{M^n_{i-1/2,j+1/2}N^n_{i,j}}{D^{n+1}_{i,j+1/2}}, & M^n_{i-1/2,j+1/2} < 0, \end{cases} & \text{(3-8c)} \\
NV_2 - NV_1 = \begin{cases} \dfrac{\left(N^n_{i,j}\right)^2}{D^{n+1}_{i,j+1/2}} - \dfrac{\left(N^n_{i,j-1}\right)^2}{D^{n+1}_{i,j-1/2}}, & N^n_{i,j} \geq 0, \\ \dfrac{\left(N^n_{i,j+1}\right)^2}{D^{n+1}_{i,j+3/2}} - \dfrac{\left(N^n_{i,j}\right)^2}{D^{n+1}_{i,j+1/2}}, & N^n_{i,j} < 0. \end{cases} & \text{(3-8d)}
\end{cases}
$$

3.2 Solution of Non-hydrostatic Model

The non-hydrostatic pressure can be implicitly calculated using the solution of shallow water equations, and then used to correct the hydrostatic solution. To distinguish the intermediate hydrostatic result from the final non-hydrostatic solution, we use $\widetilde{\mathbf{F}}(\widetilde{M}, \widetilde{N})$ and $\widetilde{\overline{\mathbf{u}}}(\widetilde{\overline{u}}, \widetilde{\overline{v}})$ to represent the horizontal volume flux and the depth-averaged horizontal velocity from the shallow water equations, while $\mathbf{F}(M, N)$ and $\overline{\mathbf{u}}(\overline{u}, \overline{v})$ for the final non-hydrostatic counterparts. To construct the implicit equation of non-hydrostatic pressure, we express the horizontal and vertical velocities as functions of q, and substitute these functions into a rewritten form of the continuity equation (2-21a). The resulting Poisson-type equation is solved efficiently with the ILU-preconditioned Bi-CGSTAB method.

First, we rewrite the continuity equation (2-21a) with horizontal and vertical velocities. Again,

this is done in dimensionless form, to clearly show that the $O(\mu^2)$ accuracy in wave dispersion is retained. According to (2-10a), by neglecting $O(\mu^4)$ terms, the dimensionless kinematic boundary conditions (2-7a, 2-7c) can be written as

$$\begin{cases} w_s = \mu^2 \left(\dfrac{\partial \eta}{\partial t} + \varepsilon \overline{\mathbf{u}} \cdot \nabla \eta \right), \\ w_b = -\mu^2 \left(\dfrac{\partial h_b}{\partial t} + \overline{\mathbf{u}} \cdot \nabla h \right). \end{cases} \tag{3-9}$$

in which w_s and w_b are the vertical velocities at the free surface and the bottom, respectively. Substituting (3-9) into (2-8), the following relation is obtained.

$$\frac{\partial}{\partial t}(\eta + h_b) + \nabla \cdot \left[(h + \varepsilon \eta) \overline{\mathbf{u}} \right] = (h + \varepsilon \eta) \nabla \cdot \overline{\mathbf{u}} + \frac{w_s - w_b}{\mu^2} = 0. \tag{3-10}$$

Thus, the 2D continuity equation can be rewritten as

$$\mu^2 \nabla \cdot \overline{\mathbf{u}} + \frac{w_s - w_b}{h + \varepsilon \eta} = 0, \tag{3-11}$$

and its dimensional form is

$$\nabla \cdot \overline{\mathbf{u}} + \frac{w_s - w_b}{D} = 0. \tag{3-12}$$

Note that in the above analysis, we have not introduced any extra equation other than the governing equations (2-21), because the kinematic boundary conditions have been used for deriving the original continuity equation (2-21a). The rewritten form of continuity equation in (3-12) is valid to $O(\mu^2)$, while the original one is an exact relation.

Then, we express horizontal velocities with q, and substitute them into (3-12). According to equation (2-21b), at the $(n+1)$-th time step, the dispersive volume flux can be expressed as

$$\mathbf{F}^{n+1} = \widetilde{\mathbf{F}}^{n+1} - \alpha \Delta t \left[D^{n+1} \nabla q + q \nabla \left(\eta - \beta h \right)^{n+1} \right], \tag{3-13}$$

where the superscript $(n+1)$ of q is dropped for simplicity. Dividing both sides of (3-13) with the total water depth, the dispersive depth-averaged horizontal velocity is

$$\overline{\mathbf{u}}^{n+1} = \widetilde{\overline{\mathbf{u}}}^{n+1} - \alpha \Delta t \left[\nabla q + q \frac{\nabla \left(\eta - \beta h \right)^{n+1}}{D^{n+1}} \right]. \tag{3-14}$$

Since w varies linearly in z direction for moderately dispersive flows, the vertical velocity at the

free surface can be determined from the vertical momentum equation (2-21c) as

$$w_s^{n+1} = w_s^n + w_b^n - w_b^{n+1} + \frac{2\Delta t}{D^{n+1}} q, \tag{3-15}$$

where the bottom vertical velocity w_b is estimated from the kinematic boundary condition (2-3c). By substituting (3-14) and (3-15) into (3-12), we obtain the Poisson-type equation of non-hydrostatic pressure, which is

$$-\alpha\Delta t \left\{ \nabla^2 q + \nabla \cdot \left[q \frac{\nabla\left(\eta - \beta h\right)^{n+1}}{D^{n+1}} \right] \right\} + \frac{2\Delta t}{\left(D^{n+1}\right)^2} q = -\nabla \cdot \widetilde{\overline{\mathbf{u}}}^{n+1} - \frac{w_s^n + w_b^n - 2w_b^{n+1}}{D^{n+1}}. \tag{3-16}$$

Discretization of (3-16) in the staggered grids shown in Figure 3.1 yields the linear algebraic equation system of q, that is

$$a1_{i,j} q_{i-1,j} + a2_{i,j} q_{i+1,j} + a3_{i,j} q_{i,j-1} + a4_{i,j} q_{i,j+1} + a5_{i,j} q_{i,j} = b_{i,j}. \tag{3-17}$$

In Cartesian coordinates, the coefficients and forcing terms in (3-17) are expressed as

$$\begin{cases} a1_{i,j} = \dfrac{\alpha\Delta t}{(\Delta x)^2}\left(-1 + \dfrac{1}{2}\varphi_{i-1,j}\right), \quad a2_{i,j} = \dfrac{\alpha\Delta t}{(\Delta x)^2}\left(-1 - \dfrac{1}{2}\varphi_{i,j}\right), \\ a3_{i,j} = \dfrac{\alpha\Delta t}{(\Delta y)^2}\left(-1 + \dfrac{1}{2}\psi_{i,j-1}\right), \quad a4_{i,j} = \dfrac{\alpha\Delta t}{(\Delta y)^2}\left(-1 - \dfrac{1}{2}\psi_{i,j}\right), \\ a5_{i,j} = \dfrac{\alpha\Delta t}{(\Delta x)^2}\left[\left(1 + \dfrac{1}{2}\varphi_{i-1,j}\right) + \left(1 - \dfrac{1}{2}\varphi_{i,j}\right)\right] \\ \qquad + \dfrac{\alpha\Delta t}{(\Delta y)^2}\left[\left(1 + \dfrac{1}{2}\psi_{i,j-1}\right) + \left(1 - \dfrac{1}{2}\psi_{i,j}\right)\right] + \dfrac{2\Delta t}{\left(D_{i,j}^{n+1}\right)^2}, \\ b_{i,j} = -\dfrac{\widetilde{\overline{u}}_{i,j}^{n+1} - \widetilde{\overline{u}}_{i-1,j}^{n+1}}{\Delta x} - \dfrac{\widetilde{\overline{v}}_{i,j}^{n+1} - \widetilde{\overline{v}}_{i,j-1}^{n+1}}{\Delta y} - \dfrac{w_{s\,i,j}^n + w_{b\,i,j}^n - 2w_{b\,i,j}^{n+1}}{D_{i,j}^{n+1}}, \end{cases} \tag{3-18}$$

in which w_b, the vertical velocity at the bottom, is calculated with an upwind scheme as

$$\begin{aligned} w_{b\,i,j}^{n+1} = &-\frac{h_{i,j}^{n+1} - h_{i,j}^n}{\Delta t} - \frac{\overline{u}_{i-1/2,j}^n}{\Delta x} \times \begin{cases} h_{i,j}^{n+1} - h_{i-1,j}^{n+1}, & \overline{u}_{i-1/2,j}^n \geq 0, \\ h_{i+1,j}^{n+1} - h_{i,j}^{n+1}, & \overline{u}_{i-1/2,j}^n < 0, \end{cases} \\ &- \frac{\overline{v}_{i,j-1/2}^n}{\Delta y} \times \begin{cases} h_{i,j}^{n+1} - h_{i,j-1}^{n+1}, & \overline{v}_{i,j-1/2}^n \geq 0, \\ h_{i,j+1}^{n+1} - h_{i,j}^{n+1}, & \overline{v}_{i,j-1/2}^n < 0. \end{cases} \end{aligned} \tag{3-19}$$

In Earth coordinates, these terms are

$$\begin{cases} a1_{i,j} = \dfrac{\alpha\Delta t}{(R\Delta x\cos y_j)^2}\big(-1+\dfrac{1}{2}\varphi_{i-1,j}\big), \quad a2_{i,j} = \dfrac{\alpha\Delta t}{(R\Delta x\cos y_j)^2}\big(-1-\dfrac{1}{2}\varphi_{i,j}\big), \\ a3_{i,j} = \dfrac{\alpha\Delta t\cos y_{j-1/2}}{(R\Delta y)^2\cos y_j}\big(-1+\dfrac{1}{2}\psi_{i,j-1}\big), \quad a4_{i,j} = \dfrac{\alpha\Delta t\cos y_{j+1/2}}{(R\Delta y)^2\cos y_j}\big(-1-\dfrac{1}{2}\psi_{i,j}\big), \\ a5_{i,j} = \dfrac{2\Delta t}{\big(D_{i,j}^{n+1}\big)^2} + \dfrac{\alpha\Delta t}{(R\Delta x\cos y_j)^2}\left[\big(1+\dfrac{1}{2}\varphi_{i-1,j}\big)+\big(1-\dfrac{1}{2}\varphi_{i,j}\big)\right] \\ \qquad + \dfrac{\alpha\Delta t}{(R\Delta y)^2\cos y_j}\left[\big(1+\dfrac{1}{2}\psi_{i,j-1}\big)\cos y_{j-1/2}+\big(1-\dfrac{1}{2}\psi_{i,j}\big)\cos y_{j+1/2}\right], \\ b_{i,j} = -\dfrac{\widetilde{\overline{u}}_{i,j}^{n+1}-\widetilde{\overline{u}}_{i-1,j}^{n+1}}{R\Delta x\cos y_j} - \dfrac{\widetilde{\overline{v}}_{i,j}^{n+1}\cos y_{j+1/2}-\widetilde{\overline{v}}_{i,j-1}^{n+1}\cos y_{j-1/2}}{R\Delta y\cos y_j} - \dfrac{w_{s\,i,j}^n + w_{b\,i,j}^n - 2w_{b\,i,j}^{n+1}}{D_{i,j}^{n+1}}, \end{cases} \tag{3-20}$$

where w_b is

$$\begin{aligned} w_{b\,i,j}^{n+1} = -\frac{h_{i,j}^{n+1}-h_{i,j}^n}{\Delta t} &- \frac{\overline{u}_{i-1/2,j}^n}{R\Delta x\cos y_j}\times\begin{cases} h_{i,j}^{n+1}-h_{i-1,j}^{n+1}, & \overline{u}_{i-1/2,j}^n \ge 0, \\ h_{i+1,j}^{n+1}-h_{i,j}^{n+1}, & \overline{u}_{i-1/2,j}^n < 0, \end{cases} \\ &- \frac{\overline{v}_{i,j-1/2}^n}{R\Delta y}\times\begin{cases} h_{i,j}^{n+1}-h_{i,j-1}^{n+1}, & \overline{v}_{i,j-1/2}^n \ge 0, \\ h_{i,j+1}^{n+1}-h_{i,j}^{n+1}, & \overline{v}_{i,j-1/2}^n < 0. \end{cases} \end{aligned} \tag{3-21}$$

In both Cartesian and Earth coordinates, the variables φ and ψ are defined as

$$\begin{cases} \varphi_{i,j} = \dfrac{\big(\eta-\beta h\big)_{i+1,j}^{n+1}-\big(\eta-\beta h\big)_{i,j}^{n+1}}{D_{i+1/2,j}^{n+1}}, \\ \psi_{i,j} = \dfrac{\big(\eta-\beta h\big)_{i,j+1}^{n+1}-\big(\eta-\beta h\big)_{i,j}^{n+1}}{D_{i,j+1/2}^{n+1}}. \end{cases} \tag{3-22}$$

The depth-averaged horizontal velocity in the above equations is simply the horizontal volume flux divided by the total water depth, that is

$$\begin{cases} \widetilde{\overline{u}}_{i,j}^{n+1} = \dfrac{\widetilde{M}_{i,j}^{n+1}}{D_{i+1/2,j}^{n+1}}, \quad \widetilde{\overline{v}}_{i,j}^{n+1} = \dfrac{\widetilde{N}_{i,j}^{n+1}}{D_{i,j+1/2}^{n+1}}, \\ \overline{u}_{i-1/2,j}^n = \dfrac{M_{i-1,j}^n + M_{i,j}^n}{2D_{i,j}^n}, \quad \overline{v}_{i,j-1/2}^n = \dfrac{N_{i,j-1}^n + N_{i,j}^n}{2D_{i,j}^n}. \end{cases} \tag{3-23}$$

The sparse linear system of non-hydrostatic pressure (3-17) is solved with the bi-conjugate gradient squared stabilized (Bi-CGSTAB) method (*van der Vorst*, 1992), and the incomplete LU (ILU) preconditioner (*Barrett et al.*, 1994) is used to speed up the convergence. The non-hydrostatic pressure at the outermost cells of the computational domain is set to be zero as the boundary condition. After solving the non-hydrostatic pressure q, the vertical velocity at the free surface w_s is calculated from (3-15), and the final dispersive volume flux (M, N) is obtained by substituting q into (3-13).

In Cartesian coordinates, the discretized value of the final dispersive volume fluxes are

$$\begin{cases} M_{i,j}^{n+1} = \widetilde{M}_{i,j}^{n+1} - \dfrac{\alpha\Delta t}{\Delta x} D_{i+1/2,j}^{n+1}\left(q_{i+1,j} - q_{i,j}\right) \\ \qquad\qquad - \dfrac{\alpha\Delta t}{2\Delta x}\left(q_{i,j} + q_{i+1,j}\right)\left[\left(\eta - \beta h\right)_{i+1,j}^{n+1} - \left(\eta - \beta h\right)_{i,j}^{n+1}\right], \\ N_{i,j}^{n+1} = \widetilde{N}_{i,j}^{n+1} - \dfrac{\alpha\Delta t}{\Delta y} D_{i,j+1/2}^{n+1}\left(q_{i,j+1} - q_{i,j}\right) \\ \qquad\qquad - \dfrac{\alpha\Delta t}{2\Delta y}\left(q_{i,j} + q_{i,j+1}\right)\left[\left(\eta - \beta h\right)_{i,j+1}^{n+1} - \left(\eta - \beta h\right)_{i,j}^{n+1}\right]. \end{cases} \tag{3-24}$$

In Earth coordinates, they are given as

$$\begin{cases} M_{i,j}^{n+1} = \widetilde{M}_{i,j}^{n+1} - \dfrac{\alpha\Delta t}{R\Delta x\cos y_j} D_{i+1/2,j}^{n+1}\left(q_{i+1,j} - q_{i,j}\right) \\ \qquad\qquad - \dfrac{\alpha\Delta t}{2R\Delta x\cos y_j}\left(q_{i,j} + q_{i+1,j}\right)\left[\left(\eta - \beta h\right)_{i+1,j}^{n+1} - \left(\eta - \beta h\right)_{i,j}^{n+1}\right], \\ N_{i,j}^{n+1} = \widetilde{N}_{i,j}^{n+1} - \dfrac{\alpha\Delta t}{R\Delta y} D_{i,j+1/2}^{n+1}\left(q_{i,j+1} - q_{i,j}\right) \\ \qquad\qquad - \dfrac{\alpha\Delta t}{2R\Delta y}\left(q_{i,j} + q_{i,j+1}\right)\left[\left(\eta - \beta h\right)_{i,j+1}^{n+1} - \left(\eta - \beta h\right)_{i,j}^{n+1}\right]. \end{cases} \tag{3-25}$$

It should be noted that the depth-integrated non-hydrostatic model is only applicable to moderate bottom slope, which is the same as the Boussinesq equations. It has been demonstrated that localized steep bottom gradients generally lead to instability of the Boussinesq-type models (*Løvholt and Pedersen*, 2009). Similarly, the solution to the linear system (3-17) becomes unstable near steep bathymetric features. For simulation of dispersive tsunamis, large bottom slope should be avoided.

3.3 Moving Boundary Technique

In numerical computation of tsunami inundation, the real topography is represented with a series of small steps located at the finite difference grids. At a submerged wet cell, the surface elevation η is above the bottom and the total water depth $D = \eta + h$ is positive. While at a dry cell, the surface elevation is below the bottom and the value of η depends on the location of the cell. If the dry cell is in the sea ($h \geq 0$), η is set to be $-h$ so that the total water depth D is zero. But if it is on the land ($h < 0$), η is assigned to be zero and D is negative. In PCOMCOT, we adopt the algorithm of *Cho and Kim* (2009) to constrain the flow direction on the wet-dry boundary according to the local topography and the surface elevation. By doing so, the moving shoreline is captured naturally from the continuity equation. This moving boundary algorithm can stably compute tsunami run-up and run-down on topographic discontinuities without special treatment of the governing equations. For simplicity, different cases in x directions involved in the wet and dry problem is presented in Figure 3.2, and the cases along y direction are handled in the same way. The various cases in terms of flow direction in Figure 3.2 are described below in detail.

(a) If the higher cell is wet ($D_{i+1} > 0$) and the water surface at the lower cell is above the bottom of the higher one ($\eta_i + h_{i+1} > 0$), then the water can flow in both directions and the flux M_i is directly calculated from the momentum equation.

(b) If the higher cell is dry ($D_{i+1} \leq 0$) and the water surface at the lower cell is above the bottom of the higher one ($\eta_i + h_{i+1} > 0$), then the water can only flow from the lower cell to the higher one. The value of M_i from the momentum equation is retained if its sign is consistent with the allowed direction, and is set to be zero if not. $-h_{i+1}$ is used as the value of η_{i+1} when calculating M_i to avoid false surface gradient due to the setting of η value on dry cells.

(c) If the higher cell is wet ($D_{i+1} > 0$) and the water surface at the lower cell is below the bottom of the higher one ($\eta_i + h_{i+1} \leq 0$), then the water can only flow from the higher cell to the lower one. The value of M_i is set to be zero if its sign from momentum equation is opposite to the allowed direction. $-h_{i+1}$ is used as the value of η_i when calculating M_i, because the water level at the lower cell would not influence the flow from above.

(d) If the higher cell is dry ($D_{i+1} \leq 0$) and the water surface at the lower cell is below the bottom of the higher one ($\eta_i + h_{i+1} \leq 0$), then the water cannot flow between these two cells and M_i

becomes zero.

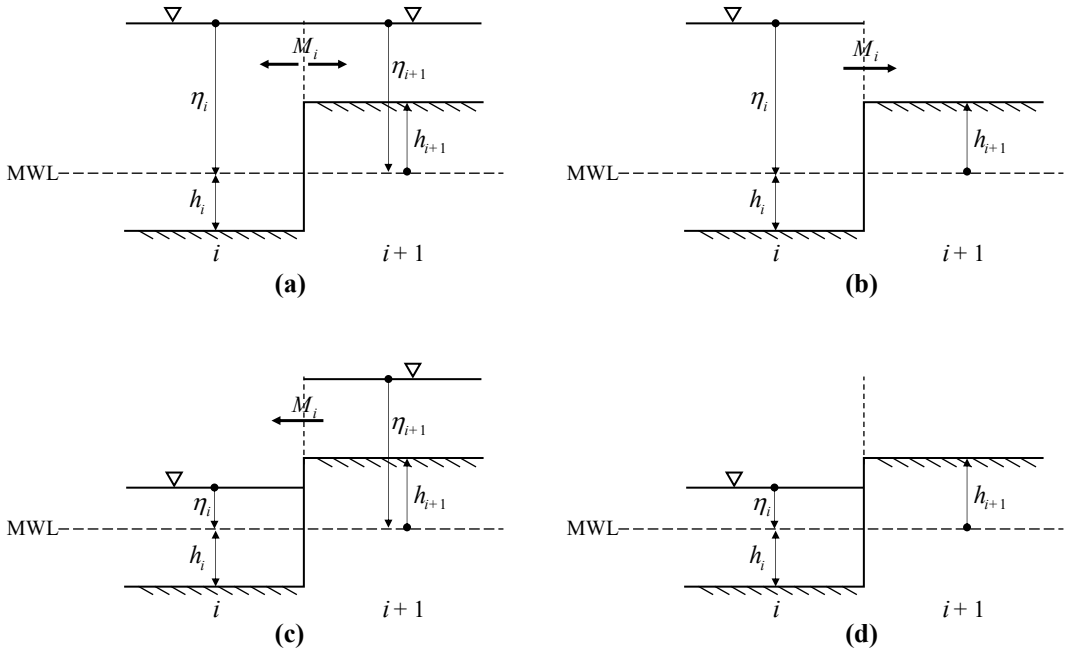

Figure 3.2 1D wet-dry cases in x direction. MWL represents the mean water level, and the arrows below the flux M denote the allowed directions of water flow.

The momentum equations together with the above constraints on flow direction determine the volume flux on the wet-dry boundary. Once the continuity equation is solved at the next time step based on the value of volume flux, the conversion between wet and dry is handled as following.

* If $\left(\eta+h\right)^n \leq 0$ and $\left(\eta+h\right)^{n+1} > \left(\eta+h\right)^n$, then water flows into the dry cell. Because there is no water in the cell at the previous time step, the total water depth D^{n+1} is actually $\left(\eta+h\right)^{n+1} - \left(\eta+h\right)^n$. Thus, the value of η^{n+1} is modified to be $\eta^{n+1} - \left(\eta+h\right)^n$ to give the correct total water depth.

* If $\left(\eta+h\right)^n > 0$ and $\left(\eta+h\right)^{n+1} < \left(\eta+h\right)^n$, then water flows out of the wet cell. If $\left(\eta+h\right)^{n+1}$ is less than a threshold, this cell is drained, and η^{n+1} is set to 0 when $h^{n+1} < 0$ or $-h^{n+1}$ when $h^{n+1} \geq 0$. Otherwise, the cell remains wet and η^{n+1} from continuity equation is retained. Note that we turn a wet cell into dry when the total water depth is less than a positive threshold rather than zero, to avoid the instability caused by a very thin water film when the

tsunami recedes.

3.4 Wave Breaking

Since the momentum equations of depth-integrated non-hydrostatic model is only valid for non-breaking waves, additional treatment is needed to handle wave breaking. In PCOMCOT, we use the eddy-viscosity scheme of *Kennedy et al.* (2000) to model the energy dissipation caused by breaking. This scheme adds artificial eddy viscosity terms to the right-hand side of the horizontal momentum equations (2-22b, 2-22c) or (2-24b, 2-24c). These terms are expressed as

$$\begin{cases} R_{bx} = \nabla \cdot (\nu \nabla M), \\ R_{by} = \nabla \cdot (\nu \nabla N), \end{cases} \tag{3-26}$$

where R_{bx} and R_{by} are the eddy viscosity terms for x and y directions, respectively. Compared with the original form of *Kennedy et al.* (2000), The cross-derivatives are removed in (3-26) to improve numerical stability (*Choi et al.*, 2018). ν is the artificial eddy viscosity varying with time and space, which is defined as

$$\nu = B\delta_b^2(h+\eta)\eta_t, \tag{3-27}$$

where the mixing length coefficient δ_b is 1.0 in PCOMCOT. The coefficient B varies smoothly from 0 to 1 to avoid a sudden start of breaking and the related instability. Details about how the value of B is evaluated can be found in the studies of *Kennedy et al.* (2000) and *Chen et al.* (2000). In brief, B is a piecewise linear function of the ratio between η_t and a parameter η_t^* which determines the onset and cessation of breaking. A breaking event starts when η_t exceeds an initial threshold $\eta_t^{(I)}$ and terminates once η_t becomes less than η_t^*. As breaking develops, the parameter η_t^* decreases gradually with the breaking event age from $\eta_t^{(I)}$ to $\eta_t^{(F)}$. To estimate the age of each breaking event in the computational region, the breaking history at a certain grid cell is tracked against the local wave celerity.

3.5 Nested Grid Configuration

A system of nested rectangular grids is employed in PCOMCOT to use various spatial resolutions for different study regions. The largest grid layer covering the entire computational region is called the 1st-level layer (top layer), and the layers directly nested in the 1st-level layer are 2nd-level

layers, and so on. For any two directly nested layers, the outer layer is called the parent layer and the inner one the child layer. Any grid size ratio between the parent and child layers is allowed. Both one-way and two-way nesting algorithms can be used for exchanging the information between two nested layers. In one-way nesting, interpolation of the solution over the parent layer provides the input boundary conditions for the child layer. While the two-way nesting involves an extra procedure of feeding back the surface elevation from the child to the parent layer.

The feed-forward from the parent to child layer is illustrated in Figure 3.3. For the time step size Δt given to the top layer, the time step size of a certain layer is set to be $\Delta t/k$, where k is the smallest integer which gives a C.F.L. (Courant-Friedrichs-Lewy) number no larger than that of the top layer. Beginning at time t, the parent layer is firstly computed with the time step size $\Delta t/k_1$ until $t + \Delta t$. The surface elevation η, volume flux (M, N), non-hydrostatic pressure q and artificial eddy viscosity ν at $t + \Delta t$ over the parent layer are interpolated to the grid cells on the boundary of the child layer. Then, the child layer is calculated with the time step size $\Delta t/k_2$, and at each step, the solution on the child layer boundary is predefined by assuming a linear change of these variables from t to $t + \Delta t$. Note that for calculation of nonlinear terms, two rows/columns of variables on the child layer boundary need to be interpolated from the parent layer. In Figure 3.3, only one row/column is plotted for simplicity.

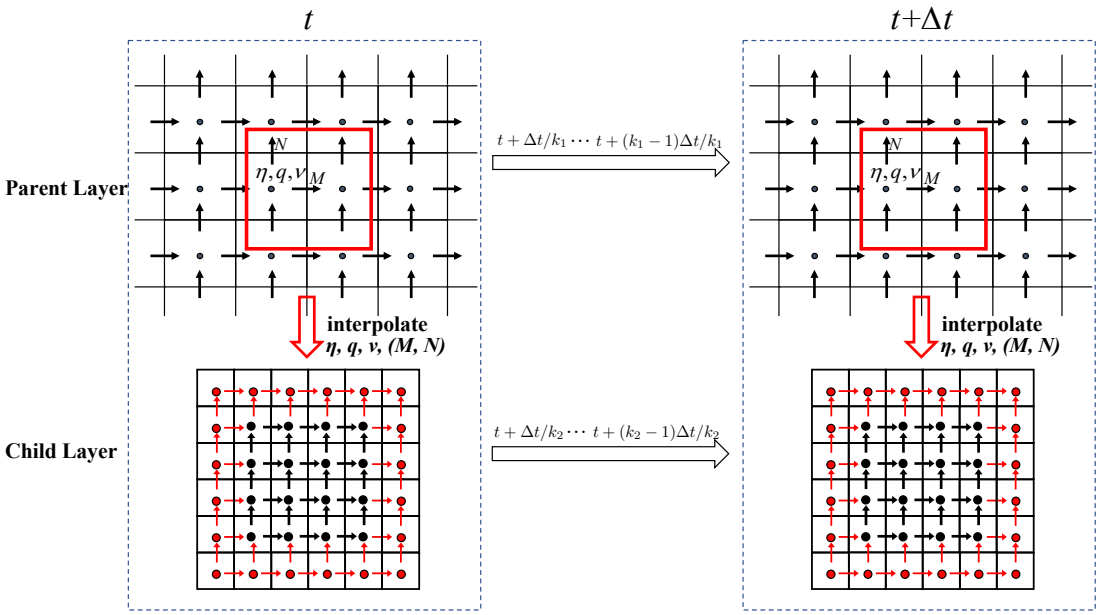

Figure 3.3 Feed-forward from the parent to the child layer. The red box in the parent layer denotes the boundary of the child layer. The red dots and arrows in the child layer represent η, q, ν and (M, N) interpolated from the parent layer, while the black dots and arrows indicate those obtained by solving the governing equations.

The feedback from the child to parent layer is sketched in Figure 3.4. If two-way nesting is used, the surface elevation over the parent layer is updated with the average value from the overlapping child layer grids when the computation in all layers reaches $t + \Delta t$. After the feedback of surface elevation, the momentum equations are solved again in the parent layer to keep (M, N), q, ν coupled with η.

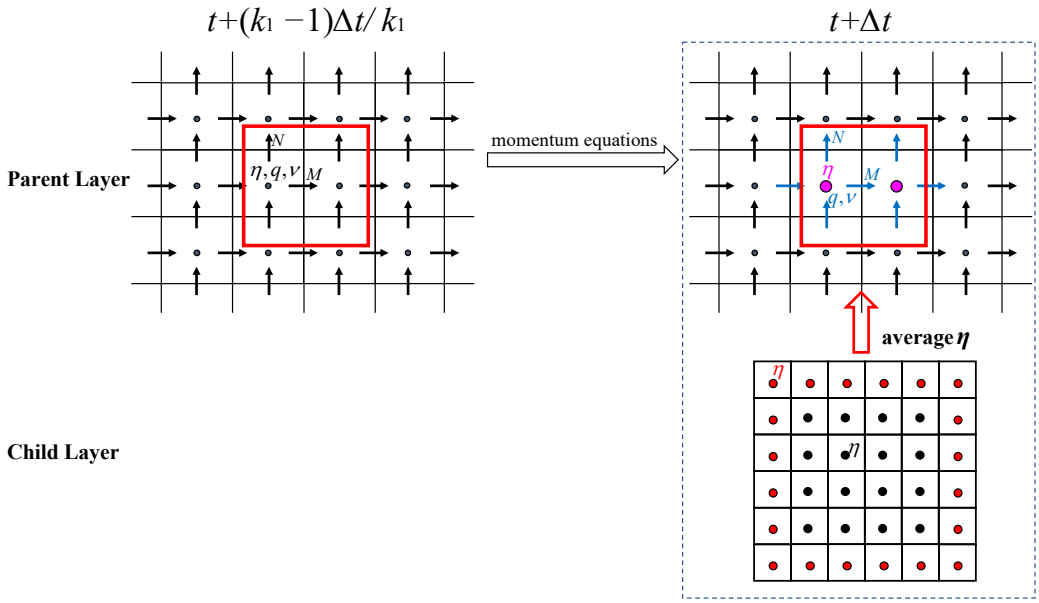

Figure 3.4 Feedback from the child to the parent layer. The red and black dots in the child layer denote the surface elevation interpolated from the parent layer and calculated from the continuity equation, respectively. In the parent layer, the magenta dots indicate the grid points where η is updated with the average value from the child layer. After feedback of η, the related q, ν and (M, N) which are colored blue are updated by solving the momentum equations again.

3.6 Parallel Implementation

3.6.1 CPU Parallelization

We use the domain decomposition technique for parallel implementation of PCOMCOT on multiple CPU cores. As shown in Figure 3.5, the domain of each grid layer is partitioned into multiple rectangular subdomains, each of which has almost the same dimension for load balance. These subdomains are assigned to different computing nodes, and data exchanges between adjacent nodes are realized with the MPI point-point communication routines. For a nested grid system, both intra- and inter-layer communication are needed. The intra-layer communication is performed between subdomains of a single layer, while the inter-layer communication transfers the data between two layers which are directly nested. To solve the Poisson-type equation of non-hydrostatic pressure, both the Bi-CGSTAB algorithm and the ILU-preconditioning are parallelized based on the same strategies of domain decomposition and data exchange.

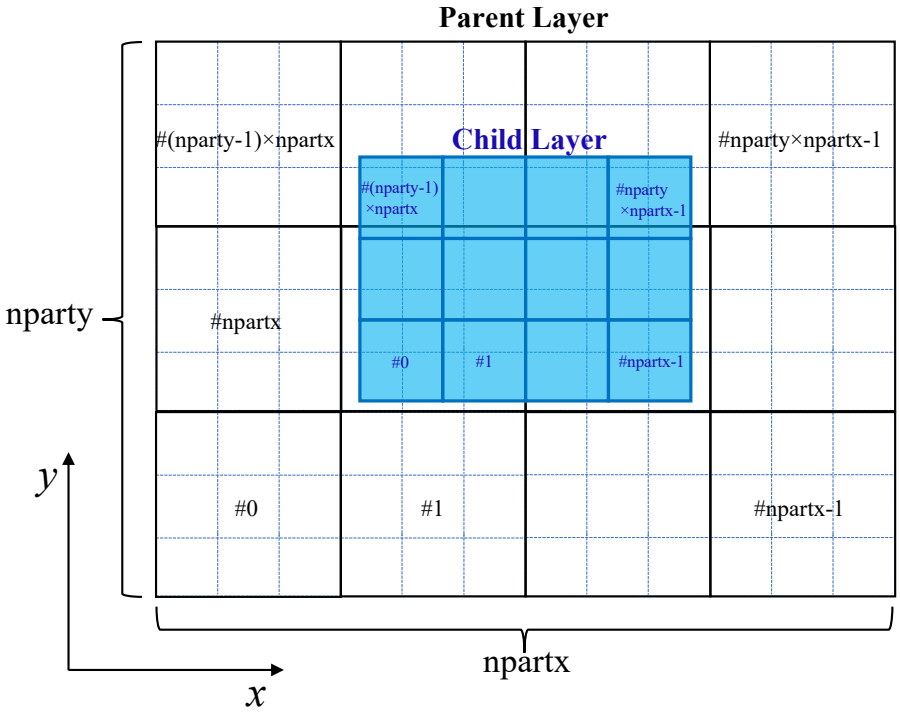

Figure 3.5 Domain decomposition for parallel computation. The subdomains of each layer are assigned to different nodes. The Numbers after # are the MPI ranks of these nodes, starting from 0. The total number of subdomains along x and y directions are npartx and nparty, respectively.

The intra-layer communication is illustrated in Figure 3.6. To calculate the variables on the boundary of a subdomain (i.e., η, q, ν, M and N), two extra rows/columns of required data are transferred from each of the four neighboring subdomains. For demonstration purpose, only one row/column is plotted here. It should be noted that, to correctly synchronize the variables around the four corners of each subdomain, data transfer needs to be performed in order. That is, each computing node must receive data from its left and right neighbors, and then from the top and bottom ones, or vice versa.

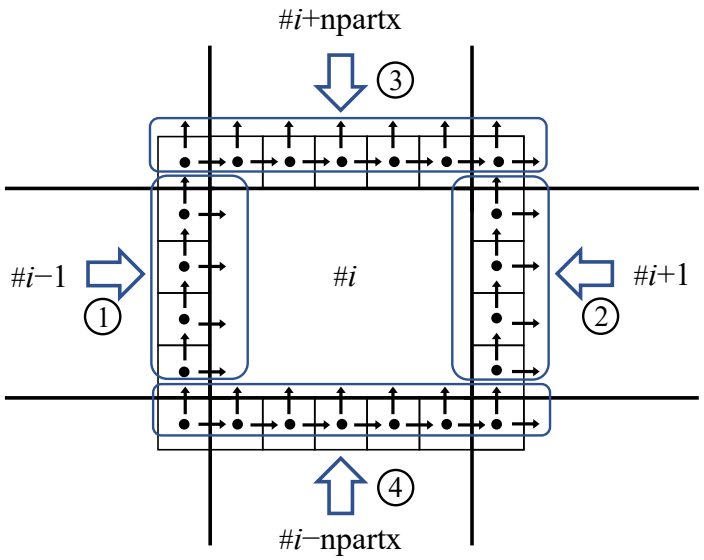

Figure 3.6 Intra-layer communication. Boundaries of subdomains are denoted by thick black lines. The dots and arrows represent the variables η, q, ν, M and N sent to node $\#i$ at each time step. The blue rounded boxes together with the large hollow arrows indicate from which nodes these variables are sent, with the circled numbers showing the order of data transfer.

Figure 3.7 displays how the inter-layer communication is done. Suppose that a grid cell on the child layer boundary is assigned to node #0, but its location in the parent layer belongs to the subdomain assigned to node #5. So, in the feed-forward from the parent to child layer, values of η, q, ν, M and N at this location are firstly interpolated on node #5 using the parent layer data. Then, these interpolated values are sent to node #0 and stored at the child layer grid. The similar procedure is done in the feedback from the child to parent layer. Suppose that a grid cell of the parent layer is assigned to node $\#(4+\mathrm{npartx})$, but its location in the child layer is covered by node #0. Thus, η at this location is calculated on node #0 by averaging the surrounding child layer data, and then sent to node $\#(4+\mathrm{npartx})$.

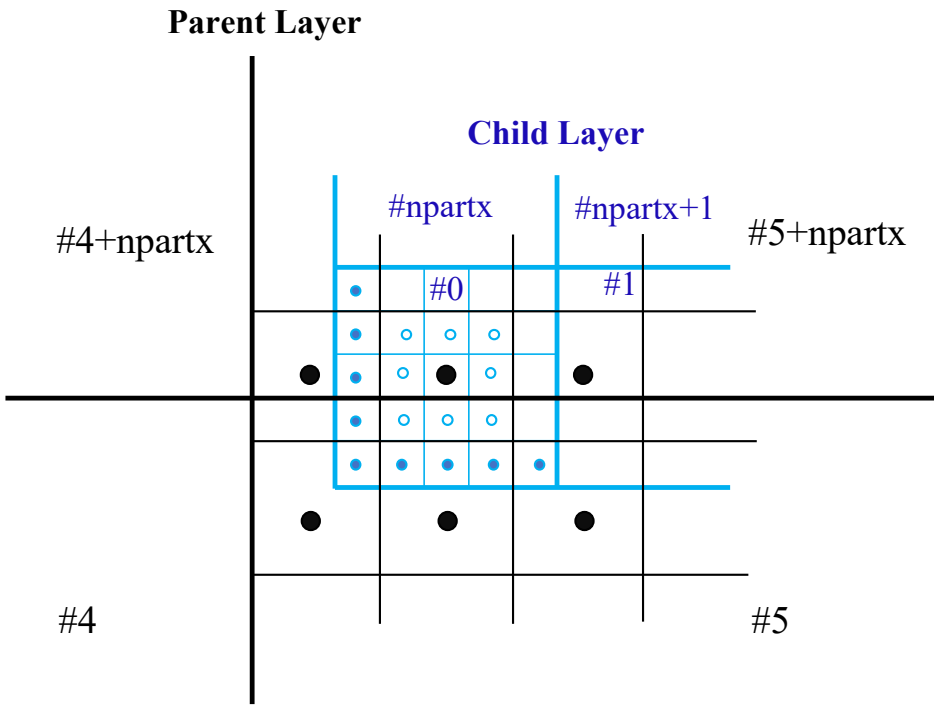

Figure 3.7 Inter-layer communication. Thick black and thick blue lines represent the boundaries of parent layer and child layer subdomains, respectively. In the child layer, the solid blue dots indicate the boundary cells where variable values are interpolated from the parent layer, while the hollow ones are the interior cells where the surface elevation is averaged to update η in the parent layer.

To parallelize the ILU-preconditioned Bi-CGSTAB algorithm (*Barrett et al.*, 1994; *van der Vorst*, 1992), all the involved matrices and vectors are distributed into the same subdomains. In every iteration, matrix and vector elements on the boundaries of subdomains are synchronized via intra-layer communication, and the scalars are updated through MPI collective communication. To enable parallel solving of the triangular preconditioning matrices, any coefficient in (3-17) relating different nodes are omitted when constructing these matrices. For example, if $q_{i,j}$ and $q_{i-1,j}$ are stored on different nodes, the coefficient $a1_{i,j}$ is assumed to be zero. We test our implicit solver against the Matlab function *bicgstab*, which solves sparse linear systems with the unpreconditioned Bi-CGSTAB (Copyright 1984-2022 The MathWorks, Inc.). As shown in Figure 3.8, with the tolerance set to be 10^{-6}, difference between the non-hydrostatic pressure given by the Matlab *bicgstab* and our parallel preconditioned version is less than 5×10^{-7}. The number of iterations in this

example is 6 for PCOMCOT running on 40 CPU cores, and 17 for the MATLAB *bicgstab*.

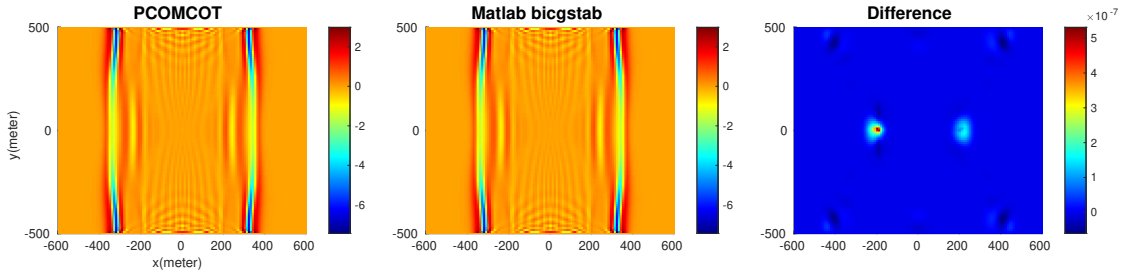

Figure 3.8 Comparison of non-hydrostatic pressure given by PCOMCOT and the Matlab function *bicgstab*.

3.6.2 GPU Parallelization

We employ the CUDA FORTRAN programming model to accelerate PCOMCOT with a NVIDIA GPU. All the computational tasks are offloaded to the GPU through CUDA kernels, i.e., subprograms which run on the device (GPU) but are called from the host (CPU). When a kernel is invoked, it launches a grid of thread blocks, with all the threads executing the same code on different data. This data parallelism is a fine-grained parallelism, which is the most efficient when having adjacent threads operate on adjacent data (*Ruetsch and Fatica*, 2024). Thus, we map the entire simulation domain to a 2D grid of thread blocks, with each thread corresponding to a single cell or cell interface. As shown in Figure 3.9, the built-in variables *BlockIdx*, *BlockDim*, and *ThreadIdx* are used to identify different threads and relate them with the array indices. This mapping strategy heavily oversubscribes GPU cores and thus effectively hides memory latency.

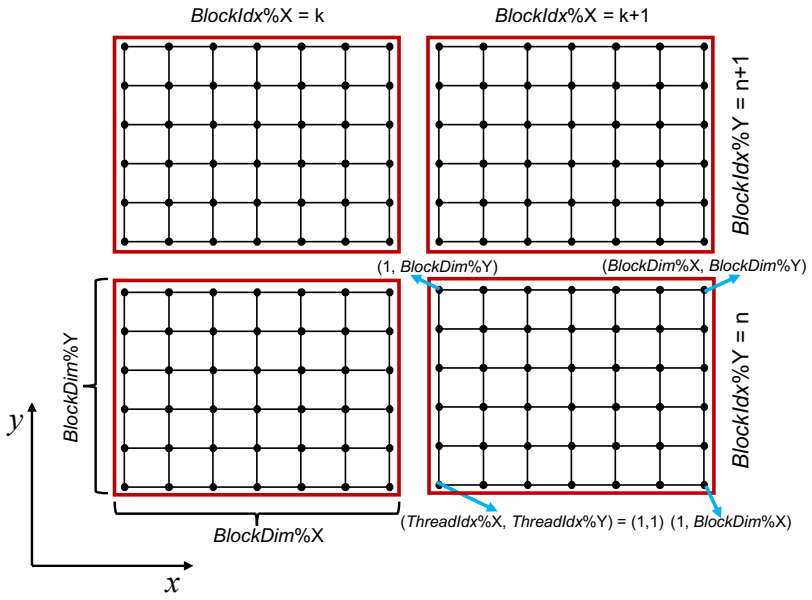

Figure 3.9 Mapping of the simulation domain to a 2D grid of thread blocks.

To efficiently solve the non-hydrostatic pressure on GPU, we utilize NVIDIA's linear algebra libraries, cuBLAS and cuSPARSE, to perform the Bi-CGSTAB method. However, due to the Poisson-type nature of the equation, the ILU preconditioning is basically sequential and not suitable for fine-grained parallelism. As the linear system (3-17) is usually diagonally dominated, we retain only the diagonal coefficients (i.e., $a5_{i,j}$) for preconditioning. Compared with the ILU-preconditioned Bi-CGSTAB, this "D-preconditioned" version requires ∼3 times more iterations for convergence, but is still much faster than the unpreconditioned version. For the case of 2011 Tohoku tsunami, the numbers of iterations required by the ILU-preconditioned, D-preconditioned, and unpreconditioned solvers are around 10, 30 and >400, respectively.

3.7 Boundary Conditions

Two types of boundary conditions – wall boundary and absorbing boundary are provided in PCOM-COT. For absorbing boundary condition, we use a combination of the L-D type sponge layer (*Larsen and Dancy*, 1983) and the friction-type sponge layer. The L-D type sponge layer directly attenuates

the variables near the domain boundary at every time step, i.e.,

$$(\eta, M, N) = (\eta, M, N)/C_s. \tag{3-28}$$

The damping coefficient C_s is defined as

$$C_s = A^{\gamma^{i-1}},\ i = 1, 2, ..., I_{width}, \tag{3-29}$$

in which A and γ are free parameters which are suggested by *Chen et al.* (1999) to be 2.0 and 0.88∼0.92, respectively. i is the grid number along the direction away from the boundary, and I_{width} is the layer width in number of points. A narrow layer (i.e., width ≈ the typical wavelength) can be used for L-D type sponge for its good efficiency. However, *Shi et al.* (2016) pointed out that this type of sponge layer generates sawtooth noises which may grow to be significant in long-term simulation. To minimize the noises and make the damping more effective, the friction-type sponge is also implemented in the same area of the L-D type. The friction-type sponge uses the friction terms in the momentum equations, and thus attenuates the waves smoothly. In the sponge layer, an extra Manning's roughness coefficient n_s, which increases gradually toward the boundary, is added to the original value. The form given by *Shi et al.* (2016) is adopted as the expression of n_s, that is

$$n_s = n_{max}\left(1 - \tanh\frac{10(i-1)}{I_{width}-1}\right), \tag{3-30}$$

where n_{max} is the maximum value of Manning's coefficient in the sponge layer. The combination of L-D and friction-type sponge layers gives satisfactory absorption in our test, without introducing oscillations to the interior of computation domains.

3.8 Dealing with Numerical Instability

The numerical model of PCOMCOT shows good stability in large-scale and long-term simulations. However, numerical instability cannot be avoided completely by any scheme. Here we give some suggestions for dealing with numerical instabilities.

1. Use the flux-centered scheme.

   As mentioned in Section 3.1, the FTCS scheme many models use to solve LSWEs may cause unphysical oscillations near shorelines, due to absence of diffusive terms. The flux-centered scheme reduces these oscillations by introducing slight numerical diffusion. Combined with the

adaptive diffusive coefficient, it generally provides satisfactory stability without overdamping the wavefield. Based on our experience, the flux-centered scheme is always recommended, as long as no overdamping is found.

2. Avoid steep bathymetry.

   The non-hydrostatic model becomes unstable on rapidly varying bathymetry. The instability related with steep bathymetry usually appears as spurious wave sources near the coast. Large gradient of water depth should be avoided when computing dispersive waves. Based on our experiences, a bottom slope less than 0.5 (i.e., $|\nabla h| < 0.5$) generally gives stable results. The *grdfilter* module of GMT (Generic Mapping Tools) software (*Wessel et al.*, 2013) can be used to remove localized steep features. By setting the **-F** option to be **-Fg***width*, we can apply a Gaussian filter to the bathymetry grids, where the parameter *width* is the width of the Gaussian function (i.e., 6 times the standard deviation) in km. According to the stability criterion of *Horrillo et al.* (2006), the parameter *width* should be ∼4.5 times the average water depth.

3. Limit huge velocity with a Froude number cap.

   In tsunami simulations, unrealistic velocity may occur in extremely shallow water. A Froude number cap is used in PCOMCOT to address this problem. In our tests, the Froude number cap is set to be 10.0, and no instability is found. A smaller value can be used to ensure model stability. Note that when calculating inundation, a Froude number cap less than 1.5 is undesirable, because it caps the velocity on the flooding water fronts (*Shi et al.*, 2016).

4. Adjust *Water depth Limit for wet => dry*.

   The parameter *Water depth Limit for wet => dry* controls the threshold value of water depth, below which the grid cell is thought to be dry. The parameter is suggested to be 0.01m for real tsunamis and 0.001m for small-scale lab scenarios. As a thin water film may cause large velocity during the run-down process, users can set larger values for this parameter when instability occurs.

Although no numerical filtering is needed in our tests, we still provide a 2D 9-point numerical filter in case there are other instabilities. The first-order filter of *Shapiro* (1970) can be applied to (η, M, N) every after a given time interval. Note that this numerical filtering is not suggested in general, because it significantly damps the wave field after being used just a few times.

4 Configuration, Input and Output

4.1 Programming Flow

The CPU version of PCOMCOT (PCOMCOT-CPU) is written using Fortran 90 and the MPI library. When running on multiple CPU cores, the computing node with MPI rank 0 is used as the master node, while the others are slave nodes. The input data are read on the master node and then broadcast to slave nodes. Numerical simulation is performed simultaneously on all nodes. For data output, simulation results on slave nodes are sent to the master node, where the output files are generated. Figure 4.1 displays the detailed programming flow of PCOMCOT-CPU.

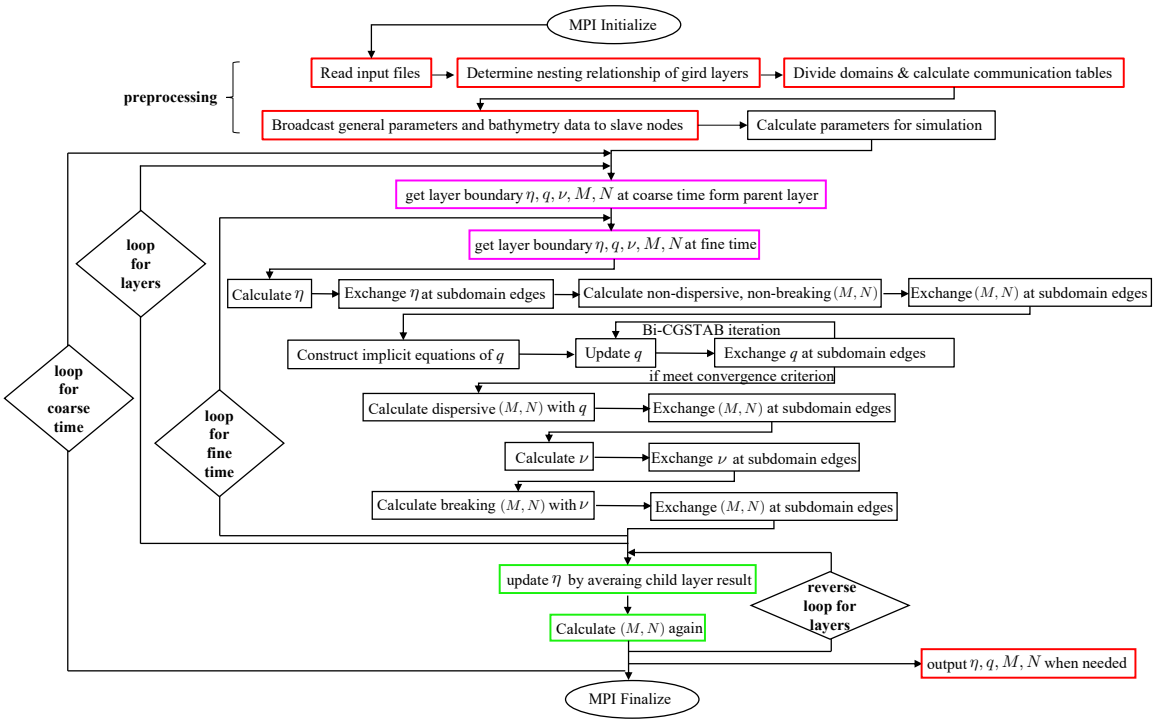

Figure 4.1 Programming flow of PCOMCOT-CPU. The red boxes indicate the steps that are done only on the master node. The magenta and green boxes denote the procedures exclusive to child and parent grid layers, respectively.

The GPU version (PCOMCOT-GPU) is written with CUDA FORTRAN. CUDA FORTRAN is a hybrid programming model, where both the CPU and GPU are utilized. The host CPU is responsible for workflow control and pre-/post-processing, while all the computations are carried

out by the GPU device. Variables are stored in both the host memory and the device global memory, and data synchronization is realized via host-device memory transfer. The programming flow of PCOMCOT-GPU is plotted in Figure 4.2.

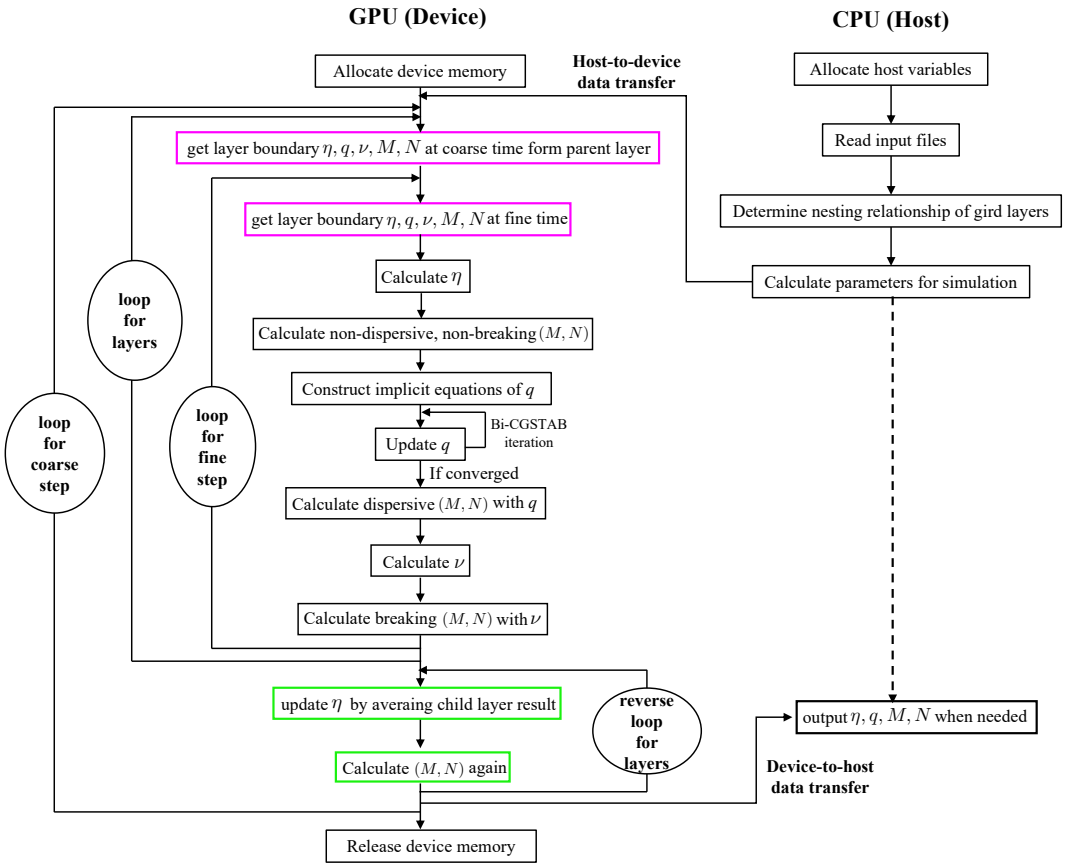

Figure 4.2 Programming flow of PCOMCOT-GPU. The magenta and green boxes denote the procedures exclusive to child and parent grid layers, respectively.

4.2 Compiling Source Files

The CPU and GPU versions of PCOMCOT have different platform requirements and compilation options. Table 4.1 provides an overview of these requirements and options.

Table 4.1 Overview of platform requirements and compilation options for PCOMCOT

Version	CPU	GPU
System	Linux, MacOS	Linux
Hardware Requirements	Intel® CPU	NVIDIA® GPU (Compute Capability $\geqslant$6.0)
Software requirements	Required: gfortran, openMPI Optional: NetCDF-Fortran	NVIDIA® HPC SDK
NetCDF Support	Yes	No
Floating-point Precision	Double	Single or Double
Compilation Options	With NetCDF: make Without NetCDF: make nocdf	Single Precision: make Double Precision: make double

4.2.1 Compilation of CPU Version

For the CPU version of PCOMCOT, all the source files and their descriptions are listed in Table 4.2. To compile these files, **gfortran** – the GNU Fortran compiler, and **OpenMPI** – a widely used message passing interface library must be installed. Besides, **netcdf-fortran**, the netCDF programming interface for Fortran is optional for compilation, which enables PCOMCOT to read input files in netCDF format.

We have compiled and tested PCOMCOT-CPU in MacOS and Linux systems. In MacOS, it is recommended to install the libraries and software mentioned above using a package management tool such as **Homebrew**. In Linux, they can be installed more easily with the built-in package manager **APT**, by executing the following commands in the terminal.

```
$ sudo apt install gcc g++ gfortran
$ sudo apt install openmpi-bin openmpi-doc libopenmpi-dev
$ sudo apt install libnetcdf-dev libnetcdff-dev
```

Table 4.2 Source Files for PCOMCOT-CPU

File Name	Description
pcomcot.f90	main program of PCOMCOT
pcomcotLIB.f90	all subroutines except solving equations
MPICommunicationLIB.f90	subroutines for MPI communication
pcomcotNetCDFlib.f90	subroutines reading netCDF files
pcomcotNetCDFlibEmpty.f90	empty subs for compiling without netCDF library
solveSWEs.f90	subroutines solving shallow water equations
dispersion.f90	subroutines calculating dispersive terms
breaker.f90	subroutines calculating wave breaking
BiCGStabLIB.f90	preconditioned Bi-CGSTAB for solving non-hydrostatic pressure
VariableDefination.f90	definition of structures used in PCOMCOT
okada.f90	Okada's formula for calculating seabed displacement

After getting everything ready, you can compile the source files by running a "make" or "make nocdf" command, depending on whether or not **netcdf-fortran** is available. To use the "make" command, the path of **netcdf-fortran** needs to be specified according to the user's environment. In the makefile, the -I flag specifies the location of the module file "netcdf.mod", and the -L flag indicates the path of the shared library. You can get these paths immediately via a "nf-config" command, after installing **netcdf-fortran** successively.

4.2.2 Compilation of GPU Version

All the source files for the GPU version of PCOMCOT are listed in Table 4.3. These files should be compiled with the CUDA Fortran compiler, which is part of NVIDIA HPC SDK (Software Development Kit), a suite of HPC compilers and libraries for CPU and GPU programming. How to download and install HPC SDK is detailed in this official guide https://docs.nvidia.com/hpc-sdk/hpc-sdk-install-guide/index.html. Note that due to conflict between the CUDA Fortran compiler and the GNU Fortran compiler, NetCDF cannot be supported by the GPU version

of PCOMCOT. Besides, NVIDIA HPC SDK is now only available for Linux system.

Table 4.3 Source Files for PCOMCOT-GPU

File Name	Description
pcomcot.cuf	main program of PCOMCOT (controlled by CPU)
pcomcotLIB.cuf	all CPU subroutines
pcomcotLIB_CUDA.cuf	all GPU subroutines except solving equations
pcomcotNetCDFlibEmpty.cuf	empty CPU subs for compiling without netCDF library
solveSWEs.cuf	GPU subroutines solving shallow water equations
dispersion.cuf	GPU subroutines calculating wave dispersion
breaker.cuf	GPU subroutines calculating wave breaking
VariableDefination.cuf	definition of host and device structures
okada.cuf	GPU subroutines for Okada's formula

After the installation, some environment settings are required to compile PCOMCOT. First, specify the path of the compiler by running

```
$ export NVDIR=/usr/local/nvidia/hpc_sdk
$ export NVARCH=`uname -s`_`uname -m`
$ export SDKVER=24.3
$ export SDKDIR=$NVDIR/$NVARCH/$SDKVER
$ export PATH=$SDKDIR/compilers/bin:$PATH
```

Here `NVDIR` is the installation location of NVIDIA HPC SDK, `NVARCH` is the system's architecture type (e.g., Linux_x86_64), and `SDKVER` is the SDK version number. Users should adjust `NVDIR` and `SDKVER` according to their own situations. Then, because the cuSPARSE library depends on the *nvJitLink* library since CUDA 12.0, we must set the environment variable `LD_LIBRARY_PATH` to include the path where the file "libnvjitlink.so" is located. Users can locate it by searching for the file name in the directory `SDKDIR`. Usually, the path of "libnvjitlink.so" can be added to `LD_LIBRARY_PATH` by running

```
$ export CUDAVER=12.3
$ export LD_LIBRARY_PATH=$SDKDIR/cuda/$CUDAVER/targets/$NVARCH/lib:$LD_LIBRARY_PATH
```

where `CUDAVER` is the cuda toolchain version included in the SDK and should be adjusted by users.

Finally, you can finish the compilation with a "make" or "make double" command, for single- and double-precision computing, respectively. We provide these two options because there is only a small number of double-precision arithmetic units in NVIDIA's consumer GPUs (e.g., GeForce and Quadro series). These GPUs can only be used for single-precision simulations. Double-precision computing is supported by NVIDIA's data center GPUs (Tesla series), but the double-precision performance is generally half of single-precision performance.

4.3 Input

There are eight input files in total for PCOMCOT, some of which are required, and some are optional. All the input files are listed in Table 4.4, along with their descriptions. These input files can be classified into three types according to their usage — control files, bathymetry files and initial condition files. More detailed instructions on how to prepare these files are given in the following subsections.

Table 4.4 Input Files for PCOMCOT

Input File Name	Description
pcomcot.ctl	basic information for simulation; required
layers.ctl	layer-specific parameters; optional
layerXX.xyz/nf	bathymetry data files; required
InitialElevation.xyz/nf	initial surface elevation; required when *Initial Condition*=0
InitialFluxM.xyz	initial volume flux in x direction; optional when *Initial Condition*=0
InitialFluxN.xyz	initial volume flux in y direction; optional when *Initial Condition*=0
FaultParameters.ctl	finite fault parameters; required when *Initial Condition*=1 or *Purpose of Calculation*=2
Stations.ctl	coordinates of output stations; required

Control files of PCOMCOT include "pcomcot.ctl", "layers.ctl" and "Stations.ctl", which are used to set the parameters for governing equations, numerical schemes, and the input/output data.

"pcomcot.ctl" is the most basic control file providing necessary information for simulation. "layers.ctl" prescribes the simulation parameters specific to each grid layer, and its parameters are just part of those in "pcomcot.ctl". If "layers.ctl" is not given by the user, the values in "pcomcot.ctl" will be used as the global values for all layers. "Stations.ctl" is required to give the locations of stations where time histories of variables are output.

The bathymetry files "layerXX.xyz/nf" store the gridded data of water depth, each of which corresponds to a single layer in the nesting system. Users can provide bathymetry files in both netCDF format with the extension ".nf" or in three-column ASCII format with the extension ".xyz".

The initial condition files "InitialElevation.xyz/nf", "InitialFluxM.xyz", "InitialFluxN.xyz", and "FaultParameters.ctl" provide the initial condition of computation. For simulating the propagation of water surface disturbance or general water waves, "InitialElevation.xyz/nf", "InitialFluxM.xyz" and "InitialFluxN.xyz" give the initial values of surface elevation and horizontal volume fluxes. For simulating earthquake tsunamis, "FaultParameters.ctl" provides the parameters of finite faults as the tsunami source.

Note that any input file in netCDF format cannot be used for the GPU version of PCOMCOT.

4.3.1 Parameters in pcomcot.ctl

The control file "pcomcot.ctl" is required to provide necessary information for PCOMCOT model. Following are descriptions of parameters in "pcomcot.ctl".

Basic Control Parameters

Purpose of Calculation: 1 – forward simulation of tsunami waves; 2 – calculate Green's functions of finite faults; 3 – calculate Green's functions of initial surface elevation.

Initial Condition: 0 – initial water surface elevation (required) and volume flux (optional) are given as initial condition, to simulate transient wave propagation; 1 – finite fault parameters are given as initial condition, to simulate earthquake tsunamis.

Coordinate System: 0 – spherical coordinates on Earth surface; 1 – Cartesian coordinates.

Total run time: total simulation time in seconds.

Time step: time step size in seconds for the top grid layer. Time step sizes for the other layers are determined automatically.

Time interval to Save Snapshots: time interval in seconds to save snapshots of the wavefield during simulation.

Save Flux: 1 – output snapshots and time histories of both surface elevation η and volume flux (M, N); 0 – only η is output.

Save Non-hydrostatic Pressure: 1 – output snapshots and time histories of non-hydrostatic pressure at the bottom; 0 – non-hydrostatic pressure is not output. Note that the non-hydrostatic pressure can be output only when wave dispersion is computed.

Minimum grids on each computing node: minimum number of grid cells assigned to each computing node when running on multiple CPU cores; makes no sense for GPU version.

Feedback to parent layer: options for different grid nesting algorithms. 0 – one-way nesting from parent to child layers; 1 – two-way nesting including feedback from child to parent layers. Note that we have not tested the two-way nesting yet.

Parameters for Surface Deformation

Consider Horizontal Motion: 1 – when calculating sea surface deformation caused by earthquakes, consider the contribution of horizontal motion to seafloor uplift using the equation of *Tanioka and Satake* (1996); 0 – do not consider horizontal motion.

Apply Kajiura filter: 1 – apply Kajiura filter to the initial surface deformation, which accounts for the low-pass filtering effect of sea water; 0 – do not apply Kajiura filter.

Use average depth for Kajiura: 1 – use the average water depth of the source region when applying Kajiura filter; 0 – use the varying local depth. We suggest setting this parameter to be 1, because using varying local depth may lead to oversmoothing near the trench.

Water depth Limit for Kajiura: the minimum water depth in meters for applying Kajiura filter. The low-pass filtering effect of sea water should be ignored in shallow water.

Parameters for Wave Physics and Numerics

Nonlinearity: parameter for wave nonlinearity. 0 – no wave nonlinearity; 1 – inclusion of nonlinear convection terms.

Dispersion: parameter for wave dispersion. 0 – no wave dispersion; 1 – inclusion of dispersive terms.

Depth change for Dispersion: parameter for the steepness of bathymetry. 0 – smooth bottom; 1 – steep bottom. For smooth bottom, the new governing equations (2-21) with $\alpha = 2/3$, $\beta = 0.5$ are used. While for steep bottom, the previous model with $\alpha = 0.5$, $\beta = 1.0$ (*Stelling and Zijlema*, 2003; *Yamazaki et al.*, 2009; *Zijlema and Stelling*, 2008) is adopted. In general, setting this parameter to be 0 provides slightly more accurate results.

Water depth Limit for Dispersion: minimum water depth in meters for calculating wave dispersion. Dispersive effects are ignored in very shallow water for both efficiency and stability.

Breaking: parameter for wave breaking; 0 – no wave breaking; 1 – inclusion of wave breaking using eddy-viscosity scheme.

Scheme for LSWEs: options for the numerical scheme to solve the shallow water equations. 0 – traditional forward time-centered space (FTCS) scheme without numerical diffusion; 1 – flux-centered scheme with extra numerical diffusion. The flux-centered scheme is recommended for better stability.

Froude number Cap to Limit velocity: the maximum value of Froude number to avoid unrealistic velocity. A Froude number cap = 10.0 is used in our tests. Smaller value can be used, but for inundation calculation, a Froude number cap less than 1.5 is not suggested.

Time interval to apply Filter: time interval in seconds to apply a 2D 9-point filter to remove 2-grid-wavelength components. It is not suggested to use this filter in general. Turn off the filter by setting this parameter to be longer than the total simulation time.

Parameters for Boundary Condition

Boundary Condition: parameter for boundary condition type; 1 – wall; 2 – sponge.

Width of Sponge (West-East): width of sponge layer (m) on the west and east boundaries.

Width of Sponge (South-North): width of sponge layer (m) on the south and north boundaries.

Maximum Manning coefficient in Sponge: maximum Manning's roughness coefficient in sponge layer.

Damping coefficient A: the free parameter A in equation (3-29) for L-D type sponge, its value is $\sim$2.0.

Damping coefficient R: the free parameter γ in equation (3-29) for L-D type sponge, its value is 0.88$\sim$0.92.

Parameters for Inundation

Permanent dry limit: maximum elevation (m) above which the grid cell is treated as permanently dry and excluded from calculation.

Water depth Limit for wet -> dry: the minimum water depth (m) for the wetting and drying scheme. 0.01 is suggested for real tsunamis, and larger value can be used to ensure stability.

Water depth limit for Bottom friction: minimum depth (m) for calculating bottom friction. 0.05 is suggested for real tsunamis.

Manning coefficient for Bottom friction: Manning roughness coefficient in Manning's formula. Its value is $\sim$0.03 in most cases.

Parameters for Computing Green's Functions (only when *Purpose of Calculation* = 3)

Source Area Starting Longitude: west longitude (degree) of tsunami source region.

Source Area Ending Longitude: east longitude (degree) of tsunami source region.

Source Area Discretizatoin Grid Size X: size (m) of sub-source region in west-east direction.

Source Area Starting Latitude: south latitude (degree) of tsunami source region.

Source Area Ending Latitude: north latitude (degree) of tsunami source region.

Source Area Discretizatoin Grid Size Y: size (m) of sub-source region in south-north direction.

Basis Function Type: type of point source function. 1 – Gaussian function; 2 – Sigmoid function.

Ratio of Gaussian Raidus / Grid Size: ratio of Gaussian function radius to the sub-source region size. This parameter determines the width of the point source.

Sigmoid Coefficient: coefficient defining the steepness of Sigmoid function.

4.3.2 Parameters in layers.ctl

The file "layers.ctl" is not required but optional. With this input file, PCOMCOT is more flexible for balance between accuracy and efficiency. In "layers.ctl", the name of each layer is a two-digital integer consistent with the corresponding bathymetry file. Figure 4.3 shows the 6 layer-specific parameters in "layers.ctl". By setting these parameters, we can adopt different wave models for different grid layers. For example, we can simulate linear dispersive waves in the large outer layers, where wave dispersion may be significant in long-distance propagation, and switch to nonlinear shallow water model in the small inner layers covering coastal areas. Besides, we can prescribe that computation does not start in a certain layer until tsunamis approach its boundary from outside, by setting the value of *Computing Start Time*. Note that not all layers must appear in "layers.ctl". If a layer is absent from "layers.ctl", its parameters are taken to be same as those in "pcomcot.ctl".

```
################################################
#                                              #
#  Layer Control file for PCOMCOT program (v2.1)   #
#                                              #
################################################
#=============================================:============================
#                   Layer Name                :     01                     |
#=============================================:============================
 Nonlinearity     (0:no/linear; 1:yes/nonlinear):      0
 Dispersion                    (0:no; 1:yes):      1
 Depth change for Dispersion (0:smooth; 1:steep):      0
 Breaking                      (0:no; 1:yes):      0
 Scheme for LSWEs      (0:FTCS; 1:flux-centered):      0
 Computing Start Time                (second):      0.0
 Time interval to Filter WaveField      (second):  50000.0

#=============================================:============================
#                   Layer Name                :     02                     |
#=============================================:============================
 Nonlinearity     (0:no/linear; 1:yes/nonlinear):      1
 Dispersion                    (0:no; 1:yes):      0
 Depth change for Dispersion (0:smooth; 1:steep):      0
 Breaking                      (0:no; 1:yes):      0
 Scheme for LSWEs      (0:FTCS; 1:flux-centered):      0
 Computing Start Time                (second):      0.0
 Time interval to Filter WaveField      (second):  50000.0
```

Figure 4.3 Parameters in layers.ctl

4.3.3 Format of Bathymetry Files

The input bathymetry files must be named as "layerXX.nf" or "layerXX.xyz", where "XX" is a two digital number from 01 to 99 and does not have to be continuous. Each bathymetry file corresponds to a single grid layer in the nesting system. All files must have the format of NetCDF (binary, ending with ".nf") or "xyz" (ASCII, ending with ".xyz"). "nf" files can be obtained from commonly used bathymetry data sets provided by different organizations (e.g., GEBCO2020) with

GMT commands (e.g., "grdcut" and "grdsample"); An "xyz" file contains three columns, with the first column indicating x coordinates, the second for y coordinates, and the last for water depth in meters. x and y coordinates should be given in meters for Cartesian coordinates and in degrees for Earth coordinates. Data in "xyz" files must be sorted by x ascending and then y ascending order, e.g.:

(Line 1:)	138.000000000	33.000000000	4033.7974
(Line 2:)	138.016666667	33.000000000	4016.6523
(Line 3:)	138.033333333	33.000000000	4005.2083
(Lines..)	...		
(Line 842:)	138.000000000	33.016666667	4054.6309
(Lines..)	...		

We can simply convert "nf" files into "xyz" files using the GMT command

```
$ gmt grd2xyz layerXX.nf | awk '{print $1,$2,-$3}' | sort -n -s -k 2 > layerXX.xyz
```

Note that "xyz" files should have positive water depth, while "nf" files should have negative water depth. For the setting of nested grids, there are two general rules. First, only one top layer is allowed; Second, one layer can contain others but cannot overlap another one without containing. There are no constraints on the location and grid size of any layer, but child layers with larger grid sizes than parent layers have not been tested.

4.3.4 Format of Initial Elevation and Flux Files

If the parameter *Initial Condition* is set to be 0 in "pcomcot.ctl", an initial elevation file named "InitialElevation.nf" or "InitialElevation.xyz" must be provided. "InitialElevation.nf/xyz" must have the same format as described above, with exactly the same x and y coordinates as the bathymetry file of the top layer. Two extra input files "InitialFluxM.xyz" and "InitialFluxN.xyz" can also be used to provide the initial volume fluxes in x and y directions, respectively. The formats of "InitialFluxM.xyz" and "InitialFluxN.xyz" are the same as "InitialElevation.xyz", but the x and y coordinates are different. Pay attention that, because PCOMCOT uses staggered grids, η, M and

N are defined on different locations. Suppose that the total number of grid cells in a certain layer are NX and NY along x and y directions, respectively. Then the dimensions of η, M and N are NX $\times$ NY, (NX $-$ 1) $\times$ NY and NX $\times$ (NY $-$ 1), respectively. Therefore, "InitialFluxM.xyz" must have the same y coordinates as the bathymetry file, but have only (NX $-$ 1) x coordinates which are larger than the corresponding ones in the bathymetry file by $\Delta x/2$. Similarly, "InitialFluxN.xyz" must have the same x coordinates as the bathymetry file of the top layer, but have only (NY $-$ 1) y coordinates which are larger than the corresponding ones in the bathymetry file by $\Delta y/2$.

4.3.5 Parameters in FaultParameters.ctl

If the parameter *Initial Condition* is set to be 1 or *Purpose of Calculation* is set to be 2 in "pcomcot.ctl", then the fault slip of an earthquake is used as the tsunami source, and finite fault parameters must be given in "FaultParameters.ctl". PCOMCOT takes the parameters of multiple rectangular finite faults as input and calculates the static seafloor deformation with Okada's half space model (*Okada*, 1985). Then, the sea surface elevation caused by bottom displacement is used as the initial condition for tsunami simulation. When the bathymetry of the source region is smooth and the water depth is much less than the dimension of bottom deformation, the surface elevation is almost identical to the bottom vertical displacement. However, if the earthquake occurs near a steep slope (e.g., a trench), the horizontal motion may be a non-negligible factor of tsunami generation, and this contribution is considered with the formula of *Tanioka and Satake* (1996). If the earthquake occurs in deep ocean, the surface deformation is smoother than the bottom deformation, because of the low-pass filtering effect of seawater. In PCOMCOT, the series representation of Kajiura filter in space domain is used to account for such effect (*Kajiura*, 1963; *Glimsdal et al.*, 2013). This form of Kajiura filter is equivalent to its original form (i.e., the $1/\cosh(kh)$ filter) in wavenumber domain.

For forward simulation (*Purpose of Calculation* = 1), all the faults with their slip given by the user are adopted in computation. For calculation of Green's functions (*Purpose of Calculation* = 2), the slip of all faults is set to be 1.0 m, and the resulting tsunami of each fault is calculated one by one. An example of "FaultParameters.ctl" is shown in Figure 4.4. Following is the explanation of each parameter.

```
#####################################################
#                                                   #
# MULTI-FAULT CONFIGURATION for PCOMCOT program v2.1  #
#                                                   #
#####################################################
#============================================:============================
# Parameters for Fault Segment 0001          :    Values                  |
#============================================:============================
 Fault Rupture Starting Time          (second):       0.0
 Focal Depth                           (meter):    6100.0
 Length of source area                 (meter):   50000.0
 Width of source area                  (meter):   25757.3
 Dislocation of fault plate            (meter):       0.000
 Rake                                 (degree):      90.0
 Strike                               (degree):     193.0
 Dip                                  (degree):      13.9
 Epicenter: Latitude                  (degree):      39.506
 Epicenter: Longitude                 (degree):     144.092

#============================================:============================
# Parameters for Fault Segment 0002          :    Values                  |
#============================================:============================
 Fault Rupture Starting Time          (second):       0.0
 Focal Depth                           (meter):    6100.0
 Length of source area                 (meter):   50000.0
 Width of source area                  (meter):   25757.3
 Dislocation of fault plate            (meter):       0.691
 Rake                                 (degree):      90.0
 Strike                               (degree):     193.0
 Dip                                  (degree):      13.9
 Epicenter: Latitude                  (degree):      39.069
 Epicenter: Longitude                 (degree):     143.961

#============================================:============================
# Parameters for Fault Segment 0003          :    Values                  |
#============================================:============================
 Fault Rupture Starting Time          (second):       0.0
 Focal Depth                           (meter):    6100.0
 Length of source area                 (meter):   50000.0
 Width of source area                  (meter):   25757.3
 Dislocation of fault plate            (meter):       0.000
 Rake                                 (degree):      90.0
 Strike                               (degree):     193.0
 Dip                                  (degree):      13.9
 Epicenter: Latitude                  (degree):      38.631
 Epicenter: Longitude                 (degree):     143.831
```

Figure 4.4 Finite fault parameters in FaultParameters.ctl

Fault Rupture Starting Time: rupture starting time of fault in seconds. Kinematic rupture process can be considered by assigning different rupture starting time to different subfaults.

Focal Depth: depth (m) of center of the rectangular fault.

Length of source area: length (m) of the rectangular fault (i.e., size along strike direction).

Width of source area: width (m) of the rectangular fault (i.e., size along dip direction).

Dislocation of fault plate: amount of slip (m) on the fault.

Rake: rake angle (degree) of the fault slip.

Strike: strike angle (degree) of the fault.

Dip: dip angle (degree) of the fault.

Epicenter's Latitude: y coordinate of the horizontal projection of the fault center. Its value is in degrees for Earth coordinates and in meters for Cartesian coordinates.

Epicenter's Longitude: x coordinate of the horizontal projection of the fault center. Its value is in degrees for Earth coordinates and in meters for Cartesian coordinates.

4.3.6 Parameters in Stations.ctl

"Stations.ctl" is a three-column ASCII file, with the first two columns indicating the x and y coordinates, and the third column giving the names of the stations, respectively. After simulation is finished, the time series of surface elevation and horizontal volume flux at the location of each station can be output. An example of "Stations.ctl" is presented in Figure 4.5.

```
220.120   53.037      46415
211.453   55.318      46409
207.584   53.766      46414
203.222   52.663      46403
196.057   50.983      46402
```

Figure 4.5 Parameters in Stations.ctl

4.4 Output

All the output files of PCOMCOT are listed in Table 4.5. PCOMCOT creates a directory "PCOMCOToutput" at the start of simulation, and puts the output files in this directory. Three types of data are output by PCOMCOT – coordinate data, wavefield data and station data. The coordinate data are stored in files "_xcoordintesXX.dat" and "_ycoordintesXX.dat", which are the coordinates of grid cells along x and y directions for No.XX layer. The wavefield data are the values of variables related with he wavefield, including bathymetry (_bathymetryXX.dat), snapshots of water surface elevation (z_XX_xxxxxx.dat), volume flux (M_XX_xxxxxx.dat, N_XX_xxxxxx.dat) and non-hydrostatic pressure at the bottom (Q_XX_xxxxxx.dat), and the maximum/minimum water surface elevation (zmax_XX.dat, zmin_XX.dat). The station data are time series of water surface

elevation (Stationyyyy.dat), volume flux (Stationyyyy_M.dat, Stationyyyy_N.dat), and the non-hydrostatic pressure (Stationyyyy_Q.dat) at the locations of stations. Snapshots and time series of non-hydrostatic pressure are output only when wave dispersion is included. The coordinate data are output in ASCII format, while the wavefield data and station data are stored in binary format.

Table 4.5 Output Files of PCOMCOT

File Name	Format	Description
_xcoordintesXX.dat	ASCII	x coordinates of No.XX layer; row vector
_ycoordintesXX.dat	ASCII	y coordinates of No.XX layer; row vector
_bathymetryXX.dat	Binary	bathymetry of No.XX layer
zmax_XX.dat	Binary	maximum water elevation of No.XX layer
zmin_XX.dat	Binary	minimum water elevation of No.XX layer
z_XX_xxxxxx.dat	Binary	snapshot of water elevation of No.XX layer at time step xxxxxx
M_XX_xxxxxx.dat	Binary	snapshot of flux component M
N_XX_xxxxxx.dat	Binary	snapshot of flux component N
Q_XX_xxxxxx.dat*	Binary	snapshot of bottom non-hydrostatic pressure Q
Stationyyyy.dat	Binary	time series of water elevation at station yyyy
Stationyyyy_M.dat	Binary	time series of flux component M
Stationyyyy_N.dat	Binary	time series of flux component N
Stationyyyy_Q.dat*	Binary	time series of bottom non-hydrostatic pressure Q

* The non-hydrostatic pressure Q is normalized by water density.

Note that the two-digit integers "XX" in the output file names are different from those in the input bathymetry file names. For output, "XX" is a continuous integer starting from 01, which indicates the order of each grid layer based on their bathymetry file names. For example, if "layer02.nf/xyz", "layer05.nf/xyz" and "layer12.nf/xyz" are provided by the user, then their corresponding code in the output files are "01", "02" and "03" respectively, despite their nesting

relationship. The six-digit integer "xxxxxx" in the snapshot file names represents the number of time step, rather than the actual time. "yyyy" is a continuous four-digit integer starting from 0001, which corresponds to the order of stations appearing in "Stations.ctl".

The structures of all the wavefield data files, i.e., "_bathymetryXX.dat", "z_XX_xxxxxx.dat", "zmax_XX.dat", "zmin_XX.dat", "Q_XX_xxxxxx.dat", "M_XX_xxxxxx.dat" and "N_XX_xxxxxx.dat" are the same. Because of the staggered grids we use, the dimensions of arrays M and N are different from the others. Content of each line in a wavefield data file is:

(Line 1:)	real data type	(integer*4)
(Line 2:)	nColumn, nRow	(2×integer*4)
(Line 3:)	the first row of data	(nColumn×real data type)
(Line 4:)	the second row of data	(nColumn×real data type)
(Lines..)	...	
(Line nRow+2:)	the last row of data	(nColumn×real data type)

The Format of station data files "Stationyyyy.dat", "Stationyyyy_M.dat", "Stationyyyy_N.dat" and "Stationyyyy_Q.dat" is:

(Line 1:)	real data type	(integer*4)
(Line 2:)	calculationPurpose, nFaults, nDataLength	(3×integer*4)
(Line 3:)	time	(nDataLength×real data type)
(Line 4:)	time series from the first fault	(nDataLength×real data type)
(Line 5:)	time series from the second fault	(nDataLength×real data type)
(Lines..)	...	
(Line nFaults+3:)	time series from the last fault	(nDataLength×real data type)

For forward simulation, there is only one time series in station data files. While for calculation of Green's functions, there are multiple time series, each of which is the Green's function

of a single subfault or subregion with initial surface displacement. Two Matlab scripts "COMCOT_readBinaryDataSnapShot.m" and "COMCOT_readBinaryDataStation.m" are provided in Appendix A, to read the binary wavefild data files and station data files, respectively.

5 Examples

We conduct a series of numerical experiments to validate PCOMCOT model for wave propagation, transformation, and inundation. For small-scale tests in Cartesian coordinates, we compute solitary wave propagation on a flat bottom, water surface oscillation in a paraboloidal basin, and run-up of solitary wave on a circular island. These simulation results are compared with analytical solutions and experimental data, and satisfactory agreement is obtained. For large-scale modeling of real events, the 2011 Tohoku earthquake tsunami is simulated in Earth coordinates, and nested grid system is used. The computed tsunami waveforms agree well with the recordings at tide gauges and DART stations. Compared with the shallow water model, the non-hydrostatic model with dispersion provides much better predictions at the DART stations in deep ocean. We have carried out all the simulations using both the CPU and GPU versions of PCOMCOT, and analyzed their performance in different situations.

5.1 Solitary Wave Propagation on Flat Bottom

As a solution of the classical Boussinesq equations, solitary wave propagation on a flat bottom is a standard test for dispersive wave models. With the balance between wave nonlinearity and dispersion, a solitary wave maintains its shape throughout the propagation. We set up a 3000 m-long, 100 m-wide channel with a constant water depth of 10 m. The x and y directions are along the length and width of the channel, respectively. The grid size is set to be $\Delta x = \Delta y = 1$ m, and 100 m-wide sponge zones are added to both ends of the channel. A 2 m-high solitary wave at the location $x = 300$ m is given as the initial condition, corresponding to wave nonlinearity $\varepsilon = A/h = 0.2$. Both the initial elevation and initial flux M should be provided in the input files. The initial flux is calculated by multiplying the analytical velocity with the total water depth. The simulation time is 200 s, and the time step is set to be 0.05 s. Definitions of necessary parameters in "pcomcot.ctl" are

Basic Control Parameters: Purpose of Calculation = 1, Initial condition = 0, Coordinate system = 1, Total run time = 200.0, Time step = 0.05;

Parameters for Surface Deformation: Apply Kajiura filter = 0;

Parameters for Wave Physics and Numerics: Nonlinearity = 1, Dispersion = 1, Depth change

for Dispersion = 0, Water depth Limit for Dispersion = 0.1, Breaking = 0, Scheme for LSWEs = 0, Froude number Cap to Limit velocity = 10.0, Time interval to apply Filter = 10000.0;

Parameters for Boundary Condition: Boundary Condition = 2, Width of Sponge (West-East) = 100.0, Width of Sponge (South-North) = 0.0, Maximum Manning coefficient in Sponge = 100.0, Damping coefficient A = 2.0, Damping coefficient R = 0.9;

Parameters for Inundation: Permanent dry limit = 50.0, Water depth Limit for wet -> dry = 0.01, Water depth limit for Bottom friction = 0.05, Manning coefficient for Bottom friction = 0.0.

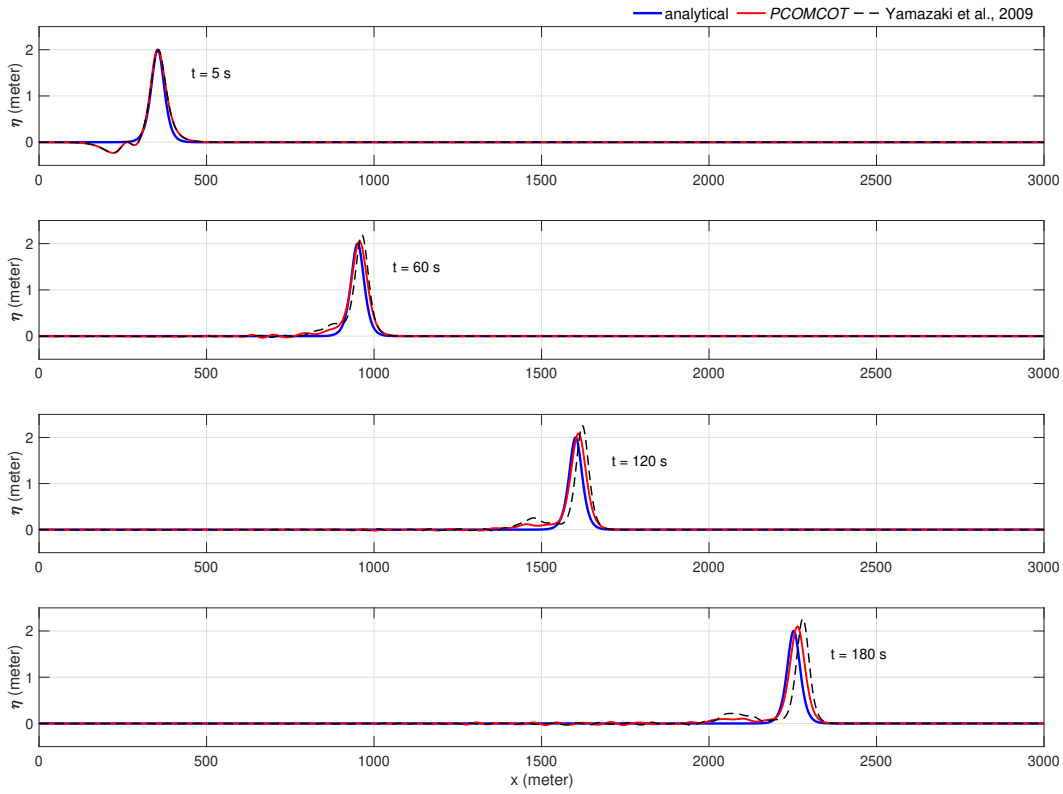

Figure 5.1 Comparison of numerical results and the analytical solution for solitary wave propagation on a flat bottom. The two models make different approximations of the vertical distribution of non-hydrostatic pressure. The governing equations of both models are the same as (2-21), except different values of α and β.

In the above list, the parameter *Depth change for Dispersion* is set to be 0 for the flat bottom,

corresponding to our new model with $\alpha = 2/3$, $\beta = 0.5$ in the governing equations (2-21). To show how the accuracy of wave dispersion is improved by our model, we also carry out the computation using the previous governing equations with $\alpha = 0.5$, $\beta = 1.0$ (*Stelling and Zijlema*, 2003; *Yamazaki et al.*, 2009), by setting the parameter *Depth change for Dispersion* to 1. The difference between our new model and the previous one lies in the assumption on the distribution of non-hydrostatic pressure. Based on the Boussinesq equations, we approximates the vertical distribution of p_{dyn} using a simple quadratic function. In comparison, the previous model assumes a linear vertical distribution, which has proved to be less reasonable.

The simulation results given by two models are presented in Figure 5.1. At the first few seconds, the wave profile is disturbed, due to using the analytical solution as the initial condition. About 5 s later, the wave profiles of both models propagate stably along the channel, with the crest height ≈ 2.06 m for our new model and 2.22 m for the previous one. It is seen that the result of our new model is very close to the analytical solution. While in the previous model, the wave profile travels faster, and there is a small trailing wave following the crest. Thus, by adopting a more realistic approximation of non-hydrostatic pressure, slightly better results can be obtained on smooth bottom. In real tsunami events where the water depth is non-uniform, the difference between these two models would be smaller.

5.2 Fluid Oscillation in a Paraboloidal Basin

The oscillation of water surface in a paraboloidal basin is used to verify the moving boundary technique for tsunami run-up. Analytical solutions for this type of motions are derived by *Thacker* (1981) and are highly valuable for calibrating numerical inundation models (*Cho and Kim*, 2009). One of the exact solutions for curved water surface oscillating in a circular paraboloidal basin is presented here and compared with our numerical results. The still water depth is given as

$$h = h_0 \left(1 - \frac{r^2}{a^2}\right), \tag{5-1}$$

where r is the distance from the rotational axis, a is the radius of the paraboloid, and h_0 is the water depth at the center. In this example, h_0 is 1 m and a is 2500 m. The expression of water

surface elevation varying with time and space is

$$\eta = h_0 \left\{ \frac{(1-A^2)^{\frac{1}{2}}}{1-A\cos\omega t} - 1 - \frac{r^2}{a^2}\left[\frac{1-A^2}{(1-A\cos\omega t)^2} - 1\right]\right\}, \tag{5-2}$$

in which the angular frequency ω and the non-dimensional parameter A are given as

$$\omega = \frac{1}{a}(8gh_0)^{\frac{1}{2}}, \quad A = \frac{a^4 - r_0^4}{a^4 + r_0^4}. \tag{5-3}$$

Here, $r_0 = 2000\,\text{m}$, $\omega = 3.54 \times 10^{-3}$ rad/s, $A = 0.419$, and the period of free oscillation is $\sim$1700 s. In numerical computation, the grid sizes Δx and Δy are set to be 10 m. The total simulation time is 7000 s covering 4 periods, and the time step size is 1 s to satisfy the C.F.L condition. The initial surface elevation is defined with the expression (5-2) at $t = 0$. For comparison with analytical solution, no bottom friction is considered. Wave dispersion is not included because the fluid motions are governed by nonlinear shallow water equations. Definitions of necessary parameters in "pcomcot.ctl" are

Basic Control Parameters: Purpose of Calculation = 1, Initial condition = 0, Coordinate system = 1, Total run time = 7000.0, Time step = 1.0;

Parameters for Surface Deformation: Apply Kajiura filter = 0;

Parameters for Wave Physics and Numerics: Nonlinearity = 1, Dispersion = 0, Breaking = 0, Scheme for LSWEs = 1, Froude number Cap to Limit velocity = 10.0, Time interval to apply Filter = 50000.0;

Parameters for Boundary Condition: Boundary Condition = 1;

Parameters for Inundation: Permanent dry limit = 50.0, Water depth Limit for wet -> dry = 0.01, Water depth limit for Bottom friction = 0.05, Manning coefficient for Bottom friction = 0.0.

At the beginning of computation, the surface is convex. Driven by gravity, the surface drops gradually with the shoreline expanding on the basin. When the run-up height reaches its maximum and the surface becomes concave, the water begins flowing back towards the center. The surface returns to its initial shape at the end of a period. Figure 5.2 displays the water surface profiles at different moments on the cross section through the rotation axis. In the first two periods,

the numerical model gives almost the same surface profiles as the analytical solutions. As the computation goes on, the numerical diffusion of the upwind scheme makes water elevation lower than the exact solutions when the surface rises.

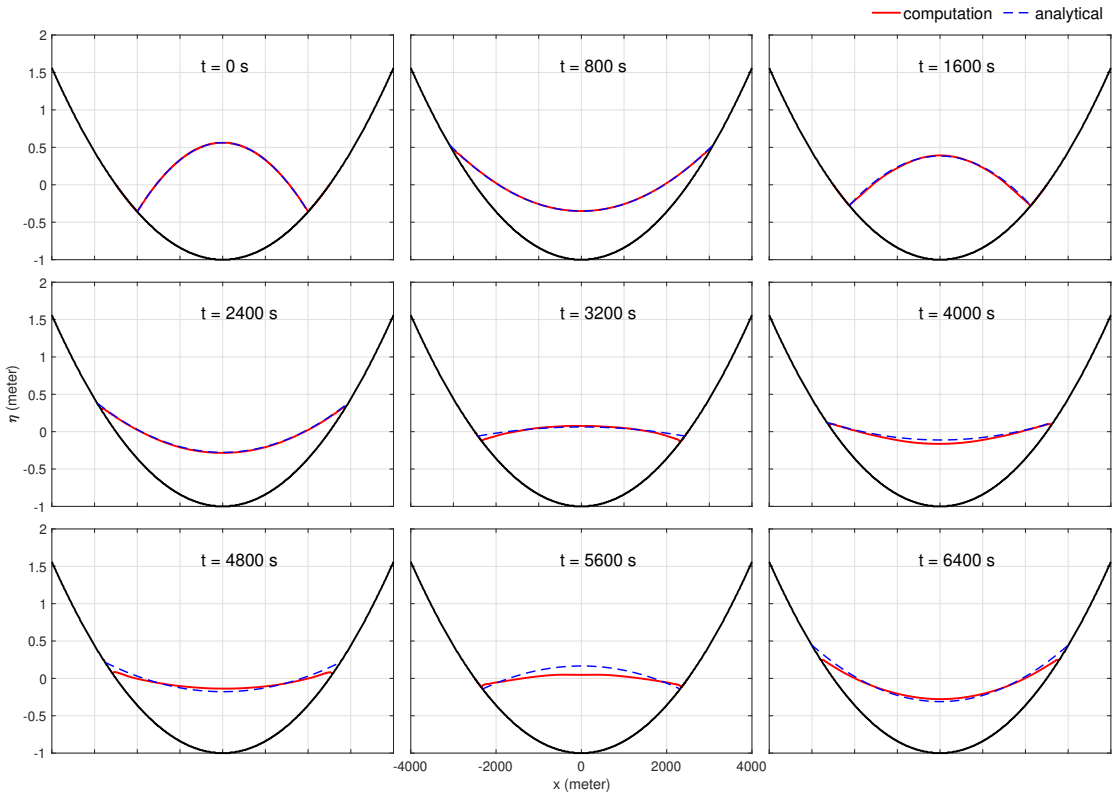

Figure 5.2 Comparison of water surface profiles on a paraboloidal basin between the analytical solution and numerical result.

5.3 Solitary Wave Run-up on a Circular Island

The 1992 Flores Island tsunami impinged unexpectedly serious damage on the lee side of Babi Island, Indonesia (*Yeh et al.*, 1993), which sparked great interest of researchers in tsunami run-up on a small island (e.g., *Briggs et al.*, 1995; *Liu et al.*, 1995; *KÂNOĞLU and SYNOLAKIS*, 1998). The large-scale laboratory experiments of interactions between solitary wave and a circular island were conducted by *Briggs et al.* (1995) at Coastal Engineering Research Center, Vicksburg, Mississippi. These experiments reveal the fact that tsunami run-up on the sheltered backside of an island can be comparable or even higher than that on the front side in certain situations. The

collected data have been used for validation of various models, including shallow water equations (*Liu et al.*, 1995), Boussinesq equations (*Chen et al.*, 2000) and non-hydrostatic model (*Yamazaki et al.*, 2009).

In the laboratory, a 62.5 cm-high, 7.2 m toe-diameter, and 2.2 m crest-diameter circular island with a 1:4 slope was located in a large basin. The initial solitary wave-like profiles were generated by a directional spectral wave generator. Experiments were performed at two different water depths ($h = 32$ cm and 42 cm), and solitary waves with three different height-to-depth ratios ($\varepsilon = A/h =$ 0.045, 0.096 and 0.181) were tested. Here we simulate the cases with water depth of 32 cm, and compare the results with the experiment data provided by NOAA Center for Tsunami Research. In our numerical model, a truncated cone with the same size is set up in the 80 m-long and 24 m-wide domain. 4 m-wide sponge zones are applied to both ends in x direction. As shown in Figure 5.3, four gauges around the island are used to record the time histories of water surface elevation. Gauge 6 is in front of the island at the toe. Gauges 9, 16 and 22 are near the still shoreline of the island, at the 0°, 90° and 180° radial lines, respectively. Coordinates of these wave gauges are given by Table 5.1. Note that the location of the island center ($x = 40$ m, $y = 0$ m) in our simulations is different from that in the lab experiments. Thus, the coordinates of gauges are adjusted to keep their relative locations to the island unchanged. The computation domain is discretized at a grid size of $\Delta x = \Delta y = 0.05$ m. A solitary wave profile with the crest at $x = 15$ m is given as the initial condition. The Manning's roughness coefficient is set to be 0.013, corresponding to smooth concrete surface. Since *Liu et al.* (1995) reported that the wave broke in the laboratory realization, the eddy-viscosity scheme is used in this example. These simulations last for 35 s, with the time step size set to be 0.01 s.

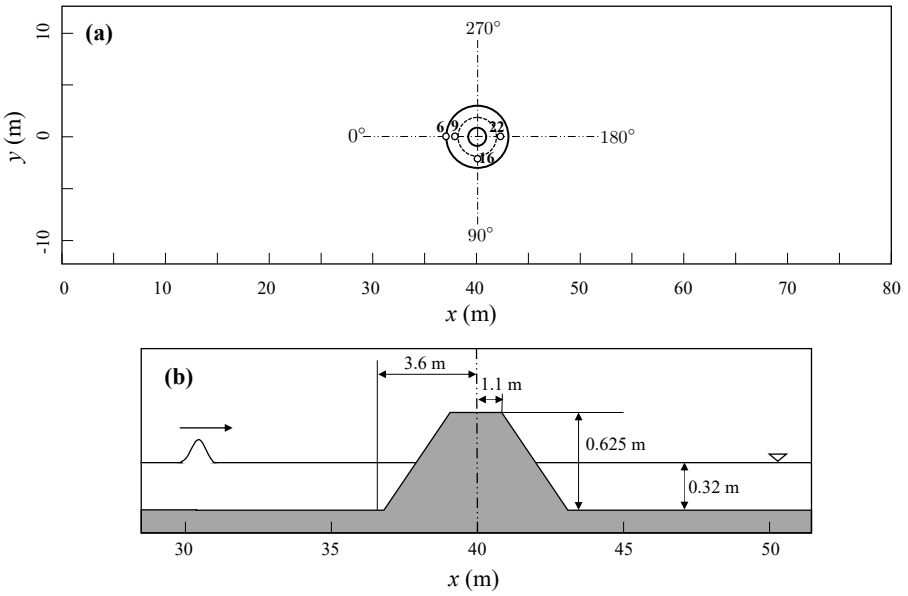

Figure 5.3 Schematic of numerical test of solitary wave run-up on a conical island. **(a)** and **(b)** are the top view and side view, respectively. In panel **(a)**, the base and the still shoreline of the island are denoted with solid and dashed lines, respectively. Locations of wave gauges are indicated by hollow dots.

Table 5.1 Coordinates of gauges in numerical test of solitary wave run-up.

gauge number	x (m)	y (m)
6	36.4	0.0
9	37.4	0.0
16	40.0	-2.58
22	42.6	0.0

Definitions of necessary parameters in "pcomcot.ctl" are

Basic Control Parameters: Purpose of Calculation = 1, Initial condition = 0, Coordinate system = 1, Total run time = 35.0, Time step = 0.01;

Parameters for Surface Deformation: Apply Kajiura filter = 0;

Parameters for Wave Physics and Numerics: Nonlinearity = 1, Dispersion = 1, Depth change for Dispersion = 0, Water depth Limit for Dispersion = 0.05, Breaking = 1, Scheme for LSWEs = 0, Froude number Cap to Limit velocity = 10.0, Time interval to apply Filter = 10000.0;

Parameters for Boundary Condition: Boundary Condition = 2, Width of Sponge (West-East) = 4.0, Width of Sponge (South-North) = 0.0, Maximum Manning coefficient in Sponge = 100.0, Damping coefficient A = 2.0, Damping coefficient R = 0.9;

Parameters for Inundation: Permanent dry limit = 50.0, Water depth Limit for wet -> dry = 0.005, Water depth limit for Bottom friction = 0.005, Manning coefficient for Bottom friction = 0.013.

The non-hydrostatic model together with eddy-viscosity scheme provide numerical solutions for solitary wave propagation and transformation around the island. Figure 5.4 and Figure 5.5 display the water surfaces at different moments in three cases. Figure 5.4 shows the moments when the incident wave reaches the front face of the island, and 2 s later when the trapped waves start propagating toward the back side. Figure 5.5 shows the process of trapped waves wrapping around the island, colliding at the lee side, and then passing across each other. For $A/h = 0.045$ and 0.096, the water surface is relatively smooth around the island, and no significant high-frequency dispersive waves can be seen. For $A/h = 0.181$, the water surface becomes steep and rough at the back side, and evident short dispersive waves are generated at the collision of trapped waves. After that, high-frequency energy is leaking continuously from the trapped waves wrapping toward the front side, leading to the mesh-like wave pattern behind the island. Besides, the steep wave profile in the case of $A/h = 0.181$ causes breaking all around the island, which reduces the run-up height at the lee side.

Figure 5.6 compares the computed waveforms at four gauges with the experiment data. For each case, to align the timing of waveforms, the measured data at all gauges are shifted by a uniform offset, so that the arrival time of wave peak at gauge 6 is consistent with the numerical result. It is seen that the numerical model reproduces the primary waves relatively well, except that the wave amplitude at gauge 22 is overestimated in the $A/h = 0.181$ case. The run-up and inundation around the island are shown in Figure 5.7. Good agreement is obtained in all directions for all the cases, especially that the inundation for $A/h = 0.181$, which is poorly reproduced by non-dispersive and non-breaking models (*Liu et al.*, 1995), are well predicted here. We notice that the computed

run-up height at the front side is slightly lower than the measured data for $A/h = 0.181$, and the depressions behind the leading crests at all gauges tend to be underestimated. This may be due to the loss of energy caused by the upwind scheme and moving boundary method. Nevertheless, the overall good agreement with laboratory data demonstrates the capability of PCOMCOT to simulate the complex processes of dispersive breaking waves.

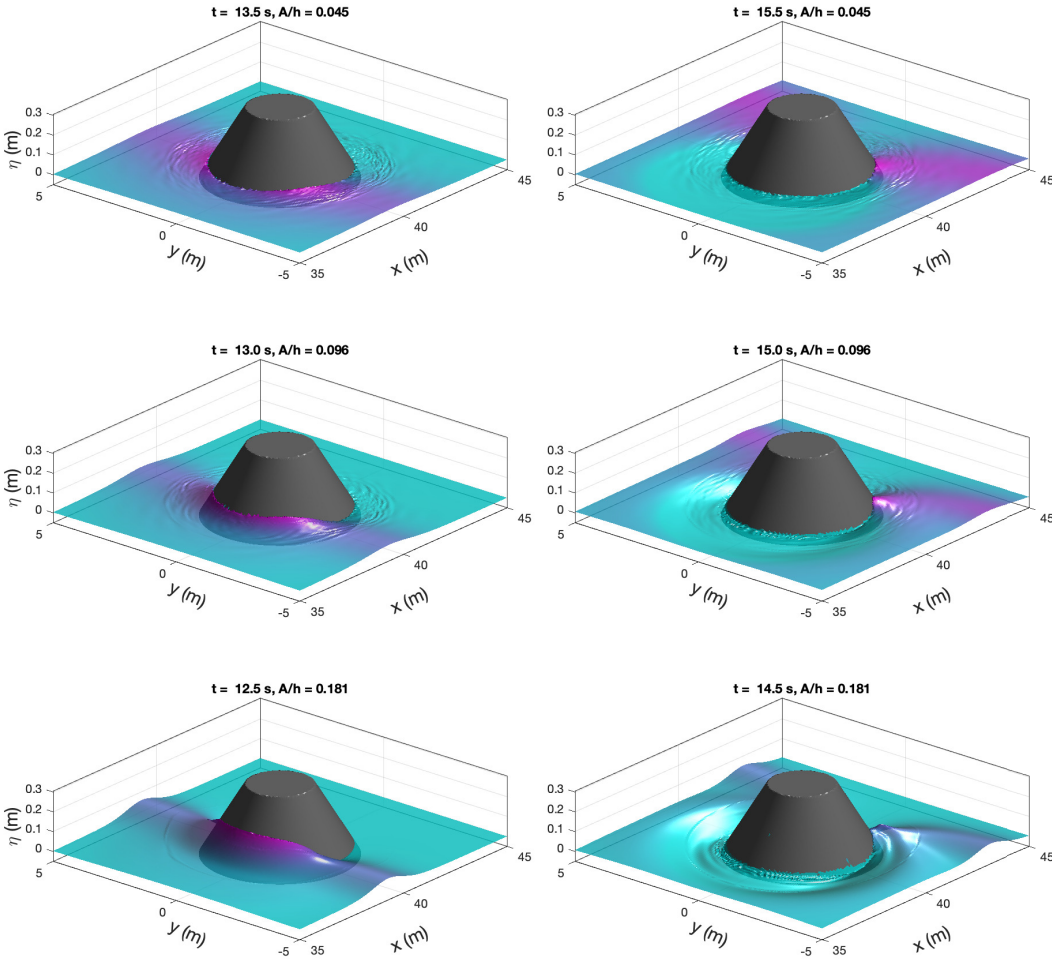

Figure 5.4 Transformation of solitary waves on the front face of a conical island.

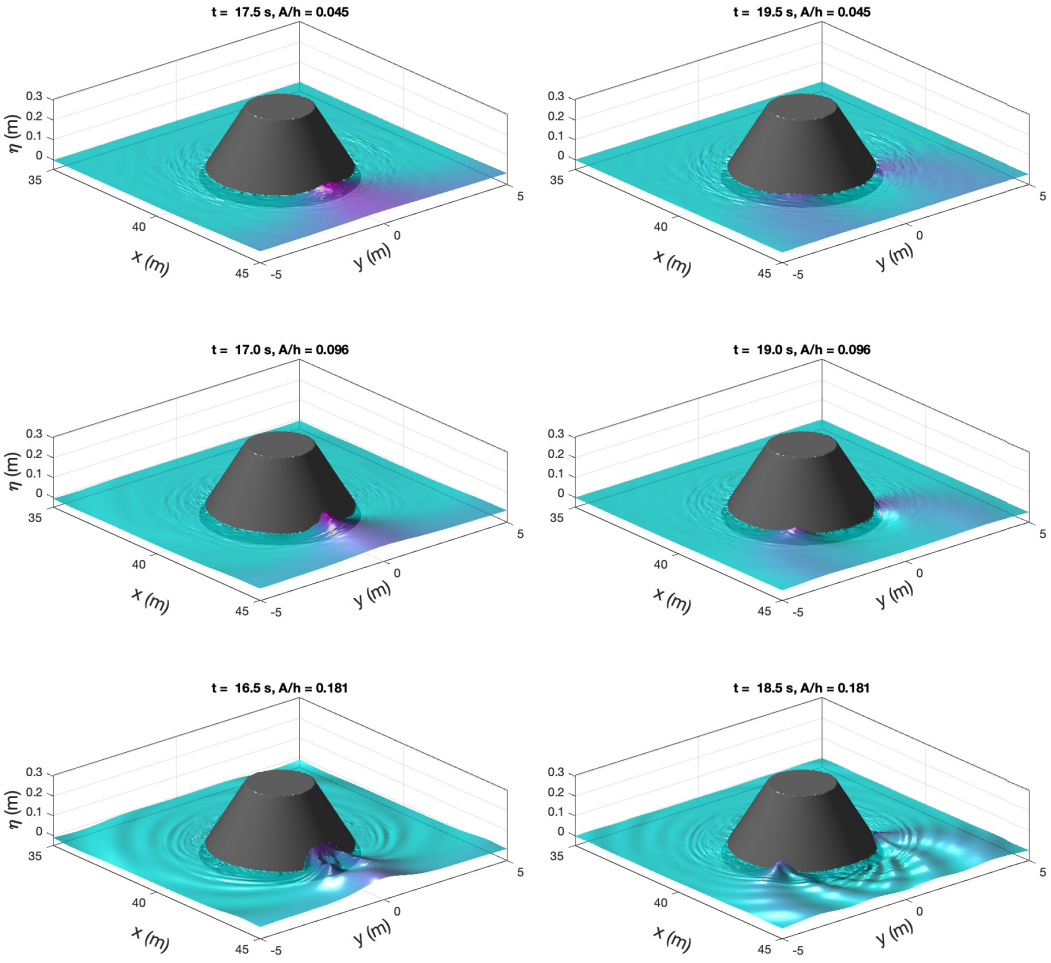

Figure 5.5 Propagation and interaction of trapped waves at the lee side of a conical island.

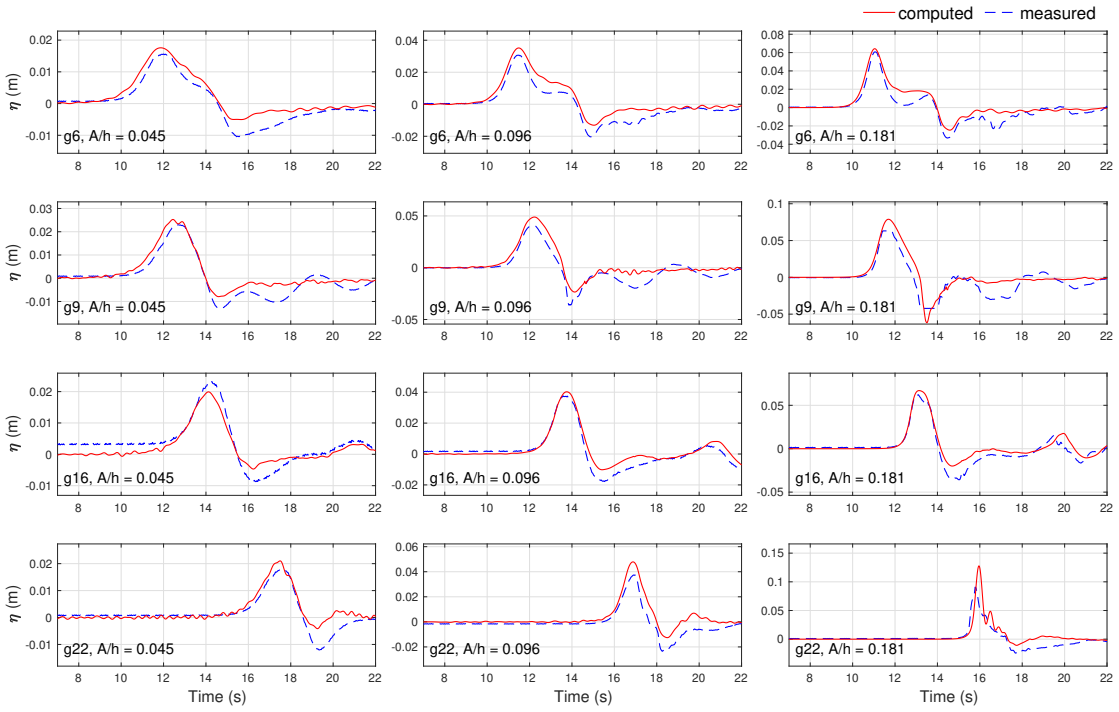

Figure 5.6 Comparison of measured and computed waveforms at four gauges.

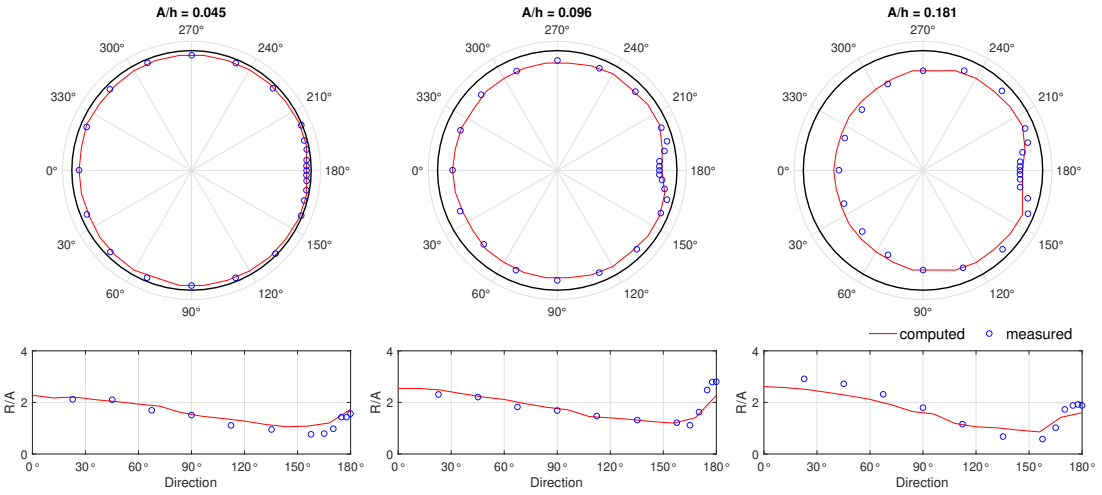

Figure 5.7 Comparison of inundation and run-up around the conical island between measurements and the numerical model. The run-up heights are normalized by the incident wave height. The 0° and 180° directions correspond to the exposed front side and the sheltered lee side, respectively.

5.4 2011 Tohoku Tsunami

The tsunami triggered by the 2011 Tohoku earthquake is used to verify our dispersive model and the nesting algorithm in a realistic scenario. This megathrust earthquake occurred near the trench to the northeast of Japan, where the Pacific plate is subducted beneath the North American plate. Extensive studies have revealed large coseismic slip extending to the updip edge of the fault (e.g., *Fujii et al.*, 2011; *Fujiwara et al.*, 2011; *Satake et al.*, 2013), which caused the unexpectedly large tsunami. The very shallow slip near the trench could lead to short-wavelength tsunami waves, so the dispersive effects in deep ocean may be more significant than in most earthquake tsunami events. Previous studies have suggested the importance of dispersive models for predicting the tsunami waveforms at some DART stations (*Baba et al.*, 2015; *Glimsdal et al.*, 2013; *Saito et al.*, 2011).

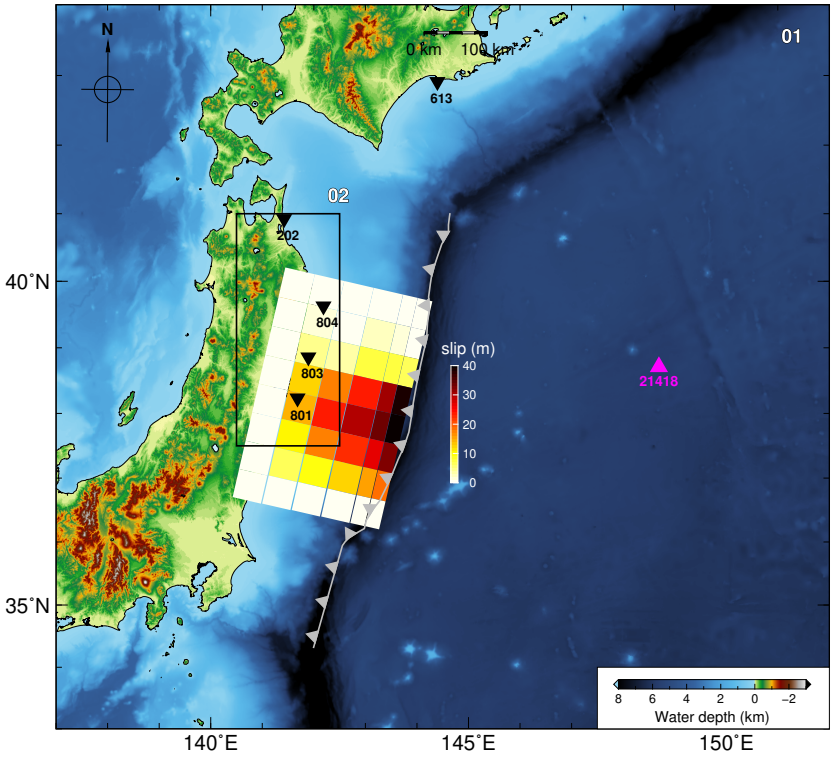

Figure 5.8 The earthquake source model and computational domains for the 2011 Tohoku tsunami. The finite-fault model with heterogeneous slip is obtained by inverting the tsunami data. Magenta triangle and black inverted triangles denote the DART station and GPS tide gauges, respectively. Rectangular boxes outline the domains of nested grid layers.

To simulate the 2011 Tohoku tsunami, we estimate a finite-fault model for the earthquake

source using an inverse algorithm. The fault geometry is obtained from the USGS moment tensor solution, and is divided into 8 and 6 subfault patches in the along-strike and down-dip directions, respectively. The slip on each subfault is then estimated by inverting the tsunami data based on the linear shallow water equations. The seafloor deformation is calculated with Okada's half-space elastic model (*Okada*, 1985), and the contribution of horizontal motion to seafloor uplift is considered with the formula of *Tanioka and Satake* (1996). The initial surface elevation is determined by applying the Kajiura filter (*Kajiura*, 1963; *Glimsdal et al.*, 2013) to the seafloor displacement.

We compute the tsunami waves in two nested grid layers. The outer layer is from 138°E to 152°E in longitude, and from 33°N to 44°N in latitude, with a resolution of 1 arcmin. The inner layer covers the northeastern coast of Japan (140.5°E $\sim$ 142.5°E, 37.5°N $\sim$ 41°N) at a resolution of 15 arcsec. A 200 km-wide sponge zone is surrounding the top layer is used to avoid wave reflection on the boundary. The bathymetry data of both layers are extracted from GEBCO2020. Six tsunami stations, including five GPS tide gauges 202, 613, 801, 803, 804 and a DART station 21418 are used for comparison between the simulation results and the tsunami observations. The earthquake source model, computational domains, and the locations of tsunami observation are displayed in Figure 5.8. Because frequency dispersion may be necessary for the transformation of tsunami waves when propagating from deep ocean to the shallow region, dispersive models are used for both layers. Wave nonlinearity is only considered in the inner layer. The total simulation time is 6000 s, with the time step size set to be 2 s for the outer layer . The time step size for the inner layer is automatically set to be 1.0 s , so that the C.F.L. number for both layers are almost the same. Definitions of necessary parameters in "pcomcot.ctl" are

Basic Control Parameters: Purpose of Calculation = 1, Initial condition = 1, Coordinate system = 0, Total run time = 6000.0, Time step = 2.0, Feedback to parent layer = 0;

Parameters for Surface Deformation: Consider Horizontal Motion = 1, Apply Kajiura filter = 1, Use average depth for Kajiura = 1, Water depth Limit for Kajiura = 100.0;

Parameters for Wave Physics and Numerics: Nonlinearity = 1, Dispersion = 1, Depth change for Dispersion = 0, Water depth Limit for Dispersion = 100.0, Breaking = 0, Scheme for LSWEs = 1, Froude number Cap to Limit velocity = 10.0, Time interval to apply Filter = 50000.0;

Parameters for Boundary Condition: Boundary Condition = 2, Width of Sponge (West-East) = 100000.0, Width of Sponge (South-North) = 100000.0, Maximum Manning coefficient in Sponge = 100.0, Damping coefficient A = 2.0, Damping coefficient R = 0.9;

Parameters for Inundation: Permanent dry limit = 100.0, Water depth Limit for wet –> dry = 0.01, Water depth limit for Bottom friction = 0.05, Manning coefficient for Bottom friction = 0.03.

The layer-specific parameters in "layers.ctl" are

Layer Name : 01

Nonlinearity = 0, Dispersion = 1, Depth change for Dispersion = 0, Breaking = 0, Scheme for LSWEs = 1, Computing Start Time = 0.0, Time interval to apply Filter = 50000.0;

Layer Name : 02

Nonlinearity = 1, Dispersion = 1, Depth change for Dispersion = 0; Breaking = 0, Scheme for LSWEs = 1, Computing Start Time = 0.0, Time interval to apply Filter = 50000.0.

Figure 5.9 displays the water surfaces in two grid layers at different moments. It can be seen that the tsunami waves propagate seamlessly across the boundary of nesting grids, and no unphysical oscillations are introduced. Comparison of water surface profiles between the dispersive and non-dispersive models in Figure 5.10 shows evident dispersive short waves caused by the large shallow slip. Figure 5.9 vividly depicts how frequency dispersion influences tsunami waves in the deep ocean and the shallow region, respectively. In the deep ocean, a series of trailing waves are generated, and both the amplitude and steepness of the leading wave are reduced. While when the wave front approaches the coast and becomes sufficiently steep, dispersion begins to cause splitting of the wave. As a result, the leading wave is amplified, and multiple short-period waves are formed near the crest.

The simulated and observed tsunami waveforms at six stations are plotted in Figure 5.11. The dispersive and non-dispersive models give almost identical results in the coast, which agree well with the observation data. On the other hand, at the DART station 21418, the dispersive model gives excellent predictions of the leading wave and the small trailing waves, which cannot be reproduced by the shallow water model. These results suggest that dispersive effects are important in deep ocean, while the shallow water equations are generally adequate for simulating local tsunamis in coastal areas.

In summary, the non-hydrostatic model for wave dispersion and the nesting algorithm perform satisfactorily in terms of accuracy and stability. For the 2011 Tohoku case, the time cost of the dispersive model is only $\sim$2.5 times longer than that of the shallow water model. The good balance between accuracy and efficiency achieved by PCOCMOT supports its application to accurate large-scale tsunami modeling.

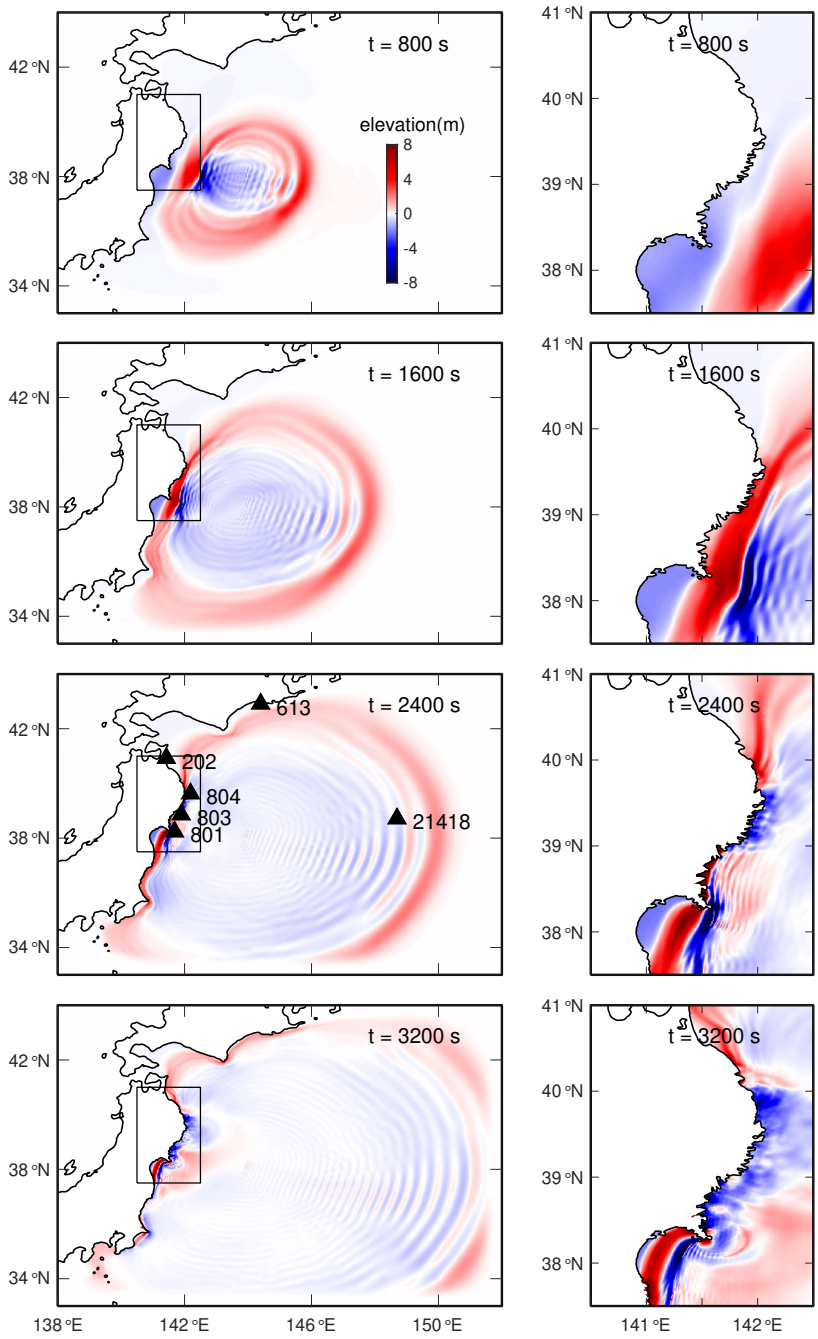

Figure 5.9 Snapshots of water surface elevation given by the dispersive model in nested grids. The left panels show the surface profiles in the parent layer, with black boxes indicating the domain of the child layer. The right panels are snapshots of the child layer at the same moments.

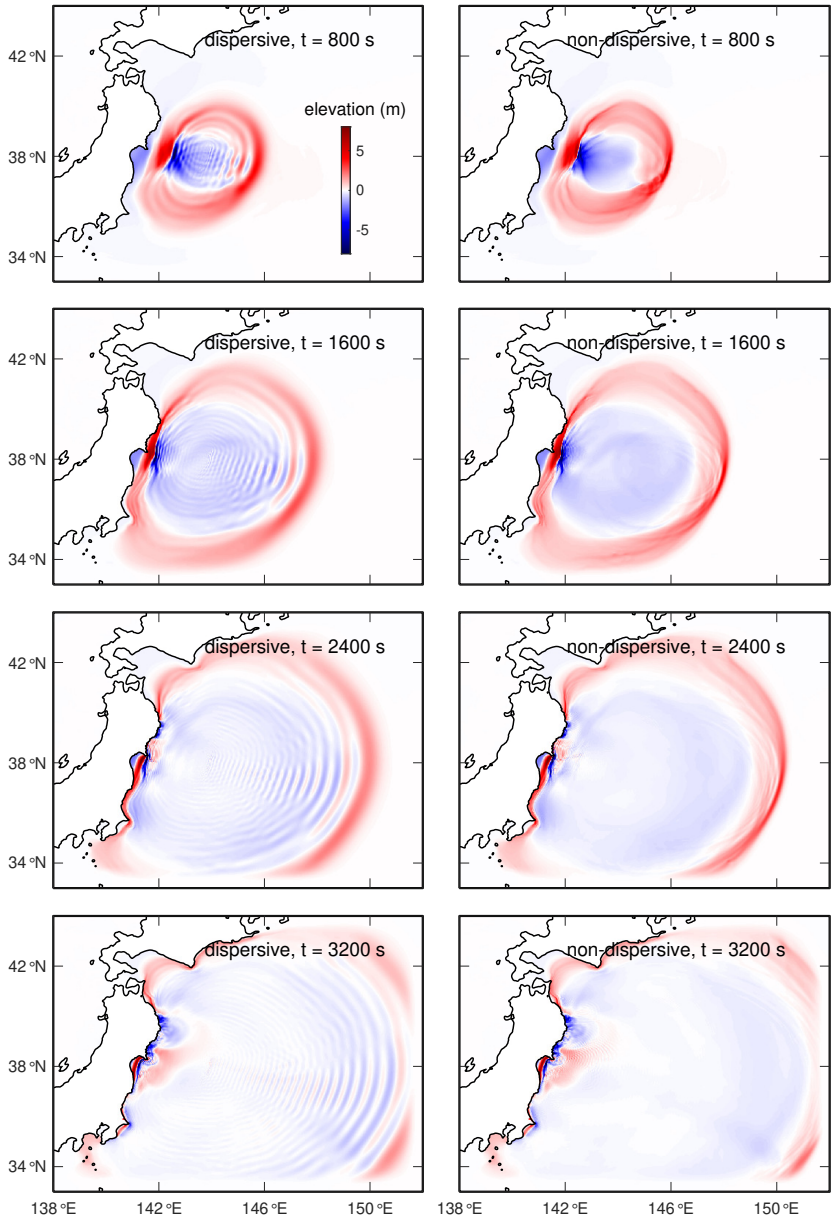

Figure 5.10 Comparison of water surface profiles between the dispersive and non-dispersive models.

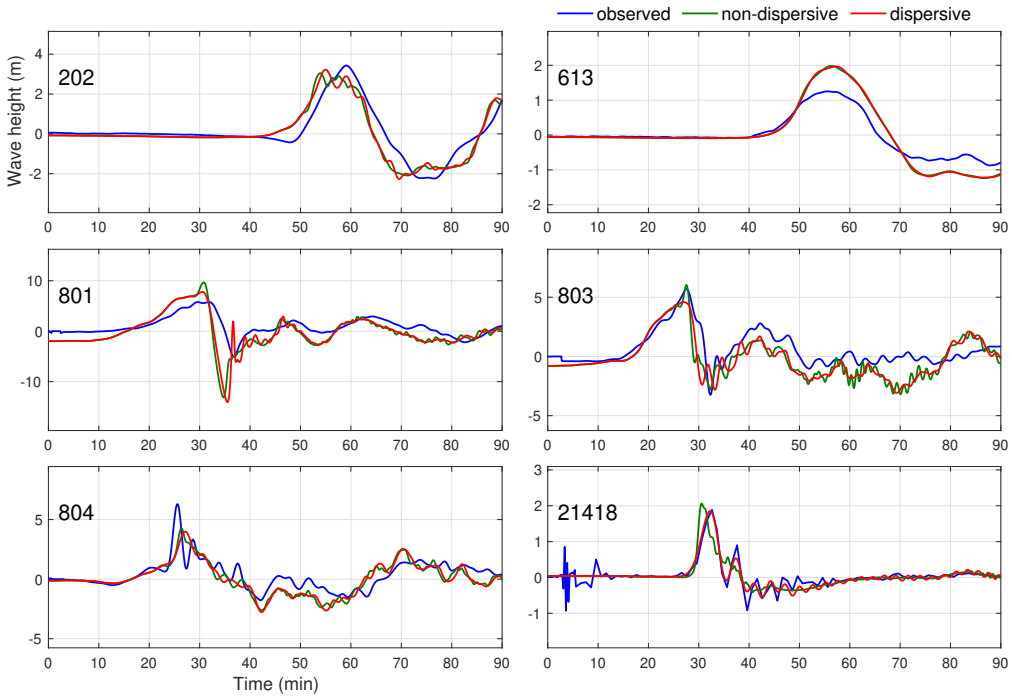

Figure 5.11 Comparison of tsunami waveforms recorded at wave gauges and computed with different models.

5.5 Performance Analysis

All the numerical tests have been conducted for both the CPU and GPU versions of PCOMCOT. Before analyzing the model performance on CPU and GPU, it is necessary to state that these two versions give exactly the same results. Take the case of 2011 Tohoku tsunami as example, the dispersive waveforms computed by the CPU and GPU versions are plotted in Figure 5.12. Double- and single-precision computing are performed on CPU and GPU, respectively. It is seen that tsunami waves given by the two versions are completely identical, and the difference in floating-point precision has no influence on accuracy of the result.

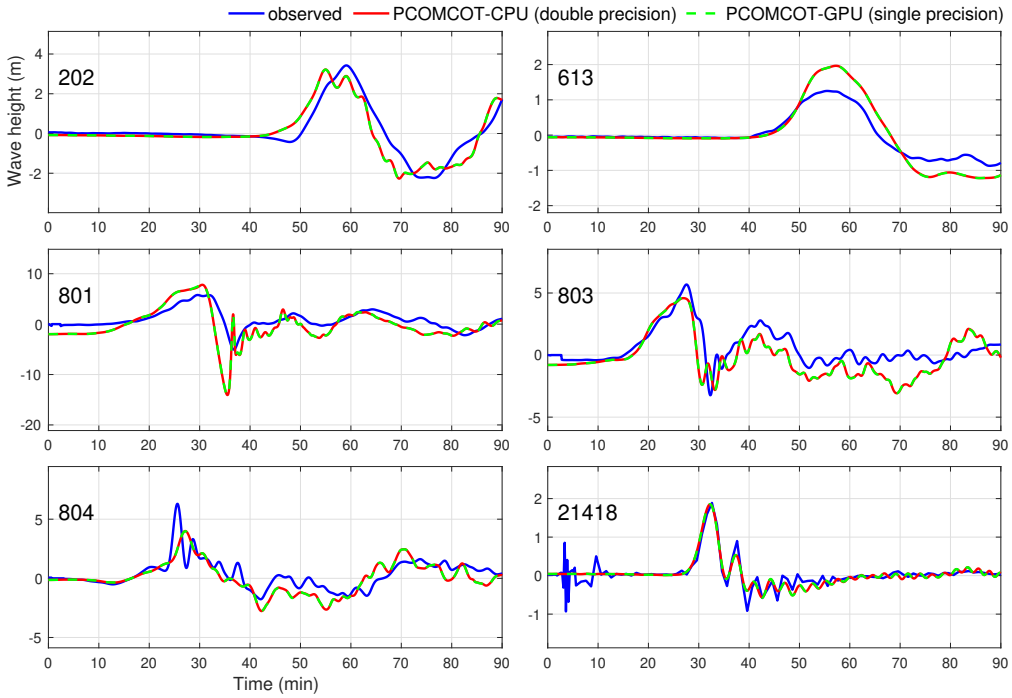

Figure 5.12 Comparison of dispersive tsunami waves computed by CPU and GPU versions of PCOMCOT.

We run the CPU version of PCOMCOT on 32 cores of an Intel Xeon Platinum 8358 processor, with a base frequency of 2.6 GHz. For the GPU version, we use three different NVIDIA Tesla GPUs — P100, V100, and A100, which represent the out-of-date, mainstream, and the most advanced architectures, respectively. On each of these platforms, we adopt four types of equations (i.e., linear/nonlinear, dispersive/non-dispersive) for simulation, and record the corresponding time cost. Table 5.2 displays the time cost for the 2011 Tohoku cause using different hardwares. The PCOMCOT model is quite efficient on the 32-core CPU, where it takes only 10 min to finish the nonlinear dispersive simulation for 1.6 hours. The model performance on P100 GPU is comparable to that on the Intel Xeon 8358 CPU. Significant decease in time cost can be seen on a V100 and A100 GPU.

Table 5.2 Time cost of PCOMCOT on different hardwares for the 2011 Tohoku case

	Intel® Xeon® 8358 CPU	NVIDIA® P100 GPU	V100 GPU	A100 GPU
linear, non-dispersive	3.5 min	4 min	50 s	30 s
nonlinear, non-dispersive	4 min	5 min	1 min	40 s
linear, dispersive	9.5 min	10 min	3.3 min	1.7 min
nonlinear, dispersive	10 min	10.5 min	3.5 min	1.8 min

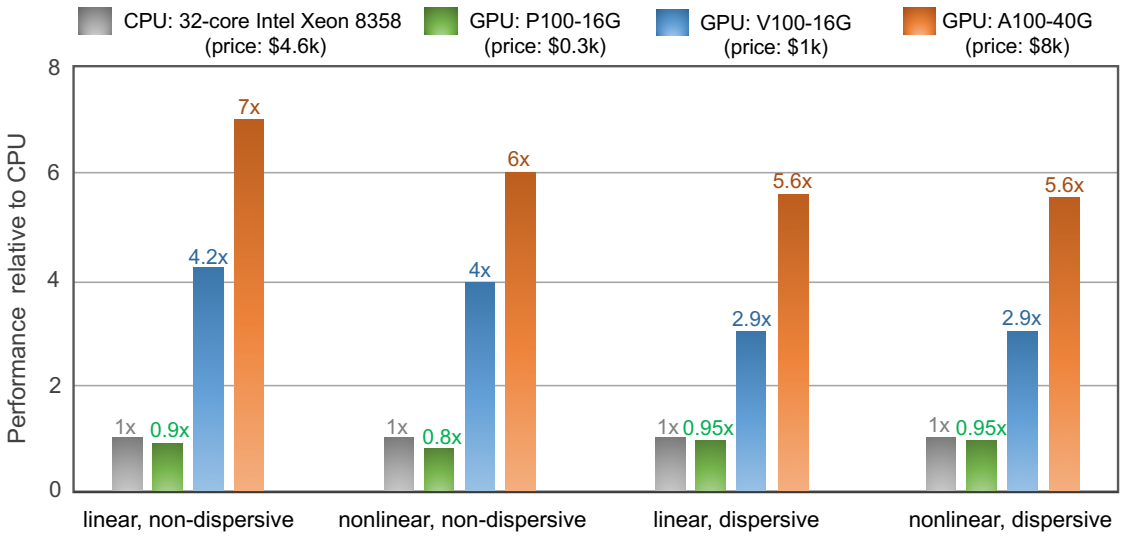

Figure 5.13 Speedup of PCOMCOT-GPU over PCOMCOT-CPU for the 2011 Tohoku case.

Figure 5.13 shows speedup of PCOMCOT-GPU over PCOMCOT-CPU, together with the current market prices for different hardwares. Using a V100 GPU, about 4 and 3 times speedup can be obtained for non-dispersive and dispersive simulations, respectively. While on an A100 GPU, a speedup ratio of ∼6 is achieved for all types of simulations. The slight decrease in speedup

ratio for dispersive simulations is due to the less effective preconditioning method used by GPU version to solve non-hydrostatic pressure. Considering both the performance and hardware price, the mainstream V100 GPU can provide significant speedup over a CPU which is $\sim$5 times more expensive.

6 Citation of PCOMCOT

To publish papers containing the use of PCOMCOT or mentioning this model, please cite our articles about the algorithms and applications of PCOMCOT. The articles on PCOMCOT include:

- Zhu, Y., C. An, H. Yu, W. Zhang, and X. Chen (2024), High-resolution tsunami hazard assessment for the guangdong-hong kong-macao greater bay area based on a non-hydrostatic tsunami model, *Science China Earth Sciences*, *67*(7), 2326–2351, doi: 10.1007/s11430-023-1300-9

- An, C., H. Liu, Z. Ren, and Y. Yuan (2018), Prediction of tsunami waves by uniform slip models, *Journal of Geophysical Research: Oceans*, *123*(11), 8366–8382, doi: 10.1029/2018JC014363

- Wang, X., and P. L.-F. Liu (2006), An analysis of 2004 Sumatra earthquake fault plane mechanisms and Indian Ocean tsunami, *Journal of Hydraulic Research*, *44*(2), 147–154, doi: 10.1080/00221686.2006.9521671

Appendix A MatLab Scripts to Read PCOMCOT Output

A.1 A sample Matlab script to read snapshots

```
function [x, y, dat] = COMCOT_readBinaryDataSnapshot(fn)
% [x, y, dat] = COMCOT_readBinaryDataSnapshot(fn)

lbslash = strfind(fn,'/');
if(isempty(lbslash))
    path = './';
    fn0 = fn;
else
    lbslashlast = max(lbslash);
    path = fn(1:lbslashlast);
    fn0 = fn(lbslashlast+1:end);
end

ii = strfind(fn0,'_');
ilayer = str2num(fn0(ii(1)+1:ii(1)+2));
if(fn0(1)=='_')
    ii = strfind(fn0,'.dat');
    ilayer=str2num(fn0(ii(1)-2:ii(1)-1));
end
if(isempty(ilayer) == 1)
    ilayer = 1;
end

x = load([path,'_xcoordinate',sprintf('%02d',ilayer),'.dat']);
y = load([path,'_ycoordinate',sprintf('%02d',ilayer),'.dat']);

if((fn0(1) == 'M') || (fn0(1) == 'm'))
        x = x(1:end-1)+0.5*(x(2)-x(1));
```

```
elseif((fn0(1) == 'N') || (fn0(1) == 'n'))
        y = y(1:end-1)+0.5*(y(2)-y(1));
end

fid = fopen(fn, 'rb');
StartTag = fread(fid, 1, 'integer*4');
fp = fread(fid, 1, 'integer*4'); % floating-point precision: 4/8
EndTag = fread(fid, 1, 'integer*4');
realType = sprintf('real*%d',fp);

StartTag = fread(fid, 1, 'integer*4');
ncol = fread(fid, 1, 'integer*4'); %NX = ncol
nrow = fread(fid, 1, 'integer*4'); %NY = nrow
EndTag = fread(fid, 1, 'integer*4');
dat = zeros(nrow, ncol);
for i = 1:nrow
    StartTag = fread(fid, 1, 'integer*4');
    dattmp = fread(fid, ncol, realType);
    dat(i,:) = dattmp(:);
    EndTag = fread(fid, 1, 'integer*4');
end
fclose(fid);
```

A.2 A sample Matlab script to read station data

```
function dat = COMCOT_readBinaryDataStation(fn)
% read data from binary file to memory
% dat = COMCOT_readBinaryDataStation(fn)

fid = fopen(fn, 'rb');
StartTag = fread(fid, 1, 'integer*4');
fp = fread(fid, 1, 'integer*4'); % floating-point precision: 4/8
EndTag = fread(fid, 1, 'integer*4');
realType = sprintf('real*%d',fp);

StartTag = fread(fid, 1, 'integer*4');
ComputeGreen = fread(fid, 1, 'integer*4');
NFaults = fread(fid, 1, 'integer*4');
NDataLength = fread(fid, 1, 'integer*4');
EndTag = fread(fid, 1, 'integer*4');

if(ComputeGreen ~= 1)
    nDatCol = NFaults+1;
else
    nDatCol = 2;
end
dat = zeros(NDataLength, nDatCol);

StartTag = fread(fid, 1, 'integer*4');
t = fread(fid, NDataLength, realType);
dat(:,1) = t(:);
EndTag = fread(fid, 1, 'integer*4');

for iCol = 1:nDatCol-1
```

```
    StartTag = fread(fid, 1, 'integer*4');
    h = fread(fid, NDataLength, realType);
    dat(:,1+iCol) = h(:);
    EndTag = fread(fid, 1, 'integer*4');
end
fclose(fid);
```

References

An, C., I. Sepúlveda, and P. L.-F. Liu (2014), Tsunami source and its validation of the 2014 Iquique, Chile, earthquake, *Geophysical Research Letters*, *41*(11), 3988–3994, doi: 10.1002/2014GL060567.

An, C., H. Liu, Z. Ren, and Y. Yuan (2018), Prediction of tsunami waves by uniform slip models, *Journal of Geophysical Research: Oceans*, *123*(11), 8366–8382, doi: 10.1029/2018JC014363.

Arcos, M. E. M., and R. J. LeVeque (2015), Validating velocities in the GeoClaw tsunami model using observations near Hawaii from the 2011 Tohoku tsunami, *Pure and Applied Geophysics*, *172*(3), 849–867, doi: 10.1007/s00024-014-0980-y.

Baba, T., N. Takahashi, Y. Kaneda, K. Ando, D. Matsuoka, and T. Kato (2015), Parallel implementation of dispersive tsunami wave modeling with a nesting algorithm for the 2011 Tohoku tsunami, *Pure and Applied Geophysics*, *172*(12), 3455–3472, doi: 10.1007/s00024-015-1049-2.

Baba, T., S. Allgeyer, J. Hossen, P. R. Cummins, H. Tsushima, K. Imai, K. Yamashita, and T. Kato (2017), Accurate numerical simulation of the far-field tsunami caused by the 2011 Tohoku earthquake, including the effects of Boussinesq dispersion, seawater density stratification, elastic loading, and gravitational potential change, *Ocean Modelling*, *111*, 46–54, doi: 10.1016/j.ocemod.2017.01.002.

Barrett, R., M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst (1994), *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, chap. 3, Society for Industrial and Applied Mathematics, doi: 10.1137/1.9781611971538.

Bates, P. D., M. S. Horritt, and T. J. Fewtrell (2010), A simple inertial formulation of the shallow water equations for efficient two-dimensional flood inundation modelling, *Journal of Hydrology*, *387*(1), 33–45, doi: 10.1016/j.jhydrol.2010.03.027.

Briggs, M. J., C. E. Synolakis, G. S. Harkins, and D. R. Green (1995), Laboratory experiments of tsunami runup on a circular island, *pure and applied geophysics*, *144*(3), 569–593.

Casulli, V. (1999), A semi-implicit finite difference method for non-hydrostatic, free-surface flows, *International Journal for Numerical Methods in Fluids*, *30*(4), 425–440, doi: 10.1002/(SICI)1097-0363(19990630)30:4⟨425::AID-FLD847⟩3.0.CO;2-D.

Chen, Q., R. A. Dalrymple, J. T. Kirby, A. B. Kennedy, and M. C. Haller (1999), Boussinesq modeling of a rip current system, *Journal of Geophysical Research: Oceans*, *104*(C9), 20,617–20,637, doi: 10.1029/1999JC900154.

Chen, Q., J. T. Kirby, R. A. Dalrymple, A. B. Kennedy, and A. Chawla (2000), Boussinesq modeling of wave transformation, breaking, and runup. II: 2D, *Journal of Waterway, Port, Coastal, and Ocean Engineering*, *126*(1), 48–56, doi: 10.1061/(ASCE)0733-950X(2000)126:1(48).

Chiang, C. M., M. Stiassnie, and D. K.-P. Yue (2005), *Theory and Applications of Ocean Surface Waves*, chap. 12, World Scientific, doi: 10.1142/5566.

Cho, Y.-S., and J.-M. Kim (2009), Moving boundary treatment in run-up process of tsunami, *Journal of Coastal Research*, pp. 482–486.

Choi, Y.-K., F. Shi, M. Malej, and J. M. Smith (2018), Performance of various shock-capturing-type reconstruction schemes in the Boussinesq wave model, FUNWAVE-TVD, *Ocean Modelling*, *131*, 86–100, doi: 10.1016/j.ocemod.2018.09.004.

de Almeida, G. A. M., P. Bates, J. E. Freer, and M. Souvignet (2012), Improving the stability of a simple formulation of the shallow water equations for 2-d flood modeling, *Water Resources Research*, *48*(5), doi: 10.1029/2011WR011570.

Fujii, Y., K. Satake, S. Sakai, M. Shinohara, and T. Kanazawa (2011), Tsunami source of the 2011 off the Pacific coast of Tohoku earthquake, *Earth, Planets and Space*, *63*(7), 815–820.

Fujiwara, T., S. Kodaira, T. No, Y. Kaiho, N. Takahashi, and Y. Kaneda (2011), The 2011 Tohoku-Oki earthquake: displacement reaching the trench axis, *Science*, *334*(6060), 1240–1240, doi: 10.1126/science.1211554.

Glimsdal, S., G. K. Pedersen, C. B. Harbitz, and F. Løvholt (2013), Dispersion of tsunamis: does it really matter?, *Natural Hazards and Earth System Sciences*, *13*(6), 1507–1526, doi: 10.5194/nhess-13-1507-2013.

Harig, S., Chaeroni, W. S. Pranowo, and J. Behrens (2008), Tsunami simulations on several scales, *Ocean Dynamics*, *58*(5), 429–440, doi: 10.1007/s10236-008-0162-5.

Heidarzadeh, M., S. Murotani, K. Satake, T. Ishibe, and A. R. Gusman (2016), Source model of the 16 September 2015 Illapel, Chile, Mw 8.4 earthquake based on teleseismic and tsunami data, *Geophysical Research Letters*, *43*(2), 643–650, doi: 10.1002/2015GL067297.

Horrillo, J., Z. Kowalik, and Y. Shigihara (2006), Wave dispersion study in the indian ocean-tsunami of december 26, 2004, *Marine Geodesy*, *29*(3), 149–166, doi: 10.1080/01490410600939140.

Imamura, F. (1996), Simulation of wave-packet propagation along sloping beach by TUNAMI-code, in *Long-Wave Runup Models*, edited by H. Yeh, P. Liu, and C. Synolakis, pp. 231–241, World Scientific.

Kajiura, K. (1963), The leading wave of a tsunami, *Bulletin of the Earthquake Research Institute, University of Tokyo*, *41*(3), 535–571.

KÂNOĞLU, U., and C. E. SYNOLAKIS (1998), Long wave runup on piecewise linear topographies, *Journal of Fluid Mechanics*, *374*, 1–28, doi: 10.1017/S0022112098002468.

Kennedy, A. B., Q. Chen, J. T. Kirby, and R. A. Dalrymple (2000), Boussinesq modeling of wave transformation, breaking, and runup. I: 1D, *Journal of Waterway, Port, Coastal, and Ocean Engineering*, *126*(1), 39–47, doi: 10.1061/(ASCE)0733-950X(2000)126:1(39).

Koçyigit, M. B., R. A. Falconer, and B. Lin (2002), Three-dimensional numerical modelling of free surface flows with non-hydrostatic pressure, *International Journal for Numerical Methods in Fluids*, *40*(9), 1145–1162, doi: 10.1002/fld.376.

Kowalik, Z., W. Knight, T. Logan, and P. Whitmore (2005), Numerical modeling of the global tsunami: Indonesian tsunami of 26 December 2004, *Science of Tsunami Hazards*, *23*(1), 40–56.

Larsen, J., and H. Dancy (1983), Open boundaries in short wave simulations — a new approach, *Coastal Engineering*, *7*(3), 285–297, doi: 10.1016/0378-3839(83)90022-4.

Lax, P., and B. Wendroff (1960), Systems of conservation laws, *Communications on Pure and Applied Mathematics*, *13*(2), 217–237, doi: 10.1002/cpa.3160130205.

LeVeque, R. J., D. L. George, and M. J. Berger (2011), Tsunami modelling with adaptively refined finite volume methods, *Acta Numerica*, *20*, 211–289, doi: 10.1017/S0962492911000043.

Liu, P., S. Woo, and Y. Cho (1998), Computer programs for tsunami propagation and inundation, *Technical report*, Cornell University, Ithaca, N.Y.

Liu, P. L. F., Y.-S. Cho, M. J. Briggs, U. Kanoglu, and C. E. Synolakis (1995), Runup of solitary waves on a circular island, *Journal of Fluid Mechanics*, *302*, 259–285, doi: 10.1017/S0022112095004095.

Løvholt, F., and G. Pedersen (2009), Instabilities of Boussinesq models in non-uniform depth, *International Journal for Numerical Methods in Fluids*, *61*(6), 606–637, doi: 10.1002/fld.1968.

Lynett, P., and P. L.-F. Liu (2004), A two-layer approach to wave modelling, *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, *460*(2049), 2637–2669, doi: 10.1098/rspa.2004.1305.

Lynett, P. J., T.-R. Wu, and P. L.-F. Liu (2002), Modeling wave runup with depth-integrated equations, *Coastal Engineering*, *46*(2), 89–107, doi: 10.1016/S0378-3839(02)00043-1.

Ma, G., F. Shi, and J. T. Kirby (2012), Shock-capturing non-hydrostatic model for fully dispersive surface wave processes, *Ocean Modelling*, *43-44*, 22–35, doi: 10.1016/j.ocemod.2011.12.002.

Oishi, Y., F. Imamura, and D. Sugawara (2015), Near-field tsunami inundation forecast using the parallel TUNAMI-N2 model: Application to the 2011 Tohoku-Oki earthquake combined with source inversions, *Geophysical Research Letters*, *42*(4), 1083–1091, doi: 10.1002/2014GL062577.

Okada, Y. (1985), Surface deformation due to shear and tensile faults in a half-space, *Bulletin of the Seismological Society of America*, *75*(4), 1135–1154, doi: 10.1785/BSSA0750041135.

Peregrine, D. H. (1967), Long waves on a beach, *Journal of Fluid Mechanics*, *27*(4), 815–827, doi: 10.1017/S0022112067002605.

Ruetsch, G., and M. Fatica (2024), *CUDA Fortran for scientists and engineers: best practices for efficient CUDA Fortran programming*, Elsevier.

Saito, T., Y. Ito, D. Inazu, and R. Hino (2011), Tsunami source of the 2011 Tohoku-Oki earthquake, Japan: Inversion analysis based on dispersive tsunami simulations, *Geophysical Research Letters*, *38*(7), doi: 10.1029/2011GL049089.

Satake, K., Y. Fujii, T. Harada, and Y. Namegaya (2013), Time and space distribution of coseismic slip of the 2011 Tohoku earthquake as inferred from tsunami waveform data, *Bulletin of the Seismological Society of America*, *103*(2B), 1473–1492, doi: 10.1785/0120120122.

Shapiro, R. (1970), Smoothing, filtering, and boundary effects, *Reviews of Geophysics*, *8*(2), 359–387, doi: 10.1029/RG008i002p00359.

Shi, F., J. T. Kirby, J. C. Harris, J. D. Geiman, and S. T. Grilli (2012), A high-order adaptive time-stepping TVD solver for Boussinesq modeling of breaking waves and coastal inundation, *Ocean Modelling*, *43-44*, 36–51, doi: 10.1016/j.ocemod.2011.12.004.

Shi, F., J. T. Kirby, B. Tehranirad, J. C. Harris, Y.-K. Choi, and M. Malej (2016), FUNWAVE-TVD fully nonlinear Boussinesq wave model with TVD solver - documentation and user's manual (version 3.0), *Center Appl. Coastal Res., Univ. Delaware (2016)*.

Sridharan, B., D. Gurivindapalli, S. N. Kuiry, V. K. Mali, N. Nithila Devi, P. D. Bates, and D. Sen (2020), Explicit expression of weighting factor for improved estimation of numerical flux in local inertial models, *Water Resources Research*, *56*(7), e2020WR027,357, doi: https://doi.org/10.1029/2020WR027357, e2020WR027357 2020WR027357.

Stelling, G., and M. Zijlema (2003), An accurate and efficient finite-difference algorithm for non-hydrostatic free-surface flow with application to wave propagation, *International Journal for Numerical Methods in Fluids*, *43*(1), 1–23, doi: 10.1002/fld.595.

Tanioka, Y., and K. Satake (1996), Tsunami generation by horizontal displacement of ocean bottom, *Geophysical Research Letters*, *23*(8), 861–864, doi: 10.1029/96GL00736.

Thacker, W. C. (1981), Some exact solutions to the nonlinear shallow-water wave equations, *Journal of Fluid Mechanics*, *107*, 499–508, doi: 10.1017/S0022112081001882.

Titov, V. V., and F. I. González (1997), Implementation and testing of the method of splitting tsunami (MOST) model, in *NOAA Technical Memorandum ERL PMEL-112*.

Titov, V. V., and C. E. Synolakis (1998), Numerical modeling of tidal wave runup, *Journal of Waterway, Port, Coastal, and Ocean Engineering*, *124*(4), 157–171, doi: 10.1061/(ASCE) 0733-950X(1998)124:4(157).

van der Vorst, H. A. (1992), Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, *SIAM Journal on Scientific and Statistical Computing*, *13*(2), 631–644, doi: 10.1137/0913035.

Wang, X., and P. L.-F. Liu (2006), An analysis of 2004 Sumatra earthquake fault plane mechanisms and Indian Ocean tsunami, *Journal of Hydraulic Research*, *44*(2), 147–154, doi: 10.1080/00221686.2006.9521671.

Wessel, P., W. H. F. Smith, R. Scharroo, J. Luis, and F. Wobbe (2013), Generic mapping tools: Improved version released, *Eos, Transactions American Geophysical Union*, *94*(45), 409–410, doi: 10.1002/2013EO450001.

Yamazaki, Y., Z. Kowalik, and K. F. Cheung (2009), Depth-integrated, non-hydrostatic model for wave breaking and run-up, *International Journal for Numerical Methods in Fluids*, *61*(5), 473–497, doi: 10.1002/fld.1952.

Yamazaki, Y., K. F. Cheung, and Z. Kowalik (2011), Depth-integrated, non-hydrostatic model with grid nesting for tsunami generation, propagation, and run-up, *International Journal for Numerical Methods in Fluids*, *67*(12), 2081–2107, doi: 10.1002/fld.2485.

Yeh, H., F. Imamura, C. Synolakis, Y. Tsuji, P. Liu, and S. Shi (1993), The Flores Island tsunamis, *Eos, Transactions American Geophysical Union*, *74*(33), 369–373, doi: 10.1029/93EO00381.

Yuan, Y., F. Shi, J. T. Kirby, and F. Yu (2020), Funwave-gpu: Multiple-gpu acceleration of a boussinesq-type wave model, *Journal of Advances in Modeling Earth Systems*, *12*(5), e2019MS001,957, doi: https://doi.org/10.1029/2019MS001957, e2019MS001957 10.1029/2019MS001957.

Zhang, C., J. T. Kirby, F. Shi, G. Ma, and S. T. Grilli (2021), A two-layer non-hydrostatic landslide model for tsunami generation on irregular bathymetry. 1. theoretical basis, *Ocean Modelling*, *159*, 101,749, doi: https://doi.org/10.1016/j.ocemod.2020.101749.

Zhu, Y., C. An, H. Yu, W. Zhang, and X. Chen (2024), High-resolution tsunami hazard assessment for the guangdong-hong kong-macao greater bay area based on a non-hydrostatic tsunami model, *Science China Earth Sciences*, *67*(7), 2326–2351, doi: 10.1007/s11430-023-1300-9.

Zijlema, M., and G. Stelling (2008), Efficient computation of surf zone waves using the nonlinear shallow water equations with non-hydrostatic pressure, *Coastal Engineering*, *55*(10), 780–790, doi: 10.1016/j.coastaleng.2008.02.020.

Zijlema, M., G. Stelling, and P. Smit (2011), SWASH: An operational public domain code for simulating wave fields and rapidly varied flows in coastal waters, *Coastal Engineering*, *58*(10), 992–1012, doi: 10.1016/j.coastaleng.2011.05.015.