INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 19, Number 2-3, Pages 175–193

AN EXPONENTIAL TIME DIFFERENCING TIME-STEPPING SCHEME FOR THE TRACER EQUATIONS IN MPAS-OCEAN

SARA CALANDRINI, PHILIP W. JONES, AND MARK R. PETERSEN

Abstract. Exponential time differencing (ETD) methods, also known as exponential integrators, constitute a class of numerical methods for the time integration of stiff systems of differential equations. This manuscript investigates an ETD scheme for solving the tracer equations appearing in primitive equation ocean models, and shows the results obtained when such a scheme is applied within a full ocean circulation model. The main idea behind the scheme is the treatment of the vertical terms (transport and diffusion) with a matrix exponential, whereas the horizontal terms are dealt with in an explicit way. The performance of the ETD scheme is compared against that of other semi-implicit time-stepping schemes for realistic ocean configurations on quasi-uniform and variable resolution meshes.

Key words. Exponential time differencing, tracer equation, MPAS-Ocean.

1. Introduction

In ocean modeling, the tracer equation describes the transport of tracers, like temperature, salinity, radioactive material (tritium), chemical concentrations (chlorofluorocarbons) and biological organisms (phytoplankton). In a modern ocean circulation model (OCM), 2 to 50 tracers are tracked in a realistic simulation meaning that 2 to 50 tracers equations have to be solved, causing a significant computational load. Hence, efficiently solving multiple tracer equations is an important task in an ocean model.

Tracer evolution is described by an advection-diffusion equation of the form,

$$\partial_t T + \nabla_x \cdot (uT) + \mathcal{D}_x T + \partial_z(wT) - \partial_z(\kappa_z \partial_z T) = q(T), \tag{1}$$

where T is the tracer concentration, $\mathbf{u} = (u, w) \in R^3$ is the velocity of water, which is split into the horizontal velocity $u \in R^2$ and the vertical velocity w, $\mathcal{D}_x$ is the horizontal diffusion operator, κ_z is the vertical diffusion coefficient, and $q(T)$ represents interior sources or sinks. Models like POP [8], MITgcm [9] and MPAS-Ocean [2] use semi-implicit time-stepping schemes for advancing the tracer equations in time, since fully explicit schemes would be bound by severe time-step restrictions associated with fast mixing processes. Tracer vertical mixing usually occurs on fast time-scales and can be induced by density differences and/or by turbulent motions. These fast processes are often represented as a very strong diffusion coefficient κ_z in the vertical diffusion term $\partial_z(\kappa_z \partial_z T)$, resulting in very short time steps when using a fully explicit time integrator. For this reason, in POP, MITgcm and MPAS-Ocean, vertical diffusion can be treated implicitly with an implicit Euler algorithm. The remaining terms of the tracer equation (horizontal and vertical advection and horizontal diffusion) are treated explicitly in all the three models and the integration can proceed with longer time steps, typically at the horizontal advective CFL timescale.

However, the explicit treatment of vertical advection may not be optimal when many vertical layers are used to resolve mixed layer processes near the surface. For

Received by the editors November 2, 2021 and, in revised form, March 7, 2022.
2000 *Mathematics Subject Classification.* 35R35, 49J40, 60G40.

example, [2] employs 1-15km horizontal mesh spacing, but a much smaller 10 m mesh spacing in the vertical. This spacing, combined with higher surface velocities can lead to small CFL constraints and an implicit treatment of vertical advection can be beneficial. Implicit-explicit (IMEX) schemes [26, 27, 28] are indeed widely used in modern high-resolution numerical weather prediction and climate models. These schemes use an implicit component to deal with the terms in the model equations describing the fastest moving waves, allowing for longer time steps than fully explicit methods. An alternative to employing an implicit step for the fastest moving waves is the use of a matrix exponential. In [10], a time-stepping scheme that treats vertical diffusion and vertical advection in an exact way with a matrix exponential has been used to solve the tracer equation on idealized test cases. This scheme consists of an exponential time differencing (ETD) method where all the vertical terms are treated with a matrix exponential, whereas the horizontal terms are dealt with in an explicit way. ETD methods, also known as exponential integrators, have recently gained attention in the atmosphere and ocean modeling community due to their stability properties that allow time-steps considerably larger than those dictated by the CFL condition [11, 12, 13, 14, 15, 16, 17]. For a review of exponential integrators we refer to [6].

In this work, we consider the ETD time-stepping scheme presented in [10], called hereafter ETD-CPG, and study its performance in MPAS-Ocean on realistic ocean test cases. The operator splitting used by ETD-CPG (vertical terms treated with a matrix exponential vs horizontal terms treated explicitly) has two main advantages. First, higher accuracy is expected compared to other semi-implicit methods for the tracer equation because of an exact treatment of the fast vertical terms. Second, a focus on the vertical terms simplifies the implementation and reduces the cost in a parallel computing context. Ocean models decompose the domain only in the horizontal so computation of vertical terms can take place solely within a node and no additional communication is needed for a parallel implementation. The addition of accelerators like Graphic Processing Units (GPUs) within a node can be used to further optimize and reduce the cost of the implementation. Exploiting this local implementation, we use an approach based on scaling and squaring relations [18, 19] for the computation of the matrix functions. This approach, already presented in [10], is based on polynomials of moderate degree and results in a consistent and stable approximation of the φ-functions described below.

ETD-CPG has already been tested on 2D idealized test cases for the tracer equation, where it was shown that larger time-steps could be taken than other semi-implicit time-stepping schemes. Since we are going to test ETD-CPG on realistic ocean test cases, the time step used will be dictated by the dynamics, i.e. the change in time of the velocity and thickness of the vertical layers. In the tests performed in this work, we use both quasi-uniform and variable-resolution meshes on a sphere and the initial conditions are interpolated from the data gathered by the Polar Science Center Hydrographic Climatology [23].

The paper is organized as follows. Section 2 describes the vertical and horizontal discretization of the tracer equation in MPAS-Ocean. Section 3 describes the ETD-CPG method, focusing on the structure of the linear operator and the computation of the matrix functions. In section 4 numerical results are presented showing the convergence of the method and its performance on two test cases with realistic ocean configurations. Finally, in section 5 we draw our conclusions.

2. Tracer Equation

In MPAS-Ocean, the governing equations are the primitive equations, which correspond to the incompressible Boussinesq equations in hydrostatic balance. Following this hydrostatic assumption, the tracer equation in continuous form can be re-written as

$$\frac{\partial(\widetilde{\rho} T)}{\partial t} + \nabla \cdot (\widetilde{\rho} T \mathbf{u}) + \frac{\partial(\widetilde{\rho} T w)}{\partial z} = D_h^T + D_\nu^T \tag{2}$$

where T is the tracer, $\mathbf{u}$ is the horizontal velocity, w is the vertical velocity, $\widetilde{\rho}$ is the pseudo-density, and z represents the vertical coordinate which is defined positive upward. The horizontal and vertical diffusion terms, D_h^T and D_ν^T, are defined as

$$D_h^T = \nabla \cdot (\widetilde{\rho} \kappa_h \nabla T), \qquad D_\nu^T = \widetilde{\rho} \frac{\partial}{\partial z}\left(\kappa_\nu \frac{\partial T}{\partial z}\right) \tag{3}$$

where κ_h and κ_ν are the horizontal and vertical diffusion, respectively. Without loss of generality, we assume that no forcing term is present.

2.1. Vertical discretization. For the vertical discretization of the tracer equation, let us define k as the index of the vertical layers. This index increases downward, so $k = 1$ is the top layer and $k = N$ is the bottom one. The z coordinate is positive upward and $z = 0$ is the mean elevation of the free surface. The tracer equation with vertical discretization is written as

$$\frac{\partial(h_k T_k)}{\partial t} + \nabla \cdot (h_k T_k \mathbf{u}_k) + \overline{T_k} w_k + \overline{T_{k+1}} w_{k+1} = [D_h^T]_k + [D_\nu^T]_k \tag{4}$$

$$[D_h^T]_k = \nabla \cdot (h_k \kappa_h \nabla T_k), \qquad [D_\nu^T]_k = h_k \delta z^m(\kappa_\nu \delta z^t(T_k)) \tag{5}$$

where w_k indicates the transport of fluid from layer k to $k-1$, and the pseudo-density $\widetilde{\rho}$ has been replaced by the layer-thickness h. The vertical operators $\overline{(\cdot)}$, $\delta z^m(\cdot)$ and $\delta z^t(\cdot)$, on a generic variable ψ_k, are defined as

$$\overline{\psi_k} = \frac{\psi_{k-1} + \psi_k}{2} \tag{6}$$

$$\delta z^m(\psi_k) = \frac{\psi_{k-1} - \psi_k}{\overline{h_k}} \tag{7}$$

$$\delta z^t(\psi_k) = \frac{\psi_k - \psi_{k+1}}{h_k} \tag{8}$$

where the superscripts m and t denote the location as the middle or top of cell k in the vertical.

In MPAS-Ocean, the vertical mesh uses an Arbitrary Lagrangian-Eulerian (ALE) algorithm, in which the vertical coordinate is Lagrangian, but can be remapped to an arbitrary target coordinate. This approach allows a great deal of freedom to choose among vertical grid types or target grid locations. In the Lagrangian evolution, the coordinates move with the flow according to the mass conservation equation, or thickness equation. This equation, discretized in the vertical, has the form

$$\frac{\partial h_k}{\partial t} + \nabla \cdot (h_k \mathbf{u}_k) + w_k - w_{k+1} = 0. \tag{9}$$

For Lagrangian or isopycnal coordinates, the mesh interfaces move with the flow so there is no flow across the interface and w_k is simply set to zero for every k. If Eulerian coordinates (e.g. depth or z-level) or other coordinate locations are desired, the coordinate and thickness is determined by the desired locations and Equation 9 is used to compute an effective velocity w_k across the interface. Using a first-order finite difference approximation for the time derivative of h at level k, we get

$$w_k = w_{k-1} - \nabla \cdot (h_k \mathbf{u}_k) - \frac{h_k^{ALE} - h_k}{\Delta t} \tag{10}$$

where the quantity h_k^{ALE} represents the desired thickness for the new time. The way h_k^{ALE} is computed determines the type of coordinates chosen. For z-level, h_k^{ALE} is computed so that all layers have a fixed thickness except for the top layer, thus

$$h_1^{ALE} = h_1^{rest} + \zeta \tag{11}$$

$$h_k^{ALE} = h_k^{rest} \text{ , for } k > 1 \tag{12}$$

where h_k^{rest} is the resting thickness, i.e. the layer thickness when the ocean is at rest, and ζ is the sea surface height. In z-star coordinates, h_k^{ALE} is computed so that sea surface height perturbations are distributed throughout the column of fluid, thus

$$h_k^{ALE} = h_k^{rest} + \zeta \frac{h_k^{rest}}{\sum_{k'} h_{k'}^{rest}} \text{ , for all } k \text{ .} \tag{13}$$

The simulations presented in Section 4 use z-level and z-star vertical coordinates. The computation of h_k^{ALE} for other coordinate systems is described in [1].

2.2. Horizontal discretization. The horizontal discretization in MPAS-Ocean consists of a C-grid, finite-difference/volume method called TRiSK, introduced for the first time in [3] and later applied to the nonlinear shallow-water equations in [4]. TRiSK is applicable to a broad class of meshes, and in MPAS-Ocean it is mostly applied to spherical centroidal Voronoi tessellations (SCVTs) [5]. To create a Voronoi tesselation on the surface of the sphere, S, we start by picking $\{x_i\}_{i=1}^n$ distinct grid points and then assign every point on S to whichever x_i it is closest to. This results in a set of Voronoi regions, $\{V_i\}_{i=1}^n$, that can be expressed as

$$V_i = \{y \in S \text{ s.t. } ||x_i - y|| < ||x_j - y|| \text{ for } j = 1, \ldots, n \text{ and } j \neq i\} \tag{14}$$

so each V_i is uniquely associated with a single grid point x_i. In a SCVT, these points $\{x_i\}_{i=1}^n$ are also approximations of the centroids (mass centers) of the Voronoi cells. A Delaunay triangulation is the dual-mesh of a Voronoi tessellation; specifying either uniquely determines the other. The centers $\{x_i\}_{i=1}^n$ of a Voronoi tessellation coincide with the vertices $\{x_v\}_{v=1}^m$ of the dual mesh triangles. In a C-grid staggering on an SCVT mesh, height, tracers, pressure and kinetic energy are defined at centers x_i of the Voronoi cells, the normal component of the velocity is located at points x_e on the cell edges, and vorticity (curl of velocity) is defined at cell vertices x_v. As shown in Fig. 1, x_e represents the intersection point between the Voronoi cell edge and the dual triangle edge, and this intersection point corresponds to the midpoint of the triangle edge. Fig. 1 also displays the the lengths d_e and l_e, where d_e measures the distance between the Voronoi cells sharing edge e, and l_e measures the length of the Voronoi edge e. In the following, the subscripts i and e indicate the discretized variables through cell centers and edges, respectively. Since we are focusing on the discretization of the tracer equation only, we will not work with variables and operators defined at cell vertices.

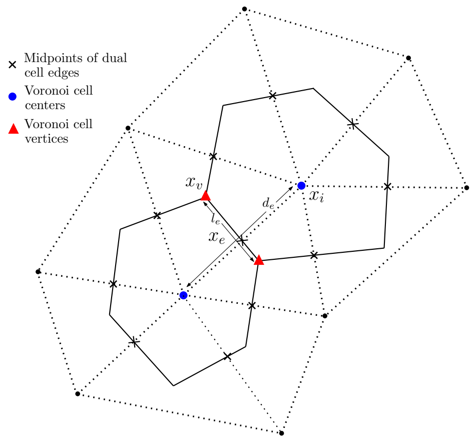

FIGURE 1. Variable collocation points for the TRiSK scheme on a SCVT.

The tracer equation with horizontal discretization can be written as

(15)
$$\frac{\partial(h_{k,i}T_{k,i})}{\partial t} + [\nabla\cdot(\widehat{(h_{k,:})}_e\widehat{(T_{k,:})}_e\mathbf{u}_{k,:})]_i + \overline{T_{k,i}}w_{k,i} + \overline{T_{k+1,i}}w_{k+1,i} = [D_h^T]_{k,i} + [D_\nu^T]_{k,i}$$

(16)
$$[D_h^T]_{k,i} = [\nabla\cdot(\widehat{(h_{k,:})}_e\,\kappa_h[\nabla T_{k,:}]_e)]_i, \qquad [D_\nu^T]_{k,i} = h_{k,i}\delta z^m(\kappa_\nu\delta z^t(T_{k,i}))$$

where each variable now has two sub-scripted indices, the first indicating the vertical layer, and the second indicating its position on the horizontal grid, namely either i or e. Colons in subscripts may be places as second index to indicate that multiple edges or cell centers are used in computing the horizontal operator. We would like to point out that the vertical transport through the sea surface and at the bottom surface is zero, i.e. $w_{1,i} = 0$ and $w_{N+1,i} = 0$. Moreover, we consider $u_{k,e} = 0$ on all boundary edges. For a generic vector field $\mathbf{Y}_k$ and variable ψ_k , the discrete horizontal operators $[\nabla\cdot\mathbf{Y}_{k,:}]_i$, $[\nabla\psi_{k,:}]_e$ and $\widehat{(\psi_{k,;})}_e$ are defined as

$$[\nabla\cdot Y_{k,:}]_i = \frac{1}{A_i}\sum_{e\in EC(i)} n_{e,i}Y_{k,e}l_e \tag{17}$$

$$[\nabla\psi_{k,:}]_e = \frac{1}{d_e}\sum_{i\in CE(e)} -n_{e,i}\psi_{k,i} \tag{18}$$

$$\widehat{(\psi_{k,;})}_e = \sum_{i\in CE(e)}\frac{\psi_{k,i}}{2} \tag{19}$$

where A_i indicates the Voronoi cell area, d_e is the distance between cell centers, l_e is edge length and $n_{e,i}$ represents the sign of the vector at edge e with respect to cell i. The sets $EC(i)$ and $CE(e)$ are defined in Table 1. For more details about the horizontal discretization of the full primitive equations refer to [2].

3. Exponential Time Differencing

Exponential integrators or exponential time differencing (ETD) schemes are a special class of time integration methods based on a splitting of the right-hand-side into a linear part and a remainder. After the derivation of an exponential integrator, we focus on the choice of the linear operator for solving the tracer

TABLE 1. Definition of element groups used to build the discrete tracer equation.

Syntax	Output
$e \in EC(i)$	Set of edges that define the boundary of the Voronoi cell i
$i \in CE(e)$	Two Voronoi cells that share edge e

equations. We prefer an approximation to the Jacobian of the right-hand-side over the full Jacobian, due to the physics of the problem and to favorable properties concerning the implementation and structure of the linear operator. We consider the ETD-CPG method presented in [10] where the full Jacobian is split into a vertical and a horizontal part. The vertical part is treated exponentially since the vertical terms (vertical diffusion and advection) are those associated to the fast time-scales in the ocean. Many ocean models, like POP, MITgcm and the current version of MPAS-Ocean, treat implicitly only the tracer vertical diffusion.

3.1. Derivation of an exponential integrator. Let

$$\partial_t T = F(T) \tag{20}$$

be a system of partial differential equations (PDEs), where $T = T(t)$ denotes the vector of the solution variables for $t \in [t_n, t_{n+1}]$, and $F(T)$ is the right-hand-side. The interval $[t_n, t_{n+1}]$ refers to one time step. Let us split the right-hand-side into a linear part and a remainder as

$$F(T) = A_n T + R(T), \tag{21}$$

where A_n represents a linear operator, and $R(T) := F(T) - A_n T$ denotes the remainder, which in general is nonlinear. Applying the variation of constants formula, the solution of (20) at time $t_{n+1} = t_n + \Delta t$ is obtained as

$$T_{n+1} = \exp(\Delta t A_n) T_n + \int_0^{\Delta t} \exp((\Delta t - \tau) A_n) R(T(t_n + \tau)) d\tau \ . \tag{22}$$

To build a concrete exponential integrator, let us consider an approximation of $R(T(t_n + \tau))$, that is its Taylor expansion truncated at s, with $s \in \mathbb{N}$. By substituting

$$R(T(t_n + \tau)) \approx \sum_{k=1}^{s} \frac{\tau^{k-1}}{(k-1)!} \left. \frac{\mathrm{d}^{k-1} R(T(t_n + \tau))}{\mathrm{d}\tau^{k-1}} \right|_{\tau=0}$$

in (22), the solution T_{n+1} can be approximated by

$$\begin{aligned} T_{n+1} \approx & \exp(\Delta t A_n) T_n \\ & + \sum_{k=1}^{s} \frac{1}{(k-1)!} \Big[\int_0^{\Delta t} \exp((\Delta t - \tau) A_n) \tau^{k-1} d\tau \Big] \left. \frac{\mathrm{d}^{k-1} R(T(t_n + \tau))}{\mathrm{d}\tau^{k-1}} \right|_{\tau=0} . \end{aligned}$$

Now, let us define

$$\varphi_k(\Delta t A_n) := \frac{1}{\Delta t^k (k-1)!} \int_0^{\Delta t} \exp((\Delta t - \tau) A_n) \tau^{k-1} d\tau, \quad k = 1, 2, \ldots, s \tag{23}$$

which are know as φ-functions. Using (23), the solution T_{n+1} can be rewritten as

$$T_{n+1} \approx \exp(\Delta t A_n) T_n + \sum_{k=1}^{s} \Delta t^k \varphi_k(\Delta t A_n) \left. \frac{\mathrm{d}^{k-1} R(T(t_n + \tau))}{\mathrm{d}\tau^{k-1}} \right|_{\tau=0} , \tag{24}$$

which is an exponential integrator with s stages.

Fundamental for any exponential integrator is the computation of the φ-functions $\varphi_k(\Delta t A_n)$. With the change of variable $\Delta t - \tau = (1-\sigma)\Delta t$, definition (23) can be rewritten as

$$\varphi_k(\Delta t A_n) = \frac{1}{(k-1)!}\int_0^1 \exp((1-\sigma)\Delta t A_n)\sigma^{k-1} d\sigma, \quad k = 1, 2, \dots, s \ , \tag{25}$$

therefore

$$\begin{aligned}\varphi_1(\Delta t A_n) &= \int_0^1 \exp((1-\sigma)\Delta t A_n) d\sigma \\ &= (\Delta t A)^{-1}(\exp(\Delta t A_n) - I) \ ,\end{aligned}$$

$$\begin{aligned}\varphi_2(\Delta t A_n) &= \int_0^1 \exp((1-\sigma)\Delta t A_n)\sigma \, d\sigma \\ &= (\Delta t A_n)^{-2}(\exp(\Delta t A_n) - \Delta t A_n - I),\end{aligned}$$

and so on for $k > 2$. In the above expressions, I indicates the identity operator. For $k = 0$, we have that $\varphi_0(\Delta t A_n) = \exp(\Delta t A_n)$. More details about the numerical computation of these matrix functions will be given in section 3.3.

The parameter s in (24) indicates the number of stages of the ETD method. By taking $s = 1$, the exponential Euler method is obtained as

$$T_{n+1} \approx \exp(\Delta t A_n) T_n + \Delta t \varphi_1(\Delta t A_n) R(T_n) \tag{26}$$

$$= T_n + \Delta t \varphi_1(\Delta t A_n) F(T_n) \ . \tag{27}$$

This method is in general a first-order accurate method, but if A_n is the Jacobian matrix of the system evaluated at t_n, then the method becomes second-order accurate [6, 20]. For the solution of the tracer equation, we are going to consider a second-order two-stage ($s = 2$) method that requires the computation of only one φ-function. This scheme can be written as

$$T_n^{(1^{st}\ stage)} = T_n + \Delta t \varphi_1(\Delta t A_n) F(T_n) \ , \tag{28}$$

$$T_{n+1} = T_n^{(1^{st}\ stage)} + \frac{1}{2}\Delta t \varphi_1(\Delta t A_n)(R(T_n^{(1^{st}\ stage)}) - R(T_n)) \ , \tag{29}$$

where the first derivative of $R(T_n + \tau)$ is approximated with a first-order finite-difference approximation

$$\left.\frac{\mathrm{d}R(T(t_n+\tau))}{\mathrm{d}\tau}\right|_{\tau=0} \approx \frac{R(T_n^{(1^{st}\ stage)}) - R(T_n)}{\Delta t} \ .$$

The scheme (28)-(29) fulfills the nonstiff order conditions described in [7] up to order two, and the stiff order conditions up to order one. This method was used in [21] for studying steady and unsteady inviscid flows, using the full Jacobian of the system as linear operator. In [10] it was tested on idealized test cases for solving the tracer equation. The choice of the operator A_n in the context of the tracer equation is described in the section below.

3.2. Exponential integrator for the tracer equation. Consider the discretized tracer equation (15), and let us rewrite it as

$$\partial_t T = F(T) = J_n T, \tag{30}$$

where $F(T)$ is the right-hand-side and $T = T(t)$ denotes the vector of tracer values for $t \in [t_n, t_{n+1}]$. Since we are assuming a zero forcing term, the tracer equation is linear in T, therefore the right-hand-side can be written as the Jacobian of the system evaluated at t_n times T.

Let us split the Jacobian J_n into a vertical and a horizontal part, i.e.

$$J_n = J_n^z + J_n^x \ , \tag{31}$$

where J_n^z contains the derivatives of the vertical terms only, and J_n^x contains the derivatives of the horizontal terms only. Thus,

$$\partial_t T = F(T) = J_n^z T + J_n^x T \ . \tag{32}$$

To apply an exponential integrator to solve (32), we have to identify a linear operator A_n and a remainder $R(T)$. Since we wish to treat fast vertical processes with the ETD approach, the term J_n^z is interpreted as the linear operator A_n, while $J_n^x T$ is the remainder $R(T)$. Thus, we adopt an exponential time differencing solver where the vertical terms are treated with a matrix exponential, whereas the horizontal are dealt with in an explicit way. Applying the splitting to the scheme (28)-(29) we have the ETD-CPG method

$$T_n^{1st\, stage} = T_n + \Delta t \varphi_1(\Delta t J_n^z) F(T_n) \ , \tag{33}$$

$$T_{n+1} = T_n^{1st\, stage} + \frac{1}{2}\Delta t \varphi_1(\Delta t J_n^z)(R_n^{1st\, stage} - R_n) \ , \tag{34}$$

which is a second-order, two-stage ETD method. The remainders are defined as

$$R_n = F(T_n) - J_n^z T_n = J_n^x T_n, \tag{35}$$

$$R_n^{1st\, stage} = F(T_n^{1st\, stage}) - J_n^z T_n^{1st\, stage} = J_n^x T_n^{1st\, stage}, \tag{36}$$

and they both take into account only the contributions from the horizontal terms. From an implementation point of view, the matrix J_n^x is never built since R_n is obtained for free from the construction of the right-hand side $F(T_n)$ needed at the first stage, and for $R_n^{1st\, stage}$ we only evaluate the horizontal terms at $T_n^{1st\, stage}$.

The linear operator J_n^z contains the derivatives of the vertical terms only, and its entries can be ordered so that J_n^z has a block diagonal structure. In the case of a flat-bottom, this matrix has size $N_z N_x$, where N_z is the total number of vertical layers, and N_x is the total number of Voronoi cells in the horizontal discretization. Since, for every layer, there is no interaction between the derivatives of the vertical terms associated with two different Voronoi cells, the derivatives associated with the same horizontal element form a submatrix of dimension $N_z \times N_z$. Therefore, J_n^z can be written has

$$J_n^z = \begin{bmatrix} J_n^{z,1} & 0 & 0 & \dots & 0 \\ 0 & J_n^{z,2} & 0 & \dots & 0 \\ 0 & 0 & J_n^{z,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & J_n^{z,N_x} \end{bmatrix} \tag{37}$$

where each block $J_n^{z,i}$ represents the derivatives of the vertical terms associated with a single Voronoi cell i in the horizontal discretization. In general, the dimension of one diagonal block depends on the number of vertical layers associated with that specific Voronoi cell. If all Voronoi cells have the same number of vertical layers, as in the case of a flat-bottom, all the blocks $J_n^{z,i}$ would have the same dimension. For complex coastlines and bathymetry, these matrices would have different sizes, depending on the number of vertical layers associated with each Voronoi cell.

The same block diagonal structure as J_n^z is inherited by $\varphi_1(\Delta t J_n^z)$ as shown by Proposition 3 in [10]. Thus,

$$\varphi_1(\Delta t J_n^z) = \begin{bmatrix} \varphi_1(\Delta t J_n^{z,1}) & 0 & 0 & \dots & 0 \\ 0 & \varphi_1(\Delta t J_n^{z,2}) & 0 & \dots & 0 \\ 0 & 0 & \varphi_1(\Delta t J_n^{z,3}) & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & \varphi_1(\Delta t J_n^{z,N_x}) \end{bmatrix} \tag{38}$$

meaning that, for every Voronoi cell i, a matrix $\varphi_1(\Delta t J_n^{z,i})$ is associated with it. From an implementation point of view, all these matrices are independent from one another, since the $J_n^{z,i}$ matrices are independent, so no inter-processor communications are needed for the computation of either the $J_n^{z,i}$ and $\varphi_1(\Delta t J_n^{z,i})$ matrices in a parallel setting.

3.3. Computation of the φ-functions. For the construction of the $\varphi_1(\Delta t J_n^{z,i})$ matrices needed in the ETD-CPG scheme, we use a method based on scaling and squaring relations, as in [10, 13]. For a generic a matrix A, the scaling and squaring method for the computation of $\exp(A)$ ([25]) exploits the fact that $\exp(A)$ can be well approximated by a rational function for small $||A||$, where $||\cdot||$ indicates any subordinate matrix norm. To guarantee that the starting matrix has a small norm, A is first scaled by 2^M, for $M \in \mathbb{N}$, so that $||A/2^M|| < 1$, and then the relation $\exp(A) = \exp(A/2^M)^{2^M}$ is used to obtain an approximation of $\exp(A)$. In [10], we extend the idea in [25] to construct a method based on scaling and squaring relations (as eq. (43) below) for the computation of the φ_k-functions. In the following, we first derive this method for a generic φ_k-function, and then consider the specific case of the φ_1-function.

Following definition (25) for a given a matrix A, $\varphi_k(2A)$ is given by

$$\varphi_k(2A) = \frac{1}{2^k(k-1)!}\int_0^2 \exp((2-\sigma)A)\sigma^{k-1}d\sigma. \tag{39}$$

Splitting the integral into the intervals (0, 1) and (1, 2) we get

$$2^k\varphi_k(2A) = \int_0^1 \exp((2-\sigma)A)\frac{\sigma^{k-1}}{(k-1)!}d\sigma + \int_1^2 \exp((2-\sigma)A)\frac{\sigma^{k-1}}{(k-1)!}d\sigma \tag{40}$$

$$= \exp(A)\int_0^1 \exp((1-\sigma)A)\frac{\sigma^{k-1}}{(k-1)!}d\sigma + \int_0^1 \exp((1-\sigma)A)\frac{(1+\sigma)^{k-1}}{(k-1)!}d\sigma \tag{41}$$

where the second integral was shifted to (0, 1). Now, using the binomial identity

$$\frac{(1+\sigma)^{k-1}}{(k-1)!} = \sum_{j=0}^{k-1}\frac{\sigma^{k-j-1}}{(k-j-1)!j!} \tag{42}$$

in the second term and the definition of φ_k and φ_{k-j} we get the recursive formula

$$2^k\varphi_k(2A) = \exp(A)\varphi_k(A) + \sum_{j=0}^{k-1}\frac{\varphi_{k-j}(A)}{j!} \tag{43}$$

where the matrix function of $2A$ is expressed as a product of two matrix functions of A plus some correction terms. The scaling and squaring method for the computation of $\varphi_k(A)$ is based on eq. (43) applied M times to a scaled matrix $A/2^M$, where $M \in \mathbb{N}$. The starting step is a Taylor expansion up to order r of $\exp(A/2^M)$

used to approximate $\varphi_k(A/2^M)$ and $\varphi_{k-j}(A/2^M)$ using their definition (25). The algorithm can be summarized as follows:

```
Algorithm 1: Computation of $\varphi_k(A)$
  Step 1. Build a polynomial approximation of $\exp(A/2^M)$, $\varphi_k(A/2^M)$ and
              $\varphi_{k-j}(A/2^M)$, for $0 \le j \le k-1$.
  Step 2. Apply the recursive formula (43) using the polynomial approximations.
  Step 3. Compute approximations for $\exp(A/2^{M-s})$ and $\varphi_{k-j}(A/2^{M-s})$, for
          $1 \le s \le M-1$.
  Step 4. Repeat Step 2 and Step 3 $M$ times.
```

For the specific case of $\varphi_1(A)$, the recursive formula looks like

$$\varphi_1(2A) = \frac{1}{2}(\exp(A) + I)\varphi_1(A) \tag{44}$$

where I is the identity matrix. The algorithm for the construction of $\varphi_1(A)$ based on the above formula can be summarized as follows:

```
Algorithm 2: Computation of $\varphi_1(A)$
  Step 1. Define $p_0^0(A/2^M)$ and $p_1^0(A/2^M)$ to be polynomial approximations of
          $\exp(A/2^M)$ and $\varphi_1(A/2^M)$ as
              $p_0^0(A/2^M) = T_r(A/2^M) + (A/2^M)^{r+1}q(A/2^M)$
              $p_1^0(A/2^M) = \frac{A^{-1}}{2^M}(p_0^0(A/2^M) - I)$
          where $T_r(z) = 1 + z + \cdots + z^r/r!$ is the Taylor approximation of $\exp(z)$ up
          to order $r$, and $q(z)$ is a remainder.
  Step 2. For $0 \le s \le M-1$, given $p_0^s(A/2^{M-s})$ and $p_1^s(A/2^{M-s})$ compute
          $p_1^{s+1}(A/2^{M-(s+1)})$ using formula (44) as
              $p_1^{s+1}(A/2^{M-(s+1)}) = \frac{1}{2}\Big(p_0^s(A/2^{M-s}) + I\Big)p_1^s(A/2^{M-s})$
  Step 3. Given $p_0^s(A/2^{M-s})$, compute $p_0^{s+1}(A/2^{M-(s+1)})$ as
              $p_0^{s+1}(A/2^{M-(s+1)}) = p_0^s(A/2^{M-s})p_0^s(A/2^{M-s})$
  Step 4. Repeat Step 2 and Step 3 until $s = M-1$.
```

Finally, the polynomial $p_1^M(A)$ will be the approximation of $\varphi_1(A)$. Since it is not our intention to provide a full account of this method based on scaling and squaring relations, refer to [10] for more details, including an error estimate for the approximation $\varphi_1(A) \approx p_1^M(A)$.

In terms of cost for the approximation $p_1^M(\Delta t J_n^{z,i}))$ needed in ETD-CPG, it is important to notice that the matrices $J_n^{z,i}$ are banded, in particular they are tridiagonal for second-order discretization schemes like the one presented in section 2.1. Working with banded matrices greatly reduces the cost of the scaling and squaring algorithm, compared to working with dense matrices. For two banded matrices with bandwidth b, their product will be banded with bandwidth $2b$, and its cost will be less than $2(1 + 2b)N_z^2$, as explained in [22]. In the context of the scaling and squaring method, M matrix-matrix products of banded matrices are performed, therefore the cost of the approximation of one $\varphi_1(J_n^{z,i})$ is $2(1+2^M b)N_z^2$. Looking at the ETD-CPG scheme (33)-(34), the computation of the N_x φ_1 matrices has to be performed only once for a given time-step, since the same matrices are needed for the first and second stage. Moreover, assuming a zero forcing term, all tracers equations share the same linear operator, i.e. J_n^z is the same for all of them. Therefore, the matrices $J_n^{z,i}$ and $\varphi_1(\Delta t J_n^{z,i})$ are the same for all tracers.

4. Numerical Results

In this section, numerical tests are presented to investigate the performance of the ETD-CPG solver (33)-(34) in MPAS-Ocean. The ETD time-stepping scheme is compared against a semi-implicit scheme named hereafter RK4+IE, which consists of a convergent operator splitting scheme that employs implicit Euler for the treatment of the vertical diffusion term and RK4 for the rest of the terms, i.e.

$$\partial_t T = F_{RK4}(t,T) + F_{IE}(t,T)$$

where

$$\begin{aligned} F_{RK4}(t,T) &= -\nabla_x \cdot (uT) - \mathcal{D}_x T - \partial_z(wT) \\ F_{IE}(t,T) &= \partial_z(\kappa_z \partial_z T) \end{aligned}$$

considering the tracer equation in continuous form.

The tests consists of realistic ocean configurations on quasi-uniform and variable resolution meshes. Smooth initial and boundary conditions are provided, where the initial conditions are interpolated from data gathered by the Polar Science Center Hydrographic Climatology [23]. All the time-steps used in the simulations are dictated by the dynamics, i.e. the change in time of the velocity and thickness of the water. For the solution of the momentum and thickness equation, the split-explicit time-stepping scheme available in MPAS-Ocean has been used. This scheme adopts a baroclinic/barotropic splitting where the barotropic velocity and total ocean depth are explicitly subcycled within each large time-step taken for the three-dimensional baroclinic velocity. For more details about this split-explicit method refer to [2].

We start by testing the ETD-CPG method in a 2D idealized case to show that the correct order of convergence in time is achieved.

4.1. Convergence. To show the correct implementation of the method in MPAS-Ocean, we test ETD-CPG on a 2D steady state test case, so the initial condition is the analytical solution which should remain constant over all time integration. We consider a planar mesh of dimensions 500m × 500m discretized with 100 Voronoi cells with $d_e = 5$m. The number of vertical layers is 50. We assume u, w and h constant in time. The velocity field is a circular, divergence-free field, which is tangential to the boundaries. It is defined as

$$(u,w) = (\psi_1(x)\psi_2{\prime}(z), -\psi_1{\prime}(x)\psi_2(z)) \tag{45}$$

where

$$\psi_1(x) = 1 - \frac{(x - \frac{x_{max}}{2})^4}{(\frac{x_{max}}{2})^4}, \qquad \psi_2(z) = 1 - \frac{(z - \frac{z_{min}}{2})^2}{(\frac{-z_{min}}{2})^2} \tag{46}$$

with $x_{max} = 500$ and $z_{min} = -500$. Fig 2 shows the horizontal, u, and vertical, w, velocity profiles. We solve for one tracer equation with initial condition

$$T(x,z) = 0.5(1 + \tanh(2\psi_3 - 1)) \tag{47}$$

where $\psi_3(x,z) = \psi_1(x)\psi_2(z)$. The tracer T goes from 0 at boundary of the domain to 1 in the center, as Fig 3 shows.

Assuming zero tracer diffusion, the initial condition (47) is an exact solution of the tracer equation. Solving the tracer equation with ETD-CPG considering no diffusion, the steady state is indeed maintained over time, as Fig 3 shows for

$t = 6$ hours. In order to investigate the convergence properties of the ETD scheme, we compare the solutions for various time-steps to a temporally over-refined solution computed with RK4. Figure 4 shows that ETD-CPG exhibits second-order accuracy, as predicted by theory.

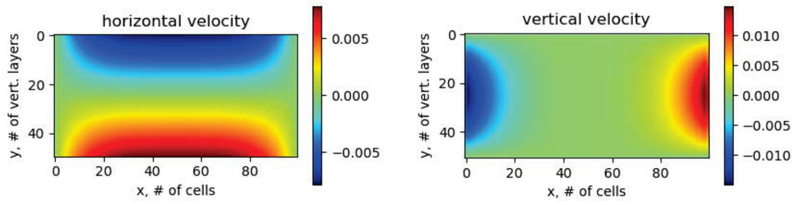

FIGURE 2. Initial conditions for horizontal and vertical velocity for the steady state test case.

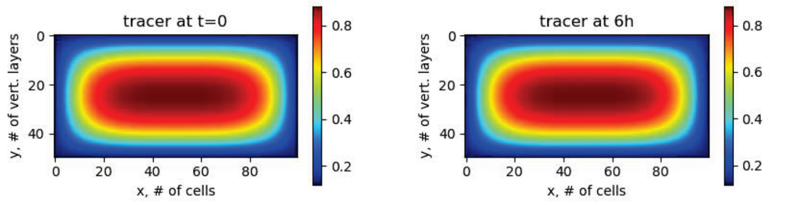

FIGURE 3. Initial condition for the tracer and steady state solution after 6 hours.

4.2. Test-case 1: quasi-uniform meshes. Now, we test the performance of ETD-CPG on a test case with a realistic ocean configuration. We consider two quasi-uniform meshes, QU120 and QU60, with resolutions of 120km and 60km, respectively. QU120 has a total of 29,223 cells, whereas QU60 has 116,643 cells. We study the performance of ETD-CPG in terms of scalability and CPU time, and compare it against the semi-implicit time-stepping scheme RK4+IE, where implicit Euler is used for the treatment of the vertical diffusion. A comparison regarding the efficiency of the schemes (error vs run-time) is also provided at the end of the section. Overall, RK4+IE is first-order accurate and treats the vertical advection explicitly, whereas ETD-CPG is second-order accurate and treats the vertical advection exactly, with a matrix exponential. For the computation of the φ_1 matrices, we chose two different values of M in the scaling and squaring algorithm, $M = 2$ and $M = 0$. With $M = 0$, we only perform the first step in Algorithm 2 presented in section 3.3, i.e. we consider a Taylor approximation of the exponential

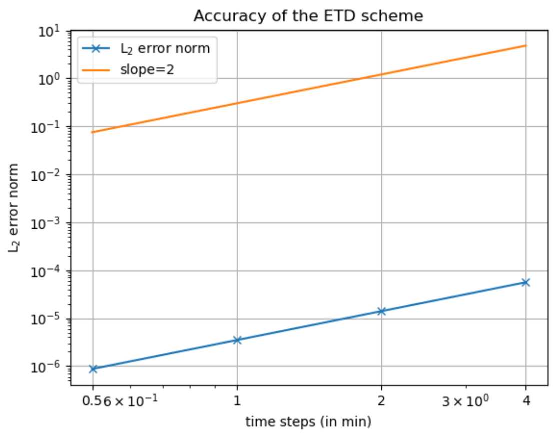

FIGURE 4. Convergence rate of the time stepping scheme.

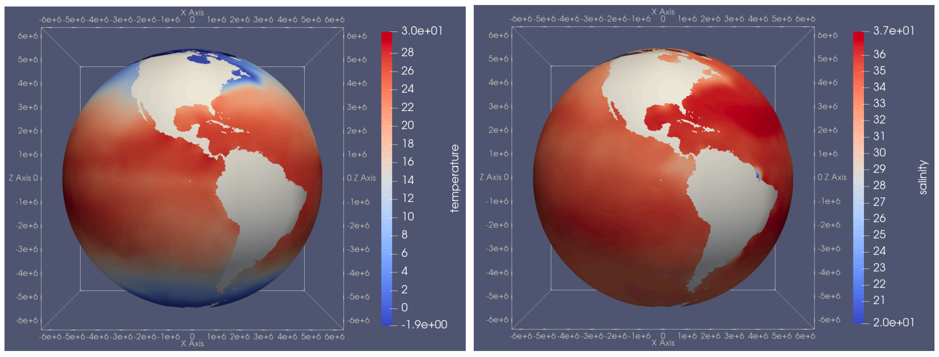

FIGURE 5. Initial conditions for temperature (left) and salinity (right) on a QU60 mesh.

and substitute it in the definition of the φ_1-function, eq. (25). From now on, we are going to call ETD-2 the ETD-CPG method that uses the scaling and squaring algorithm with $M = 2$, whereas ETD-0 will refer to the ETD-CPG method that uses the scaling a scaling algorithm with $M = 0$. In ETD-0 the computation of the φ_1 matrices is less accurate than in ETD-2, but for realistic test cases, like those presented in this work, this loss of accuracy does not significantly affect the overall accuracy and stability of the solution. From a physics point of view, in realistic global ocean simulations, most processes do not produce a significant movement of energy or waves in the vertical direction, as forcing by the wind or by currents in the upper layers of water are involved, where transport happens manly in the horizontal direction and at large time-scales. Mathematically, this behaviour is reflected in the structure of the $\Delta t J_n^{z,i}$ matrices, specifically in having $\Delta t J_n^{z,i}$ matrices with a norm close to one. For this reason, since we are approximating φ_1-functions of 'well-behaved' matrices, using $M = 2$ or $M = 0$ does not have a significant impact on the overall accuracy and stability of the ETD method. In localized phenomena where processes happening at faster time-scales are predominant, like eddies, $M = 0$ may not provide an accurate enough approximation for the φ_1-functions, because the $\Delta t J_n^{z,i}$ matrices might have a norm considerably greater than one.

Table 2. CPU time for the QU120 and QU60 mesh for different time-steps. The number of vertical layers is 64 and the run time is 6 hours. For all the runs, 32 cores have been used.

	QU120			QU60		
dt	RK4 + IE	ETD-2	ETD-0	RK4 + IE	ETD-2	ETD-0
60m	15.86s	4.58s	3.12s	55.91s	17.38s	11.72s
30m	31.77s	9.13s	6.20s	111.69s	34.68s	23.30s
15m	63.43s	18.22s	12.36s	223.80s	69.22s	46.36s
8m	118.14s	34.12s	23.17s	418.45s	129.99s	86.87s

In this test case, the computation of the tracer advection tendencity uses a standard finite volume (FV) algorithm in MPAS-O discretization (eqs. (4) and (15)). Specifically, horizontal advection of tracers uses a third-order FV algorithm, whereas for the vertical advection a second-order FV algorithm is used. We solve for five tracers, where two of them (temperature and salinity) are active, meaning they feed back on the dynamics through the density and pressure terms. The remaining three are passive and are simply transported by the flow. The initial conditions for temperature and salinity are interpolated from the Polar Science Center Hydrographic Climatology [23] and are shown in Fig. 5. All five tracers share the same linear operator J_n^z, so the matrices $\varphi_1(\Delta t J_n^{z,i})$ are the same for all tracers and are computed only once per time-step. The model is run for six hours, and the time steps used for the simulations are dictated by the dynamics, meaning that we use (baroclinic) time-steps for which the split-explicit scheme in MPAS-Ocean is stable in resolving the momentum and thickness equations. In all the tables presented, the average wallclock times over three repeated simulations are shown.

Table 2 shows the CPU times for RK4+IE, ETD-2 and ETD-0 on the meshes QU120 and QU60. Four different time-steps have been used, with 60m being the largest possible time-step dictated by the dynamics to ensure stability. For the QU120 mesh, we see that ETD-2 is 3.5 times faster than RK4+IE for all time-steps considered, and the gain is even bigger with ETD-0, which is 5.1 times faster than RK4+IE. Similar speed-ups can be seen for the QU60 mesh, where ETD-2 is 3.2 times faster than RK4+IE for all time-steps considered, and ETD-0 is 4.8 times faster. We expected to have bigger gains with ETD-0 since it is less expensive than ETD-2. The large improvement in time is primarily because the older RK4 scheme requires multiple evaluations of all the tendency (RHS) terms for a given time step while the ETD schemes only require one evaluation per step. All simulations in Table 2 have been run with 64 vertical layers, which is a realistic value for present day ocean simulations. Having 64 layers in the vertical implies that the φ_1 matrices in the simulations presented are, at most, 64×64, i.e. the ETD procedure requires forming many (one for each horizontal cell) 64×64 matrices in parallel.

In table 3, we investigate how the number of vertical layers impacts the performance of the ETD methods in terms of CPU time. For all the runs, the largest possible time-step (dt = 60m), has been used. For ETD-2 and ETD-0 we expect the CPU time to quadruple when doubling the number of layers, since matrices with N_z^2 elements are multiplied. The CPU time should only double for RK4+IE, so the performance advantage for ETD is expected to decrease with the number of layers. This behavior is indeed demonstrated in Table 3. With fewer layers, the matrix multiplication is a relatively small fraction of the total cost and the time increases

TABLE 3. CPU time for the QU120 and QU60 mesh for different numbers of vertical layers. The time-step used is dt= 60m. For all the runs, 32 cores have been used.

	QU120			QU60		
layers	RK4+IE	ETD-2	ETD-0	RK4+IE	ETD-2	ETD-0
16	4.16s	0.75s	0.65s	14.89s	2.75	2.45s
32	8.05s	1.65s	1.35s	28.73s	6.35s	5.13s
64	15.79s	4.58s	3.11s	55.91s	17.38s	11.72s
128	31.00s	15.74s	10.48s	108.02s	60.73s	38.24s

TABLE 4. Scalability for QU60 with two different numbers of layers. The largest time step, dt= 60m, was used for all simulations.

	16 layers			64 layers		
cores	RK4 + IE	ETD-2	ETD-0	RK4 + IE	ETD-2	ETD-0
8	52.94s	10.14s	8.84s	197.43s	63.24s	40.41s
16	28.55s	5.40s	4.80s	106.39s	33.81s	22.58s
32	14.89s	2.72s	2.42s	55.94s	17.40s	11.68s
64	7.94s	1.42s	1.27s	30.01s	9.18s	6.30s
128	4.37s	0.73s	0.67s	16.04s	4.67s	3.22s

linearly with number of levels for all schemes. As the number of layers increases, the expected quadrupling begins to become evident, especially when moving from 64 to 128 levels. This range is particularly relevant since current production configurations use 60-100 layers, with the choice of resolution governed by the need to accurately represent surface mixed-layer processes and bathymetric features while minimizing overall cost of simulation. Nevertheless, ETD-2 and ETD-0 are considerably faster than RK4+IE for all cases considered. The largest difference is with 16 layers, where ETD-2 and ETD-0 are more than 5.4 times faster than RK4+IE for both mesh considered. With 128 vertical layers, ETD-2 is 2 times faster than RK4+IE on QU120 and 1.8 times faster on QU60, whereas ETD-0 is 3 times faster than RK4+IE on QU120 and 2.8 times faster on QU60. Since this is an initial implementation, there remain a number of optimizations that can still be exploited to reduce costs. In addition to CPU optimizations, the use of accelerators or even tensor cores would dramatically reduce the overall cost.

Finally, Table 4 shows the scalability of the two ETD methods compared with that of RK4+IE for two different numbers of vertical layers. Only the QU60 mesh was used, and the largest time step, dt= 60m, was adopted for all simulations. All three methods exhibit good scalability, meaning that the CPU times nearly halve by doubling the number of cores, as is expected for a scheme that does not involve communication between cores.

As mentioned at the beginning of the section, another important advantage of the ETD schemes is that they are more accurate than RK4+IE. Fig. 6 shows the efficiency (error vs run-time) of the three schemes on the QU60 mesh for the four time-steps used in Table 2 (60m, 30m, 15m and 8m). ETD-0 and ETD-2 both show second-order accuracy whereas RK4+IE shows first-order. For all five tracers (temperature, salinity and three passive tracers) the convergence rates coincide. To produce the plot, velocity and thickness of the vertical layers have been fixed

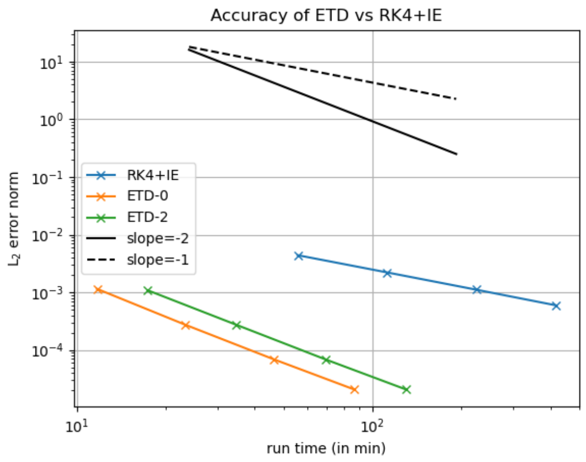

Figure 6. Convergence rate of RK4+IE, ETD-0 and ETD-2 on the QU60 mesh. Simulation have been run with time-steps of 60m, 30m, 15m and 8m for all three schemes.

(maintained constant) over time, and errors have been computed against a reference solution obtained with RK4 with a small time-step.

4.3. Test-case 2: variable resolution mesh. In this test case, a variable resolution mesh named EC60to30 is used. The resolution varies from 30 km at the equator and poles to 60 km at the mid-latitudes. The number of vertical layers used is 60. The grid contains 235,160 cells, 714,274 edges, and 478,835 vertices on each layer. As for the previous test case, we use a third-order accurate standard FV scheme in MPAS-O discretization for the computation of the horizontal tracer advection tendency and a second-order FV scheme for the vertical advection tendency. We solve for 5 tracers, two of them being temperature and salinity, whereas the remaining three are passive tracers. The initial conditions for temperature and salinity are interpolated from the Polar Science Center Hydrographic Climatology, and are shown in Fig. 7. The performance of ETD-2 and ETD-0 in terms of CPU time is shown in Table 5 compared with that of RK4+IE. The averaged wallclock times are presented after repeating each simulation three times. The model is run for 10 days, and 128 cores have been used for all runs. Results are shown for three different time-steps with 90 minutes being the largest possible stable time-step allowed by the dynamics. As for the previous test case, ETD-2 and ETD-0 are considerably faster than RK4+IE. For all time-steps considered, ETD-2 is 3.15 times faster than RK4+IE, and ETD-0 is 5.36 times faster. Since in realistic global ocean simulations most processes occur in the horizontal and involve large scales, the lower accuracy in the computation of the φ_1-function in ETD-0 does not compromise the overall accuracy and stability of the exponential integrator, allowing a speed-up of more than 40% over ETD-2.

5. Conclusions

In this work, we presented an exponential integrator for solving the tracer equations used in ocean modeling and studied its performance on global realistic ocean test cases. The method is a second-order two-stage ETD scheme that treats all

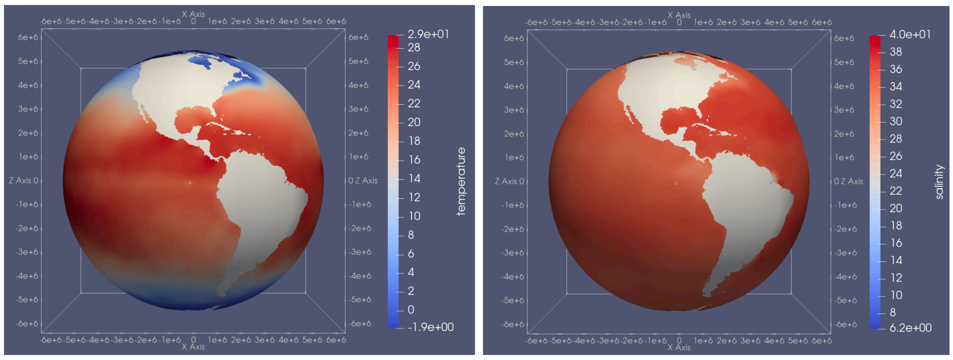

Figure 7. Initial conditions for temperature (left) and salinity (right) on a EC60to30 mesh.

Table 5. CPU time for the EC60to30 test case for different time-steps. the number of vertical layers is 60. For all runs, 128 cores have been used.

	EC60to30		
dt	RK4 + IE	ETD-2	ETD-0
90m	813.71s	258.47s	151.71s
60m	1221.65s	388.14s	227.66s
30m	2441.69s	773.99s	454.08s

the vertical terms with a matrix exponential and all the horizontal terms explicitly. The vertical terms are treated exponentially to avoid the restrictive explicit time step constraints due to fast mixing time-scales and the use of smaller vertical mesh spacing near the surface in ocean models. The computation of the matrix functions $\varphi_1(\Delta J_n^{z,i})$ uses a scaling and squaring method based on polynomials of moderate degree. In the two global ocean test cases presented, we chose two different values of M in the scaling and squaring algorithm, $M = 2$ and $M = 0$, producing the ETD-2 and ETD-0 methods, respectively. We compared the performance of ETD-2 and ETD-0 against that of another semi-implicit method, RK4+IE, where implicit Euler is used for the treatment of the vertical diffusion. In the tests presented, we found that ETD-2 and ETD-0 are in general faster than RK4+IE, and at the same time more accurate, since they are second-order accurate whereas RK4+IE is only first-order accurate. Considering 60 layers in the vertical, which is a realistic value for present day ocean simulations, ETD-2 is shown to be 3 times faster than RK4+IE in both test cases. The gain in time produced by ETD-0 is even bigger, with this method being 4.8 to 5.3 times faster than RK4+IE in the two test cases presented. The bigger gain produced by ETD-0 is due to the fact that in this method the $\varphi_1(\Delta J_n^{z,i})$ are computed less accurately than in ETD-2. Since in global realistic ocean simulations most processes happen in the horizontal at large time-scale, we can take advantage of the physics of the problem and compute the φ_1-functions in a cheaper way ($M = 0$) without compromising the overall accuracy and stability of the simulations. For more localized phenomena where fast vertical mixing and/or fast vertical transport are predominant, then the accuracy in the computation of the φ_1 would have a bigger impact, and $M = 0$ may not provide an accurate enough approximation for the φ_1-functions. In test case 1, we also showed that considering a larger number of vertical layers, 128, the ETD methods presented are from 2 to 3 times faster than RK4+IE, and they show good parallel efficiency with different number of cores ranging from 8 to 128.

In conclusion, the ETD method studied in this work appears to be a valid time-stepping scheme for solving the tracer equations in modern ocean models. Future work will consists in coupling this time-stepping scheme with the semi-implicit scheme for the dynamics presented in [24] to use even larger time-steps.

Acknowledgments

This work was supported by the Efficient and Scalable Time-Stepping Algorithms for Ocean System Simulations as part of the Scientific Discovery through Advanced Computing (SciDAC) program within the DOE Office of Science, Offices of Biological and Environmental Research and Advanced Scientific Computing Research. This research used resources provided by the Los Alamos National Laboratory Institutional Computing Program, which is supported by the U.S. Department of Energy National Nuclear Security Administration under Contract No. 89233218C-NA000001.

References

[1] M.R. Petersen, D. Jacobsen, W. Douglas, T.D. Ringler, M.W. Hecht, M.E. Maltrud, Evaluation of the arbitrary Lagrangian-Eulerian vertical coordinate method in the MPAS-Ocean model, Ocean Modelling, 86, pp. 93-113, 2015.

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

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

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

[5] L. Ju, T. Ringler, M. Gunzburger. Voronoi tessellations and their application to climate and global modeling. Numerical techniques for global atmospheric models. In: P.H. Lauritzen, C. Jablonowski, M.A. Taylor, R.D. Nair, (Eds.), Springer Lecture Notes in Computational Science and Engineering, pp. 1-30, 2010.

[6] M. Hochbruck, A. Ostermann. Exponential integrators, Acta Numerica, 19, pp. 209-286, 2010.

[7] M. Hochbruck, A. Ostermann. Explicit exponential RungeCKutta methods for semilinear parabolic problems, SIAM Journal on Numerical Analysis, 43 (3), pp. 1069-1090, 2005.

[8] R. Smith, P. Jones, B. Briegleb, F. Bryan, G. Danabasoglu, J. Dennis, J. Dukowicz, C. Eden, B. Fox-Kemper, P. Gent, et al. The parallel ocean program (POP) reference manual ocean component of the community climate system model (CCSM) and community earth system model (CESM), Rep. LAUR-01853 141, 2010, pp. 1-140.

[9] A. Adcroft, J.M. Campin, S. Dutkiewicz, C. Evangelinos, D. Ferreira, G. Forget, B. Fox-Kemper, P. Heimbach, C. Hill, E. Hill. MITgcm documentation, Release checkpoint67a-12-gbf23121, 19, 2018.

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

[11] K. Pieper, K.C. Sockwell, M. Gunzburger. Exponential time differencing for mimetic multi-layer ocean models, Journal of Computational Physics, 398, p. 108900, 2019.

[12] M. Schreiber, N. Schaeffer, R, Loft, Exponential integrators with parallel-in-time rational approximations for the shallow-water equations on the rotating sphere, Parallel Computing, 85, pp. 56-65, 2019.

[13] S. Calandrini, K. Pieper, M. Gunzburger. Numerical analyses of exponential time-differencing schemes for the solution of atmospheric models, Quarterly Journal of the Royal Meteorological Society, 147 (736), pp. 1477–1496, 2021.

[14] C. Clancy, J.A. Pudykiewicz. On the use of exponential time integration methods in atmospheric models, Tellus A: Dynamic Meteorology and Oceanography, 65 (1), 20898, 2013.
[15] F. Garcia, L. Bonaventura, M. Net, J. Sanchez. Exponential versus IMEX high-order time integrators for thermal convection in rotating spherical shells, Journal of Computational Physics, 264, pp. 41-54, 2014.
[16] M. Schreiber, N. Schaeffer, R. Loft. Exponential integrators with parallel-in-time rational approximations for the shallow-water equations on the rotating sphere, Parallel Computing, 85, pp. 56-65, 2019.
[17] 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, pp. 827-848, 2016.
[18] G. Beylkin, J.M. Keiser, L. Vozovoi. A new class of time discretization schemes for the solution of nonlinear PDEs, Journal of Computational Physics, 147 (2), pp. 362-387, 1998.
[19] S. Koikari. An error analysis of the modified scaling and squaring method, Computers and Mathematics with Applications, 53 (8), pp. 1293-1305, 2007.
[20] V.T.Luan, J.A. Pudykiewicz, D.R. Reynolds. Further development of efficient and accurate time integration schemes for meteorological models, Journal of Computational Physics, 376, pp. 817-837, 2019.
[21] J. Huang, L. Ju, B. Wu. An exponential time-integrator scheme for steady and unsteady inviscid flows, Journal of Computational Physics, 365, pp. 206-225, 2018.
[22] M. Yano, J. D. Penn, G. Konidaris, and A. T. Patera. Math, Numerics and Programming: For Mechanical Engineers. Cambridge, MA, USA: MIT OpenCourseWare, 2013.
[23] M. Steele, R. Morley, W. Ermold, PHC: A global ocean hydrography with a high-quality arctic ocean, Journal of Climate, 14 (9), pp. 2079-2087, 2001.
[24] H.G. Kang, K.J. Evans, M.R. Petersen, M. R., P.W. Jones, S. Bishnu. A scalable semi-Implicit barotropic mode solver for the MPAS-Ocean, Journal of Advances in Modeling Earth Systems, 13 (4), e2020MS002238, 2021.
[25] N.J. Higham. The scaling and squaring method for the matrix exponential revisited, SIAM Journal on Matrix Analysis and Applications, 26 (4), pp. 1179-1193, 2005.
[26] R. Chinomona, D.R. Reynolds. Implicit-explicit multirate infinitesimal GARK methods, SIAM Journal on Scientific Computing, 43 (5), pp. A3082-A3113, 2021.
[27] F.X. Giraldo, J.F. Kelly, E.M. Constantinescu. Implicit-explicit formulations of a three-dimensional nonhydrostatic unified model of the atmosphere (NUMA), SIAM Journal on Scientific Computing 35 (5), pp. B1162-B1194, 2013.
[28] C.J. Vogl, A. Steyer, D.R. Reynolds, P.A. Ullrich, C.S. Woodward. Evaluation of implicit-explicit additive Runge-Kutta integrators for the HOMME-NH dynamical core, Journal of Advances in Modeling Earth Systems, 11 (12), pp. 4228-4244, 2019.

Los Alamos National Laboratory, Los Alamos, NM, 87544
E-mail: `scalandrini@lanl.gov`

Los Alamos National Laboratory, Los Alamos, NM, 87544
E-mail: `pwjones@lanl.gov`

Los Alamos National Laboratory, Los Alamos, NM, 87544
E-mail: `mpetersen@lanl.gov`