Parallel Exponential Time Differencing Methods for Geophysical Flow Simulations

Rihui Lan[a], Wei Leng[b], Zhu Wang[a], Lili Ju[a,*], Max Gunzburger[c]

[a]*Department of Mathematics, University of South Carolina, Columbia, SC 29208*
[b]*State Key Laboratory of Scientific and Engineering Computing,*
Chinese Academy of Sciences, Beijing 100190, China
[c]*Department of Scientific Computing, Florida State University, Tallahassee, FL 32306*

Abstract

Two ocean models are considered for geophysical flow simulations: the multi-layer shallow water equations and the multi-layer primitive equations. For the former, we investigate the parallel performance of exponential time differencing (ETD) methods, including exponential Rosenbrock-Euler, ETD2wave, and B-ETD2wave. For the latter, we take advantage of the splitting of barotropic and baroclinic modes and propose a new two-level method in which an ETD method is applied to solve the fast barotropic mode. These methods could improve the computational efficiency of numerical simulations because ETD methods allow for much larger time step sizes than traditional explicit time-stepping techniques that are commonly used in existing computational ocean models. Several standard benchmark tests for ocean modeling are performed and comparison of the numerical results demonstrate a great potential of applying the parallel ETD methods for simulating real-world geophysical flows.

Keywords: Shallow water equations, primitive equations, exponential time differencing, parallel computing, barotropic/baroclinic splitting

1. Introduction

Without a doubt, the ocean is vitally important to everyone and significantly affects our daily life. To understand and predict ocean circulation, many numerical models have been developed based on fundamental laws of physics with a focus on the properties of geophysical flows. So far, the primitive equations, built upon the Boussinesq approximation, are widely used to model global or regional oceanic circulation, including the velocity, depth (or layer thickness), pressure, and tracers such as temperature, salinity, or chemicals. Due to the large aspect ratio, the ocean model can usually be simplified by regarding the ocean as a single-layer or a stack of immiscible layers for which each one is characterized by a constant density. The layered model is more accurate in representing vertical profiles and is often used to model the actual ocean. The reader is referred to [5] for additional details. On the other hand, numerical simulations of the layered models become more challenging as the oceanic dynamical systems are of very large scales. In addition, oceanic flows are impacted by many other factors, including the coupling of external and internal gravity waves

---

and planetary rotation which results in ocean dynamics involving multiple time scales. Therefore, to achieve accurate and stable numerical schemes, one has to choose small time step schemes such as in explicit Runge-Kutta methods, which have good parallel scalability and are commonly used in existing numerical ocean models.

To better handle the multiple time scales, some techniques based on the splitting strategies have been developed. For instance, in [10, 33], Günther and Sandu designed a generalized additive Runge-Kutta (GARK) method which allows fast and slow modes to use different stages and time step sizes. However, GARK needs extra effort devoted to the coupling conditions between different modes. In [1], Bleck and Smith applied a splitting idea to predict the Atlantic ocean dynamics under the isopycnal coordinate system. This approach separates the ocean motion into two modes: the barotropic (fast) mode and baroclinic (slow) mode. The fast mode is obtained via vertical averaging and the slow mode consists of the difference between the original velocity and the fast mode. Based on Bleck and Smith's work, in [11, 12, 13, 14], Higdon et al. developed a two-level time-stepping method, one of the split-explicit time-stepping methods, to speed up the simulation . In their approach, the baroclinic mode is first advanced with a large time step size and after that, an iterative scheme is applied to determine the barotropic mode. However, the sub-stepping could still be time consuming. Therefore, to further improve the computational efficiency, in this work we introduce a new two-level method that replaces the sub-stepping for the barotropic mode with an exponential time differencing (ETD) method.

Exponential-integrator based methods, as an alternative to explicit or implicit time stepping, were introduced as early as the 1960s [3, 29] to solve autonomous systems of first-order ordinary differential equations (ODEs). The systems are solved exactly via the variation of constants formulas or integrating factors, after which the temporal integrals of the matrix exponentials are approximated. These methods did not gain much attention until the early 1990's because of the difficulty encountered in evaluating exponential functions of matrices. Thanks to the progress in computer science and numerical linear algebra [7, 15, 16, 23, 32, 34], this method has received a renewed interest in solving various large systems of stiff semilinear or nonlinear systems [4, 17, 40]. As for its application to climate sciences, Clancy and Pudykiewicz investigated the potential of ETD methods in atmospheric models. They concluded that these schemes are more efficient than the semi-implicit methods by allowing far longer time-steps, but one needs to efficiently calculate the products of matrix exponentials and vectors. In [8], Gaudreault and Pudykiewicz utilized Incomplete Orthogonalization Method (IOM) to evaluate the matrix exponential functions. Recently, the ETD methods were applied to speed up the simulations of the rotating multi-layer shallow water model in [28, 22]. Pieper et al.[28] reformulated the original equations into a Hamiltonian framework, then developed the ETD-wave and the barotropic ETD methods. By comparing the average simulated years per real day, they numerically showed the ETD methods are more efficient than the classical explicit Runge-Kutta methods. Whereas only sequential algorithms are addressed in those works, as we know, parallel computing is a powerful tool to accelerate scientific computing and have, necessarily, been widely used in numerical climate models. Hence, in this paper, we will further implement and investigate the performance of parallel ETD methods based on domain decomposition. In addition, there has not been much work applying ETD methods to the primitive equations, with the most related research being that of Calandrini et al.[2], in which the authors propose an ETD method to speed up the tracer equations in the primitive equations system, and use a second-order ETD method for solving the dynamics. As mentioned earlier, we will also attempt to replace the sub-steppings for the barotropic mode in the two-level time-stepping method with ETD.

In this paper we mainly focus on the parallel implementation of the ETD methods for simulating two ocean models: the multi-layer shallow water equations and the multi-layer primitive equations.

The computational settings and testbeds we used are taken from *MPAS-Ocean* [27], a numerical ocean model developed at the Los Alamos National Laboratory for the simulation of the ocean system across scales in time and space. The rest of paper is outlined as follows. We first present the two types of ocean dynamics equations in Section 2. For spatial discretization, we apply the TRiSK scheme that is briefly introduced in Section 3. Section 4 covers the illustration of different ETD methods for solving these two models and their parallel implementation. Numerical experiments are performed and discussed in Section 5. Finally, some conclusions are drawn in Section 6.

2. Ocean dynamics and related models

In this section we briefly review two widely-used ocean models: one is governed by the multi-layer rotating shallow water equations (SWEs), the other is by the multi-layer primitive equations. For the case of shallow water equations, we consider the equations reduced by depth integration from the primitive equations and their variation in the framework of Hamiltonian systems developed in [28]. For the primitive equations, we specifically consider the ones used in *MPAS-Ocean* ([27]) which are the incompressible Boussinesq equations in hydrostatic balance and include several tracer equations.

2.1. Shallow water equations

For geophysical flow with a vertical dimension much smaller than the horizontal scale, the shallow water equations can be used to describe the fluid motion. This is a typical situation when fluid flows in the ocean, coastal regions, estuaries and rivers are concerned. The single-layer rotating shallow water model defined on a surface Ω is governed by:

$$\begin{cases} \dfrac{\partial h}{\partial t} = -\nabla \cdot (h\boldsymbol{u}), \\ \dfrac{\partial \boldsymbol{u}}{\partial t} = -\nabla \left(K[\boldsymbol{u}] + g(h+b) \right) - q[h,\boldsymbol{u}]\hat{\boldsymbol{k}} \times (h\boldsymbol{u}) + \mathcal{G}(h,u), \end{cases} \tag{1}$$

where $h = h(t,\boldsymbol{x})$ is the fluid thickness, $\boldsymbol{u} = \boldsymbol{u}(t,\boldsymbol{x})$ is the velocity tangential to the surface Ω (i.e., $\boldsymbol{u} \cdot \hat{\boldsymbol{k}} = 0$) and the velocity is subject to zero normal flow boundary condition (i.e., $\boldsymbol{u} \cdot \hat{n} = 0$ on $\partial\Omega$). In addition, $K[\boldsymbol{u}] = |\boldsymbol{u}|^2/2$ is the kinetic energy, $\hat{\boldsymbol{k}} \times \boldsymbol{u}$ is the perpendicular velocity which is usually denoted by $\boldsymbol{u}^\perp$, and $q[h,\boldsymbol{u}] = (\hat{\boldsymbol{k}} \cdot \nabla \times \boldsymbol{u} + f)/h$ is the potential vorticity with f being the Coriolis parameter. As discussed in [28], the SWEs can be reformulated in the Hamiltonian framework by introducing the following Hamiltonian

$$\mathcal{H}[h,\boldsymbol{u}] = \int_\Omega \left(hK[\boldsymbol{u}] + \frac{g}{2}(h+b)^2 \right) \, \mathrm{d}\mathbf{x}. \tag{2}$$

Let $V = (h,\boldsymbol{u})$ belong to the solution space $\mathcal{X} = L^2(\Omega) \times (L^2(\Omega))^2$, that is equipped with the inner product, for any $\phi = (\phi_h, \phi_{\boldsymbol{u}}), \psi = (\psi_h, \psi_{\boldsymbol{u}}) \in \mathcal{X}$,

$$(\phi, \psi)_\mathcal{X} = \int_\Omega (\phi_h \psi_h + \phi_{\boldsymbol{u}} \cdot \psi_{\boldsymbol{u}}) \, \mathrm{d}\mathbf{x}.$$

Then the functional derivative of $\mathcal{H}$ is given by

$$\delta\mathcal{H}[V] = \frac{\delta\mathcal{H}}{\delta V}[V] = \begin{bmatrix} K[\boldsymbol{u}] + g(h+b) \\ h\boldsymbol{u} \end{bmatrix}, \tag{3}$$

3

and its directional derivative is

$$\mathcal{H}'[V;W] = \left(\frac{\delta\mathcal{H}}{\delta V}[V], W\right)_{\mathcal{X}} = \int_\Omega ((K[\boldsymbol{u}] + g(h+b))h_w + h\boldsymbol{u}\cdot\boldsymbol{u}_w)\,\mathrm{d}\mathbf{x} \tag{4}$$

for any $W = (h_w, \boldsymbol{u}_w)$. Define the skew-symmetric operator $\mathcal{J}$ as

$$\mathcal{J}[h,\boldsymbol{u}] := \begin{bmatrix} 0 & -\nabla\cdot \\ -\nabla & -q[h,\boldsymbol{u}]\hat{\boldsymbol{k}}\times \end{bmatrix}. \tag{5}$$

Then the SWEs (1) can be recast in the Hamiltonian framework as:

$$\frac{\partial V}{\partial t} = \mathcal{J}\delta\mathcal{H}[V] + \begin{bmatrix} 0 \\ \mathcal{G}[h,\boldsymbol{u}] \end{bmatrix}. \tag{6}$$

2.1.1. Multi-layer shallow water equations

Besides the ambient rotation, stratification effects play an essential role in geophysical fluid dynamics. Layered models include the density differences that depict the stratified fluid as a finite number of moving layers, stacked one upon another, with an in-layer constant density. Such models can be derived by partitioning the vertical range into L segments and imposing distinct density values in different layers that increase downward (i.e., $\rho_i < \rho_{i+1}, i = 1, \ldots, L-1$). The multi-layer SWEs on Ω in correspondence to the classic SWEs (1) read: for $k = 1, \ldots, L$,

$$\begin{cases} \dfrac{\partial h_k}{\partial t} = -\nabla\cdot(h_k\boldsymbol{u}_k), \\ \dfrac{\partial \boldsymbol{u}_k}{\partial t} = -\nabla\left(K[\boldsymbol{u}_k] + (g/\rho_k)\,p_k[h]\right) - q[h_k,\boldsymbol{u}_k]\hat{\boldsymbol{k}}\times(h_k\boldsymbol{u}_k) + \mathcal{G}_k\left(h,\boldsymbol{u}\right), \end{cases} \tag{7}$$

where h_k and $\boldsymbol{u}_k$ are the thickness and velocity of layer k, and $K[\boldsymbol{u}_k]$ and $q[h_k,\boldsymbol{u}_k]$ are defined similarly as those in the single-layer case.

Define $V = (h,\boldsymbol{u}) \in \mathcal{X}^L$, the Cartesian production of $\mathcal{X}$ for L times, which consists of the variables of all layers. For any $\phi = (\phi_h, \phi_{\boldsymbol{u}}), \psi = (\psi_h, \psi_{\boldsymbol{u}}) \in \mathcal{X}^L$, the corresponding inner produce is defined by

$$(\phi,\psi)_{\mathcal{X}^L} = \sum_{k=1}^L \int_\Omega \left(\phi_k^h\psi_k^h + \phi_k^{\boldsymbol{u}}\cdot\psi_k^{\boldsymbol{u}}\right)\,\mathrm{d}\mathbf{x}.$$

Different from the single-layer case, the pressure term $p_k[h]$ combines all the layers above layer k, defined as

$$p_k[h] = \rho_k\eta_{k+1}[h] + \sum_{l=1}^k \rho_l h_l = \rho_k\eta_k[h] + \sum_{l=1}^{k-1} \rho_l h_l, \tag{8}$$

where η_k is the layer coordinates defined by $\eta_{L+1} = b$ with b being the bathymetry, and for $k = 1, \ldots, L$, $\eta_k[h] = b + \sum_{l=k}^L h_l$. Following the constructions in [28, 37], the multi-layer Hamiltonian $\mathcal{H}$ is defined as

$$\mathcal{H}[V] = \sum_{l=1}^k \rho_k \int_\Omega \left(h_k K[\boldsymbol{u}_k] + gh_k(\eta_{k+1}[h] + h_k/2)\right)\,\mathrm{d}\mathbf{x}, \tag{9}$$

and the skew-symmetric operator $\mathcal{J}[V]$ is

$$\mathcal{J}[V] := \mathrm{diag}_{k=1,\ldots,L}\frac{1}{\rho_k}\mathcal{J}[h_k,\boldsymbol{u}_k]. \tag{10}$$

4

2.1.2. Jacobian Matrix

For the special ETD-Euler method (i.e. the exponential Rosenbrock-Euler) to be discussed in Section 4, one needs to evaluate the Jacobian matrix of the equation (7). Hence, we present the Jacobian matrix $J_k(h, \boldsymbol{u})$ here, that is,

$$J_k(h, \boldsymbol{u}) = \begin{bmatrix} -\nabla \cdot (\bullet \boldsymbol{u}_k) & -\nabla \cdot (h_k \bullet) \\ J_{2,1} & J_{2,2} \end{bmatrix}, \tag{11}$$

where the two entries $J_{2,1}$ and $J_{2,2}$ are defined by

$$J_{2,1} = -\nabla \left(g/\rho_k \frac{\partial P_k}{\partial h} \right) - \frac{\partial q[h_k, \boldsymbol{u}_k]}{\partial h} \hat{\boldsymbol{k}} \times (h_k \boldsymbol{u}_k) - q[h_k, \boldsymbol{u}_k] \hat{\boldsymbol{k}} \times (\bullet \boldsymbol{u}_k),$$

$$J_{2,2} = -\nabla \boldsymbol{u}_k - \frac{\partial q[h_k, \boldsymbol{u}_k]}{\partial \boldsymbol{u}} \hat{\boldsymbol{k}} \times (h_k \boldsymbol{u}_k) - q[h_k, \boldsymbol{u}_k] \hat{\boldsymbol{k}} \times (h_k \bullet),$$

and $\bullet$ is the placeholder to fill in the elements when the Jacobian matrix works on a vector. The derivatives with respect to layer i's thickness and normal velocity are respectively

$$\frac{\partial P_k}{\partial h_i} = \begin{cases} \rho_k, & \text{if } i \geq k, \\ \rho_i, & \text{if } i < k, \end{cases}$$

$$\frac{\partial q[h_k, \boldsymbol{u}_k]}{\partial h_i} = -\frac{\hat{\boldsymbol{k}} \cdot \nabla \times \boldsymbol{u}_k + f}{h_k^2} \delta_{i,k},$$

$$\frac{\partial q[h_k, \boldsymbol{u}_k]}{\partial \boldsymbol{u}_i} = \frac{\hat{\boldsymbol{k}} \cdot \nabla \times \bullet}{h_k} \delta_{i,k}.$$

Notice that evaluations of $-\nabla \cdot (\bullet \boldsymbol{u}_k)$, $-\nabla \cdot (h_k \bullet)$ and $J_{2,2}$ only involve quantities from layer k and, for calculating $J_{2,1}$, only the term $\frac{\partial P_k}{\partial h}$ uses the thickness and densities from the other layers. Therefore, we can further split the Jacobian matrix as the sum of two terms:

$$J_k(h, \boldsymbol{u}) = J_k^P(h, \boldsymbol{u}) + J_k^R(h, \boldsymbol{u}), \tag{12}$$

where

$$J_k^P(h, \boldsymbol{u}) = \begin{bmatrix} 0 & 0 \\ -\nabla \left(g/\rho_k \frac{\partial P_k}{\partial h} \right) & 0 \end{bmatrix}, \qquad J_k^R = J_k(h, \boldsymbol{u}) - J_k^P(h, \boldsymbol{u}).$$

Such splitting significantly simplifies the coding effort and improves the efficiency in the construction of the Jacobian matrices in our parallel implementation.

2.2. Primitive equations

Different from the SWEs, the primitive equations describe the incompressible Boussinesq equations in hydrostatic balance [30, 27, 26]. The single-layer model consists of the following equations:

- Thickness equation:

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\boldsymbol{u}) + \frac{\partial}{\partial z}(hw) = 0. \tag{13}$$

- Momentum equation:

$$\frac{\partial \boldsymbol{u}}{\partial t} + \frac{1}{2}\nabla |\boldsymbol{u}|^2 + (\boldsymbol{k} \cdot \nabla \times \boldsymbol{u})\boldsymbol{u}^\perp + f\boldsymbol{u}^\perp + w\frac{\partial \boldsymbol{u}}{\partial z} = -\frac{1}{\rho_0}\nabla p + \nu_h \nabla^2 \boldsymbol{u} + \frac{\partial}{\partial z}(\nu_v \frac{\partial \boldsymbol{u}}{\partial z}). \tag{14}$$

5

- Tracer equations:

$$\frac{\partial h\varphi}{\partial t} + \nabla \cdot (h\varphi \boldsymbol{u}) + \frac{\partial}{\partial z}(h\varphi w) = \nabla \cdot (h\kappa_h \nabla \varphi) + h\frac{\partial}{\partial z}(\kappa_v \frac{\partial \varphi}{\partial z}). \tag{15}$$

- Hydrostatic condition:

$$p = p^s(x,y) + \int_z^{z^s} \rho g \, \mathrm{d}z'. \tag{16}$$

- Equation of state:

$$\rho = f_{\mathrm{eos}}(\Theta, S, p). \tag{17}$$

The definitions of variables are listed in Table 1.

Variables	Definition
h	Layer thickness
$\boldsymbol{u}$, w	Horizontal and vertical velocity
p, $p^s(x,y)$	Pressure and the top pressure of layer k
Θ	Potential temperature
S	Salinity
ρ, ρ_0	Density and referential density
g	Gravity acceleration
φ	Generic tracer (Θ or S)
z, z^s	Vertical coordinate and the z-location of the top boundary location
ν_h, ν_v	Viscosity
κ_h, κ_v	Tracer diffusion
f	Coriolis parameter
f_{eos}	Equation of state

Table 1: List of definitions of variables.

2.2.1. Multi-layer primitive equations

Taking the consideration of the stratification effects, let us partition the vertical range into L segments and discretize the primitive equations in the vertical direction. First, we refer to the methods from [2, 30] for the vertical discretization. Let ϕ_k be the vertical average of the generic variable ϕ in layer k and adopt the following notations from [30, Appendix A.3]:

$$\begin{cases} \overline{(\phi^t_:)}_k^m = (\phi_k^t + \phi_{k+1}^t)/2, \\ \overline{(\phi^m_:)}_k^t = (\phi_{k-1}^m + \phi_k^m)/2, \\ \delta z_k^m(\phi^t_:) = \dfrac{\phi_k^t - \phi_{k+1}^t}{h_k}, \\ \delta z_k^t(\phi^m_:) = \dfrac{\phi_{k-1}^m - \phi_k^m}{\overline{(h)}_k^t}, \end{cases}$$

where the superscripts m and t denote the location as the middle and top of layer k in the vertical. Then one can discretize the vertical derivatives as

$$
\begin{cases}
\dfrac{\partial}{\partial z}(h_k w_k) \approx w_k^t - w_{k+1}^t, \\[2mm]
w_k \dfrac{\partial \boldsymbol{u}_k}{\partial z} \approx \overline{(w_:^t \delta z^t(\boldsymbol{u}_:))}_k^m, \\[2mm]
\dfrac{\partial}{\partial z}(h_k \varphi_k w_k) \approx \overline{(\varphi_:^m)}_k^t w_k^t - \overline{(\varphi_:^m)}_{k+1}^t w_{k+1}^t,
\end{cases}
$$

and

$$
\begin{cases}
\dfrac{\partial}{\partial z}\left(\nu_v \dfrac{\partial \boldsymbol{u}_k}{\partial z}\right) \approx \delta z_k^m (\nu_v \delta z^t(\boldsymbol{u}_:)), \\[2mm]
\dfrac{\partial}{\partial z}\left(\kappa_v \dfrac{\partial \varphi_k}{\partial z}\right) \approx \delta z_k^m (\kappa_v \delta z^t(\varphi_:)).
\end{cases}
$$

Now, we introduce the multi-layer primitive equations with the vertical discretization: for $k = 1, \ldots, L$,

$$
\begin{cases}
\dfrac{\partial h_k}{\partial t} + \nabla \cdot (h_k \boldsymbol{u}_k) + w_k^t - w_{k+1}^t = 0, & (18) \\[3mm]
\dfrac{\partial \boldsymbol{u}_k}{\partial t} + \dfrac{1}{2}\nabla |\boldsymbol{u}_k|^2 + (\boldsymbol{k} \cdot \nabla \times \boldsymbol{u}_k)\boldsymbol{u}_k^\perp + f \boldsymbol{u}_k^\perp + \overline{(w_:^t \delta z^t(\boldsymbol{u}_:))}_k^m = -\dfrac{1}{\rho_0}\nabla p_k + \nu_h \nabla^2 \boldsymbol{u}_k \\[3mm]
\qquad\qquad + \delta z_k^m (\nu_v \delta z^t(\boldsymbol{u}_:)), & (19) \\[3mm]
\dfrac{\partial h_k \varphi_k}{\partial t} + \nabla \cdot (h_k \varphi_k \boldsymbol{u}_k) + \overline{(\varphi_:^m)}_k^t w_k^t - \overline{(\varphi_:^m)}_{k+1}^t w_{k+1}^t = \nabla \cdot (h_k \kappa_h \nabla \varphi_k) + h_k \delta z_k^m (\kappa_v \delta z^t(\varphi_:)), & (20) \\[3mm]
p_k = p_k^s(x, y) + \displaystyle\sum_{l=1}^{k-1} \rho_l g h_l + \dfrac{1}{2}\rho_k g h_k, & (21) \\[3mm]
\rho_k = f_{\mathrm{eos}}(\Theta_k, S_k, p_k). & (22)
\end{cases}
$$

The computation of w_k^t via (18) is highly dependent on the chosen vertical coordinate. For example, w_k^t is set to zero in the idealized isopycnal vertical coordinate since there is no vertical transport. For the z-level vertical coordinate, all layers have a fixed thickness except for the top layer (k=1). Hence w_k^t is computed with $\frac{\partial h_k}{\partial t} = 0$ for $k > 1$. For the z-star vertical coordinate, the layer thickness is proportional to the sea surface height (SSH). Therefore, w_k^t is non-zero for all the layers when the ocean is not at rest. M. Petersen et al. [26] considered the arbitrary Lagrangian-Eulerian (ALE) vertical coordinate to obtain w_k^t in a unified way, in which a new quantity, named h_k^{ALE}, is introduced to represent the expected new layer thickness, i.e.,

$$
w_k^t = w_{k+1}^t - \nabla \cdot (h_k \boldsymbol{u}_k) - \frac{h_k^{\mathrm{ALE}} - h_k}{\Delta t}.
$$

We refer the readers to [26] for more details. In this paper, we will consider the z-star vertical coordinate in our numerical experiments, which is the default setting in MPAS-Ocean.

As mentioned in the introduction, the primitive equations involve the barotropic mode for the dynamics and require a small time-stepping size for stable simulations with explicit time stepping schemes. To design an efficient numerical scheme, transformed dynamics equations are derived to treat the barotropic (fast) mode and baroclinic (slow) modes separately. Define the barotropic velocity $\overline{\boldsymbol{u}}$ and baroclinic velocity $\boldsymbol{u}_k'$ as

$$
\overline{\boldsymbol{u}} = \sum_{k=1}^{L} h_k \boldsymbol{u}_k \bigg/ \sum_{k=1}^{L} h_k \ \text{ and } \ \boldsymbol{u}_k' = \boldsymbol{u}_k - \overline{\boldsymbol{u}}, \qquad k = 1, \ldots, L. \tag{23}
$$

Let $\zeta = \sum_{k=1}^{L} h_k + b - H$ be the sea surface height (SSH) with b being the bottom elevation above a reference level and H the height of the reference surface, and let Δz_1 be the top layer's original thickness. To derive the equation for ζ, we consider the velocity (u, v, w) where u, v are z-independent, then integrate the continuity equation over the entire fluid depth yields

$$0 = \int_{b}^{b+\sum_{k=1}^{L} h_k} \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) dz = \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \int_{b}^{b+\sum_{k=1}^{L} h_k} dz + w \big|_{b}^{b+\sum_{k=1}^{L} h_k}. \tag{24}$$

The kinematic conditions [5, Chapter 7] of the geophysical flow give the following two boundary conditions

$$w(z = b + \textstyle\sum_{k=1}^{L} h_k) = \frac{\partial \zeta}{\partial t} + u \frac{\partial \zeta}{\partial x} + v \frac{\partial \zeta}{\partial y}, \quad w(z = b) = u \frac{\partial b}{\partial x} + v \frac{\partial b}{\partial y}.$$

Plugging the above two equations into (24) gives

$$\frac{\partial \zeta}{\partial t} = -\frac{\partial (u \sum_{k=1}^{L} h_k)}{\partial x} - \frac{\partial (v \sum_{k=1}^{L} h_k)}{\partial y}. \tag{25}$$

For more details about ζ, the readers are referred to [5, Chapter 7]. By calculating the the layer-thickness-weighted average of (19) and combining (25), the barotropic thickness and momentum equations are then given by

$$
\begin{cases}
\dfrac{\partial \zeta}{\partial t} = -\nabla \cdot \left(\overline{\boldsymbol{u}} \sum_{k=1}^{L} h_k \right), & (26) \\[4mm]
\dfrac{\partial \overline{\boldsymbol{u}}}{\partial t} = -f \overline{\boldsymbol{u}}^{\perp} - g \nabla \zeta + \overline{\boldsymbol{G}}(\boldsymbol{u}_k, \boldsymbol{u}_k'), & (27)
\end{cases}
$$

where $\overline{\boldsymbol{G}}(\boldsymbol{u}_k, \boldsymbol{u}_k')$ includes all remaining terms in the barotropic equation. Subtracting (27) from (19) yields the baroclinic momentum equation: for $k = 1, \ldots, L$,

$$\frac{\partial \boldsymbol{u}_k'}{\partial t} = -f \boldsymbol{u}_k'^{\perp} + \boldsymbol{T}^{\boldsymbol{u}}(\boldsymbol{u}_k, w_k, p_k) + \delta z_k^m (\nu_v \delta z^t(\boldsymbol{u}_:)) + g \nabla \zeta - \overline{\boldsymbol{G}}(\boldsymbol{u}_k, \boldsymbol{u}_k'), \tag{28}$$

where

$$\boldsymbol{T}^{\boldsymbol{u}}(\boldsymbol{u}_k, \; w_k, p_k) = -\frac{1}{2} \nabla |\boldsymbol{u}_k|^2 - (\boldsymbol{k} \cdot \nabla \times \boldsymbol{u}_k) \boldsymbol{u}_k^{\perp} - \overline{(w_: ^t \delta z^t(\boldsymbol{u}_:))}_k^m - \frac{1}{\rho_0} \nabla p_k + \nu_h \nabla^2 \boldsymbol{u}_k.$$

3. The TRiSK schemes for spatial discretizations

The multi-layer rotating SWE (7), the layer thickness equation (18), and the barotropic and baroclinic equations (26-28) will be discretized by the mimetic TRiSK scheme [39, 31, 38] as done in *MPAS-Ocean* to ensure the properties of the continuous system, such as energy and mass conservations on unstructured, locally orthogonal meshes. The TRiSK scheme utilizes the staggered C-grid, which is comprised of spherical centroidal Voronoi tessellation (SCVT) as the primal grid and its corresponding Delaunay triangulation as the dual grid, see Figure 1 for illustration. The layer thickness h_k, the vertical velocity w, the pressure p and the tracer φ are defined on the cell's center $\boldsymbol{x}_i$, the horizontal velocity $\boldsymbol{u}_k$ is defined at $\boldsymbol{x}_e$, the intersection point between the primal and dual edges. The vorticity is located at the center $\boldsymbol{x}_v$ of the dual grid's cell. We list all the notations for the mesh information from [24, 31] in Tables 2-4, and the corresponding differential operators in Table 5.

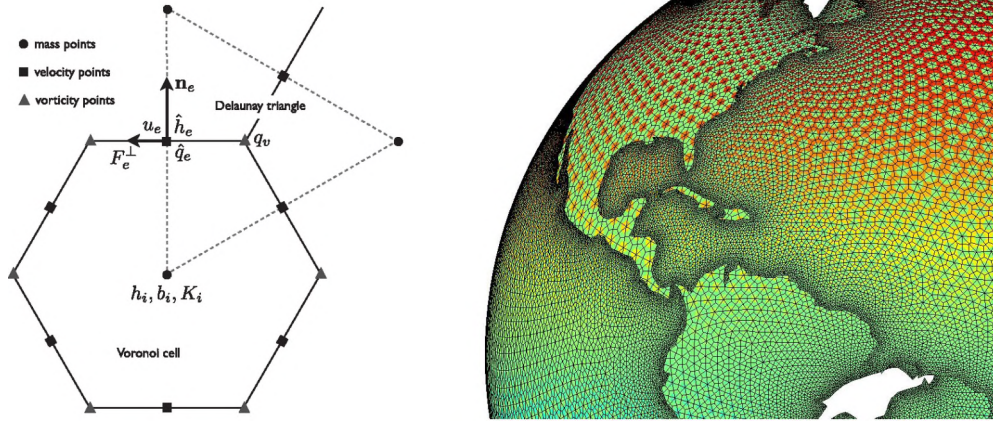The discretization of the multi-layer SWE and primitive equations in TRiSK scheme are given as follow:

Figure 1: An illustration of spherical centroidal Voronoi tessellation and its dual Delaunay triangulation for ocean modeling.

Notations	Definition
$\boldsymbol{x}_i$	Centers of primal mesh cells
$\boldsymbol{x}_v$	Centers of dual mesh cells
$\boldsymbol{x}_e$	The intersection point between the primal and dual edges
P_i	Primal cells corresponding to $\boldsymbol{x}_i$
D_v	Dual cells corresponding to $\boldsymbol{x}_v$
l_e	Length of the primal edge e
d_e	Length of the dual edge intersecting e
A_i	Area of the primal cell P_i
A_v^d	Area of the dual cell D_v
A_e^e	Area associated with the primal edge e: $A_e^e = 1/2 l_e d_e$

Table 2: List of the mesh elements and quantities.

Notations	Definition
$e \in EC(i)$	Set of edges of the primal cell P_i
$i \in CE(e)$	Two primal cells either side of primal edge e
$e \in EV(v)$	Set of primal edges sharing vertex v
$i \in CV(v)$	Set of primal cells having v as their vertex
$v \in VE(e)$	Two endpoints of primal edge e
$e' \in ECP(e)$	Set of primal edges nearby e

Table 3: List of the mesh connectivity.

- Multi-layer shallow water equations

$$\begin{cases} \frac{\partial h_k}{\partial t} = -\nabla_{E \to I} \cdot \left(\{h_k\}_E * \boldsymbol{u}_k \right), \\ \frac{\partial \boldsymbol{u}_k}{\partial t} = Q[h_k, \boldsymbol{u}_k](\{h_k\}_E * \boldsymbol{u}_k) - \nabla_{I \to E} \left(K\lfloor\{\boldsymbol{u}_k\}_I\rfloor + (g/\rho_k)p_k\lfloor\{h\}_E\rfloor \right) + G_k(h, \boldsymbol{u}), \end{cases} \quad (29)$$

where $*$ denotes the point- or component-wise product. The term $Q[h_k, \boldsymbol{u}_k](\cdot)$ is an operator and varies accordingly with the time-stepping methods. Let

$$q[h_k, \boldsymbol{u}_k] = \left((\hat{\boldsymbol{k}} \cdot \nabla\times)_{E \to V} \boldsymbol{u}_k + f \right) / \{h_k\}_V,$$

9

Notations	Definition
$\boldsymbol{n}_e$	The unit vector at $\boldsymbol{x}_e$ normal to e in the direction corresponding to positive u_e
$\boldsymbol{t}_e$	The unit tangential vector at $\boldsymbol{x}_e$: $\boldsymbol{t}_e = \boldsymbol{k} \times \boldsymbol{n}_e$
$n_{e,i}$	Normal indicator function $$n_{e,i} = \begin{cases} 1 & \text{if } \boldsymbol{n}_e \text{ is an outward normal of } P_i, \\ -1 & \text{otherwise.} \end{cases}$$
$t_{e,v}$	Tangential indicator function $$t_{e,v} = \begin{cases} 1 & \text{if } v \text{ is an outward normal of } \boldsymbol{k} \times \boldsymbol{n}_e, \\ -1 & \text{otherwise.} \end{cases}$$ i.e., counterclockwise circulation about vertex v contributes positively to the vorticity at v.

Table 4: List of the normal and tangential vectors.

Divergence :	$(\nabla \cdot_{E \to I} \boldsymbol{y})_e$	$=$	$(1/A_i) \sum_{e \in EC(i)} n_{e,i} l_e \boldsymbol{y}_e$
Gradient :	$(\nabla_{I \to E} \boldsymbol{y})_i$	$=$	$(1/d_e) \sum_{i \in CE(e)} -n_{e,i} \boldsymbol{y}_i$
Curl:	$((\hat{\boldsymbol{k}} \cdot \nabla \times)_{E \to V} \boldsymbol{y})_v$	$=$	$(1/A_v) \sum_{e \in EV(v)} t_{e,v} d_e \boldsymbol{y}_e$
Perpendicular Gradient:	$(\nabla^{\perp}{}_{V \to E} \boldsymbol{y})_e$	$=$	$(1/l_e) \sum_{v \in VE(e)} t_{e,v} \boldsymbol{y}_v$
Perpendicular Flux:	$(\hat{\boldsymbol{k}} \times_{E \to E} \boldsymbol{y})_e$	$=$	$(1/d_e) \sum_{e' \in ECP(e)} w_{e,e'} l_e \boldsymbol{y}_{e'}$
Cell to Vertex interpolation:	$\{\boldsymbol{y}\}_{V,v}$	$=$	$(1/A_v) \sum_{i \in CV(v)} R_{i,v} A_i \boldsymbol{y}_i$
Vertex to Edge interpolation:	$\{\boldsymbol{y}\}_{E,e}$	$=$	$\sum_{v \in VE(e)} \boldsymbol{y}_v / 2$
Edge to Cell interpolation:	$\{\boldsymbol{y}\}_{I,i}$	$=$	$(1/A_i) \sum_{e \in EC(i)} \boldsymbol{y}_e A_e / 2$
Cell to Edge interpolation:	$\{\boldsymbol{y}\}_{E,e}$	$=$	$\sum_{i \in CE(e)} \boldsymbol{y}_i / 2$

Table 5: Summary of the discrete operators given concretely in terms of geometrical quantities [31].

then we define

$$Q[h_k, \boldsymbol{u}_k](\boldsymbol{y}) = \begin{cases} -\{q[h_k, \boldsymbol{u}_k]\}_E * \left(\hat{\boldsymbol{k}} \times_{E \to E} \boldsymbol{y}\right), & \text{in ETD-Euler,} \\ \frac{1}{2}\left(\{q[h_k, \boldsymbol{u}_k]\}_E * \left(\hat{\boldsymbol{k}} \times_{E \to E} \boldsymbol{y}\right) + \hat{\boldsymbol{k}} \times_{E \to E} \{q[h_k, \boldsymbol{u}_k]\}_E * \boldsymbol{y}\right), & \text{in ETD-Wave.} \end{cases}$$

The second form comes from the skew-symmetry of $\hat{\boldsymbol{k}} \times_{E \to E}$.

- Multi-layer primitive equations

$$\begin{cases} \frac{\partial h_k}{\partial t} + \nabla_{E \to I} \cdot (\{h_k\}_E * \boldsymbol{u}_k) + w_k^t - w_{k+1}^t = 0, \\[2mm] \frac{\partial \boldsymbol{u}_k}{\partial t} + \nabla_{I \to E} K\left[\{\boldsymbol{u}_k\}_I\right] + \left(\left\{(\hat{\boldsymbol{k}} \cdot \nabla \times)_{E \to V} \boldsymbol{u}_k\right\}_E + f\right) * \left(\hat{\boldsymbol{k}} \times_{E \to E} \boldsymbol{u}_k\right) + \overline{(w_{\cdot}^t \delta z^t(\boldsymbol{u}_{\cdot}))}_k^m \\ \qquad = -\frac{1}{\rho_0} \nabla_{I \to E} p_k + \nabla_{I \to E} (\nabla_{E \to I} \cdot \boldsymbol{u}_k) + \nabla^{\perp}{}_{V \to E} (\hat{\boldsymbol{k}} \cdot \nabla \times)_{E \to V} \boldsymbol{u}_k + \delta z_k^m (\nu_v \delta z^t(\boldsymbol{u}_{\cdot})), \\[2mm] \frac{\partial h_k \varphi_k}{\partial t} + \nabla_{E \to I} \cdot (\{h_k \varphi_k\}_E * \boldsymbol{u}_k) + \overline{(\varphi_{\cdot}^m)}_k^t w_k^t - \overline{(\varphi_{\cdot}^m)}_{k+1}^t w_{k+1}^t \\ \qquad = \nabla_{E \to I} \cdot (\kappa_h \{h_k\}_E * \nabla_{I \to E} \varphi_k) + h_k \delta z_k^m (\kappa_v \delta z^t(\varphi_{\cdot})), \end{cases} \tag{30}$$

where we applied $\nabla^2 \boldsymbol{u} = \nabla(\nabla \cdot \boldsymbol{u}) + \boldsymbol{k} \times \nabla(\boldsymbol{k} \cdot \nabla \times \boldsymbol{u})$.

4. Exponential time differencing methods and parallel implementations

In this section, we first briefly go over and discuss the classic ETD method [18] and the recently derived ETD-wave and Barotropic ETD-wave methods in [28] for the multi-layer SWEs. Then, we propose a new two-level ETD method to solve the multi-layer primitive equations in the barotropic/baroclinic splitting fashion. Finally we discuss their parallel implementation through domain decomposition.

4.1. Classic ETD schemes

Consider the following nonlinear parabolic equation on the time interval $[t_n, t_n + \tau]$:

$$\partial_t V = F[V] = A_n V + N[V], \tag{31}$$

where V is the unknown variable, $F[V]$ is the flux, A_n is a linear operator, and $N[V]$ is a nonlinear term. By using the integrating factor $e^{-t \Delta A_n}$, (31) can be advanced to a future time $t_n + \Delta t$ provided V_n, the value of V at t_n:

$$
\begin{aligned}
V(t_n + \Delta t) &= e^{\Delta t A_n} V_n + \int_{t_n}^{t_n + \Delta t} e^{(t_n + \Delta t - \tau) A_n} N(V(\tau)) \, \mathrm{d}\tau \\
&= V_n + \Delta t \varphi_1(\Delta t A_n) A_n V_n + \Delta t \int_0^1 e^{\Delta t (1 - \theta) A_n} N(V(t_n + \theta \Delta t)) \, \mathrm{d}\theta.
\end{aligned}
\tag{32}
$$

If we replace $N(V(t_n + \theta \Delta t))$ with a polynomial approximation in time

$$N(V(t_n + \theta \Delta t)) \approx \sum_{s=2}^{S} \frac{\theta^{s-1}}{(s-1)!} b_{n,s}, \tag{33}$$

where $b_{n,s}$ approximates the derivatives $\frac{d^{s-1} N(V(t_n + \theta \Delta t))}{d\theta^{s-1}} \big|_{\theta=0}$, then $V(t_{n+1})$ can be approximated with S internal stages as

$$V(t_{n+1}) \approx V_{n+1} = V_n + \Delta t \Big(\varphi_1(\Delta t A_n) F[V_n] + \sum_{s=2}^{S} \varphi_s(\Delta t A_n) b_{n,s} \Big),$$

where the $\varphi-$functions are defined by

$$\varphi_s(z) = \int_0^1 \exp((1 - \sigma) z) \frac{\sigma^{s-1}}{(s-1)!} \, \mathrm{d}\sigma = \sum_{k=0}^{\infty} \frac{z^k}{(k+s)!}. \tag{34}$$

We call the method with $S = 1$, ETD-Euler method, that is

$$V_{n+1} = V_n + \Delta t \varphi_1(\Delta t A_n) F[V_n]. \tag{35}$$

If $A_n = \frac{\partial F[V]}{\partial V} \big|_{t=t_n}$, then (35) is also specially called *exponential Rosenbrock-Euler* method, which is of second order accuracy in time for autonomous systems. Let us focus on the space-discrete case, i.e. (31) is a ODE system where V is a vector and A_n is a matrix. To compute $\varphi_1(\Delta t A_n) F[V_n]$, or more generally $\varphi_s(A_n) b$, we will use Krylov subspace methods [7, 32]. The essential idea is to approximate the matrix A_n with a matrix H_M of smaller size but preserves as much information of A_n as possible. Usually, it can be done in two steps.

- Firstly, one constructs the orthonormal basis $\boldsymbol{V}_M$ of the Krylov subspace

$$K_M(A_n, b) = \text{span}\{b, A_n b, \cdots, A_n^{M-1} b\} \qquad (36)$$

by the Arnoldi process. It also generates an unreduced upper Hessenberg matrix H_M, satisfying the recurrence formula

$$A_n \boldsymbol{V}_M = \boldsymbol{V}_M H_M + \boldsymbol{h}_{M+1,M} \boldsymbol{v}_{M+1} \boldsymbol{e}_M^T, \quad \boldsymbol{V}_M = (v_1, v_2, \cdots, v_M).$$

Hence,

$$\varphi_s(\Delta t A_n) b \approx \boldsymbol{V}_M \varphi_s(\boldsymbol{V}_M^T \Delta t A_n \boldsymbol{V}_M) \boldsymbol{V}_M^T b = \|b\|_0 \boldsymbol{V}_M \varphi_s(\Delta t H_M) \boldsymbol{e}_1, \qquad (37)$$

where $\|b\|_0 = \sqrt{b^T b}$. If the matrix A_n has some (skew-)symmetric properties, then the Arnoldi process can be replaced by the (skew-)Lanczos process to reduce the Gram-Schmidt process and obtain a tridiagonal matrix H_M.

- Secondly, one instead computes $\varphi_s(\Delta t H_M) \boldsymbol{e}_1$ and then $\|b\|_0 \boldsymbol{V}_M \varphi_s(\Delta t H_M) \boldsymbol{e}_1$ as an approximation of $\varphi_s(A_n) b$ due to (37). For that, many methods such as Padé approximation or scaling-doubling method can be applied [23, 15, 34]. In our numerical tests in Section 4.5, we will use the function "dgpadm" from the software package Expokit [34] which uses the scaling-doubling method.

4.2. ETD-wave method for multi-layer SWEs

The linearization of the Hamiltonian formalism (6) of the multi-layer SWEs around the reference state $V^{\text{ref}} = \left(h^{\text{ref}}, \boldsymbol{u}^{\text{ref}} \right)$ is given by

$$\partial_t W = \mathcal{J}'[V^{\text{ref}}; W] \delta \mathcal{H}[V^{\text{ref}}] + \mathcal{J}[V^{\text{ref}}] \delta^2 \mathcal{H}[V^{\text{ref}}] W, \quad W(0) = W_0, \qquad (38)$$

where $V = V^{\text{ref}} + W$ and $W = (h_w, \boldsymbol{u}_w)$ is the perturbation. Picking $\boldsymbol{u}^{\text{ref}} = 0$, then (38) is reduced to

$$\partial_t W = \mathcal{J}[V^{\text{ref}}] \delta^2 \mathcal{H}[V^{\text{ref}}] W = AW, \qquad (39)$$

where $\mathcal{J}$ and $\delta^2 \mathcal{H}$ have the following symmetry properties,

$$(\boldsymbol{W}, \mathcal{J}\boldsymbol{V})_{\mathcal{X}^L} = -(\mathcal{J}\boldsymbol{W}, \boldsymbol{V})_{\mathcal{X}^L}, \quad (\boldsymbol{W}, \delta^2 \mathcal{H} \boldsymbol{V})_{\mathcal{X}^L} = (\delta^2 \mathcal{H} \boldsymbol{W}, \boldsymbol{V})_{\mathcal{X}^L}.$$

In order to apply the symmetry property on the discrete level, we first specify some spaces for the discrete quantities. Let $\boldsymbol{X} = \boldsymbol{X}_I \times \boldsymbol{X}_E \in R^{N_I} \times R^{N_E}$, where N_I and N_E are the numbers of cells and edges on each layer. We assume for each layer $h_k \in \boldsymbol{X}_I$, $\boldsymbol{u}_k \in \boldsymbol{X}_E$, and the whole system solution $(h, \boldsymbol{u}) \in \boldsymbol{X}^L = \boldsymbol{X}_I^L \times \boldsymbol{X}_E^L$. Define the mass matrix $\boldsymbol{M}_{\boldsymbol{X}^L}$ containing L copies of the cell and edge areas on the diagonal, then

$$\boldsymbol{M}_H A = -A^T \boldsymbol{M}_H, \quad \text{where } \boldsymbol{M}_H = \boldsymbol{M}_{\boldsymbol{X}^L} \delta^2 \mathcal{H}.$$

Based on this property, we can introduce the skew-Lancos process to calculate $\varphi_s(\Delta t A_n) b$ at each time step under the $\boldsymbol{M}_{\boldsymbol{X}^L}$−norm, which is defined as

$$(u, v)_{\boldsymbol{M}_{\boldsymbol{X}^L}} = u^T \boldsymbol{M}_{\boldsymbol{X}^L} v,$$

where u, v are two vectors. The method, ETDSwave, is to pick "S" internal stages, and use the approximation (39) with the skew-Lancos process to find the lower dimensional matrix $\boldsymbol{H}_M$, then calculate the matrix exponential function.

4.3. Barotropic ETD-wave method for multi-layer SWEs

The barotropic ETD-wave method for solving multi-layer SWEs approximates the whole system by using some fast modes. It introduces a reduced-layer space

$$\boldsymbol{X}^{\widehat{L}} = \boldsymbol{X}_I^{\widehat{L}} \times \boldsymbol{X}_E^{\widehat{L}}, \quad \text{for } 1 \leq \widehat{L} << L,$$

and define the mapping $\Psi : \boldsymbol{X}^{\widehat{L}} \to \boldsymbol{X}^L$ with the structure

$$\Psi = \begin{bmatrix} \Psi_h & 0 \\ 0 & \Psi_u \end{bmatrix},$$

where Ψ_h and Ψ_u are defined by

$$[\Psi_h \hat{h}]_i = \sum_{j=1}^{\widehat{L}} \Psi_h^{j,i} \hat{h}_{k,i}, \forall \hat{h} \in \boldsymbol{X}_I^{\widehat{L}}$$

$$[\Psi_u \hat{\boldsymbol{u}}]_i = \sum_{j=1}^{\widehat{L}} \Psi_{\boldsymbol{u}}^{j,i} \hat{\boldsymbol{u}}_{k,i}, \forall \hat{\boldsymbol{u}} \in \boldsymbol{X}_E^{\widehat{L}}.$$

Here $\hat{h}_{k,i}$ and $\hat{\boldsymbol{u}}_{k,i}$ are the ith components of $\hat{h}_k$ and $\hat{\boldsymbol{u}}_k$, respectively. $\Psi_h^{j,i}$ and $\Psi_{\boldsymbol{u}}^{j,i}$ correspond to the jth fastest vertical height mode [28, Section 2.3]. Then we define the reduced Hamiltonian as

$$\hat{\mathcal{H}}^{\text{ref}}(\widehat{V}) = \mathcal{H}(\Psi\widehat{V}) = \frac{1}{2}\left(\Psi\widehat{V}, \delta^2\mathcal{H}\Psi\widehat{V}\right)_{\mathcal{X}^L} = \frac{1}{2}\left(\widehat{V}, \delta^2\hat{\mathcal{H}}\widehat{V}\right)_{\mathcal{X}^L}, \tag{40}$$

where $\delta^2\hat{\mathcal{H}}$ is the corresponding reduced Hamiltonian matrix and $\delta^2\hat{\mathcal{H}} = \Psi^T\delta^2\mathcal{H}\Psi$. As we can see that the mapping Ψ is singular, we will define its inverse mapping through the Moore-Penrose pseudoinverse

$$\Psi^\dagger : \boldsymbol{X}^L \to \boldsymbol{X}^{\widehat{L}}, \ \Psi^\dagger = \left(\delta^2\hat{\mathcal{H}}\right)^{-1}\Psi^T\delta^2\mathcal{H}. \tag{41}$$

The following proposition shows $\Psi^\dagger$ optimally projects the full solution V onto the reduced layer space $\boldsymbol{X}^{\widehat{L}}$ under the energy norm $\|\cdot\|_{\delta^2 H}$.

Proposition 1. [28, Proposition 5.1] *The restriction matrix $\Psi^\dagger$ gives the solution to the following minimization problem: for any $V \in \boldsymbol{X}^L$, we have $\Psi^\dagger V = \widehat{V}$ where*

$$\widehat{V} = \underset{\widehat{V} \in \boldsymbol{X}^{\widehat{L}}}{\arg\min} \left\|V - \Psi\widehat{V}\right\|_{\delta^2 H} = \underset{\widehat{V} \in \boldsymbol{X}^{\widehat{L}}}{\arg\min} \frac{1}{2}\left(V - \Psi\widehat{V}, \delta^2 H(V - \Psi\widehat{V})\right)_{\boldsymbol{X}^L}. \tag{42}$$

We can define the orthogonal projection $P : \boldsymbol{X}^L \to \boldsymbol{X}^L$ as

$$\boldsymbol{P} = \Psi\Psi^\dagger, \quad \text{such that } \boldsymbol{P}V = \Psi\widehat{V}.$$

Then, we define

$$A_p = \boldsymbol{P}A\boldsymbol{P} = \Psi\widehat{A}\Psi^\dagger, \quad \text{where } \widehat{A} = \Psi^\dagger A\Psi.$$

Based on the above framework of the reduced layer model, we can reformulate $\varphi_s(A)$ with respect to the reduced matrix $\widehat{A}$.

Proposition 2. [28, Proposition 5.3] *Let $A_p = \boldsymbol{P}A\boldsymbol{P}$ and $\widehat{A} = \Psi^\dagger A \Psi$. Then, it holds for any $s \geq 0$ that*

$$\varphi_s(A_p) = \frac{1}{s!}(Id - \boldsymbol{P}) + \Psi\varphi_s(\widehat{A})\Psi^\dagger. \tag{43}$$

Thereby, we can compute $\varphi_s(\Delta t\widehat{A})\hat{b}_s$ instead of $\varphi_s(\Delta tA_p)b_s$ in the ETD method. Since the barotropic modes usually evolve much faster than the remaining modes in realistic global ocean simulations, we choose the fast barotropic modes to approximate the solutions and ignore the slow modes. The obtained ETD method is referred to as B-ETDSwave methods in [28] with "S" denoting the number of internal stages.

4.4. Two-level ETD method for multi-layer primitive equations

By following the framework of the two-level time-stepping method [13] currently used in MPAS-Ocean, we next propose a new two-level ETD-based method for solving the multi-layer primitive equations, in which the same time step size Δt is able to be used in advancing both barotropic and baroclinic modes. The following three stages will be repeated twice in a predictor-corrector way.

- **Stage 1: Advance the baroclinic mode (3D) explicitly.**

  As suggested in [13], we first ignore the barotropic forcing term $\overline{\boldsymbol{G}}$, and apply the forward Euler method to predict the baroclinic velocity as

  $$\tilde{\boldsymbol{u}}'_{k,n+1} = \boldsymbol{u}'_{k,n} + \Delta t\left(-f\boldsymbol{u}^\perp_{k,n} + \boldsymbol{T^u}(\boldsymbol{u}_{k,n}, w_{k,n}, p_{k,n}) + g\nabla\zeta_n\right). \tag{44}$$

  Due to the layer thickness average free property of the baroclinic velocity, that is,

  $$\sum_{k=1}^{L} h_k\boldsymbol{u}'_{k,t}\Big/\sum_{k=1}^{L} h_k = 0,$$

  we can obtain

  $$\overline{\boldsymbol{G}} = \frac{1}{\Delta t}\sum_{k=1}^{L} h_k\tilde{\boldsymbol{u}}'_{k,n+1}\Big/\sum_{k=1}^{L} h_k, \tag{45}$$

  and then correct the baroclinic velocity by

  $$\boldsymbol{u}'_{k,n+1} = \tilde{\boldsymbol{u}}'_{k,n+1} - \Delta t\overline{\boldsymbol{G}}. \tag{46}$$

- **Stage 2: Compute the barotropic mode (2D) by ETD method.**

  At this stage, we are going to solve (26)-(27). The barotropic forcing term $\overline{\boldsymbol{G}}$ in (27) is obtained from Stage 1. These two equations can be put in the general form:

  $$\frac{\partial V}{\partial t} = -\boldsymbol{F}(V) + \boldsymbol{b}, \tag{47}$$

  where $V = (\zeta, \overline{\boldsymbol{u}})^T$, $F(V) = \left(\nabla \cdot (\overline{\boldsymbol{u}}\sum_{k=1}^{L} h_k), f\overline{\boldsymbol{u}}^\perp + g\nabla\zeta\right)^T$, and $\boldsymbol{b} = \left(0, \overline{\boldsymbol{G}}\right)^T$. The associated Jacobian matrix is

  $$J_n := \frac{\partial F(V)}{\partial V}\Big|_{t=t_n} = \begin{bmatrix} -\nabla \cdot (\bullet\overline{\boldsymbol{u}}_n) & -\nabla \cdot \left(\bullet\sum_{k=1}^{L} h_{n,k}\right) \\ -g\nabla\bullet & -f\boldsymbol{k} \times \bullet \end{bmatrix}. \tag{48}$$

14

According to (32), given V_n, we can advance (47) as

$$\begin{cases} V_{n+1} = V_n + \Delta t \varphi_1(-\Delta t J_n)\left(-\boldsymbol{F}(V_n) + \boldsymbol{b}_n\right), \\ V_{n+1/2} = 1/2(V_n + V_{n+1}), \end{cases} \qquad (49)$$

where we choose the left endpoint rule for the integral. This method is referred to as "ETD-SE1". As mentioned in [21], it essentially subsamples the high-frequency barotropic motions and consequently alias high-frequency energy onto lower frequencies if we only advance the barotropic mode to t_{n+1}. There is one solution suggested in [21] to average over the baroclinic time step, and this is best done by integrating the barotropic equations forward over $2\Delta t$, then average with the solution at t_n. So we will consider the following approximation to calculate $\overline{\boldsymbol{u}}_{n+1}$

$$\begin{cases} V_{n+2} \approx V_n + 2\Delta t \varphi_1(-2\Delta t J_n)\left(-\boldsymbol{F}(V_n) + \boldsymbol{b}_n\right), \\ V_{n+1} = 1/2(V_n + V_{n+2}). \end{cases} \qquad (50)$$

This method is referred to as "ETD-SE2". Accordingly, we will refer the split-explicit methods currently adopted in MPAS-Ocean as "SE1" or "SE2" for advancing the barotropic mode to Δt or $2\Delta t$.

- **Stage 3: Update thickness, tracers, density and pressure by forward-Euler scheme.**

  Notice that the equations belong to the hyperbolic type of advection equations. As suggested in [35], we better choose the transport velocity at the intermediate time level $t_{n+1/2}$. While the transport velocity is split into two parts, we approximate the baroclinic velocity at $t_{n+1/2}$ by averaging the velocities at the two consecutive time steps. However, we will consider the barotropic velocity at t_{n+1} due to the sub-sampling issue mentioned in Stage 2.

Remark 4.1. *To increase its stability, we might also consider the sub-stepping process at Stage 2 in the way it is done in MPAS-Ocean, but it requires less steps because the ETD schemes allow larger time-step sizes than the forward Euler scheme.*

Remark 4.2. *For the vertical diffusion terms in the momentum and tracer equations, we implicitly solve them by the back-ward Euler scheme after the above predictor-corrector iteration. This is referred to as the vertical mixing and fulfilled by MPAS-Ocean.*

4.5. Domain decomposition and parallel implementations

The major computational cost for the above ETD methods lies on evaluations of matrix exponential and vector products. Even if the Krylov subspace methods are used, a large amount of matrix-vector products are still required in both Arnoldi and Lanczos processes. Therefore, parallel implementation of these ETD methods on large distributed systems mainly focuses on parallelizing these products efficiently. Due to its cross-platform portability and high performance, our code is developed with the message passing interface (MPI) [9].

For both layered SWE and primitive equation models, we first partition the target three-dimensional domain into several vertical layers, and each layer shares the same horizontal domain meshed by the SCVTs together with a dual Delaunay triangulation. To achieve high parallelization efficiency, we further decompose the horizontal domain into N_p subdomains, where N_p is the number of the processors. Six horizontal partitions generated by "METIS" [20] are shown in Figures 2 and 3. To avoid the communication between processors due to the pressure, we assign all the vertical layers relating to the same sub-domain to the same processor. More specifically, suppose the Voronoi cell centers and vertices are labeled as $\{N_{i,k} \mid i = 1, \ldots, M, \text{ and } k = 1, \ldots, L\}$, where M is

the number of centers and vertices on the horizontal mesh, L is the number of layers. The indices $1, \ldots, M$ will be separated into N_p groups based on the domain decomposition, i.e., there are N_p index sets: $\tilde{I}_1, \cdots, \tilde{I}_{N_p}$, such that $\tilde{I}_i \cap \tilde{I}_j = \emptyset$, for $i \neq j$. In order to share the information among processors, we need to assign some interface cells from their neighbor subdomains. In our implementation, three outer layer cells will be used. This would lead to new index sets $I_1, \cdots, I_{N_p}$, such that $\tilde{I}_i \subset I_i$. Hence, the ith process is assigned with the indices $\{N_{i,k} \mid i \in I_i, \text{ and } k = 1, \ldots, L\}$. We use the sophisticated parallel package Epetra [6] from Trilinos [41] to take care of the MPI data structure and inter-process communication.
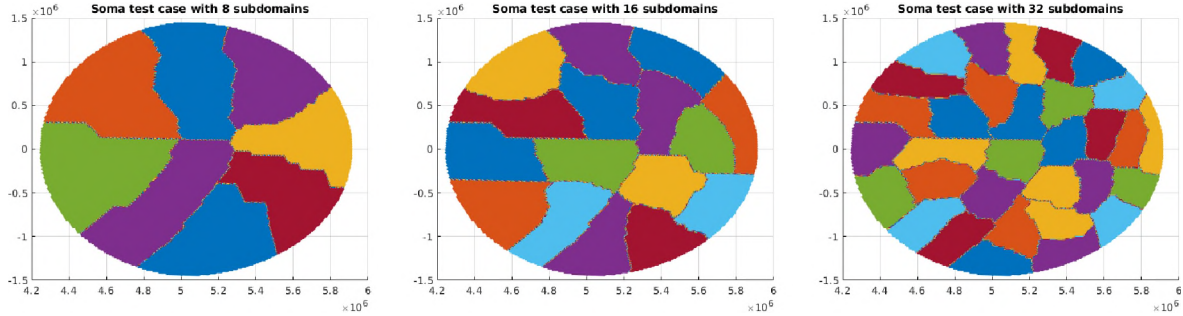
Figure 2: Domain decomposition examples for the SOMA test cases.
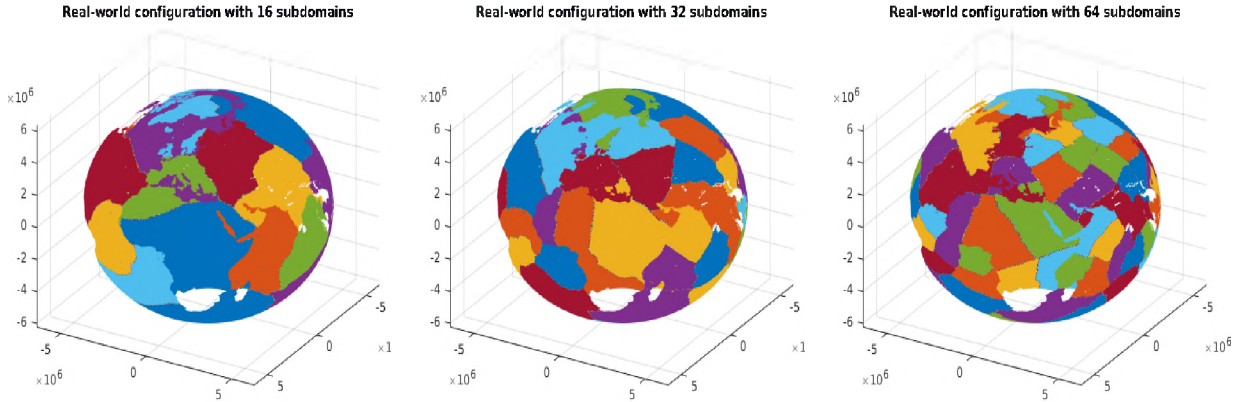
Figure 3: Sample domain decompositions for the global ocean test case.

5. Numerical experiments

In [28], the authors tested ETD2wave and B-ETD2wave methods on multi-layer shallow water equations (7). Hence in this paper, we mainly focus on their parallel implementation. We choose the test cases from MPAS-Ocean Version 7.0. Without special notification, our numerical experiments are carried on a project-owned partition of the Cori at the National Energy Research Scientific Computing Center (NERSC). Cori is a Cray XC40 with a peak performance of about 30 petaflops, which is comprised of 2,388 Intel Xeon "Haswell" processor nodes and 9,688 Intel Xeon Phi "Knight's Landing" (KNL) nodes. Our codes run on "Haswell" processor nodes, where each node has two 16-core Intel Xeon "Haswell" (E5-2698 v3, 2.3 GHz) processors and 128 GB DDR4 2133 MHz memory.

5.1. The SOMA test case for the shallow water equations with three layers

In this test, the spatial domain is a circular basin centered at the point $\boldsymbol{x}_c$ (latitude $\theta_c = 35°$ and longitude $\alpha_c = 0°$) with radius 1250 km, lying on the surface of the sphere of radius $R = 6371.22$ km. The fluid depth in the basin varies from 2.5 km at the center to 100 m on the coastal shelf. The initial interfaces of the three-layer locations are at $\eta_1^0 = 0$ m, $\eta_2^0 = -250$ m, and $\eta_3^0 = -700$ m and the layer densities are $(\rho_1, \rho_2, \rho_3) = (1025, 1027, 1028)$ kg/m^3. Three SCVT meshes of different resolutions on each layer are used:

i. 16 km resolution with 22,007 cells, 66,560 edges and 44,554 vertices;

ii. 8 km resolution with 88,056 cells, 265,245 edges and 177,190 vertices;

iii. 4 km resolution with 352,256 cells, 1,058,922 edges and 706,667 vertices.

The maximum dimension of Krylov subspaces used in ETD methods is set to be 25. We run 15-day simulations with the time step-size $\Delta t = 107$ s by the three parallel ETD methods: Rosenbrock-Euler, ETD2wave, and B-ETD2wave. In addition, we carry out the same simulation using the fourth-order Runge-Kutta (RK4) method with the small time step-size $\Delta t = 10.7$ s and take its results as the benchmark solution for computing errors. The simulated layer thickness h_1 and velocity $\boldsymbol{u}_1$ of the first layer by parallel implementations of the ETD methods are shown in Figure 4. The corresponding differences between these solutions and the RK4 solution are shown in Figure 5, together with the quantitative results of relative l_∞ and average l_2 errors reported in Table 6. As the second order method, ETD2wave has the best performance among these three methods, which is with the smallest errors comparing to RK4. Since B-ETD2wave method is a model reduction method, it considers the reduced model and bears some extra reduction error. Exponential Rosenbrock-Euler has larger errors than ETD2wave method, but smaller errors than B-ETD2wave.
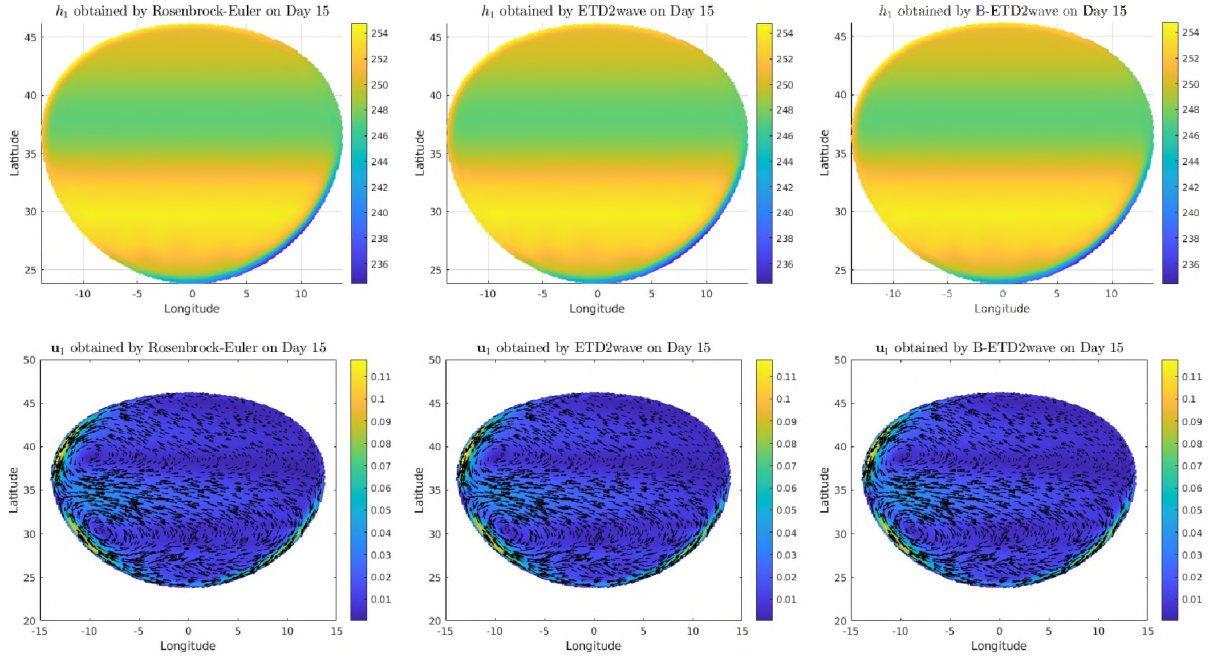

Figure 4: Simulated layer thickness (top row) and velocity (bottom row)) of the first layer on Day 15 by ETD methods for the SOMA test case. Latitude and Longitude are in degrees.
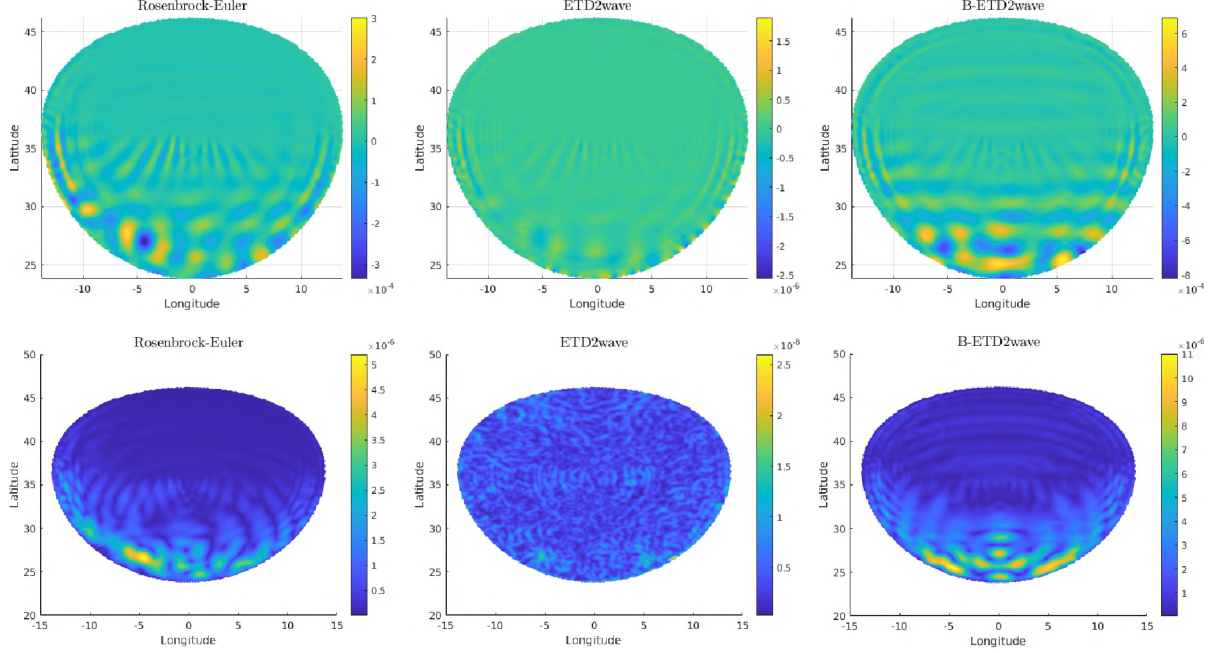
Figure 5: The differences in simulated layer thickness (top row) and velocity (bottom row) of the first layer on Day 15 between the ETD solutions and the RK4 solution for the SOMA test case. Latitude and Longitude are in degrees.

		Rosenbrock-Euler	ETD2wave	B-ETD2wave
Rel. l_∞	h_1	1.3128e-06	1.0066e-08	3.2326e-06
	$\boldsymbol{u}_1$	4.5907e-05	2.7242e-07	9.4878e-05
Avg. l_2	h_1	4.5363e-05	1.3442e-07	1.1972e-04
	$\boldsymbol{u}_1$	6.3922e-07	3.1764e-09	1.6630e-06

Table 6: The relative l_∞ and average l_2 errors of simulated layer thickness and velocity of the first layer on Day 15 by the parallel ETD methods.

To measure the parallel efficiency, we define $E_p = \frac{r \cdot T_r}{p \cdot T_p}$, where p is the number of used cores and T_p is the associated CPU time, r is the minimum number of cores considered in the test case and the corresponding CPU time T_r is regarded as the reference. The simulation performances are reported in Tables 7-9.

Cores	Rosenbrock-Euler		ETD2wave		B-ETD2wave	
	Time (s)	Efficiency	Time (s)	Efficiency	Time (s)	Efficiency
8	194.64	–	76.64	–	27.99	–
16	100.14	92%	43.11	89%	17.87	78%
32	84.82	58%	27.2	70%	12.56	56%
64	35.71	68%	16.10	60%	8.17	43%
128	18.57	66%	10.57	45%	6.61	26%

Table 7: CPU times and parallel efficiency for the SOMA test case: 16 km resolution mesh.

It is seen that the parallel efficiency of the three methods increases as the resolution increases. A higher resolution mesh contains more degrees of freedom (vertices, edges, and cells), which results in more computing tasks to each core and leads to the improvement on parallel efficiency. Because of some model reduction, B-ETD2wave achieves a better overall performance than the other

Cores	Rosenbrock-Euler		ETD2wave		B-ETD2wave	
	Time (s)	Efficiency	Time (s)	Efficiency	Time (s)	Efficiency
8	855.73	-	470.61	-	140.25	-
16	482.32	89%	281.73	84%	78.26	90%
32	343.71	62%	216.39	54%	51.67	68%
64	173.84	62%	93.15	63%	26.47	66%
128	92.20	58%	33.67	87%	14.51	60%
256	51.95	51%	19.39	76%	11.76	37%

Table 8: CPU times and parallel efficiency for the SOMA test case: 8 km resolution mesh.

Cores	Rosenbrock-Euler		ETD2wave		B-ETD2wave	
	Time (s)	Efficiency	Time (s)	Efficiency	Time (s)	Efficiency
16	2316.33	-	1414.57	-	528.82	-
32	1657.91	70%	1069.01	66%	364.62	73%
64	737.64	79%	478.49	74%	157.13	84%
128	369.09	78%	221.02	80%	53.99	122%
256	188.15	77%	101.67	87%	29.01	114%

Table 9: CPU times and parallel efficiency for the SOMA test case: 4 km resolution mesh.

two. For instance, when the smaller number of cores are used, the B-ETD2wave scheme is about five times faster than the exponential Rosenbrock-Euler and three times faster than ETD2wave. Since the exponential Rosenbrock-Euler requires more local computing on the Arnoldi process, its parallel efficiency is better than that of the other two methods for the 16 and 8 km resolution cases. On the 4 km resolution mesh, its parallel performance becomes worse than the other two methods. A potential reason is that we take the computing time of 16 processes as the reference. Exponential Rosenbrock-Euler has a better balance between the inter-process communication and local computing with 16 processes on Cori. Eventually, the inter-process communication wins over local computing, which can be seen from the overall decreasing efficiency. Whereas for the other two methods, 16 processes are not their optimal task decomposition and thus, their performance keeps increasing and even gives super-linear speedups.

5.2. The baroclinic eddies test case for the primitive equations with twenty layers

Next we consider a test case of the primitive equations with twenty layers from MPAS-Ocean [25, 26, 30] imported from [19]. The domain consists of a horizontally periodic channel of latitudinal extent 440 km and longitudinal extent 160 km, with a flat bottom of 1 km vertical depth. The channel is on a f-plane [5, 26] with the Coriolis parameter $f = 1.2 \times 10^{-4}$ s^{-1}. The initial temperature decreases downward in the meridional direction. A cosine shape temperature perturbation with a wavelength of 120 km in the zonal direction is used to investigate the baroclinic instability and diagnose spurious mixing via the resting potential energy (RPE) [19]. It is defined by

$$\text{RPE} = g \int z \rho^*(z) \, dV,$$

where z is the z-location, and $\rho^*(z)$ is the sorted density which is constant horizontally and monotonically increasing along the depth. The way to compute RPE is given by

$$\text{RPE} = g \sum_j z^*(j) \rho^*(j) V^*(j).$$

To reduce the effect of irrelevances to the mixing, it is commonly to consider the normalized RPE, i.e.,

$$\frac{\text{RPE}(t) - \text{RPE}(0)}{\text{RPE}(0)}.$$

Thus, we will measure the normalized RPE for each case. For more details on RPE, we refer to [26, Section 2.3].

5.2.1. Accuracy test

In this subsection, we first test the accuracy of the proposed methods and carry out 15 days simulation on a 10-km-resolution SCVT mesh containing 3,920 cells, 11,840 edges, and 7,920 vertices. The horizontal viscosity is set as $\nu_h = 10 \text{ m}^2\text{s}^{-1}$, and choose all other parameters as the default value in MPAS-Ocean. Considering that it is a relatively small scale problem, we run all numerical experiments in this test with 6 cores. The maximum dimension of Krylov subspaces used in ETD methods is set to be 25. The numerical results for the surface temperature produced by ETD-SE2, SE2, and RK4 schemes are shown in Figures 6, 7, and 8 respectively, and all of them perform very similarly. In addition, we test ETD-SE1 for comparison and present its results in Figure 9. It is observed that the temperature obtain by ETD-SE1 is less diffusive than that of ETD-SE2 and RK4. The inaccuracy is mainly caused by the sub-sampling influence discussed in Section 4.4.
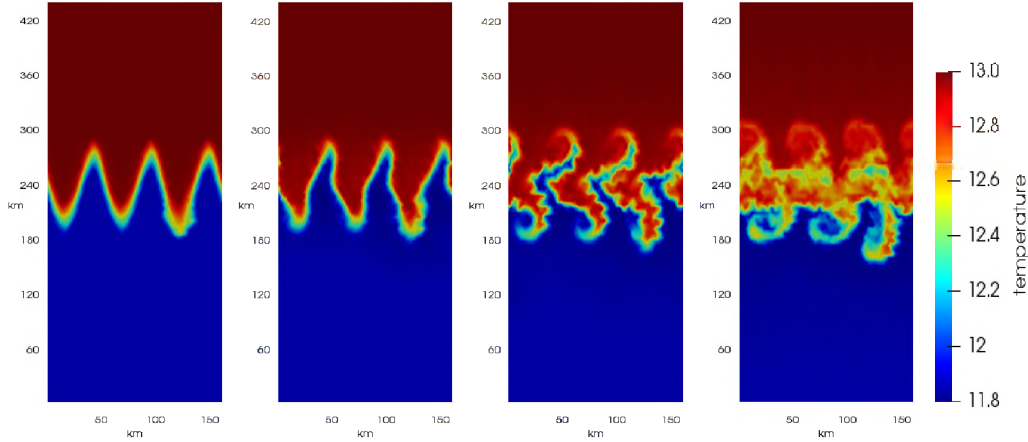
Figure 6: Simulated surface temperatures by ETD-SE2 with $\Delta t = 60$ s for the baroclinic eddies test case. From left to right are the results for day 1, 5, 10 and 15.

To test the accuracy of these methods, we choose the classical RK4 with $\Delta t = 1$ s as the benchmark solution and compare the errors of the surface temperature at 1 hour. The relative l_∞ errors of simulated surface temperature by ETD-SE2 and SE2 are listed in Table 10. Since we develop our code in MPAS-Ocean by replacing its barotropic sub-stepping part with the ETD method, we compare the time cost of computing the barotropic velocity (Stage 2 in Section 4.4) with four cores. The results are listed in Table 10, in which Δt is the time step size for both baroclinic and barotropic modes in ETD-SE2, Δt and $\Delta_{\text{btr}}t$ are the ones for the baroclinic and barotropic modes, respectively, in SE2. From Table 10, we observe that both ETD-SE2 and SE2 only achieves the first-order accuracy in time. It is probably because one advances the barotropic mode over $2\Delta t$ to avoid sub-sampling its high frequency in the current MPAS-Ocean numerical model. Since MPAS is based on the MPDATA method, it suggests using the velocity at the middle time level, $t_{n+1/2}$, to solve the transport equation. But we input the barotropic velocity at t_{n+1} instead of
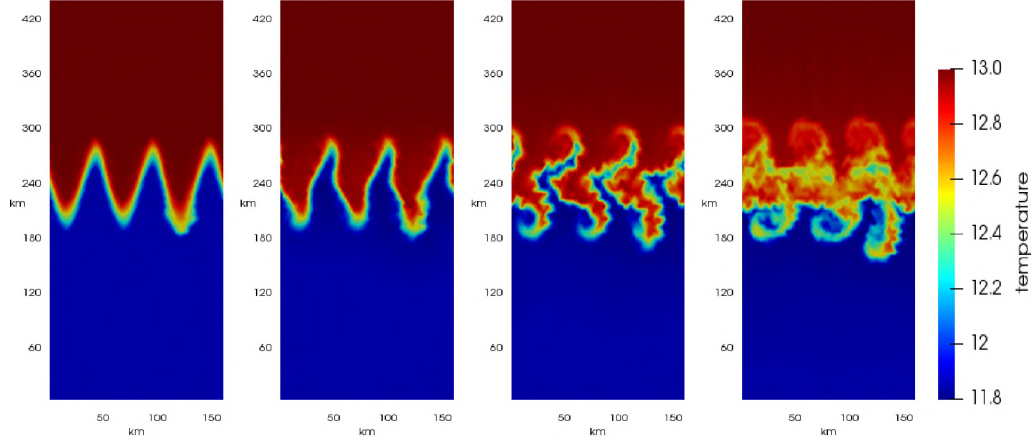
20

Figure 7: Simulated surface temperatures by SE2 in MPAS with $\Delta t = 60$ s and $\Delta t_{\text{btr}} = 4s$ for the baroclinic eddies test case. From left to right are the results for day 1, 5, 10 and 15.
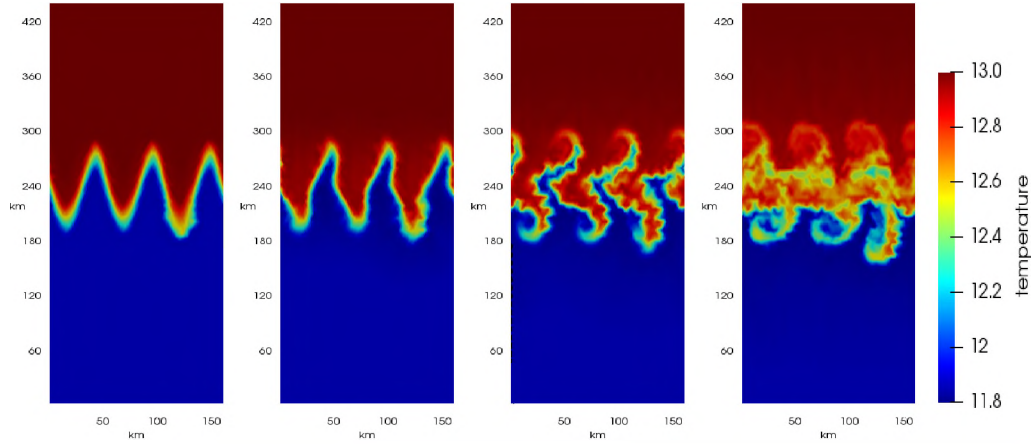

Figure 8: Simulated surface temperatures by RK4 in MPAS with $\Delta t = 15$ s for the baroclinic eddies test case. From left to right are the results for day 1, 5, 10 and 15.
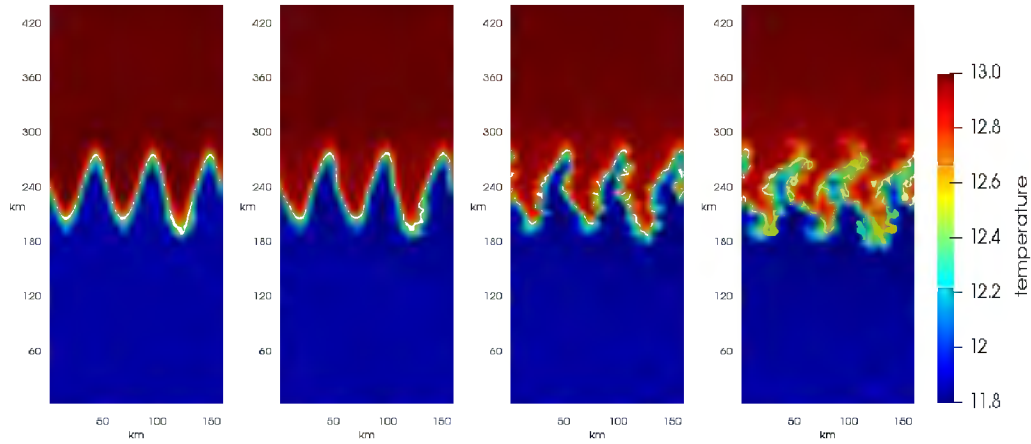

Figure 9: Simulated surface temperatures by ETD-SE1 with $\Delta t = 60$ s for the baroclinic eddies test case. From left to right are the results for day 1, 5, 10 and 15.

21

$t_{n+1/2}$, so it might degrade the accuracy. On the other hand, we can see that our ETD-SE2 spends less time than SE2 (about a half of the time cost of SE2) thus is more efficient in this case. This gain comes from the single step at solving the barotropic velocity, while the original sub-stepping method in SE2 needs multiple small time steps.

$\Delta t(\Delta_{\text{btr}}t)$	ETD-SE2			SE2		
	Error	Rate	Time (s)	Error	Rate	Time (s)
60s(8s)	6.1029e-05	–	0.37	6.1164e-05	–	0.69
30s(4s)	3.0709e-05	0.99	0.75	3.0776e-05	0.99	1.37
15s(2s)	1.4884e-05	1.04	1.46	1.4927e-05	1.04	2.75
8s(1s)	7.4326e-06	1.00	2.86	7.4590e-06	1.00	5.40

Table 10: The relative l_∞ errors and convergence rates of simulated surface temperature and the corresponding CPU times by ETD-SE2 and SE2 with 6 cores for the baroclinic eddies test case.

5.2.2. Long-term simulation and the resting potential energy

In this subsection, we perform a 200-day simulation with 4 km resolution mesh containing 5,040 cells, 15,200 edges, and 10,160 vertices. We use the default setting of the parameter values in MPAS-Ocean, and vary the horizontal viscosity from 1 to 200 m^{2}s^{-1}. The simulated surface temperatures under five different horizontal viscosities, $\nu_h = 200, 20, 10, 5$ and 1, produced by ETD-SE2 with $\Delta t = 25$ s, are shown in Figure 10. In addition, we plot and compare in Figure 11 the corresponding evolutions of the normalized resting potential energy (RPE) of the fluids under these horizontal viscosities. Since the kinetic energy gets lower at higher viscosity, the eddies get larger and the mixing slower along the increasing of the viscosity. Consequently, the higher the viscosity is, the slower the RPE increases along the time. All these phenomena are clearly observed in the simulations.

5.3. The global ocean test case for the primitive equations with sixty layers

In this part, let us consider a test with global real-world configuration. The mesh provided by MPAS-Ocean, EC60to30, varies from 30 km resolution at the equator and poles to 60 km resolution at the mid-latitudes and uses 60 vertical layers. The grid contains 235,160 cells, 714,274 edges, and 478,835 vertices on each layer. The Coriolis parameter $f = 2\omega \sin(\Phi)$, where ω is the angular velocity of Earth's rotation, Φ is the latitude [5]. The initial conditions for temperature and salinity are interpolated from the Polar Science Center Hydrographic Climatology, version 3 [36] that are presented in Figure 12. We run the 15 days simulation using ETD-SE2 with $\Delta t = 60$ s and show the resulting surface temperature and salinity in Figure 13. The increments of the two states generated by ETD-SE2 and RK4 with $\Delta t = 60$ s are shown and compared in Figures 14, and the results show that they perform very similarly.

In addition, we test the parallel efficiency of ETD-SE2 and SE2 by running 1 hour simulation and evaluating E_p as done in Section 5.1. To obtain the comparable accuracy, we again set the maximum dimension of Krylov subspace to be 25 in ETD-SE2 and take the time step $\Delta t = 60$ s. In SE2, we take the same $\Delta t = 60$ s for the baroclinic mode and $\Delta t_{btr} = 1$ s for the baratropic mode. Comparing with the benchmark solution of RK4 with $\Delta t = 5$ s, the l_∞ error of simulated surface temperature by ETD-SE2 and SE2 are 7.5515e-05 and 7.5773e-05, respectively. Their computation times with different number of cores up to 256 are reported in Table 11, from which we observe that ETD-SE2 costs less computational times than SE2 in these cases (i.e., when the number of used cores is not very large), but it has relatively lower parallel efficiency. That is partially due to the fact that ETD-SE2 needs more inter-process communication due to the Arnoldi process,
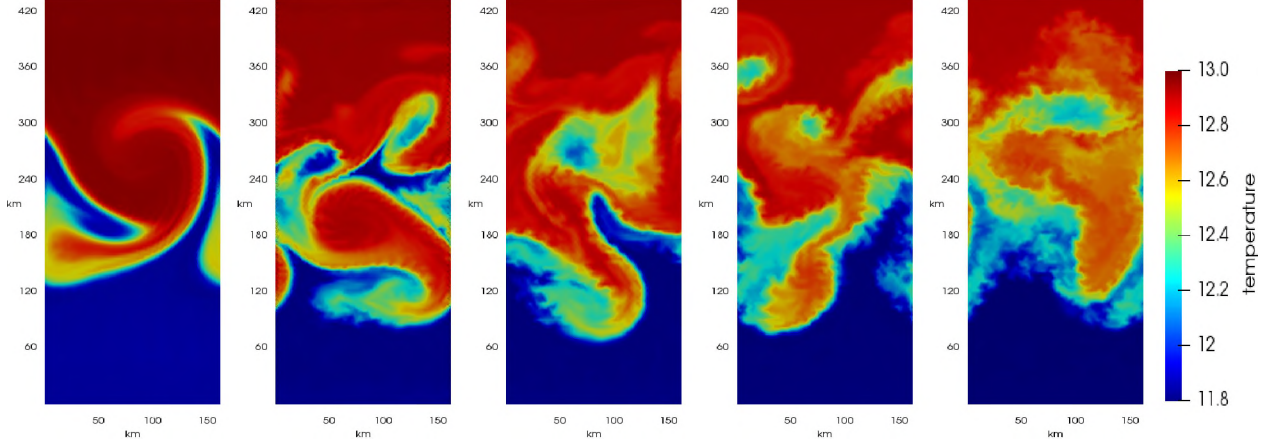
(a) Day 50 with different horizontal viscosities $\nu_h = 200, \; 20, \; 10, \; 5$, and 1.
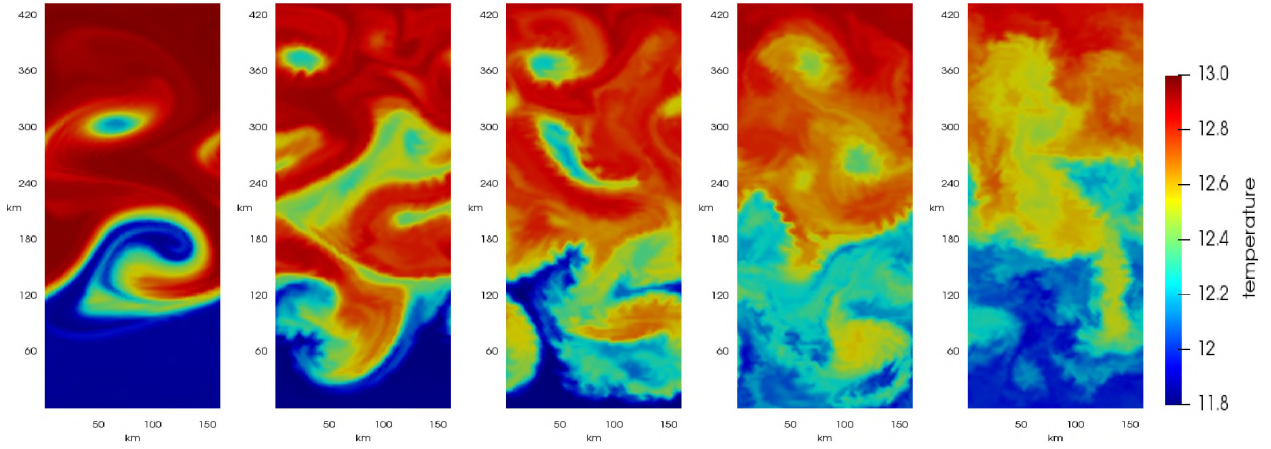
(b) Day 100 with different horizontal viscosities $\nu_h = 200, \; 20, \; 10, \; 5$, and 1.
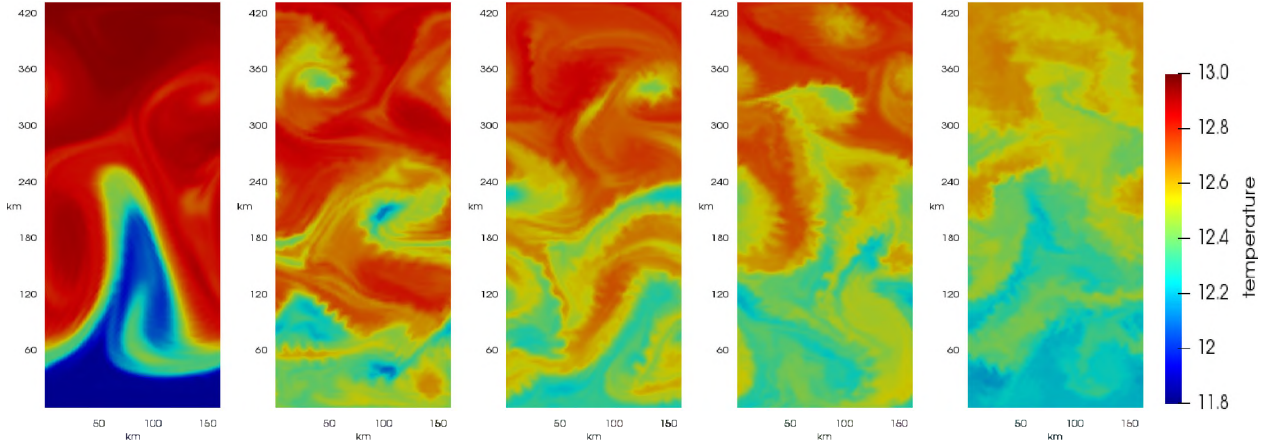
(c) Day 200 with different horizontal viscosities $\nu_h = 200, \; 20, \; 10, \; 5$, and 1.

Figure 10: Simulated surface temperatures by ETD-SE2 with $\Delta t = 25$ s for the baroclinic eddies test case under five different horizontal viscosities $\nu_h = 200, 20, 10, 5$, and 1 (from left to right). From top to bottom are the results on days 50, 100, and 200, respectively.

its orthogonalization requires several matrix-vector multiplications and evaluate the norm of the resulting vectors. Therefore, its efficiency decreases as the number of cores increases. SE2 only
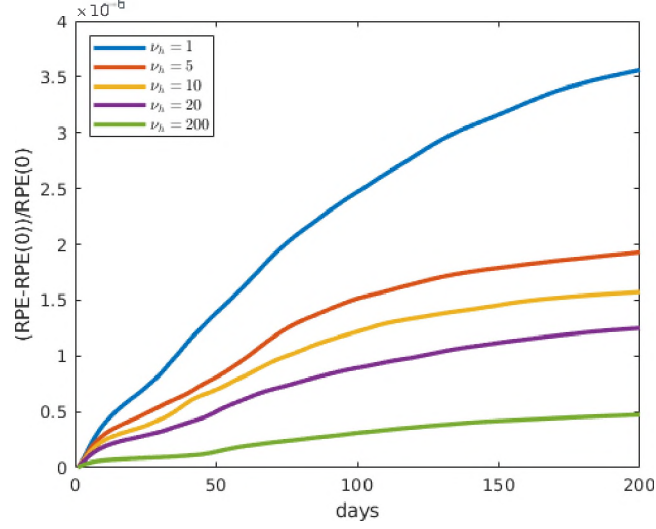
Figure 11: Evolutions of the normalized resting potential energy under different horizontal viscosities $\nu_h = 200, 20, 10, 5,$ and 1, respectively.

needs the inter-process communication to update the values on the halo cells at the end of each sub-time-stepping, which gives super-linear speedups again.
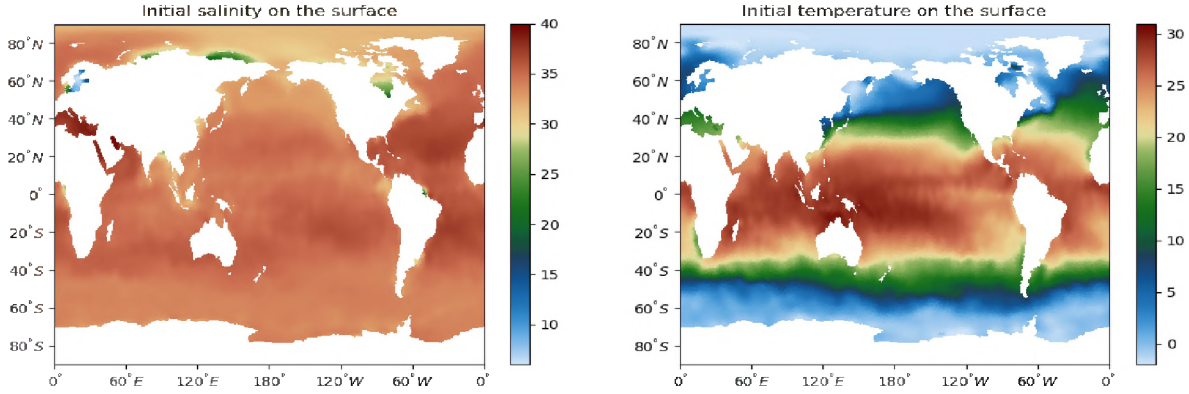

Figure 12: Initial values of the salinity and temperature for the global ocean test case.

Cores	ETD-SE2		SE2	
	Time (s)	Efficiency	Time (s)	Efficiency
16	44.44	–	225.61	–
32	27.72	80.2%	126.01	89.5%
64	16.68	66.6%	65.61	86.0%
128	11.33	49.0%	27.90	100.9%
256	6.85	40.6%	11.59	121.4%

Table 11: CPU times and parallel efficiency of ETD-SE2 and SE2 for the global ocean test case.
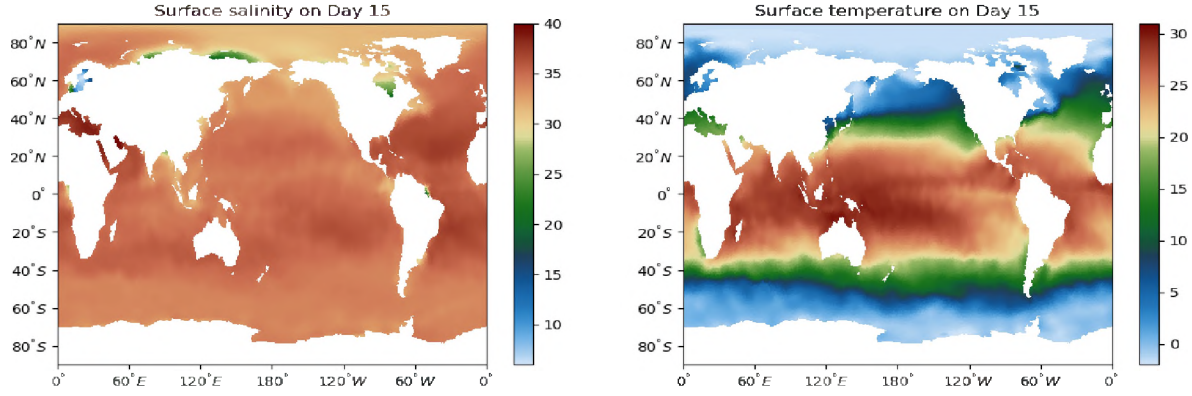
Figure 13: Simulated surface salinity and temperature on Day 15 by ETD-SE2 with $\Delta t = 60$ s for the global ocean test case.
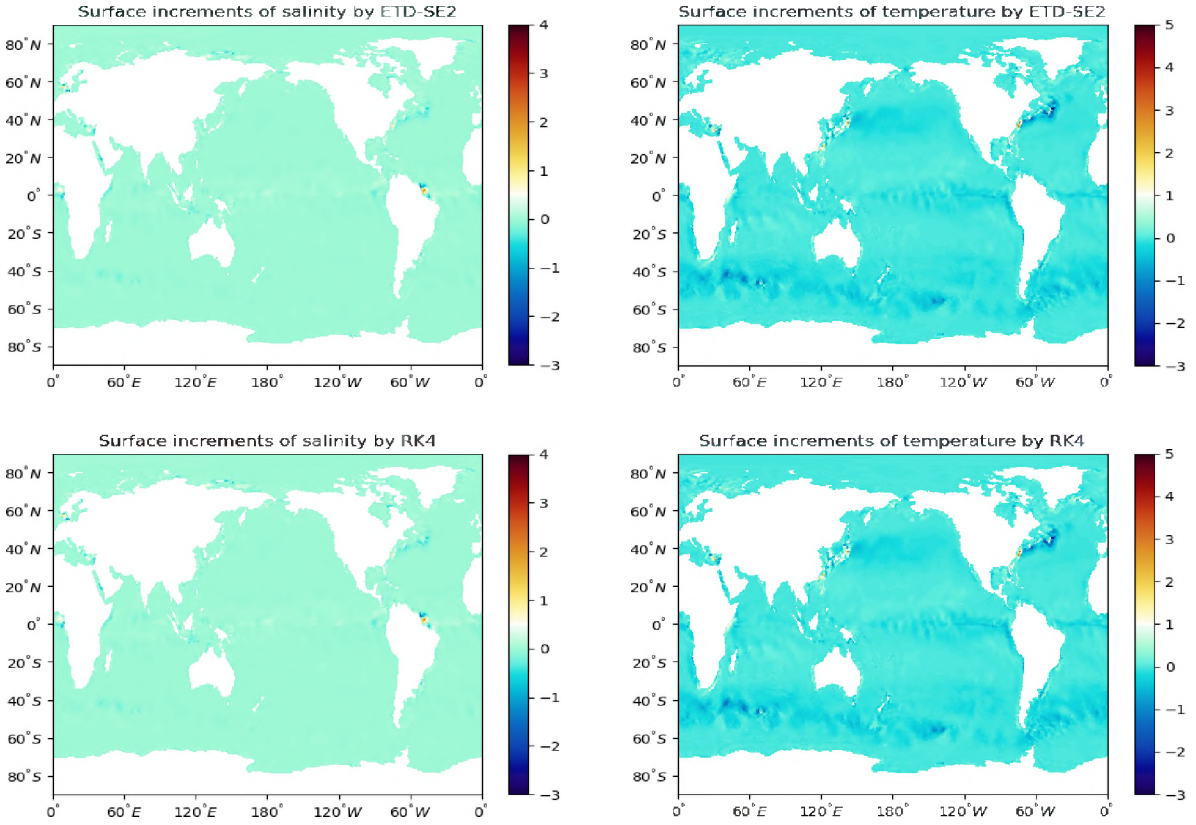

Figure 14: Simulated increments of the surface salinity and temperature on Day 15 by ETD-SE2 (top panel) and RK4 (bottom panel) with $\Delta t = 60$ s.

6. Conclusions

In this paper, we have fulfilled and test the parallel implementation of several exponential time differencing methods for simulating the ocean dynamics, governed by either the multi-layer SWEs or the multi-layer primitive equations. Since B-ETD2wave is developed on a reduced layered model, it costs less computations than exponential Rosenbrock-Euler and ETD2Wave in solving

multi-layer SWEs. We also designed a new ETD-SE2 method for solving the multilayer primitive equations. After splitting the oceanic motion into the baroclinic and barotropic components, we use the forward Euler scheme for the baroclinic mode and the exponential Rosenbrock-Euler scheme for the barotropic mode in the current MPAS-Ocean framework. Several standard numerical tests are performed and the comparison results demonstrate a great potential of applying the parallel ETD methods in simulating real-world geophysical flows.

Since the tracers' equations, like temperature and salinity, are the advection equations, MPAS-Ocean adopts the method proposed in [13], which is to utilize the MPDATA method and transport the tracers by the advective velocity at the intermediate time $t_{n+1/2}$. Smolarkiewicz and Margolin [35] designed the MPDATA method based on a general advection equation without considering the sub-sampling issue mentioned in [21]. As a future work, we will carefully design the new MPDATA method with the splitting transport velocities at different time levels to improve the temporal accuracy.

References

[1] R. Bleck and L. T. Smith. A wind-driven isopycnic coordinate model of the north and equatorial atlantic ocean: 1. model development and supporting experiments. *Journal of Geophysical Research: Oceans*, 95(C3):3273–3285, 1990.

[2] S. Calandrini, K. Pieper, and M. D. Gunzburger. Exponential time differencing for the tracer equations appearing in primitive equation ocean models. *Computer Methods in Applied Mechanics and Engineering*, 365:113002, 2020.

[3] J. Certaine. The solution of ordinary differential equations with large time constants. *Mathematical methods for digital computers*, 1:128–132, 1960.

[4] S. M. Cox and P. C. Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

[5] B. Cushman-Roisin and J.-M. Beckers. *Introduction to geophysical fluid dynamics: physical and numerical aspects*. Academic press, 2011.

[6] The Epetra Project Team. *The Epetra Project Website – https://trilinos.github.io/epetra.html*.

[7] E. Gallopoulous and Y. Saad. Efficient solution of parabolic equations by krylov approximation methods. *SIAM Journal on Scientific Computing*, 13:1236–1264, 1992.

[8] S. Gaudreault and J. A. Pudykiewicz. An efficient exponential time integration method for the numerical solution of the shallow water equations on the sphere. *Journal of Computational Physics*, 322:827–848, 2016.

[9] W. Gropp, W. D. Gropp, E. Lusk, A. Skjellum, and A. D. F. E. E. Lusk. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

[10] M. Günther and A. Sandu. Multirate generalized additive runge kutta methods. *Numerische Mathematik*, 133(3):497–524, 2016.

[11] R. L. Higdon. Implementation of a barotropic–baroclinic time splitting for isopycnic coordinate ocean modeling. *Journal of Computational Physics*, 148(2):579–604, 1999.

[12] R. L. Higdon. A two-level time-stepping method for layered ocean circulation models. *Journal of Computational Physics*, 177(1):59–94, 2002.

[13] R. L. Higdon. A two-level time-stepping method for layered ocean circulation models: further development and testing. *Journal of Computational Physics*, 206(2):463–504, 2005.

[14] R. L. Higdon and R. A. de Szoeke. Barotropic-baroclinic time splitting for ocean circulation modeling. *Journal of Computational Physics*, 135(1):30–53, 1997.

[15] N. J. Higham. The scaling and squaring method for the matrix exponential revisited. *SIAM Journal on Matrix Analysis and Applications*, 26(4):1179–1193, 2005.

[16] M. Hochbruck and C. Lubich. On krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 34(5):1911–1925, 1997.

[17] M. Hochbruck, C. Lubich, and H. Selhofer. Exponential integrators for large systems of differential equations. *SIAM Journal on Scientific Computing*, 19(5):1552–1574, 1998.

[18] M. Hochbruck, A. Ostermann, and J. Schweitzer. Exponential rosenbrock-type methods. *SIAM Journal on Numerical Analysis*, 47(1):786–803, 2009.

[19] M. Ilıcak, A. J. Adcroft, S. M. Griffies, and R. W. Hallberg. Spurious dianeutral mixing and the role of momentum closure. *Ocean Modelling*, 45:37–58, 2012.

[20] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[21] J. Marshall, K. Emanuel, and A. Adcroft. 12.950 atmospheric and oceanic modeling, Spring 2004. Massachusetts Institute of Technology: MIT OpenCourseWare, https://ocw.mit.edu.

[22] X. Meng, T.-T.-P. Hoang, Z. Wang, and L. Ju. Localized exponential time differencing method for shallow water equations: Algorithms and numerical study. *Communications in Computational Physics*, 29(1):80–110, 2020.

[23] C. Moler and C. Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM review*, 45(1):3–49, 2003.

[24] P. S. Peixoto. Accuracy analysis of mimetic finite volume operators on geodesic grids and a consistent alternative. *Journal of Computational Physics*, 310:127–160, 2016.

[25] M. R. Petersen, X. S. Asay-Davis, D. W. Jacobsen, M. E. Maltrud, T. D. Ringler, L. Van Roekel, C. Veneziani, and P. J. Wolfram Jr. Mpas-ocean model user's guide version 6.0. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2018.

[26] M. R. Petersen, D. W. Jacobsen, T. D. Ringler, M. W. Hecht, and M. E. Maltrud. Evaluation of the arbitrary lagrangian–eulerian vertical coordinate method in the mpas-ocean model. *Ocean Modelling*, 86:93–113, 2015.

[27] M. R. Petersen, T. D. Ringler, and D. W. Jacobsen. Mpas-ocean model user's guide. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2013.

[28] K. Pieper, K. C. Sockwell, and M. Gunzburger. Exponential time differencing for mimetic multilayer ocean models. *Journal of Computational Physics*, 398:108900, 2019.

[29] D. A. Pope. An exponential method of numerical integration of ordinary differential equations. *Communications of the ACM*, 6(8):491–493, 1963.

[30] T. Ringler, M. Petersen, R. L. Higdon, D. Jacobsen, P. W. Jones, and M. Maltrud. A multi-resolution approach to global ocean modeling. *Ocean Modelling*, 69:211–232, 2013.

[31] T. D. Ringler, J. Thuburn, J. B. Klemp, and W. C. Skamarock. A unified approach to energy conservation and potential vorticity dynamics for arbitrarily-structured c-grids. *Journal of Computational Physics*, 229(9):3065–3090, 2010.

[32] Y. Saad. Analysis of some krylov subspace approximations to the matrix exponential operator. *SIAM Journal on Numerical Analysis*, 29:209–228, 1992.

[33] A. Sandu and M. Günther. A generalized-structure approach to additive runge–kutta methods. *SIAM Journal on Numerical Analysis*, 53(1):17–42, 2015.

[34] R. B. Sidje. Expokit: A software package for computing matrix exponentials. *ACM Transactions on Mathematical Software (TOMS)*, 24(1):130–156, 1998.

[35] P. K. Smolarkiewicz and L. G. Margolin. Mpdata: A finite-difference solver for geophysical flows. *Journal of Computational Physics*, 140(2):459–480, 1998.

[36] M. Steele, R. Morley, and W. Ermold. Phc: A global ocean hydrography with a high-quality arctic ocean. *Journal of Climate*, 14(9):2079–2087, 2001.

[37] A. L. Stewart and P. J. Dellar. An energy and potential enstrophy conserving numerical scheme for the multi-layer shallow water equations with complete coriolis force. *Journal of Computational Physics*, 313:99–120, 2016.

[38] J. Thuburn and C. J. Cotter. A framework for mimetic discretization of the rotating shallow-water equations on arbitrary polygonal grids. *SIAM Journal on Scientific Computing*, 34(3):B203–B225, 2012.

[39] J. Thuburn, T. D. Ringler, W. C. Skamarock, and J. B. Klemp. Numerical representation of geostrophic modes on arbitrarily structured c-grids. *Journal of Computational Physics*, 228(22):8321–8335, 2009.

[40] M. Tokman. Efficient integration of large stiff systems of odes with exponential propagation iterative (epi) methods. *Journal of Computational Physics*, 213(2):748–776, 2006.

[41] The Trilinos Project Team. *The Trilinos Project Website – https://trilinos.github.io/*.