High-order Tensor-Train Finite Volume Methods for Shallow Water Equations

M. Engin Danis,[a] Duc P. Truong,[a] Derek DeSantis,[b] Jeremy Lilly,[b] Mark R. Petersen,[b] Kim Ø. Rasmussen,[a] and Boian S. Alexandrov,[a]

[a] *Theoretical Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

[b] *Computer, Computational and Statistical Sciences Division, Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

Corresponding author: M. Engin Danis, danis@lanl.gov

ABSTRACT: In this paper, we introduce high-order tensor-train (TT) finite volume methods for the Shallow Water Equations (SWEs). We present the implementation of the 3$^{\text{rd}}$ order Upwind and the 5$^{\text{th}}$ order Upwind and Weighted Essentially Non-Oscillatory (WENO) reconstruction schemes in the TT format. It is shown in detail that the linear upwind schemes can be implemented by directly manipulating the TT cores while the WENO scheme requires the use of TT cross interpolation for the nonlinear reconstruction. In the development of numerical fluxes, we directly compute the flux for the linear SWEs without using TT rounding or cross interpolation. For the nonlinear SWEs where the TT reciprocal of the shallow water layer thickness is needed for fluxes, we develop an approximation algorithm using Taylor series to compute the TT reciprocal. The performance of the TT finite volume solver with linear and nonlinear reconstruction options is investigated under a physically relevant set of validation problems. In all test cases, the TT finite volume method maintains the formal high-order accuracy of the corresponding traditional finite volume method. In terms of speed, the TT solver achieves up to 124x acceleration of the traditional full-tensor scheme.

1. Introduction

As computer architectures evolve, new algorithms are often needed to make the best use of the latest equipment. For example, in the 1990s there was a major transition from vector supercomputers to distributed memory clusters, and internode communication became the slowest part of a simulation. In global atmospheric models this spurred a transition from spectral methods (Kiehl et al. 1998), which require global communications for the transforms between physical and spectral space, to methods such as finite volume (Thuburn et al. 2009) and spectral element (Dennis et al. 2005), which only require local communication with nearby processors.

In recent years there has been another major change in the landscape of supercomputer architectures, as the main driver for sales became machine learning (ML) and artificial intelligence (AI) applications. Tensor cores are a new generation of chips specifically designed for AI and deep learning, such as the NVIDIA Volta, Turing, and Ampere classes of GPUs. Because AI, rather than computational physics, is driving the direction of commodity architecture, developers of numerical methods must seek out algorithms that are performant on this new hardware. The AI revolution has also spurred the development of libraries that are specifically tuned to run AI algorithms, such as Neural Networks, on tensor cores. Here we present recently developed numerical methods that leverage tensor networks. These methods manipulate large-scale data based on generalizations of the singular value decomposition to higher dimensional tensor arrays.

A second driver of new algorithm development is to greatly increase the speed and resolution of global climate simulations. High-resolution global simulations are now typically 6km to 10km in the ocean and atmosphere (Caldwell et al. 2019) and the latest cloud-resolving simulations are at 3km resolution (Donahue et al. 2024). This involves millions of horizontal cells and up to 128 vertical layers. In addition, climate research requires long simulations for spin-up, and ensembles of simulations to investigate the intrinsic variability of the climate system (Kay et al. 2015). All these factors taken together produce the "curse of dimensionality", where simulation campaigns in climate science require many months on large supercomputers.

Tensor Networks (TNs) (Cichocki 2014) are a promising new approach that mitigate the effects of the curse of dimensionality, and may also take advantage of specialized AI hardware. TNs are a generalization of matrix factorizations to higher dimensions, whereby multidimensional data structures (tensors) are decomposed into manageable blocks. The computations normally performed on the large dataset (e.g. finite differences) can alternatively be performed on these smaller tensor components, drastically reducing computational costs. The most popular TN method is known as the Tensor Train (TT). In TT format, the large dataset is decomposed into a sequence of lower-dimensional tensors (cores), linked together in a chain (train), to efficiently represent and manipulate large-scale data (Oseledets and Tyrtyshnikov 2010).

TN techniques show great promise to attack the curse of dimensionality across a large range of problems. Just in the last few years, tensor methods have been used to model the Navier-Stokes equations in a number of standard test cases: A backward-facing step (Demir et al. 2024), a lid-driven cavity (Kiffner and Jaksch 2023), a turbulent flow in a channel (von Larcher and Klein 2019), and a 3-dimensional Taylor-Green vortex (Gourianov 2022). In recent work, we used TT methods to accelerate compressible flow simulations by up to 1000 times (Danis et al. 2025) and solved the time-independent Boltzmann neutron transport equation with tensor networks (Truong et al. 2024). These successes show that a tensor-based approach to fluids problems is both possible and promises greatly increased model efficiency. There are similarities between the methods presented here to those in (Danis et al. 2025), but the present work serves to compliment (Danis et al. 2025) by *extending these TN techniques and results to models of oceanic or atmospheric circulation*, which has not been done to date. A discussion comparing methods from (Danis et al. 2025) to the those in present work can be found in Section 4b. Successful compression and speedup of geophysical fluid simulations via TN has the promise to radically alter the landscape of weather and climate modeling, allowing researchers to explore higher resolutions and larger ensembles.

Any new numerical method for atmosphere and ocean models must progress through a sequence of verification steps in order to be accepted by the community. The shallow water equations (SWEs) are a reduced equation set used as a first step for modeling geophysical fluids at the climate scale (Thuburn et al. 2009; Weller et al. 2009; Archibald et al. 2011; Lilly et al. 2023). In particular, the SWEs serve as a suitable starting point for methods targeting layered Boussinesq models, which can be thought of as a vertical stack of shallow water models with additional vertical and surface processes. The SWEs contain the relevant dynamics of atmospheric and oceanic flows: the Coriolis force and pressure gradient term for geostrophic balance, as well as horizontal advection of momentum and mass. At the same time, the SWEs are simple enough for rapid code development. Critically, the SWEs may be tested against exact solutions during model development (Bishnu et al.

2024), which is not the case for the subsequent levels of complexity in the development sequence (Petersen et al. 2015). The key assumptions of the SWEs are that horizontal scales of motion are much larger than the vertical, which means it is hydrostatic, and that the fluid is incompressible so that it has uniform density (Cushman-Roisin and Beckers 2011). This conveniently avoids the complexities of the equation of state, moist dynamics and clouds in the atmosphere, and vertical advection. Published test sets using the SWEs have become an essential component of model development and verification (Williamson et al. 1992; Calandrini et al. 2021).

This article assesses the computational advantages of TN in modeling the SWEs across a range of test cases. In particular, we focus on TT methods for high-order finite volume methods for the SWEs. The paper is organized as follows. In Section 2, we review the SWEs and our high-order methods for solving them. Section 3 we cover the basics of the TT decomposition, and in Section 4 we discuss how the finite volume scheme can be formulated in the TT format. We end with Section 5 covering the results for a series of test cases.

2. Governing Equations and the Numerical Method

In this section, we will review the shallow water equations (SWEs) and the finite volume method used to solve these equations. This discussion will only involve essential information for a typical finite volume implementation on traditional grids to lay the ground for the tensor-train implementation.

a. *Shallow Water Equations*

In this study, we consider both linear and nonlinear SWEs with a flat bottom topography. Both equations are solved in the conservative form,

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} = \boldsymbol{S}, \tag{1}$$

where $\boldsymbol{U}$ is the vector of conserved variables, $\boldsymbol{F}, \boldsymbol{G}$ are the fluxes in x and y directions, and $\boldsymbol{S}$ is the source term.

In the linear case, Equation (1) is solved with

$$\boldsymbol{U} = \begin{pmatrix} \eta \\ u \\ v \end{pmatrix}, \qquad \boldsymbol{F} = \begin{pmatrix} Hu \\ g\eta \\ 0 \end{pmatrix}, \qquad \boldsymbol{G} = \begin{pmatrix} Hv \\ 0 \\ g\eta \end{pmatrix}, \qquad \boldsymbol{S} = \begin{pmatrix} 0 \\ fv \\ -fu \end{pmatrix}, \tag{2}$$

where the vector of conserved variables, $\boldsymbol{U}$, consists of the surface elevation η, the x-velocity u and the y-velocity v. Furthermore, H is the mean depth of the fluid at rest, f is the Coriolis parameter, and g is the acceleration of gravity.

In the nonlinear case, Equation (1) is solved with

$$\boldsymbol{U} = \begin{pmatrix} h \\ hu \\ hv \end{pmatrix}, \qquad \boldsymbol{F} = \begin{pmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{pmatrix}, \qquad \boldsymbol{G} = \begin{pmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{pmatrix}, \qquad \boldsymbol{S} = \begin{pmatrix} 0 \\ fhv \\ -fhu \end{pmatrix}, \tag{3}$$

where h is the fluid layer thickness.

b. *High-order Finite Volume Method for Hyperbolic Conservation Laws*

For simplicity, let us consider the 2-dimensional scalar hyperbolic conservation law,

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} = 0, \tag{4}$$

where u is a generic conserved variable (not to be confused with the x-velocity), $f(u)$ and $g(u)$ are fluxes in x and y directions respectively, and we assume that proper initial and boundary conditions are provided. On a uniform mesh with grid spacing Δx and Δy in x and y directions, a finite volume method solves Equation (4) for the cell averages of u in a given cell (i, j),

$$\widetilde{\overline{u}}_{i,j} = \frac{1}{\Delta x \Delta y} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} u \, \mathrm{d}x \, \mathrm{d}y\,, \tag{5}$$

where $\overline{u}$ denotes cell average in x and $\widetilde{u}$ denotes cell average in y.

In this spirit, the semi-discrete cell-averaged form of Equation (4) for a cell (i,j) is given as

$$\begin{aligned}\frac{d\widetilde{\overline{u}}_{i,j}}{dt}&+\frac{1}{\Delta x\Delta y}\int_{y_{j-1/2}}^{y_{j+1/2}}\left(f\left(u\left(x_{i+1/2},y\right)\right)-f\left(u\left(x_{i-1/2},y\right)\right)\right)\mathrm{d}y\\&+\frac{1}{\Delta x\Delta y}\int_{x_{i-1/2}}^{x_{i+1/2}}\left(g\left(u\left(x,y_{j+1/2}\right)\right)-g\left(u\left(x,y_{j-1/2}\right)\right)\right)\mathrm{d}x=0,\end{aligned}\tag{6}$$

which can be approximated as

$$\frac{d\widetilde{\overline{u}}_{i,j}}{dt}+\frac{\widehat{f}_{i+1/2,j}-\widehat{f}_{i-1/2,j}}{\Delta x}+\frac{\widehat{g}_{i,j+1/2}-\widehat{g}_{i,j-1/2}}{\Delta y}=0,\tag{7}$$

where the numerical fluxes $\widehat{f}_{i\pm1/2,j}$ and $\widehat{g}_{i,j\pm1/2}$ approximate the surface integrals by the 1-dimensional Gauss-Legendre quadrature rule with quadrature points δ_m and weights w_m:

$$\begin{aligned}\widehat{f}_{i\pm1/2,j}&=\sum_m w_m\widehat{f}\left(u^-_{i\pm1/2,y_j+\delta_m\Delta y},u^+_{i\pm1/2,y_j+\delta_m\Delta y}\right),\\\widehat{g}_{i,j\pm1/2}&=\sum_m w_m\widehat{g}\left(u^-_{x_i+\delta_m\Delta x,j\pm1/2},u^+_{x_i+\delta_m\Delta x,j\pm1/2}\right),\end{aligned}\tag{8}$$

where $\widehat{f}(u^-,u^+)$ and $\widehat{g}(u^-,u^+)$ denote the local Lax-Friedrichs flux. In the x-direction, for example, the local Lax-Friedrichs flux is defined as

$$\widehat{f}(u^-,u^+)=\frac{1}{2}\left(f(u^-)+f(u^+)\right)-\frac{\lambda}{2}\left(u^+-u^-\right),\tag{9}$$

where $u^\pm$ are point-wise values at the cell interfaces approximated by a high-order reconstruction method from the cell-averages $\widetilde{\overline{u}}_{i,j}$ and $\lambda=\max_{u\in(u^-,u^+)}|f'(u)|$.

c. *High-order Reconstructions*

In this paper, we implement the 3rd order upwind-biased (Upwind3), 5th order upwind-biased (Upwind5), and 5th order Weighted Essentially Non-Oscillatory (WENO5) reconstruction methods. Upwind methods are based on solution reconstruction using the ideal weights obtained from the best polynomial approximation that matches the cell-averages in the relevant computational stencil. However, they become extremely oscillatory when the numerical solution develops discontinuities, such as shock waves, which makes the numerical solution eventually unstable. For those cases, WENO methods provide a robust numerical solution near discontinuities while maintaining the high-order accuracy in the smooth regions of the solution by blending the ideal reconstruction weights with smoothness indicators to obtain a nonlinear reconstruction. We refer the interested readers to (Shu 1998) for more details.

To obtain a high-order 2-dimensional finite volume discretization, we follow the dimension-by-dimension reconstruction method in (Shu 1998; Shi et al. 2002). This involves two 1-dimensional reconstruction steps for each direction. In Figure 1, the reconstruction procedure for the 3$^{\text{rd}}$ order reconstruction in the x-direction is depicted, for which we only use two Gauss quadrature points for approximating the surface integrals. Let $\bar{\cdot}$ and $\tilde{\cdot}$ denote cell averages in the x- and y-directions respectively. Then, starting with the cell average $\tilde{\bar{u}}_{i,j}$, Step 1 is to perform the first 1-dimensional reconstruction in the x-direction at $x = x_{i\pm1/2}$; this gives 1-dimensional cell averages $\tilde{u}^{\pm}_{i\mp1/2}$. Then in Step 2, we perform a second 1-dimensional reconstruction, now in the y-direction, to obtain point-wise values $u^{\pm}_{i\mp1/2,y_j+\delta_2\Delta y}$ at the Gauss quadrature points at $x = x_{i\pm1/2}$ and $y \in \{y_j+\delta_1\Delta y, y_j+\delta_2\Delta y\}$.

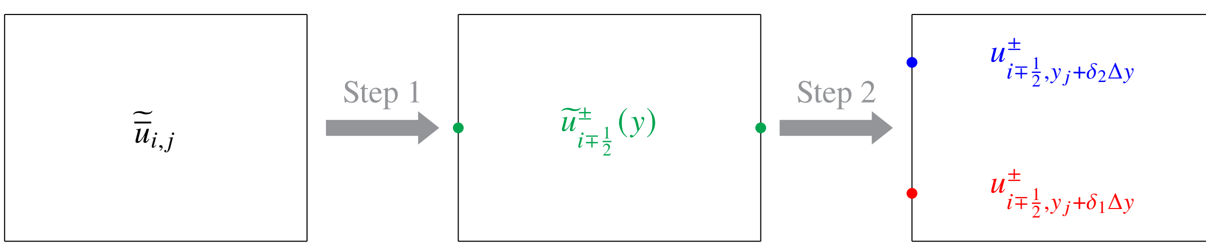


FIG. 1: Step-by-step 3$^{\text{rd}}$ order reconstruction from cell averages in the x-direction.

The 5$^{\text{th}}$ order reconstruction Steps 1 and 2 for Upwind5 and WENO5 are similar; we present the details for all three reconstructions in Appendix A1.

d. *Finite Volume Method for the Shallow Water Equations*

In this study, we solve the cell-averaged SWEs,

$$\frac{\partial \tilde{\overline{\boldsymbol{U}}}_{i,j}}{\partial t} + \frac{\widehat{\boldsymbol{F}}_{i+1/2,j} - \widehat{\boldsymbol{F}}_{i-1/2,j}}{\Delta x} + \frac{\widehat{\boldsymbol{G}}_{i,j+1/2} - \widehat{\boldsymbol{G}}_{i,j-1/2}}{\Delta y} = \tilde{\overline{\boldsymbol{S}}}_{i,j}\,, \tag{10}$$

on a uniform Cartesian mesh using the finite volume method where the fluxes vectors are computed by the Gauss-Legendre quadrature rule

$$\begin{aligned}
\widehat{\boldsymbol{F}}_{i\pm1/2,j} &= \sum_m w_m \widehat{\boldsymbol{F}}\left(\boldsymbol{U}^-_{i\pm1/2,y_j+\delta_m\Delta y}, \boldsymbol{U}^+_{i\pm1/2,y_j+\delta_m\Delta y}\right), \\
\widehat{\boldsymbol{G}}_{i,j\pm1/2} &= \sum_m w_m \widehat{\boldsymbol{G}}\left(\boldsymbol{U}^-_{x_i+\delta_m\Delta x,j\pm1/2}, \boldsymbol{U}^+_{x_i+\delta_m\Delta x,j\pm1/2}\right),
\end{aligned} \tag{11}$$

with the Local Lax-Friedrichs flux

$$\begin{aligned}
\widehat{\boldsymbol{F}}\left(\boldsymbol{U}^-,\boldsymbol{U}^+\right) &= \frac{1}{2}\left(\boldsymbol{F}(\boldsymbol{U}^-)+\boldsymbol{F}(\boldsymbol{U}^+)\right) - \frac{\lambda_F}{2}\left(\boldsymbol{U}^+ - \boldsymbol{U}^-\right), \\
\widehat{\boldsymbol{G}}\left(\boldsymbol{U}^-,\boldsymbol{U}^+\right) &= \frac{1}{2}\left(\boldsymbol{G}(\boldsymbol{U}^-)+\boldsymbol{G}(\boldsymbol{U}^+)\right) - \frac{\lambda_G}{2}\left(\boldsymbol{U}^+ - \boldsymbol{U}^-\right),
\end{aligned} \tag{12}$$

where λ_F and λ_G are the maximum eigenvalues of the flux Jacobians $\left|\partial \boldsymbol{F}/\partial \boldsymbol{U}\right|$ and $\left|\partial \boldsymbol{G}/\partial \boldsymbol{U}\right|$, respectively. The high-order reconstruction is performed by applying the procedures discussed in Section 2c to $\boldsymbol{U}$ in a component-by-component fashion.

3. Tensor Train Decomposition

In this section we briefly introduce the tensor notation and the tensor train manipulation techniques we apply in this work. The goal of the machinery introduced in this section and the methods described in Section 4 is to use the tensor train format to obtain an approximation to the model state that is significantly compressed, then to evolve that compressed state forward in time. By performing all necessary operations in this compressed space, we can greatly reduce the number of floating point operations required to advance the model, thereby obtaining significant computational speedup. Figure 2 serves to provide a high-level explanation of the goal of tensor train methods like the ones used here.

For clarity, in the context of this work, a tensor is exactly a multi-dimensional array of arbitrary dimension, consisting of real entries, subject to nonlinear point-wise operations and linear transformations. For a positive integer d and a sequence of mode sizes $n_1, n_2, \ldots, n_d$, a tensor $\mathcal{X}$ of the given shape is an element of $\mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$. In this way, a $n_1 \times n_2$ matrix is a 2-dimensional tensor in $\mathbb{R}^{n_1\times n_2}$ and a length n_1 vector is a 1-dimensional tensor in $\mathbb{R}^{n_1}$. A d-dimensional tensor in $\mathbb{R}^{n_1\times n_2\times\cdots\times n_d}$ is

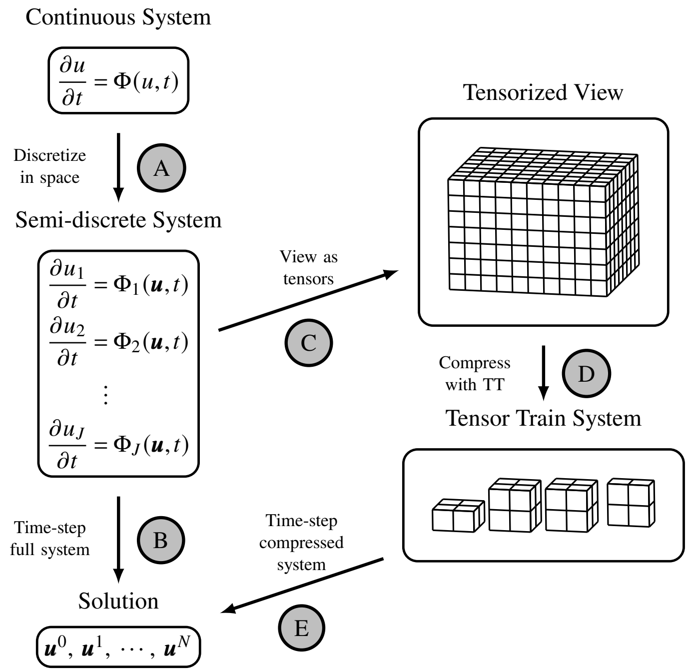


FIG. 2: Tensor train methods find speedups by avoiding time-stepping the full discrete system directly. Standard methods involve (A) discretizing the PDE in space, followed by (B) the use of an iterative time-stepping algorithm. Tensor train methods can be understood to instead (C) treat the discrete system as a large dimensional tensor, on which (D) a tensor train is fit. Speedup is found by (E) time-stepping in the reduced dimensional latent space of the tensor train, which scales linearly with the size of the problem instead of polynomially as in (B).

indexed by a multi-index $\boldsymbol{i} = (i_1, i_2, \ldots, i_d)$, where $i_k \in \{1, 2, \ldots, n_k\}$ for all $k = 1, 2, \ldots, d$; this is written as $\mathcal{X}(i_1, i_2, \ldots, i_d) \in \mathbb{R}$.

a. *Tensor Train*

Tensor train, or the TT-format of a tensor, represents a tensor of arbitrary dimension as a product of so-called *cores*, which are either 2- or 3-dimensional tensors (Oseledets and Tyrtyshnikov 2010). Generally, a TT representation $\mathcal{X}_{TT}$ of a d-dimensional tensor $\mathcal{X}$ is defined as

$$\mathcal{X}_{TT}(i_1, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_{d-1}}^{r_1, \ldots, r_{d-1}} \mathcal{G}_1(1, i_1, \alpha_1) \mathcal{G}_2(\alpha_1, i_2, \alpha_2) \ldots \mathcal{G}_{d-1}(\alpha_{d-2}, i_{d-1}, \alpha_{d-1}) \mathcal{G}_d(\alpha_{d-1}, i_d, 1), \tag{13}$$

where $\|\mathcal{X}_{TT} - \mathcal{X}\|_F < \varepsilon$, for a prescribed value of ε. Here, $\|\cdot\|_F$ is the Frobenius norm. The entries of the integer array $\boldsymbol{r} = [r_1, \ldots, r_{d-1}]$ are called TT-ranks, and $\mathcal{G}_k$ are called TT-cores. Equivalently, we can also denote the TT-format by the multiple matrix product,

$$\mathcal{X}_{TT}(i_1, i_2, \ldots, i_d) = \boldsymbol{G}_1(i_1)\boldsymbol{G}_2(i_2)\ldots\boldsymbol{G}_d(i_d)\,, \tag{14}$$

where each term $\left(\boldsymbol{G}_k(i_k)\right)_{\alpha_{k-1},\alpha_k}$, $i_k = 1, 2, \ldots, n_k$, $k = 1, 2, \ldots, d$, is a matrix of size $r_{k-1} \times r_k$ (where $r_0 = r_d = 1$). Therefore, the $\mathcal{G}_k(:, i_k, :)$ are a set of matrix slices $\boldsymbol{G}_k(i_k)$ that are labeled with the single index i_k. Since each TT-core only depends on a single mode index of the full tensor $\mathcal{X}$, e.g., i_k, the TT-format effectively embodies a discrete separation of variables.

Assuming that $n_k = \mathcal{O}(n)$ and $r_k = \mathcal{O}(r)$ for some non-negative integers n and r, and for all $k = 1, 2, \ldots, d$, the total number of elements that TT-format stores is proportional to $\mathcal{O}(2nr + (d-2)nr^2)$, which is linear with the number of dimensions d. In this way, the TT-ranks $\boldsymbol{r} = [r_1, \ldots, r_{d-1}]$ quantify the effectiveness of the TT compression. When the TT-ranks are relatively small with respect to the problem size, a TT-based approach is referred to as a *low-rank approximation* (Bachmayr 2023).

In the case that $\mathcal{X}$ is a matrix, its TT-format is simplified to the following,

$$\mathcal{X}_{TT}(i_1, i_2) = \boldsymbol{G}_1(i_1)\boldsymbol{G}_2(i_2)\,, \tag{15}$$

Given that the shallow water problems we investigate in this work have two spatial dimensions, this decomposition will be the one we use to represent the solutions at each time step.

Note that both a tensor $\mathcal{X}$ and its TT representation $\mathcal{X}_{TT}$ share the same set of indices; if $\mathcal{X}$ represents data at a given set of grid points, $\mathcal{X}_{TT}$ also represents data at the same set of grid points. The difference is that the TT representation does not necessarily explicitly store data at each grid point, but rather returns these data via the tensor contraction operation given in Equation (13). In this way, both $\mathcal{X}$ and $\mathcal{X}_{TT}$ share the same shape, but do not explicitly store the same number of elements.

Tensor train is an advantageous low rank format for many reasons, but perhaps one of the most important is its compatibility with basic linear operations. For example, arithmetic operations such as addition, scalar multiplication, and pointwise multiplication can be performed on the TT cores without the need to form the complete tensor (Oseledets 2011a). Importantly, tensor trains are also compatible with linear transformations via linear operators. That is, given a tensor $\mathcal{X}$ and a linear operator L such that $L(\mathcal{X}) = \mathcal{Y}$, one can form a matrix product operator (MPO) in the TT format of L, call this A_L, such that $A_L \mathcal{X}_{TT} = \mathcal{Y}_{TT}$. Here, the operation $A_L \mathcal{X}_{TT}$ is a generalization of the matrix product that is computed entirely in the TT format (Oseledets 2011a).

However, when these basic operations are performed in the TT format, the rank of the resulting TT will grow relative to that of the input TTs. To reap the benefit of low rank compression for iterative schemes, a method known as TT rounding has been developed. We discuss this next.

b. *TT Rounding*

Given a tensor, $\mathcal{X}$, represented in TT-format, $\mathcal{X}_{TT}$, with TT-ranks $\boldsymbol{r} = [r_1, \ldots, r_{d-1}]$, one often wishes to find an even more compact TT representation $\mathcal{Y}_{TT}$, with TT-ranks $\boldsymbol{r}' = [r'_1, \ldots, r'_{d-1}]$ such that $r'_i \leq r_i$ for $i = 1, \ldots, d-1$. This is because, as discussed above, as certain common operations are applied to TTs, the ranks of resultant TTs can quickly grow, causing the benefits of the initial compression to disappear. To obtain a $\mathcal{Y}_{TT}$ such that $\|\mathcal{X}_{TT} - \mathcal{Y}_{TT}\|_F < \varepsilon_{TT}$ for a prescribed ε_{TT}, one can apply a so-called TT rounding procedure. This type of procedure is also called truncation or recompression, and is discussed in detail in (Oseledets 2011a). To illustrate the procedure in a simple case, consider the TT of a matrix like that in Equation (15). In this case, the TT rounding algorithm consists of two steps. First, the second core $\boldsymbol{G}_2$ is orthogonalized using RQ decomposition[1]. Thereafter, singular value decomposition (SVD) truncation at tolerance ε_{TT} is applied to the product $\boldsymbol{G}_1\boldsymbol{R}$ to arrive at a new decomposition $\mathcal{Y}_{TT}$. In this study, we denote TT rounding by $\mathcal{Y}_{TT} = \text{round}(\mathcal{X}_{TT}, \varepsilon_{TT})$. The computational cost of TT rounding is nontrivial, especially for TTs with large TT-ranks. As such, we choose carefully when to apply TT rounding, as described in Section 4d.

c. *TT Cross Interpolation*

It should be noted that not all operations can be directly computed in the TT format. Relevant examples discussed in Section 4 include $\sqrt{\mathcal{X}_{TT}}$, $|\mathcal{X}_{TT}|$, and $1/\mathcal{X}_{TT}$. However, nonlinear quantities such as these can be efficiently approximated via so-called TT cross interpolation. TT cross interpolation is a technique used to construct a TT representation of a tensor without needing to form the entire tensor explicitly. This method is particularly valuable when dealing with very large tensors or in situations where calculations are impractical due to limitations in TT arithmetic. Stemming

[1] RQ decomposition is analogous to QR decomposition, except that the rows of the input matrix are orthogonalized, rather than the columns.

from the skeleton (or CUR) decomposition (Mahoney and Drineas 2009), in combination with the Maximum Volume Principle (Goreinov et al. 2010), heuristic cross interpolation algorithms for tensor train, such as Alternating Minimal Energy (AMEn) (Dolgov and Savostyanov 2014), or Density Matrix Renormalization Group (DMRG) (Savostyanov and Oseledets 2011) have been developed. In this work, we use an implementation of AMEn algorithm, `amen_cross`, which is available in MATLAB TT-Toolbox (Oseledets 2014).

4. Tensorization of the Finite Volume Scheme

In this section, we will discuss the tensorization of the finite volume method for the shallow water equations.

a. *Tensor Train Finite Volume (TT-FV) Methods for Hyperbolic Conservation Laws*

We follow the methods discussed in (Danis et al. 2025). Start with the full-tensor form of the SWEs with "*loop indices*,"

$$\frac{\partial \widetilde{\overline{\boldsymbol{U}}}_{i,j}}{\partial t} + \frac{\widehat{\boldsymbol{F}}_{i+1/2,j} - \widehat{\boldsymbol{F}}_{i-1/2,j}}{\Delta x} + \frac{\widehat{\boldsymbol{G}}_{i,j+1/2} - \widehat{\boldsymbol{G}}_{i,j-1/2}}{\Delta y} = \widetilde{\overline{\boldsymbol{S}}}_{i,j}\,. \tag{16}$$

On a structured Cartesian mesh, we can introduce the shift operators in x and y, and rewrite the flux terms as

$$\begin{aligned} \widehat{\boldsymbol{F}}_{i+\frac{1}{2},j} &= T^x_{i,j}\widehat{\boldsymbol{F}}_{i-\frac{1}{2},j}\,, \\ \widehat{\boldsymbol{G}}_{i,j+\frac{1}{2}} &= T^y_{i,j}\widehat{\boldsymbol{G}}_{i,j-\frac{1}{2}}\,. \end{aligned} \tag{17}$$

Substituting Equation (17) into Equation (16) and dropping the *loop indices*, we obtain the vectorized form of the SWEs, which we will refer to as the "*full-tensor*" form:

$$\frac{\partial \widetilde{\overline{\boldsymbol{U}}}}{\partial t} + \frac{1}{\Delta x}\left(T^x - 1\right)\widehat{\boldsymbol{F}} + \frac{1}{\Delta y}\left(T^y - 1\right)\widehat{\boldsymbol{G}} = \widetilde{\overline{\boldsymbol{S}}}\,. \tag{18}$$

Note that terms in Equation (18) correspond to 2-dimensional pre-stored arrays. Generally speaking, the full-tensor approach is slower than using the loop indices unless explicitly parallelized or vectorized. However, the full-tensor format naturally lends itself to compression of the total degrees of freedom through the TT format. Specifically, we can simply replace the full-tensor terms with their TT counterparts:

$$\frac{\partial \widetilde{\overline{\boldsymbol{U}}}_{TT}}{\partial t} + \frac{1}{\Delta x}\left(T^x - 1\right)\widehat{\boldsymbol{F}}_{TT} + \frac{1}{\Delta y}\left(T^y - 1\right)\widehat{\boldsymbol{G}}_{TT} = \widetilde{\overline{\boldsymbol{S}}}_{TT}\,. \tag{19}$$

In Equation (19), all the terms are now in the TT format. This includes the conserved variables, the numerical fluxes, as well the shift operators. The arithmetic operations such as the MPO multiplication $\left(T^x - 1\right)\widehat{\boldsymbol{F}}_{TT}$ and addition are all carried out in the TT format as discussed in Section 3. To simplify notation, we continue to write the shift operators, now MPOs in the TT format, as T^x and T^y.

To perform the TT evolution of the SWEs, the tensor train terms of Equation (19) as well as the weights of the chosen finite volume method must be computed in an efficient manner. The numerical fluxes $\widehat{\boldsymbol{F}}_{TT}$ and $\widehat{\boldsymbol{G}}_{TT}$ as well as the (WENO) finite volume weights require careful consideration; these terms involve nonlinear operations on these TTs. We discuss this in the next subsections.

b. *Computing the Fluxes in the Tensor-Train Format*

To continue the formulation of our TT finite volume methods, we need to specify how the numerical fluxes are efficiently calculated in the TT format.

In this study, we implement the TT format of the Local Lax-Friedrichs flux similar to that suggested by (Danis and Alexandrov 2023). For example, the fluxes in the x-direction will be computed in the TT format as

$$\widehat{\boldsymbol{F}}_{TT}\left(\boldsymbol{U}_{TT}^{-}, \boldsymbol{U}_{TT}^{+}\right) = \frac{1}{2}\left(\boldsymbol{F}(\boldsymbol{U}_{TT}^{-}) + \boldsymbol{F}(\boldsymbol{U}_{TT}^{+})\right) - \frac{\lambda_{F,TT}}{2}\left(\boldsymbol{U}_{TT}^{+} - \boldsymbol{U}_{TT}^{-}\right)\,. \tag{20}$$

However, the linear and nonlinear SWEs differ in the computation of each individual term in Equation (20). For the linear SWEs the physical flux terms, $\boldsymbol{F}(\boldsymbol{U}_{TT}^{\pm})$ can be directly computed, without relying on special considerations such as TT cross interpolation. Additionally, the eigenvalue $\lambda_{F,TT} = \sqrt{gH}$ is a constant for the linear equations. In contrast, for the nonlinear SWE equations $1/h_{TT}$ must be computed for the physical fluxes $\boldsymbol{F}$ and $\boldsymbol{G}$ (in order to recover u from the conserved quantity hu), and $\lambda_{F,TT} = |u_{TT}| + \sqrt{g h_{TT}}$ and $\lambda_{G,TT} = |v_{TT}| + \sqrt{g h_{TT}}$ must be computed as the eigenvalues of the flux Jacobians of $\boldsymbol{F}$ and $\boldsymbol{G}$ respectively. These nonlinear functions of the TTs cannot be computed directly, therefore we employ the AMEn method to compute eigenvalues $\lambda_{F,TT}$ and $\lambda_{G,TT}$ and the Taylor series approximation given in Algorithm 1, to compute the reciprocal of the tensor train h_{TT}.

Algorithm 1: Taylor Series Approximation to Tensor Train Reciprocal

Data: x_{TT}, $\varepsilon_{TT} > 0$
Result: $y_{TT} = 1/x_{TT}$

$y_{TT} \leftarrow 1$;
$\Delta y_{TT} \leftarrow 1$;
$Err \leftarrow 1$;
$N \leftarrow \mathrm{numel}(x_{TT})$;
$x_{avg} \leftarrow \mathrm{sum}(x_{TT})/N$;
$\tilde{x}_{TT} \leftarrow \mathrm{round}(1 - x_{TT}/x_{avg}, \varepsilon_{TT})$;
while $Err > \varepsilon_{TT}$ **do**
  $\Delta y_{TT} \leftarrow \mathrm{round}(\Delta y_{TT} * \tilde{x}_{TT}, \varepsilon_{TT})$;
  $y_{TT} \leftarrow \mathrm{round}(y_{TT} + \Delta y_{TT}, \varepsilon_{TT})$;
  $Err \leftarrow \|\Delta y_{TT}\|_F / \sqrt{N}$;
$y_{TT} \leftarrow y_{TT}/x_{avg}$;

After obtaining $1/h_{TT}$, the components of the flux vectors in Equation (3) can be directly computed. Note that we could have also employed a TT cross interpolation method to compute $1/h_{TT}$ instead of Algorithm 1. In the numerical examples considered in this study, however, we found that Taylor series approximation to $1/h_{TT}$ is as fast as the AMEn method. This is possibly because the shallow water layer thickness h can be decomposed as a superposition of a large mean value and small amplitude oscillations, i.e. $h = H + \eta(x,y,t)$ where $|\eta| \ll H$, which leads to a very fast and robust convergence of the Taylor series approximation for $1/h_{TT}$.

Note also that this method is different than the LF-cross method developed in (Danis et al. 2025), where the complete flux vector of the compressible Euler equations is computed with a single cross interpolation using the AMEn method. In the context of the finite difference method for solving compressible flow equations, the LF-cross method was reported to be faster than a similar method presented here that approximates $1/\rho_{TT}$ (reciprocal of density) with the TT cross interpolation to compute the flux vector components of compressible Euler equations directly. This was thought to be due to slow TT-rounding while each flux vector component was estimated. However, for the finite volume method for solving the shallow water equations, we found that the LF-cross approach is considerably slower than the present approach. This might be due to two reasons. First, the compressible Euler equations have more conserved variables than the SWEs, and therefore, more floating point operations are needed to compute fluxes in compressible flow, meaning that the rounding routine will be called more often and each call will more expensive due to the additional cost of quadrature rules used in high-order finite volume schemes. Second, computing $1/h_{TT}$ for the SWEs is likely a much simpler task than computing the $1/\rho_{TT}$ for the compressible flow. As mentioned above, in the case of the SWEs, h is simply a superposition of a mean depth and small-amplitude waves, but in the compressible flow, the density field ρ can have large variations, and even, discontinuities such as strong shock waves. Therefore, the nonlinear flux calculations should be designed or selected according to the particular application, especially when the computation of the reciprocal of a tensor train is required. Looking forward to extending these methods to layered Boussinesq models appropriate for ocean and atmosphere circulation at the climate scale, we expect that the method of computing $1/h_{TT}$ presented here in Algorithm 1 will extend naturally; the barotropic mode of a layered Boussinesq model is exactly the SWEs. The baroclinic mode is given by a vertical stack of shallow water models that communicate via vertical fluxes; it is likely that these will be well treated by the approach from Algorithm 1 as well.

c. *High-order Reconstructions in the Tensor-Train Format*

Finally, to complete our TT finite volume methods, we specify how the high-order reconstructions are handled in the TT format. Here, we consider two types of high-order variable reconstruction methods – linear (Upwind3 and Upwind5) and nonlinear (WENO5) reconstructions as discussed in Section 2c and Appendix A1. For linear reconstruction schemes, we are able to directly manipulate the TT cores, which is more computationally efficient than manipulating the entire TT. However, for nonlinear WENO reconstruction, this is not possible because of the presence of inverse quadratic terms in in the WENO weights. Therefore, we employ TT cross interpolation via the AMEn method for the WENO reconstruction.

1 Linear Reconstruction Methods

The linear reconstruction in the TT format in a given direction is applied only to the TT core corresponding to that direction, which significantly reduces the computational costs. To exploit this, we modify the algorithm presented in Appendix A1. At a high level, the algorithm consists of two steps: Step 1 is "De-cell" averaging over x followed by Step 2, "De-cell" averaging over y.

Note that, in the full-tensor format, Step 1 is followed by Step 2 for each time a reconstruction is needed along a cell interface. This means that, in a 2-dimensional setting, Step 2 is applied twice to close the cell boundaries. However, this is not needed in the TT format.

To illustrate the idea, we adopt the notation used in (Oseledets 2011b) and denote the elementwise values of a cell-averaged tensor $\widetilde{\overline{u}}$ as

$$\widetilde{\overline{u}}(i,j) = \overline{u}_1(i)\widetilde{u}_2(j)\,, \tag{21}$$

Note that the first core $\overline{u}_1$ is written with a bar to denote the cell-averaging operator in x and the second core $\widetilde{u}_2$ is written with a tilde to denote the cell-averaging operator in y. This implies that Step 2 in Appendix A1 can be applied to each core independently and even simultaneously.

TT-Reconstruction Step 1 starts with reconstructing the cores at the quadrature points $u_1(x_i + \delta_m\Delta x)$ and $u_2(y_j + \delta_m\Delta y)$. This step is almost identical to Step 2 in Appendix A1 except for that fact that the reconstruction is only applied to TT cores rather than to the full-tensor.

TT-Reconstruction Step 2 is similar to Step 1 in Appendix A1, and again, we only apply reconstruction to the TT-cores. For example, a reconstruction in the x-direction first compute $u_1(i \pm 1/2)^{\mp}$, and then, the result is combined with $u_2(y_j + \delta_m\Delta y)$ prepared in TT-Reconstruction Step 1:

$$u(i \pm 1/2, y_j + \delta_m\Delta y)^{\mp} = u_1(i \pm 1/2)^{\mp} u_2(y_j + \delta_m\Delta y)\,. \tag{22}$$

Similarly, for the reconstruction in the y-direction, TT-Reconstruction Step 2 computes $u_2(j \pm 1/2)^{\mp}$ and combines this with $u_1(x_i + \delta_m\Delta x)$ prepared in TT-Reconstruction Step 1:

$$u(x_i + \delta_m\Delta x, j \pm 1/2)^{\mp} = u_1(x_i + \delta_m\Delta x) u_2(j \pm 1/2)^{\mp}\,. \tag{23}$$

2 Nonlinear Reconstruction Method

The WENO scheme performs the nonlinear reconstruction using the TT cross interpolation by applying Step 1 and Step 2 of Appendix A1 in the same order. In fact, TT-WENO Step 1 is similar to the finite difference WENO-cross method proposed in (Danis et al. 2025), i.e. Step 1 is applied in an equation-by-equation fashion using the TT cross interpolation as detailed in Algorithm 2. Here, *funWENO* is a function that takes as input the x-direction 5-stencil $\boldsymbol{S}$ =

$\{(T^x)^{-2}\widetilde{\widetilde{v}},\ (T^x)^{-1}\widetilde{\widetilde{v}},\ \widetilde{\widetilde{v}},\ T^x\widetilde{\widetilde{v}},\ (T^x)^2\widetilde{\widetilde{v}}\}$ of a conserved, cell-averaged quantity $\widetilde{\widetilde{v}}$, and returns the cell-averaged values $\widetilde{v}^{\pm}$ in the y-direction. In plain language, *funWENO* applies Step 1 of Appendix A1 for WENO5 to the full-tensor problem. The AMEn method is then able to approximate the application of this function to the TT quantity $\widetilde{\widetilde{v}}_{TT}$ without ever forming the full tensor.

Algorithm 2: TT-WENO Step 1 for the "de-cell" averaging over x

---

Data: A component of the cell-averaged conserved variable vector $\widetilde{\widetilde{v}}_{TT}$, function *funWENO*, and the convergence criterion ε_{TT}.
Result: Cell-averaged $\widetilde{v}_{TT}^{\pm}$ in y-direction at the interface locations.

1. Collect the 5-cell stencil of $\widetilde{\widetilde{v}}_{TT}$ in the x-direction into the array $\boldsymbol{S} = \{(T^x)^{-2}\widetilde{\widetilde{v}}_{TT},\ (T^x)^{-1}\widetilde{\widetilde{v}}_{TT},\ \widetilde{\widetilde{v}}_{TT},\ T^x\widetilde{\widetilde{v}}_{TT},\ (T^x)^2\widetilde{\widetilde{v}}_{TT}\}$.
2. Set the initial guess $v_0 = \widetilde{\widetilde{v}}_{TT}$.
3. Perform cross interpolation for + side: $\widetilde{v}_{TT}^{+}$ =AMEn(*funWENO*, $\boldsymbol{S}$, v_0, ε_{TT},+1)
4. Perform cross interpolation for − side: $\widetilde{v}_{TT}^{-}$ =AMEn(*funWENO*, $\boldsymbol{S}$, v_0, ε_{TT},−1)

Similarly, TT-WENO Step 2 applies Step 2 of Appendix A1 for each conserved variable using the TT cross interpolation, see Algorithm 3. However, it performs the WENO reconstruction at all quadrature points with a single cross interpolation. Similar to the above, *funWENOquad* applies Step 2 of Appendix A1 for WENO5 to $\widetilde{v}^{\pm}$.

Note that the *AMEn-cross* routine in the MATLAB Toolbox returns a TT of shape $1\times N_x\times N_y\times N_q$, where N_x is the number of grid points in the x-direction, N_y is the number of grid points in the y-directions, and N_q is the number of quadrature points. Therefore, as a last step, it is reshaped and permuted to obtain a new TT of shape $1\times N_x\times N_yN_q\times 1$ for the reconstruction in the x-direction and $1\times N_xN_q\times N_y\times 1$ for the reconstruction in the y-direction.

Algorithm 3: TT-WENO Step 2 for for the "de-cell" averaging in y

---

Data: Cell-averaged $\widetilde{v}_{TT}^{\pm}$ in y-direction obtained from TT-WENO Step 1, function *funWENOquad*, and the convergence criterion ε_{TT}.
Result: $v_{TT}^{\pm}$ at the interface locations.

1. Collect the 5-cell stencil of v_{TT} in the x-direction into the arrays $\boldsymbol{S}^{\pm} = \{(T^y)^{-2}\widetilde{v}_{TT}^{\pm},\ (T^y)^{-1}\widetilde{v}_{TT}^{\pm},\ \widetilde{v}_{TT}^{\pm},\ T_y\widetilde{v}_{TT}^{\pm},\ (T^y)^2\widetilde{v}_{TT}^{\pm}\}$.
2. Set the initial guesses as $v_0^{\pm} = \widetilde{v}_{TT}^{\pm}$.
3. Perform cross interpolation for + side: v_{TT}^{+} =AMEn(*funWENOquad*, $\boldsymbol{S}^{+}$, v_0^{+}, ε_{TT},+1)
4. Perform cross interpolation for − side: v_{TT}^{-} =AMEn(*funWENOquad*, $\boldsymbol{S}^{-}$, v_0^{-}, ε_{TT},−1)
5. Reshape and permute $v_{TT}^{\pm}$ to return two TTs of size $1\times N_x\times N_yN_q\times 1$.

---

d. *Choosing a Suitable ε_{TT}*

The choice of ε_{TT} determines the overall accuracy and the speed of the TT solver, as demonstrated in (Danis et al. 2025). In this study, we modify the ε_{TT} formula slightly to accommodate both the 3rd and 5th order schemes as

$$\varepsilon_{TT} = C_\varepsilon \frac{V^{1/2} \Delta x^{p-1/2}}{\max_{q \in \boldsymbol{U}_{TT}} \|q\|_F}, \tag{24}$$

where $\|\cdot\|_F$ is the Frobenius norm, p denotes the order of accuracy of the TT-FV scheme, V is the volume of the computational domain, and C_ε is a problem dependent variable (see (Danis et al. 2025)).

Ideally, the choice of C_ε should be made to ensure that the TT truncation errors are less than or equal to the discretization errors of the underlying numerical scheme such that the formal convergence rate is maintained in the TT format. However, this requires an apriori knowledge of the discretization error, which is generally not available. If a very large C_ε is set, then the numerical TT solution will attain a relatively low-rank structure but the formal accuracy of the underlying scheme will be lost. On the other hand, a very small C_ε will maintain the accuracy, but the TT ranks will be unnecessarily large. In the present study, thanks to the nondimensionalization of the SWEs, $C_\varepsilon = 1$ is found to be sufficient for obtaining high-order results without increasing the TT ranks in the majority of the numerical examples. Despite this success, however, choosing a suitable value for C_ε still remains as an empirical process. To determine C_ε, we recommend a procedure analogous to a grid independence study – a common practice applied in Computational Fluid Dynamics (CFD) for identifying the finest grid resolution at which further refinement does not significantly alter the numerical solution. To ensure C_ε-independence, one could start with a relatively large value of C_ε and incrementally decrease it until the numerical solution becomes independent of C_ε.

Note that if one of the conserved variables is exactly zero, choosing ε_{TT} according to Equation (24) may still result in rank growth. This is primarily because ε_{TT} becomes comparable to the numerical noise generated for that conserved variable, which is known to result in rank growth. A robust workaround is obtained by replacing the max operator in the denominator of Equation (24) with the norm of the relevant conserved variable itself. For example, consider the conserved variable

$u_{TT} \in \boldsymbol{U}_{TT}$ of the linear SWEs Equation (2), for which we calculate

$$\varepsilon^u_{TT} = \min\left(10^{-3}, C_\varepsilon \frac{V^{1/2}\Delta x^{p-1/2}}{\|u\|_F}\right), \tag{25}$$

at the beginning of a time step. Note that the new dynamic formula is clipped at 10^{-3} to avoid accuracy loss if $\|u\|_F \to 0$, where the upper bound 10^{-3} is empirically deemed sufficient to limit the TT truncation error such that the relative error $\|u - u^*\|_F / \|u\|_F \leq 10^{-3}$ is achieved after performing $u = \mathrm{round}(u^*, \varepsilon_{TT})$.

Equipped with Equation (25), the 3rd order strong stability-preserving Runge-Kutta method (Shu and Osher 1988; Gottlieb et al. 2001) for a given semi-discrete form of u_{TT},

$$\frac{du_{TT}}{dt} = L(u_{TT}), \tag{26}$$

is then defined in TT format as

$$\begin{aligned} u^{(1)}_{TT} &= \mathcal{F}(u^n_{TT}), \\ u^{(2)}_{TT} &= \mathrm{round}\left(\frac{3}{4}u^n_{TT} + \frac{1}{4}\mathcal{F}(u^{(1)}_{TT}),\, \varepsilon^u_{TT}\right), \\ u^{n+1}_{TT} &= \mathrm{round}\left(\frac{1}{3}u^n_{TT} + \frac{2}{3}\mathcal{F}(u^{(2)}_{TT}),\, \varepsilon^u_{TT}\right), \end{aligned} \tag{27}$$

where $\mathcal{F}(u)$ is the forward Euler step

$$\mathcal{F}(u) = \mathrm{round}\left(u + \Delta t L(u),\, \varepsilon^u_{TT}\right). \tag{28}$$

Note that the same procedure is applied for the time integration of other conserved variables. For all other rounding operations, we use the standard formula given in Equation (24).

5. Numerical Results

In this section, we demonstrate the performance of the TT finite volume solver in a series of five numerical test cases. The first four of these cases are taken from the validation suite introduced in (Bishnu et al. 2024); the Coastal Kelvin Wave, Inertia-Gravity Wave, Barotropic Tide, and Manufactured Solution cases are all solved on a square computational domain $\Omega = [0, L] \times [0, L]$.

For these cases, unless otherwise stated, we set $g = 10$ m/s^2, $f = 10^{-4}$ 1/s, $H = 1000$ m, $c = \sqrt{gH} = 100$ m/s, and $R = c/f = 10^6$ m as in (Bishnu et al. 2024). For the fifth and final test case, we present a flow characterized by a stable barotropic jet, similar to that from (Galewsky et al. 2004), adapted to the plane; we describe the relevant details in Section 5e.

In all cases, the nondimensional forms of Equations (1) to (3) are solved (see Appendix A1) using the 3$^{\text{rd}}$ order strong stability-preserving Runge-Kutta method (Shu and Osher 1988; Gottlieb et al. 2001). For the 5$^{\text{th}}$ order reconstruction schemes, we set the time step Δt proportional to $\Delta x^{5/3}$ to maintain accuracy, and the proportionality constant is chosen such that the numerical solution remains stable for all time steps. Note that all time step sizes will be reported in terms of the nondimensional variables. In all TT simulations, Equations (24) and (25) are calculated using only the nondimensional variables and the L_2-error is defined relative to the L_2-error of the same reconstruction method at the coarsest grid level. Both tensor-train and full-tensor simulations are performed on Apple M1 Max chip using MATLAB by ensuring a single-thread execution. To optimize the efficiency in MATLAB and avoid performance losses due to for-loops, the full-tensor solver is implemented as suggested in (Danis et al. 2025) to make use of MATLAB's vectorization. Finally, all surface integrals are computed using Gauss-Legendre quadrature rule in the TT format, as suggested in (Alexandrov et al. 2023). See Table 1 for a summary of the numerical examples along with nondimensional time step to grid size ratios and C_ε in Equation (25).

Test Case	**Upwind3**		**Upwind5**		**WENO5**	
	$\Delta t/\Delta x$	C_ε	$\Delta t/\Delta x^{5/3}$	C_ε	$\Delta t/\Delta x^{5/3}$	C_ε
Coastal Kelvin Wave	5×10^{-5}	1	2.5×10^{-4}	1	2.5×10^{-4}	1000
Inertia-Gravity Wave	10^{-4}	1	10^{-3}	1	10^{-3}	500
Barotropic Tide	2.5×10^{-4}	1	5×10^{-4}	1	5×10^{-4}	1
Manufactured Solution	5×10^{-5}	10^{-4}	5×10^{-3}	1	5×10^{-3}	1
Stable Barotropic Jet	5×10^{-3}	1	5×10^{-1}	1	5×10^{-1}	1

TABLE 1: Summary of test cases.

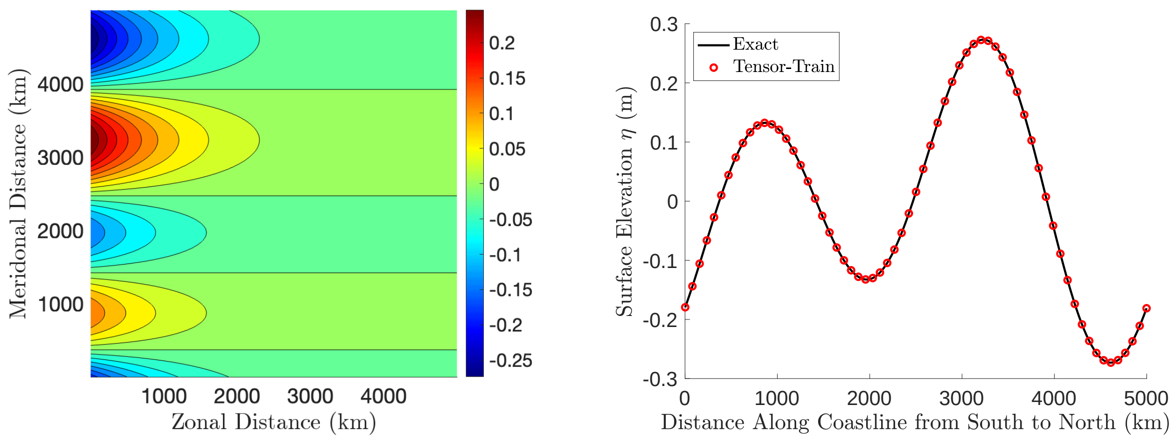
Finally, we should be careful to consider and discuss the differences between the results presented here for the SWEs and the results from (Danis et al. 2025) for the compressible Euler equations. First, the SWE configurations considered here are chosen specifically to show the performance of our TT-FV methods as applied to problems relevant to large scale geophysical flows, whereas the experiments presented in (Danis et al. 2025) show that similar TT-FV methods can accurately resolve shocks and flow structures arising from turbulent motions, which are not present in hydrostatic models of the ocean and atmosphere. Additionally, as previously discussed in Section 4b, the method used to compute the flux in the present case was chosen as it is more computationally efficient for the SWE configurations considered here. Finally, one can note that, in the best cases, the speedups reported in (Danis et al. 2025) are an order of magnitude larger than the speedups reported here. In part, this could be because of how effectively the model state is able to be compressed by TT across different types of flows, but it is very likely that the biggest contribution to this difference in performance is the larger number of unknowns in the compressible Euler problems. The compressible Euler equations have five unknown quantities in a 3-dimensional domain, while the SWEs have only three unknowns in a 2-dimensional domain. It is well known that the effectiveness of TT compression increases with problem size, so it is natural that the larger problem sizes from (Danis et al. 2025) would lead to more speedup. This is good news for the present case, because as future work extends these TT-FV methods to more realistic configurations in layered, 3-dimensional models, we expect the speedups presented here not just to generalize, but to improve.

a. *Coastal Kelvin Wave*

In this example, we solve the linear SWEs, Equations (1) and (2), up to the final time $T = 3$ hours and set $L = 5 \times 10^6$ m. Coastal Kelvin Waves occur when the Coriolis force is balanced against a topographic boundary, and are found in real-world observations (Johnson and O'Brien 1990). In this idealized configuration, the coastline is in the non-periodic x direction and nondispersive waves travel along the boundary in the periodic y direction. The exact solution is plotted in Figure 3 and the initial conditions are composed of two wave modes:

$$\begin{aligned}
\eta &= -H\left\{\hat{\eta}^{(1)}\sin\left(k_y^{(1)}(y+ct)\right)+\hat{\eta}^{(2)}\sin\left(k_y^{(1)}(y+ct)\right)\right\}\exp\left(-x/R\right),\\
u &= 0, \\
v &= \sqrt{gH}\left\{\hat{\eta}^{(1)}\sin\left(k_y^{(1)}(y+ct)\right)+\hat{\eta}^{(2)}\sin\left(k_y^{(1)}(y+ct)\right)\right\}\exp\left(-x/R\right),
\end{aligned} \tag{29}$$

where $\hat{\eta}^{(1)} = \hat{\eta}^{(2)}/2 = 10^{-4}$ m, and $k_y^{(1)} = k_y^{(2)}/2 = 2\pi/L$.



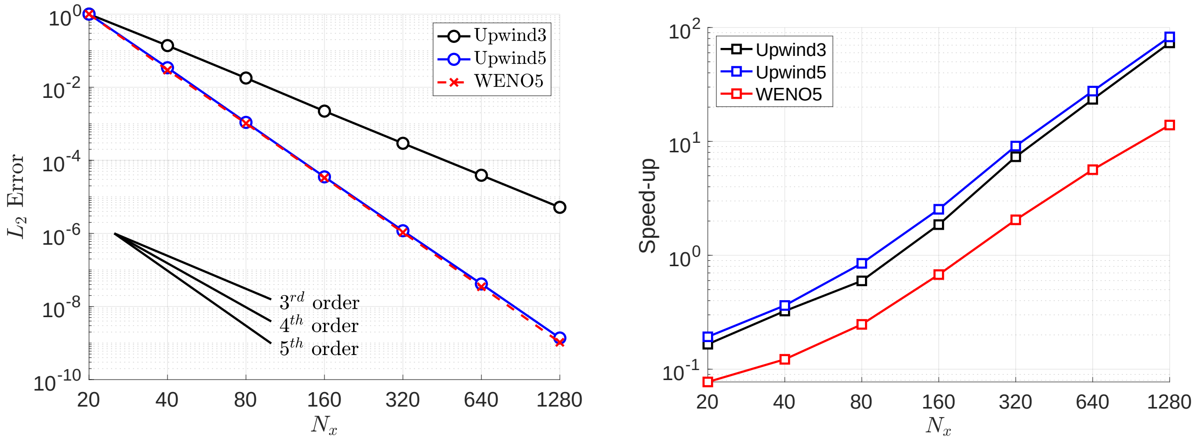
(a) Contour plot of η produced by the TT method

(b) η profile

FIG. 3: Exact and TT solutions of surface elevation for Coastal Kelvin Wave at $T = 3$ hours. The TT solution is obtained with Upwind5 on 1280×1280 grid.

We use the exact solution to set the boundary condition in the $x-$direction, but in the $y-$direction, we consider a periodic boundary. Furthermore, for both upwind methods we use $C_\varepsilon = 1$ in Equations (24) and (25) to calculate ε_{TT}, but for WENO5 we use $C_\varepsilon = 1000$. To guarantee accuracy and stability of the numerical solution, we also consider a time step of $\Delta t/\Delta x = 5 \times 10^{-5}$ for the Upwind3 method and $\Delta t/\Delta x^{5/3} = 2.5 \times 10^{-4}$ for the 5$^{\text{th}}$ order methods.

Figure 4 shows the performance of Upwind3, Upwind5 and WENO5 methods. On the left, all TT reconstruction methods are observed to maintain their formal L_2-convergence order for η. On the right, the various methods are compared in terms of the acceleration with respect to their corresponding standard full-tensor versions: At the finest grid level, TT-Upwind5 achieves 83x acceleration while Upwind3 has a speed-up of 73x. Note that Upwind3 is still less expensive than Upwind5. Therefore, the higher speed-up value of Upwind5 simply means that it becomes more efficient in accelerating its full-tensor version for this linear problem. On the other hand, the TT-WENO5 method only achieves the significantly lower speed-up of 14x. This clearly shows the cost of the TT cross interpolation used in the WENO5 scheme, even when the problem is linear and smooth.



(a) L_2 error for η

(b) Speed-up

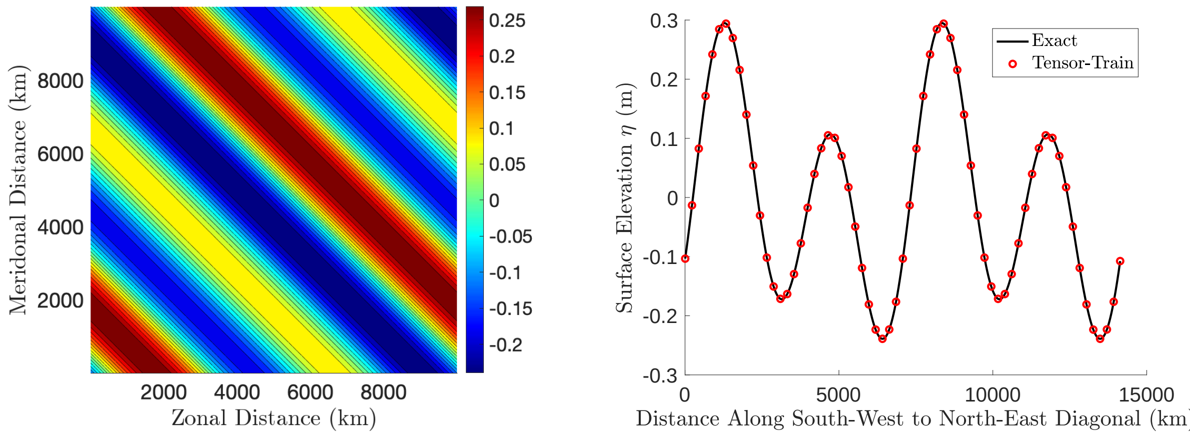
FIG. 4: L_2 error of surface elevation and speed-up for Coastal Kelvin Wave at $T = 3$ hours.

b. *Inertia-Gravity Wave*

Inertia-gravity waves are an important component of geophysical turbulence (Young 2021). They occur in the open ocean, and are gravity waves where particles oscillate in an ellipse due to the rotation of the Earth. The idealized test problem is linear with a flat bottom, like the Coastal Kelvin Wave, but the domain is periodic in both directions. Specifically, we solve the linear SWEs, Equations (1) and (2), up to the final time $T = 3$ hours with $L = 10^7$ m. We consider the general solution,

$$\begin{aligned}
\eta &= \hat{\eta}\cos\left(k_x x + k_y y - \hat{\omega} t\right), \\
u &= \frac{g\hat{\eta}}{\hat{\omega}^2 - f^2}\left\{\hat{\omega} k_x \cos\left(k_x x + k_y y - \hat{\omega} t\right) - f k_y \sin\left(k_x x + k_y y - \hat{\omega} t\right)\right\}, \\
v &= \frac{g\hat{\eta}}{\hat{\omega}^2 - f^2}\left\{\hat{\omega} k_y \cos\left(k_x x + k_y y - \hat{\omega} t\right) + f k_x \sin\left(k_x x + k_y y - \hat{\omega} t\right)\right\},
\end{aligned} \tag{30}$$

where $\hat{\omega} = \sqrt{c^2\left(k_x^2 + k_y^2\right) + f^2}$. The particular solution plotted in Figure 5 is constructed to be the superposition of two wave modes for $\eta^{(1)} = \eta^{(2)}/2 = 10^{-1}$ m, $k_x^{(1)} = k_x^{(2)}/2 = 2\pi/L$, and $k_y^{(1)} = k_y^{(2)}/2 = 2\pi/L$ and initial conditions are set from these at $t = 0$.



(a) Contour plot of η produced by the TT method

(b) η profile

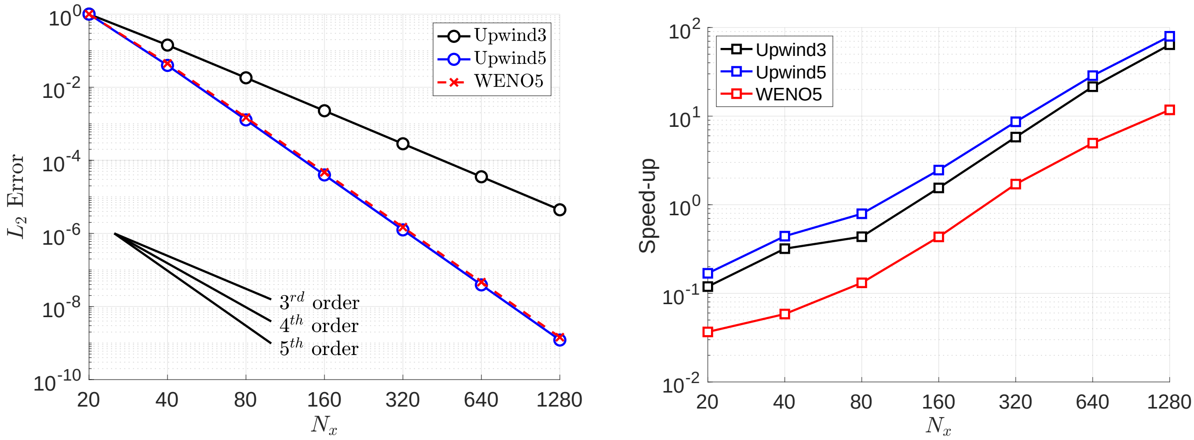
FIG. 5: Exact and TT solutions of surface elevation for Inertia-Gravity Wave at $T = 3$ hours. The TT solution is obtained with Upwind5 on 1280×1280 grid.

As in the previous example, Equations (24) and (25) uses $C_\varepsilon = 1$ for both upwind methods, but it sets $C_\varepsilon = 500$ for the WENO5 scheme in this example. Time step sizes are also set as $\Delta t/\Delta x = 10^{-4}$ for the Upwind3 method and $\Delta t/\Delta x^{5/3} = 10^{-3}$ for the 5^{th} order methods.

In Figure 6, the L_2−convergence of η and speed-up of the TT solver are shown. Our TT solver recovers the formal order of accuracy as in the previous example. Similar to the previous test case, TT-Upwind5 results in a slightly better speed-up than TT-Upwind3, and TT-WENO5 gives considerably less acceleration. At the finest grid level, the speed-up achieved by the TT solver are 79x for Upwind5, 64x for Upwind3, and 12x for WENO5.

c. *Barotropic Tide*

The Barotropic Tide case is an idealized representation of a coastal tide on a continental shelf (Clarke and Battisti 1981). Like the previous cases it tests the linear SWEs, Equations (1) and (2), but now on a doubly non-periodic domain. We solve to a final time of $T = 30$ minutes, where



(a) L_2 error for η

(b) Speed-up

FIG. 6: L_2 error of surface elevation and speed-up for Inertia-Gravity Wave at $T = 3$ hours.

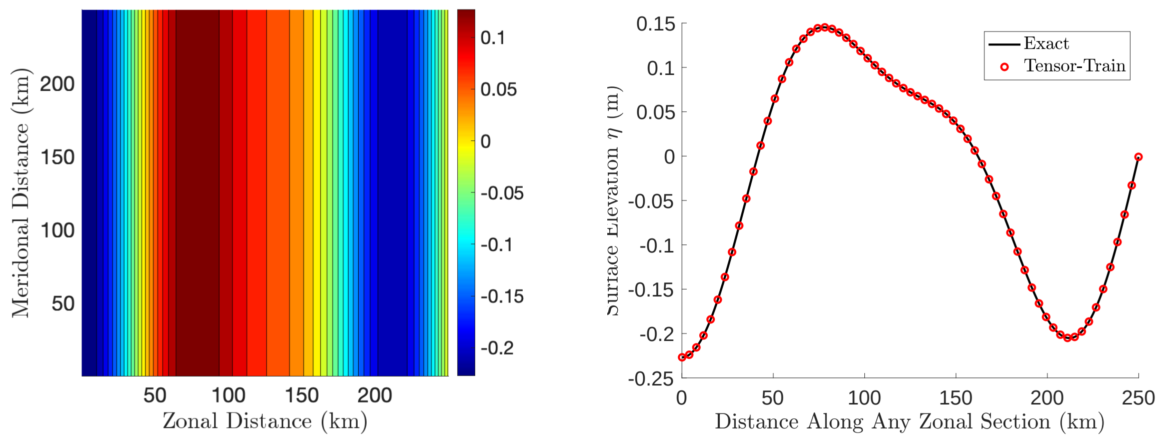
$L = 25 \times 10^4$ m. The general solution considered here is given as,

$$
\begin{aligned}
\eta &= \hat{\eta} \cos(kx) \cos(\omega t), \\
u &= \frac{g\hat{\eta}\omega k}{\hat{\omega}^2 - f^2} \sin(kx) \sin(\omega t), \\
v &= \frac{g\hat{\eta} f k}{\hat{\omega}^2 - f^2} \sin(kx) \cos(\omega t),
\end{aligned}
\tag{31}
$$

where $\omega = \sqrt{gHk^2 + f^2}$. We again consider a particular solution shown in Figure 7 constructed as a superposition of two wave modes, but now we set $\eta^{(1)} = \eta^{(2)}/2 = 0.2$ m, $k^{(1)} = 2\pi/\lambda^{(1)}$ and $k^{(2)} = 2\pi/\lambda^{(2)}$, where $\lambda^{(1)} = 4L/5$ and $\lambda^{(2)} = 4L/9$. Note that $H = 200\ m$ in this example. Physically, the wavelengths are chosen to satisfy the conditions for tidal resonance in this domain (Bishnu et al. 2024). The initial conditions at $t = 0$ are set from the exact solution while the boundary condition in the x-direction is taken from the exact solution and is assumed periodic in the y-direction.

In this test case, we set $C_\varepsilon = 1$ for all reconstruction schemes, and consider $\Delta t/\Delta x = 2.5 \times 10^{-4}$ for Upwind3 and $\Delta t/\Delta x^{5/3} = 5 \times 10^{-4}$ for the 5^{th} order methods.

The results reported in Figure 8 follow the trends already observed in the previous linear test cases: Our TT solver achieves the formal order of accuracy, the highest speed-up is obtained for

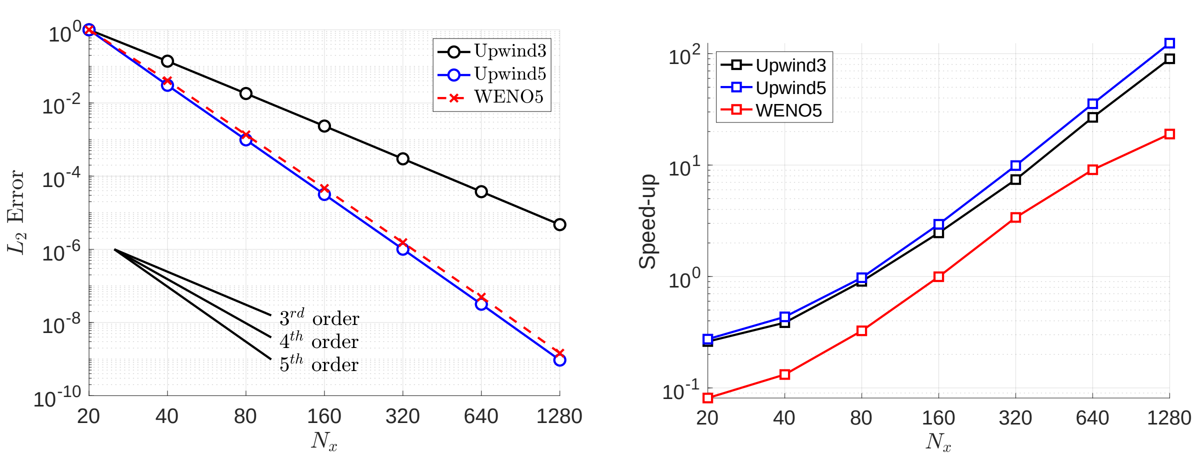


(a) Contour plot of η produced by the TT method

(b) η profile

FIG. 7: Exact and TT solutions of surface elevation for Barotropic Tide at $T = 30$ minutes. The TT solution is obtained with Upwind5 on 1280×1280 grid.

Upwind5 and the WENO5 gives the lowest. At the finest grid level, the speed-up achieved by the TT solver are 124x for Upwind5, 89x for Upwind3, and 19x for WENO5.



(a) L_2 error for η

(b) Speed-up

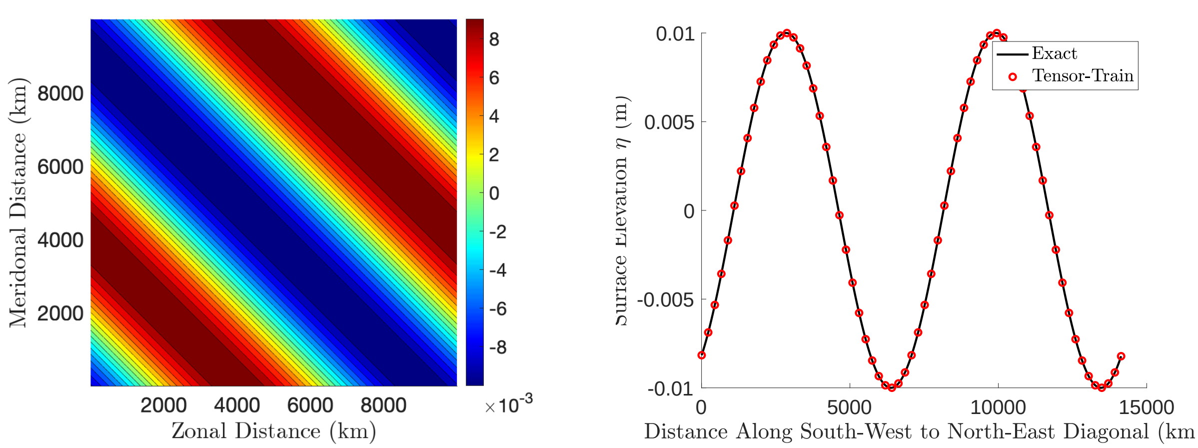
FIG. 8: L_2 error of surface elevation and speed-up for Barotropic Tide at $T = 30$ minutes.

d. *Manufactured Solution*

In this test case, we turn our attention to the nonlinear SWEs and solve Equations (1) and (3) up to the final time $T = 3$ hours, where we set $L = 10^7$ m. The manufactured solution does not have an analog observed in nature, but is important because an analytic solution may be derived that tests all terms in the SWEs. Subsequent SWE test cases for geophysical turbulence with more complex behavior do not have analytic solutions (Williamson et al. 1992). Using the method of manufactured solutions (Roache 2019), we enforce

$$\begin{aligned}
\eta &= \hat{\eta} \sin\left(k_x x + k_y y - \hat{\omega} t\right), \\
u &= \hat{u} \cos\left(k_x x + k_y y - \hat{\omega} t\right), \\
v &= 0,
\end{aligned} \tag{32}$$

where $\hat{\eta} = 10^{-2}$ m , $\hat{u} = 10^{-2}$ m/s, $k_x = k_y = 2\pi/L$, and $\hat{\omega} = \sqrt{c^2\left(k_x^2 + k_y^2\right)}$, as depicted in Figure 9. We consider periodic boundaries and impose the initial condition from the manufactured solution at $t = 0$. In this nonlinear test case, we set $C_\varepsilon = 10^{-4}$ for the Upwind3 scheme to maintain its 3rd



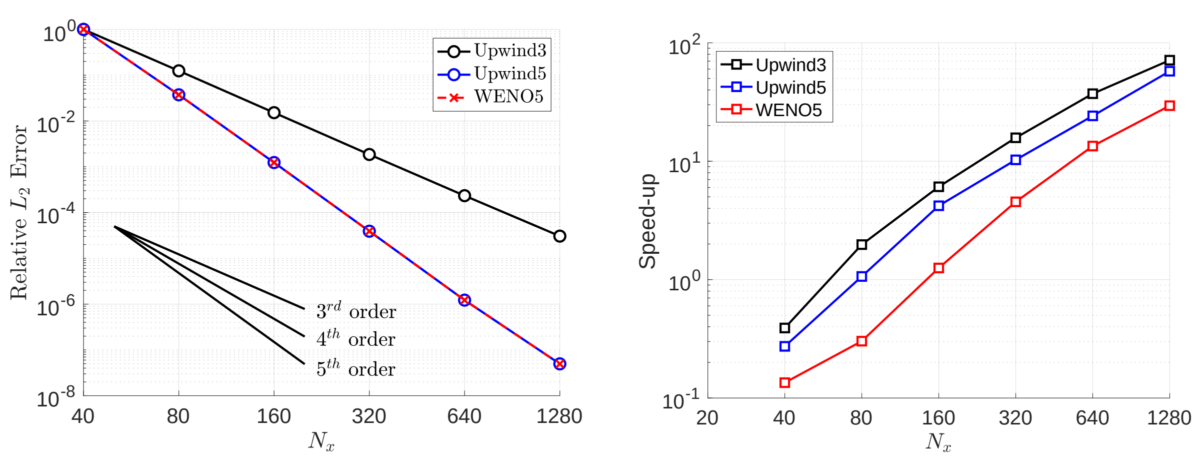
(a) Contour plot of η produced by the TT method

(b) η profile

FIG. 9: Exact and TT solutions of surface elevation for the Manufactured Solution at $T = 3$ hours. The TT solution is obtained with Upwind5 on 1280×1280 grid.

order accuracy. For the 5th order methods, however, $C_\varepsilon = 1$ is deemed to be sufficient. The time stepping is performed according to $\Delta t/\Delta x = 5\times 10^{-5}$ for Upwind3 and $\Delta t/\Delta x^{5/3} = 5\times 10^{-3}$ for the 5^{th} order methods.

In Figure 10, we observed the formal 3^{rd} order L_2 convergence of Upwind3 and 5^{th} order convergence of Upwind5 and WENO5, as in the linear cases. However, we see that Upwind3 gives a better acceleration than Upwind5 unlike the linear test cases. This may be due to the nonlinear nature of the fluxes in Equation (3). Recall that TT rounding operation is performed after almost each TT multiplication and addition operation to avoid the rank growth. Therefore, TT rounding is applied several times for the nonlinear fluxes while no rounding is needed for the linear fluxes. Since higher-order schemes require more quadrature points to compute fluxes, the additional cost of TT rounding is naturally more expensive for higher-order method for. In addition, the nonlinear fluxes need to compute $1/h_{TT}$ using Algorithm 1, which is also needed to be carried out on larger arrays for higher-order schemes. As a result, the different mechanics of flux computation in linear and nonlinear problems lead to different trends in terms of the speed-up. At the finest grid level, the speed-up achieved by the TT solver are 71x for Upwind3, 57x for Upwind5, and 29x for WENO5.



(a) L_2 error for h

(b) Speed-up

FIG. 10: L_2 errors of the shallow water layer thickness and speed-up for Manufactured Solution at $T = 3$ hours.

e. *Stable Barotropic Jet*

In this final test case, we simulate a flow similar to that from the stable barotropic jet test case from (Galewsky et al. 2004), adapted for the plane. We initialize the flow to a steady state characterized by a strong, balanced flow in the x-direction. We run our model for five simulated days to show that it correctly maintains this steady state. In addition to showing that our TT solver produces flows equivalent to those produced by standard methods, we also show that the TT solver conserves thickness and energy within the expected error range. The unstable barotropic jet test case from (Galewsky et al. 2004) will be included in future work, as the more complex dynamics will likely benefit from the further development of our TT methods.

We solve the nonlinear SWEs (Equations (1) and (3)) with $g = 9.80616$ m/s^2, on the domain $\Omega = [0, L] \times \left[0, \frac{L}{2}\right]$, with $L = 2\pi \times 6371220$ m (the circumference of Earth). The computational domain is discretized by $N_x \times N_y$ grids, where $N_x = 5 \times 2^n$ for $n = 3, 4, \dots, 9$ and $N_y = N_x/2$. The domain is periodic in x, with no-flow boundary conditions at $y = 0$ and $y = \frac{L}{2}$. To aid readability and implementation, we define two functions $\theta : [0, L] \to [0, 2\pi]$ and $\phi : \left[0, \frac{L}{2}\right] \to \left[\frac{-\pi}{2}, \frac{\pi}{2}\right]$ that map the x and y coordinates to domains normally associated with longitude and latitude,

$$
\begin{aligned}
\theta(x) &= \frac{2\pi}{L} x\,, \\
\phi(y) &= \frac{2\pi}{L} y - \frac{\pi}{2}\,.
\end{aligned}
\tag{33}
$$

The fluid velocity is initialized to so that the vertical component is equal to zero, and the horizontal component depends only on y,

$$
u(y) = \begin{cases} 0 & \phi(y) \le \phi_0 \\ C \exp\left(\frac{1}{\left(\phi(y)-\phi_0\right)\left(\phi(y)-\phi_1\right)}\right) & \phi_0 < \phi(y) < \phi_1\,, \\ 0 & \phi(y) \ge \phi_1 \end{cases}
\tag{34}
$$

where $\phi_0 = \frac{-\pi}{7}$ and $\phi_1 = \frac{\pi}{7}$, $C = u_{\max} \exp\left(\frac{4}{(\phi_1 - \phi_0)^2}\right)$, and $u_{\max} = 80$ m/s. Then, a balanced initial condition for the thickness can be obtained by setting the time-derivative of the velocity to zero, inserting Equation (34) into the momentum equation, and solving for the thickness h. This gives us,

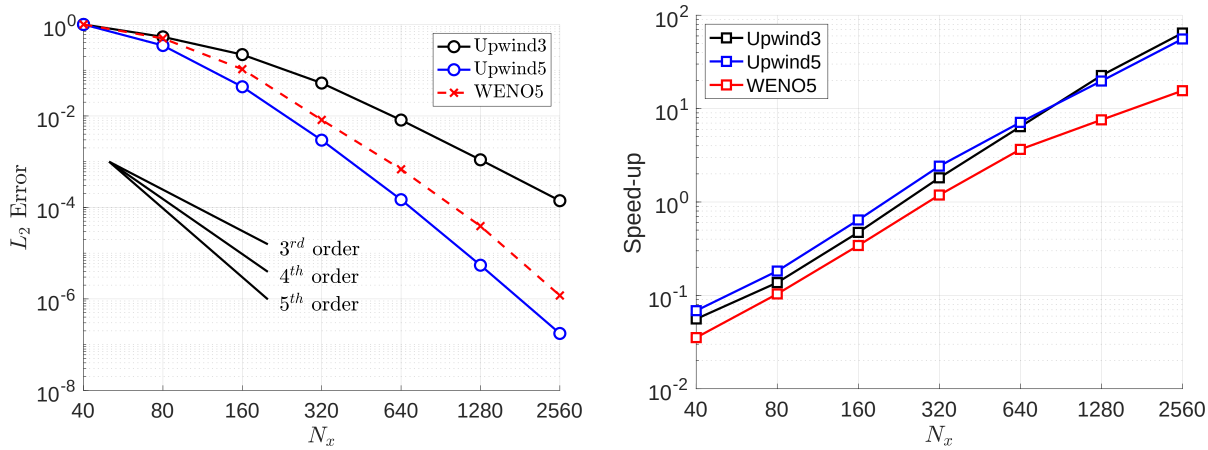
$$h_{\text{balance}}(y) = h_0 - \frac{1}{g}\int_0^y f\,u(y')\;\mathrm{d}y', \tag{35}$$

where h_0 is a constant chosen so that the average value of the thickness is equal to 10,000 m. For simplicity, we choose a constant value for the Coriolis force $f = 2\omega\sin\left(\frac{\pi}{4}\right)$, where $\omega = 7.292\times10^{-5}$ s^{-1} is the angular velocity of the Earth's rotation.

In Figure 11a, L_2-errors are measured by comparing the solution at $T = 5$ days to the exact solution taken from the initial conditions. All reconstruction schemes recover their respective formal convergence rate as the mesh is gradually refined. Unlike the previous cases, Upwind5 attains error values almost an order of magnitude smaller than those obtained by WENO5. This result is a clear outcome of smaller numerical dissipation levels of Upwind5 compared to WENO5, resulting in smaller modifications of the stable initial condition during the course of the simulation. Figure 11b shows the speed-up values for all reconstruction schemes. The full-tensor simulations for the grid levels $n = 3, 4, \ldots, 7$ were run up to the final time $T = 5$ days. At the grid levels $n = 8$ and 9, the full-tensor simulation duration was measured for 100 times steps, which is then extrapolated to estimate the total computational cost at $T = 5$ days. Note that the tensor-train simulations were run up to the final time $T = 5$ days at all grid levels. As in the previous examples, the cross interpolation procedure used for WENO5 results in lower levels acceleration. At the finest grid level, speed-up values are 64.5 for Upwind3, 55.7 for Upwind5 and 15.6 for WENO5.

The global conservation properties of the proposed TT schemes are investigated in Figure 12. Total mass is calculated as $h(t) = \int_\Omega h(t)\,\mathrm{d}A$ and total energy is calculated as $E(t) = \int_\Omega \frac{1}{2}(u(t)^2 + v(t)^2)\,\mathrm{d}A$ at any given time t. Tensor-train errors are denoted by solid lines and full-tensor errors are denoted by symbols with the same color code for the same grid level. The vertical axis shows the difference between the total mass/energy at time $T = 0$ versus the total mass/energy at time $T = t$, referred to as the total mass/energy error. Finally, this error is normalized by dividing by the total energy at time $T = 0$. If these quantities were exactly conserved, the values on the vertical axis would be zero. This way of showing global conservation over time is similar to that used in (Calandrini et al. 2021).

Figures 12a and 12c compare total mass errors of the tensor-train solver to those of the standard full-tensor solver for Upwind3 and Upwind5, respectively. Note that h is a conserved variable solved in Equation (1). Therefore, the mismatch between the tensor-train and full-tensor error



(a) L_2 error for h

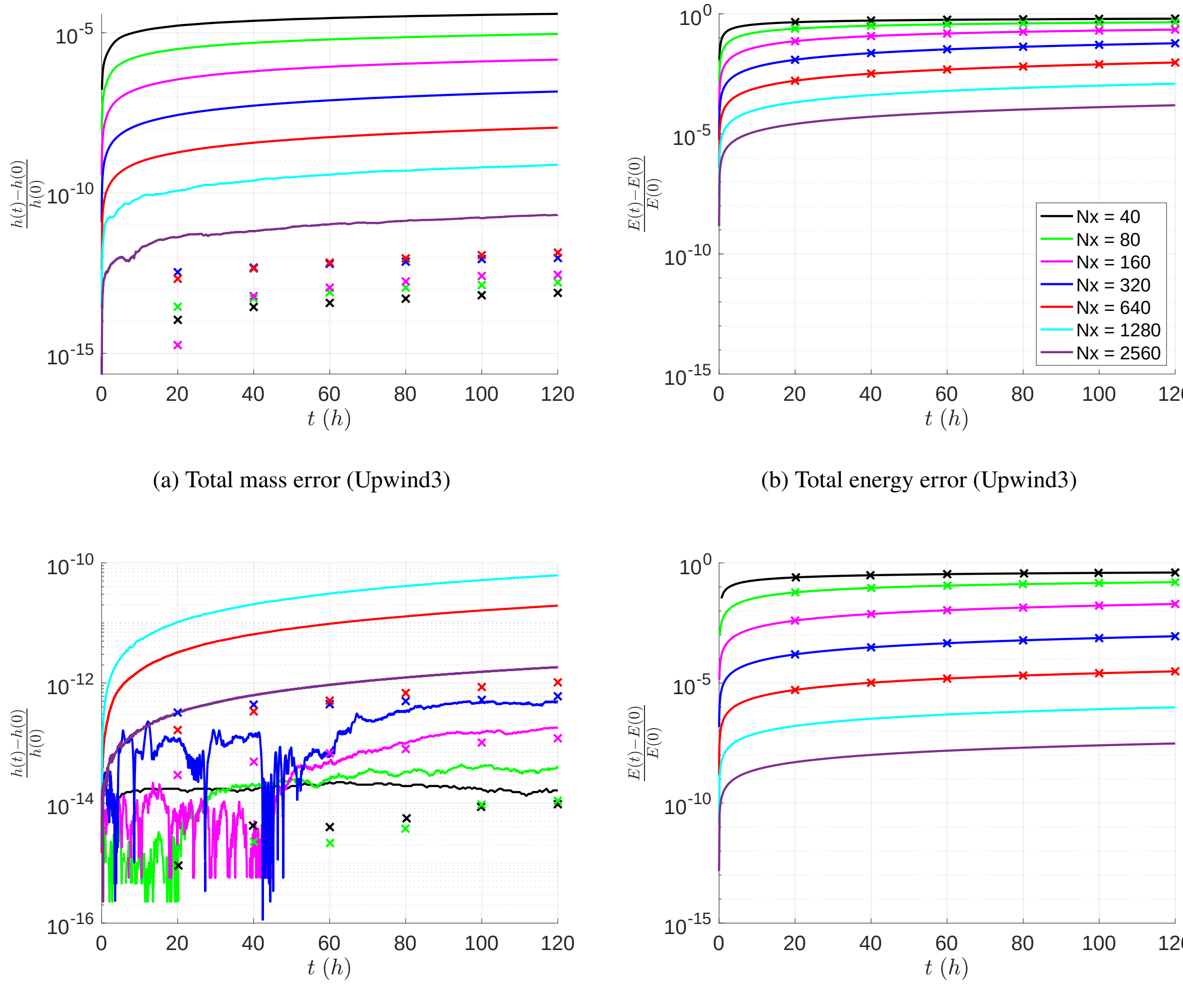
(b) Speed-up

FIG. 11: L_2 errors of the shallow water layer thickness and speed-up for the Stable Barotropic Jet case at $T = 5$ days.

levels is a direct measure of applying TT rounding/truncation at the end of each Runge-Kutta stage, where additional numerical dissipation could be introduced. Despite this mismatch, however, the total mass errors for the TT solvers stabilizes over time and tend to converge asymptotically to a steady state value. On the other hand, it is interesting that the TT solver resulted in total energy errors that are almost identical to those obtained by the full-tensor solvers, as shown in Figures 12b and 12d. In all cases, Upwind5 provides lower levels of total mass/energy errors, demonstrating the advantage of using higher-order schemes where numerical dissipation is lower.

6. Conclusion

In this study, we developed a high-order tensor-train finite volume solver for linear and nonlinear shallow water equations (SWEs). The implementation of the 3^{rd} order Upwind3, 5^{th} order Upwind5, and 5^{th} order WENO5 reconstruction schemes in the tensor-train format were presented in detail. We demonstrated that all reconstruction schemes achieved their formal order of convergence in the TT format for the linear and nonlinear SWEs. The upwind methods perform reconstruction procedures that directly modify TT cores while the WENO scheme uses TT cross interpolation, due to which the WENO scheme was the method with lowest speed-up values among all numerical results. For the linear SWEs, the fluxes are computed directly without using any TT rounding



(a) Total mass error (Upwind3)

(b) Total energy error (Upwind3)

(c) Total mass error (Upwind5)

(d) Total energy error (Upwind5)

Fig. 12: Conservation of total mass and energy for the Stable Barotropic Jet case. Tensor-train errors are denoted by solid lines and full-tensor errors are denoted by symbols with the same color code for the same grid level.

or cross interpolation, which resulted in speed-up values up to 124x compared to the standard full-tensor implementation. However, the nonlinear SWEs need to calculate the reciprocal of the shallow water layer thickness (SWLT) in the TT format followed by applying TT rounding several times, which was reflected in our numerical results by reduced the speed-up values compared to the linear case. To compute the TT reciprocal of the SWLT, we employed an approximation based on Taylor series, which turned out to be quite efficient since the SWLT can usually be decomposed as a large mean value and small oscillations. Overall, we showed that the TT finite volume method maintains the accuracy of the underlying numerical discretization while significantly accelerating it for the SWEs.

The long term goal of future work will be to adapt TT methods to layered Boussinesq models of the ocean and atmosphere at the climate scale. In the short term, there are two open questions of particular interest. First, we wish to extend these methods to spherical, unstructured grids like those employed by the ocean and atmosphere components of the U.S. Department of Energy's Energy Exascale Earth System Model (E3SM) (Golaz et al. 2022); it is not clear how the complex geometry of these more sophisticated grids will affect the compression obtained by the TT methods as we additionally move towards more complex flows. The compression of the model state by TT methods is the primary source of speedup in this study, so it needs to be understood how exactly this compression is affected by increasing the complexity of the problem. Eventually, we hope to apply TT methods to 3-dimensional ocean and atmosphere domains; we are confident that this will provided increased speedup versus standard methods as the benefit of TT compression increases with problem size and dimension. Second, as these methods are advanced towards climate applications, we need to compare performance gains against leadership-class climate codes. This will require that we effectively leverage high-performance computing (HPC) hardware and libraries so that a direct comparison can be made. We expect that moving towards HPC architectures will reveal additional speedups, as much of the currently emerging HPC hardware specifically targets the types of tensor operations that are used by TT methods. The results obtained here and in similar works on TT methods suggest that these methods could precipitate a paradigm shift in how we model geophysical fluids.

Acknowledgments. The authors gratefully acknowledge the support of the Laboratory Directed Research and Development (LDRD) program of Los Alamos National Laboratory under project numbers 20230067DR and 20240782ER and ISTI Rapid Response 20248215CT-IST. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001). The authors thank Siddhartha Bishnu and Joseph Schoonover for useful discussions about the test cases.

Data availability statement. The data that support the findings of this research are available from the corresponding author upon reasonable request.

APPENDIX

A1. High Order Reconstructions for Upwind and WENO Methods

Here, we present the details necessary to implement the Upwind3, Upwind5, and WENO5 reconstruction schemes.

a. *Step 1: "De-cell" averaging over x*

First, we discuss the reconstruction of $\widetilde{u}^-_{i+\frac{1}{2},j}$, which is a 1-dimensional cell average in y. The procedure to reconstruct $\widetilde{u}^+_{i-\frac{1}{2},j}$ is similar (see (Shu 1998)) therefore it will not be presented here. First, we set $v_{i,j} = \widetilde{\overline{u}}_{i,j}$ but the index j is dropped in $v_{i,j}$ below for simplicity:

$$\widetilde{u}^-_{i+\frac{1}{2},j} = \sum_{r=0}^{k} \omega_r v^{(r)}_{i+1/2} . \tag{A1}$$

For the 3rd order upwind method (Upwind3), $k = 1$ and

$$\begin{aligned} v^{(0)}_{i+1/2} &= \frac{1}{2}\left(v_i + v_{i+1}\right) , \\ v^{(1)}_{i+1/2} &= \frac{1}{2}\left(-v_{i-1} + 3v_i\right) , \end{aligned} \tag{A2}$$

while for the 5th order Upwind5 and WENO5 schemes $k = 2$ and

$$\begin{aligned} v^{(0)}_{i+1/2} &= \frac{1}{6}\left(2v_i + 5v_{i+1} - v_{i+2}\right) , \\ v^{(1)}_{i+1/2} &= \frac{1}{6}\left(-v_{i-1} + 5v_i + 2v_{i+1}\right) , \\ v^{(2)}_{i+1/2} &= \frac{1}{6}\left(2v_{i-2} - 7v_{i-1} + 11v_i\right) . \end{aligned} \tag{A3}$$

In the upwind methods ω_r are determined from the ideal linear weights. For example, $\omega_0 = 2/3$ and $\omega_1 = 1/3$ for Upwind3 while $\omega_0 = 3/10$, $\omega_1 = 3/5$ and $\omega_2 = 1/10$ for Upwind5. For the WENO5 reconstruction, we compute the nonlinear weights using

$$\omega_r = \frac{\alpha_r}{\sum_{s=0}^{2} \alpha_s} , \tag{A4}$$

where $\alpha_r = d_r/\left(\beta_r+\varepsilon\right)^2$ for $r = 0,1,2$, $\varepsilon = \Delta x^2$ as suggested by (Don and Borges 2013), $d_0 = 3/10$, $d_1 = 3/5$ and $d_2 = 1/10$ (same as the ideal linear weights), and the smoothness indicators are given as

$$
\begin{aligned}
\beta_0 &= \frac{13}{12}\left(v_i - 2v_{i+1} + v_{i+2}\right)^2 + \frac{1}{4}\left(3v_i - 4v_{i+1} + v_{i+2}\right)^2 , \\
\beta_1 &= \frac{13}{12}\left(v_{i-1} - 2v_i + v_{i+1}\right)^2 + \frac{1}{4}\left(v_{i-1} - v_{i+1}\right)^2 , \\
\beta_2 &= \frac{13}{12}\left(v_{i-2} - 2v_{i-1} + v_i\right)^2 + \frac{1}{4}\left(v_{i-2} - 4v_{i-1} + 3v_i\right)^2 .
\end{aligned} \tag{A5}
$$

b. *Step 2: "De-cell" averaging over y*

Now that we have 1-dimensional cell averages in y, we will next complete the "de-cell" averaging by reconstructing point-wise values at $(x_{i\pm1/2}, y_j + \delta_m \Delta y)$ for each quadrature point m. The procedure is similar to Step 1, but linear and nonlinear reconstruction weights are specifically defined for each quadrature point m.

As in Step 1, we will set $v_{i+1/2,j} = \widetilde{u}^{-}_{i+\frac{1}{2},j}$ but drop the index $i+1/2$ in $v_{i+1/2,,j}$ for simplicity. Then,

$$
u^{-}_{i+\frac{1}{2}, y_j+\delta_m\Delta y} = \sum_{r=0}^{k} \omega_m^{(r)} v_m^{(r)} , \tag{A6}
$$

where

$$
v_m^{(r)} = \sum_{l=0}^{k} c_{rl}^{m} v_{j-r+l} , \tag{A7}
$$

and $\{c_{rl}\}^m$ denotes the components of the coefficient matrix C^m for the particular quadrature point m.

For the Upwind3 scheme where $k = 1$, we perform the reconstruction procedure on 2 quadrature points. The coefficient matrix for each quadrature point is given as

$$
C^{m=1} = \begin{pmatrix} 808/627 & -390/1351 \\ 390/1351 & 961/1351 \end{pmatrix}, \qquad C^{m=2} = \begin{pmatrix} 961/1351 & 390/1351 \\ -390/1351 & 808/627 \end{pmatrix}, \tag{A8}
$$

and the linear weights are $\omega_m^{(r)} = 1/2$ for $r = 0,1$ and $m = 1,2$.

For the Upwind5 and WENO5 schemes where $k = 2$, we consider a reconstruction on 3 quadrature points. The coefficient matrices for these points is given as

$$
\begin{aligned}
C^{m=1} &= \begin{pmatrix} 4725/2927 & -2051/2438 & 249/1097 \\ 249/1097 & 14/15 & -467/2913 \\ -467/2913 & 366/517 & 2209/4883 \end{pmatrix}, \\
C^{m=2} &= \begin{pmatrix} 23/24 & 1/12 & -1/24 \\ -1/24 & 13/12 & -1/24 \\ -1/24 & 1/12 & 23/24 \end{pmatrix}, \\
C^{m=3} &= \begin{pmatrix} 2209/4883 & 366/517 & -467/2913 \\ -467/2913 & 14/15 & 249/1097 \\ 249/1097 & -2051/2438 & 4725/2927 \end{pmatrix}.
\end{aligned}
\tag{A9}
$$

For the 5th order schemes with 3 quadrature points along the cell interfaces, the ideal linear coefficients of the second quadrature point, $m = 2$, become negative and these are treated according to the method presented in (Shi et al. 2002). Following their method, we first set the linear weights for the first and the last quadrature point as:

$$
\begin{aligned}
&\gamma_{m=1}^{(r=0)} = 882/6305, \quad \gamma_{m=1}^{(r=1)} = 403/655, \quad \gamma_{m=1}^{(r=2)} = 463/1891, \\
&\gamma_{m=3}^{(r=0)} = 463/1891, \quad \gamma_{m=3}^{(r=1)} = 403/655, \quad \gamma_{m=3}^{(r=2)} = 882/6305.
\end{aligned}
$$

Then, the split coefficients are used for the second quadrature point:

$$
\begin{aligned}
&\gamma_{m=2}^{(r=0)+} = 9/214, \quad \gamma_{m=2}^{(r=1)+} = 98/107, \quad \gamma_{m=2}^{(r=2)+} = 9/214, \\
&\gamma_{m=2}^{(r=0)-} = 9/67, \quad \gamma_{m=2}^{(r=1)-} = 49/67, \quad \gamma_{m=2}^{(r=2)-} = 9/67.
\end{aligned}
$$

The Upwind5 scheme simply sets $\omega_m^{(r)} = \gamma_m^{(r)}$ for $m = 1, 3$ and $r = 0, 1, 2$ as these coefficients are already positive, then applies Equation (A6). For the second quadrature point $m = 2$, we perform the reconstruction using

$$
u^{-}_{i+\frac{1}{2}, y_j + \delta_2 \Delta y} = \sigma^{+} u^{+} - \sigma^{-} u^{-}, \tag{A10}
$$

where $\sigma^+ = 107/40$ and $\sigma^- = 67/40$ with

$$u^\pm = \sum_{r=0}^{k} \gamma_{m=2}^{(r)\pm} v_{m=2}^{(r)}, \tag{A11}$$

The WENO5 scheme is similar to Upwind5 but it computes the nonlinear weights, instead. For $m = 1, 3$, it uses Equation (A6) with the nonlinear weights

$$\omega_m^{(r)} = \frac{\alpha_m^{(r)}}{\sum_{s=0}^{2} \alpha_m^{(s)}}, \tag{A12}$$

where $\alpha_m^{(r)} = \gamma_m^{(r)} / \left(\beta_r + \varepsilon\right)^2$ for $r = 0, 1, 2$ and the smoothness indicators are given as

$$\begin{aligned}
\beta_0 &= \frac{13}{12}\left(v_j - 2v_{j+1} + v_{j+2}\right)^2 + \frac{1}{4}\left(3v_j - 4v_{j+1} + v_{j+2}\right)^2, \\
\beta_1 &= \frac{13}{12}\left(v_{j-1} - 2v_j + v_{j+1}\right)^2 + \frac{1}{4}\left(v_{j-1} - v_{j+1}\right)^2, \\
\beta_2 &= \frac{13}{12}\left(v_{j-2} - 2v_{j-1} + v_j\right)^2 + \frac{1}{4}\left(v_{j-2} - 4v_{j-1} + 3v_j\right)^2.
\end{aligned} \tag{A13}$$

For the second quadrature point $m = 2$, WENO5 uses Equation (A10) with the same $\sigma^\pm$ but it sets $u^\pm$ as

$$u^\pm = \sum_{r=0}^{k} \omega_{m=2}^{(r)\pm} v_{m=2}^{(r)}, \tag{A14}$$

where $\alpha_2^{(r)\pm} = \gamma_{m=2}^{(r)\pm} / \left(\beta_r + \varepsilon\right)^2$ for $r = 0, 1, 2$.

For the reconstruction procedure in the y-direction, the above-mentioned procedures are repeated by applying Step 1 in the y-direction to construct $\overline{u}_{i,j+1/2}$ (a 1-dimensional cell average in x) and Step 2 in the x-direction to compute $u^\pm_{x_i+\delta_m \Delta x, y_{j\mp1/2}}$ on quadrature points m.

A2. Nondimensional Form of the Shallow Water Equations

Here, we will briefly describe the ‘‘nondimensionalization’’ of the governing equations. Denoting dimensional variables by a star, let us rewrite the shallow water equations:

$$\frac{\partial \boldsymbol{U}^*}{\partial t^*} + \frac{\partial \boldsymbol{F}^*}{\partial x^*} + \frac{\partial \boldsymbol{G}^*}{\partial y^*} = \boldsymbol{S}^*. \tag{A15}$$

For the linear case, we have

$$\boldsymbol{U}^* = \begin{pmatrix} \eta^* \\ u^* \\ v^* \end{pmatrix}, \qquad \boldsymbol{F}^* = \begin{pmatrix} H^* u^* \\ g^* \eta^* \\ 0 \end{pmatrix}, \qquad \boldsymbol{G}^* = \begin{pmatrix} H^* v^* \\ 0 \\ g^* \eta^* \end{pmatrix}, \qquad \boldsymbol{S}^* = \begin{pmatrix} 0 \\ f^* v^* \\ -f^* u^* \end{pmatrix}, \tag{A16}$$

and for the nonlinear case,

$$\boldsymbol{U}^* = \begin{pmatrix} h^* \\ h^* u^* \\ h^* v^* \end{pmatrix}, \qquad \boldsymbol{F}^* = \begin{pmatrix} h^* u^* \\ h^* u^{*2} + \frac{1}{2} g^* h^{*2} \\ h^* u^* v^* \end{pmatrix}, \qquad \boldsymbol{G}^* = \begin{pmatrix} h^* v^* \\ h^* u^* v^* \\ h^* v^{*2} + \frac{1}{2} g^* h^{*2} \end{pmatrix}, \qquad \boldsymbol{S}^* = \begin{pmatrix} 0 \\ f^* h^* v^* \\ -f^* h^* u^* \end{pmatrix}. \tag{A17}$$

Next, we define characteristic scales for each variable in the SWEs, such as the reference length L^*_{ref}, reference velocity U^*_{ref}, reference time $t^*_{ref} = L^*_{ref}/U^*_{ref}$, reference height H^*_{ref}, reference acceleration $g^*_{ref} = U^{*2}_{ref}/H^*_{ref}$, reference frequency $f^*_{ref} = U^*_{ref}/L^*_{ref}$. Then, the nondimensional parameters are defined as

$$\begin{aligned} t &= \frac{t^*}{t^*_{ref}}, \quad x = \frac{x^*}{L^*_{ref}}, \quad y = \frac{y^*}{L^*_{ref}}, \\ \eta &= \frac{\eta^*}{H^*_{ref}}, \quad h = \frac{h^*}{H^*_{ref}}, \\ u &= \frac{u^*}{U^*_{ref}}, \quad v = \frac{v^*}{U^*_{ref}}, \\ g &= \frac{g^*}{g^*_{ref}}, \quad f = \frac{f^*}{f^*_{ref}}. \end{aligned} \tag{A18}$$

Substituting these into Equations (A15) to (A17) gives Equations (1) to (3). Note that both set of equations have the same form. However, in the dimensional form, C_ε in Equations (24) and (25) needs to attain extremely small values, e.g. $C_\varepsilon = 10^{-18} - 10^{-22}$, due to extremely large computational domains. On the other hand, the nondimensional form significantly decreases this ambiguity and $C_\varepsilon = 1$ seems to work well in the most numerical examples discussed in Section 5.

Finally, we summarize the reference values used in the nondimensionalization of the governing equations in Table A1 below.

Test Case	L^*_{ref} (m)	U^*_{ref} (m/s)	H^*_{ref} (m)
Coastal Kelvin Wave	5×10^6	5×10^{-3}	0.1
Inertia-Gravity Wave	10^7	1.622×10^{-3}	0.2
Barotropic Tide	25×10^4	3.163×10^{-3}	0.2
Manufactured Solution	10^7	10^{-2}	500
Stable Barotropic Jet	$2\pi\times6371220$	20	8000

TABLE A1: Characteristic scales for numerical examples.

References

Alexandrov, B., G. Manzini, E. W. Skau, P. M. D. Truong, and R. G. Vuchov, 2023: Challenging the curse of dimensionality in multidimensional numerical integration by using a low-rank tensor-train format. *Mathematics*, **11 (3)**, 534.

Archibald, R., K. J. Evans, J. Drake, and J. B. White, 2011: Multiwavelet discontinuous galerkin-accelerated exact linear part (elp) method for the shallow-water equations on the cubed sphere. *Monthly Weather Review*, **139 (2)**, 457 – 473, https://doi.org/10.1175/2010MWR3271.1.

Bachmayr, M., 2023: Low-rank tensor methods for partial differential equations. *Acta Numerica*, **32**, 1–121.

Bishnu, S., M. R. Petersen, B. Quaife, and J. Schoonover, 2024: A verification suite of test cases for the barotropic solver of ocean models. *Journal of Advances in Modeling Earth Systems*, **16 (4)**, e2022MS003 545, https://doi.org/https://doi.org/10.1029/2022MS003545, https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2022MS003545.

Calandrini, S., D. Engwirda, and M. R. Petersen, 2021: Comparing numerical accuracy and stability for different horizontal discretizations in MPAS-Ocean. *Ocean Modelling*, **168**, 101 908, https://doi.org/https://doi.org/10.1016/j.ocemod.2021.101908.

Caldwell, P. M., and Coauthors, 2019: The DOE E3SM Coupled Model Version 1: Description and Results at High Resolution. *Journal of Advances in Modeling Earth Systems*, **11 (12)**, 4095–4146, https://doi.org/10.1029/2019MS001870.

Cichocki, A., 2014: Tensor networks for big data analytics and large-scale optimization problems. *arXiv preprint arXiv:1407.3124*.

Clarke, A. J., and D. S. Battisti, 1981: The effect of continental shelves on tides. *Deep Sea Research Part A. Oceanographic Research Papers*, **28 (7)**, 665–682, https://doi.org/https://doi.org/10.1016/0198-0149(81)90128-X.

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

Danis, M. E., and B. Alexandrov, 2023: Navier-Stokes: Tensorization of the Sod and Blow-off problems, https://www.osti.gov/biblio/2346026, LANL, LA-UR-24-24283. Tech. rep., Los Alamos National Laboratory.

Danis, M. E., D. Truong, I. Boureima, O. Korobkin, K. Ø. Rasmussen, and B. S. Alexandrov, 2025: Tensor-train WENO scheme for compressible flows. *Journal of Computational Physics*, **529**, 113 891, https://doi.org/10.1016/j.jcp.2025.113891.

Demir, S., R. Johnson, B. T. Bojko, A. T. Corrigan, D. A. Kessler, P. Guthrey, J. Burmark, and S. Irving, 2024: *Three-Dimensional Compressible Chemically Reacting Computational Fluid Dynamics with Tensor Trains*. AIAA, https://doi.org/10.2514/6.2024-1337, URL https://arc.aiaa.org/doi/abs/10.2514/6.2024-1337, https://arc.aiaa.org/doi/pdf/10.2514/6.2024-1337.

Dennis, J., A. Fournier, W. F. Spotz, A. St-Cyr, M. A. Taylor, S. J. Thomas, and H. Tufo, 2005: High-Resolution Mesh Convergence Properties and Parallel Efficiency of a Spectral Element Atmospheric Dynamical Core. *The International Journal of High Performance Computing Applications*, **19 (3)**, 225–235, https://doi.org/10.1177/1094342005056108.

Dolgov, S. V., and D. V. Savostyanov, 2014: Alternating minimal energy methods for linear systems in higher dimensions. *SIAM Journal on Scientific Computing*, **36 (5)**, A2248–A2271.

Don, W.-S., and R. Borges, 2013: Accuracy of the weighted essentially non-oscillatory conservative finite difference schemes. *Journal of Computational Physics*, **250**, 347–372, https://doi.org/https://doi.org/10.1016/j.jcp.2013.05.018.

Donahue, A. S., P. M. Caldwell, and Coauthors, 2024: To exascale and beyond—the simple cloud-resolving e3sm atmosphere model (SCREAM), a performance portable global atmosphere model for cloud-resolving scales. *Journal of Advances in Modeling Earth Systems*, **16 (7)**, e2024MS004 314, https://doi.org/https://doi.org/10.1029/2024MS004314, https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2024MS004314.

Galewsky, J., R. K. Scott, and L. M. Polvani, 2004: An initial-value problem for testing numerical models of the global shallow-water equations. *Tellus A: Dynamic Meteorology and Oceanography*, **56 (5)**, 429–440, https://doi.org/10.3402/tellusa.v56i5.14436.

Golaz, J.-C., and Coauthors, 2022: The DOE E3SM Model Version 2: Overview of the Physical Model and Initial Model Evaluation. *Journal of Advances in Modeling Earth Systems*, **14 (12)**, e2022MS003 156, https://doi.org/10.1029/2022MS003156.

Goreinov, S. A., I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, 2010: How to find a good submatrix. *Matrix Methods: Theory, Algorithms And Applications: Dedicated to the Memory of Gene Golub*, World Scientific, 247–256.

Gottlieb, S., C.-W. Shu, and E. Tadmor, 2001: Strong stability-preserving high-order time discretization methods. *SIAM Review*, **43 (1)**, 89–112, https://doi.org/10.1137/S003614450036757X, https://doi.org/10.1137/S003614450036757X.

Gourianov, N., 2022: Exploiting the structure of turbulence with tensor networks. http://purl.org/dc/dcmitype/Text, University of Oxford.

Johnson, M. A., and J. J. O'Brien, 1990: The role of coastal kelvin waves on the northeast pacific ocean. *Journal of Marine Systems*, **1 (1)**, 29–38, https://doi.org/https://doi.org/10.1016/0924-7963(90)90085-O.

Kay, J. E., and Coauthors, 2015: The community earth system model (cesm) large ensemble project: A community resource for studying climate change in the presence of internal climate variability. *Bulletin of the American Meteorological Society*, **96 (8)**, 1333 – 1349, https://doi.org/10.1175/BAMS-D-13-00255.1.

Kiehl, J. T., J. J. Hack, G. B. Bonan, B. A. Boville, D. L. Williamson, and P. J. Rasch, 1998: The national center for atmospheric research community climate model: Ccm3. *Journal of Climate*, **11 (6)**, 1131 – 1149, https://doi.org/10.1175/1520-0442(1998)011<1131:TNCFAR>2.0.CO;2.

Kiffner, M., and D. Jaksch, 2023: Tensor network reduced order models for wall-bounded flows. *Phys. Rev. Fluids*, **8**, 124 101, https://doi.org/10.1103/PhysRevFluids.8.124101.

Lilly, J. R., D. Engwirda, G. Capodaglio, R. L. Higdon, and M. R. Petersen, 2023: Cfl optimized forward–backward runge–kutta schemes for the shallow-water equations. *Monthly Weather Review*, **151 (12)**, 3191 – 3208, https://doi.org/10.1175/MWR-D-23-0113.1.

Mahoney, M. W., and P. Drineas, 2009: Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, **106 (3)**, 697–702.

Oseledets, I., and E. Tyrtyshnikov, 2010: Tt-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, **432 (1)**, 70–88.

Oseledets, I. V., 2011a: Tensor-train decomposition. *SIAM Journal on Scientific Computing*, **33 (5)**, 2295–2317.

Oseledets, I. V., 2011b: Tensor-train decomposition. *SIAM Journal on Scientific Computing*, **33 (5)**, 2295–2317, https://doi.org/10.1137/090752286, https://doi.org/10.1137/090752286.

Oseledets, I. V., 2014: oseledets/TT-Toolbox. GitHub, URL https://github.com/oseledets/TT-Toolbox.

Petersen, M. R., D. W. Jacobsen, T. D. Ringler, M. W. Hecht, and M. E. Maltrud, 2015: Evaluation of the arbitrary Lagrangian–Eulerian vertical coordinate method in the MPAS-Ocean model. *Ocean Modelling*, **86**, 93–113, https://doi.org/10.1016/j.ocemod.2014.12.004.

Roache, P. J., 2019: *The Method of Manufactured Solutions for Code Verification*, 295–318. Springer International Publishing, Cham, https://doi.org/10.1007/978-3-319-70766-2_12, URL https://doi.org/10.1007/978-3-319-70766-2_12.

Savostyanov, D., and I. Oseledets, 2011: Fast adaptive interpolation of multi-dimensional arrays in tensor train format. *The 2011 International Workshop on Multidimensional (nD) Systems*, IEEE, 1–8.

Shi, J., C. Hu, and C.-W. Shu, 2002: A technique of treating negative weights in weno schemes. *Journal of Computational Physics*, **175 (1)**, 108–127, https://doi.org/https://doi.org/10.1006/jcph.2001.6892.

Shu, C.-W., 1998: *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, 325–432. Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/BFb0096355, URL https://doi.org/10.1007/BFb0096355.

Shu, C.-W., and S. Osher, 1988: Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, **77 (2)**, 439–471, https://doi.org/https://doi.org/10.1016/0021-9991(88)90177-5.

Thuburn, J., T. D. Ringler, W. C. Skamarock, and J. B. Klemp, 2009: Numerical representation of geostrophic modes on arbitrarily structured C-grids. *Journal of Computational Physics*, **228 (22)**, 8321–8335, https://doi.org/10.1016/j.jcp.2009.08.006.

Truong, D. P., M. I. Ortega, I. Boureima, G. Manzini, K. Ø. Rasmussen, and B. S. Alexandrov, 2024: Tensor networks for solving the time-independent boltzmann neutron transport equation. *Journal of Computational Physics*, **507**, 112 943.

von Larcher, T., and R. Klein, 2019: On identification of self-similar characteristics using the tensor train decomposition method with application to channel turbulence flow. *Theorecical Computational Fluid Dynamics*, **33**, 141–159.

Weller, H., H. G. Weller, and A. Fournier, 2009: Voronoi, Delaunay, and Block-Structured Mesh Refinement for Solution of the Shallow-Water Equations on the Sphere. *Monthly Weather Review*, https://doi.org/10.1175/2009MWR2917.1.

Williamson, D. L., J. B. Drake, J. J. Hack, R. Jakob, and P. N. Swarztrauber, 1992: A standard test set for numerical approximations to the shallow water equations in spherical geometry. *Journal of Computational Physics*, **102 (1)**, 211–224.

Young, W. R., 2021: Inertia-gravity waves and geostrophic turbulence. *Journal of Fluid Mechanics*, **920**, F1, https://doi.org/10.1017/jfm.2021.334.