Hindawi
International Journal of Computer Games Technology
Volume 2022, Article ID 4138315, 11 pages
https://doi.org/10.1155/2022/4138315

Research Article

Evaluating the Stability of Numerical Schemes for Fluid Solvers in Game Technology

Craig R. Stark [1] **and Declan A. Diver** [2]

[1]*Complex Multiscale Dynamics Group, Division of Games Technology and Mathematics, Abertay University, Dundee DD1 1HG, UK*
[2]*School of Physics and Astronomy, University of Glasgow, Glasgow G12 8QQ, UK*

Correspondence should be addressed to Craig R. Stark; c.stark@abertay.ac.uk

Received 11 November 2021; Accepted 10 May 2022; Published 2 June 2022

Academic Editor: Peican Zhu

A variety of numerical techniques have been explored to solve the shallow water equations in real-time water simulations for computer graphics applications. However, determining the stability of a numerical algorithm is a complex and involved task when a coupled set of nonlinear partial differential equations need to be solved. This paper proposes a novel and simple technique to compare the relative empirical stability of finite difference (or any grid-based scheme) algorithms by solving the inviscid Burgers' equation to analyse their respective breaking times. To exemplify the method to evaluate numerical stability, a range of finite difference schemes is considered. The technique is effective at evaluating the relative stability of the considered schemes and demonstrates that the conservative schemes have superior stability.

1. Introduction

Fluid dynamics is widely used in computer game technology to simulate a range of phenomena including water, smoke, and soft bodies. A variety of numerical techniques have been explored for simulating fluids in real-time such as position-based dynamics (e.g., [1, 2]), the finite-volume method (e.g., [3–6]), and the finite-element method [7]. There are two broad modelling approaches to describing fluids: grid-based Eulerian or particle-based Lagrangian methods [8, 9]. The Lagrangian approach follows an individual fluid parcel as it moves through space and time, whereas the Eulerian approach describes the fluid motion from specific fixed points in space as time passes. Both approaches have enjoyed widespread use in game technology. Particle-based approaches are advantageous when considering arbitrary boundaries, fluid mixing [10], and interactions with rigid bodies [11]. Grid-based approaches are widely used in computer graphics applications and can attain higher numerical accuracy since dealing with spatial derivatives is easier to accommodate on a fixed grid. However, in contrast to particle-based methods, grid-based approaches have difficulty ensuring the conservation of mass and can be computationally slower. Furthermore, in real-time water simulations in games, grid-based methods perform much better at tracking smooth water surfaces [9].

Smooth Particle Hydrodynamics (SPH) is a popular particle-based method for simulating fluids in game technology, computer animation, virtual reality, and the movie industry. For example, incompressible SPH is a promising numerical scheme for large-scale and large-deformation simulations used in interactive fluid flow simulations [12]. Two of the challenges faced by SPH is unstable solid boundary handling and numerical dissipation, both of which inhibit stable and realistic fluid evolution. Using a position-based dynamics framework integrated into the SPH solver [2, 13], these issues can be overcome [14]. SPH is also useful for simulating viscoelastic materials such as gels, gelatin, and mucus by applying a velocity correction to limit the fluid deformation, making it attractive to soft-body simulation as well. This produces visually accurate results but is offset by computational performance costs; however,

simulations can be accelerated using GPUs [15]. Recently, neural networks have been used to accelerate particle-based fluid simulations that run significantly faster than a GPU Position-Based Fluid Solver whilst preserving visual quality [16].

The Lattice-Boltzmann method (LBM) is a grid-based method that indirectly solves the fluid dynamical equations by solving the Boltzmann equation that describes the underlying particle distribution function [17–19]. The Boltzmann equation is easier to numerically solve in contrast to the classical fluid equations, and LBM can be run efficiently on massively parallel architectures. LBM is useful for solving complex, coupled, and multiphase flows and can accommodate complex boundaries with relative ease [17]. However, tracking and preserving small-scale features, such as a fluid drop or splash, is a challenging problem. Recent work has looked at a novel grid-particle method for reconstructing distribution functions of interface grids and coupling the reconstruction method with the LBM and Volume of Fluid method to track the free surface [20]. The method enhances the accuracy of the reconstruction and helps preserve the fluid surface detail.

The shallow water equations (SWEs) are a simplified set of fluid flow equations that are widely used in computer graphics applications. The SWEs are well suited to game technology applications due a number of practical reasons: in contrast to the full 3D Navier-Stokes equations, the relative simplicity of the SWEs lead to a significant performance advantage; the solutions produce a full velocity field which is useful for providing plausible interactivity and solid-fluid coupling, and the rendering of the simulation results can be undertaken with common, fast rendering techniques [8, 21]. In contrast, full 3D simulations require surface tracking, volumetric rendering techniques, or mesh generation algorithms for visualisation. Although the SWEs can describe complex nonlinear fluid phenomena such as vortices, it is unable to handle breaking waves or splashing phenomena [8, 21].

A variety of numerical methods exist for solving the SWEs. Finite difference (FD) techniques are particularly popular in real-time applications due to the relative simplicity of their implementation and their ability to capture complex phenomena such as shocks. In particular, a range of finite difference techniques has been applied to the conservative form of the shallow water equations including the Lax-Friedrichs (LF) scheme, the Richtmyer two-step Lax-Wendroff (LW) method, and the McCormack (MC) method [22, 23]. Variations of these popular algorithms have also been investigated such as the corrected LF (CF); composite schemes, e.g., LWLF, MCLF, and CFLC [24, 25]; filtered LW, MC, and CF schemes; and the Picard Integral Formulation of the Weighted Essentially Non-Oscillatory (WENO) scheme [26]. Many of the numerical schemes, such as the LW, MC, and CF methods, suffer from oscillatory phenomenon (high-frequency jitter) as a result of instability. As a consequence, attempts are made to mitigate instabilities from occurring. The filtered schemes tackle the stability issues by using an oscillation smoothing method, for example, Liang et al. [27] and Hsieh et al. [28] for stabilizing their tsunami simulations using the MC scheme. Similarly, Ransom and Younis [29] used a total-variation diminishing limiter with the MC method to help avoid oscillatory behaviour, hence stabilizing the scheme.

Recent studies have proposed a generalized finite difference-split coefficient matrix method along with the flux limiter technique to eliminate potential numerical oscillations that could lead to instability [30]. Li et al. presented a fifth-order weighted essentially nonoscillatory scheme for simulating dam break flows in a finite difference framework [31] that produces smaller truncation errors and provides the same accuracy order and stability as contemporary WENO schemes. To accommodate more complex systems, such as flat-bottom geometry, a new discretization of the source term of the SWEs is suggested by Prieto-Arranz et al. [32]. In the approach, a Smooth Particle Arbitrary Lagrangian-Eulerian formulation based on Riemann solvers is used to solve the SWEs where stability is achieved using a posteriori MOOD paradigm. Benchmarking demonstrates that the MOOD limiting procedure is able to prevent artificial oscillations occurring in the neighbourhood of discontinuities and shocks. Wu et al. presented a high-order entropy stable discontinuous Galerkin method for solving the SWEs on curved triangular meshes that preserves a semi-discrete entropy inequality and remains well-balanced for continuous bathymetry profiles [33]. Such an approach is advantageous as it can accommodate complex geometries through unstructured meshes; it is simple to parallelize and is able to take advantage of acceleration techniques using GPUs.

Determining the stability of a numerical algorithm is a complex task. For linear cases, the von Neumann stability analysis [22] can be applied analytically but is intractable in nontrivial scenarios including complex nonlinear problems, where a set of coupled partial differential equations have to be solved. In nontrivial scenarios, elaborate numerical tests can be used (such as circular dam breaks and flows around a bump) to give an in-depth characterisation of the numerical algorithms performance (e.g., see Parna et al. [26]), but this approach can be time-consuming, complex to set up, and difficult to analyse.

This paper proposes a novel and simple technique to compare the relative empirical stability of finite difference algorithms (or any grid-based scheme) by solving the inviscid Burgers' equation to analyse their respective breaking times. A similar technique has been used to determine suitable plasma fluid solvers in astrophysics [34]. The proposed technique provides a quick and easy way of determining the stability of a proposed algorithm prior to more thorough stability tests tailored to the specific system to be solved and its application. The proposed method is in keeping with the method of A- and L-stability where numerical methods for stiff problems involving ordinary differential equations are analysed by applying them to the test equation $y' = ky$ for $y(0) = 1$, $k \in \mathbb{C}$. In the fluid context, the inviscid Burgers' equation emulates the form of partial differential equations (PDEs) in which there is an advective term such as in Euler's equations of fluid dynamics and the SWEs.

The inviscid Burgers' equation (or the nonlinear wave equation) is given by

$$\partial_t u + u\partial_x u = 0. \tag{1}$$

It describes a nonlinear wave propagating in the positive x-direction with a speed proportional to its amplitude, u. Therefore, larger wave amplitudes propagate faster than smaller components: this causes the wave to break due to nonlinear steepening, like the phenomenon of breaker ocean waves, yielding an advective instability. Obtaining the characteristics for the first-order PDE reveals a solution that may become triple-valued after a period of time—this is wave breaking. The onset of wave breaking is characterised by the solution having a vertical tangent at the leading wave edge. The timescale for this to first occur is the breaking time. In the numerical context, multivalued solutions are not possible and the numerical schemes become unstable. The ensuing nonlinear instability manifests itself as a series of oscillating spikes that form in the wake of the leading wave edge. The relative empirical stability of two numerical methods can be assessed by analysing their respective breaking times. Due to the nature of the instability, where a discontinuity forms due the solution becoming triple-valued, the method also gives an indication of how well numerical schemes deal with discontinuities such as shocks and boundary interactions. To exemplify the method to evaluate numerical stability, a range of FD schemes will be considered.

This paper is structured as follows. The Methodology introduces and derives the FD algorithms that will be used to exemplify the stability analysis technique. It outlines the initial and boundary conditions for the stability evaluation and calculates the corresponding analytical solution, including the breaking time and location, for the system. The Methodology concludes by defining a quantitative metric to evaluate the numerical stability of the numerical schemes. Following this, the Results and Discussion evaluates the numerical stability of the FD algorithms using the quantitative metric. In this paper, a comparison is made of the relative stability of FD numerical methods, but other grid-based numerical schemes could be similarly analysed.

2. Methodology

The finite difference methods are popular numerical schemes for solving partial differential equations (PDEs). The utility and efficacy of these numerical methods depend upon a number of criteria including stability, accuracy, and ease of implementation.

The LW method transforms a continuous problem into a discrete one by replacing spatial and temporal derivatives with second-order accurate finite difference approximations, thus reducing the problem to an iterative algebraic exercise [22]. The LW method is reliable, stable, and accurate. However, depending on the complexity of the individual PDE or if the system is governed by a coupled set of PDEs, implementing LW can be very complicated.

Finite Difference Time Domain (FDTD) is an alternative numerical method [35–37]. It entails recasting the PDE as a series of ordinary differential equations (ODEs) with respect to time by discretizing the spatial domain and replacing the spatial derivatives with finite difference approximations. The ODEs are then solved numerically via an ODE solver such as the Runge-Kutta or leap-frog methods. Its main advantage lies in its ease of implementation.

2.1. Lax-Wendroff Method. To obtain a finite difference solution of (1), the domain described by the PDE is defined in terms of a rectilinear grid, its edges parallel to the x and t axes. We define discrete coordinates so that an arbitrary grid point is given by $(x, t) = (mh, nk)$ such that $m \in [0, m_{\max}]$ and $n \in [0, n_{\max}]$, where m, n are integers and h, k are the grid spacings in the x, t coordinates, respectively. Writing $u(nk, mh) = u_m^n$, the finite difference formulae are obtained from the Taylor expansion of u about t in the neighbourhood of k, while holding x fixed [22],

$$u_m^{n+1} = \exp(k\partial_t) u_m^n, \tag{2}$$

$$\approx \left(1 + k\partial_t + \frac{1}{2}k^2\partial_t^2\right) u_m^n. \tag{3}$$

This is second-order accurate in time. The LW method contains an inherent diffusion term, ∂_t^2, that helps dampen the instability. For the case of the linear wave equation, $\partial_t u_m^n$ is replaced by $-a\partial_x u_m^n$ and $\partial_t^2 u_m^n$ by $a^2\partial_x^2 u_m^n$, where a is a constant wave speed. Therefore,

$$u_m^{n+1} = \left(1 - ka\partial_x + \frac{1}{2}k^2a^2\partial_x^2\right) u_m^n. \tag{4}$$

Using the finite difference approximations:

$$\partial_x u_m^n = \frac{u_{m+1}^n - u_{m-1}^n}{2h} + O\left(h^2\right), \tag{5}$$

$$\partial_x^2 u_m^n = \frac{u_{m+1}^n - 2u_m^n + u_{m-1}^n}{h^2} + O\left(h^2\right). \tag{6}$$

This yields the Lax-Wendroff algorithm for the linear wave equation,

$$\begin{aligned} u_m^{n+1} = {} & \left(1 - p^2a^2\right) u_m^n - \frac{1}{2}pa\left(u_{m+1}^n - u_{m-1}^n\right) \\ & + \frac{1}{2}p^2a^2\left(u_{m+1}^n + u_{m-1}^n\right), \end{aligned} \tag{7}$$

where $p = k/h$ is the mesh ratio. This algorithm is second-order accurate in space and time. One of the advantages of the finite difference method is that it is very easily extended to the solution of nonlinear equations since in all but special cases many of the methods and proofs are invariant [22].

For the nonlinear equation (1), the finite difference formula (3) is used where $\partial_t u_m^n$ is replaced by $-u_m^n \partial_x u_m^n$, using (1), and $\partial_t^2 u_m^n$ is replaced by the following,

$$\partial_t^2 u_m^n = -\partial_t (u_m^n \partial_x u_m^n), \tag{8}$$

$$= -\partial_t u_m^n \partial_x u_m^n - u_m^n \partial_t (\partial_x u_m^n), \tag{9}$$

$$= u_m^n (\partial_x u_m^n)^2 + u_m^n \partial_x (u_m^n \partial_x u_m^n), \tag{10}$$

$$= 2u_m^n (\partial_x u_m^n)^2 + (u_m^n)^2 \partial_x^2 u_m^n. \tag{11}$$

Substituting into the finite difference expression (3) yields

$$u_m^{n+1} = u_m^n - k u_m^n \partial_x u_m^n + k^2 u_m^n (\partial_x u_m^n)^2 + \frac{1}{2} k^2 (u_m^n)^2 \partial_x^2 u_m^n. \tag{12}$$

The Lax-Wendroff algorithm for the nonlinear wave equation becomes

$$u_m^{n+1} = u_m^n - \frac{1}{2} p u_m^n (u_{m+1}^n - u_{m-1}^n) + \frac{1}{4} p^2 u_m^n (u_{m+1}^n - u_{m-1}^n)^2 + \frac{1}{2} p^2 (u_m^n)^2 (u_{m+1}^n - 2u_m^n + u_{m-1}^n), \tag{13}$$

where the finite difference approximations (5) and (6) have been used.

Numerical algorithms for solving partial differential equations are only useful if they are convergent and stable. A finite difference algorithm is deemed convergent if the difference between the theoretical solutions of the differential and difference equations at a fixed coordinate (x, t) tends to zero when the number of grid points in a fixed size numerical domain is increased: $h, k \longrightarrow 0$ and $m, n \longrightarrow \infty$. An algorithm is considered stable when the difference between the theoretical and numerical solutions of the difference equation remains bounded as n tends to infinity. The von Neumann stability analysis for the linear wave equation algorithm states that the scheme is stable provided $0 < p|a| \leq 1$, which appropriately coincides with the Courant-Friedrichs-Lewy (C.F.L) condition for the convergence of the algorithm. This result is used as a guide for the stability and convergence of the nonlinear expression; ergo, $0 < p|u_m^n| \leq 1$ [22].

An alternative numerical algorithm is possible if the second righthand term of (10) is differenced directly: substituting (10) into (3) gives

$$u_m^{n+1} = u_m^n - k u_m^n \partial_x u_m^n + \frac{1}{2} k^2 u_m^n (\partial_x u_m^n)^2 + \frac{1}{2} k^2 u_m^n \partial_x (u_m^n \partial_x u_m^n), \tag{14}$$

where

$$\partial_x (u_m^n \partial_x u_m^n) = \frac{1}{2h} \partial_x [u_m^n (u_{m+1}^n - u_{m-1}^n)] = \frac{1}{4h^2} [u_{m+1}^n (u_{m+2}^n - u_m^n) - u_{m-1}^n (n_m^n - u_{m-2}^n)]. \tag{15}$$

This yields an alternative algorithm

$$u_m^{n+1} = u_m^n - \frac{1}{2} p u_m^n (u_{m+1}^n - u_{m-1}^n) + \frac{1}{8} p^2 u_m^n (u_{m+1}^n - u_{m-1}^n)^2 + \frac{1}{8} p^2 u_m^n [u_{m+1}^n (u_{m+2}^n - u_m^n) - u_{m-1}^n (u_m^n - u_{m-2}^n)]. \tag{16}$$

Note that this version uses next nearest neighbours. Additional finite difference algorithms can be derived by considering the conservative form of the nonlinear wave equation,

$$\partial_t u + \partial_x f = 0, \qquad f(u, x, t) = \frac{u^2}{2}. \tag{17}$$

Following the previous derivation and using (3), the $\partial_t^2 u_m^n$ term is replaced with

$$\partial_t^2 u = -\partial_x (\partial_t f) = -\partial_x (\partial_u f \partial_t u) = \partial_x (\partial_u f \partial_x f) = \partial_x (u \partial_x f), \tag{18}$$

$$= \partial_x u \partial_x f + u \partial_x^2 f. \tag{19}$$

Substituting (19) into (3) and using $\partial_t u_m^n = -\partial_x f_m^n$ yields

$$u_m^{n+1} = u_m^n - k \partial_x f_m^n + \frac{1}{2} k^2 \partial_x u_m^n \partial_x f_m^n + \frac{1}{2} k^2 u_m^n \partial_x^2 f_m^n, \tag{20}$$

where $f_m^n = (u_m^n)^2/2$. The Lax-Wendroff algorithm for the conservative form of the nonlinear wave equation is

$$u_m^{n+1} = u_m^n \left[1 + \frac{1}{4} p^2 \left[(u_{m+1}^n)^2 + (u_{m-1}^n)^2 - 2(u_m^n)^2\right]\right] - \frac{1}{4} p \left[(u_{m+1}^n)^2 - (u_{m-1}^n)^2\right] + \frac{1}{16} p^2 (u_{m+1}^n - u_{m-1}^n) \cdot \left[(u_{m+1}^n)^2 - (u_{m-1}^n)^2\right]. \tag{21}$$

Using (18) in (3) instead of (19) yields

$$u_m^{n+1} = u_m^n - k \partial_x f_m^n + \frac{1}{2} k^2 \partial_x (u_m^n \partial_x f_m^n). \tag{22}$$

Differencing the derivative in the final term directly produces an alternative conservative algorithm:

$$\partial_x(u_m^n \partial_x f_m^n) = \frac{1}{2h}\partial_x[u_m^n(f_{m+1}^n - f_{m-1}^n)] = \frac{1}{4h^2}[u_{m+1}^n(f_{m+2}^n - f_m^n) - u_{m-1}^n(f_m^n - f_{m-2}^n)]. \tag{23}$$

Therefore,

$$u_m^{n+1} = u_m^n\left[1 - \frac{1}{16}p^2 u_m^n(u_{m+1}^n + u_{m-1}^n)\right] - \frac{1}{4}p\left[(u_{m+1}^n)^2 - (u_{m-1}^n)^2\right] + \frac{1}{16}p^2\left[u_{m+1}^n(u_{m+2}^n)^2 + u_{m-1}^n(u_{m-2}^n)^2\right]. \tag{24}$$

2.2. Finite Difference Time Domain Method. The FDTD method is as follows. Using the predefined discrete coordinates $x = mh$ and writing $u(t, mh) = u_m(t) = u_m$, the spatial derivative in the nonlinear equation was replaced by the central difference formula (5), yielding

$$\partial_t u_m = -u_m \frac{(u_{m+1} - u_{m-1})}{2h}. \tag{25}$$

This is a first-order ODE with respect to time where h is the distance between successive spatial grid points. Solving using a 4th-order Runge-Kutta method gives

$$k_1 = -\frac{\Delta t}{2h}u_m(u_{m+1} - u_{m-1}), \tag{26}$$

$$k_2 = -\frac{\Delta t}{2h}\left(u_m + \frac{k_1}{2}\right)(u_{m+1} - u_{m-1}), \tag{27}$$

$$k_3 = -\frac{\Delta t}{2h}\left(u_m + \frac{k_2}{2}\right)(u_{m+1} - u_{m-1}), \tag{28}$$

$$k_4 = -\frac{\Delta t}{2h}(u_m + k_3)(u_{m+1} - u_{m-1}), \tag{29}$$

$$\Delta t = \frac{T}{N}, \tag{30}$$

$$u_m^{n+1} = u_m^n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(\Delta t^5), \tag{31}$$

where Δt is the time step, N is the number of time steps, and T is the integration time such that $T = N\Delta t$. By considering the conservative form of the nonlinear wave equation, another FDTD algorithm can be derived. Substituting for the central difference formula yields

$$\partial_t u_m = -\frac{1}{4h}\left[(u_{m+1})^2 - (u_{m-1})^2\right]. \tag{32}$$

The corresponding Runge-Kutta algorithm is, therefore,

$$k_1 = -\frac{\Delta t}{4h}\left[(u_{m+1})^2 - (u_{m-1})^2\right], \quad k_1 = k_2 = k_3 = k_4, \quad u_m^{n+1} = u_m^n + k_1 + O(\Delta t^5). \tag{33}$$

Both the FDTD algorithms described here are second-order accurate in space and fourth-order accurate in time. For stability, the algorithms require that $v\Delta t \le h$, where v is the maximum expected phase velocity. This ensures that the solution cannot vary significantly over one spatial increment during one temporal step [36].

2.3. Initial and Boundary Conditions. To compare the FD algorithms, they were used to solve the nonlinear wave equation, the inviscid Burgers' equation,

$$\partial_t u + u\partial_x u = 0. \tag{34}$$

For the initial conditions, the wave amplitude was perturbed from a uniform equilibrium in the shape of a Gaussian waveform

$$u_m^0 = A_0 \exp\left[-\varsigma(m - m_0)^2\right], \tag{35}$$

$\forall m$, where A_0 is the amplitude of the perturbation, ς defines its width, and m_0 is the centre of the computational domain (see $n = 0$ plot of Figure 1). Each algorithm was computed for a prescribed length of time $T = n_{\max}k = N\Delta t$ for the specified initial condition. So that the LW and FDTD methods could be compared, the values of $p = k/h$ and Δt had to be chosen carefully. Setting $N = n_{\max}$ requires that $\Delta t = k$ implying $p = k/h = \Delta t/h$. For the simulations, the boundary conditions were $u_0^n = u_{m_{\max}}^n = 0$ for all n. For the simulations discussed here, the parameters listed in Table 1 were used, without loss of generality.

2.4. Analytical Considerations of the Inviscid Burgers' Equation. To evaluate the relative empirical stability of the FD algorithms, it is instructive to evaluate the breaking time and position analytically for the prescribed initial and boundary conditions. The inviscid Burgers' equation is given by

$$\partial_t u + u\partial_x u = 0. \tag{36}$$

Performing the change of variables $x = mh$ and $t = nk$ yields

$$\partial_n u + pu\partial_m u = 0, \tag{37}$$

where $p = k/h$. For the initial conditions, a Gaussian wave packet is prescribed:

$$u(n = 0, m) = A_0 \exp\left[-\varsigma(m - m_0)^2\right]. \tag{38}$$

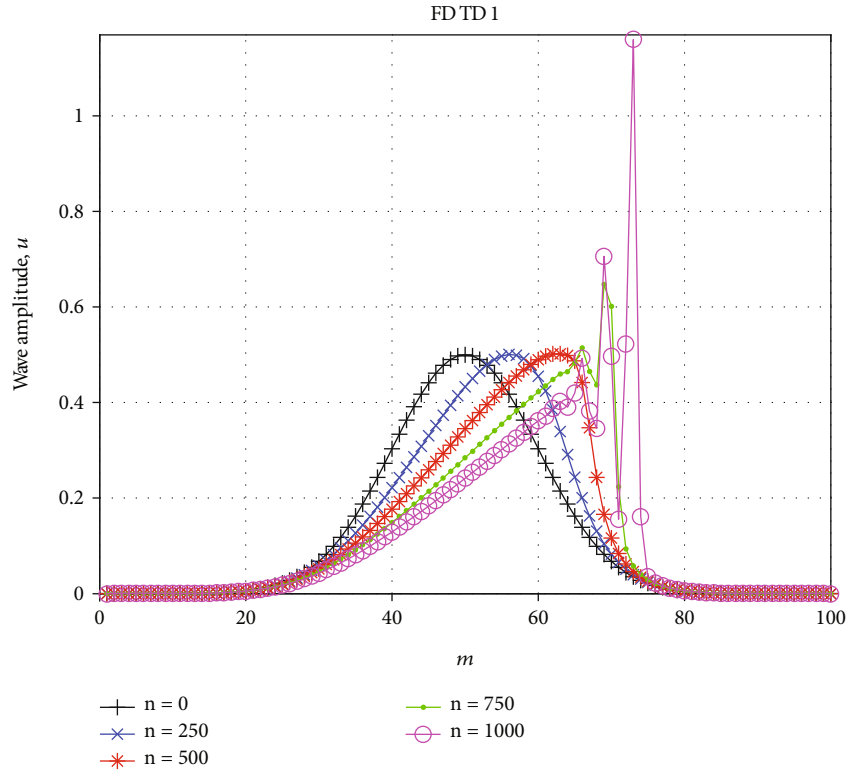

Figure 1: Solution of (1) obtained from the FDTD algorithm (FDTD1) (31). Plots display spatial structure, m, for time steps $n = 0$ (initial conditions), 250, 500, 750, and 1000 exhibiting the evolution of the nonlinear wave and the advective instability. The nonlinear wave has a propagation speed proportional to its amplitude; therefore, the crest of the perturbed wave structure moves faster than its base. As a result, the crest of the wave tries to overtake the smaller amplitudes requiring a multivalued solution, forbidden in this numerical framework resulting in instability. The FDTD solution plotted here exhibits the qualitative behaviour of the numerical methods used in this paper.

Table 1: Parameters for numerical simulations.

Parameter	Value	
A_0	5.0×10^{-1}	Amplitude of the perturbation
m_0	50	Centre of computational domain
$m_{\max}$	100	Number of spatial points
N, $n_{\max}$	1000	Number of time steps
Δt, k	1.0×10^{-2}	Temporal step size
h	0.2	Spatial mesh increment
ς	5.0×10^{-3}	Gaussian coefficient
p	0.05	Mesh ratio

A solution to this initial value problem can be sought using the method of characteristics. The characteristics are

$$\sqrt{\varsigma}(m - m_0) = \sqrt{\varsigma} p A_0 e^{-\sigma^2} n + \sigma, \tag{39}$$

where σ is a constant on a given characteristic. The implicit solution for u is

$$u(n, m) = A_0 \exp\left(-\varsigma(m - m_0 - upn)^2\right). \tag{40}$$

One can rewrite the solution as an explicit function for m,

$$m = upn + m_0 \pm \sqrt{\frac{-\ln(u/A_0)}{\varsigma}}. \tag{41}$$

A plot of u versus m for various n evaluated using (41) is plotted in Figure 2. The solution shows the wave amplitude as a function of position for various values of n up to, including and beyond, the breaking time.

The solution (Equation (40)) may become triple-valued after a period of time—this is wave breaking. The onset of wave breaking is characterised by the solution having a vertical tangent at the leading wave edge. The breaking time is the time at which this first occurs. Recall the characteristics,

$$\sqrt{\varsigma}(m - m_0) = F(\sigma)n + \sigma, \tag{42}$$

where $F(\sigma) = \sqrt{\varsigma} p A_0 e^{-\sigma^2}$. Defining $G(\sigma) = -(1/F'(\sigma))$, one can find a particular value of $\sigma = \sigma_b$ such that $G'(\sigma_b) = 0$. Using σ_b, the breaking time is simply

$$\begin{aligned} n_b &= G(\sigma_b), \\ &= \frac{1}{pA_0}\sqrt{\frac{e}{2\varsigma}}. \end{aligned} \tag{43}$$

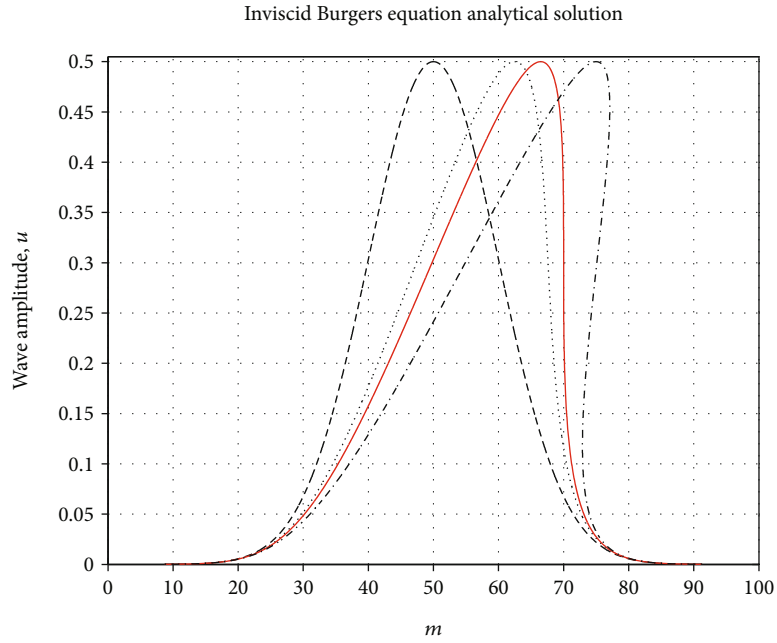

Figure 2: Plot of the analytical solution (41) of the inviscid Burgers' equation for $n = 0$ (dashed line), $n = 500$ (dotted line), $n = n_b = 659.5$ (solid line), and $n = 1000$ (dash-dot line).

Given the breaking time n_b, one can calculate the corresponding breaking location, m_b, by solving the characteristics for m with $\sigma = \sigma_b$ and $n = n_b$,

$$\sqrt{\varsigma}(m_b - m_0) = F(\sigma_b)n_b + \sigma_b. \tag{44}$$

Therefore, the breaking location

$$m_b = \sqrt{\frac{2}{\varsigma}} + m_0. \tag{45}$$

For the particular problem described in this manuscript, parameterized by the values in Table 1, the breaking time and location are $n_b = 659.5$ and $m_b = 70$, respectively (illustrated by the solid curve in Figure 2). In comparison, for the numerical solutions, the wave will break earlier ($n < n_b$) and at a premature location ($m < m_b$), which is to be expected since the numerical grid resolution will be defeated long before the leading wave edge approaches an infinite gradient.

2.5. Evaluating the Numerical Stability. To compare and quantify the development of the instability, we define the fractional change of the numerical solution relative to the analytical solution,

$$\delta = \frac{\left|u_0 - u_{m_{bs}}^{n_{bs}}\right|}{u_0}, \tag{46}$$

where $u_0 = u(n = n_{bs}, m = m_{bs})$ is the analytical solution (Equation (40)) evaluated at $n = n_{bs}$ and $m = m_{bs}$, the breaking time and breaking location of algorithm s, respectively, and $u_{m_{bs}}^{n_{bs}}$ is the numerical solution of algorithm s, evaluated at $n = n_{bs}$ and $m = m_{bs}$. The earlier the instability evolves for a particular algorithm, the smaller the value of n_{bs} and the poorer the algorithm performs. The bigger the value of δ, the poorer the stability of the algorithm. Note that the definition of δ is consistent with the quantity, Z_m^n, defined by Mitchell and Griffiths [22], to determine the stability of a finite difference numerical algorithm.

To determine the breaking time, n_{bs}, and breaking location, m_{bs}, for algorithm s, for each time step, $n \in [0, n_{\max}]$, the maximum value of the wave amplitude, $(u_m^n)_{\max}$, is determined for $m \in [0, m_{\max}]$. Using this, the absolute magnitude of the difference between the maximum amplitude and the amplitude of the initial perturbation, A_0, is calculated: $|A_0 - (u_m^n)_{\max}|$. In the first instance that this quantity is greater than or equal to a threshold value, α_0, then the corresponding values of m and n are the breaking location, m_{bs}, and breaking time, n_{bs}, of algorithm s, respectively. In this paper, the threshold value, α_0, is set to be 1% of the amplitude of the initial perturbation, $\alpha_0 = 0.01A_0 = 5 \times 10^{-3}$.

3. Results and Discussion

In the discussion that follows, LW1, LW2, LW3, LW4, FDTD1, and FDTD2 denote numerical solutions of (1) obtained from the formulae (13), (16), (21), (24), (31), and (33), respectively. Figure 1 shows the evolution of the nonlinear wave equation solved using FDTD1. The plot encapsulates qualitatively the typical behaviour of the nonlinear wave equation as a function of position and time. The initial, perturbed wave-form propagates in the positive x-direction with a speed proportional to its amplitude. Therefore, small amplitude wave components propagate slower than their larger counterparts, leading to wave steepening. In general,

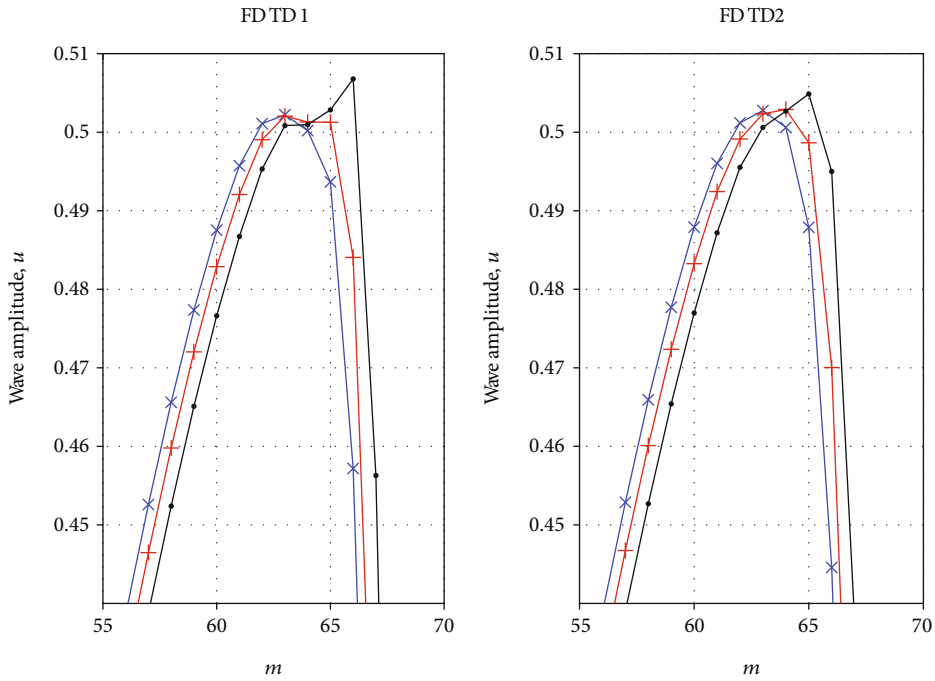

Figure 3: (a) Solution of (1) obtained from the FDTD algorithm (FDTD1: Equation (31)), displaying the spatial structure, $m \in [55, 70]$, for time steps $n = 510$ (cross), $n = 530$ (plus sign), and $n = 555$ (point) exhibiting the evolution of the nonlinear wave and the advective instability. (b) Solution of (1) obtained from the conservative FDTD algorithm (FDTD2: Equation (33)), displaying the spatial structure for the same time steps. The plots illustrate the formation of the initial spike, as a result of the instability that disrupts the onward solution. FDTD1 and FDTD2 span the range of δ calculated in this paper.

as fast moving wave elements overtake those moving slower, the wave solution becomes multivalued for a single value of x. Within the numerical context discussed here, multivalued solutions are not possible and the numerical schemes become unstable. The ensuing nonlinear instability manifests itself as a series of oscillating spikes that form in the wake of the leading wave edge. As the leading wave edge tries to overtake further wave elements, more spikes are formed in its wake. Figure 3 exhibits the initial development of the advective instability using FDTD1 and FDTD2 as example cases. The plot shows a close-up of the wave crest for time steps $n = 510$ (cross), $n = 530$ (plus sign), and $n = 555$ (point), illustrating the formation of the initial spike that disrupts the solution.

To quantify the development of the instability, we calculate the fractional change of the numerical solution relative to the analytical solution, δ. For the algorithms LW1, LW2, LW3, LW4, FDTD1, and FDTD2 considered here, $\delta \approx 0.0413$, 0.0412, 0.0273, 0.0279, 0.0473, and 0.017, respectively (see Table 2), demonstrating that in order of stability from best to worst, we have FDTD2, LW3, LW4, LW2, LW1, and FDTD1. The nonconservative algorithms appear to have inferior stability in comparison to the conservative algorithms. Of the nonconservative algorithms, the FDTD1 solution has the least stability, while the two nonconservative LW methods (LW1 and LW2) are virtually identical. Of the conservative algorithms, the FDTD2 solution has the most stability, and very little separates the two LW solutions (LW3 and LW4). Although the FDTD2 algorithm appears to break before the LW algorithms (as indicated by its n_{bs} value), its comparison to the analytical solution for the same position and time indicates that it ultimately has superior stability. Note that δ can also be viewed as a measure of accuracy, since instability leads to inaccuracy and so is a dependent concept.

Table 2: Results for the fractional change of the numerical solution relative to the analytical solution for $n = n_{bs}$ and $m = m_{bs}$.

Algorithm, s	n_{bs}	m_{bs}	δ
LW1 (Equation (13))	559	66	0.0413
LW2 (Equation (16))	558	66	0.0412
LW3 (Equation (21))	577	66	0.0273
LW4 (Equation (24))	576	66	0.0279
FDTD1 (Equation (31))	553	66	0.0473
FDTD2 (Equation (33))	557	65	0.017

4. Conclusion

This paper proposes a novel and simple technique to compare the relative empirical stability of finite difference algorithms (or any grid-based scheme) by solving the inviscid Burgers' equation to analyse their respective breaking times. The proposed technique provides a quick and easy way of determining the stability of a proposed algorithm prior to

more thorough stability tests tailored to the specific system to be solved. The relative empirical stability of two numerical methods can be assessed by analysing their respective breaking times. Due to the nature of the instability, where a discontinuity forms due the solution becoming triple-valued, the method also gives an indication of how well numerical schemes deal with discontinuities such as shocks and boundary interactions. The proposed method is analogous to the A-stability method but tailored to PDEs in which the advective derivative is key.

The technique is effective at determining the relative stability of grid-based numerical algorithms. It demonstrates that the conservative schemes behave very similarly to one another as do the nonconservative schemes and that the stability of the conservative algorithms is marginally better than the nonconservative algorithms. In order of most to least stable, we have the conservative FDTD algorithm (FDTD2), the conservative Lax-Wendroff algorithms (LW3 then LW4), the nonconservative Lax-Wendroff algorithms (LW2 then LW1), and the nonconservative FDTD algorithm (FDTD1). The LW algorithms contain an inherent diffusive term that can help to dampen the instability, quenching the wave steepening for a while. In contrast, the FDTD algorithms lack a diffusive term to help counteract the inevitable instability. In spite of this, the FDTD algorithms do well to sustain coherent behaviour. In principle, one could add a stabilizing diffusive term:

$$\partial_t u + u\partial_x u - \nu\partial_x^2 u = 0. \tag{47}$$

This is Burgers' equation where ν is the viscosity coefficient. If the diffusive term quenches the wave steepening, a stable solitary wave structure persists. Alternatively, the time-fractional inviscid Burgers' equation is worth investigating: the inclusion of a fractional time derivative may stave off the breaking time for longer, potentially improving stability [38]. Fractional-order differential equations (FDEs) have gained importance and popularity recently in their application to physical systems due to their nonlocal properties. This means that the next evolutionary state of the system depends on its historical states not just its current state. Recent investigations have analysed the time-fractional Navier-Stokes equation [39] and the SWEs [40] for bespoke systems. Many numerical schemes have been proposed for solving FDEs [41, 42] as well as for those applicable to fluid simulation, e.g., the fractional Burgers equation [43], the fractional diffusion equation [44] (relevant to the incompressible Navier-Stokes equation), and the fractional parabolic differential equations [45] (relevant to vorticity-stream function formulation of fluids).

The efficacy of a numerical algorithm depends upon a number of criteria including stability, accuracy, and ease of implementation. LW methods are, by definition, second-order accurate in space and time. In general, the choice of the ODE solver in FDTD dictates the temporal order of accuracy: here, a fourth-order Runge-Kutta method was used. However, other techniques can be used such as the leap-frog method. In this respect, FDTD has superior versatility since greater temporal accuracy can be achieved. Additionally, FDTD can be easily modified to have a higher spatial accuracy by simply using higher order finite difference approximations of the spatial derivatives used. Extending the LW method in such a way is nontrivial. In comparison to the LW method, FDTD is very easy to implement, particularly when a system of coupled differential equations is considered. For a system of equations, the LW algorithm can become hard to obtain as the Jacobian of the system becomes more involved. With this in mind, it would seem reasonable to opt for FDTD for certain problems, especially when a system of differential equations is being considered, such as for the numerical solution of the shallow water equations.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

CRS gratefully acknowledges support from the Division of Games Technology and Mathematics at Abertay University.

References

[1] J. J. Monaghan, "Smoothed particle hydrodynamics," *Annual Review of Astronomy and Astrophysics*, vol. 30, no. 1, pp. 543–574, 1992.

[2] M. Macklin and M. Müller, "Position based fluids," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 1, 2013.

[3] M. Castro, S. Ortega, M. De la Asuncion, J. M. Mantas, and J. M. Gallardo, "GPU computing for shallow water flow simulation based on finite volume schemes," *Comptes Rendus Mecanique - C R MEC*, vol. 339, no. 2, pp. 165–184, 2011.

[4] R. Vacondio, A. Pal, and P. Mignosa, "GPU-enhanced finite volume shallow water solver for fast flood simulations," *Environmental Modelling & Software*, vol. 57, pp. 60–75, 2014.

[5] G. Petaccia, F. Leporati, and E. Torti, "OpenMP and cuda simulations of Sella Zerbino dam break on unstructured grids," *Computational Geosciences*, vol. 20, no. 5, pp. 1123–1132, 2016.

[6] A. Kurganov and G. Petrova, "A second-order well-balanced positivity preserving central-upwind scheme for the saint-venant system," *Communications in Mathematical Sciences*, vol. 5, no. 1, pp. 133–160, 2007.

[7] M. Ai, A. Du, H. Xu, and J. Niu, "An improved shallow water equation model for water animation," *AIP Conference Proceedings*, vol. 1820, no. 1, article 080006, 2017.

[8] R. Bridson, *Fluid Simulation for Computer Graphics. 1em plus 0.5em minus 0.4em*, CRC Press, 2018.

[9] C. Braley and A. Sandu, *Fluid Simulation for Computer Graphics: a Tutorial in Grid Based and Particle Based Methods*, Virginia Tech, Blacksburg, 2009.

[10] T. Yang, J. Chang, B. Ren, M. Lin, J. Zhang, and S.-M. Hu, "Fast multiple-fluid simulation using Helmholtz free energy," *ACM Transactions on Graphics*, vol. 34, pp. 1–11, 2015.

[11] X. Zhang and S. Liu, "SPH haptic interaction with multiple-fluid simulation," *Virtual Reality*, vol. 21, no. 4, pp. 165–175, 2017.

[12] M. Hassaballah, A. M. Aly, and A. Abdelnaim, "Interactive fluid flow simulation in computer graphics using incompressible smoothed particle hydrodynamics," *Computer Animation and Virtual Worlds*, vol. 31, no. 2, article e1916, 2020.

[13] J. Bender, M. Müller, and M. Macklin, *A survey on position based dynamics*, Eurographics, 2017.

[14] X. Shao, E. Liao, and F. Zhang, "Improving SPH fluid simulation using position based dynamics," *IEEE Access*, vol. 5, pp. 13901–13908, 2017.

[15] C. J. Ž. dos Santos Brito, A. L. B. V.-e. Silva, M. W. S. Almeida, V. Teichrieb, and J. M. X. N. Teixeira, "Large viscoelastic fluid simulation on GPU," in *2017 16th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 134–143, Curitiba – PR – Brazil, 2017.

[16] E. Tumanov, D. Korobchenko, and N. Chentanez, "Data-driven particle-based liquid simulation with deep learning utilizing sub-pixel convolution," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 4, no. 1, 2021.

[17] A. Mohamad, *Lattice Boltzmann Method: Fundamentals and Engineering Applications with Computer Codes*, Springer, London, 2019.

[18] J. Zhou, "Lattice Boltzmann model for the shallow water equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 191, pp. 3527–3539, 2002.

[19] Z. Ru, H. Liu, G. Tu et al., "A lattice Boltzmann model for shallow water equations on the smoothed quadtree grid," *International Journal for Numerical Methods in Fluids*, vol. 94, no. 4, pp. 295–315, 2022.

[20] Z. Fengquan, Q. Wei, and Z. Wu, "A coupled grid-particle method for fluid animation on GPU," *Wireless Communications and Mobile Computing*, vol. 2020, Article ID 8865931, 13 pages, 2020.

[21] P. Parna, *Shallow water equations in real-time computer graphics, [Ph.D. thesis]*, Abertay University, United Kingdom, Jan, 2020.

[22] A. Mitchell and D. Griffiths, *The Finite Difference Method in Partial Differential Equations*, Wiley & Sons, Incorporated, 1980.

[23] R. LeVeque, *Numerical Methods for Conservation Laws*, Lectures in mathematics ETH Zürich, Birkhäuser Verlag, 1990.

[24] R. Liska and B. Wendroff, "Composite schemes for conservation laws," *SIAM Journal on Numerical Analysis*, vol. 35, no. 6, pp. 2250–2271, 1998.

[25] R. Liska and B. Wendroff, "Two-dimensional shallow water equations by composite schemes," *International Journal for Numerical Methods in Fluids*, vol. 30, no. 4, pp. 461–479, 1999.

[26] P. Parna, K. Meyer, and R. Falconer, "GPU driven finite difference WENO scheme for real time solution of the shallow water equations," *Computers & Fluids*, vol. 161, pp. 107–120, 2018.

[27] W.-Y. Liang, T.-J. Hsieh, M. T. Satria et al., "A GPU-based simulation of tsunami propagation and inundation," in *International Conference on Algorithms and Architectures for Parallel Processing*, pp. 593–603, Berlin, Heidelberg, 2009.

[28] T.-J. Hsieh, W.-Y. Liang, Y.-L. Chang, M. T. Satria, and B. Huang, "Parallel tsunami simulation and visualization on tiled display wall using OpenGL shading language," *Journal of the Chinese Institute of Engineers*, vol. 36, pp. 1–10, 2012.

[29] O. Ransom and B. Younis, "Explicit gpu based second-order finite-difference modeling on a high resolution surface, Feather River, California," *Water Resources Management*, vol. 30, no. 1, pp. 261–277, 2016.

[30] P.-W. Li, C.-M. Fan, and J. Grabski, "A meshless generalized finite difference method for solving shallow water equations with the flux limiter technique," *Engineering Analysis with Boundary Elements*, vol. 131, pp. 159–173, 2021.

[31] X. Li, G. Li, and Y. Ge, "A new fifth-order finite difference weno scheme for dam-break simulations," *Advances in Applied Mathematics and Mechanics*, vol. 13, no. 1, pp. 58–82, 2020.

[32] A. Prieto-Arranz, L. Ramirez, I. Couceiro, I. Colominas, and X. Nogueira, "A well-balanced SPH-ALE scheme for shallow water applications," *Journal of Scientific Computing*, vol. 88, no. 3, p. SEP, 2021.

[33] X. Wu, E. J. Kubatko, and J. Chan, "High-order entropy stable discontinuous Galerkin methods for the shallow water equations: curved triangular meshes and GPU acceleration," *Computers & Mathematics with Applications*, vol. 82, pp. 179–199, 2021.

[34] C. R. Stark, *Plasma processes in pulsar environments, [Ph.D. thesis]*, University of Glasgow, United Kingdom, 2008.

[35] K. Yee, "Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media," *IEEE Transactions on Antennas and Propagation*, vol. 14, no. 3, pp. 302–307, 1966.

[36] A. Taflove and M. Brodwin, "Numerical solution of steady-state electromagnetic scattering problems using the time-dependent Maxwell's equations," *IRE Transactions on Microwave Theory and Techniques*, vol. 23, no. 8, pp. 623–630, 1975.

[37] A. Taflove and S. Hagness, *Computational Electrodynamics: the Finite-Difference Time-Domain Method*, Artech House, 2nd edition, 2000.

[38] G. Coclite, S. Dipierro, F. Maddalena, and E. Valdinoci, "Singularity formation in fractional Burgers' equations," *Journal of Nonlinear Science*, vol. 30, no. 4, pp. 1285–1305, 2020.

[39] D. Kumar, J. Singh, and D. S. Kumar, "A fractional model of Navier-Stokes equation arising in unsteady flow of a viscous fluid," *Journal of the Association of Arab Universities for Basic and Applied Sciences*, vol. 17, p. 2, 2015.

[40] D. S. Kumar, A. Kumar, Z. Odibat, M. Aldhaifallah, and K. Nisar, "A comparison study of two modified analytical approach for the solution of nonlinear fractional shallow water equations in fluid flow," *AIMS Mathematics*, vol. 5, pp. 3035–3055, 2020.

[41] N. Zabidi, Z. Abdul Majid, A. Kilicman, and F. Rabiei, "Numerical solutions of fractional differential equations by using fractional explicit Adams method," *Mathematics*, vol. 11, 2020.

[42] F. Alharbi, A. Zidan, M. Naeem, R. Shah, and K. Nonlaopon, "Numerical investigation of fractional-order differential equations via -Haar-wavelet method," *Journal of Function Spaces*, vol. 2021, Article ID 3084110, 14 pages, 2021.

[43] D. Vieru, C. Fetecau, N. A. Shah, and J. Chung, "Numerical approaches of the generalized time-fractional Burgers

equation with time-variable coefficients," *Journal of Function Spaces*, vol. 2021, Article ID 8803182, 14 pages, 2021.

[44] F. Ghaffar, S. Ullah, N. Badshah, and N. Khan, "A higher-order unconditionally stable scheme for the solution of fractional diffusion equation," *Mathematical Methods in the Applied Sciences*, vol. 44, no. 4, pp. 3004–3022, 2021.

[45] S. Ullah, S. Zulfiqar, A. Buhader, and N. Khan, "Analysis of Caputo-Fabrizio fractional order semi-linear parabolic equations via effective amalgamated technique," *Physica Scripta*, vol. 96, article 035214, 2021.