CRWR Online Report 08-04

# A Time-Centered Split for Implicit Discretization of Unsteady Advection Problems

**by**

**Shipeng Fu, B.S.; M.S.**

**and**

**Ben R. Hodges, Ph.D.**

The University of Texas at Austin
May 13, 2008
This document is available online via World Wide Web at
http://www.crwr.utexas.edu/online.html

## Acknowledgements

I would like to begin by expressing my gratitude to my advisor Dr. Ben Hodges. I am thankful for his guidance and support, and I appreciate the freedom he has given me to explore and pursue my own research interests. I would also like to thank Dr. Graham Carey, Dr. Randall Charbeneau, Dr. Spyros Kinnas and Dr. Daene Mckinney for serving on my committee.

The faculty members, facilities, and the world class program of EWRE have made my research possible. I am grateful to Dr. Kinnas and Dr. Carey for their input in the development of the numerical methods in this dissertation. I have gone to Dr. Liljestrand, Dr. Lawler and Dr. Kinnas many times for help and advice, and they have always been happy to help. I want to thank them for their encouragement and advice. I would like to thank Dr. Charbeneau for giving me the opportunity to work with lab experiments (so I could finally play with the real water!). I would like to thank Sharon Bernard, Marcy Betak, and Susan Swanson-Cartwright at the CRWR Office for helping with all my administrative paperwork and questions.

During my Ph.D study at UT, my friends in Austin have been extremely helpful and supportive. I could not finish this without their friendship. I want to particularly thank Paula Kulis, Li-Jung Chen and their wonderful husbands for feeding me meals and listening to my whines. I also want to thank Paula for many discussions in fluid mechanics and beyond. I want to thank Yihsiang Yu and Vimal Vinayan for their insightful discussions on numerical methods and wave mechanics. I want to thank Becky Teasley and John Allen, Rebekah and Nate Johnson, and Jeremy Seibert for graciously reading and providing valuable feedbacks on my dissertation. I want to thank all my friends in EWRE and CRWR. Our lunch topic at CRWR is always intellectually stimulating, and sometimes wild. I also want to thank all the members in the BUSTED (Best Underpaid Students Turning into Environmental Doctors) club. I really benefited from the support from everyone. I will miss you girls. I want to thank my running buddy, Susan De Long, for many interesting discussions about Yoga and philosophy.

I am thankful to my parents. I thank them for teaching me good values, and encouraging me to think independently since I was very young. My diligent father and curious mother have always been my role models and inspiration.

My deepest thanks go to my best friend and husband, Yunzhi Yang. I am really fortunate to have such a wonderful life partner accompanying me for this journey. I want to thank him for truly understanding me, encouraging me to strive for excellence, and bringing life into perspective.

# Abstract

Environmental flows (e.g. river and atmospheric flows) governed by the shallow water equations (SWE) are usually dominated by the advective mechanism over multiple time-scales. The combination of time dependency and nonlinear advection creates difficulties in the numerical solution of the SWE. A fully-implicit scheme is desirable because a relatively large time step may be used in a simulation. However, nonlinearity in a fully implicit method results in a system of nonlinear equations to be solved at each time step. To address this difficulty, a new method for implicit solution of unsteady nonlinear advection equations is developed in this research. This Time-Centered Split (TCS) method uses a nested application of the midpoint rule to computationally decouple advection terms in a temporally second-order accurate time-marching discretization. The method requires solution of only two sets of linear equations without an outer iteration, and is theoretically applicable to quadratically-nonlinear coupled equations for any number of variables.

To explore its characteristics, the TCS algorithm is first applied to one-dimensional problems and compared to the conventional nonlinear solution methods. The temporal accuracy and practical stability of the method is confirmed using these 1D examples. It is shown that TCS can computationally linearize unsteady nonlinear advection problems without either 1) outer iteration or 2) calculation of the Jacobian. A family of the TCS method is created in one general form by introducing weighting factors to different terms. We prove both analytically and by examples that the value of the weighting factors does not affect the order of accuracy of the scheme. In addition, the TCS method can not only computationally linearize but also decouple an equation system of coupled variables using special combinations of weighting factors. Hence, the TCS method provides flexibilities and efficiency in applications.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1 Introduction

Models of river flow using mathematic tools have been developed since last half century (Cao and Carling 2002). Various numerical river models have been created to study flood control (Dutta et al. 2007; Liao et al. 2007), water allocation (Fleckenstein et al. 2006; Luo 2007), sediment and morphological evolution (Giri and Shimizu 2006; Le et al. 2006) and so on. The river flow is usually modeled at a coarse-grid scale due to the limitations of the computational power (Dedong et al. 2007; Shaw et al. 2005; Thouvenin et al. 2007). However, there exist some particular river systems such as a river with many large woody debris (LWD), as shown in Figure 1.1. The flow information around LWD is at subgrid-scale, which can't satisfactorily be modeled at the conventional coarse-grid.

In addition, the river flow is often simulated using two-dimensional (2D) depth averaged shallow water equations (SWE) as the governing equations (Chen and Peng 2006; Nguyen et al. 2006; Weerakoon et al. 2003). However, the unsteadiness and the dominant advective mechanism of a river flow create difficulties in SWE based numerical simulation because of the combination of the "stiffness" (i.e. the unsteady term) and "nonlinearity" (i.e. the nonlinear advection term) in the momentum equations. Therefore, in this research, we develop a new numerical algorithm to address the difficulties in simulating the unsteady nonlinear advection. This numerical scheme can also serve as a platform to perform the new conceptual model which can integrate the subgrid-scale physics into a coarse-grid scale model. The detailed development of the conceptual model can be found in Fu and Hodges (2005). First, let us start with the background information of the LWD.

## 1.1 BACKGROUND

Large woody debris (LWD) refers to woody material such as fallen tree trunks or root balls that become lodged in stream channels. Figure 1.1 gives an example of a river laden with large woody debris (LWD). The conventional size of LWD is diameters larger than 0.1 m and lengths greater than 1.0 m (Keller and Swanson 1979; Andrus et al. 1988). LWD accumulations have been historically called "snags", a pejorative term reflecting a perception of LWD as a nuisance to river navigation and efficient water use. River "improvement" schemes typically involve removing LWD (Shields & Nunnaly 1984; Gippel 1995), to improve water conveyance, rejuvenate channels, lessen the risk of damage to bridges, improve recreational amenity, and remove barriers to fish migration (Harmon et al. 1986). However, from an ecological perspective, LWD provides a stable substratum for microorganisms, algae and invertebrates (Minshall 1984; Brown & May 2000; Statzner & Higler 1986). This, in turn, creates grater local productivity for higher trophic levels such as invertebrates and fish (Price & Lovett 2002). Furthermore, LWD enhances hydraulic diversity (i.e. a wide range of flow conditions), which in turn enhances diversity in fish habitat, e.g. providing low-velocity refuges sought by many fish species (Benke et al. 1985; Matthews & Hill 1980), and also providing cover from predators (Angermeier & Karr 1984, Everett & Ruiz, 1993), foraging habitat, and

spawning substratum - all of which vary dynamically with flow rate (Bao & Mathews 1991; Mathews & Tallent 1997; Marzolf 1978). In short, LWD creates greater habitat complexity (O'Conner 1991), which should produce greater biodiversity (Gerhard & Reich 2000).

Figure 1.1 Sulphur River (Texas) large woody debris at low-flow conditions

(Photo courtesy of Texas Water Development Board)

Over the past two decades, a wide variety of studies demonstrate how woody debris may be a dominant player in aquatic habitat. Angermeier & Karr (1984) selectively modified a stream by inducing and removing woody debris, subsequently showing a greater abundance of fish and benthic invertebrates correlated with the introduced debris. Benke et al. (1985) examined a low-gradient stream, showing that only 4% of habitat surface was in LWD and yet this supported 60% of the invertebrate biomass and 16% of overall production. Similar results are noted in Benke et al. 1984 and Jacobi & Benke1991. Effects of flooding on fish movements and distribution (Hill & Grossman 1987; Harvey et al. 1999) have confirmed the importance of habitat complexity (including LWD) for population persistence. More subtle and indirect effects on habitat have been attributed to LWD's modification of the flow field and water level (Triska & Cromack 1980; Sedell et al. 1982; Wallace & Benke 1984). Near LWD, reduced velocities allow retention of organic matter, building up ecologically-important debris dams, while water lever effects at high flow rates can influence the seed dispersal from riparian vegetation (Merritt & Wohl 2002). Studies drawing similar conclusions have been conducted over a variety of locales and river types, ranging from salmon spawning streams in Alaska (Dolloff 1986), low-gradient rivers in West Virginia (Lobb & Orth 1991) to an ephemeral river in Africa (Jacobson et al. 1999). The common theme is that LWD plays a varied and critical role in aquatic habitat for rivers and streams – a role determined by the flow field near LWD. Moreover, water management agencies are

interested in including the effects of LWD in aquatic habitat assessments used for water resource allocations.

The flow structure around LWD is very complicated and varied, which has been revealed by several previous field studies (e.g. Mutz 2000; Beebe 2001; Daniels & Rhoads 2003). Moreover, there is a large difference in scales between hydraulic processes and the critical ecological processes. Because of this scale difference, the small-scale flow heterogeneity (important to biota) is poorly represented in the channel-scale numerical models currently used for aquatic habitat analysis (e.g. Waddle 2001). To address this problem, a new conceptual model is proposed to integrate the subgrid-scale physics into a coarse-grid scale model (Fu and Hodges, 2005).

### 1.2 CHALLENGE AND MOTIVATION OF THE NUMERICAL MODEL

In addition to the scale difference, another challenge for solving the river flow with LWD is to build a numerical model to simulate the unsteady river flow. River flow is usually characterized as shallow-water flow. As shown in Figure 1.1, the flow around LWD is indeed shallow as most of the LWD are exposed out of the water surface. This kind of river system is usually modeled by 2D depth averaged shallow water equations (SWE) (Arega et al. 2008; Burguete et al. 2006; Gejadze and Monnier 2007). 2D depth averaged SWE can be obtained by integrating the Navier-Stokes equation over the water depth with the underlying hydrostatic assumption. One of the difficulties in solving SWE numerically is the combination of the "stiffness" (i.e. the unsteady term) and "nonlinearity" (i.e. the nonlinear advection term) in the momentum equations. Numerous finite-difference numerical models for solving SWE have been developed over the past several decades. Among these developed numerical models, explicit and semi-implicit schemes are the most widely used. For example, in the atmospheric or weather research community, explicit methods such as Leapfrog and Lax-Wendroff-type are the most popular ones (Mendez-Nunez and Carroll 1993). Casulli (Casulli 1990) proposed a semi-implicit scheme, on which a number of variations are based. A common property for both the explicit and semi-implicit methods is that they explicitly discretize the advection term. Hence, the nonlinearity does not arise in an explicit or a semi-implicit sachem. However, their stability is restricted by the Courant-Friedrichs-Lewy (CFL) condition because of the explicitness. A fully implicit scheme is desirable because we can use a relative large time step in the simulation. A fully implicit solution for an unsteady nonlinear advection problem involves solving a system of nonlinear equation at each step. The conventional techniques for solving a system of nonlinear equations require either 1) iterations of root-finding procedure or/and 2) calculations of the Jacobian. These two processes make a fully implicit solution more complex and computationally expensive. These motivate this research to develop a new implicit scheme to solve unsteady nonlinear advection problems.

## 1.3 OBJECTIVE

The primary objective of this research is:

Develop a new implicit solution for unsteady nonlinear advection problems. The new numerical algorithm should have the advantage in both stability and efficiency while keeping the 2nd-order temporal accuracy as in most existing SWE models. In addition, this new method would provide a novel approach in decoupling a coupled equation system.

## 1.4 APPROACH

To achieve the objective of this research, several steps need to be taken to guide this study. The major steps are as follows:

*Literature review*

- Review current finite-difference models for the 2D depth averaged SWE.
- Examine how the nonlinear advection term is treated in different temporal discretizations.
- Review the merits and limitations of the existing models.

This part of the dissertation work is presented in Chapter 2.

*Analytical development of the new numerical algorithm*

Illustrate the theoretical principle and concept of the new algorithm. This part of the dissertation work is presented in Chapter 3.

*Verify the new method using 1D test case*

- Test the new algorithm using a 1D PDE with time dependent nonlinear advection term.
- Test the new algorithm using 1D ordinary differential equation (ODE).
- Explore the characteristics of the method in accuracy, stability and efficiency.

This part of the dissertation work can be found in Chapter 4.

*Verify the new method using a multi-variable and multi-dimensional problem*

- Theoretically develop the new algorithm in multi-variable and multi-dimensional problems.
- Demonstrate and analyze the decoupling process of the new method.
- Conduct initial test cases of the numerical algorithm in 2D depth averaged SWE.

This work can be found in Chapter 5 and Chapter 6.

## 1.5 BROAD APPLICATION

The initial motivation and application for this research is simulating a river flow with LWD. The newly developed numerical algorithm for solving unsteady nonlinear advection problems is not limited only in simulating river flows. In most of the environmental flows such as atmospheric flow, ocean circulation, storm surge and etc., advection is the dominant mechanism and we are often interested in simulating these flows with a range of different time scales. Therefore, the difficulties arising from the combination of the unsteadiness and nonlinear advection exist in modeling many types of environmental flows. This new numerical method provides a new approach to address this difficulty in these and many other research areas.

# Chapter 2 Review of Finite Difference Schemes for Solving the Unsteady Nonlinear Advection in the Shallow Water Equations

Environmental flows (e.g. river and atmospheric flows) governed by the SWE are usually dominated by the advective mechanism over multiple time-scales (Vreugdenhil 1994). The combination of time dependency and nonlinear advection creates difficulties in the numerical solution of the shallow water equations (SWE) (Bourchtein and Bourchtein 2006). To address this difficulty, numerous numerical schemes have been developed over the past several decades. Iskandarani et al (2005) reviewed the finite element/finite volume methods in solving advection equations. More reviews about finite element methods can be found in the article by Thomee (2001). The research herein focuses on developing a finite difference algorithm for solving unsteady nonlinear advection problems. To review the existing finite difference techniques, we first classify time-marching algorithms into three categories: explicit, semi-implicit and implicit methods. In the explicit and semi-implicit methods, the nonlinear advection terms are treated explicitly. Hence, the issue of solving a nonlinear matrix inversion does not arise in these two temporal discretizations. However, to solve a fully-implicit system, different computational linearization methods are required treat the nonlinearity, so that the inversion of a nonlinear matrix can be obtained. Therefore, we briefly review the explicit and semi-implicit models for solving the SWE and examine different linearization approaches in the implicit system. Comparisons among different existing methods are summarized in the last section of this chapter.

## 2.1 EXPLICIT METHOD

Fully explicit methods are popular in atmospheric or weather research community. The most widely used explicit methods in atmospheric research are Leapfrog and Lax-wendroff-type schemes (Mendez-Nunez and Carroll 1993). The Leapfrog method is a one-step method, requiring two time levels of known values to compute an unknown level. Modifications of the Leapfrog methods are used in simulating ocean circulation (Cho and Yoon 1998; Fujima and Shigemura 2000). The Lax-Wendroff-type method is a two-step Predictor-Corrector method. One of the popular variations of the Lax-wendroff scheme is developed by MacCormack (1969). The MacCormack scheme is extended to many research areas such as aerospace engineering (Hirose et al. 1991; Kurbatskii and Mankbadi 2004) and hydrological science in simulating free surface flow (Fennema and Chaudhry 1990; Garcia and Kahawita 1986; Kazezyılmaz-Alhana et al. 2005; Keshari and Koo 2007; Li and Jackson 2007), groundwater flow (Keshari and Koo 2007) and dam-break shock wave (Li and Jackson 2007). Another commonly used explicit Predictor-Corrector method is the Runge-Kutta method (Delis and Katsaounis 2005; Zhou et al. 2007). The Leapfrog and the two-step Predictor-Corrector methods can maintain $2^{nd}$-order temporal accuracy. Although simpler explicit schemes such as

forward-time scheme is also used in solving SWE (Murillo et al. 2008), it can only maintain 1st-order temporal accuracy.

Explicit schemes are straightforward and easy to implement. Nonlinear advective terms in partial differential equations (PDE) are readily-computed from known time-levels using fully-explicit time-marching methods. However, the implementations of explicit methods are restricted by stability requirements. Runge-Kutta and MacCormack methods typically require an excessively small time step to satisfy the Courant-Friedrichs-Lewy (CFL) condition(Mendez-Nunez and Carroll 1993); the Leapfrog approach must be applied with appropriate numerical strategies such as alternate time levels of the solution (Agoshkov et al. 1994; Peyret and Taylor 1983; Zhou 2002) or sophisticated mode-splitting (e.g.Blumberg and Mellor 1987).

## 2.2 SEMI-IMPLICIT METHOD

Semi-implicit methods have been developed and widely used in temporal discretized SWE. In most semi-implicit methods, the nonlinear advection terms are discretized explicitly (Bonaventura and Rosatti 2002; Bourchtein and Bourchtein 2007; Casulli 1990; Casulli and Cattani 1994; Casulli and Cheng 1992; Kar 2006; Spitaleri and Corinaldesi 1997) so that the issue of solving a nonlinear matrix inversion does not arise. For instance, Environmental Fluid Dynamics Computer code (EFDC) uses a three-level semi-implicit method, in which advection terms are explicitly discretized using upwind scheme. As a result, the stability is constrained by the explicit advection terms (Hamrick 1992). Some other models (Kar 2006) use the explicit Runge-Kutta method to calculate the advection term. It is obvious that the stability of this treatment is constrained by the CFL number.

To overcome the stability constraint, different numerical treatments are introduced to calculate the linearized advection term. One of the most widely used techniques is the semi-Lagrangian method (Cullen 2001; Rosatti et al. 2005). Casulli (Casulli 1990; Casulli and Cattani 1994; Casulli and Cheng 1992) used this method in UnTRIM and the earlier version, TRIM. Some latest models including ELCIRC (Zhang et al. 2004) and ELCOM(Hodges 2000) followed this idea. Although Casulli's approach has been successful and stable, its accuracy is relatively poor (Hodges, 2004).

The main advantage of the semi-Lagrangian method is that the stability is not constrained by the CFL condition (Barros and Garcia 2007). However, a set of trajectory equations need to be solved at each time step and the maximum allowable time step is restricted by the convergence criteria of the iteration of the trajectory equations (Bourchtein and Bourchtein 2007).

## 2.3 IMPLICIT METHOD

An obvious advantage of a fully-implicit method is that the time step is not restricted by the CFL number. Various fully-implicit finite difference schemes have been developed. Steppeler (2006) solved a fully implicit, SWE based meteorological model using the Fourier transformation method at each grid point and each time step. Some

researchers derived a fully-implicit scheme based on Alternating Direction Implicit (ADI) method (Szymkiewicz 1992; Wilders et al. 1988). Yuan and Wu solved implicit Navier-Stokes equations using a staggered finite difference Crank-Nicolson scheme (Yuan and Wu 2004). Burguete and Garcia-Navarro (Burguete and Garcia-Navarro 2004) implemented a first order upwind implicit scheme to simulate river hydraulics problem.

Although the fully-implicit methods are more accurate and robust, they are less common, possibly due to computational complexity and expense (Turek 1996). Nonlinearity in a fully-implicit method results in a system of nonlinear algebraic equations to be solved at each time step. Existing strategies for solving implicit nonlinear equations include iterative and non-iterative methods (Moin 2001). In the following, we will review the conventional techniques for implicit nonlinear solutions.

The Newton method and Picard method (Ferziger and Peric 1999; Lehmann and Ackerer 1998; Paniconi et al. 1991) are the most widely used two-level iterative techniques for implicit time-marching of nonlinear equations. For steady-state nonlinear problems, Newton and Picard iterative algorithms have proven quite successful (Paniconi and Putti 1994). These methods may be thought of as successive linear solutions of $\tilde{\mathbf{A}}\mathbf{x} = \mathbf{b}$, where $\tilde{\mathbf{A}}$ is an approximation of $\mathbf{A}$ that is iteratively refined to solve the nonlinear problem. However, for time-evolving CFD problems, each time-marching step using the Newton or Picard method requires an outer iteration applied over an inner solution of a linear equation set. When the inner problem also requires an iterative solution (as is common in CFD), the time march for this doubly-iterative approach is computationally expensive. The principal differences between the Picard and Newton methods are that the former is easier to implement and requires fewer computations per outer iteration but has only 1st-order convergence, whereas the latter is more difficult to implement but provides 2nd-order convergence. Thus, which method is appropriate depends upon the difficulty in implementing the Newton method compared to the slower convergence of the Picard method. A further implicit technique, local linearization, has not been widely used in CFD but does provide time-marching with 2nd-order accuracy (Lomax et al. 1999) without an outer iteration. By providing a single linear approximation of the nonlinear problem, local linearization side-steps the convergence issue of the outer iteration for Newton/Picard techniques. However, local linearization requires discretization of the Jacobian, which is often difficult to derive and implement for typical CFD applications.

To better illustrate these existing implicit linearization methods, we use a simple scalar ODE with a quadratic nonlinearity as an example. The nonlinear ODE is written as

$$\frac{d\psi}{dt} = f(\psi, t) \tag{2.1}$$

Using a finite-difference Crank-Nicolson discretization (the simplest 2nd-order implicit method), the above can be approximated as

$$\psi^{n+1} = \psi^{n} + \frac{\Delta t}{2}\left\{f\left(\psi^{n+1}\right) + f\left(\psi^{n}\right)\right\} + O\left(\Delta t^{3}\right) \tag{2.2}$$

where superscripts indicate the discrete time step and $f\left(\psi^{n+1}\right)$ implies a quadratic nonlinear relationship in $\psi^{n+1}$ (e.g. $\psi^{n+1}\,\psi^{n+1}$ or $\psi_x^{n+1}\,\psi^{n+1}$). Equation (2.2) is our example of an implicit nonlinear equation. In the following, we will demonstrate the principles of existing techniques to linearize Equation (2.2) computationally.

### 2.3.1 Newton methods

The Newton method is one kind of root-finding algorithm. It starts with an initial guess and iteratively estimates the root of the function. The Newton method can be derived from a Taylor series expansion or a geometric proof (Amat et al. 2003). One of the advantages of the Newton method is that it can be applied to equation systems with complex nonlinearities and keeps the quadratic convergence rate. In petroleum engineering and other research areas concerning flow through porous media, the Newton method is a standard approach because of the complex nonlinearities (Cao and Sun 2005; Dettmer and Peric 2007; Kwok and Tchelepi 2007). Using the Newton method, the linearized equation system of Equation (2.2) can be written as

$$\left(\psi^{n+1}\right)^{k+1} = \left(\psi^{n+1}\right)^{k} - \frac{\left[g\left(\psi^{n+1}\right)\right]^{k}}{\left(\frac{\partial g}{\partial \psi^{n+1}}\right)^{k}} \tag{2.3}$$

where the outer superscript indicates the iteration number and

$$g\left(\psi^{n+1}\right) = \psi^{n+1} - \psi^{n} - \frac{\Delta t}{2}\left\{f\left(\psi^{n}\right) + f\left(\psi^{n+1}\right)\right\} \tag{2.4}$$

In general, we can use the value at the previous time step as the value at the first iteration, which means

$$\left(\psi^{n+1}\right)^{1} = \psi^{n} \tag{2.5}$$

Equation (2.3) is solved repeatedly until the difference between two successive iterations satisfies the pre-defined convergence criteria. In Equation (2.3), a first order derivative $\left(\partial g / \partial \psi^{n+1}\right)$ must be calculated and updated at each iteration. If $\psi$ is a vector and discretized in space, the resulting $\left(\partial g / \partial \psi^{n+1}\right)$ is a matrix and called the Jacobian of the linearized equation system. However, the Jacobian matrix may be computationally expensive and difficult to calculate analytically (Ferziger and Peric 1999; Niet et al. 2007).

A number of approaches have been proposed to modify Newton method. For example, it can be modified with a Chebyshev approximation to accelerate the convergence (Bagatur 2007). To simplify the Newton method, much research has been carried out to simplify the calculation of the Jacobian matrix. Niet et al. (2007) evaluated the Jacobian matrix using a partial-analytical and partial-numerical technique in an ocean-climate model. Li (1993) attempted to reduce the effort to compute the derivatives.

A Jacobian-free Newton-Krylov (JFNK) method has been developed recently to solve nonlinear equation systems (Brown and Saad 1990; Chan and Jackson 1984; Knoll and Keyes 2004; Mousseau et al. 2002; Reisner et al. 2005; Wubs et al. 2006). The key to JFNK solver is approximating the Jacobian-vector product iteratively instead of evaluating each element of the Jacobian matrix. Although all these modifications reduce the computation effort of Jacobian, inconvenient iteration still remains in the Newton method.

### 2.3.2 Picard Iteration

The Picard iteration is more straightforward than the Newton method. In the Picard iteration, the previous outer iteration value, $\left(\psi^{n+1}\right)^{k}$, is substituted for $\psi^{n+1}$ on the RHS of Equation (2.2), resulting in

$$\left(\psi^{n+1}\right)^{k+1} = \psi^{n} + \frac{\Delta t}{2}\left\{f\left(\psi^{n}\right) + f\left[\left(\psi^{n+1}\right)^{k}\right]\right\} \tag{2.6}$$

As in the Newton iteration, Equation (2.6) is solved repeatedly until the difference between two successive iterations satisfies pre-defined convergence criteria. The same initial condition, Equation (2.5) is used to start the solution. Due to the simplicity, Picard iteration has been widely used in fully implicit nonlinear solver in Computational Fluid Dynamics (CFD)(Clement et al. 1994; Kwag 2000; Webster 2007). Different modifications of the Picard method have been reported in the literature. For example, Celia et al.(1987) proposed a modified Picard iteration that is a combination of Picard iteration and Newton iteration. However, this and other modified Picard methods remain linearly convergent (Lehmann and Ackerer 1998). A detailed comparison of the Picard and Newton methods is provided by Paniconi and Putti (1994). In general, Newton methods require more computations per iteration, but converge more rapidly (quadratic) than Picard methods (linear). However, for advection equations, the overall computational cost of Newton methods is typically greater than Picard methods because of the computational cost of the Jacobian (Ferziger and Peric 1999).

### 2.3.2 Local linearization

In contrast to the above-mentioned Newton and Picard methods, local linearization is a non-iterative method. Let's first review the derivation of the local linearization techniques, which are based on a Taylor-series expansion for $f^{n+1}$ about $t_n$ in Equation (2.2). The following derivation is based on Lomax et al.(1999),

$$f^{n+1} = f^{n} + \Delta\psi\left(\frac{\partial f}{\partial \psi}\right)^{n} + \Delta t\left(\frac{\partial f}{\partial t}\right)^{n} + O\left(\Delta t^{2}\right) \tag{2.7}$$

where $\Delta\psi = \psi^{n+1} - \psi^n$. For advection equations, f is generally not an explicit function of t, so $\Delta t\left(\partial f / \partial t\right)^n$ in Equation (2.7) is zero. Substituting Equation (2.7) into Equation (2.2) provides

$$\psi^{n+1} = \psi^n + \frac{\Delta t}{2}\left\{ f^n + \Delta\psi\left(\frac{\partial f}{\partial \psi}\right)^n + f^n \right\} + O\left(\Delta t^3\right) \tag{2.8}$$

which is equivalent to

$$\left[1 - \frac{\Delta t}{2}\left(\frac{\partial f}{\partial \psi}\right)^n\right]\Delta\psi = \Delta t f^n \tag{2.9}$$

Equation (2.9) is a linear second-order discrete form of the original nonlinear ODE and can be solved by any number of standard techniques once $\left(\partial f / \partial \psi\right)^n$ is explicitly computed. However, as in the Newton method, the evaluation of the Jacobian $\left(\partial f / \partial \psi\right)^n$ complicates the overall computation.

## 2.4 SUMMARY

SWE is widely used in simulating environmental flows. Mathematically, the stiffness and nonlinear advection increases the difficulty in the numerical solutions of SWE. Explicit methods can easily solve the nonlinearity but are restricted by the CFL stability criteria. Semi-implicit methods treat the advection term explicitly but need additional numerical treatment such as the semi-Lagrangian method to calculate advection terms for a higher CFL number. A fully implicit method is desirable but requires further computational linearization to solve a nonlinear system of equations. Existing implicit linearization techniques (including Newton, Picard and local linearization methods) require either 1) additional explicit derivative evaluations to provide an approximate linear problem, or 2) an outer iteration that converges an inner approximate linear problem. These two characteristics make implicit systems complex and expensive to solve. To address the nonlinearity in a fully implicit system, in the next chapter we propose a new computational linearization method that can linearize the nonlinear advection term without either 1) the outer iteration or 2) computation of the Jacobian.

# Chapter 3 Theoretical Development of the Time-Centered Split (TCS) Method

In Chapter 2 we reviewed the current implicit linearization methods that require either an outer iteration or a computation of the Jacobian. In this chapter, we develop a new method that is similar to local linearization in that it allows non-iterative discretization of a temporally $2^{nd}$-order approximation of the nonlinear equation set. Instead of requiring the Jacobian, the new method splits the time-marching nonlinear problem into two sets of linear problems that are solved in succession. The new method has both the computational efficiency of a non-iterative local linearization method and the implementation simplicity of a Picard iterative method. We call this the Time-Centered-Split (TCS) method.

The theory of the TCS method is first derived using a generic single variable nonlinear equation in this chapter. The TCS method can generate different discrete forms by introducing the time-centered split to different terms. This advantage is demonstrated by applying the TCS method to a 1D advection-diffusion equation (Burgers' equation). Four different TCS formats such as TCSF (split the flux term), TCSG (split the gradient term), TCSF-D (split the flux and diffusion terms) and TCSG-D (split the gradient and the diffusion terms) are derived and presented for the 1D Burgers' equation. One of the key advantages of the TCS method is that it provides non-iterative coupling between multiple variables. A 1D coupled advection diffusion equation and scalar transport equation are used as an example to illustrate this advantage. A summary of the principles of the TCS method and its advantages are provided in the last section.

## 3.1 THE THEORY OF TIME-CENTERED SPLITTING

### 3.1.1 Computational Splitting of the Nonlinear Term

The TCS method is based on a nested application of the midpoint rule (i.e. a centered-time approximation). Midpoint rule discretizations are often used for time-marching to obtain second-order temporal accuracy (Ferziger and Peric 1999). A common approach for nonlinear time-marching in meteorology and oceanography is application of the midpoint rule in an explicit formulation known as the 3-level Leapfrog method (Dubois et al. 2005; Fujima and Shigemura 2000). The explicit Leapfrog method can be written as

$$\phi^{n+1} = \phi^{n-1} + f\left(\phi^n\phi^n, \phi^n\right)2\Delta t + O(\Delta t^3) \tag{3.1}$$

where superscripts represent time levels and $\Delta t$ is the model time step. Equation (3.1) can be seen as a generic form of a nonlinear equation. The function f is a linear operator of $\phi\phi$ and $\phi$. Inside the function f, the product term $\phi\phi$ represents a generic quadratic nonlinear term. This quadratic nonlinear term in a flow and transport problem is of the

interest in this research. To develop the TCS method, we note that in the same vein as Equation (3.1), the midpoint rule can be written across only two time levels as

$$\phi^{n+1} = \phi^{n} + f\left(\phi^{n+1/2}\phi^{n+1/2}, \phi^{n+1/2}\right)\Delta t + O(\Delta t^{3}) \tag{3.2}$$

Equation (3.2) has the desirable property that the time n+1/2 information is retained only within the computation of the n to n+1 time step so that the advance is not leapfrogging over alternating data. By introducing a time-centered linear approximation of

$$\phi^{n+1/2} = \left(\phi^{n} + \phi^{n+1}\right)/2 + O\left(\Delta t^{2}\right) \tag{3.3}$$

in one part of the nonlinear product $\phi^{n+1/2}\phi^{n+1/2}$ in Equation (3.2), the discrete equation becomes computationally linear in time (i.e. no products with the same time level):

$$\phi^{n+1} = \phi^{n} + f\left\{\left[\frac{\phi^{n} + \phi^{n+1}}{2} + O\left(\Delta t^{2}\right)\right]\phi^{n+1/2}, \phi^{n+1/2}\right\}\Delta t + O(\Delta t^{3}) \tag{3.4}$$

Reorganizing Equation (3.4) provides

$$\phi^{n+1} = \phi^{n} + f\left(\phi^{n}\phi^{n+1/2}, \phi^{n+1/2}\right)\frac{\Delta t}{2} + f\left(\phi^{n+1}\phi^{n+1/2}, \phi^{n+1/2}\right)\frac{\Delta t}{2} + O(\Delta t^{3}) \tag{3.5}$$

Equation (3.5) is computationally split into two steps and an intermediate variable $\phi^{*}$ is defined as,

$$\phi^{*} = \phi^{n} + f\left(\phi^{n}\phi^{n+1/2}, \phi^{n+1/2}\right)\frac{\Delta t}{2} \tag{3.6}$$

Subtracting Equation (3.6) from Equation (3.5) provides

$$\phi^{n+1} = \phi^{*} + f\left(\phi^{n+1}\phi^{n+1/2}, \phi^{n+1/2}\right)\frac{\Delta t}{2} + O(\Delta t^{3}) \tag{3.7}$$

After introducing the above time-centered split, Equations (3.6) and (3.7) are both computationally linear. Furthermore, Equation (3.6) is similar to the implicit Euler approximation of $\phi^{n+1/2}$ written as:

$$\phi^{n+1/2} = \phi^{n} + f\left(\phi^{n+1/2}\phi^{n+1/2}, \phi^{n+1/2}\right)\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^{2} \tag{3.8}$$

Using the Taylor expansion, $\phi^{n+1/2}$ can be expanded as:

$$\phi^{n+1/2} = \phi^{n} + \phi_{t}\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^{2} \tag{3.9}$$

Substituting Equation (3.9) into the nonlinear term of Equation (3.8), provides

$$\phi^{n+1/2} = \phi^{n} + f\left\{\left[\phi^{n} + O\left(\frac{\Delta t}{2}\right)\right]\phi^{n+1/2}, \phi^{n+1/2}\right\}\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^{2} \tag{3.10}$$

Grouping the higher order terms,

$$\phi^{n+1/2} = \phi^n + f\left(\phi^n \phi^{n+1/2}, \phi^{n+1/2}\right)\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2 \tag{3.11}$$

Substituting Equation (3.6) into Equation (3.11), results in

$$\phi^{n+1/2} = \phi^* + O\left(\frac{\Delta t}{2}\right)^2 \tag{3.12}$$

Substitution of Equation (3.12) into Equations (3.6) and (3.7) provides a two-step method

$$\phi^* = \phi^n + f\left(\phi^n \phi^*, \phi^*\right)\frac{\Delta t}{2} + O(\Delta t^3) \tag{3.13}$$

$$\phi^{n+1} = \phi^* + f\left(\phi^{n+1} \phi^*, \phi^*\right)\frac{\Delta t}{2} + O(\Delta t^3) \tag{3.14}$$

Equations (3.13) and (3.14) are a computationally linearized implicit equation system. The summation of Equations (3.13) and (3.14) is a 2nd-order computational equivalent of Equation (3.2).

*Remark:*

Although the derivation above provides a two-step direct linearization method, an outer iteration might also be combined for a big $\Delta t$ value. This process can be illustrated as the following:

1. obtain $\phi^{n+1}$ from Equations (3.13) and (3.14).
2. calculate $\phi^{n+1/2}$ using $\left(\phi^{n+1} + \phi^n\right)/2$, where $\phi^{n+1}$ is calculated from step 1.
3. calculate a new $\phi^{n+1}$ using Equations (3.13) and (3.14) with the updated $\phi^{n+1/2}$ in step 2.

Repeat the procedure until the difference between the old and new $\phi^{n+1}$ is acceptable.

### 3.1.2 Computationally Splitting the Linear Term

In the previous section, we introduced time-centered split only into the nonlinear term of $\phi\phi$. In addition to splitting the nonlinear product term, one can also use the same splitting idea in the linear term $\phi$, following

$$\phi^{n+1} = \phi^n + f\left\{\left[\frac{\phi^n + \phi^{n+1}}{2} + O\left(\Delta t^2\right)\right]\phi^{n+1/2}, \left[\frac{\phi^n + \phi^{n+1}}{2} + O\left(\Delta t^2\right)\right]\right\}\Delta t + O(\Delta t^3) \tag{3.15}$$

Equation (3.15) is guaranteed 2nd-order accurate in time because of Equation (3.3). Reorganizing Equation (3.15) and grouping the higher order terms,

$$\phi^{n+1} = \phi^n + f\left\{\phi^n\phi^{n+1/2}, \phi^{n+1}\phi^{n+1/2}, \phi^n, \phi^{n+1}\right\}\frac{\Delta t}{2} + O(\Delta t^3) \tag{3.16}$$

In Equation (3.16), not only the nonlinear term but also the linear term is split into two parts. Therefore, a different intermediate variable $\phi^*$ is defined as

$$\phi^* = \phi^n + f\left(\phi^n\phi^{n+1/2}, \phi^n\right)\frac{\Delta t}{2} \tag{3.17}$$

The second step of the splitting system then becomes,

$$\phi^{n+1} = \phi^* + f\left(\phi^{n+1}\phi^{n+1/2}, \phi^{n+1}\right)\frac{\Delta t}{2} \tag{3.18}$$

Equations (3.17) and (3.18) are different from the split system in the previous section. As a result, the correspondence between $\phi^*$ and $\phi^{n+1/2}$ is proved differently. Instead of the implicit Euler equation, the explicit Euler approximation for $\phi^{n+1/2}$ is introduced,

$$\phi^{n+1/2} = \phi^n + f\left(\phi^n\phi^n, \phi^n\right)\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2 \tag{3.19}$$

Substituting the Taylor expansion of $\phi^{n+1/2}$ into Equation (3.17) and grouping the higher order terms,

$$\phi^* = \phi^n + f\left(\phi^n\phi^n, \phi^n\right)\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2 \tag{3.20}$$

Substituting Equation (3.20) into Equation (3.19), we obtain

$$\phi^{n+1/2} = \phi^* + O\left(\frac{\Delta t}{2}\right)^2 \tag{3.21}$$

Thus, we can use $\phi^*$ to replace $\phi^{n+1/2}$ in Equations (3.17) and (3.18). A different set of computationally linearized two-step equations can be written as:

$$\phi^* = \phi^n + f\left(\phi^n\phi^*, \phi^n\right)\frac{\Delta t}{2} \tag{3.22}$$

$$\phi^{n+1} = \phi^* + f\left(\phi^{n+1}\phi^*, \phi^{n+1}\right)\frac{\Delta t}{2} \tag{3.23}$$

The key to the TCS method is the second-order time-splitting of quadratic nonlinear terms to two different time levels, i.e. $\phi^n\phi^*$ and $\phi^{n+1}\phi^*$. The result is discretely linear in any time-level of information. A further application of the same splitting in the linear terms will give another set of discrete linearized equations. The above derivation for the single variable $\phi$ can be readily extended to a vector of variables, $[\phi_1, \phi_2, ...\phi_N]$, or variables and linear operators, e.g. $[\phi_1, L(\phi_1), \phi_2, L(\phi_2)...\phi_N, L(\phi_2)]$. However, as additional variables (or operators) are introduced, the discretization method has multiple implementations. For example, even with a 1D advection diffusion equation, we can

obtain at least four different TCS discrete formats. In the next section, we will present these four different TCS formats by applying the TCS method to a 1D Burgers' equation.

### 3.2 DERIVATION OF THE TCS METHOD IN A 1D ADVECTION-DIFFUSION EQUATION

The Burgers' equation is the simplest advection-diffusion test case and can be viewed as a prototype of the Navier-Stokes equation or the SWE. The non-conservative Burgers' equation can be written as:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{3.24}$$

To develop the TCS method, we note that in the same vein as Equation (3.7), the midpoint rule can be written across only two time levels as

$$u^{n+1} = u^n + \left(-u^{n+1/2}\delta_x u^{n+1/2} + \nu\delta_x^2 u^{n+1/2}\right)\Delta t + O\left(\Delta t^3\right) \tag{3.25}$$

where $\delta_x$ is the shorthand notation of the generic discretized spatial derivative of 'u'.

#### 3.2.1 The TCSF Discrete Format

The nonlinear advection term in Equation (3.24) is constructed by the flux part 'u' and the gradient part, '$\partial u/\partial x$'. The time-centered splitting maybe applied to either part of the advection term. We first introduce a time-centered linear approximation of $u^{n+1/2} = \left(u^n + u^{n+1}\right)/2 + O\left(\Delta t^2\right)$ to the flux term in the nonlinear product in Equation (3.25). The discrete equation becomes computationally linear in time (i.e. no products with the same time level):

$$u^{n+1} = u^n + \left(-\left[\frac{u^{n+1}+u^n}{2}\right]\delta_x u^{n+1/2} + \nu\delta_x^2 u^{n+1/2}\right)\Delta t + O\left(\Delta t^3\right) \tag{3.26}$$

Multiplying the products in Equation (3.26) provides

$$u^{n+1} = u^n + \left(-u^{n+1}\delta_x u^{n+1/2}\right)\frac{\Delta t}{2} + \left(-u^n\delta_x u^{n+1/2}\right)\frac{\Delta t}{2} + \left(\nu\delta_x^2 u^{n+1/2}\right)\Delta t + O\left(\Delta t^3\right) \tag{3.27}$$

A computational-splitting technique is used to obtain a numerically-solvable set of equations from the discrete form of Equation (3.27). Defining a generic intermediate variable $u^*$ as

$$u^* = u^n + \left(-u^n\delta_x u^{n+1/2}\right)\frac{\Delta t}{2} + \left(\nu\delta_x^2 u^{n+1/2}\right)\frac{\Delta t}{2} \tag{3.28}$$

and subtracting Equation (3.28) from Equation (3.27) provides

$$u^{n+1} = u^* + \left(-u^{n+1}\delta_x u^{n+1/2}\right)\frac{\Delta t}{2} + \left(\nu\delta_x^2 u^{n+1/2}\right)\frac{\Delta t}{2} + O\left(\Delta t^3\right) \tag{3.29}$$

Using an implicit Euler approximation of $u^{n+1/2}$ and the same mathematical derivation as in Equation (3.12), we can prove

$$u^{n+1/2} = u^* + O\left(\frac{\Delta t^2}{4}\right) \tag{3.30}$$

Substitution of Equation (3.30) into Equations (3.28) and (3.29) provides a two-step method, which is called the TCSF method:

$$u^* = u^n + \frac{\Delta t}{2}\left\{-u^n \delta_x u^* + \nu \delta_x^2 u^*\right\} \tag{3.31}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left\{-u^{n+1} \delta_x u^* + \nu \delta_x^2 u^*\right\} + O\left(\Delta t^3\right) \tag{3.32}$$

The summation of Equations (3.31) and (3.32) is equivalent to the original Equation (3.25). The first step is an implicit solution for the variable $u^*$ involving only $u^n$ and with an implicit discretization of the diffusion term. $u^{n+1}$ in the second step can be explicitly calculated.

### 3.2.2 The TCSG Discrete Format

Splitting the gradient term instead of the flux term in Equation (3.25) results in a second basic form of TCS. Thus, instead of Equation (3.26), we obtain

$$u^{n+1} = u^n + \left(-u^{n+1/2}\delta_x\left[\frac{u^{n+1} + u^n}{2}\right] + \nu \delta_x^2 u^{n+1/2}\right)\Delta t + O(\Delta t^3) \tag{3.33}$$

For this second form, we define a slightly different intermediate variable as

$$u^* = u^n + \left(-u^{n+1/2}\delta_x u^n\right)\frac{\Delta t}{2} + \left(\nu \delta_x^2 u^{n+1/2}\right)\frac{\Delta t}{2} \tag{3.34}$$

Similarly in Equation (3.30), we have

$$u^{n+1/2} = u^* + O\left(\frac{\Delta t^2}{4}\right) \tag{3.35}$$

Subtracting Equation (3.34) from (3.33) and substituting Equation (3.35) in the same manner as the transition from Equation (3.27) through (3.32), we obtain a second discrete split form that we will call TCSG

$$u^* = u^n + \frac{\Delta t}{2}\left\{-u^* \delta_x u^n + \nu \delta_x^2 u^*\right\} \tag{3.36}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left\{-u^* \delta_x u^{n+1} + \nu \delta_x^2 u^*\right\} \tag{3.37}$$

The summation of Equations (3.36) and (3.37) is equivalent to the original Equation (3.25). The first step is an implicit solution for the variable $u^*$ involving $u^n$ and with an

implicit discretization of the diffusion term. The second step is an implicit solution for the variable $u^{n+1}$ involving only $u^*$ and with an explicit discretization of the diffusion term.

### 3.2.3 The TCSF-D Discrete Format

The TCSF and TCSG forms are obtained by introducing the time-centered split into the advection term. Further approximation of the diffusion term using the same time splitting techniques will create different discretizations. For instance, in Equation (3.26), we can substitute the time splitting into both the flux and diffusion terms,

$$u^{n+1} = u^n + \left( -\left[ \frac{u^{n+1} + u^n}{2} \right] \delta_x u^{n+1/2} + \nu \delta_x^2 \left[ \frac{u^{n+1} + u^n}{2} \right] \right) \Delta t + O\left(\Delta t^3\right) \tag{3.38}$$

Defining $u^*$ as,

$$u^* = u^n + \left( -u^n \delta_x u^{n+1/2} \right) \frac{\Delta t}{2} + \left( \nu \delta_x^2 u^n \right) \frac{\Delta t}{2} \tag{3.39}$$

the second step follows as,

$$u^{n+1} = u^* + \left( -u^{n+1} \delta_x u^{n+1/2} \right) \frac{\Delta t}{2} + \left( \nu \delta_x^2 u^{n+1} \right) \frac{\Delta t}{2} + O\left(\Delta t^3\right) \tag{3.40}$$

Using an explicit Euler approximation of $u^{n+1/2}$ and the same mathematical derivation as in Equation (3.21), we can prove that

$$u^{n+1/2} = u^* + O\left( \frac{\Delta t^2}{4} \right) \tag{3.41}$$

Substituting Equation (3.30) into Equations (3.39) and (3.40), we obtained the TCSF-D format as

$$u^* = u^n + \frac{\Delta t}{2} \left\{ -u^n \delta_x u^* + \nu \delta_x^2 u^n \right\} \tag{3.42}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2} \left\{ -u^{n+1} \delta_x u^* + \nu \delta_x^2 u^{n+1} \right\} + O\left(\Delta t^3\right) \tag{3.43}$$

We call Equations (3.42) and (3.43) the TCSF-D method because we apply the time-centered splitting to both the flux term and the diffusion term. The summation of Equations (3.42) and (3.43) is equivalent to the original Equation (3.25). The first step is an implicit solution for the variable $u^*$ involving $u^n$ and with an explicit discretization of the diffusion term. The second step is an implicit solution for the variable $u^{n+1}$ involving only $u^*$ and with an implicit discretization of the diffusion term.

### 3.2.4 The TCSG-D Discrete Format

Similar to the development of the TCSF-D method, substitution of time-centered splitting into the diffusion term in the TCSG provides another set of two-step equations as

$$u^* = u^n + \frac{\Delta t}{2}\left\{-u^*\delta_x u^n + \nu\delta_x^2 u^n\right\} \tag{3.44}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left\{-u^*\delta_x u^{n+1} + \nu\delta_x^2 u^{n+1}\right\} \tag{3.45}$$

Equations (3.44) and (3.45) are called the TCSG-D discrete form. The summation of Equations (3.44) and (3.45) is equivalent to the original Equation (3.25). $u^*$ in the first step can be explicitly calculated and the diffusion term is explicitly discretized. The second step is an implicit solution for the variable $u^{n+1}$ involving only $u^*$ and with an implicit discretization of the diffusion term.

We have derived four discrete forms of the TCS method applied to a 1D Burgers' equation in the previous sections. More possible discretizations will be generated when we apply the TCS method to an equation system with multiple variables. We will discuss this characteristic in Chapter 5.

## 3.3 TCS FOR COUPLED MOMENTUM AND SCALAR TRANSPORT

A key advantage of the TCS method is that it provides non-iterative coupling between multiple variables. The method is best understood through application to a simple model problem. First, we will consider the 1D advection-diffusion equation for a scalar $\psi$

$$\frac{\partial\psi}{\partial t} + u\frac{\partial\psi}{\partial x} = \kappa\frac{\partial^2\psi}{\partial x^2} \tag{3.46}$$

where u is the velocity and $\kappa$ is a diffusion coefficient. Equation (3.46) can be coupled to the 1D Burgers' equation for momentum

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{3.47}$$

Applying TCSF to Equation (3.46) provides

$$\psi^* = \psi^n - \frac{\Delta t}{2}\left\{u^n\frac{\partial\psi^*}{\partial x} - \kappa\frac{\partial^2\psi^*}{\partial x^2}\right\} \tag{3.48}$$

$$\psi^{n+1} = \psi^* - \frac{\Delta t}{2}\left\{u^{n+1}\frac{\partial\psi^*}{\partial x} - \kappa\frac{\partial^2\psi^*}{\partial x^2}\right\} \tag{3.49}$$

The first step, Equation (3.48), is an implicit equation for $\psi^*$ involving only time 'n' values of u, whereas the second step, Equation (3.49), linearly couples solution of $\psi^{n+1}$ to

$u^{n+1}$. To complete the algorithm, Equation (3.47) can be similarly discretized so that the coupled TCSF method for both scalar transport and momentum can be written as linear operators,

$$\left[ 1+\frac{\Delta t}{2}\left\{ u^n \frac{\partial(\ )}{\partial x} - \kappa \frac{\partial^2(\ )}{\partial x^2} \right\} \right] \psi^* = \psi^n \tag{3.50}$$

$$\left[ 1+\left\{ \frac{\Delta t}{2} u^n \frac{\partial(\ )}{\partial x} - \nu \frac{\partial^2(\ )}{\partial x^2} \right\} \right] u^* = u^n \tag{3.51}$$

$$\begin{bmatrix} 1 & \frac{\Delta t}{2}\frac{\partial \psi^*}{\partial x} \\ 0 & 1+\frac{\Delta t}{2}\frac{\partial u^*}{\partial x} \end{bmatrix} \begin{bmatrix} \psi^{n+1} \\ u^{n+1} \end{bmatrix} = \begin{bmatrix} \psi^* + \frac{\Delta t}{2}\kappa \frac{\partial^2 \psi^*}{\partial x^2} \\ u^* + \frac{\Delta t}{2}\nu \frac{\partial^2 u^*}{\partial x^2} \end{bmatrix} \tag{3.52}$$

where the parentheses indicate a spatial derivative operator on the term to the right of the brackets. The first step of the TSCF method for N variables over Q grid cells results in N independent linear problems of order Q. The second step of TCSF results in a single linear problem of order NQ. This series of linear problems is a 2nd-order temporal equivalent of the original time-marching, coupled, nonlinear momentum-advection problem. As the TCS uses an intermediate solution $(\psi^*, u^*)$ followed by final solution $(\psi^{n+1}, u^{n+1})$, it resembles predictor-corrector schemes. However, classic predictor-corrector methods are formulated with an explicit predictor of the time n+1 values followed by an implicit corrector to the time n+1 values, which makes the classic predictor-corrector methods restricted by the CFL condition; furthermore, predictor-corrector methods do not provide an avenue for a simple nonlinear solution (see discussions in Lomax et al, 1999 and Tannehill et al, 1997). In contrast, the new TCS method provides an implicit linearized predictor of the time n+1/2 values that are used in a coupled implicit solution of the time n+1 values. This new approach provides a simple method of linearly-coupling equations that are nonlinearly-coupled in the original problem. The method requires only linear matrix solutions and does not require the outer iteration of Picard or Newton methods. As compared to the functional Jacobians required for local linearization and Newton iteration, the TCS coefficient matrices are relatively easy to derive and have forms very similar to the original model problem.

### 3.4 SUMMARY

A new computational linearization method, the TCS, is derived and analyzed in this chapter. Without iteration and calculation of the Jacobian, the TCS method splits the quadratic nonlinear term into two steps so that each step is computationally linear. That is, for time marching from known state $\mathbf{x}^n$ to unknown state $\mathbf{x}^{n+1}$ we use a linear solution of $\tilde{\mathbf{A}}(\mathbf{x}^n)\tilde{\mathbf{x}} = \tilde{\mathbf{b}}$ followed by $\tilde{\tilde{\mathbf{A}}}(\tilde{\mathbf{x}})\mathbf{x}^{n+1} = \tilde{\tilde{\mathbf{b}}}$ that is a second-order equivalent to

$\mathbf{A}(\mathbf{x}^{n+1})\mathbf{x}^{n+1} = \mathbf{b}$. In addition to the linearization, the TCS method can generate different TCS discrete forms. All different TCS formats are mathematically equivalent to the original implicit midpoint rule discretization. As a result, they share the same 2nd-order temporal accuracy. Furthermore, we also demonstrated that the TCS method can couple multiple variables without iterations. Characteristics of the TCS method such as accuracy, stability and efficiency will be explored in the next chapter using examples with analytical solutions.

# Chapter 4 Implementation of the TCS Method in 1D Problems

The theory of the TCS method is illustrated in Chapter 3. To investigate the characteristics of the new method, we apply the TCS method to three different test cases: a 1D conservative Burgers' equation, a 1D non-conservative Burgers' equation and a 1D nonlinear ordinary differential equation (ODE). For each test case, the TCS algorithm is compared to the conventional implicit nonlinear solution methods (local linearization, Picard iteration and Newton iteration) applied to Crank-Nicolson discretization. The temporal accuracy of different TCS discretizations is verified by all three test cases. The practical stability of the TCS method is confirmed using the unsteady flow test case with an analytical solution in both conservative and non-conservative forms. The method is shown to require computational effort similar to local linearization, but does not require discrete computation of a functional Jacobian for solution.

## 4.1 Application of the TCS Method to the 1D Conservative Burgers' Equation

### 4.1.1 Discrete formats of the 1D Conservative Burger's equation using various methods

Burgers' equation provides a useful model problem for comparing the TCS to other nonlinear solution methods. We will begin from the conservative form of Burgers' equation. The application of the TCS method to the non-conservative Burgers' equation will be discussed in the next section. The conservative form of the 1D Burgers' equation is:

$$\frac{\partial u}{\partial t}+\frac{1}{2}\frac{\partial}{\partial x}\left(u^2\right)=\nu\frac{\partial^2 u}{\partial x^2} \tag{4.1}$$

***Time-Centered Split Method***

For an equation with one variable and a simple quadratic nonlinearity, TCSF and TCSG collapse into a single form. For Equation (4.1), the TCSF and TCSG methods are identical, and can be presented as linear operators

$$\left[1+\frac{\Delta t}{2}\left\{\frac{1}{2}\frac{\partial u^n(\ )}{\partial x}-\nu\frac{\partial^2(\ )}{\partial x^2}\right\}\right]u^*=u^n \tag{4.2}$$

$$\left[1+\frac{\Delta t}{4}\frac{\partial u^*(\ )}{\partial x}\right]u^{n+1}=u^*+\frac{\Delta t\nu}{2}\frac{\partial^2 u^*}{\partial x^2} \tag{4.3}$$

For simplicity in the following discussion, we use the generic symbols **A**, **x** and **b** to represent a coefficient matrix, the left-hand-side (LHS) variable vector and the right-hand-side (RHS) known vector in a matrix equation of the form of **Ax** = **b** for various solution methods. The number of grid points in space is Q. A central difference spatial discretization is applied to derivatives in Equation (4.2) and(4.3). It is useful to define a viscous scale, $\gamma$,

$$\gamma \equiv \frac{\nu \Delta t}{\Delta x^2} \tag{4.4}$$

and the 'i' grid cell CFL number for the $L \in \{n, *, n+1\}$ time level for $i = \{1, 2, ...Q\}$ as

$$C_i^L \equiv \frac{u_i^L \Delta t}{\Delta x} \tag{4.5}$$

Similarly, it will be useful to also define a diffusion operator at any time level as

$$D_i^L \equiv \gamma\left(u_{i+1}^L - 2u_i^L + u_{i-1}^L\right) \tag{4.6}$$

and a nonlinear adjective gradient operator

$$G_i^L \equiv C_{i+1}^L u_{i+1}^L - C_{i-1}^L u_{i-1}^L \tag{4.7}$$

It follows that Eq, (4.2), the first step of TCSF can be written in the form **Ax** = **b** over Q grid points where the **A** matrix is tridiagonal such that

$$\mathbf{A} = \begin{bmatrix} 1+\gamma & -\frac{\gamma}{2} + \frac{C_2^n}{8} & & & 0 \\ -\frac{\gamma}{2} - \frac{C_1^n}{8} & 1+\gamma & -\frac{\gamma}{2} + \frac{C_3^n}{8} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma}{2} - \frac{C_{Q-2}^n}{8} & 1+\gamma & -\frac{\gamma}{2} + \frac{C_Q^n}{8} \\ 0 & & & -\frac{\gamma}{2} - \frac{C_{Q-1}^n}{8} & 1+\gamma \end{bmatrix} \tag{4.8}$$

$$\mathbf{x} = \left[u_1^*, u_2^*, ...u_Q\right]^T \tag{4.9}$$

$$\mathbf{b} = \begin{bmatrix} u_1^n + \left(\frac{\gamma}{2} + \frac{C_0^n}{8}\right) u_0^* \\ u_2^n \\ u_3^n \\ \vdots \\ u_{Q-1}^n \\ u_Q^n + \left(\frac{\gamma}{2} - \frac{C_{Q+1}^n}{8}\right) u_{Q+1}^* \end{bmatrix} \tag{4.10}$$

with $C_0^n, u_0^*, C_{Q+1}^n$ and $u_{Q+1}^*$ implemented as Dirichlet boundary conditions. Neumann boundary conditions can also be readily invoked, but are not presented here for brevity.

The second step of TCS for the 1D Burgers' equation can be as evaluated from another **Ax** = **b** problem using

$$\mathbf{A} = \begin{bmatrix} 1 & +\frac{C_2^*}{8} & & & 0 \\ -\frac{C_1^*}{8} & 1 & +\frac{C_3^*}{8} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{C_{Q-2}^*}{8} & 1 & +\frac{C_Q^*}{8} \\ 0 & & & -\frac{C_{Q-1}^*}{8} & 1 \end{bmatrix} \tag{4.11}$$

$$\mathbf{x} = \left[u_1^{n+1}, u_2^{n+1}, \ldots u_Q^{n+1}\right]^T \tag{4.12}$$

$$\mathbf{b} = \begin{bmatrix} u_1^* + \frac{1}{2}D_1^* + \frac{C_0^*}{8} \\ u_2^* + \frac{1}{2}D_2^* \\ u_3^* + \frac{1}{2}D_3^* \\ \vdots \\ u_{Q-1}^* + \frac{1}{2}D_{Q-1}^* \\ u_Q^* + \frac{1}{2}D_Q^* - \frac{C_{Q+1}^*}{8} \end{bmatrix} \tag{4.13}$$

Thus, the TCSF method for the 1D conservative Burgers' equation requires solution of two tridiagonal linear problems at each time step.

If we further split the diffusion term, we obtain the TCSF-D form for Equation (4.1) as,

$$\left[ 1 + \frac{\Delta t}{4}\frac{\partial u^n(\ )}{\partial x} \right] u^* = u^n + \frac{\Delta t \nu}{2}\frac{\partial^2 u^n}{\partial x^2} \tag{4.14}$$

$$\left[ 1 + \frac{\Delta t}{2}\left\{ \frac{1}{2}\frac{\partial u^*(\ )}{\partial x} - \nu\frac{\partial^2(\ )}{\partial x^2} \right\} \right] u^{n+1} = u^* \tag{4.15}$$

The first step is a matrix equation **Ax=b** with

$$\mathbf{A} = \begin{bmatrix} 1 & +\frac{C_2^n}{8} & & & 0 \\ -\frac{C_1^n}{8} & 1 & +\frac{C_3^n}{8} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{C_{Q-2}^n}{8} & 1 & +\frac{C_Q^n}{8} \\ 0 & & & -\frac{C_{Q-1}^n}{8} & 1 \end{bmatrix} \tag{4.16}$$

$$\mathbf{x} = \left[ u_1^*, u_2^*, \ldots u_Q^* \right]^T \tag{4.17}$$

$$\mathbf{b} = \begin{bmatrix} u_1^n + \frac{1}{2} D_1^n + \frac{C_0^n}{8} \\ u_2^n + \frac{1}{2} D_2^n \\ u_3^n + \frac{1}{2} D_3^n \\ \vdots \\ u_{Q-1}^n + \frac{1}{2} D_{Q-1}^n \\ u_Q^n + \frac{1}{2} D_Q^n - \frac{C_{Q+1}^n}{8} \end{bmatrix} \tag{4.18}$$

In the second step, we have

$$\mathbf{A} = \begin{bmatrix} 1+\gamma & -\frac{\gamma}{2} + \frac{C_2^*}{8} & & & 0 \\ -\frac{\gamma}{2} - \frac{C_1^*}{8} & 1+\gamma & -\frac{\gamma}{2} + \frac{C_3^*}{8} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma}{2} - \frac{C_{Q-2}^*}{8} & 1+\gamma & -\frac{\gamma}{2} + \frac{C_Q^*}{8} \\ 0 & & & -\frac{\gamma}{2} - \frac{C_{Q-1}^*}{8} & 1+\gamma \end{bmatrix} \tag{4.19}$$

$$\mathbf{x} = \left[ u_1^{n+1}, u_2^{n+1}, \ldots u_Q^{n+1} \right]^T \tag{4.20}$$

$$\mathbf{b} = \begin{bmatrix} u_1^* + \left(\frac{\gamma}{2} + \frac{C_0^n}{8}\right) u_0^{n+1} \\ u_2^* \\ u_3^* \\ \vdots \\ u_{Q-1}^* \\ u_Q^* + \left(\frac{\gamma}{2} - \frac{C_{Q+1}^n}{8}\right) u_{Q+1}^{n+1} \end{bmatrix} \tag{4.21}$$

Similar to TCSF, the TCSF-D method for the 1D conservative Burgers' equation also requires solution of two tridiagonal linear problems at each time step. However, TCSF-D reverses the solution process in TCSF. The first step in TCSF-D is in the same structure as the second step in TCSF; the second step in TCSF-D is in the same structure as the first step in TCSF.

To compare the above TCS methods to other temporally 2nd-order accurate implicit nonlinear solution methods, we apply Crank-Nicolson 2nd-order temporal and central difference spatial 2nd-order discretization to Equation (4.1), resulting in

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{2}\left\{\frac{u_{i+1}^n u_{i+1}^n - u_{i-1}^n u_{i-1}^n}{4\Delta x} - \nu\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} + \frac{u_{i+1}^{n+1} u_{i+1}^{n+1} - u_{i-1}^{n+1} u_{i-1}^{n+1}}{4\Delta x} - \nu\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2}\right\} \tag{4.22}$$

Conventional linearization methods such as Picard iteration, Newton iteration and local linearization methods are used to solve Equation (4.22).

***Picard Iteration***

The simplest approach to implement for a nonlinear equation such as Equation (4.22) is a lagged-coefficient iteration (Tannehill et al. 1997,pg 450) , which is a form of Picard iteration. A linear inner equation is formed by estimating the time n+1 flux as $\left(u_i^{n+1}\right)^k$, where the additional superscript 'k' is introduced as the outer iteration counter. Equation (4.22) can then be represented as the Picard method

$$\left(u_i^{n+1}\right)^{k+1} = u_i^n - \frac{\Delta t}{2}\left\{\frac{u_{i+1}^n u_{i+1}^n - u_{i-1}^n u_{i-1}^n}{4\Delta x} - \nu\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\right.$$
$$+\frac{\left(u_{i+1}^{n+1}\right)^k\left(u_{i+1}^{n+1}\right)^{k+1} - \left(u_{i-1}^{n+1}\right)^k\left(u_{i-1}^{n+1}\right)^{k+1}}{4\Delta x} \qquad (4.23)$$
$$\left. - \nu\frac{\left(u_{i+1}^{n+1}\right)^{k+1} - 2\left(u_i^{n+1}\right)^{k+1} + \left(u_{i-1}^{n+1}\right)^{k+1}}{\Delta x^2}\right\}$$

which is solved for k={1,2,3...} until an appropriate convergence criterion is reached. . Thus, each outer iteration requires an inner linear solution of the form **Ax** = **b.** For comparison with other methods, it is useful to define an $\mathbf{A}^k\mathbf{x}^{k+1} = \mathbf{b} + \mathbf{c}^k$ form in which the **b** vector requires only a single computation in each time step whereas the **c** vector is recomputed in each outer iteration. For Equation (4.23), the result is

$$\mathbf{A}^k = \begin{bmatrix} 1+\gamma & -\frac{\gamma}{2}+\frac{C_3^k}{8} & & & 0 \\ -\frac{\gamma}{2}-\frac{C_2^k}{8} & 1+\gamma & -\frac{\gamma}{2}+\frac{C_4^k}{8} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma}{2}-\frac{C_{Q-2}^k}{8} & 1+\gamma & -\frac{\gamma}{2}+\frac{C_Q^k}{8} \\ 0 & & & -\frac{\gamma}{2}-\frac{C_{Q-1}^k}{8} & 1+\gamma \end{bmatrix} \qquad (4.24)$$

$$\mathbf{x}^{k+1} = \left[\left(u_1^{n+1}\right)^{k+1}, \left(u_2^{n+1}\right)^{k+1}, \ldots \left(u_Q^{n+1}\right)^{k+1}\right]^T \qquad (4.25)$$

$$\mathbf{b} = \begin{bmatrix} u_1^n - \frac{1}{8}G_1^n + \frac{1}{2}D_1^n \\ u_2^n - \frac{1}{8}G_2^n + \frac{1}{2}D_2^n \\ u_3^n - \frac{1}{8}G_3^n + \frac{1}{2}D_3^n \\ \vdots \\ u_{Q-1}^n - \frac{1}{8}G_{Q-1}^n + \frac{1}{2}D_{Q-1}^n \\ u_Q^n - \frac{1}{8}G_Q^n + \frac{1}{2}D_Q^n \end{bmatrix} \tag{4.26}$$

$$\mathbf{c}^k = \begin{bmatrix} \frac{\gamma}{2} + \frac{1}{8}C_0^k \\ 0 \\ 0 \\ \vdots \\ 0 \\ \frac{\gamma}{2} - \frac{1}{8}C_{Q+1}^k \end{bmatrix} \tag{4.27}$$

For the Picard iteration, the first outer iterative solution is started with $\left(u_i^{n+1}\right)^1 = u_i^n$ . The outer iteration is stopped when $L\left\{\left(\mathbf{u}^{n+1}\right)^{k+1} - \left(\mathbf{u}^{n+1}\right)^k\right\} < \varepsilon$ where $L$ is an appropriate linear norm acting on the vector $\mathbf{u}$ and $\varepsilon$ is the desired convergence. The above method can be expected to have no better than first order convergence for the outer iteration. Solution for each time step requires multiple tridiagonal solutions of the $\mathbf{Ax} = \mathbf{b} + \mathbf{c}$ problem where the **A** matrix and boundary conditions (possibly) in **c** are re-calculated at each outer iteration.

***Newton Iteration***

The Newton method is commonly used for accelerated convergence in iterative solution of nonlinear problems. Using classic Newton iteration method, the time-march from 'n' to 'n+1' of Equation (4.22) can be written as the root-finding problem for the function g as

$$g_i\left(u_{i-1}^{n+1},u_i^{n+1},u_{i+1}^{n+1}\right)=u_i^{n+1}-u_i^n + \frac{\Delta t}{2}\left\{\frac{u_{i+1}^{n+1}u_{i+1}^{n+1}-u_{i-1}^{n+1}u_{i-1}^{n+1}}{4\Delta x}-\nu\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\Delta x^2}\right. \\ \left. +\frac{u_{i+1}^{n}u_{i+1}^{n}-u_{i-1}^{n}u_{i-1}^{n}}{2\Delta x}-\nu\frac{u_{i+1}^{n}-2u_i^{n}+u_{i-1}^{n}}{\Delta x^2}\right\}=0 \tag{4.28}$$

Applying Newton iteration(Ferziger and Peric 1999), Equation (4.28) can be written as

$$\left(\frac{\partial g_i}{\partial u_j^{n+1}}\right)^k\left(\Delta u_j^{n+1}\right)^{k+1}=-g_i^k \tag{4.29}$$

where

$$\left(\Delta u_j^{n+1}\right)^{k+1}=\left(u_j^{n+1}\right)^{k+1}-\left(u_j^{n+1}\right)^k \tag{4.30}$$

and $\left(\frac{\partial g_i}{\partial u_j^{n+1}}\right)^k$ is the Jacobian matrix, evaluated using term by term discretizations of Equation (4.28). The result can be presented as a linearized equation system $\mathbf{A}^k\mathbf{x}^{k+1}=\mathbf{b}+\mathbf{c}^k$ with

$$\mathbf{A}^k=\begin{bmatrix} 1+\gamma & \frac{C_2^k}{4}-\frac{\gamma}{2} & & & 0 \\ -\frac{C_1^k}{4}-\frac{\gamma}{2} & 1+\gamma & \frac{C_3^k}{4}-\frac{\gamma}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{C_{Q-2}^k}{4}-\frac{\gamma}{2} & 1+\gamma & \frac{\gamma}{2}-\frac{C_Q^k}{4} \\ 0 & & & -\frac{C_{Q-1}^k}{4}-\frac{\gamma}{2} & 1+\gamma \end{bmatrix} \tag{4.31}$$

$$\mathbf{x}^{k+1}=\left[\left(\Delta u_1^{n+1}\right)^{k+1},\left(\Delta u_2^{n+1}\right)^{k+1},\ldots\left(\Delta u_Q^{n+1}\right)^{k+1}\right]^T \tag{4.32}$$

$$\mathbf{b} = -\begin{bmatrix} u_1^n + \frac{G_1^n}{8} - \frac{D_1^n}{2} \\ u_2^n + \frac{G_2^n}{8} - \frac{D_2^n}{2} \\ \vdots \\ u_{Q-1}^n + \frac{G_{Q-1}^n}{8} - \frac{D_{Q-1}^n}{2} \\ u_Q^n + \frac{G_Q^n}{8} - \frac{D_Q^n}{2} \end{bmatrix} \tag{4.33}$$

$$\mathbf{c}^k = -\begin{bmatrix} u_1^k + \frac{G_1^k}{8} - \frac{D_1^k}{2} + \frac{c_0^k}{4} + \frac{\gamma}{2} \\ u_2^k + \frac{G_2^k}{8} - \frac{D_2^k}{2} \\ \vdots \\ u_{Q-1}^k + \frac{G_{Q-1}^k}{8} - \frac{D_{Q-1}^k}{2} \\ u_Q^k + \frac{G_Q^k}{8} - \frac{D_Q^k}{2} - \frac{c_{Q+1}^k}{4} + \frac{\gamma}{2} \end{bmatrix} \tag{4.34}$$

In the present 1D single-variable example, the Jacobian is relatively easy to compute and thus it is easy to form the **A** matrix. However, in multi-dimensional, multi-variable equation systems, the element-by-element calculation of the **A** matrix from $\partial g / \partial u$ is generally difficult to derive and code. As the **A** matrix must be recomputed for every outer iteration, the resulting outer iterations can be computationally expensive.

***Local Linearization***

Local linearization is not widely used for time-marching CFD problems, but is presented here because it has some similarities to the TCS method. Lomax et al. (1999) provides an example of solving an ODE using Local linearization which we extend to solving Equation (4.22). Local linearization approximates the time march to the "n+1" time level using "n" time level and a Taylor expansion. The result is a linear equation set without an outer iteration. To apply Local Linearization, Equation (4.22) can be written as,

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{2}\left[ F_i^{n+1}\left(u_{i-1}^{n+1}, u_i^{n+1}, u_{i-1}^{n+1}\right) + F_i^n\left(u_{i-1}^{n+1}, u_i^{n+1}, u_{i-1}^{n+1}\right)\right] \tag{4.35}$$

where

$$F_i^L\left(u_{i-1}^L, u_i^L, u_{i+1}^L\right) = -\frac{u_{i+1}^L u_{i+1}^L - u_{i-1}^L u_{i-1}^L}{4\Delta x} + \nu\frac{u_{i+1}^L - 2u_i^L + u_{i-1}^L}{\Delta x^2} \tag{4.36}$$

for $L \in \{n, n+1\}$. Following Lomax et al. (1999), apply a Taylor expansion onto $F_i^{n+1}$ ,

$$\begin{aligned} u_i^{n+1} = u_i^n + \frac{\Delta t}{2}\Big\{ & F_i^n\left(u_{i-1}^{n+1}, u_i^{n+1}, u_{i-1}^{n+1}\right) \\ & + \sum_{j=i-1}^{i+1}\left(u_j^{n+1} - u_j^n\right)\frac{\partial F_i^n\left(u_{i-1}^{n+1}, u_i^{n+1}, u_{i-1}^{n+1}\right)}{\partial u_j^n} \\ & + F_i^n\left(u_{i-1}^{n+1}, u_i^{n+1}, u_{i-1}^{n+1}\right)\Big\} + O\left(\Delta t^3\right) \end{aligned} \tag{4.37}$$

Equation (4.37) can be reorganized as

$$\left[1 - \frac{\Delta t}{2}\frac{\partial F_i^n}{\partial u_j^n}\right]\Delta u_j^{n+1} = \Delta t F_i^n \tag{4.38}$$

where $\partial F_i^n / \partial u_j^n$ is the Jacobian and

$$\Delta u_j^{n+1} = u_j^{n+1} - u_j^n \tag{4.39}$$

Therefore, the Local Linearization system can be written as $\mathbf{A}\mathbf{x}^{n+1} = \mathbf{b}$ :

$$\mathbf{A} = \begin{bmatrix} 1+\gamma & \frac{C_2^n}{4} - \frac{\gamma}{2} & & & 0 \\ -\frac{C_1^n}{4} - \frac{\gamma}{2} & 1+\gamma & \frac{C_3^n}{4} - \frac{\gamma}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{C_{Q-2}^n}{4} - \frac{\gamma}{2} & 1+\gamma & \frac{\gamma}{2} - \frac{C_Q^n}{4} \\ 0 & & & -\frac{C_{Q-1}^n}{4} - \frac{\gamma}{2} & 1+\gamma \end{bmatrix} \tag{4.40}$$

$$\mathbf{x}^{n+1} = \left[\Delta u_1^{n+1}, \Delta u_2^{n+1}, \ldots \Delta u_Q^{n+1}\right]^T \tag{4.41}$$

$$\mathbf{b} = \begin{bmatrix} -\frac{G_1^n}{4} + D_1^n + \frac{C_0^n}{4} + \frac{\gamma}{2} \\ -\frac{G_2^n}{4} + D_2^n \\ \vdots \\ -\frac{G_{Q-1}^n}{4} + D_{Q-1}^n \\ -\frac{G_Q^n}{4} + D_Q^n - \frac{C_{Q+1}^n}{4} + \frac{\gamma}{2} \end{bmatrix} \tag{4.42}$$

Much like the TCS method, Local Linearization provides a linear equation system that is a second-order approximation of the original nonlinear system and does not require an outer iteration. In the above 1D single variable example, the Jacobian is relatively easy to compute and derive the **A** matrix, however, in multi-dimensional, multi-variable systems the Jacobian may be difficult to derive and compute. Indeed, the difficulty of deriving the Jacobian for general multi-dimensional systems is arguably the principal reason that local linearization has not been widely used in CFD.

### 4.1.2 Results and Discussion

***Accuracy***

The accuracy of the TCS method is demonstrated and compared to the other nonlinear methods for the unsteady 1D Burgers' equation. Over the domain $0 \le x \le 1$ with initial and boundary conditions of

$$u(0,t) = 0 \tag{4.43}$$

$$u(1,t) = 0 \tag{4.44}$$

$$u(x,0) = \sin(\pi x) \quad : \quad 0 < x < 1 \tag{4.45}$$

the 1D Burgers' equation has a solution constructed from truncated Fourier series:

$$u_{analytical}(x,t) = 2\pi\nu \frac{\sum_{k=1}^{K} a_m \exp(-k^2\pi^2\nu t) k \sin(k\pi x)}{a_0 + \sum_{k=1}^{K} a_k \exp(-k^2\pi^2\nu t)\cos(k\pi x)} \quad (k = 1,2,3,...K) \tag{4.46}$$

where

$$a_0 = \int_0^1 \exp\left\{-(2\pi\nu)^{-1}\left[1-\cos(\pi x)\right]\right\} dx \tag{4.47}$$

$$a_k = 2\int_0^1 \exp\left\{-(2\pi\nu)^{-1}\left[1-\cos(\pi x)\right]\right\}\cos(k\pi x)\,dx \tag{4.48}$$

The solution becomes exact as $K \to \infty$. This specific case of Burgers' equation has been frequently used to verify numerical methods (Kadalbajoo and Awasthi 2006; Kutluay et al. 1999). A discrete approximation of Equation (4.46) for a truncated series of K=30 terms is plotted in Figure 4.1. The integrals in Equations (4.47) and (4.48) are numerically evaluated using "quad" function in MATLAB with $\Delta x$ =0.01 and $\nu = 0.05$.

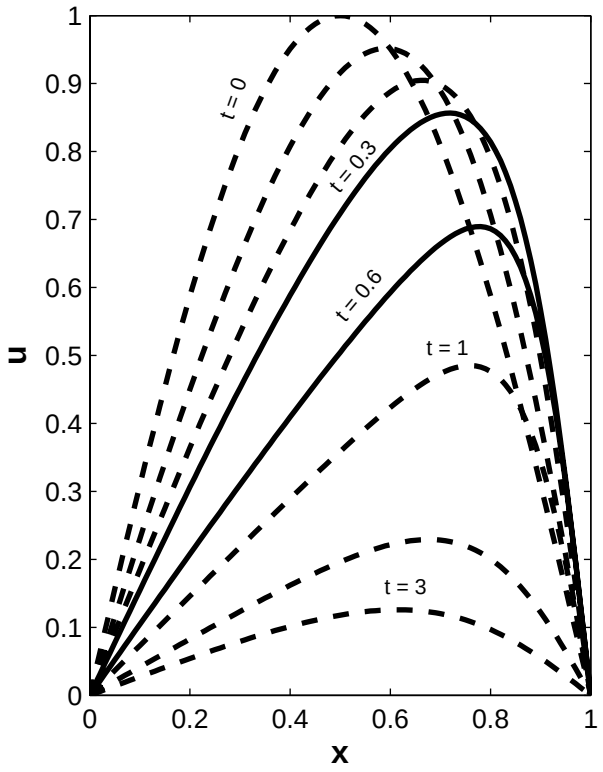


Figure 4.1 Solution of 1D Burgers' equation evolving in time for $t \in \{0, 0.1, 0.2, 0.3, 0.6, 1, 2, 3\}$ using $\nu = 0.05$ along with the initial and boundary conditions of Equations (4.43) through (4.45).

We have solved this 1D Burgers' equation for the various nonlinear methods discretized in the previous session. All methods use 2nd-order central difference spatial discretization and a sufficiently fine uniform mesh ($\Delta x = 0.02$) such that error is controlled by temporal accuracy. As a basis for error comparison, we use the time of maximum absolute error found in these two TCS methods, which is determined from Figure 4.2 as t = 0.3 for both the TCSF and TCSF-D. Note that the maximum error

occurs when the solution reaches its maximum steepness in Figure 4.1. With a smaller $\nu$, the similar asymptotic behavior is obtained when $\Delta t$ is sufficiently small.

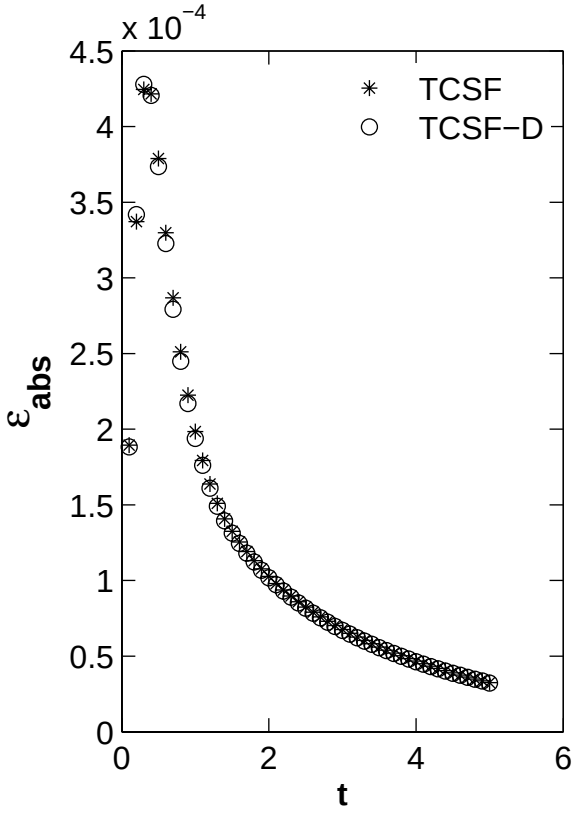


Figure 4.2 Absolute error time evolution for numerical solutions of 1D conservative Burgers' equation for TCSF and TCSF-D using $\{\Delta t = 0.01,\ \Delta x = 1/50,$ $\nu = 0.05\}$, where $\varepsilon_{abs}(t) = \left| \frac{1}{50} \sum_{i=1}^{50} u_{model}(x_i, t) - u_{analytical}(x_i, t) \right|$ and $u_{analytical}(x, t)$ is numerically calculated from Equation (4.46) with K=30.

To evaluate the model error, we examine performance with nine different time steps over the range $0.006 \le \Delta t \le 0.15$ and with grid cell spacing of $\Delta x \in \left\{2 \times 10^{-2}, 5 \times 10^{-3}, 1.25 \times 10^{-3}\right\}$. Based on the Burgers' equation solution for $u_{analytical}$, the test conditions cover the range $0.25 \le CFL \le 50$, where

$$CFL = \max\left|u_{analytical}(t)\right|\frac{\Delta t}{\Delta x} \tag{4.49}$$

is the maximum CFL number. The error for the different methods has been estimated using the RMS error, $L_2$ and $L_\infty$ over Q grid points. These approaches are similar to Rueda and Schladow (2002), where

$$\varepsilon_{RMS} = \sqrt{\frac{1}{Q}\sum_{j=1}^{Q}\left(u_j - \tilde{u}_j\right)^2} \tag{4.50}$$

$$L_2 = \frac{\left[\sum_{k=1}^{M}\left(u_k - \tilde{u}_k\right)^2\right]^{1/2}}{\left[\sum_{k=1}^{M}\left(\tilde{u}_k\right)^2\right]^{1/2}} \tag{4.51}$$

$$L_\infty = \frac{\max\left|u_k - \tilde{u}_k\right|}{\max\left|\tilde{u}_k\right|} \quad : \quad 1 \le k \le M \tag{4.52}$$

and $\tilde{u}$ is the model solution using an extremely fine time step such that $0.3/10^4$, which is 0.5% of the smallest time step used in the other simulations. Using a fine-time step solution is a standard approach (e.g. Ferziger and Peric, 1999) and is preferred over the approximate numerical solution of Equation (4.46), so that the accuracy measure is not distorted by the differences in the numerical approximations. Similarly, for consistency in the inter-model comparison, each model is compared to the small time-step solution for that model (e.g., the Picard method is compared to the small time-step solution of the Picard method, not to a small time-step TCSF solution). Figure 4.3 shows the RMS error associated with different methods over the tested ranges of CFL. It can be seen that all the methods provide 2nd-order accuracy for fixed $\Delta x$ and decreasing $\Delta t$. Error magnitudes are substantially similar using the same $\Delta x$ and $\Delta t$ except for the TCSF-D method. The TCSF-D method has a noticeably smaller error than all the other methods. $L_2$ and $L_\infty$ norms behave similar to RMS error and are plotted in Figure 4.4 and Figure 4.5 respectively.

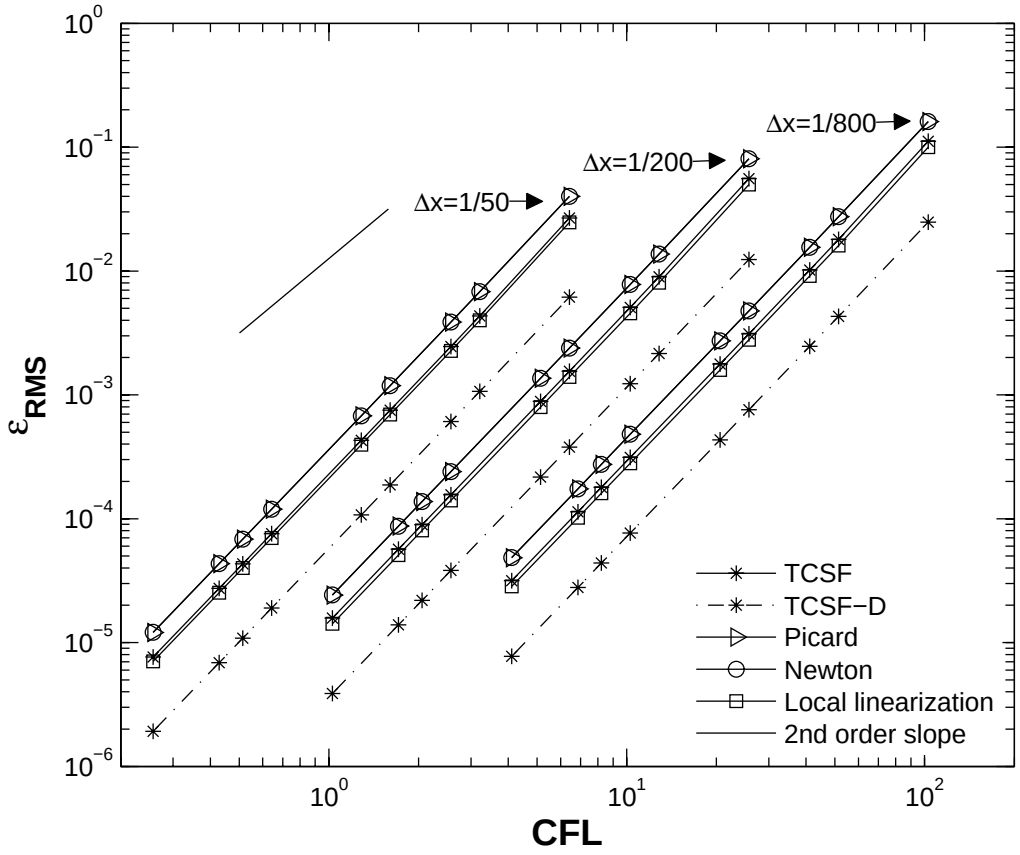


Figure 4.3 $\varepsilon_{RMS}$ vs. CFL number of various methods for solution of 1D conservative Burgers' equation with three different $\Delta x$, where $\nu = 0.05$ and $\Delta t = 0.3/\Gamma$ with $\Gamma \in \{50, 30, 25, 20, 10, 8, 5, 4, 2\}$.

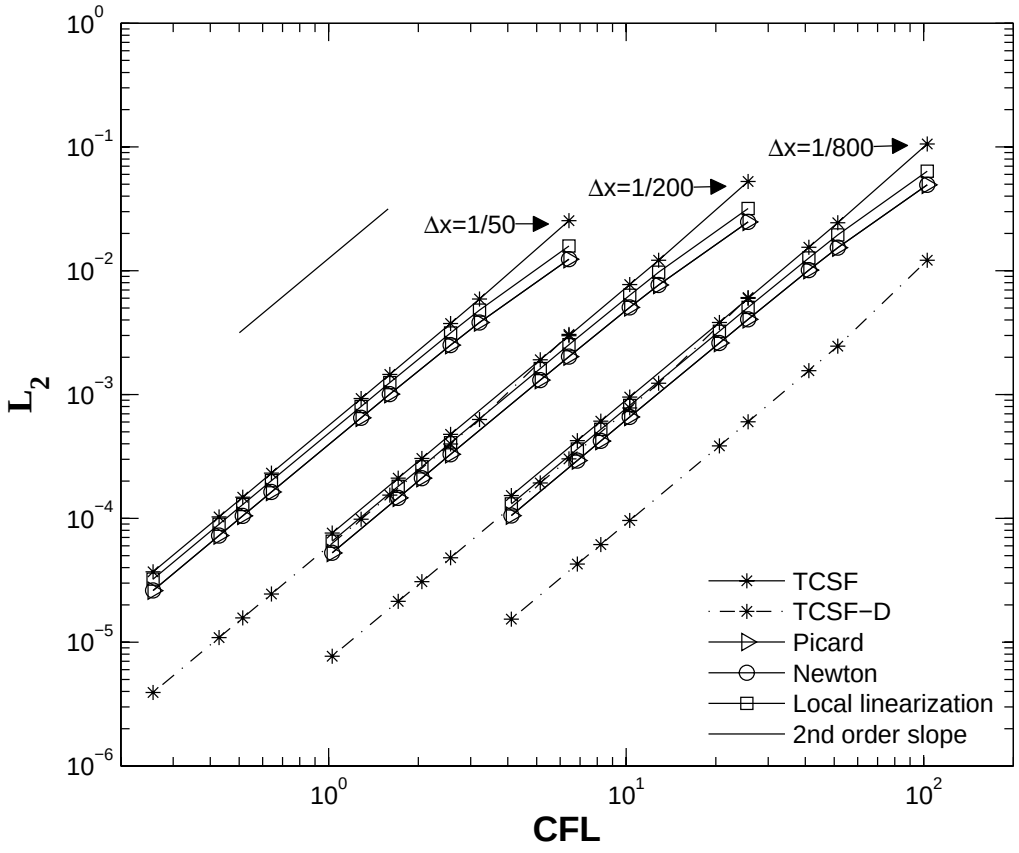


Figure 4.4 $L_2$ norm vs. CFL number of various methods for solution of 1D conservative Burgers' equation with three different $\Delta x$, where $\nu = 0.05$ and $\Delta t = 0.3/\Gamma$ with $\Gamma \in \{50, 30, 25, 20, 10, 8, 5, 4, 2\}$.

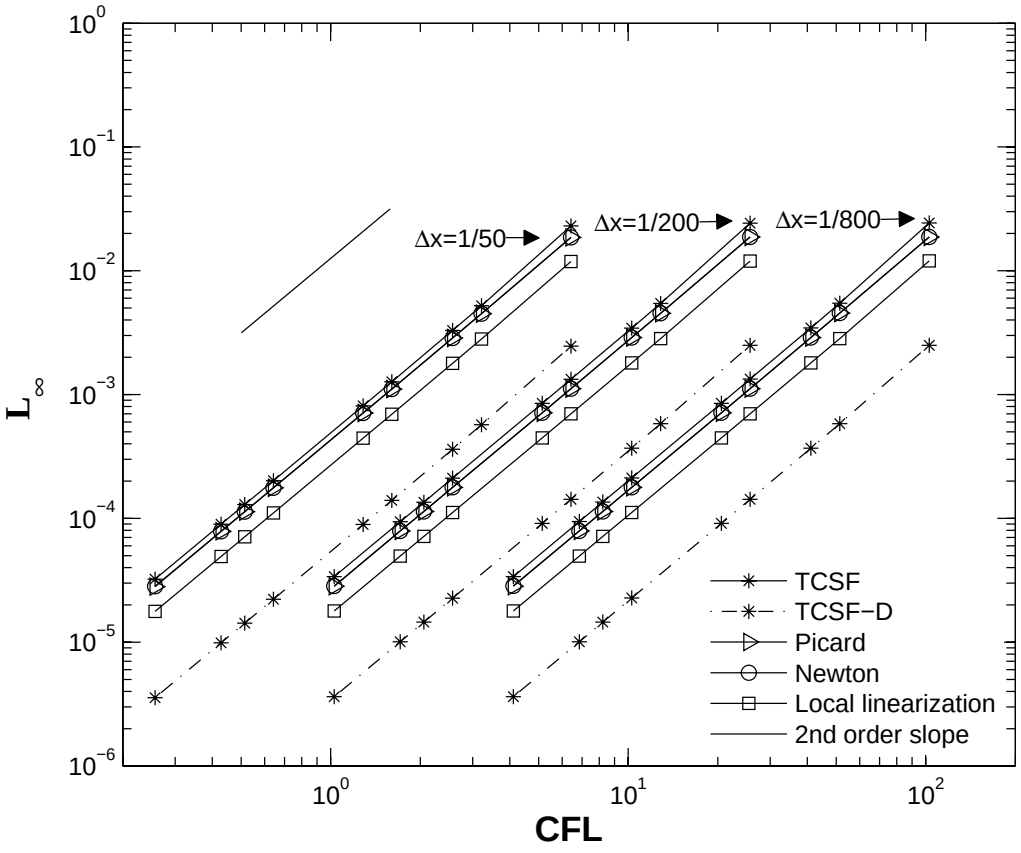


Figure 4.5 $L_\infty$ norm vs. CFL number of various methods for solution of 1D conservative Burgers' equation with three different $\Delta x$, where $\nu = 0.05$ and $\Delta t = 0.3/\Gamma$ with $\Gamma \in \{50, 30, 25, 20, 10, 8, 5, 4, 2\}$.

***Stability***

Nonlinear stability cannot be generally proven, but is instead demonstrated by examples. As shown in Figure 4.3, all five numerical methods remain stable over the full range of tested conditions. The stability of Picard iteration, Newton iteration and Local linearization may be expected because all the underlying discretizations begin with the Crank-Nicolson scheme. The TCS method is based on a midpoint rule discretization that is split, which cannot be guranteed stable even if the split matrices are linearly stable. Our tests thus far have shown very promising stability characteristics; however, such result cannot be presumed definitive. The new TCS method may theoretically be applied for any class of quadratically-nonlinear coupled equations, but its stability characteristics for different equations remains a subject for future investigation.

The TCS method is in an implicit format, thus it is expected to have stability advantage over the explicit methods. To compare the TCS method with explicit methods, we solve the same 1D conservative Burgers' equation using Runge-Kutta 2nd-Order (RK2) and Runge-Kutta 4th-Order (RK4) methods. The RMS error over a range of CFL numbers is plotted in Figure 4.3 for the TCSF, TCSF-D, RK2 and RK4 methods. In this comparison, we study the performance with 12 different time steps over the range $0.0006 \le \Delta t \le 0.06$ and with grid cell spacing of $\Delta x = 0.02$. As a result, the test conditions cover the range, $0.0013 \le CFL \le 2.57$. The points of th RK2 and RK4 methods are only shown in the left part of Figure 4.6 because these two explicit methods become unstable when $CFL \ge 0.3$. Contrarily, the TCSF and TCSF-D methods remain stable in the whole range of CFL numbers.

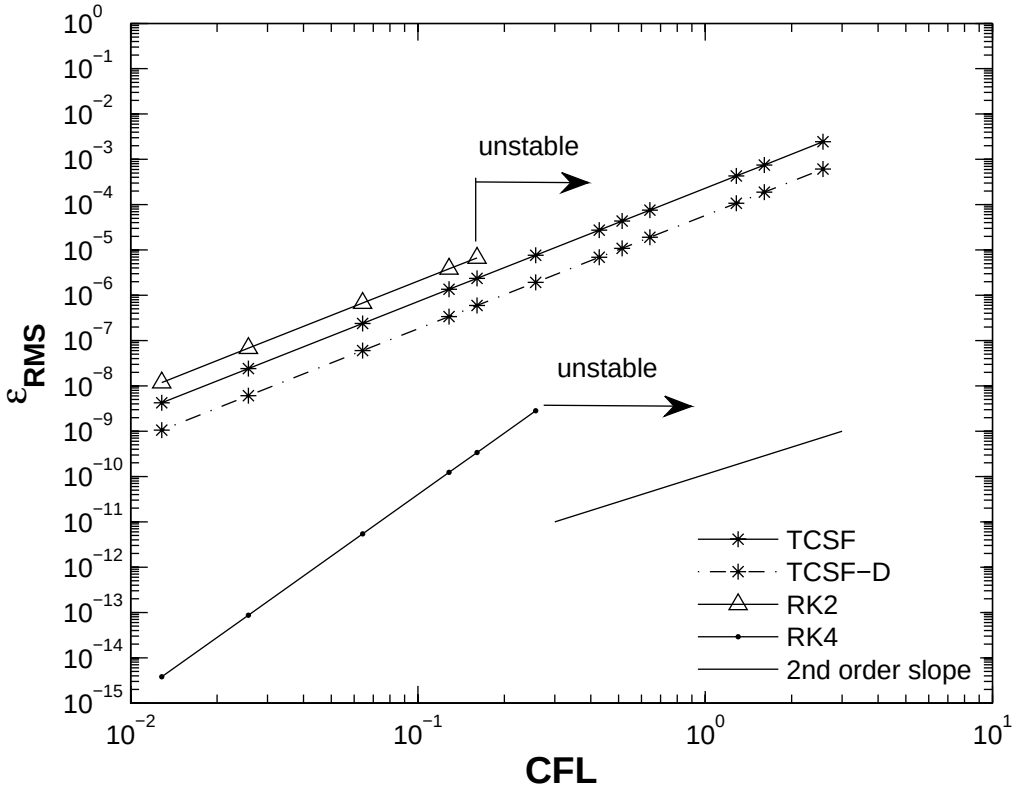


Figure 4.6 $\varepsilon_{RMS}$ vs. CFL number of TCSF, RK2 and RK4 methods for solution of 1D conservative Burgers' equation, where $\nu = 0.05$ $\Delta x$ =1/50, and $\Delta t = 0.3/\Gamma$ with $\Gamma \in \{500, 200, 100, 80, 50, 30, 25, 20, 10, 8, 5\}$.

***Computational Requirements***

To gain a better understanding of the computational requirements of the different methods, it is useful to compare the ideal operation counts using tridiagonal solutions of the **Ax** = **b** problems. We focus on idealized operation counts rather than CPU time as we solve the above discrete problems using a MATLAB script, for which the software overhead tends to dominate the model runtime. Based on algorithms in Press et al (1992) and Pozrikidis et al (2005, pg50), a tridiagonal solution ideally requires 8Q operations and a pentadiagonal solution requires 24Q operations, where Q is the number of grid points. Here we do not distinguish between addition, multiplication or division operations. If the Newton method requires R outer iterations for convergence, the Picard method should require $R^2$ outer iterations for the same convergence level. A tridiagonal Matrix algorithm (TDMA) is used for each method. A comparison of operations per grid point for the different methods is shown in Figure 4.7. Although these operation counts are idealized and only for the 1D Burgers' equation with direct inner solutions, they represent the general trends that should be expected in any comparison of nonlinear solution methods. The operation counts for both the TCSF and TCSF-D methods are identical because the solution processes for both methods are the same although they are in a reversed manner. For this 1D test case with an analytical Jacobian, the TCS method does not have any computational efficiency advantage over Local Linearization. However, the TCS derivation does not require a functional Jacobian computation, so extending the method to multiple dimensions and multiple variables is arguably easier, an idea that will be explored in next chapter.

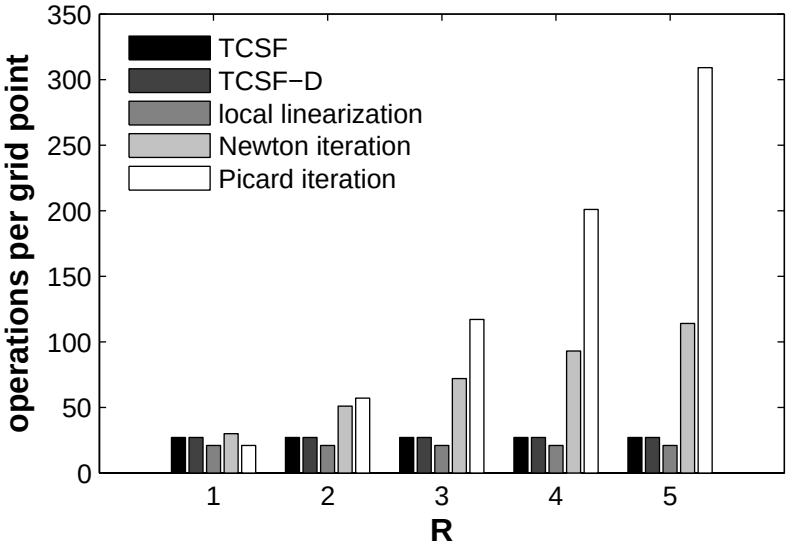

Figure 4.7 Ideal operations per grid point for one time step using various nonlinear solution methods for 1D conservative Burgers' equation. R is the number of outer iterations taken by Newton method. It is assumed that the Picard method convergences in $R^2$ outer iterations.

## 4.2 APPLICATIONS OF THE TCS METHOD TO THE 1D NON-CONSERVATIVE BURGERS' EQUATION

Unlike in the conservative Burgers' equation, the TCSF and TCSG formats remain distinct for the non-conservative Burgers' equation. Applying the TCS method to the non-conservative Burgers' equation can create four different discretizations: the TCSF, TCSG, TCSF-D and the TCSG-D. In this section, we analyze all four different discretizations in the same vein as in the conservative case.

### 4.2.1 Discrete formats of the 1D Non-Conservative Burger's equation using various methods

***The TCSF Discrete Form***

The non-conservative Burgers' equation can be written as:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2} \tag{4.53}$$

Introducing the approximation in the flux term results in the TCSF form as below:

$$u^* = u^n + \frac{\Delta t}{2}\left\{-u^n\frac{\partial u^*}{\partial x} + \nu\frac{\partial^2 u^*}{\partial x^2}\right\} \tag{4.54}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left\{-u^{n+1}\frac{\partial u^*}{\partial x} + \nu\frac{\partial^2 u^*}{\partial x^2}\right\} \tag{4.55}$$

Similar to the analysis in the conservative Burger's equation, in the following discussion, we examine each method in a generic matrix equation, **Ax=b**. A central difference discretization is used for all the spatial derivatives. The same parameters such as the number of spatial grid points Q, the viscous scale $\gamma$, the grid cell CFL number $C_i^L$, the diffusion operator $D_i^L$, and the nonlinear adjective gradient operator $G_i^L$ are also used in the following discussion. In the first step of the TCSF,

$$\mathbf{A}=\begin{bmatrix} 1+\gamma & -\frac{\gamma}{2}+\frac{C_2^n}{4} & 0 & 0 & 0 \\ -\frac{\gamma}{2}-\frac{C_1^n}{4} & 1+\gamma & -\frac{\gamma}{2}+\frac{C_3^n}{4} & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -\frac{\gamma}{2}-\frac{C_{Q-2}^n}{4} & 1+\gamma & -\frac{\gamma}{2}+\frac{C_Q^n}{4} \\ 0 & 0 & 0 & -\frac{\gamma}{2}-\frac{c_{Q-1}}{4} & 1+\gamma \end{bmatrix} \tag{4.56}$$

$$\mathbf{x}=\left[u_1^*,u_2^*,...u_Q^*\right]^T \tag{4.57}$$

$$\mathbf{b}=\begin{bmatrix} u_1^n+\left(\frac{\gamma}{2}+\frac{C_0^n}{4}\right)u_0^* \\ u_2^n \\ u_3^n \\ \vdots \\ u_{Q-1}^n \\ u_Q^n+\left(\frac{\gamma}{2}-\frac{C_{Q+1}^n}{4}\right)u_{Q+1}^* \end{bmatrix} \tag{4.58}$$

The second step of the TCSF is in an explicit format, which can be represented as

$$u_i^{n+1}=\frac{u_i^*+\frac{\gamma}{2}\left(u_{i+1}^*-2u_i^*+u_{i-1}^*\right)}{1+\frac{1}{4}\left(c_{i+1}^*-c_{i-1}^*\right)} \quad ; \quad i=\{1,2,...Q\} \tag{4.59}$$

Thus, the TCSF solution for the 1D non-conservative Burgers' equation only needs one tridagonal linear problem at the first step and an explicit solution for $u^{n+1}$ at the second step.

***The TCSG Discrete Form***

If the linear approximation is introduced in the gradient rather than the flux part, the resulting discrete equations are the TCSG form:

$$u^*=u^n+\frac{\Delta t}{2}\left\{-u^*\frac{\partial u^n}{\partial x}+\nu\frac{\partial^2 u^*}{\partial x^2}\right\} \tag{4.60}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left\{-u^* \frac{\partial u^{n+1}}{\partial x} + \nu \frac{\partial^2 u^*}{\partial x^2}\right\} \tag{4.61}$$

The first step of the TCSG has

$$\mathbf{A} = \begin{bmatrix} 1+\frac{\left(C_3^n - C_1^n\right)}{4}+\gamma & -\frac{\gamma}{2} & & & 0 \\ -\frac{\gamma}{2} & 1+\frac{\left(C_4^n - C_2^n\right)}{4}+\gamma & -\frac{\gamma}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma}{2} & 1+\frac{\left(C_Q^n - C_{Q-2}^n\right)}{4}+\gamma & -\frac{\gamma}{2} \\ 0 & & & -\frac{\gamma}{2} & 1+\frac{\left(C_{Q+1}^n - C_{Q-1}^n\right)}{4}+\gamma \end{bmatrix} \tag{4.62}$$

$$\mathbf{x} = \left[u_1^*, u_2^*, \ldots u_Q^*\right]^T \tag{4.63}$$

and

$$\mathbf{b} = \begin{bmatrix} u_1^n + \frac{\gamma}{2}u_1^* \\ u_2^n \\ u_3^n \\ \vdots \\ u_{Q-1}^n \\ u_Q^n + \frac{\gamma}{2}u_{Q+1}^* \end{bmatrix} \tag{4.64}$$

In the second step of the TCSG,

$$\mathbf{A}=\begin{bmatrix} 1 & \frac{C_1^*}{4} & & & & 0 \\ -\frac{C_2^*}{4} & 1 & \frac{C_2^*}{4} & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\frac{C_{Q-1}^*}{4} & 1 & \frac{C_{Q-1}^*}{4} & \\ 0 & & & -\frac{C_Q^*}{4} & 1 & \frac{C_Q^*}{4} \end{bmatrix} \tag{4.65}$$

$$\mathbf{x}=\left[u_1^{n+1},u_2^{n+1},...u_Q^{n+1}\right]^T \tag{4.66}$$

and

$$\mathbf{b}=\begin{bmatrix} u_1^*+\frac{D_1^*}{2}+\frac{C_1^*}{4}u_1^{n+1} \\ u_2^*+\frac{D_2^*}{2} \\ \vdots \\ u_{Q-1}^*+\frac{D_{Q-1}^*}{2} \\ u_Q^*+\frac{D_Q^*}{2}-\frac{C_Q^*}{4}u_Q^{n+1} \end{bmatrix} \tag{4.67}$$

Thus the TCSG method for the 1D non-conservative Burgers' equation requires solution of two tridiagonal linear problems at each time step.

***The TCSF-D Discrete Form***

If we further split the diffusion term in the TCSF, we obtain the TCSF-D form. The TCSF-D form for Equation (4.53) can be written as,

$$u^*=u^n+\frac{\Delta t}{2}\left\{-u^n\frac{\partial u^*}{\partial x}+\nu\frac{\partial^2 u^n}{\partial x^2}\right\} \tag{4.68}$$

$$u^{n+1}=u^*+\frac{\Delta t}{2}\left\{-u^{n+1}\frac{\partial u^*}{\partial x}+\nu\frac{\partial^2 u^{n+1}}{\partial x^2}\right\} \tag{4.69}$$

In the first step,

$$\mathbf{A}=\begin{bmatrix} 1 & \frac{C_1^n}{4} & & & & 0 \\ -\frac{C_2^n}{4} & 1 & \frac{C_2^n}{4} & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\frac{C_{Q-1}^n}{4} & 1 & \frac{C_{Q-1}^n}{4} & \\ 0 & & & -\frac{C_Q^n}{4} & 1 & \frac{C_Q^n}{4} \end{bmatrix} \tag{4.70}$$

$$\mathbf{x}=\left[u_1^*, u_2^*, \ldots u_Q^*\right]^T \tag{4.71}$$

and

$$\mathbf{b}=\begin{bmatrix} u_1^n+\frac{D_1^n}{2}+\frac{C_1^n}{4}u_1^* \\ u_2^n+\frac{D_2^n}{2} \\ \vdots \\ u_{Q-1}^n+\frac{D_{Q-1}^n}{2} \\ u_Q^n+\frac{D_Q^n}{2}-\frac{C_Q^n}{4}u_Q^* \end{bmatrix} \tag{4.72}$$

In the second step,

$$\mathbf{A}=\begin{bmatrix} 1+\frac{\left(C_3^*-C_1^*\right)}{4}+\gamma & -\frac{\gamma}{2} & & & 0 \\ -\frac{\gamma}{2} & 1+\frac{\left(C_4^*-C_2^*\right)}{4}+\gamma & -\frac{\gamma}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma}{2} & 1+\frac{\left(C_Q^*-C_{Q-2}^*\right)}{4}+\gamma & -\frac{\gamma}{2} \\ 0 & & & -\frac{\gamma}{2} & 1+\frac{\left(C_{Q+1}^*-C_{Q-1}^*\right)}{4}+\gamma \end{bmatrix} \tag{4.73}$$

$$\mathbf{x}=\left[u_1^{n+1}, u_2^{n+1}, \ldots u_Q^{n+1}\right]^T \tag{4.74}$$

and

$$\mathbf{b} = \begin{bmatrix} u_1^* + \frac{\gamma}{2} u_1^{n+1} \\ u_2^* \\ u_3^* \\ \vdots \\ u_{Q-1}^* \\ u_Q^* + \frac{\gamma}{2} u_{Q+1}^{n+1} \end{bmatrix} \tag{4.75}$$

It can be observed that the TCSF-D method reverses the solution process of the TCSG method. The first step in TCSF-D is in the same structure as the second step in TCSG; the second step in TCSF-D is in the same structure as the first step in TCSG.

***The TCSG-D Discrete Form***

The TCSG-D is obtained by further splitting the diffusion term in the TCSG method written as

$$u^* = u^n + \frac{\Delta t}{2}\left\{-u^* \frac{\partial u^n}{\partial x} + \nu \frac{\partial^2 u^n}{\partial x^2}\right\} \tag{4.76}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left\{-u^* \frac{\partial u^{n+1}}{\partial x} + \nu \frac{\partial^2 u^{n+1}}{\partial x^2}\right\} \tag{4.77}$$

$u^*$ in the first step of Equation (4.76) can be explicitly calculated,

$$u_i^* = \frac{u_i^n + \frac{\gamma}{2}\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right)}{1 + \frac{1}{4}\left(c_{i+1}^n - c_{i-1}^n\right)} \quad ; \quad i = \{1, 2, ...Q\} \tag{4.78}$$

The second step is a matrix equation with

$$\mathbf{A}=\begin{bmatrix} 1+\gamma & -\frac{\gamma}{2}+\frac{C_2^*}{4} & 0 & 0 & 0 \\ -\frac{\gamma}{2}-\frac{C_1^*}{4} & 1+\gamma & -\frac{\gamma}{2}+\frac{C_3^*}{4} & 0 & 0 \\ 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -\frac{\gamma}{2}-\frac{C_{Q-2}^*}{4} & 1+\gamma & -\frac{\gamma}{2}+\frac{C_Q^*}{4} \\ 0 & 0 & 0 & -\frac{\gamma}{2}-\frac{C_{Q-1}^*}{4} & 1+\gamma \end{bmatrix} \tag{4.79}$$

$$\mathbf{x}=\left[u_1^{n+1},u_2^{n+1},...u_Q^{n+1}\right]^T \tag{4.80}$$

$$\mathbf{b}=\begin{bmatrix} u_1^*+\left(\frac{\gamma}{2}+\frac{C_0^*}{4}\right)u_0^{n+1} \\ u_2^* \\ u_3^* \\ \vdots \\ u_{Q-1}^* \\ u_Q^*+\left(\frac{\gamma}{2}-\frac{C_{Q+1}^*}{4}\right)u_{Q+1}^{n+1} \end{bmatrix} \tag{4.81}$$

The TCSG-D method reverses the solution process of the TCSF. The first step in TCSG-D is in the same structure as in the second step in TCSF; the second step in TCSG-D is in the same structure as in the first step in TCSF.

Similar to the last section, we first apply CN 2nd-order discretization to Equation (4.53), such that Equation (4.53) is

$$\begin{aligned} u_i^{n+1}=u_i^n-\frac{\Delta t}{2}\Bigg\{ & u_i^n\frac{u_{i+1}^n-u_{i-1}^n}{2\Delta x}-\nu\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{\Delta x^2} \\ & +u_i^{n+1}\frac{u_{i+1}^{n+1}-u_{i-1}^{n+1}}{2\Delta x}-\nu\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\Delta x^2}\Bigg\} \end{aligned} \tag{4.82}$$

Equation (4.82) is then linearized using conventional Picard, Newton and local linearization methods.

***Picard iteration***

Applying Picard iteration to Equation (4.82), the linearized equation can be written as,

$$\begin{aligned}\left(u_i^{n+1}\right)^{k+1} = u_i^n - \frac{\Delta t}{2}\Bigg\{ & u_i^n \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - \nu \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2} \\ & + \left(u_i^{n+1}\right)^k \frac{\left(u_{i+1}^{n+1}\right)^{k+1} - \left(u_{i-1}^{n+1}\right)^{k+1}}{2\Delta x} \\ & - \nu \frac{\left(u_{i+1}^{n+1}\right)^{k+1} - 2\left(u_i^{n+1}\right)^{k+1} + \left(u_{i-1}^{n+1}\right)^{k+1}}{\Delta x^2} \Bigg\}\end{aligned} \tag{4.83}$$

where the superscripts n and k indicate the time step and iteration number respectively. with

$$\mathbf{A}^k = \begin{bmatrix} 1+\gamma & -\frac{\gamma}{2} + \frac{C_3^k}{4} & & & 0 \\ -\frac{\gamma}{2} - \frac{C_2^k}{4} & 1+\gamma & -\frac{\gamma}{2} + \frac{C_4^k}{4} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma}{2} - \frac{C_{Q-2}^k}{4} & 1+\gamma & -\frac{\gamma}{2} + \frac{C_Q^k}{4} \\ 0 & & & -\frac{\gamma}{2} - \frac{C_M^k}{4} & 1+\gamma \end{bmatrix} \tag{4.84}$$

$$\mathbf{x}^{k+1} = \left[\left(u_1^{n+1}\right)^{k+1}, \left(u_2^{n+1}\right)^{k+1}, \ldots \left(u_Q^{n+1}\right)^{k+1}\right]^T \tag{4.85}$$

$$\mathbf{b}=\begin{bmatrix} u_1^n-\frac{D_1^n}{2}-\frac{G_1^n}{4}+\frac{\gamma}{2}+\frac{C_1^k}{4} \\ u_2^n-\frac{D_2^n}{2}-\frac{G_2^n}{4} \\ \vdots \\ u_{Q-1}^n-\frac{D_{Q-1}^n}{2}-\frac{G_{Q-1}^n}{4} \\ u_Q^n-\frac{D_Q^n}{2}-\frac{G_Q^n}{4}+\frac{\gamma}{2}-\frac{C_Q^k}{4} \end{bmatrix} \tag{4.86}$$

$$\mathbf{c}^k=\begin{bmatrix} \frac{\gamma}{2}+\frac{C_1^k}{4} \\ 0 \\ \vdots \\ 0 \\ \frac{\gamma}{2}-\frac{C_{Q+1}^k}{4} \end{bmatrix} \tag{4.87}$$

***Newton Iteration***

Applying the Newton iteration to Equation (4.82), the linearized equation can be written as:

$$\left(\frac{\partial g_i}{\partial u_j^{n+1}}\right)^k\left(\Delta u_j^{n+1}\right)^{k+1}=-g_i^k \tag{4.88}$$

with

$$\begin{aligned} g_i\left(u_{i-1}^{n+1},u_i^{n+1},u_{i+1}^{n+1}\right)&=u_i^{n+1}-u_i^n \\ &+\frac{\Delta t}{2}\left\{u_i^{n+1}\frac{u_{i+1}^{n+1}-u_{i-1}^{n+1}}{2\Delta x}-\nu\frac{u_{i+1}^{n+1}-2u_i^{n+1}+u_{i-1}^{n+1}}{\Delta x^2}\right. \\ &\left.+u_i^n\frac{u_{i+1}^n-u_{i-1}^n}{2\Delta x}-\nu\frac{u_{i+1}^n-2u_i^n+u_{i-1}^n}{\Delta x^2}\right\}=0 \end{aligned} \tag{4.89}$$

and

$$\left(\Delta u^{n+1}\right)^k=\left(u^{n+1}\right)^{k+1}-\left(u^{n+1}\right)^k \tag{4.90}$$

Equation (4.88) is a matrix equation **Ax=b** with

$$\mathbf{A}^{k}=\begin{bmatrix} 1+\frac{G_2^k}{4}+\gamma & -\frac{\gamma}{2}+\frac{C_3^k}{4} & & & 0 \\ -\frac{\gamma}{2}-\frac{C_2^k}{4} & 1+\frac{G_3^k}{4}+\gamma & -\frac{\gamma}{2}+\frac{C_4^k}{4} & & \\ & \ddots & \ddots & \ddots & \\ & & -\frac{\gamma}{2}-\frac{C_{Q-2}^k}{4} & 1+\frac{G_{Q-1}^k}{4}+\gamma & -\frac{\gamma}{2}+\frac{C_Q^k}{4} \\ 0 & & & \frac{\gamma}{2}-\frac{C_{Q-1}^k}{4} & -1+\frac{G_Q^k}{4}+\gamma \end{bmatrix} \tag{4.91}$$

$$\mathbf{x}^{k+1}=\left[\left(\Delta u_1^{n+1}\right)^{k+1},\left(\Delta u_2^{n+1}\right)^{k+1},\ldots\left(\Delta u_Q^{n+1}\right)^{k+1}\right]^{T} \tag{4.92}$$

$$\mathbf{b}=-\begin{bmatrix} u_1^n+\frac{G_1^n}{4}-\frac{D_1^n}{2} \\ u_2^n+\frac{G_2^n}{4}-\frac{D_2^n}{2} \\ \vdots \\ u_{Q-1}^n+\frac{G_{Q-1}^n}{4}-\frac{D_{Q-1}^n}{2} \\ u_Q^n+\frac{G_Q^n}{4}-\frac{D_Q^n}{2} \end{bmatrix} \tag{4.93}$$

$$\mathbf{C}^{k}=\begin{bmatrix} u_1^k+\frac{G_1^k}{4}-\frac{D_1^k}{2}+\frac{\gamma}{2}+\frac{C_0^k}{4} \\ u_2^k+\frac{G_2^k}{4}-\frac{D_2^k}{2} \\ \vdots \\ u_{Q\text{-}1}^k+\frac{G_{Q\text{-}1}^k}{4}-\frac{D_{Q\text{-}1}^k}{2} \\ u_Q^k+\frac{G_Q^k}{4}-\frac{D_Q^k}{2}+\frac{\gamma}{2}-\frac{C_{Q+1}^k}{4} \end{bmatrix} \tag{4.94}$$

***Local linearization***

Applying Local linearization to Equation (4.82), the linearized equation system can be written as:

$$\left[1-\frac{1}{2}\Delta t\left(\frac{\partial F_i}{\partial u_j}\right)^n\right]\Delta u_j^n=\Delta t F_i^n \tag{4.95}$$

where

$$F_i^L\left(u_{i-1}^L,u_i^L,u_{i+1}^L\right)=-u_i^L\frac{u_{i+1}^L-u_{i-1}^L}{2\Delta x}+\nu\frac{u_{i+1}^L-2u_i^L+u_{i-1}^L}{\Delta x^2} \tag{4.96}$$

for $L\in\{n,n+1\}$ and

$$\Delta u_i^n=u_i^{n+1}-u_i^n \tag{4.97}$$

Therefore, Equation (4.95) is a matrix equation **Ax=b** with

$$\mathbf{A}=\begin{bmatrix} 1+\frac{G_2}{4}+\gamma & -\frac{\gamma}{2}+\frac{C_3^n}{4} & & & 0\\ -\frac{\gamma}{2}-\frac{C_2^n}{4} & 1+\frac{G_3}{4}+\gamma & -\frac{\gamma}{2}+\frac{C_4^n}{4} & & \\ & \ddots & \ddots & \ddots & \\ & -\frac{\gamma}{2}-\frac{C_{Q-2}^n}{4} & 1+\frac{G_{Q-1}}{4}+\gamma & -\frac{\gamma}{2}+\frac{C_Q}{4} & \\ 0 & & -\frac{\gamma}{2}-\frac{C_{Q-1}}{4} & 1+\frac{G_Q}{4}+\gamma & \end{bmatrix} \tag{4.98}$$

$$\mathbf{x}^{n+1}=\left[\Delta u_1^{n+1},\Delta u_2^{n+1},\ldots\Delta u_Q^{n+1}\right]^T \tag{4.99}$$

$$\mathbf{b}=\begin{bmatrix} -\frac{1}{2}G_1^n+D_1^n+\frac{\gamma}{2}+\frac{C_0^n}{4}\\ -\frac{1}{2}G_2^n+D_2^n\\ \vdots\\ -\frac{1}{2}G_{Q-1}^n+D_{Q-1}^n\\ -\frac{1}{2}G_Q^n+D_Q^n+\frac{\gamma}{2}-\frac{C_{Q+1}^n}{4}\end{bmatrix} \tag{4.100}$$

### 4.2.2 Results and Discussion

***Accuracy***

As in the analysis of the conservative Burgers' equation, we first locate the maximum absolute error for all four discrete forms of the TCS method. As shown in Figure 4.8, the maximum absolute error occurs at t=0.6 for all four forms. The exact solution of the Burgers' equation at t=0.6 can be found in Figure 4.3.

To evaluate the model error, we examine performance with seven different time steps over the range, $0.012 \le \Delta t \le 0.12$, and with grid cell spacing of $\Delta x \in \left\{2\times10^{-2}, 5\times10^{-3}, 1.25\times10^{-3}\right\}$. As a result, the test conditions cover the range, $0.41 \le CFL \le 66.2$. Similar to section 4.1, the fine-time step solution $\tilde{u}$ is used in the analysis of RMS error, $L_2$ norm and $L_\infty$ norm. Here $\tilde{u}$ is the model solution of an extremely fine time step such that $\Delta t = 0.6/10^4$, which is 0.5% of the smallest time step used in the other simulations. The $\varepsilon_{RMS}$, $L_2$ norm, and $L_\infty$ norm associated with different methods over the tested ranges of CFL numbers are shown in Figure 4.9, 4.10 and 4.11 respectively. All methods are proven to be $2^{nd}$-order temporal accurate for fixed $\Delta x$ and decreasing $\Delta t$. For the same $\Delta x$ and $\Delta t$, it can be seen that the difference in relative error among all methods is within one order of magnitude except for the TCSF method. The TCSF method has a noticeably smaller error than all the other methods, and about 1 magnitude smaller error than the TCSG-D form.

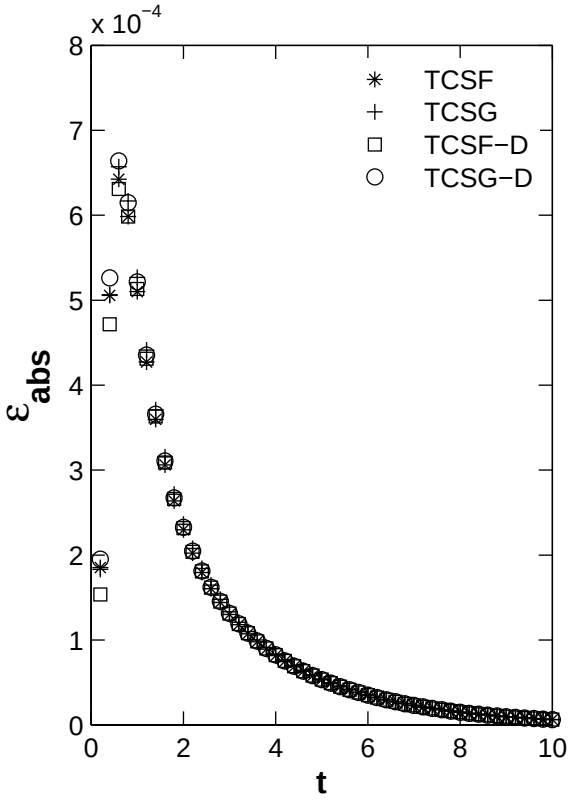


Figure 4.8 Absolute error time evolution for numerical solutions of 1D non-conservative Burgers' equation for the TCSF and TCSG methods using $\{\Delta t = 0.01, \Delta x = 1/50, \nu = 0.05\}$ where $\varepsilon_{abs}(t) = \left| \frac{1}{50} \sum_{i=1}^{50} u_{model}(x_i, t) - u_{analytical}(x_i, t) \right|$ and $u_{analytical}(x, t)$ is numerically calculated from Equation (4.46) with K=30.

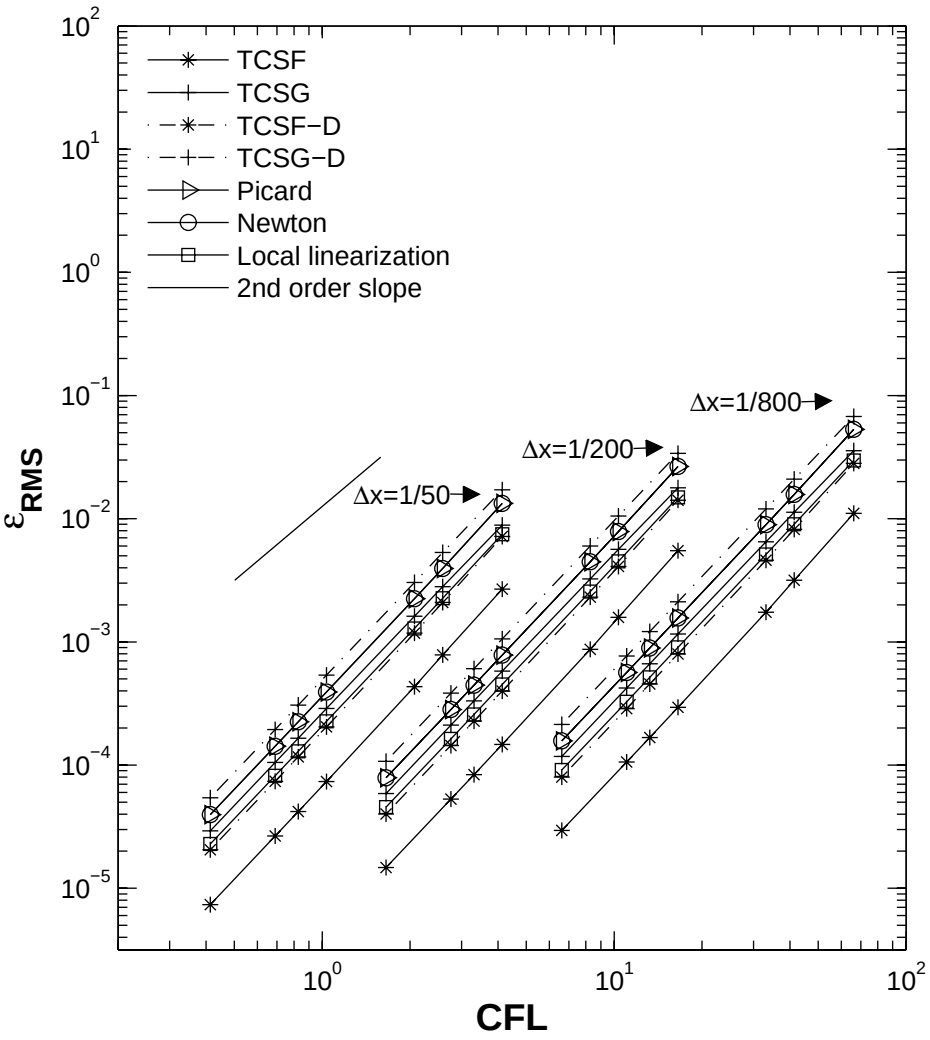


Figure 4.9 $\varepsilon_{RMS}$ vs. CFL numbers of various methods for solution of 1D non-conservative Burgers' equation at different CFL numbers, where $\nu=0.05$, $\Delta x=1/50$ and $\Delta t=0.6/\Gamma$ with $\Gamma \in \{50, 30, 25, 20, 10, 8, 5\}$.

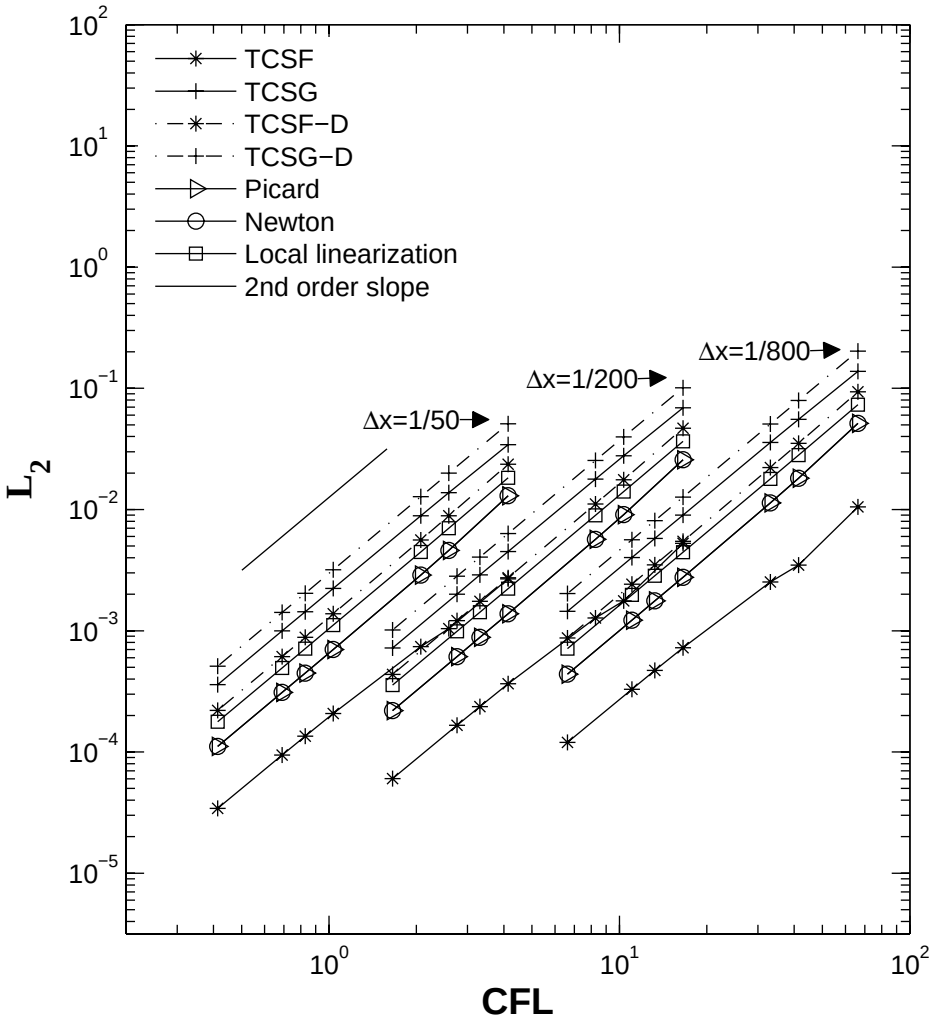


Figure 4.10 $L_2$ norm vs. CFL numbers of various methods for solution of 1D non-conservative Burgers' equation at different CFL numbers, where $\nu = 0.05$, $\Delta x = 1/50$ and $\Delta t = 0.6/\Gamma$ with $\Gamma \in \{50, 30, 25, 20, 10, 8, 5\}$.

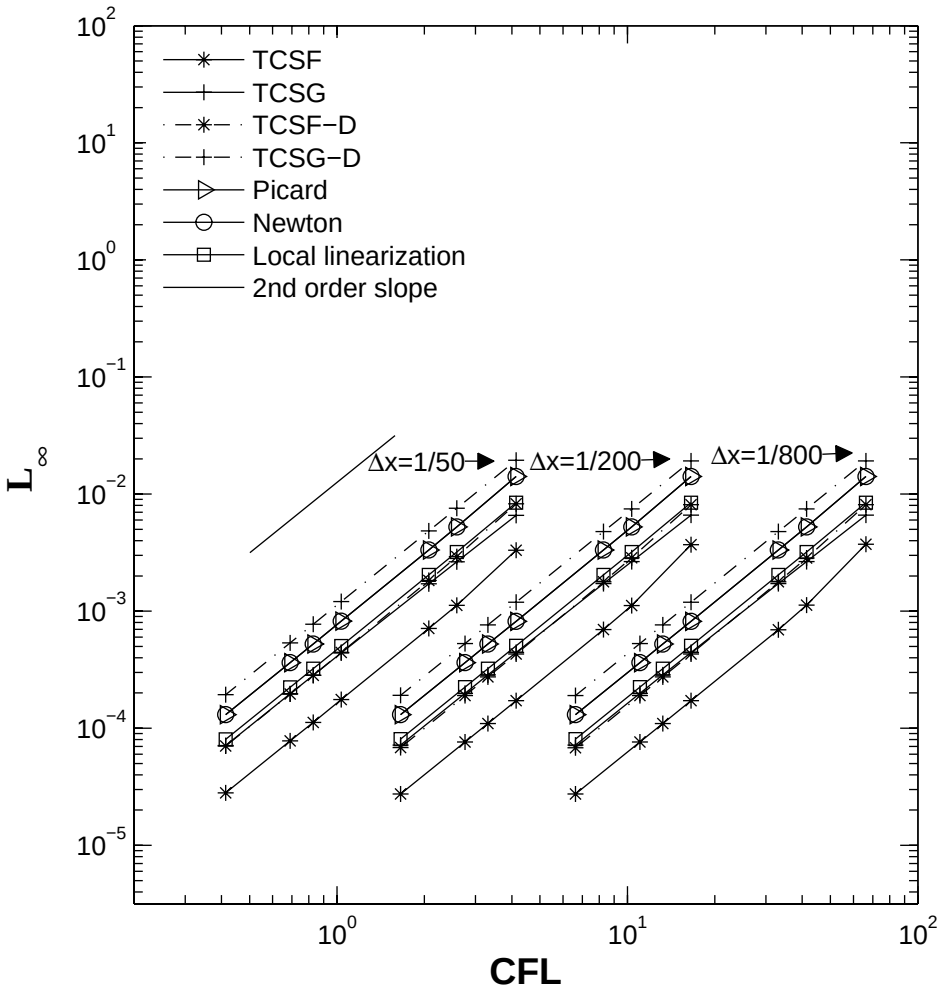


Figure 4.11 $L_\infty$ norm vs. CFL numbers of various methods for solution of 1D non-conservative Burgers' equation at different CFL numbers, where $\nu=0.05$, $\Delta x=1/50$ and $\Delta t=0.6/\Gamma$ with $\Gamma \in \{50, 30, 25, 20, 10, 8, 5\}$.

***Stability***

As shown in Figure 4.9, Figure 4.10 and Figure 4.11, all the methods remain stable over the full range of tested conditions. The RK2 and RK4 methods are also used to solve this 1D non-conservative Burgers' equation. The RMS error over a range of CFL for the TCSF, TCSG, TCSF-D, TCSG-D, RK2 and RK4 methods is plotted in

Figure 4.12. In this comparison, $0.0006 \le \Delta t \le 0.06$ and $\Delta x = 0.02$ provide test conditions $0.013 \le CFL \le 2.57$. The RK methods become unstable when CFL>0.1. However, the TCS methods remain stable in the whole range of CFL number.

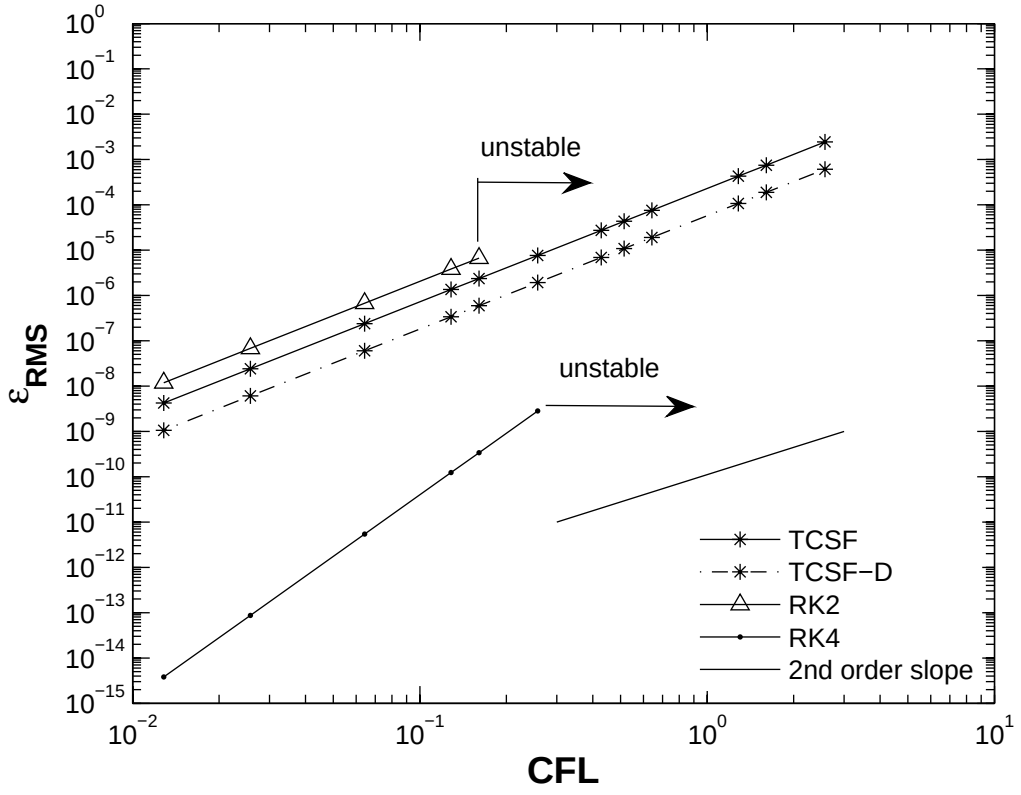


Figure 4.12 Temporal accuracy of the TCS and RK methods for solution of 1D non-conservative Burgers' equation at different CFL numbers, where $\nu = 0.05$, $\Delta x = 1/50$ and $\Delta t = 0.6/\Gamma$ with $\Gamma \in \{500, 200, 100, 80, 50, 30, 25, 20, 10, 8, 5\}$.

***Computational requirement***

The same assessment of operation counts is used in this non-conservative Burgers' equation. All the methods perform similar to the non-conservative Burgers' equation. The TCSF and TCSG-D have the identical operation counts because they have the same solution process only in a reverse mode. The same correspondence can be found between the TCSG and TCSF-D for the same reason. In this 1D non-conservative Burgers' equation test case, the TCSF and TCSG-D have the least operation counts.

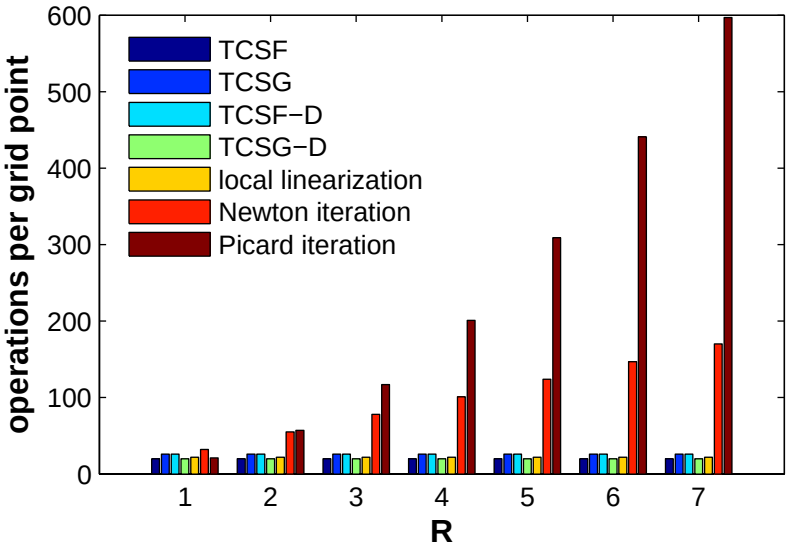


Figure 4.13 Ideal operations per grid point for one time step using various nonlinear solution methods for 1D Non-conservative Burgers equation. R is the number of outer iterations taken by Newton method. It is assumed that the Picard method convergences in $R^2$ outer iterations.

## 4.3 Application of the TCS Method to a 1D Nonlinear Ordinary Differential Equation

The TCS method can be extended to any equation system with quadratic nonlinearities. To test this general capability, we use a nonlinear ODE as our example:

$$\frac{dy}{dt} + y(1-y) = 0 \tag{4.101}$$

For the initial condition $y(0) = 1/2$, Equation (4.101) has the analytical solution (Moin 2001)

$$y = \left[1 + \exp(t)\right]^{-1} \tag{4.102}$$

### 4.3.1 The TCS discretizations

***The TCSF Discretization***

As in the 1D conservative Burgers' equation, the TCSF and TCSG are the same for Equation (4.101). Applying the TCSF method to Equation (4.101) provides

$$y^* = y^n + \left(y^n y^* - y^*\right)\frac{\Delta t}{2} \tag{4.103}$$

$$y^{n+1} = y^* + \left(y^{n+1} y^* - y^*\right)\frac{\Delta t}{2} \tag{4.104}$$

The discrete form can be written from Equation (4.103) and (4.104) as

$$y^* = \frac{y^n}{1 - \frac{\Delta t}{2}\left[y^n - 1\right]} \tag{4.105}$$

$$y^{n+1} = \frac{y^* - \frac{\Delta t}{2} y^*}{1 - \frac{\Delta t}{2} y^*} \tag{4.106}$$

***The TCSF-D Discretization***

Similarly, the TCSF-D discrete format for Equation (4.101) is written as

$$y^* = \frac{y^n - y^n \frac{\Delta t}{2}}{1 - y^n \frac{\Delta t}{2}} \tag{4.107}$$

$$y^{n+1} = \frac{y^*}{1 - \frac{\Delta t}{2}\left(y^* - 1\right)} \tag{4.108}$$

It can be seen that Equation (4.108) has the same structure as Equation (4.105) and Equation (4.107) has the same structure as Equation (4.106). Therefore, the TCSF and TCSF-D discrete forms for Equation (4.101) are similar but reversed.

### 4.3.2 Conventional linearization methods

Crank-Nicolson (CN) discretized Equation (4.101) is,

$$y^{n+1} = y^n + \frac{\Delta t}{2}\left(y^n - 1\right)y^n + \frac{\Delta t}{2}\left(y^{n+1} - 1\right)y^{n+1} \tag{4.109}$$

To linearize the nonlinear term on right hand side of Equation (4.109), we use the Newton iteration, Picard iteration and local linearization methods.

***Newton method***

Applying the Newton iteration approach, Equation (4.109) can be written as:

$$\left(y^{n+1}\right)^{k+1} = \left(y^{n+1}\right)^{k} - \frac{g\left(y^{n+1}\right)^{k}}{\left(\frac{\partial g}{\partial y^{n+1}}\right)_{k}} \tag{4.110}$$

where the outer superscript indicates the outer iteration number and

$$g\left(y^{n+1}\right) = y^{n+1} - \frac{\Delta t}{2}\left(y^{n+1}-1\right)y^{n+1} - y^{n} - \frac{\Delta t}{2}\left(y^{n}-1\right)y^{n} \tag{4.111}$$

Substituting Equation (4.111) into Equation (4.110), the original ODE in the Newton linearized CN discretized form is

$$\left(y^{n+1}\right)^{k+1} = \left(y^{n+1}\right)^{k} - \frac{\left(y^{n+1}\right)^{k} - \frac{\Delta t}{2}\left[\left(y^{n+1}\right)^{k}-1\right]\left(y^{n+1}\right)^{k} - y^{n} - \frac{\Delta t}{2}\left(y^{n}-1\right)y^{n}}{1-\frac{\Delta t}{2}\left[2\left(y^{n+1}\right)^{k}-1\right]} \tag{4.112}$$

$y_0^{n+1} = y^n$ can be used as the initial condition to start the solution.

***Picard method***

A second conventional approach, Picard iteration, in Equation (4.109), results in

$$\left(y^{n+1}\right)^{k+1} = y^{n} + \frac{\Delta t}{2}\left(y^{n}-1\right)y^{n} + \frac{\Delta t}{2}\left[\left(y^{n+1}\right)^{k}-1\right]\left(y^{n+1}\right)^{k} \tag{4.113}$$

The same initial condition, $y_0^{n+1} = y^n$ can be used to start the solution.

***Local linearization***

Applying the third conventional method, local linearization method, Equation (4.109) can be written as

$$y^{n+1} = y^{n} + \frac{\Delta t f^{n}}{\left[1-\frac{\Delta t}{2}\left(\frac{\partial f}{\partial y}\right)^{n}\right]} \tag{4.114}$$

where

$$f = \left(y-1\right)y \tag{4.115}$$

Substituting Equation (4.115) into Equation (4.114) provides the locally-linearized Equation (4.109) as

$$y^{n+1} = y^n + \frac{\Delta t\left(y^n - 1\right)y^n}{\left[1 - \frac{\Delta t}{2}\left(2y^n - 1\right)\right]} \tag{4.116}$$

In this specific case, the local linearization, TCSF and TCSF-D are mathematically equivalent. If we substitute Equation (4.105) to Equation (4.106) and substitute Equation (4.107) to Equation (4.108), we will have two expressions which are exactly the same as Equation (4.116).

### 4.3.3 Comparisons between the TCS and other methods

The evolution of the absolute errors for two TCS methods applied to the ODE is shown in Figure 4.14. The maximum absolute errors for both TCS discrete formats occur at time t=1.5.

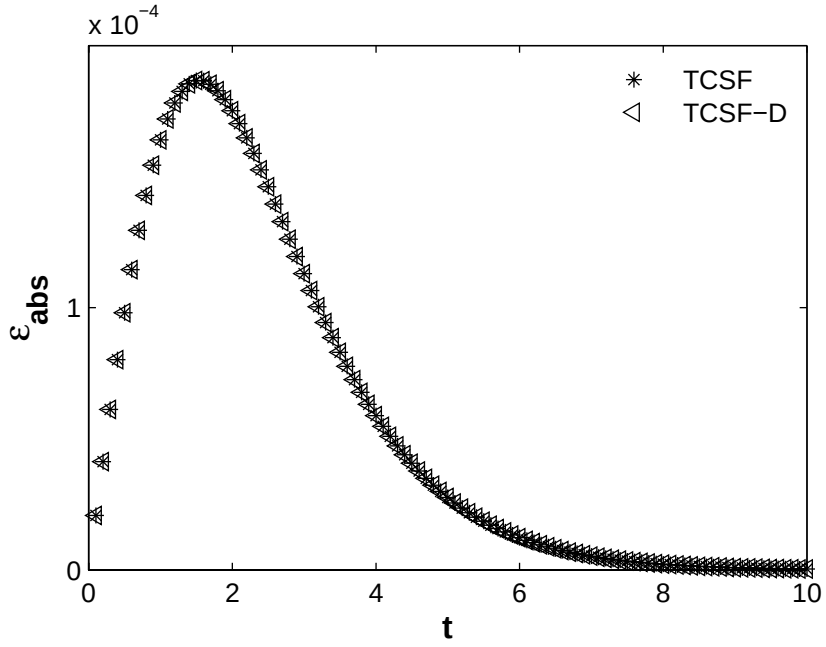


Figure 4.14 Time evolution of absolute errors for TCS methods applied to the ODE, Equation (4.101), for $\Delta t = 0.1$, where $\varepsilon_{abs}(t) = \left|y_{model}(t) - y_{analytical}(t)\right|$.

Figure 4.15 provides a comparison of root-mean-square (RMS) error over $\Gamma$ time steps prior to maximum error, defined as

$$\varepsilon_{RMS} \equiv \left\{ \frac{1}{\Gamma} \sum_{i=1}^{\Gamma} \left[ y_{model}(t_i) - y_{analytical}(t_i) \right]^2 \right\}^{1/2} \tag{4.117}$$

where $t_{max} = 1.5$ and $\Gamma \in \{10, 50, 100, 200, 500, 600, 800, 1000, 1200, 1500, 2000\}$. All tested methods provide second-order accuracy. In this particular case, the iterative methods have slightly smaller error than non-iterative methods. The errors from the three non-iterative methods are identical since TCSF, TCSF-D and local linearization are mathematically equivalent.

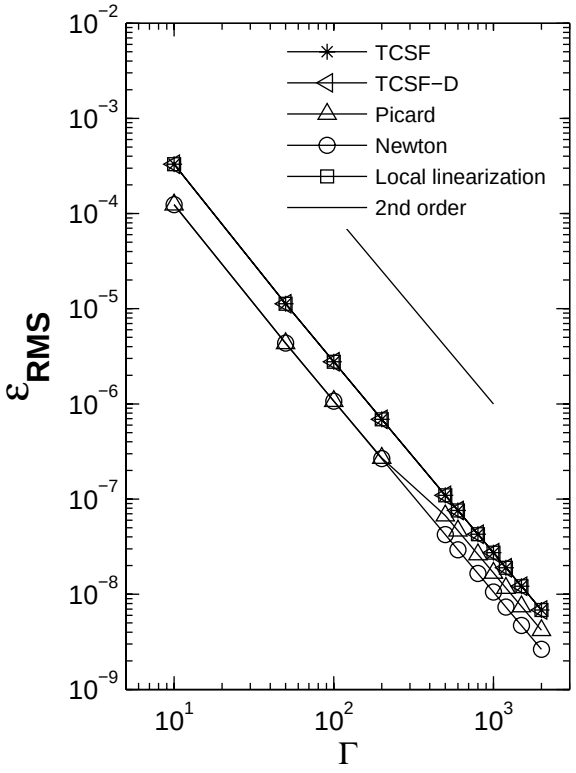


Figure 4.15 RMS error from Equation (4.117) for discrete solutions of the ODE, Equation (4.101), computed using a range of $\Gamma$ time steps.

## 4.4 SUMMARY

The TCS method has been tested on three different test cases: a 1D conservative Burgers' equation, a 1D non-conservative Burgers' equation and an ODE. The TCS method has multiple discrete forms for each test case. All the TCS discretizations have been compared to conventional implicit linearization methods, including Picard iteration, Newton iteration and local linearization methods. All the TCS discretizations

demonstrate 2nd-order temporal accuracy and are shown stable up to a CFL *O* (10). All the TCS discretizations are proved stable when the CFL value is in the order of 10. The stability advantage of the TCS methods has been further demonstrated by the comparison with the RK methods. The TCS methods require fewer operation counts than Picard and Newton method, but are similar to local linearization. The principal advantage of the TCS method over local linearization is the relative ease with which the TCS method can be derived and implemented as it does not require discrete evaluation of a function Jacobian. The TCS method is applicable to any quadratically-nonlinear problems, which is verified by the ODE case.

For each test case, different TCS discretizations perform similarly in temporal accuracy, stability and efficiency. However, difference can be observed among different discretizations. In the 1D conservative Burgers' equation, the TCSF and TCSG collapse into one single form. The TCSF-D reverses the solution process of the TCSF. Nevertheless, the TCSF-D has a noticeably smaller relative error than the TCSF and other methods. In the 1D non-conservative Burgers' equation, the TCSF and TCSG hold distinct forms. The TCSF-D reverses the solution process of the TCSG, and the TCSG-D reverses the solution process of the TCSF. Among the four discretizations, the TCSF has the smallest relative error, which is about one order of magnitude smaller than the TCSG-D. In the 1D nonlinear ODE case, the TCSF and TCSG remain the same. The TCSF-D reverses the solution process of the TCSF, but they have the same relative error.

# Chapter 5 The TCS Family Method

In Chapter 3, we revealed that the key to the TCS method is the 2nd-order accurate time-centered splitting. Applying this splitting to different terms, different TCS discretizations can be generated. In Chapter 4, we compared the TCS discretizations with conventional linearization methods using various examples. In this chapter, we derive the TCS family method in one general form by introducing weighting factors to different terms. A significance of this TCS family method is that the value of the weighting factors does not affect the order of accuracy of the scheme. Since weighting factors can be any value between zero and one while still providing essentially the same advantages, it may provide a flexible approach for a variety of problems. To examine the properties of weighting factors, the 1D non-conservative Burgers' equation is solved using different combinations of weighting factors. Furthermore, we demonstrate that the TCS method can computationally decouple an equation system of coupled variables using special combinations of weighting factors. This advantage makes the TCS method more efficient in solving coupled multi-variable equation systems.

## 5.1 Derivation of the TCS Family Method

Let us start a time marching differential equation from a simplest form:

$$\frac{\partial \phi}{\partial t} = f\left\{F\left[M(\phi)L(\phi)\right], N(\phi)\right\} \tag{5.1}$$

where $\phi$ s a generic variable, F, M, L and N represent generic linear operators. The function f is a linear function of $F\left[M(\phi)L(\phi)\right]$ and $N(\phi)$. Equation (5.1) then represents a time marching differential equation with a quadratic nonlinearity of $F\left[M(\phi)L(\phi)\right]$ and a linear operator of the variable, $N(\phi)$. This quadratic nonlinearity can be: the product of the variable itself, $\phi^2$ if F, M and L are equal to 1; the product of the variable and the gradient of the variable, $\phi\partial\phi/\partial x$, if F=1, M=1 and L=$\partial(\ )/\partial x$; the gradient of the product of the variable, $\partial(\phi\phi)/\partial x$ if M=1, L=1 and F=$\partial(\ )/\partial x$. Thus, Equation (5.1) represents a generic time marching differential equation with any kind of quadratic nonlinearity. Equation (5.1) can be discretized using the midpoint rule, written as

$$\phi^{n+1} = \phi^n + f\left\{F\left[M\left(\phi^{n+1/2}\right)L\left(\phi^{n+1/2}\right)\right], N\left(\phi^{n+1/2}\right)\right\}\Delta t + O\left(\Delta t^3\right) \tag{5.2}$$

In the previous chapters, we demonstrated that the time-centered split can be introduced to the quadratic nonlinear term and/or the linear term. Introducing two weighting factors,

$\theta_1$ and $\theta_2$, and combining with the time-centered linear approximation, $\phi^{n+1/2} = \frac{\phi^n + \phi^{n+1}}{2} + O\left(\Delta t^2\right)$, Equation (5.2) can be written as a general discretization

$$\phi^{n+1} = \phi^n + f\left\{\theta_1 F\left[M\left(\frac{\phi^n + \phi^{n+1}}{2}\right)L\left(\phi^{n+1/2}\right)\right] + \left(1-\theta_1\right)F\left[M\left(\phi^{n+1/2}\right)L\left(\frac{\phi^n + \phi^{n+1}}{2}\right)\right], \\ \theta_2 N\left(\phi^{n+1/2}\right) + \left(1-\theta_2\right)N\left(\frac{\phi^n + \phi^{n+1}}{2}\right)\right\}\Delta t + O\left(\Delta t^3\right) \tag{5.3}$$

where the weighting factor $\theta_i$ $(i \in \{1,2\})$ is required that $0 \le \theta_i \le 1$. Equation (5.3) is mathematically equivalent to Equation (5.2). Similar to the derivation in Chapter 3, an intermediate variable $\phi^*$ is introduced as

$$\phi^* = \phi^n + f\left\{\theta_1 F\left[M\left(\phi^n\right)L\left(\phi^{n+1/2}\right)\right] + \left(1-\theta_1\right)F\left[M\left(\phi^{n+1/2}\right)L\left(\phi^n\right)\right], \\ \theta_2 N\left(\phi^{n+1/2}\right) + \left(1-\theta_2\right)N\left(\phi^n\right)\right\}\frac{\Delta t}{2} \tag{5.4}$$

Subtracting Equation (5.4) from Equation (5.3), the second step is written as:

$$\phi^{n+1} = \phi^* + f\left\{\theta_1 F\left[M\left(\phi^{n+1}\right)L\left(\phi^{n+1/2}\right)\right] + \left(1-\theta_1\right)F\left[M\left(\phi^{n+1/2}\right)L\left(\phi^{n+1}\right)\right], \\ \theta_2 N\left(\phi^{n+1/2}\right) + \left(1-\theta_2\right)N\left(\phi^{n+1}\right)\right\}\frac{\Delta t}{2} + O\left(\Delta t^3\right) \tag{5.5}$$

Expanding $\phi^{n+1/2}$ in Equation (5.4) using the Taylor expansion,

$$\phi^* = \phi^n + f\left\{\theta_1 F\left[M\left(\phi^n\right)L\left(\phi^n + \phi_t\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2\right)\right] \\ + \left(1-\theta_1\right)F\left[M\left(\phi^n + \phi_t\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2\right)L\left(\phi^n\right)\right], \\ \theta_2 N\left(\phi^n + \phi_t\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2\right) + \left(1-\theta_2\right)N\left(\phi^n\right)\right\}\frac{\Delta t}{2} \tag{5.6}$$

Organizing Equation (5.6) and grouping the higher order terms,

$$\phi^* = \phi^n + f\left\{F\left[M\left(\phi^n\right)L\left(\phi^n\right)\right], N\left(\phi^n\right)\right\}\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2 \tag{5.7}$$

$\phi^{n+1/2}$ can be approximated using the explicit Euler approximation as

$$\phi^{n+1/2} = \phi^n + f\left\{F\left[M\left(\phi^n\right)L\left(\phi^n\right)\right], N\left(\phi^n\right)\right\}\frac{\Delta t}{2} + O\left(\frac{\Delta t}{2}\right)^2 \tag{5.8}$$

Therefore, we obtain the correspondence between $\phi^*$ and $\phi^{n+1/2}$ from Equations (5.7) and (5.8) such that

$$\phi^* = \phi^{n+1/2} + O\left(\Delta t^2\right) \tag{5.9}$$

Substituting Equation (5.9) into Equations (5.4) and (5.5), the original equation set becomes a computationally linearized two-step equation system:

$$\begin{aligned}\phi^* = \phi^n + f\Big\{&\theta_1 F\left[M\left(\phi^n\right)L\left(\phi^*\right)\right] + \left(1-\theta_1\right)F\left[M\left(\phi^*\right)L\left(\phi^n\right)\right],\\ &\theta_2 N\left(\phi^*\right) + \left(1-\theta_2\right)N\left(\phi^n\right)\Big\}\frac{\Delta t}{2}\end{aligned} \tag{5.10}$$

$$\begin{aligned}\phi^{n+1} = \phi^* + f\Big\{&\theta_1 F\left[M\left(\phi^{n+1}\right)L\left(\phi^*\right)\right] + \left(1-\theta_1\right)F\left[M\left(\phi^*\right)L\left(\phi^{n+1}\right)\right],\\ &\theta_2 N\left(\phi^*\right) + \left(1-\theta_2\right)N\left(\phi^{n+1}\right)\Big\}\frac{\Delta t}{2} + O\left(\Delta t^3\right)\end{aligned} \tag{5.11}$$

The weighting factor, $\theta_i$, distinguishes this derivation above from the original derivation of TCS method in Chapter 3 because the value of $\theta_i$ will not affect the 2nd-order accuracy of the TCS methods. In addition, there is no dependent relationship between $\theta_1$ and $\theta_2$. In other words, as long as $0 \le \theta_i \le 1$, Equations (5.10) and (5.11) are a 2nd-order equivalent to Equation (5.2). Although the derivation above uses a single variable, the same mathematic principles apply to any quadratic nonlinearity. In ordinary differential equations (ODE), the nonlinearity is simply $x^2$ or xy; in partial differential equations (PDE), this quadratic nonlinearity can be $U\partial U/\partial x$ or $\partial(HU)/\partial x$. Therefore, the derivation above is applicable to an ODE or ODE system with quadratic nonlinearity and typical flow transport equation such as 1D advection-diffusion equation or shallow water equations. In addition, the more variables are involved in an equation system, the more weighting factors that can be introduced. As a result, the TCS method provides not a single kind, but a whole family of TCS discretizations.

As we illustrated in the previous chapters, all the TCS family methods keep 2nd-order temporal accuracy and they share the same simplicity advantages in computation. However, with different combinations of $\theta_i$, the numerical solutions perform differently. In the next section, we will use the 1D non-conservative Burgers' equation as an example to explore the different performance of different TCS discretizations.

## 5.2 APPLICATION OF THE TCS FAMILY METHOD TO THE 1D NON-CONSERVATIVE BURGERS' EQUATION

With the introduction of the weighting factors $\theta_1$ and $\theta_2$, the general format of the TCS discretized non-conservative Burgers' equation is written as,

$$u^* = u^n + \frac{\Delta t}{2}\left\{\theta_1\left(-u^n\delta_x u^*\right) + \left(1-\theta_1\right)\left(-u^*\delta_x u^n\right) + \theta_2\left(\nu\delta_x^2 u^*\right) + \left(1-\theta_2\right)\nu\delta_x^2 u^n\right\} \quad (5.12)$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left\{\theta_1\left(-u^{n+1}\delta_x u^*\right) + \left(1-\theta_1\right)\left(-u^*\delta_x u^{n+1}\right) + \theta_2\nu\delta_x^2 u^* + \left(1-\theta_2\right)\nu\delta_x^2 u^{n+1}\right\} \quad (5.13)$$

The above equations are a two-step matrix equation system in the format of **Ax=b**. Using a central difference in spatial derivatives, the LHS coefficient matrix **A** in the first step is tridiagonal such that

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & & & & & & 0 \\ a_{2,1} & a_{2,2} & a_{2,3} & & & & & \\ & a_{3,2} & a_{3,3} & a_{3,4} & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & a_{Q-2,N-3} & a_{Q-2,Q-2} & a_{Q-2,Q-1} & & \\ & & & & a_{Q-1,N-2} & a_{Q-1,Q-1} & a_{Q-1,Q} & \\ 0 & & & & & a_{Q,Q-1} & a_{Q,Q} & \end{bmatrix} \quad (5.14)$$

where Q is the number of the grid points in space, and the non-zero components of **A** for $i = \{1, 2...Q\}$ are of the form

$$a_{i,j} = \begin{cases} -\frac{\theta_1}{4}C_i^n - \frac{\theta_2}{2}\gamma & : \quad j = i-1 \\ 1 + \frac{1}{4}\theta_1\left[C_{i+1}^n - C_{i-1}^n\right] + \theta_2\gamma & : \quad j = i \\ +\frac{\theta_1}{4}C_i^n - \frac{\theta_2}{2}\gamma & : \quad j = i+1 \end{cases} \quad (5.15)$$

Also in the first step,

$$\mathbf{x} = \left[u_1^*, u_2^*, ...u_Q^*\right]^T \quad (5.16)$$

and

$$\mathbf{b}=\begin{bmatrix} u_1^n+\dfrac{(1-\theta_2)\gamma D_1^n}{2}+\dfrac{\theta_1 C_1^n}{4}+\dfrac{\theta_2\gamma}{2} \\ u_2^n+\dfrac{(1-\theta_2)\gamma D_2^n}{2} \\ \vdots \\ u_{Q-1}^n+\dfrac{(1-\theta_2)\gamma D_{Q-1}^n}{2} \\ u_Q^n+\dfrac{(1-\theta_2)\gamma D_Q^n}{2}-\dfrac{\theta_1 C_Q^n}{4}+\dfrac{\theta_2\gamma}{2} \end{bmatrix} \tag{5.17}$$

In the second step, we also have a tridiagonal coefficient matrix **A** with different tridiagonal elements as

$$a_{i,j}=\begin{cases} -\dfrac{1-\theta_1}{4}C_i^*-\dfrac{1-\theta_2}{2}\gamma & : \quad j=i-1 \\ 1+\dfrac{1}{4}(1-\theta_1)\left[C_{i+1}^*-C_{i-1}^*\right]+(1-\theta_2)\gamma & : \quad j=i \\ +\dfrac{1-\theta_1}{4}C_i^*-\dfrac{1-\theta_2}{2}\gamma & : \quad j=i+1 \end{cases} \tag{5.18}$$

and

$$\mathbf{x}=\left[u_1^{n+1},u_2^{n+1},...u_Q^{n+1}\right]^T \tag{5.19}$$

$$\mathbf{b}=\begin{bmatrix} u_1^*+\dfrac{\theta_2\gamma D_1^*}{2}+\dfrac{(1-\theta_1)C_1^*}{4}+\dfrac{(1-\theta_2)\gamma}{2} \\ u_2^*+\dfrac{\theta_2\gamma D_2^*}{2} \\ \vdots \\ u_{Q-1}^*+\dfrac{\theta_2\gamma D_{Q-1}^*}{2} \\ u_Q^*+\dfrac{\theta_2\gamma D_Q^*}{2}-\dfrac{(1-\theta_1)C_Q^*}{4}+\dfrac{(1-\theta_2)\gamma}{2} \end{bmatrix} \tag{5.20}$$

where $C$, $\gamma$ and $D$ in these two steps have the same definitions as in Chapter 4.

## 5.3 RESULTS AND DISCUSSION

To explore the properties of different weighting factors, the 1D non-conservative Burgers' equation is solved using Equations (5.12) and (5.13) with different combinations of $\theta_1$ and $\theta_2$. Before we analyze the results from various combinations of $\theta_1$ and $\theta_2$, it is necessary to review their meanings. In the TCS scheme, $u^{n+1/2}$ in the Burgers' equation is either approximated using the time-centered splitting as $u^{n+1/2} = \left(u^n + u^{n+1}\right)/2 + O\left(\Delta t^2\right)$ or $u^*$ as $u^{n+1/2} = u^* + O\left(\Delta t^2\right)$. Different $\theta_i$ indicates different approximations of $u^{n+1/2}$. When $\theta_i$=0 or $\theta_i$=1, $u^{n+1/2}$ is approximated only using one kind of the approximations. When $\theta_i$ lies between 0 and 1, $u^{n+1/2}$ is approximated by the combinations of these two approximations. Table 5.1 lists the mathematical meaning of different values of $\theta_i$.

Table 5.1 The mathematical meaning of the value of $\theta_1$ and $\theta_2$.

| value | $\theta_1$ | $\theta_2$ |
|---|---|---|
| 0 | split the entire gradient term and approximate the entire flux term using $u^*$ | split the entire diffusion term |
| $0 <$ and $< 1$ | split part of the gradient and flux terms approximate part of the gradient and flux terms using $u^*$ | split part of the diffusion term and approximate part of the diffusion term using $u^*$ |
| 1 | split the entire flux term and approximate the entire gradient term using $u^*$ | approximate the entire diffusion term using $u^*$ |

### 5.3.1 Accuracy of Different Weighting Factors

The error analysis of different TCS discretizations is evaluated in the same vein as in Chapter 4. In figure 5.1, the RMS error is plotted against a range of CFL numbers for six different $\theta_1$ values ($\theta_1 \in \{0,\ 1/3,\ 1/2,\ 2/3,\ 3/4,\ 1\}$). For each $\theta_1$ value, four different $\theta_2$ values ($\theta_2 \in \{0,\ 1/3,\ 1/2,\ 1\}$) are chosen for comparison, resulting twenty four different TCS discretizations.

The total marching time t=0.6 is used as the basis for error comparison in Figure 5.1 because the maximum absolute error occurs at t=0.6 (as demonstrated in Chapter 4). The CFL number is defined using $u_{analytical}$, $\Delta x = 1/50$ and $\Delta t = 0.6/\Gamma$, where $\Gamma$ is the

number of the time steps and $\Gamma \in \{5, 8, 10, 20, 25, 30, 50\}$. Thus, the tested CFL number covers the range of $0.4 \le \mathrm{CFL} \le 4.2$. It can be observed that every TCS discretization has the same 2nd-order temporal accuracy and remains stable in the whole range of the CFL numbers. The computation requirement of each TCS discretization is essentially the same since they share the same general structure of matrix equations as in Equations (5.14) to (5.20). Figure 5.1 clearly shows that for a given value of $\theta_1$, any choice of $\theta_2$ gives an error of the same magnitude. The reverse also holds true, as shown in Figure 5.2. The RMS error is plotted for six values of $\theta_2$, choosing four $\theta_1$ values for each $\theta_2$ value. Again, given a value for $\theta_2$, the choice of $\theta_1$ affects the RMS error less than or equal to one order of magnitude.

Though $\theta_1 = 2/3,\ \theta_2 = 1/2$ gives the lowest RMS error in these two examples, the best combination for the smallest error is problem-specific; error arises from approximations of different terms. To obtain optimal combinations of $\theta_i$ for the lowest error, one needs to examine the nature of the solution and each term at the designated marching time. For this Burgers' equation, the solution u, the gradient term $\partial u / \partial x$, and the diffusion term $\nu \partial^2 u / \partial x^2$ all need to be taken into consideration because approximations have been introduced to all of these terms.

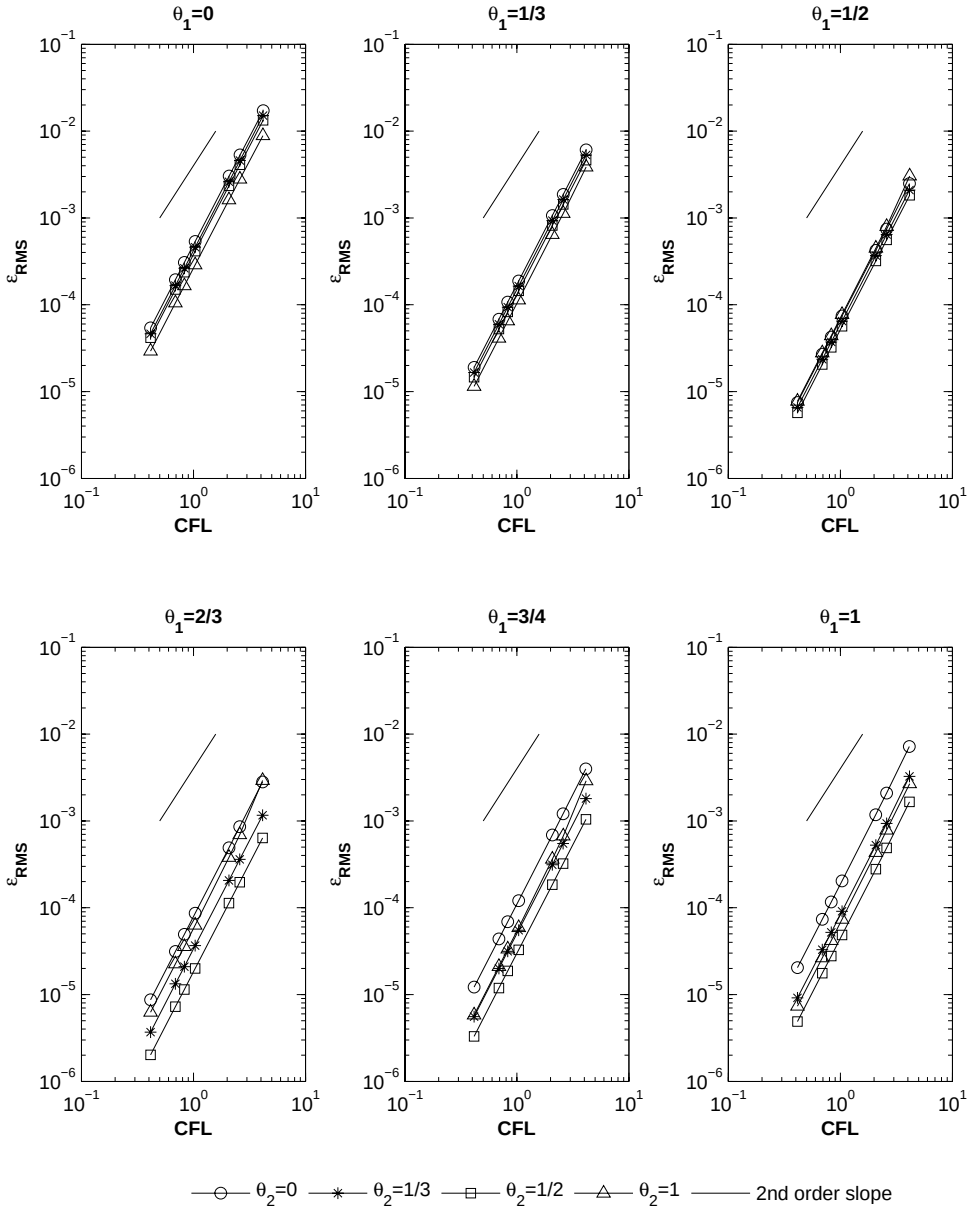


Figure 5.1 Temporal accuracy of various combinations of $\theta_1$ and $\theta_2$ for solution of the Burgers' equation at t=0.6, where $\Delta x = 1/50$, $\nu = 0.05$ and $\Delta t = 0.6/\Gamma$ ($\Gamma \in \{5, 8, 10, 20, 25, 30, 50\}$).

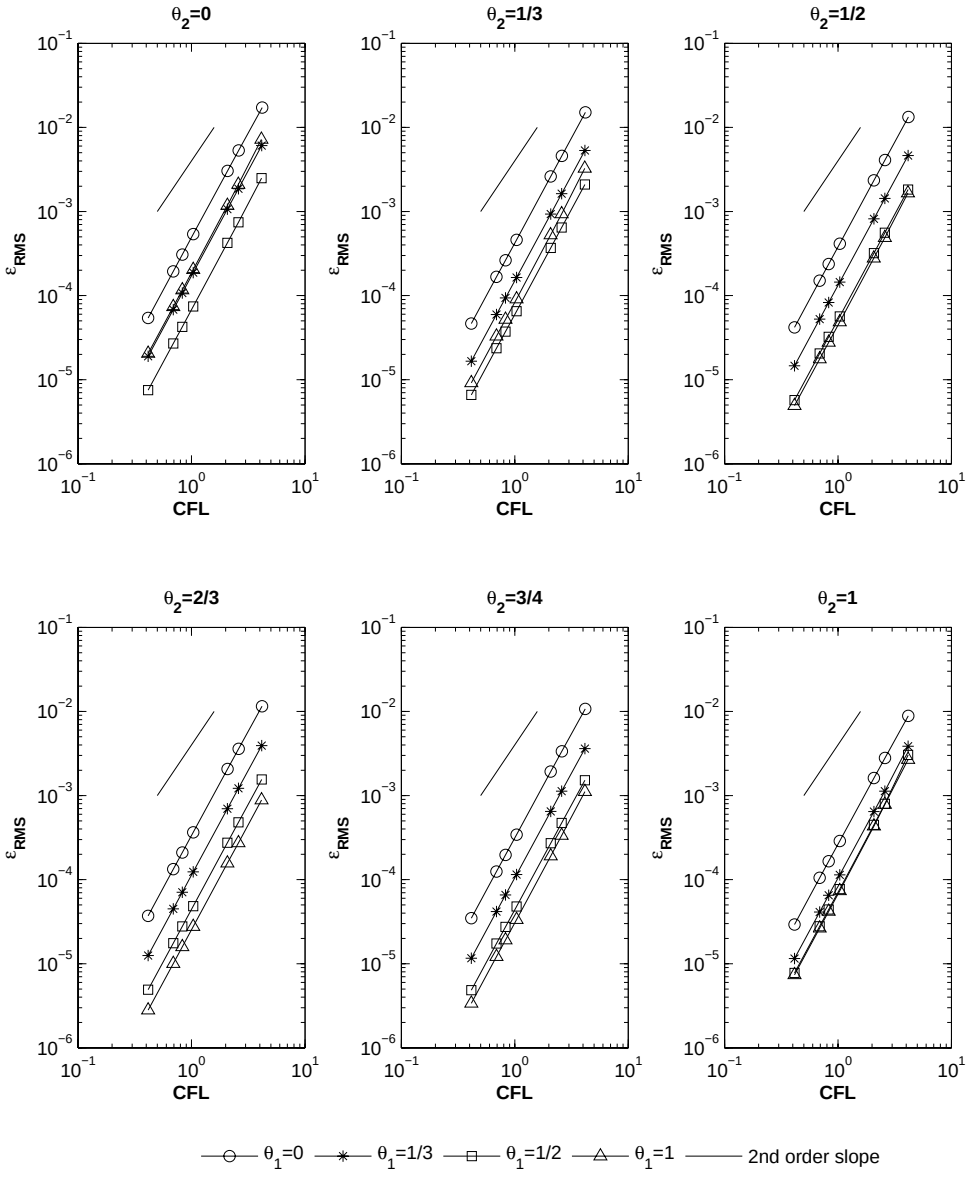


Figure 5.2 Temporal accuracy of various combinations of $\theta_2$ and $\theta_1$ for solution of the Burgers' equation at t=0.6, where $\Delta x = 1/50$, $\nu = 0.05$ and $\Delta t = 0.6/\Gamma$ ($\Gamma \in \{5, 8, 10, 20, 25, 30, 50\}$).

### 5.3.2 Stability of Different Weighting Factors

Figure 5.1 and 5.2 display the temporal accuracy of the solution of the Burgers' equation at t=0.6. All TCS discretizations produce stable solutions for the whole range of tested CFL numbers. To further explore the stability characteristics, we investigate the performance of the TCS family method at t=0.3, 1 and 2, and these results are shown in Figure 5.3, 5.4 and 5.5, respectively. Similar to Figure 5.1, the RMS error is plotted against a range of CFL numbers for several choices of $\theta_2$ given a $\theta_1$ value. Similar to t=0.6, all 24 TCS discretizations stay stable at t=0.3.

At t=1and t=2, though, we find different results. At t=1 (Figure 5.4), for any given $\theta_1$ value, the solution of $\theta_2 = 1$ becomes unstable with the increase of the CFL number, but the solutions for any $\theta_2 \neq 1$ remain stable. At t=2, the solutions of both $\theta_2 = 0$ and $\theta_2 = 1$ become unstable with the increase of the CFL number, but the solutions with any $0 < \theta_2 < 1$ remain stable.

To understand this stability property associated with the $\theta_2$ value, we need to examine the nature of the solution and the meaning of the weighting factor $\theta_2$. The solution of the Burgers' equation evolving with time is plotted in Figure 4.1. It can be observed that after t=0.6, damping dominates the solution, which suggests that the diffusion effect dominates. The weighting factor $\theta_2$ controls the approximation of the diffusion term as shown in Table 5.1. If $\theta_2 = 0$, the entire diffusion term, $\nu\partial^2 u^{n+1/2}/\partial x^2$, is completely approximated using the time-centered split, which means $u^{n+1/2}$ in the diffusion term is first approximated as $u^{n+1/2} \approx \left(u^n + u^{n+1}\right)/2$ and then we split the diffusion terms into two steps as $\nu\partial^2 u^n/\partial x^2$ and $\nu\partial^2 u^{n+1}/\partial x^2$. If $\theta_2 = 1$, the entire diffusion term, $\nu\partial^2 u^{n+1/2}/\partial x^2$, is completely approximated using $u^*$, which means $\nu\partial^2 u^{n+1/2}/\partial x^2 \approx \nu\partial^2 u^*/\partial x^2$. Figures 5.1 to 5.5 indicate that the TCS method remains stable and the stability is not affected by the values of the weighing factors. If the diffusion dominates the solution, the weighting factor associated with the diffusion term can affect the stability. Our results suggest that if the weighting factor, $\theta_2 \neq 0$ and $\theta_2 \neq 1$, the solutions remain stable for the tested CFL numbers. In other words, using only one kind of approximation for the diffusion term, the solution may become unstable with the increase of the CFL number; using a combination of the two kinds of approximations for the diffusion term can stabilize the solution.

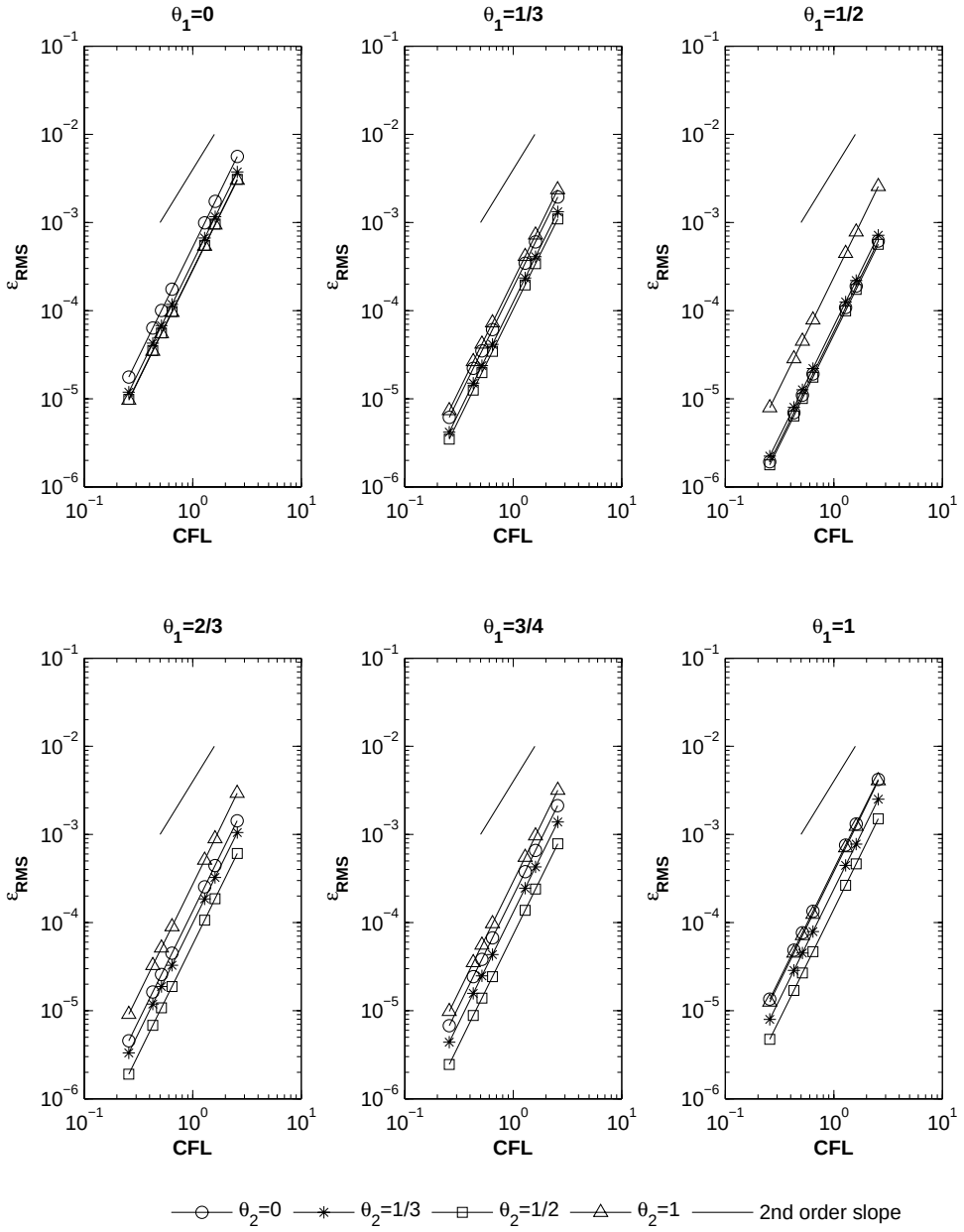


Figure 5.3 Temporal accuracy of various combinations of $\theta_2$ and $\theta_1$ for solution of the Burgers' equation at t=0.3, where $\Delta x = 1/50$, $\nu = 0.05$ and $\Delta t = 0.3/\Gamma$ ($\Gamma \in \{5, 8, 10, 20, 25, 30, 50\}$).

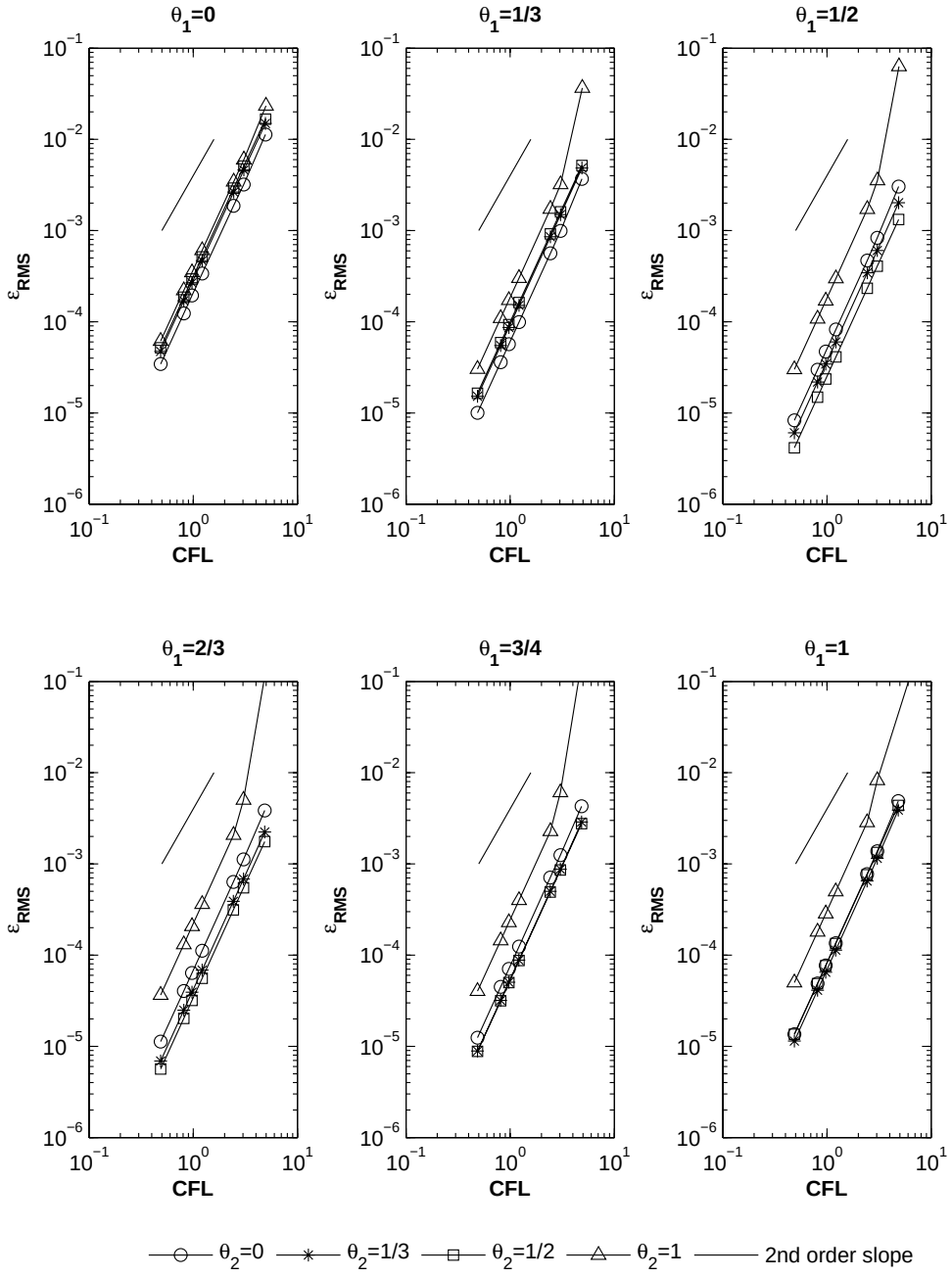


Figure 5.4 Temporal accuracy of various combinations of $\theta_2$ and $\theta_1$ for solution of the Burgers' equation at t=1, where $\Delta x = 1/50$, $\nu = 0.05$ and $\Delta t = 1/\Gamma$ ($\Gamma \in \{5, 8, 10, 20, 25, 30, 50\}$).

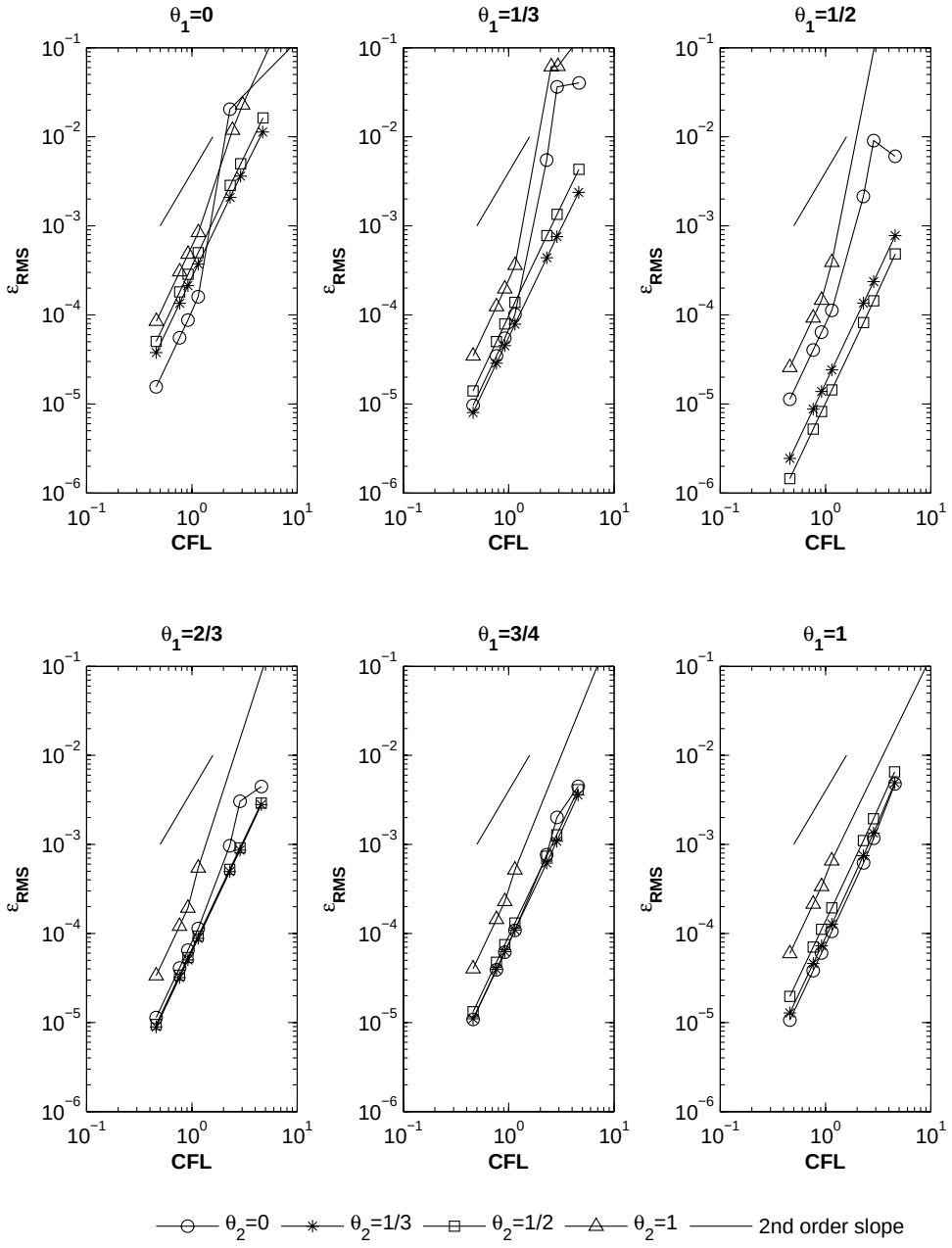


Figure 5.5 Temporal accuracy of various combinations of $\theta_2$ and $\theta_1$ for solution of the Burgers' equation at t=2, where $\Delta x = 1/50$, $\nu = 0.05$ and $\Delta t = 2/\Gamma$ ($\Gamma \in \{5, 8, 10, 20, 25, 30, 50\}$).

The general form of TCS method is demonstrated using the 1D non-conservative Burgers' equation. The same application of the weighting factors can be used for an equation system with coupled variables. Hence, a general form of a two-step linearized coupled equation system can be created using the same principle. Since the values of the weighting factors will not affect the 2nd-order temporal accuracy of the scheme, TCS method may provide a flexible approach to solve different problems. Furthermore, properly chosen weighting factors in the TCS method can computationally decouple a system of coupled equations. This decoupling advantage of the TCS method is analyzed in the following section.

## 5.4 COMPUTATIONAL DECOUPLING

In the previous sections, we demonstrated the effects of weighting factors in the TCS method on the accuracy and stability of the solution. In this section, we will introduce the decoupling advantage of the TCS method. The two-dimensional Burgers' equation provides a convenient example:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \nu\frac{\partial^2 u}{\partial x^2} + \nu\frac{\partial^2 u}{\partial y^2} \tag{5.21}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \nu\frac{\partial^2 v}{\partial x^2} + \nu\frac{\partial^2 v}{\partial y^2} \tag{5.22}$$

Equations (5.21) and (5.22) are a two-variable two-dimensional system with coupled variables 'u' and 'v'. If we use the 2nd-order temporal accurate C-N method to discretize equations (5.21) and (5.22), the resulting equations become

$$\begin{aligned} u^{n+1} = u^n &+ \frac{\Delta t}{2}\left\{-u^n\frac{\partial u^n}{\partial x} - v^n\frac{\partial u^n}{\partial y} + \nu\frac{\partial^2 u^n}{\partial x^2} + \nu\frac{\partial^2 u^n}{\partial y^2}\right\} \\ &+ \frac{\Delta t}{2}\left\{-u^{n+1}\frac{\partial u^{n+1}}{\partial x} - v^{n+1}\frac{\partial v^{n+1}}{\partial y} + \nu\frac{\partial^2 u^{n+1}}{\partial x^2} + \nu\frac{\partial^2 u^{n+1}}{\partial y^2}\right\} \end{aligned} \tag{5.23}$$

$$\begin{aligned} v^{n+1} = v^n &+ \frac{\Delta t}{2}\left\{-u^n\frac{\partial v^n}{\partial x} - v^n\frac{\partial v^n}{\partial y} + \nu\frac{\partial^2 v^n}{\partial x^2} + \nu\frac{\partial^2 v^n}{\partial y^2}\right\} \\ &+ \frac{\Delta t}{2}\left\{-u^{n+1}\frac{\partial v^{n+1}}{\partial x} - v^{n+1}\frac{\partial v^{n+1}}{\partial y} + \nu\frac{\partial^2 v^{n+1}}{\partial x^2} + \nu\frac{\partial^2 v^{n+1}}{\partial y^2}\right\} \end{aligned} \tag{5.24}$$

Equations (5.23) and (5.24) are a coupled nonlinear equation system. Conventionally, to solve a coupled equation system, either a simultaneous solution or a sequential solution method can be used (Ferziger and Peric, 1996). However, such an equation system is too complex and expensive to solve simultaneously due to its nonlinearity. Therefore, sequential solution methods are more appropriate. To solve this equation system

sequentially, inner iterations are required to solve Equations (5.23) and (5.24) individually, and outer iterations are required to satisfy Equations (5.23) and (5.24) together.

Alternatively, we can decouple and linearize Equations (5.23) and (5.24) using TCS method without iterations by choosing specific weighting factors. Applying the TCS method to Equation (5.21) results in a two-step system with a specific $\theta_i$ weighting factor for each term

$$\begin{aligned} u^* = u^n + \frac{\Delta t}{2}\Big\{&\theta_1\left(-u^n\delta_x u^*\right) + \left(1-\theta_1\right)\left(-u^*\delta_x u^n\right) \\ &+ \theta_2\left(-v^n\delta_y u^*\right) + \left(1-\theta_2\right)\left(-v^*\delta_y u^n\right) \\ &+ \theta_3\left(\nu\delta_x^2 u^*\right) + \left(1-\theta_3\right)\nu\delta_x^2 u^n \\ &+ \theta_4\left(\nu\delta_y^2 u^*\right) + \left(1-\theta_4\right)\nu\delta_y^2 u^n\Big\} \end{aligned} \tag{5.25}$$

$$\begin{aligned} u^{n+1} = u^* + \frac{\Delta t}{2}\Big\{&\theta_1\left(-u^{n+1}\delta_x u^*\right) + \left(1-\theta_1\right)\left(-u^*\delta_x u^{n+1}\right) \\ &+ \theta_2\left(-v^{n+1}\delta_y u^*\right) + \left(1-\theta_2\right)\left(-v^*\delta_y u^{n+1}\right) \\ &+ \theta_3\nu\delta_x^2 u^* + \left(1-\theta_3\right)\nu\delta_x^2 u^{n+1} \\ &+ \theta_4\nu\delta_y^2 u^* + \left(1-\theta_4\right)\nu\delta_y^2 u^{n+1}\Big\} \end{aligned} \tag{5.26}$$

Applying the TCS method to Equation (5.22) results in a two-step system:

$$\begin{aligned} v^* = v^n + \frac{\Delta t}{2}\Big\{&\theta_5\left(-u^n\delta_x v^*\right) + \left(1-\theta_5\right)\left(-u^*\delta_x v^n\right) \\ &+ \theta_6\left(-v^n\delta_y v^*\right) + \left(1-\theta_6\right)\left(-v^*\delta_y v^n\right) \\ &+ \theta_7\left(\nu\delta_x^2 v^*\right) + \left(1-\theta_7\right)\nu\delta_x^2 v^n \\ &+ \theta_8\left(\nu\delta_y^2 v^*\right) + \left(1-\theta_8\right)\nu\delta_y^2 v^n\Big\} \end{aligned} \tag{5.27}$$

$$\begin{aligned} v^{n+1} = v^* + \frac{\Delta t}{2}\Big\{&\theta_5\left(-u^{n+1}\delta_x v^*\right) + \left(1-\theta_5\right)\left(-u^*\delta_x v^{n+1}\right) \\ &+ \theta_6\left(-v^{n+1}\delta_y v^*\right) + \left(1-\theta_6\right)\left(-v^*\delta_y v^{n+1}\right) \\ &+ \theta_7\nu\delta_x^2 v^* + \left(1-\theta_7\right)\nu\delta_x^2 v^{n+1} \\ &+ \theta_8\nu\delta_y^2 v^* + \left(1-\theta_8\right)\nu\delta_y^2 v^{n+1}\Big\} \end{aligned} \tag{5.28}$$

For this two-variable two-dimensional Burgers' equation, eight different weighting factors are applied in the general TCS discretized form. Observing Equations (5.21) and (5.22), we discover that the coupling only occurs in the nonlinear advection term.

Therefore, the weighting factors for the diffusion terms will not affect the computational decoupling.

In the following discussion, we omit the diffusion terms and their associated weighting factors. We then discover that the coupling occurs between the nonlinear advection terms ' $v\partial u/\partial y$ ' and ' $u\partial v/\partial x$ '. The two weighting factors for these two terms are $\theta_2$ and $\theta_5$ respectively and these two factors are the key to computationally decoupling the equation system. The other two advection terms ' $u\partial u/\partial x$ ' and ' $v\partial v/\partial y$ ' are not coupled. Therefore, we have more choices for their weighting factors $\theta_1$ and $\theta_6$. To completely decouple u and v, Table 5.2 shows the possible combinations of the weighting factors for each advection term.

Table 5.2 Weighting factors for computational decoupling the 2D Burgers' equation

| $\theta_2$ | $\theta_5$ | $\theta_1$ | $\theta_6$ |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| | | 1 | 0 |
| | | 0 | 1 |
| | | 1 | 1 |
| 1 | 0 | 0 | 0 |
| | | 1 | 0 |
| | | 0 | 1 |
| | | 1 | 1 |

It can be seen from Table 5.2 that two kinds of combinations of $\theta_2$ and $\theta_5$ can decouple the 2D Burgers' equation. For each pair of $\theta_2$ and $\theta_5$, four different combinations of $\theta_1$ and $\theta_6$ can be chosen such as $\theta_1 = \theta_6 = 0$, $\theta_1 = 1$ and $\theta_6 = 0$, $\theta_1 = \theta_6 = 1$, $\theta_1 = 0$ and $\theta_6 = 1$. To illustrate the procedure of decoupling, we use the values from the first row in Table 5.2, applying $\theta_2 = 0,\ \theta_5 = 1,\ \theta_1 = 0,\ \theta_6 = 0$ to Equations (5.25) to (5.28) and reordering the equation system to

$$v^* = v^n + \frac{\Delta t}{2}\left(-u^n\delta_x v^* - v^*\delta_y v^n\right) \tag{5.29}$$

$$u^* = u^n + \frac{\Delta t}{2}\left(-u^*\delta_x u^n - v^*\delta_y u^n\right) \tag{5.30}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left(-u^*\delta_x u^{n+1} - v^*\delta_y u^{n+1}\right) \tag{5.31}$$

$$v^{n+1} = v^* + \frac{\Delta t}{2}\left(-u^{n+1}\delta_x v^* - v^*\delta_y v^{n+1}\right) \tag{5.32}$$

Each step from Equation (5.29) to (5.32) is computationally linearized, and each step has only one unknown variable. Variables "u" and "v" are effectively decoupled in each step. The solution process for these chosen weighting factors is as follows:

1. Solve for $v^*$ using $u^n$ and $v^n$
2. Solve for $u^*$ using $u^n$ and $v^*$ obtained from step 1
3. Solve for $u^{n+1}$ using $u^*$ obtained from step 2 and $v^*$ obtained from step 1
4. Solve for $v^{n+1}$ using $u^{n+1}$ obtained from step 3 and $v^*$ obtained from step 1

A different combination of weighting factors from Table 5.2 results in a different equation system, and the order of the solution procedure is changed appropriately. Using the following set of weighting factors in Equations (5.25) to (5.28): $\theta_2 = 1$, $\theta_5 = 0$, $\theta_1 = 0$, and $\theta_6 = 0$, results in the following system:

$$u^* = u^n + \frac{\Delta t}{2}\left(-u^*\delta_x u^n - v^n\delta_y u^*\right) \tag{5.33}$$

$$v^* = v^n + \frac{\Delta t}{2}\left(-u^*\delta_x v^n - v^*\delta_y v^n\right) \tag{5.34}$$

$$v^{n+1} = v^* + \frac{\Delta t}{2}\left(-u^*\delta_x v^{n+1} - v^*\delta_y v^{n+1}\right) \tag{5.35}$$

$$u^{n+1} = u^* + \frac{\Delta t}{2}\left(-u^*\delta_x u^{n+1} - v^{n+1}\delta_y u^*\right) \tag{5.36}$$

The corresponding solution process is:

1. Solve for $u^*$ using $u^n$ and $v^n$
2. Solve for $v^*$ using $v^n$ and $u^*$ obtained from step 1
3. Solve for $v^{n+1}$ using $v^*$ obtained from step 2 and $u^*$ obtained from step 1
4. Solve for $u^{n+1}$ using $v^{n+1}$ obtained from step 3 and $u^*$ obtained from step 1

This computational decoupling technique can be extended to equation systems with more than two variables. The underlying principle for decoupling is that when the weighting factor is switched from 0 to 1, the approximation for the variable $\phi^{n+1/2}$ is changed from $\phi^*$ to either $\phi^n$ or $\phi^{n+1}$. Consequently, we can change the variable in one equation from unknown to known.

## 5.5 SUMMARY

A general form of the TCS family method is developed in this chapter by introducing a weighting factor $\theta_i$ for each term in a quadratic nonlinear differential equation. The 1D non-conservative Burgers' equation is used as an example to test how the weighting factors affect the solutions. The value of $\theta_i$ controls the approximations of

the advection and diffusion terms. If $\theta_i$=0 or $\theta_i$=1, the advection term or the diffusion term is approximated only by one kind of approximation; if $\theta_i$ lies between 0 and 1, the advection term or the diffusion term is approximated by the combination of two approximations. Although various TCS discretizations can be generated by changing $\theta_i$ the 2nd order temporal accuracy of the TCS method is not affected by the choice of the weighting factor. We proved analytically and by examples that the TCS family method is 2nd-order temporal accurate if $0 \le \theta_i \le 1$. Different combinations of $\theta_i$ value have been tested using 1D Burgers' equation as an example. The results presented here show that the optimum combination of the weighting factors for the most accurate solutions is problem-specific. However, our results indicate that the optimal values for the weighting factors lie between 0 and 1 such that the advection term and diffusion term are approximated using the combination of two different approximations. Our examples also demonstrate the stability advantage of the TCS method. When the advection effect is dominant, the stability is not affected by $\theta_i$ value and all the tested TCS discretizations remain stable for the whole range of CFL numbers. When the diffusion effect is dominant, the TCS method stays stable under all $\theta_i$ values except 0 or 1 for the diffusion term. This observation suggests that the solution from the TCS method may be stabilized by adjusting the weighting factors. Most of the problems in Environmental Fluid Mechanics are dominated by advection. Hence, the TCS family method has a promising stability advantage in simulating environmental flows.

In addition to its accuracy and stability advantages, the TCS method is also capable of decoupling coupled equation systems by choosing specific combinations of the weighting factors. This advantage is displayed using the two-dimensional Burgers' equation as an example. Compared to the conventional decoupling techniques, no iteration is required in the TCS method. Therefore, the TCS method is more efficient in solving coupled nonlinear equation systems. Moreover, the TCS family method provides theoretically unlimited possible discretizations for one problem. This, in turn, provides flexibility in discretizing specific problems.

In the next chapter, we will apply the TCS method to a multi-variable and multi-dimensional equation system: depth averaged shallow water equations (SWE). The advantage of computational linearization and decoupling of the TCS method will be further illustrated in the next chapter. Numerical experiments will be carried out to test the TCS discretized depth averaged SWE.

# Chapter 6 Application of the TCS Method to a 2D Depth Averaged Shallow Water Equations (SWE)

The TCS (Time-Centered Split) family of methods are derived and analyzed in Chapter 5. To investigate the application of the TCS method in a multi-variable and multi-dimension equation system, depth averaged SWE (shallow water equations) is solved using the TCS method in this chapter. The computational decoupling characteristics of the TCS method are explored using depth averaged SWE as an example. The decoupling procedure is discussed and three representative linearized and decoupled TCS discretizations are presented in this Chapter. Numerical experiments such as a one-dimensional standing wave and a two-dimensional standing wave in a rectangular domain are performed to verify the TCS numerical model. In the one-dimensional standing wave case, the numerical results are compared with the analytical solutions; and in the two-dimensional standing wave case, the characteristics and performance from three TCS discretizations are compared and discussed.

## 6.1. THE 2D DEPTH AVERAGED SWE

The 2D depth averaged SWE are obtained by integrating the 3D incompressible Navier-Stokes equations over the water depth with the two following assumptions: 1) neglecting the vertical velocity and acceleration; 2) applying a hydrostatic pressure distribution (Vreugdenhil 1994). These equations are widely used in hydraulic science and engineering (Ancey et al. 2008; Arega et al. 2008; Hunter et al. 2008). The definition of "shallow" requires that the vertical scale of the flow is small compared to the horizontal scale. In nature or common engineering practice, many types of flows fall into this category such as atmospheric flow (Mohebalhojeh and Dritschel 2007), river flow (Arega et al. 2008), and storm surge (Bajo et al. 2007). Although the 2D SWE cannot simulate vertical velocity gradients, they are useful for flows where strong turbulence provides complete vertical mixing of momentum or for flows dominated by barotropic motions. The 2D shallow water equations can be written as:

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} = -g\frac{\partial \zeta}{\partial x} + \nu\left(\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2}\right) \tag{6.1}$$

$$\frac{\partial V}{\partial t} + U\frac{\partial V}{\partial x} + V\frac{\partial V}{\partial y} = -g\frac{\partial \zeta}{\partial y} + \nu\left(\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2}\right) \tag{6.2}$$

$$\frac{\partial H}{\partial t} + \frac{\partial HU}{\partial x} + \frac{\partial HV}{\partial y} = 0 \tag{6.3}$$

$$\zeta = H + Z_b \tag{6.4}$$

where $U$ and $V$ are depth-averaged velocities in $x$ and $y$ directions; $g$ is the gravitational acceleration; $H$ is the water depth; $\zeta$ is the water surface elevation; and

$Z_b$ is the bottom elevation. The above statement of the 2D SWE neglects the turbulence closure and bottom stress for the test case of the TCS numerical method.

### 6.2 THE TCS DISCRETIZED SWE

Applying the midpoint rule between time n and n+1 along with central-differencing of spatial derivatives on an Arakawa C grid (Arakawa and Lamb, 1977), as shown in Figure 6.1, Equations (6.1), (6.2) and (6.3) can be discretized as a set of coupled nonlinear equations:

$$\begin{aligned} U_{i,j+1/2}^{n+1} = U_{i,j+1/2}^{n} + \Delta t \Bigg\{ & -U_{i,j+1/2}^{n+1/2} \frac{\left(U_{i,j+3/2}^{n+1/2} - U_{i,j-1/2}^{n+1/2}\right)}{2\Delta x} - V_{i,j+1/2}^{n+1/2} \frac{\left(U_{i+1,j+1/2}^{n+1/2} - U_{i-1,j+1/2}^{n+1/2}\right)}{2\Delta y} \\ & + \frac{\nu}{\Delta x^2}\left(U_{i,j-1/2}^{n+1/2} - 2U_{i,j+1/2}^{n+1/2} + U_{i,j+3/2}^{n+1/2}\right) \\ & + \frac{\nu}{\Delta y^2}\left(U_{i-1,j+1/2}^{n+1/2} - 2U_{i,j+1/2}^{n+1/2} + U_{i+1,j+1/2}^{n+1/2}\right) \\ & - \frac{g}{\Delta x}\left(\zeta_{i,j+1}^{n+1/2} - \zeta_{i,j}^{n+1/2}\right) \Bigg\} \end{aligned} \tag{6.5}$$

$$\begin{aligned} V_{i+1/2,j}^{n+1} = V_{i+1/2,j}^{n} + \Delta t \Bigg\{ & -U_{i+1/2,j}^{n+1/2} \frac{\left(V_{i+1/2,j+1}^{n+1/2} - V_{i+1/2,j-1}^{n+1/2}\right)}{2\Delta x} - V_{i+1/2,j}^{n+1/2} \frac{\left(V_{i+3/2,j}^{n+1/2} - V_{i-1/2,j}^{n+1/2}\right)}{2\Delta y} \\ & + \frac{\nu}{\Delta x^2}\left(V_{i+1/2,j-1}^{n+1/2} - 2V_{i+1/2,j}^{n+1/2} + V_{i+1/2,j+1}^{n+1/2}\right) \\ & + \frac{\nu}{\Delta y^2}\left(V_{i-1/2,j}^{n+1/2} - 2V_{i+1/2,j}^{n+1/2} + V_{i+1/2,j}^{n+1/2}\right) \\ & - \frac{g}{\Delta y}\left(\zeta_{i+1,j}^{n+1/2} - \zeta_{i,j}^{n+1/2}\right) \Bigg\} \end{aligned} \tag{6.6}$$

$$\begin{aligned} H_{i,j}^{n+1} = H_{i,j}^{n} + \Delta t \Bigg\{ & -\frac{1}{\Delta x}\left(H_{i,j+1/2}^{n+1/2} U_{i,j+1/2}^{n+1/2} - H_{i,j-1/2}^{n+1/2} U_{i,j-1/2}^{n+1/2}\right) \\ & - \frac{1}{\Delta y}\left(H_{i+1/2,j}^{n+1/2} V_{i+1/2,j}^{n+1/2} - H_{i-1/2,j}^{n+1/2} V_{i-1/2,j}^{n+1/2}\right) \Bigg\} \end{aligned} \tag{6.7}$$

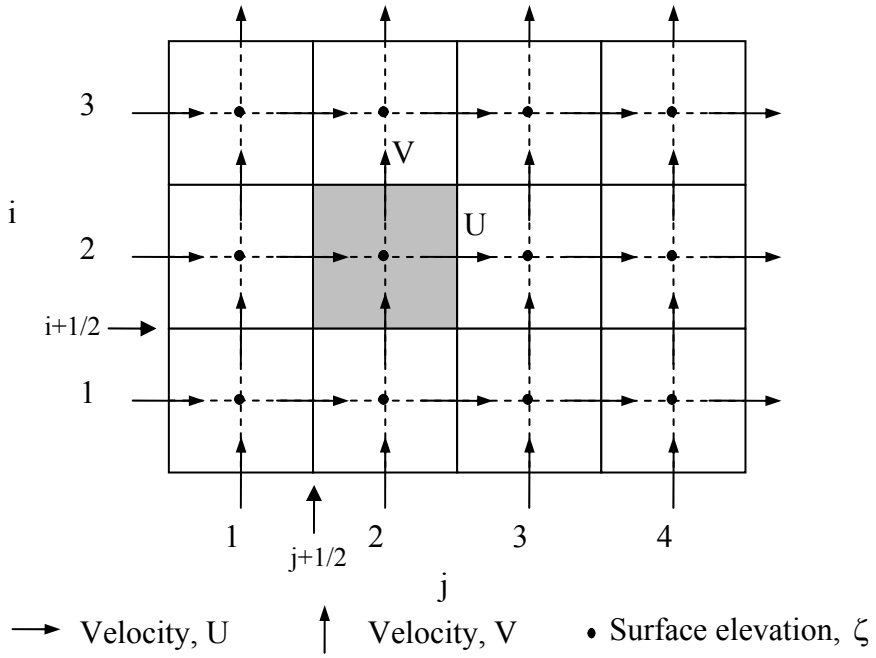


Figure 6.1 Illustration of Arakawa C grid.

As illustrated in Figure 6.1, Arakawa C grid is one kind of staggered grid. Cartesian staggered grid is standard in modeling SWE (Schoenstadt 1980; Stelling and Duinmeijer 2003). In Equations (6.5) to (6.7), the discrete value of the velocity component U is defined at (i, j+1/2); and the value of the velocity component V is defined at (i+1/2, j); and the discrete value of surface elevation $\zeta$ is defined at (i, j). In Equation (6.5), V (i, j+1/2) is obtained by the linear interpolation of the values: V (i+1/2, j), V (i-1/2, j), V (i+1/2, j+1), and V (i-1/2, j+1). In Equation (6.6), U (i+1/2, j) is obtained by the linear interpolation of the values: U (i, j+1/2), U (i+1, j+1/2), U (i, j-1/2), and U (i+1, j-1/2). In Equation (6.7), H (i, j+1/2) is obtained by the linear interpolation of the values of H (i, j) and H (i, j+1). H (I, j-1/2) is obtained by the linear interpolation of the values of H (i, j) and H (i, j-1).

To simplify exposition of the TCS method, we will use the shorthand notation $\delta_x$ and $\delta_y$ for spatial derivatives in x and y, resulting in:

$$U^{n+1} = U^n + \frac{\Delta t}{2}\left[-U^{n+1/2}\delta_x U^{n+1/2} - V^{n+1/2}\delta_y U^{n+1/2} + \nu\left(\delta_x^2 U^{n+1/2} + \delta_y^2 U^{n+1/2}\right) - g\delta_x \zeta^{n+1/2}\right] \quad (6.8)$$

$$V^{n+1} = V^n + \frac{\Delta t}{2}\left[-U^{n+1/2}\delta_x V^{n+1/2} - V^{n+1/2}\delta_y V^{n+1/2} + \nu\left(\delta_x^2 V^{n+1/2} + \delta_y^2 V^{n+1/2}\right) - g\delta_y \zeta^{n+1/2}\right] \quad (6.9)$$

$$\zeta^{n+1} = \zeta^{n} + \frac{\Delta t}{2}\left[-\delta_x\left(H^{n+1/2}U^{n+1/2}\right) - \delta_y\left(H^{n+1/2}V^{n+1/2}\right)\right] \tag{6.10}$$

Equations (6.8) through (6.10) are a nonlinear three-variable coupled time dependent system. However, this complex equation system can be computationally linearized and decoupled by applying the TCS method. The general form of TCS discretized SWE U-velocity equation is:

$$\begin{aligned} U^{*} = U^{n} + \frac{\Delta t}{2}\Big\{&\theta_1\left(-U^{n}\delta_x U^{*}\right) + \left(1-\theta_1\right)\left(-U^{*}\delta_x U^{n}\right) \\ &+ \theta_2\left(-V^{n}\delta_y U^{*}\right) + \left(1-\theta_2\right)\left(-V^{*}\delta_y U^{n}\right) \\ &+ \theta_3\left(\nu\delta_x^2 U^{*}\right) + \left(1-\theta_3\right)\nu\delta_x^2 U^{n} \\ &+ \theta_4\left(\nu\delta_y^2 U^{*}\right) + \left(1-\theta_4\right)\nu\delta_y^2 U^{n} \\ &- \theta_5 g\delta_x\zeta^{*} - \left(1-\theta_5\right)g\delta_x\zeta^{n}\Big\} \end{aligned} \tag{6.11}$$

$$\begin{aligned} U^{n+1} = U^{*} + \frac{\Delta t}{2}\Big\{&\theta_1\left(-U^{n+1}\delta_x U^{*}\right) + \left(1-\theta_1\right)\left(-U^{*}\delta_x U^{n+1}\right) \\ &+ \theta_2\left(-V^{n+1}\delta_y U^{*}\right) + \left(1-\theta_2\right)\left(-V^{*}\delta_y U^{n+1}\right) \\ &+ \theta_3\nu\delta_x^2 U^{*} + \left(1-\theta_3\right)\nu\delta_x^2 U^{n+1} \\ &+ \theta_4\nu\delta_y^2 U^{*} + \left(1-\theta_4\right)\nu\delta_y^2 U^{n+1} \\ &- \theta_5 g\delta_x\zeta^{*} - \left(1-\theta_5\right)g\delta_x\zeta^{n+1}\Big\} \end{aligned} \tag{6.12}$$

where the summation of Equations (6.11) and (6.12) is Equation (6.8). Similarly, for the other equations, it follows that,

$$\begin{aligned} V^{*} = V^{n} + \frac{\Delta t}{2}\Big\{&\theta_6\left(-U^{n}\delta_x V^{*}\right) + \left(1-\theta_6\right)\left(-U^{*}\delta_x V^{n}\right) \\ &+ \theta_7\left(-V^{n}\delta_y V^{*}\right) + \left(1-\theta_7\right)\left(-V^{*}\delta_y V^{n}\right) \\ &+ \theta_8\left(\nu\delta_x^2 V^{*}\right) + \left(1-\theta_8\right)\nu\delta_x^2 V^{n} \\ &+ \theta_9\left(\nu\delta_y^2 V^{*}\right) + \left(1-\theta_9\right)\nu\delta_y^2 V^{n} \\ &- \theta_{10} g\delta_y\zeta^{*} - \left(1-\theta_{10}\right)g\delta_y\zeta^{n}\Big\} \end{aligned} \tag{6.13}$$

$$
\begin{aligned}
V^{n+1} = V^{*} + \frac{\Delta t}{2}\Big\{ & \theta_6\left(-U^{n+1}\delta_x V^{*}\right) + \left(1-\theta_6\right)\left(-U^{*}\delta_x V^{n+1}\right) \\
& + \theta_7\left(-V^{n+1}\delta_y V^{*}\right) + \left(1-\theta_7\right)\left(-V^{*}\delta_y V^{n+1}\right) \\
& + \theta_8 \nu\delta_x^2 V^{*} + \left(1-\theta_8\right)\nu\delta_x^2 V^{n+1} \\
& + \theta_9 \nu\delta_y^2 V^{*} + \left(1-\theta_9\right)\nu\delta_y^2 V^{n+1}\Big\} \\
& - \theta_{10} g\delta_y \zeta^{*} - \left(1-\theta_{10}\right) g\delta_y \zeta^{n+1}\Big\}
\end{aligned}
\tag{6.14}
$$

and the summation of Equations (6.13) and (6.14) is equivalent to Equation (6.9). The two steps of the $\zeta$-equation are:

$$
\begin{aligned}
\zeta^{*} = \zeta^{n} - \frac{\Delta t}{2}\Big\{ & \theta_{11}\delta_x\left(H^{*}U^{n}\right) + \left(1-\theta_{11}\right)\delta_x\left(H^{n}U^{*}\right) \\
& + \theta_{12}\delta_y\left(H^{*}V^{n}\right) + \left(1-\theta_{12}\right)\delta_y\left(H^{n}V^{*}\right)\Big\}
\end{aligned}
\tag{6.15}
$$

$$
\begin{aligned}
\zeta^{n+1} = \zeta^{*} - \frac{\Delta t}{2}\Big\{ & \theta_{11}\delta_x\left(H^{*}U^{n+1}\right) + \left(1-\theta_{11}\right)\delta_x\left(H^{n+1}U^{*}\right) \\
& + \theta_{12}\delta_y\left(H^{*}V^{n+1}\right) + \left(1-\theta_{12}\right)\delta_y\left(H^{n+1}V^{*}\right)\Big\}
\end{aligned}
\tag{6.16}
$$

and the summation of Equation (6.15) and (6.16) is equivalent to Equation (6.10). In the above equation system, $\theta_i$ is the weighting factor and $i \in \{1, 2, \ldots, 12\}$ for each term. As discussed in Chapter 5, properly chosen $\theta_i$ can computationally decouple and linearize the equation system. In the next section, we will illustrate the possible decoupled discretizations for Equations (6.1) through (6.4).

## 6.3 DECOUPLING THE SWE

Observing Equations (6.1) through (6.4), we discover that the advection terms $V\partial U/\partial y$ and $U\partial V/\partial x$ couple the two momentum equations for U and V. $g\partial\zeta/\partial x$ and $\partial HU/\partial x$ couple the U-momentum equation and the continuity equation. $g\partial\zeta/\partial y$ and $\partial HV/\partial y$ couple the V-momentum equation and the continuity equation. Consequently, the weighting factors associated with these terms are crucial in the decoupling process. By examining Equations (6.11) through (6.16), the weighting factors are $\theta_2, \theta_5, \theta_6, \theta_{10}, \theta_{11}$, and $\theta_{12}$. The other weighting factors, including the ones for the viscous terms, $\theta_3, \theta_4, \theta_8, \theta_9$ and the ones for non-coupled advection terms, $\theta_1, \theta_7$, are not involved in the decoupling process. In other words, the values of the weighting factors, $\theta_1, \theta_3, \theta_4, \theta_7, \theta_8$, and $\theta_9$, will not affect the decoupling process.

There are limited combinations of the weighting factors, $\theta_2, \theta_5, \theta_6, \theta_{10}, \theta_{11}$, and $\theta_{12}$ that will produce a completely decoupled equation system. Each combination will create

a unique discretization and a solution order. The principal for the decoupling process is that only one dependent variable can appear in each equation and we can change the number of the variables by changing the values of the weighting factors. This can be better understood by the following example:

- If Equation (6.11) for $U^*$ is chosen to be solved first, $\theta_2 = 1$ and $\theta_5 = 0$ are necessary conditions, because only one dependent variable $U^*$ can be included. Thus, the weighting factors for $V^*$ and $\zeta^*$ have to be equal to 0. It follows that Equation (6.12) for $U^{n+1}$ will include $V^{n+1}$ and $\zeta^{n+1}$. To have a decoupled solution, Equation (6.12) for $U^{n+1}$ has to be solved at the last. This unique solution sequence holds true for each variable. If $V^*$ or $\zeta^*$ is chosen to be solved first, then $V^{n+1}$ or $\zeta^{n+1}$ has to be solved last.
- If Equation (6.13) for $V^*$ is solved right after $U^*$, $\theta_6 = 0$ and $\theta_{10} = 0$ are necessary conditions. In Equation (6.13), $\zeta^*$ cannot be included, so $\theta_{10}$ has to equal to 0, which causes $\zeta^{n+1}$ to be included in Equation (6.14) . To have a decoupled solution, Equation (6.16) for $\zeta^{n+1}$ has to be solved before Equation (6.14) for $V^{n+1}$. Furthermore, in Equation (6.14), $U^{n+1}$ cannot be included since $U^{n+1}$ will be obtained at the end, therefore $\theta_6$ has to equal to 0. After choosing the second equation, two more weighting factors are decided and solution order is decided as $U^*$, $V^*$, $\zeta^*$ then $\zeta^{n+1}$, $V^{n+1}$ and $U^{n+1}$.
- Since $\zeta^{n+1}$ has to be solved before $U^{n+1}$ and $V^{n+1}$, $\theta_{11} = 0$ and $\theta_{12} = 0$ are necessary conditions because $U^{n+1}$ and $V^{n+1}$ cannot be included in Equation (6.16).
- Hence, from the procedure described above, we can see that a solution order determines a unique combination of weighting factors. Since we have six equations in the system, six solution orders exist. Table 6.1 lists all six solution orders and their correspondent combinations of the weighting factors.

Table 6.1 Weighting factors and the solution orders for decoupled Equations (6.11) through (6.16).

| row | $\theta_2$ | $\theta_5$ | $\theta_6$ | $\theta_{10}$ | $\theta_{11}$ | $\theta_{12}$ | solution order |
|---|---|---|---|---|---|---|---|
| a | 0 | 1 | 1 | 0 | 1 | 0 | $V^*, \zeta^*, U^*, U^{n+1}, \zeta^{n+1}, V^{n+1}$ |
| b | 1 | 0 | 0 | 1 | 0 | 1 | $U^*, \zeta^*, V^*, V^{n+1}, \zeta^{n+1}, U^{n+1}$ |
| c | 1 | 0 | 0 | 0 | 0 | 0 | $U^*, V^*, \zeta^*, \zeta^{n+1}, V^{n+1}, U^{n+1}$ |
| d | 0 | 0 | 1 | 0 | 0 | 0 | $V^*, U^*, \zeta^*, \zeta^{n+1}, U^{n+1}, V^{n+1}$ |
| e | 1 | 1 | 0 | 1 | 1 | 1 | $\zeta^*, U^*, V^*, V^{n+1}, U^{n+1}, \zeta^{n+1}$ |
| f | 0 | 1 | 1 | 1 | 1 | 1 | $\zeta^*, V^*, U^*, U^{n+1}, V^{n+1}, \zeta^{n+1}$ |

## 6.4 CHARACTERISTICS OF THE TCS DECOUPLED EQUATION SYSTEM

The main advantage of the TCS method is that it can completely decouple and linearize an equation system with quadratic nonlinearity. As shown in the previous section, multiple TCS decoupled and linearized discretizations exist. To explore this characteristic of the TCS method, we choose values from row a, c and e in Table 6.1 to create three different TCS discretizations since the continuity equation is solved in different orders in these three rows.

Choosing weighting factors from row a, Equations (6.11) to (6.16) become a two step computationally linearized and decoupled equation system. The implicit steps for solution of intermediate variables ( $U^*, V^*, \zeta^*$ ) are

$$V^* = V^n + \frac{\Delta t}{2}\left[-U^n\delta_x V^* - V^n\delta_y V^* + \nu\left(\delta_x^2 V^* + \delta_y^2 V^*\right) - g\delta_x\zeta^n\right] \tag{6.17}$$

$$\zeta^* = \zeta^n - \frac{\Delta t}{2}\left[\delta_x\left(H^* U^n\right) + \delta_y\left(H^n V^*\right)\right] \tag{6.18}$$

$$U^* = U^n + \frac{\Delta t}{2}\left[-U^*\delta_x U^n - V^*\delta_y U^n + \nu\left(\delta_x^2 U^* + \delta_y^2 U^*\right) - g\delta_x\zeta^*\right] \tag{6.19}$$

where $\zeta^* - Z_b = H^*$. The implicit steps for the 'n+1' values are sequentially solved as

$$U^{n+1} = U^* + \frac{\Delta t}{2}\left[-U^*\delta_x U^{n+1} - V^*\delta_y U^{n+1} + \nu\left(\delta_x^2 U^* + \delta_y^2 U^*\right) - g\delta_x\zeta^*\right] \tag{6.20}$$

$$\zeta^{n+1} = \zeta^* - \frac{\Delta t}{2}\left[\delta_x\left(H^* U^{n+1}\right) + \delta_y\left(H^{n+1} V^*\right)\right] \tag{6.21}$$

$$V^{n+1} = V^* + \frac{\Delta t}{2}\left[-U^{n+1}\delta_x V^* - V^{n+1}\delta_y V^* + \nu\left(\delta_x^2 V^* + \delta_y^2 V^*\right) - g\delta_y\zeta^{n+1}\right] \tag{6.22}$$

As we stated in section 6.3, weighting factors for the viscous terms and non-coupled advection terms will not affect the decoupling process, so the values for these weighting factors can be anything from 0 to 1. In Equations (6.17) through (6.22), the weighting factors for the viscous terms: $\theta_3 = \theta_4 = \theta_8 = \theta_9 = 1$ and the weighting factors for the non-coupled advection terms: $\theta_1 = 1$ and $\theta_7 = 1$. Thus the original set of the coupled nonlinear equations becomes a sequence of linear implicit equations for each variable. All variables in Equations (6.17) through (6.22) are solved in order as: $V^*, \zeta^*, U^*, U^{n+1}, \zeta^{n+1}$, and $V^{n+1}$. We call the solution from the above equation system "TCS solution 1".

In the same vein, if we choose the weighting factors at row c combining $\theta_3 = \theta_4 = \theta_8 = \theta_9 = 1$, $\theta_1 = 1$ and $\theta_7 = 1$, a different decoupled and linearized discretization is obtained. In the first step, intermediate variables $(U^*, V^*, \zeta^*)$ are solved as

$$U^* = U^n + \frac{\Delta t}{2}\left[-U^n\delta_x U^* - V^n\delta_y U^* + \nu\left(\delta_x^2 U^* + \delta_y^2 U^*\right) - g\delta_x\zeta^n\right] \tag{6.23}$$

$$V^* = V^n + \frac{\Delta t}{2}\left[-U^*\delta_x V^n - V^*\delta_y V^n + \nu\left(\delta_x^2 V^* + \delta_y^2 V^*\right) - g\delta_y\zeta^n\right] \tag{6.24}$$

$$\zeta^* = \zeta^n - \frac{\Delta t}{2}\left[\delta_x\left(H^n U^*\right) + \delta_y\left(H^n V^*\right)\right] \tag{6.25}$$

and in the second step,

$$\zeta^{n+1} = \zeta^* - \frac{\Delta t}{2}\left[\delta_x\left(H^{n+1} U^*\right) + \delta_y\left(H^{n+1} V^*\right)\right] \tag{6.26}$$

$$V^{n+1} = V^* + \frac{\Delta t}{2}\left[-U^*\delta_x V^{n+1} - V^*\delta_y V^{n+1} + \nu\left(\delta_x^2 V^* + \delta_y^2 V^*\right) - g\delta_y\zeta^{n+1}\right] \tag{6.27}$$

$$U^{n+1} = U^* + \frac{\Delta t}{2}\left[-U^{n+1}\delta_x U^* - V^{n+1}\delta_y U^* + \nu\left(\delta_x^2 U^* + \delta_y^2 U^*\right) - g\delta_x\zeta^{n+1}\right] \tag{6.28}$$

The solution order of the equation set (6.17) to (6.22) is changed to $U^*, V^*, \zeta^*, \zeta^{n+1}, V^{n+1}, U^{n+1}$. The solution from Equations (6.23) to (6.28) is called "TCS solution 2".

Similarly, anther set of equations can be obtained by using the weighting factors in row e combining $\theta_3 = \theta_4 = \theta_8 = \theta_9 = 1$, $\theta_1 = 1$ and $\theta_7 = 0$. In the first step,

$$\zeta^* = \zeta^n - \frac{\Delta t}{2}\left[\delta_x\left(H^* U^n\right) + \delta_y\left(H^* V^n\right)\right] \tag{6.29}$$

$$U^* = U^n + \frac{\Delta t}{2}\left[-U^n\delta_x U^* - V^n\delta_y U^* + \nu\left(\delta_x^2 U^* + \delta_y^2 U^*\right) - g\delta_x\zeta^*\right] \tag{6.30}$$

$$V^* = V^n + \frac{\Delta t}{2}\left[-U^*\delta_x V^n - V^*\delta_y V^n + \nu\left(\delta_x^2 V^* + \delta_y^2 V^*\right) - g\delta_y\zeta^*\right] \tag{6.31}$$

and in the second step,

$$V^{n+1} = V^* + \frac{\Delta t}{2}\left[-U^*\delta_x V^{n+1} - V^*\delta_y V^{n+1} + \nu\left(\delta_x^2 V^* + \delta_y^2 V^*\right) - g\delta_y\zeta^*\right] \tag{6.32}$$

$$U^{n+1} = U^* + \frac{\Delta t}{2}\left[-U^{n+1}\delta_x U^* - V^{n+1}\delta_y U^* + \nu\left(\delta_x^2 U^* + \delta_y^2 U^*\right) - g\delta_x\zeta^*\right] \tag{6.33}$$

$$\zeta^{n+1} = \zeta^* - \frac{\Delta t}{2}\left[\delta_x\left(H^*U^{n+1}\right) + \delta_y\left(H^*V^{n+1}\right)\right] \tag{6.34}$$

In the above equation system, the solution order is $\zeta^*, U^*, V^*, V^{n+1}, U^{n+1}, \zeta^{n+1}$. We call this "TCS solution 3".

Each TCS discretization has a different solution procedure. Table 6.2 provides the detailed solution procedures for each TCS method.

Table 6.2 Solution procedures of the three TCS discretizations

| Step | TCS solution 1 | TCS solution 2 | TCS solution 3 |
|---|---|---|---|
| 1 | $V^* \leftarrow U^n, V^n, \zeta^n$ | $U^* \leftarrow U^n, V^n, \zeta^n$ | $\zeta^* \leftarrow U^n, V^n, \zeta^n$ |
| | $\zeta^* \leftarrow U^n, V^*, \zeta^n$ | $V^* \leftarrow U^*, V^n, \zeta^n$ | $U^* \leftarrow U^n, V^n, \zeta^*$ |
| | $U^* \leftarrow U^n, V^*, \zeta^*$ | $\zeta^* \leftarrow U^*, V^*, \zeta^n$ | $V^* \leftarrow U^*, V^n, \zeta^n$ |
| 2 | $U^{n+1} \leftarrow U^*, V^*, \zeta^*$ | $\zeta^{n+1} \leftarrow U^*, V^*, \zeta^*$ | $V^{n+1} \leftarrow U^*, V^*, \zeta^*$ |
| | $\zeta^{n+1} \leftarrow U^{n+1}, V^*, \zeta^*$ | $V^{n+1} \leftarrow U^*, V^*, \zeta^{n+1}$ | $U^{n+1} \leftarrow U^*, V^{n+1}, \zeta^*$ |
| | $V^{n+1} \leftarrow U^{n+1}, V^*, \zeta^{n+1}$ | $U^{n+1} \leftarrow U^*, V^{n+1}, \zeta^{n+1}$ | $\zeta^{n+1} \leftarrow U^{n+1}, V^{n+1}, \zeta^*$ |

## 6.5 NUMERICAL TESTS

To test the general performance of the TCS method in solving 2D depth averaged SWE, a series of numerical experiments are presented in this section. In these numerical experiments, we want to test 1) how the method treats the nonlinear terms such as $U\partial V/\partial x$, $V\partial U/\partial y$, $\partial HU/\partial x$ and $\partial HV/\partial y$; 2) how the method decouples the terms U, V, and H; and 3) How different discretizations and different solution procedures affect the solutions.

A one-dimensional standing wave in a closed basin is first tested to compare our numerical model with the analytical solution. A two-dimensional standing wave in a closed basin is simulated to investigate the different characteristics among three TCS

solutions. As an initial attempt to simulating an open boundary case, a one-dimensional wave traveling through an open boundary system is tested in Appendix A.

### 6.5.1 One dimensional Standing Waves

As a first test of the TCS method, a free surface standing wave in a rectangular basin has been simulated and is shown schematically in Figure 6.2. Free-slip boundary conditions are enforced on all walls. An equally-spaced $20\times5$ mesh is applied to the computational domain.

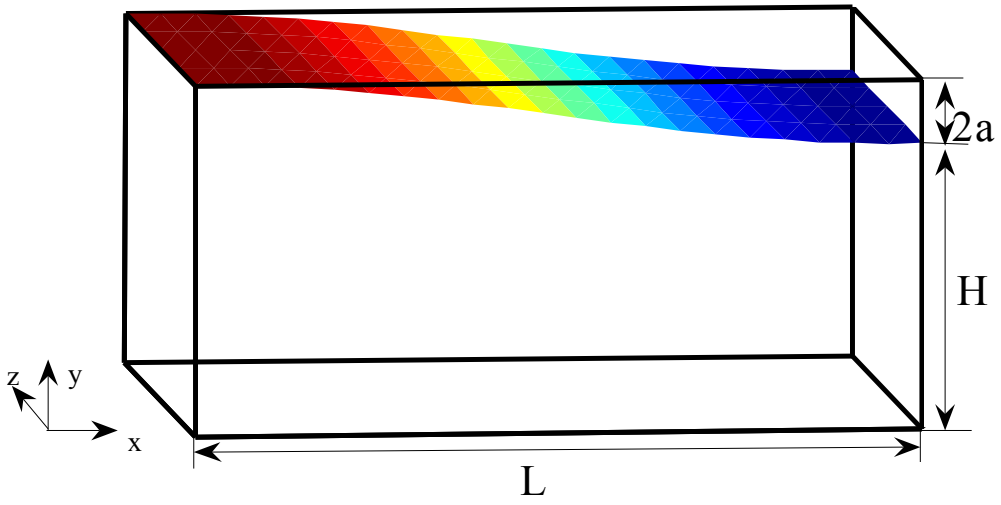


Figure 6.2 A standing wave in a rectangular basin

***Analytical Solution***

The period of an inviscid free surface wave is (Dean and Dalrymple 1998):

$$T=\sqrt{\frac{2\pi\lambda}{g}\left[\tanh\left(kH\right)\right]^{-1}} \tag{6.35}$$

According to linear wave theory, the wave function is a sinusoid for small amplitude waves. The surface height (h) above the still water level is:

$$h(x,t)=a\sin\left(kx\right)\sin\left(\sigma t\right) \tag{6.36}$$

where the frequency ($\sigma$) is $2\pi/T$. For a viscous unconfined wave in deep water, the evolution of the wave amplitude can be approximated as (Lamb 1932),

$$a(t)=a(0)e^{-2\nu k^2 t} \tag{6.37}$$

where ‘a’ is the free surface wave amplitude, ‘$\nu$’ is kinematic viscosity and ‘$k$’ is the wavenumber. However, wave damping based on Equation (6.37) is unlikely to be well-represented in a 2D depth-averaged model as vertical velocity and vertical velocity shears are part of the closure term. The Reynolds number is defined as:

$$\mathrm{Re} = \frac{Lu}{\nu} \tag{6.38}$$

where $L$ is the length of the basin, $\nu$ is the kinematic viscosity and $u$ is the characteristic Cartesian fluid velocity based on the wave amplitude (a) and the wave frequency ($\sigma$) :

$$u = a\sigma \tag{6.39}$$

***Results and Discussion***

Simulations have been run for different cases to examine a variety of model characteristics. Three simulations with different initial wave steepness (i.e. ratio of wave amplitude to basin length) are presented in Figure 6.3. The three simulations were run in an inviscid flow to minimize the viscous damping effects. In Figure 6.3, results of the simulations illustrate the nondimensionalized water surface elevation (i.e. $(\zeta - H)/a$) at $x = \Delta x/2$, which is the grid cell center closest to the left wall. The important parameters for these three test cases are listed in Table 6.3.

Table 6.3 Parameters of simulations of a 1D standing wave

| case | H/L | a/L | $L/\lambda$ |
|---|---|---|---|
| 6.4a | 0.05 | 0.05% | 0.5 |
| 6.4b | 0.05 | 0.2% | 0.5 |
| 6.4c | 0.05 | 0.5% | 0.5 |

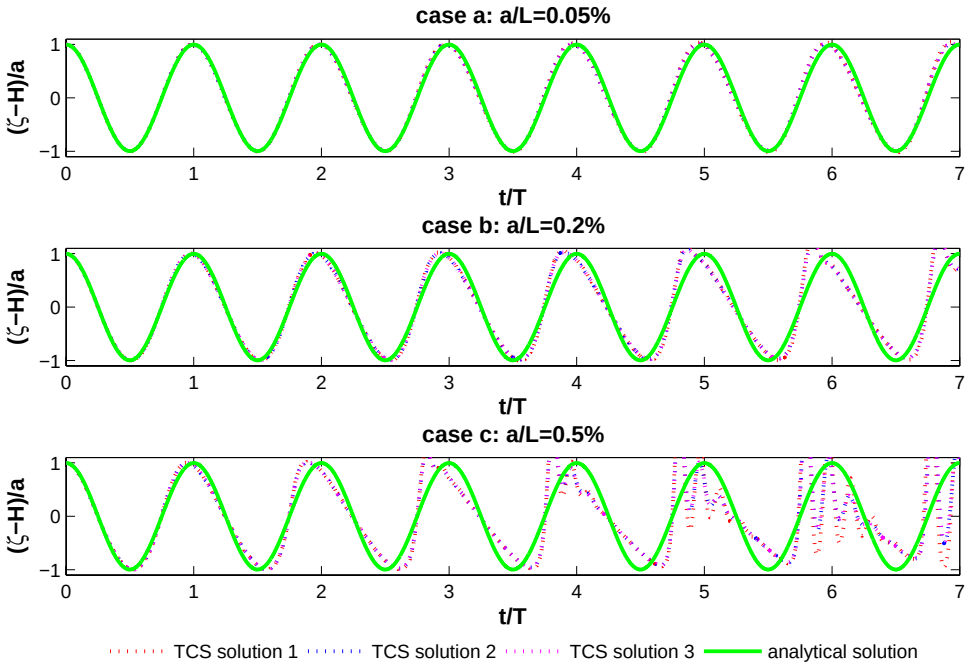


Figure 6.3 Simulations in the inviscid flow with different wave steepness

Figure 6.3 shows results of the three different initial wave steepness. The non-linearity of the advection term causes nonlinear wave steepening. Hydrostatic approximation from the depth averaged SWE enhances the nonlinear steepening because the dispersive effects from the non-hydrostatic effects are neglected (Daily and Imberger 2003). This nonlinear steepening can be observed from Figure 6.3. Furthermore, with the increase of the nonlinearity of the wave (i.e. the initial wave steepness), the nonlinear-steepening effects increase. In case 6.3a, a very small initial wave steepness is introduced and the simulation shows little nonlinear effects. When the initial steepness is increased in case 6.3b and 6.3c, the nonlinear effects become apparent. In theory, a series of solitary wave can be formed when the nonlinear steepening is balanced by the non-hydrostatic pressure gradient (Miropol'sky 2001). However, In Figure 6.3c, a train of solitary waves can be observed from the TCS simulations, which are solutions under hydrostatic approximation. This is probably due to the numerical dispersion, which plays an opposition effect to the nonlinear steepening. This phenomena has been reported in other literatures (Hodges et al. 2006; Wadzuk 2004).

Figure 6.4 displays the viscous damping effects in our shallow water model. The Reynolds number is equal to 0.1, which indicates a very viscous flow. The simulation results damped faster than the analytical solution. This is mainly a result of the staggered grids used in the numerical model. The staggered grids necessitate averaging from the cell surfaces, which creates numerical dissipation (Garcia and Kahawita 1986).

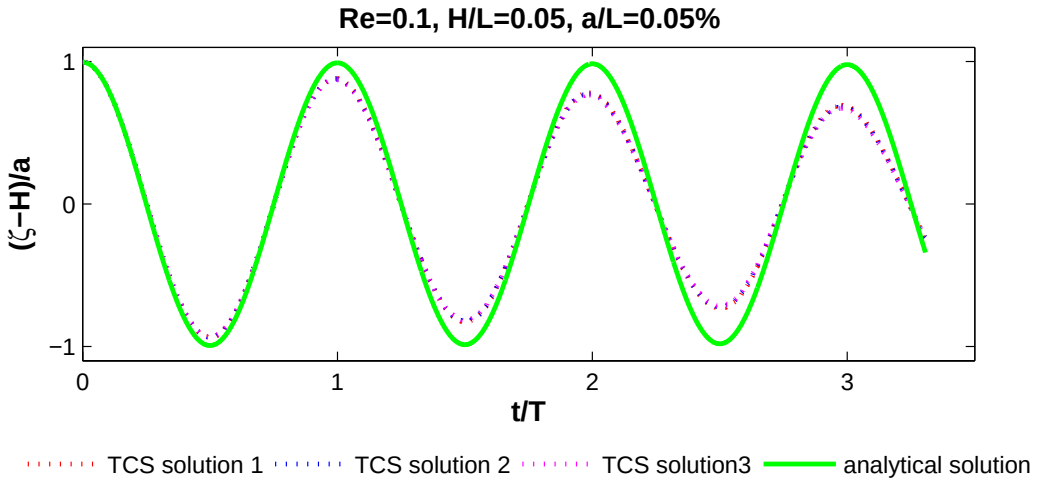


Figure 6.4 Viscous damping effects in the shallow water model

In Figures 6.3 and 6.4, the results from the three TCS discretizations are indistinguishable. To have a better comparison among three TCS solutions, we run a simulation of a two dimensional standing wave in a square box in the next section.

### 6.5.2 Two-dimensional Standing Waves

In this test case, the ability of the TCS method to simulate a two dimensional finite-amplitude free surface wave in a square box is investigated and the performances from different TCS discretizations are compared.

A two-dimensional free surface standing wave in a square box is shown in Figure 6.5. This initial wave is linearly composed by two one-dimensional (one in x-direction and the other one in y direction) identical and orthogonal standing waves, which is also shown in Figure 6.5. This superimposed condition creates a two dimensional standing wave that is exactly symmetric to the diagonal plane of the square box. The horizontal scale of the square box is $10\text{m} \times 10\,\text{m}$, and the initial still water elevation is 0.5m. The initial wave amplitude for the two one-dimensional waves is 0.005m. The computational domain is illustrated in Figure 6.6. The surface elevation is measured from the still water surface and nondimensionalized by the wave amplitude which is the summation of the one dimensional wave amplitudes. The horizontal length scales are normalized by the wave length. A $20 \times 20\,\text{mesh}$ is used in this simulation and the free slip condition is enforced on all the side walls. The simulation is run in an inviscid flow condition to eliminate the viscous effects. Table 6.4 listed all the important parameters in the simulations of this 2D standing wave.

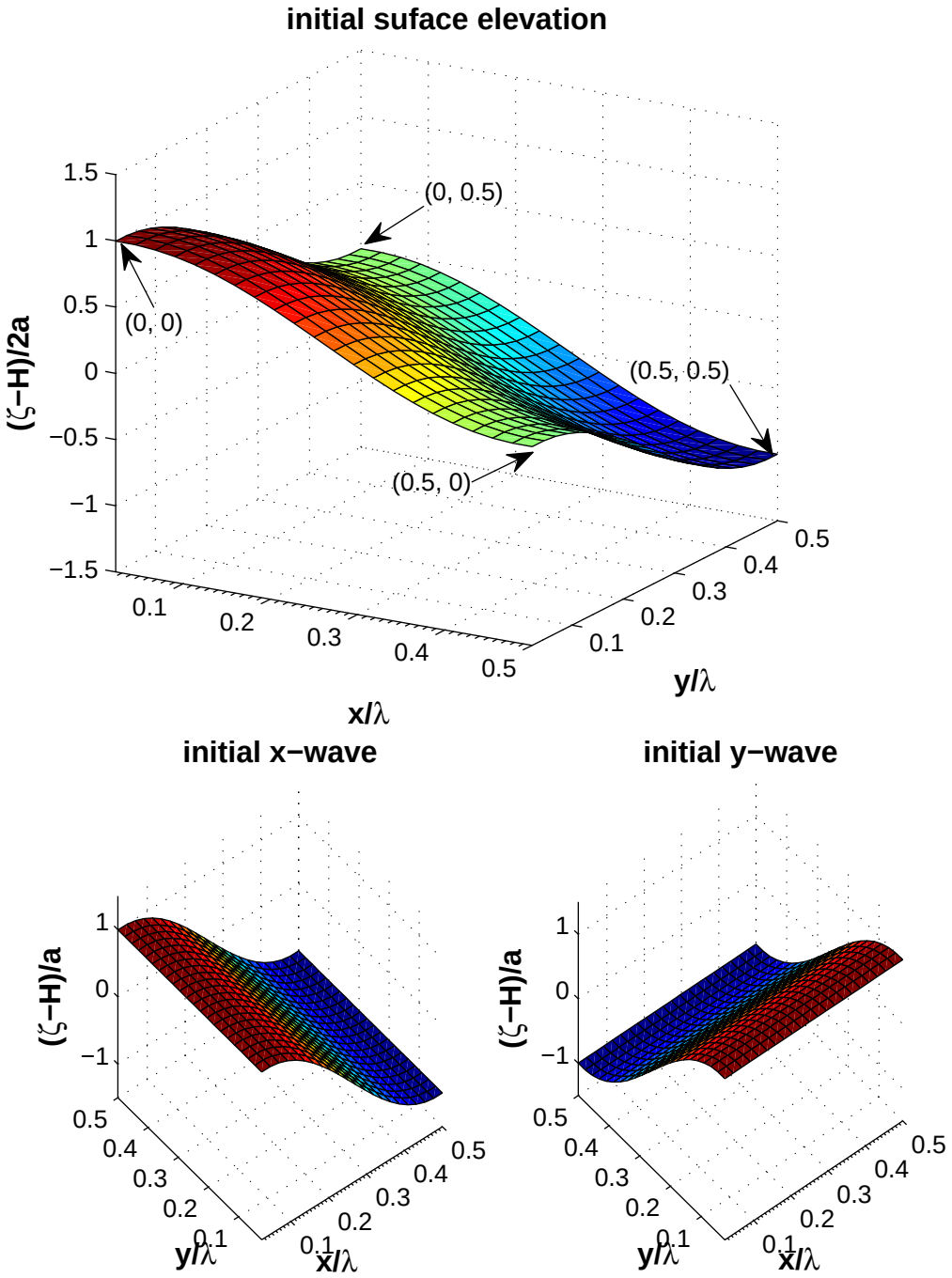


Figure 6.5 The initial 2D standing wave and its decomposed x and y direction 1D wave

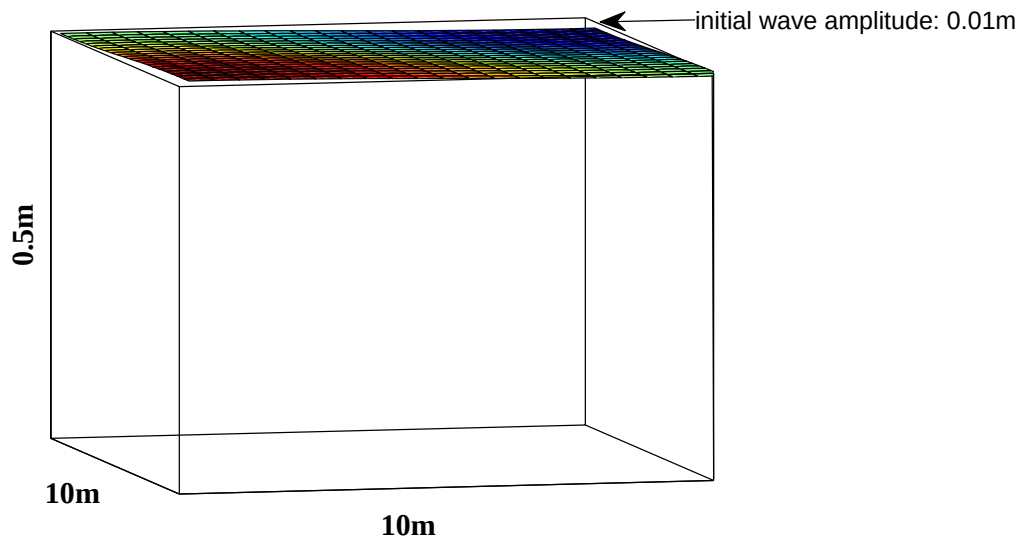


Figure 6.6 Computational domain (not to scale)

Table 6.4 Parameters of simulations of the 2D standing wave

| L | $\lambda$ | a | H | $\Delta x$ | $\Delta y$ | $\Delta t$ | $\nu$ |
|---|---|---|---|---|---|---|---|
| 10m | 20m | 0.005m | 0.5m | 0.5m | 0.5m | 0.01s | 0 |

Although six representative decoupled linearized TCS discretizations and their correspondent solution orders are listed in Table 6.1, only three discretizations need to be tested here because velocity U and V are given symmetrically. In TCS solution 1, the surface elevation $\zeta^*$ and $\zeta^{n+1}$ are solved in between two velocity equations; In TCS solution 2, the surface elevation $\zeta^*$ is solved after the two velocity equations and $\zeta^{n+1}$ is solved before the two velocity equations; In TCS solution 3, $\zeta^*$ is solved before the two velocity equations and $\zeta^{n+1}$ is solved after the two velocity equations.

To explore how different discretizations and solution orders affect the results. We first compare the results from TCS solutions 1, 2 and 3 at different simulation times. The simulation time is normalized by the wave period, T. In Figure 6.7, the surface elevation contour calculated using these three TCS forms are plotted for the whole computational domain at time 4T. The surface elevation is measured from the still water surface and normalized by the amplitude. The horizontal length scale is normalized by the wave length, $\lambda$. The surface elevation contour from TCS solution 2 and 3 are nearly identical, and the result from TCS solution 1 behaves differently than that from solution 2 and 3.

The result from each solution has a symmetric shape. When the simulation time is increased to 15T (results shown in Figure 6.8), TCS solution 2 and 3 still perform similarly, but TCS solution 1 has an observable difference. Similar phenomena can be better observed in the velocity U and V contours. Figure 6.9 and 6.10 demonstrate the U-velocity simulated using three TCS methods at 4T and 15 T respectively. Figure 6.11 and 6.12 display the V-velocity at 4T and 15T respectively. Each velocity component in the figures is normalized by its maximum value. The results form TCS solution 2 and 3 are almost the same, but the result from TCS solution is different.

This can be analyzed by the solution procedures of the three tested TCS discretizations presented in Table 6.2. First of all, in TCS solution 1, the continuity equation (solving $\zeta$) is solved in between two momentum equations (solving U and V); but in both TCS solutions 2 and 3, the two momentum equations are solved consecutively. The only difference between TCS solution 2 and 3 is that one solve momentum equations first, the other solve continuity equation first in one time step. However, this difference between solution 2 and 3 could be reduced with the marching of the simulation time. Second, in TCS solution 1, the surface elevation $\zeta^*$ is solved using the velocity components, $U^n$ and $V^*$ in the first step; and $\zeta^{n+1}$ is solved using the velocity components, $U^{n+1}$ and $V^*$ in the second step. Thus, in TCS solution 1, the surface elevation is obtained using a velocity vector field which is constructed by the components from different time steps. Third, we further discover that water depth H is not used symmetrically in the continuity equation. In Equation (6.18), $H^n$ is used in the x direction, but $H^*$ is used in the y direction to solve $\zeta^*$ in the first step; in Equation (6.21), $H^*$ is used in x direction but $H^{n+1}$ is used in the y direction to solve $\zeta^{n+1}$. All of these asymmetric solution procedures introduce extra numerical errors since our initial condition is symmetric. However, in both TCS solutions 2 and 3, the surface elevation is obtained using the velocity components and water depth at the same time step although the velocity field and water depth are not exactly symmetric because of the sequential solution procedure. The difference caused by different solution procedure is further exemplified by a cross mode analysis in the following.

With the given symmetric initial condition, ideally, we should expect this standing wave oscillates along one diagonal line and no wave motion should be expected in the cross direction. That means, the wave surface oscillates at point (0, 0) and (0.5, 0.5) with an amplitude 0.01m ($2 \times 0.005$m), and the wave surface should be fixed at point (0.5, 0) and (0, 0.5). However, the TCS decoupled equation system is solved in a sequential order, which causes each equation to be solved anisotropically. This in turn induces numerical error that causes cross mode wave motion. The similar phenomena have been observed in Alternate Direction Implicit (ADI) method. In ADI method, this cross mode is induced by solving the equation in different coordinate direction as observed by Hodges (1997). To examine how cross mode is established in the three TCS discretizations, we plot the surface displacement at point (0.5, 0) and (0, 0.5) against simulation time in Figure 6.13. The surface displacement is normalized by the wave amplitude and the simulation time is normalized by wave period. The results from TCS solutions 2 and 3 are indistinguishable. The cross mode at the two monitored points are

nearly identical in these two TCS solutions and gradually increase with time. At time 15 T, the surface displacement at those two points is almost 25% of the wave amplitude. In contrast, this numerical cross mode from solution 1 behaves rather differently. The surface at the two monitored points oscillates with an amplitude and a period the same as the initial standing wave. In addition, the oscillation of the cross direction is one half phase behind the initial standing wave. To have a better visualization of the cross mode, Figure 6.14 and Figure 6.15 display the snapshots of the standing wave at 0.5 T and 15T respectively. At 0.5 T, no obvious motion can be observed at the two monitored points solved from TCS solutions 2 and 3. With the increase of the simulation time, at time 15 T, an observable surface displacement can be discovered from both TCS solution 2 and 3. In contrast, a maximum oscillation with a displacement of 2a can be observed for TCS solution 1 at 0.5 T, but no obvious motion can be found at time 15T. This is probably because in TCS solution 1, the surface elevation is solved using asymmetric velocity components and water depth components. There is a half time step difference between x and y direction in both velocity and water depth components, which causes the wave at the cross direction oscillates with a half phase lag.

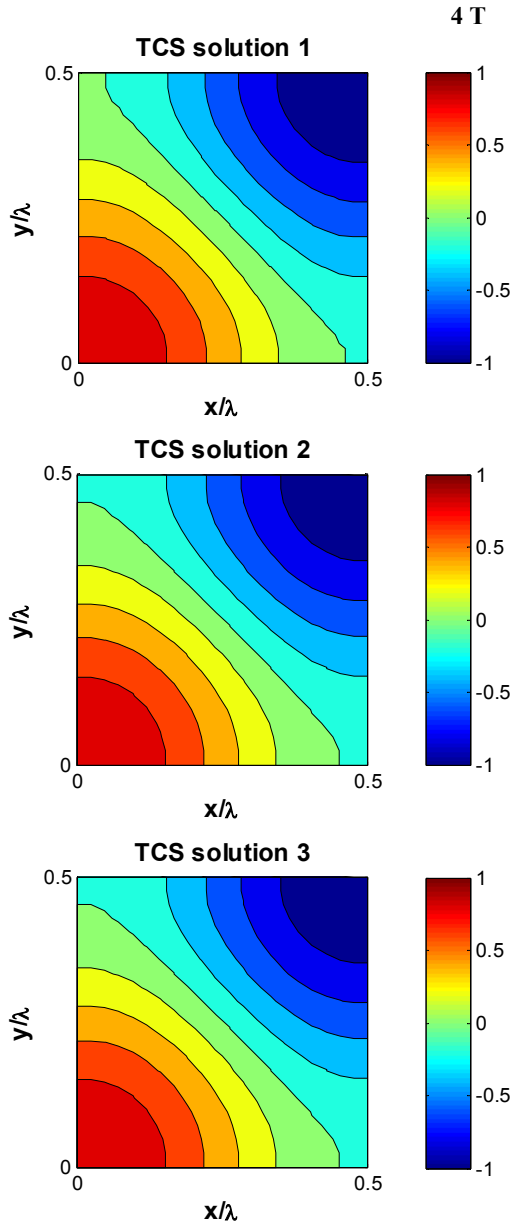


Figure 6.7 Normalized surface elevation contours in the computational domain at time 4T.

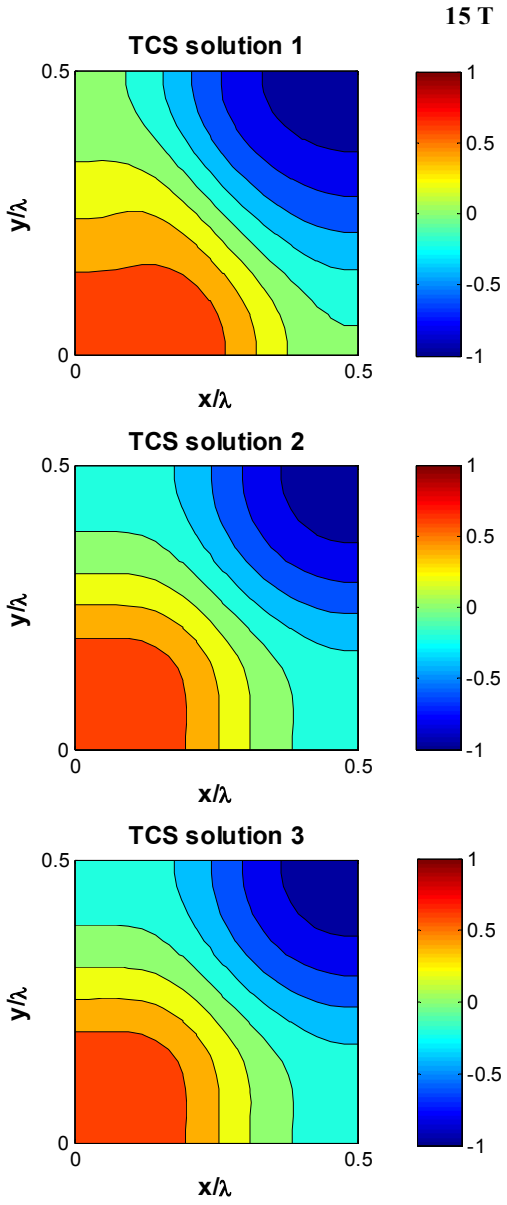


Figure 6.8 Normalized surface elevation contours in the computational domain at time 15T.

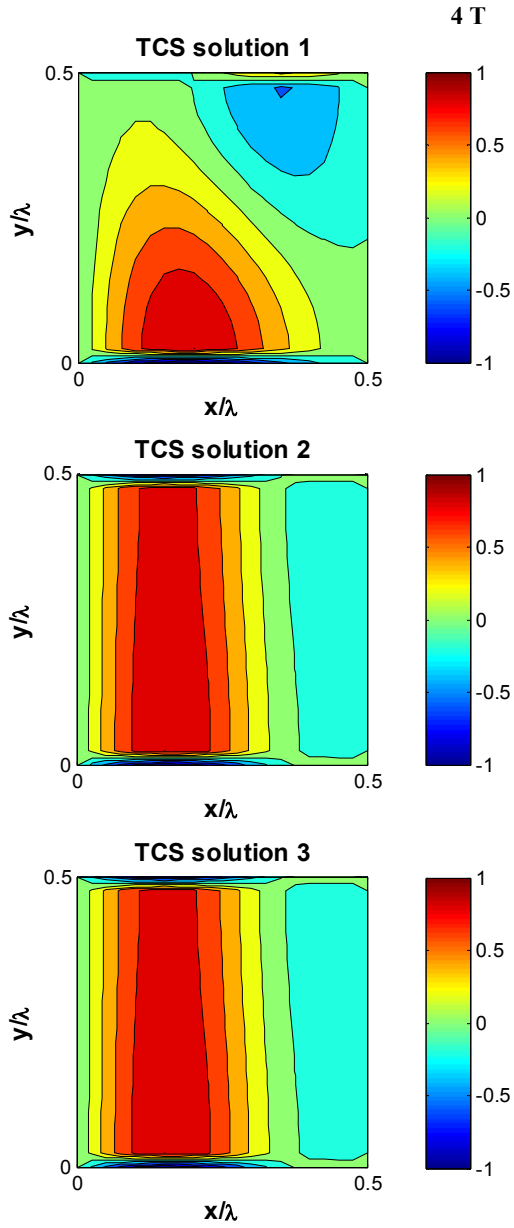


Figure 6.9 Normalized U-velocity contours in the computational domain at time 4T.

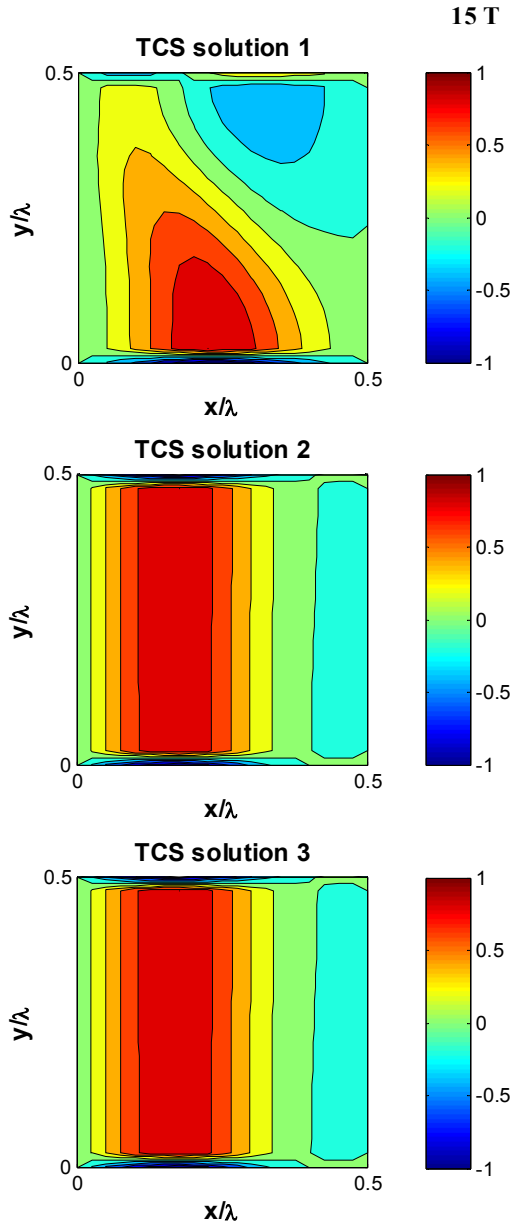


Figure 6.10 Normalized U-velocity contours in the computational domain at time 15T.

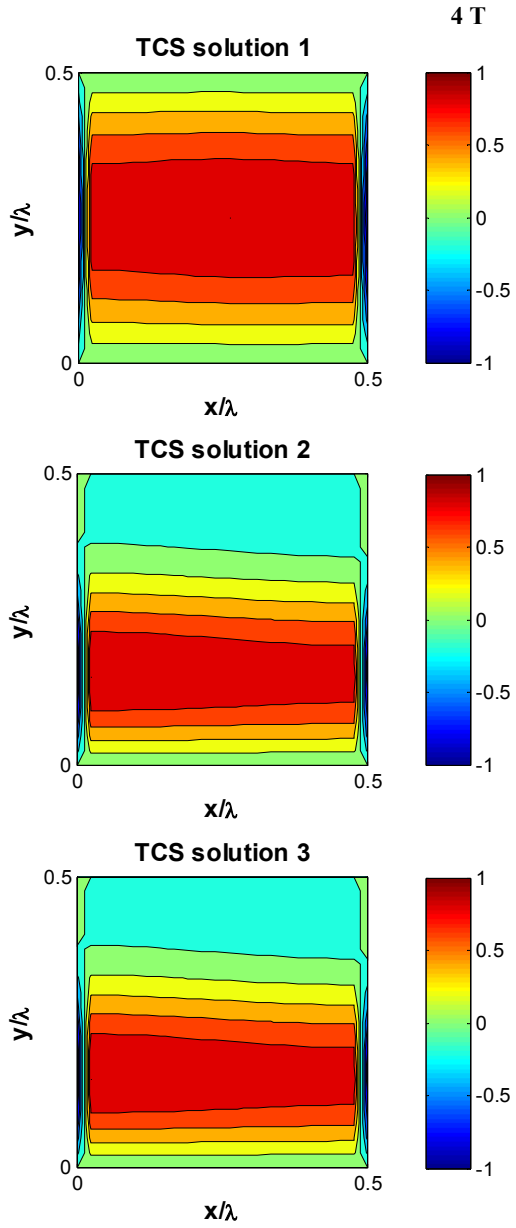


Figure 6.11 Normalized V-velocity contours in the computational domain at time 4T.

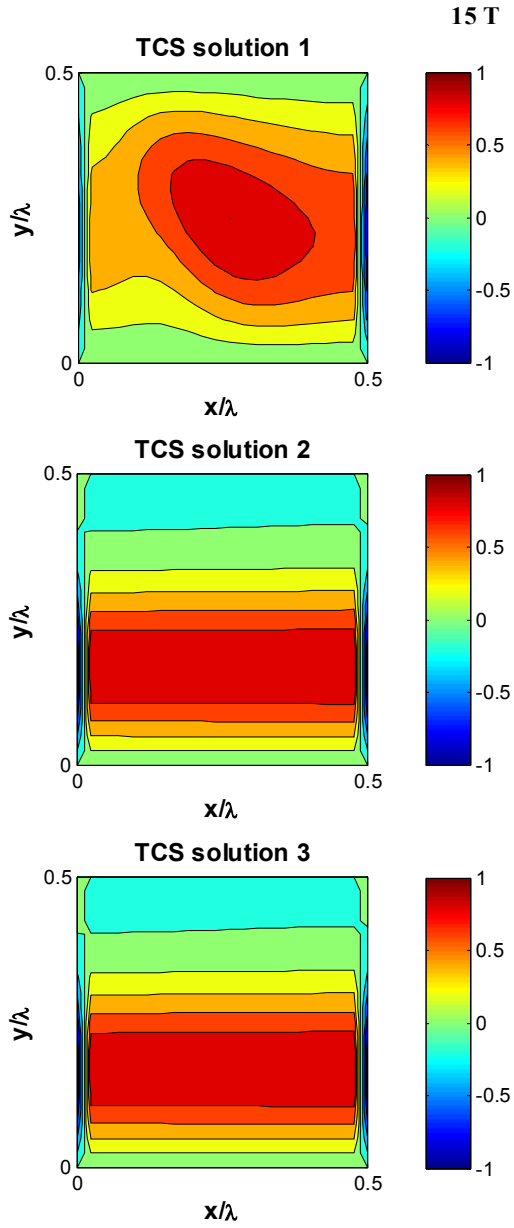


Figure 6.12 Normalized V-velocity contours in the computational domain at time 15T.

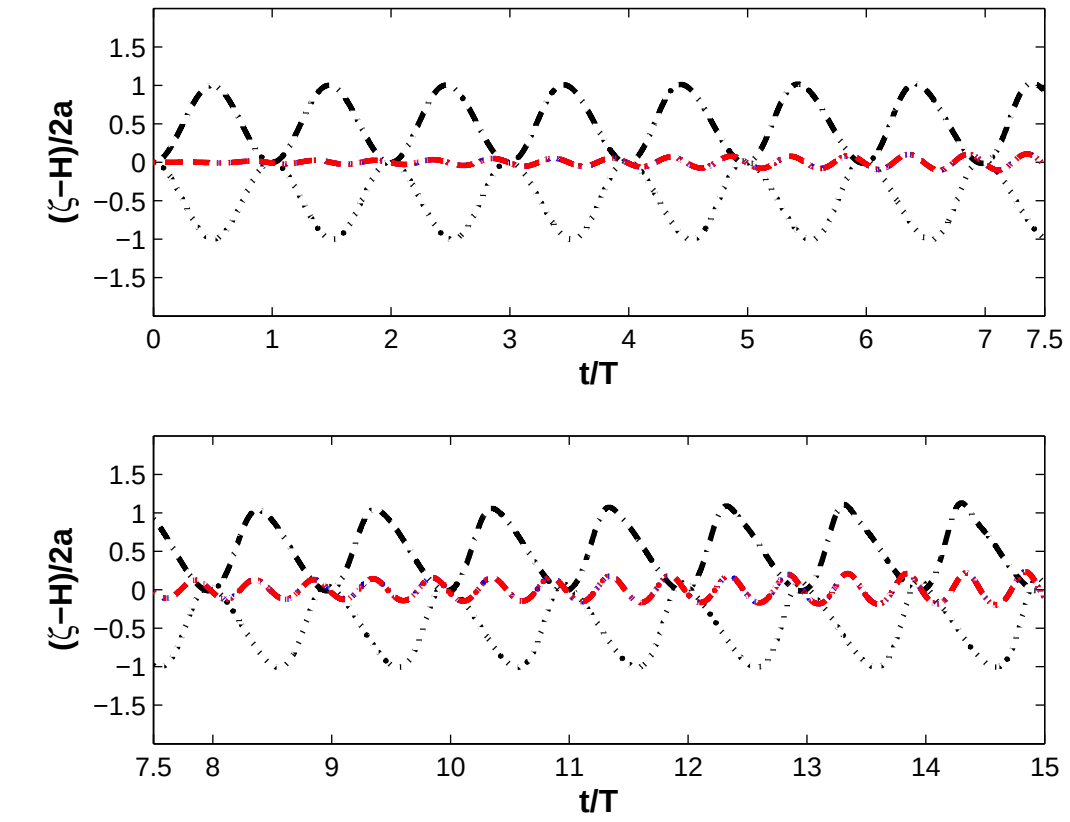


Figure 6.13 Surface displacement at points (0.5, 0) and (0, 0.5) simulated using three TCS solutions

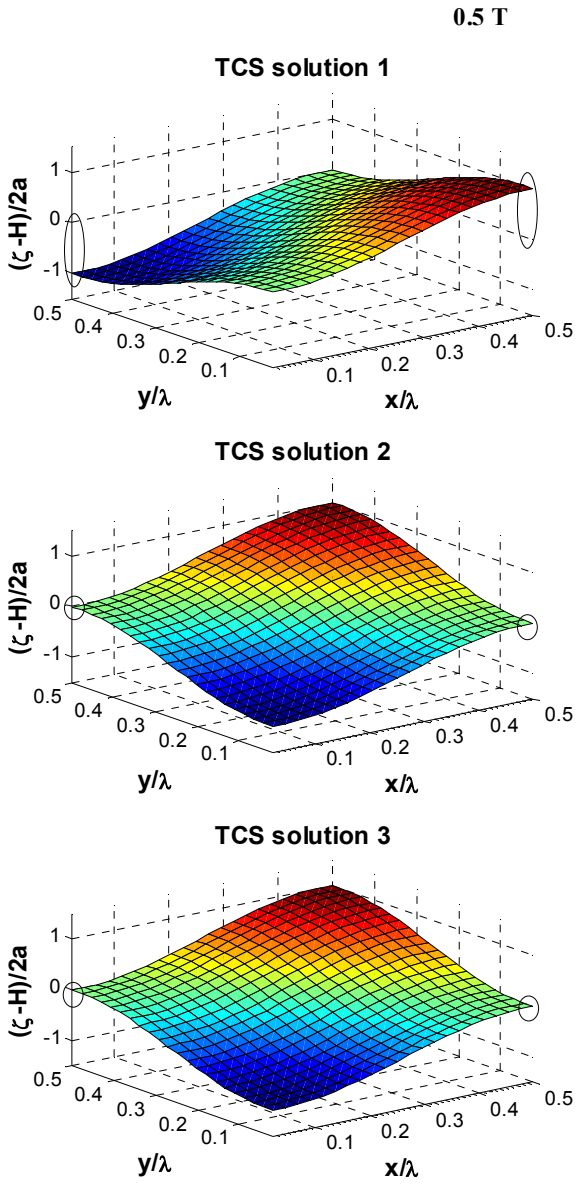


Figure 6.14 Snapshot of the standing wave at 0.5 T. The surface displacements at the two monitored points (0.5, 0) and (0, 0.5) are circled.

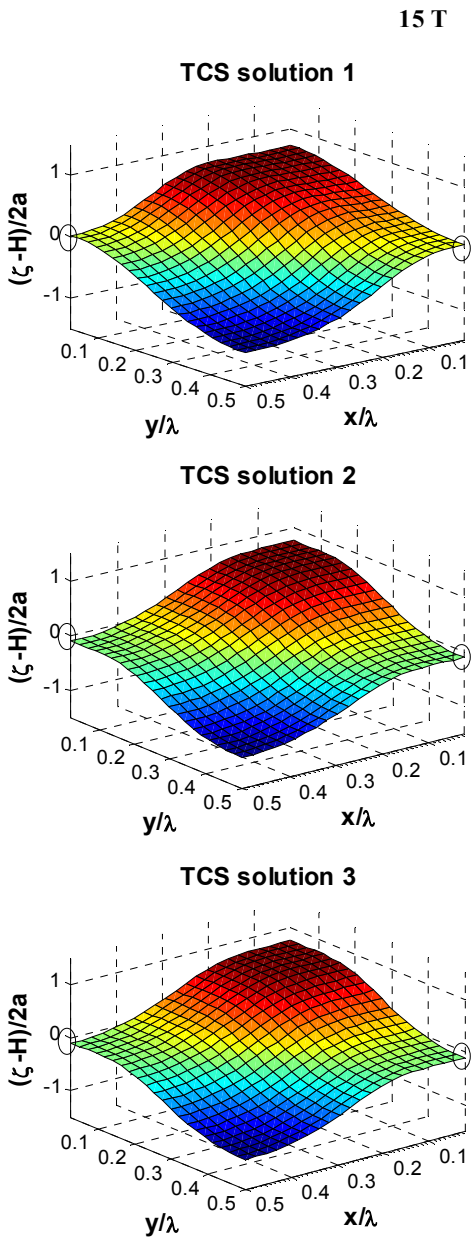


Figure 6.15 Snapshot of the standing wave at 15 T. The surface displacements at the two monitored points (0.5, 0) and (0, 0.5) are circled.

## 6.6 SUMMARY

In this chapter, the TCS method is applied to a multi-variable and multi-dimension equation system: the depth averaged SWE. The TCS method not only computationally discretizes the nonlinear term in the equation set, but also decouples the variables in the SWE. Three TCS discretizations and are derived for the SWE. Each TCS decoupled and linearized equation system has a unique solution order.

A one-dimensional and two-dimensional standing waves in a closed rectangular domain are simulated using all three TCS discretizations. In the one dimensional standing wave case, we compare our numerical results with the analytical solutions. The results demonstrate that our TCS method well capture the nonlinear effects and display the combination effects of nonlinear steepening and hydrostatic approximation. In the viscous flow test case, the staggered grids used in the simulation causes numerical dissipation. In the two-dimensional standing wave case, we compare the performance from three representative TCS solutions: TCS solutions 1, 2 and 3. TCS solutions 2 and 3 are solved similarly because the surface elevation is obtained using the velocity components at the same time step. However, TCS solution 1 performs differently because the surface elevation is obtained using the velocity components from different time step.

The anisotropically solution process also causes numerical cross mode. The cross mode analysis shows that TCS solution 2 and 3 has smaller numerical cross mode compare to TCS solution 1. However, the cross mode gradually increase with simulation time. The solution procedure in TCS solution 1 causes wave oscillate in the cross direction. Therefore, when we apply TCS method to decouple the equation system, we have to be aware of the numerical errors generated by the decoupling process.

# Chapter 7 Conclusions and Recommendations

The main objective for this research is:

> Develop a new implicit solution for unsteady nonlinear advection problems. The new numerical algorithm should have the advantage in both stability and efficiency while keep the $2^{nd}$-order temporal accuracy as in the most existing shallow water equations (SWE) models. In addition, this new method would provide a novel approach in decoupling a coupled equation system.

This objective has been accomplished. A new finite difference temporal scheme (Time-centered split method) is developed to solve the unsteady nonlinear advection problems. This new method provides a general approach for solving any unsteady quadratic nonlinearity. In this chapter, we summarize 1) discussions 2) conclusions 3) recommendations for future work.

## 7.1 SUMMARY OF DISCUSSION

A new computational linearization method, time-centered split (TCS) method, is derived from the midpoint rule temporal discretization. The fundamental principle of the TCS method is that it splits the quadratic nonlinear term into two steps so that each step is computationally linear. The two-step equation system is $2^{nd}$-oder equivalent to the original midpoint rule discretization. The essential theoretical development of the TCS method is illustrated in Chapter 3. The derivations in Chapter 3 demonstrate one of the most important significances of the TCS method: it is a direct linearization method while in an implicit format. The conventional implicit nonlinear solutions including Newton method, Picard method and local linearization method require either 1) outer iteration ( Newton and Picard methods) or 2) calculations of the Jacobian (Newton and local linearization methods). Therefore, compared to the conventional implicit nonlinear solutions, the TCS method has the advantage in efficiency and stability.

In a time marching differential equation, the time-centered split concept can be applied to different terms: either part of the quadratic nonlinear term and the linear term. Different TCS discretizations can be generated when we apply the split to different terms. Based on this characteristic of the TCS method, a general form of the TCS family method is created in Chapter 5. A weighting factor $\theta_i$ $(0 \le \theta_i \le 1)$ is introduced in the TCS general form. The weighting factor $\theta_i$ combines different approximations (splitting on different terms) of the nonlinear terms and the linear terms. The value of $\theta_i$ will not affect the $2^{nd}$-order temporal accuracy of the TCS method. The theoretical proof is shown in Chapter 5. This revealed another important significance of the TCS method: unlimited TCS discretized forms can be created for one problem. Thus, the TCS method provides

flexibility when we solve a specific problem because we can choose the discretizations based on the specific requirement of the problem.

The significance of the weighting factors is further exemplified when the TCS method is applied to a multi-variable and multi-dimension equation system. Properly chosen weighting factors can not only linearize but also decouple the equation system without outer iterations. This property theoretically enhances the efficiency advantage of the TCS method. The 2D Burgers' equation is used as the example to show the process and principle of the decoupling in Chapter 5. The decoupling characteristics of the TCS method is analyzed in detail using 2D depth averaged shallow water equations (SWE) in Chapter 6. Both cases show that the TCS method can computationally linearize and decouple an equation system without outer iterations. Each variable in the TCS linearized and decoupled two-step system is solved in a sequential order.

To verify and explore the basic characteristics of the TCS method, a 1D problem is first chosen as the test case. In Chapter 4, three 1D differential equations: 1D conservative Burgers' equation, 1D non-conservative Burgers' equation and 1D nonlinear ordinary differential equation (ODE) are solved using the TCS method. In all the three equations, we apply the time-centered split to different terms: either flux or gradient part in the nonlinear advection term in the Burgers' equation, the quadratic nonlinear term of the ODE, the diffusion term in the Burgers' equation and the linear term in the ODE. Consequently, multiple TCS disretizations are created such as TCSF (split the flux term), TCSG (split the gradient term), TCSF-D (split flux and diffusion/linear term) and TCSG-D (split the gradient and the diffusion/linear term). To compare to the conventional implicit linearization method, all three equations are also solved using the implicit Crank-Nicolson scheme combining with the conventional computational linearization methods: Newton method, Picard method and local linearization method. In the two Burgers' equation cases, each TCS discretization demonstrates $2^{nd}$-order temporal accuracy and remains stable up to a CFL $O$ (10). The stability advantage of the TCS method is further demonstrated by comparing it to the Runge-Kutta (RK) method. The efficiency of the TCS method is examined using operation count at each grid point at each time step. Results in the two 1D Burgers' equations show that the TCS method requires fewer operations than Newton and Picard methods but has a similar computation expense to the local linearization method. The principal advantage of the TCS method over local linearization is the relative ease with which the TCS method can be derived and implemented as it does not require discrete evaluation of a function Jacobian. The test case in 1D nonlinear ODE verified that the TCS method can be applied to any quadratic nonlinearity.

In the 1D ODE test case, two different TCS discretizations collapse into one expression, which is also mathematically equivalent to the local linearization. However, in both conservative and non-conservative Burgers' equation cases, different TCS disretizations have different relative errors although they perform similarly in temporal accuracy, stability and efficiency. This difference can also be observed in Chapter 5. Because of the introduction of the weighting factors, more TCS discretizations are tested for the 1D non-conservative Burgers' equation in Chapter 5. Our results show that changing $\theta_i$ will not affect the overall $2^{nd}$-order temporal accuracy of all the TCS

discretizations. However, the relative accuracy of each TCS discretization is different. The results show that the most accurate solution occurs when the two weighting factors equal to some value between 0 and 1 but not 0 or 1. The similar results can also be observed for the stability. The stability is also enhanced when the weighting factors are not equal to 0 or1. This can be explained by mathematical meaning of the weighting factors. The weighting factors control how we approximate each term in the equation. When the weighting factors equal to 0 or1, we only use one kind of approximation: either approximate $u^{n+1/2}$ using the time-centered split completely or approximate $u^{n+1/2}$ using $u^{*}$ completely. Our results suggest that combing both approximations will enhance the performance of the TCS method in both accuracy and stability. However, the best combinations of the weighting factors for accuracy and stability is problem-specific and one needs to analyze each approximated term and the nature of the solution itself at the designated marching time. This phenomenon verified that TCS method provides flexibility in solving a specific problem because a more accurate or stable solution can be obtained by changing the choice of the weighting factors.

In Chapter 6, the TCS method is used to solve 2D depth averaged SWE. We discover that six terms in the SWE couple the entire equation system. These six terms are $V\partial U/\partial y$, $U\partial V/\partial x$, $\partial HU/\partial x$, $\partial HV/\partial y$, $\partial\zeta/\partial x$ and $\partial\zeta/\partial y$. Their associated weighting factors are crucial in the decoupling process. It is revealed that only six combinations of these weighting factors can fully decouple the SWE. Each combination of the weighting fact has a unique solution order. Three representative TCS solutions are chosen to solve the SWE based on the solution order of the continuity equation. In the one-dimensional standing wave case, how the TCS method treat the nonlinear term is tested. The results show that that the TCS methods well capture the nonlinear effects and display the combination effects of numerical dispersion and hydrostatic approximation. How the solution order affects the results is analyzed in the two-dimensional standing wave case. The results show that TCS solution 1 perform differently than TCS solutions 2 and 3 because the surface elevation is obtained using the velocity components from different time step in TCS solution 1. The fully-decoupled TCS-discretized SWE is solved in a sequential order, which causes each equation solved anisotropically. This in turn causes the numerical cross modes. The cross mode from TCS solution 2 and 3 behave similarly: the cross mode gradually increases with time. However, the solution procedure in TCS solution 1 causes wave oscillate in the cross direction with the same amplitude as the initial standing wave and half phase behind the initial standing wave. Hence, we need to take into account the numerical errors generated by the decoupling process when we apply the TCS method to decouple an equation system.

## 7.2 CONCLUSIONS

The conclusions that may be drawn from the present work are:

- A new time marching numerical algorithm for solving unsteady nonlinear advection problems is proposed using a time-centered split (TCS) technique.
- The principle advantage of the TCS method is it computationally linearizes the implicit nonlinear advection without either 1) iterations or 2) calculations of the Jacobian.
- The TCS method is $2^{nd}$-order temporal accurate and has advantages in stability and efficiency.
- A family of the TCS discretizations can be generated using the same principle. This provides flexibility when solving a specific problem using the TCS method.
- The TCS method can fully decouple an equation system. However, additional numerical error is introduced through the decoupling process.

## 7.3 RECOMMENDATIONS FOR FUTURE WORK

The TCS method is proposed and verified using 1D transport equation, 1D nonlinear ODE, 2D transport equation. The following potential application is recommended for future work.

*Application of the TCS method to a coupled nonlinear ODE system*

This research is motivated by solving unsteady nonlinear advection problems. 1D and 2D flow transport problems are tested using the TCS method. However, the new TCS method provides a general approach to solve any quadratic nonlinearity. Although a 1D nonlinear ODE is used as an example in Chapter 3, application of the TCS method to the nonlinear ODE can be further investigated. A coupled quadratic nonlinear ODE system can be used as an example to verify the capability of the TCS method in computational linearization and decoupling. Moreover, various TCS discretizations can be compared and tested by changing the weighting factors.

*More test cases for the SWE*

In Chapter 6, initial tests of 2D depth averaged SWE have been conducted. All the test cases in 2D standing wave are in an inviscid flow condition. Since the weighting factors associated with the viscous terms will not affect the decoupling process, the inviscid flow condition is a valid test condition for investigating decoupling process. However, the weighting factor associated with the non-coupled advection terms such as $U\partial U/\partial x$ and $V\partial V/\partial y$ can be any value from 0 to 1. In Chapter 6, we didn't change the weighting factors for these two terms to see how these two will affect the solution. Based on the analysis in Chapter 5, the numerical error induced by the decoupling might be decreased by choosing different weighting factors for these two terms. An additional test for numerical error could be conducted by decreasing the time interval $\Delta t$.

*Solve problems with discontinued flow conditions*

We often need to solve a transport problem with a discontinuity flow condition, for example, hydraulic jump. All the test cases conducted in this research use central difference spatial discretizations, which cannot represent the sharp front of a shock wave. Test cases can be designed by the TCS temporal discretization combining upwind spatial discretization to simulate a shock wave.

# Appendix A Test of a Progressive Wave in an Open Boundary System

Another way to verify our numerical method is to simulate a wave progressing in an open boundary system. This test case simulates a free surface oscillation in a rectangular channel, which is open at both upstream (inlet) and downstream (outlet boundary). This also could be thought of as a step leading to the simulation of a river flow. In this section, a progressive wave is introduced by oscillating the inlet boundary of an open quiescent water body. A sponge layer is successfully applied as the open boundary condition at the outlet of the domain. Our numerical models satisfactorily simulate a progressive wave traveling through an open boundary system. As shown in Chapter 6, TCS solutions 2 and 3 give similar results. Thus, in this test, only results from TCS solution 1 and 2 are presented and compared.

## A.1 Initial and Inlet Boundary Condition

Initially, the water is at rest with uniform depth:

$$H(x,0) = h_0 \tag{A.1}$$

The inlet water level suddenly raised at t=0 and oscillated as a cosine wave described as

$$H(0,t) = h_0 + a\cos(\sigma t) \tag{A.2}$$

where a is the progressive wave amplitude and $\sigma$ is the frequency. The initial condition and inlet boundary condition is shown in Figure A.1 and A.2.


Figure A.1 Initial water level, H (x, 0), in the rectangular open channel

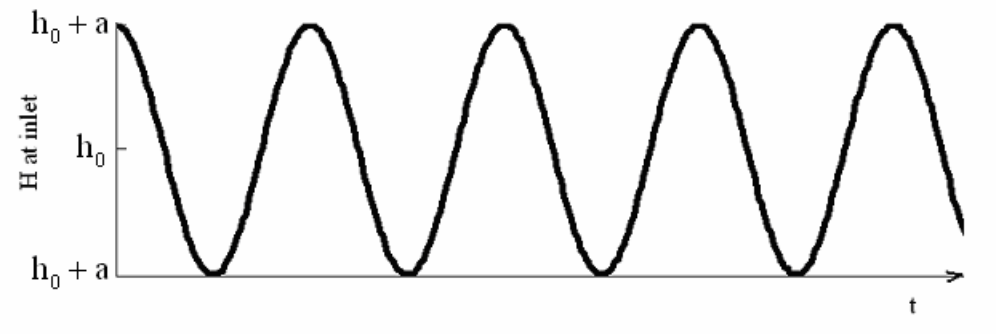


Figure A.2 Inlet boundary condition H (0, t)

## A.2 OUTLET BOUNDARY CONDITION

One of the challenges to simulate an open boundary system with a finite computational domain is that open boundary conditions are required at both the inlet and outlet. Developing outlet non-reflective boundary conditions has been the subject of extensive research (Blayo and Debreu 2005; Tsynkov 1998).

In the present work we apply an artificial damping layer (or sponge layer) upstream of the outlet boundary and downstream of the "test section" (i.e. the computational domain of interest). In this approach, an artificial damping function is prescribed over a range of grid cells to dissipate the surface wave and its reflections before it propagates back into the test section (Durran 1999). Although a sponge layer has additional computational costs associated with computations outside of the test section, the ratio of the additional cost to the initial cost is generally small (Blayo and Debreu 2005). In the present work, the sponge layer damping function uses an increasing viscosity from the end of the test section to the outlet (Vinayan 2003). We use an inviscid progressive wave for the test case, so the viscosity is set to be zero within test section. In general, the viscosity is represented as a function of position such that $0 \le \nu(x) \le \nu_{max}$.

If we define the viscosity is a function of x,

$$\nu(x) = \nu_{max} f(x) \tag{A.3}$$

where, $f(x)$ is a function satisfying the following conditions:

$$\begin{aligned} f(x_r) &= 0 \\ f(x_s) &= 1 \\ \frac{df(x_r)}{dx} &= 0 \\ \frac{df(x_s)}{dx} &= 0 \end{aligned} \tag{A.4}$$

where $x_r$ and $x_s$ are the upstream and downstream limits of the sponge layer, as shown in Figure A.3. If the artificial viscosity changes rapidly, spurious reflections may also occur. A gradual change, as shown in Figure A.3, can be invoked by defining $f(x)$ as

$$f(x) = \frac{1}{2}\left[1 - \cos\left(\pi \frac{x - x_r}{x_s - x_r}\right)\right] \tag{A.5}$$

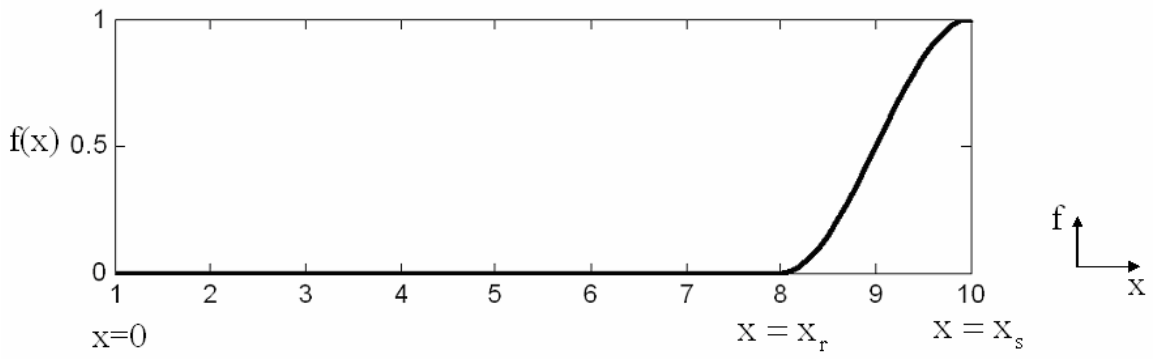


Figure A.3 Shape function of viscosity

Using the above definitions, a sponge layer is defined by its length ($x_s - x_r$) and its maximum viscosity $\nu_{max}$. In this work, $x_s - x_r = 2\lambda$ and $\nu_{max}$ is basically a relative big number obtained by trial and error. The sponge layer arrangement is shown in Figure A.4.

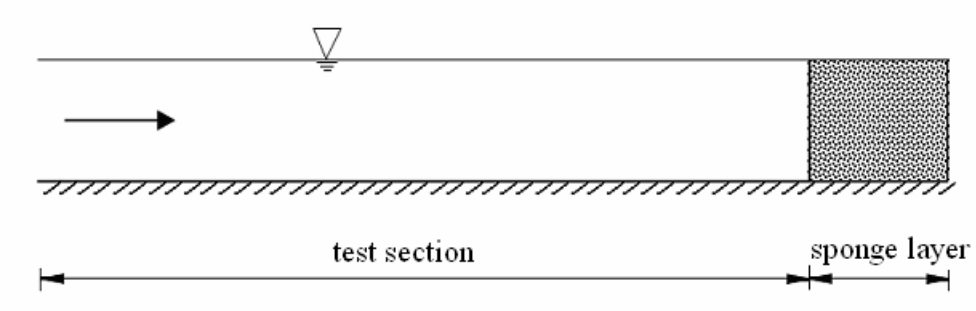


Figure A.4 An open boundary rectangular channel with a sponge layer

These initial and boundary conditions generate a traveling wave with height 2a, progressing along the channel. To see the damping effects of the sponge layer, we monitored the surface height of point A, which is the point in the test section closest to

the sponge layer, and B, which is the point within the sponge layer closest to the outlet, as shown in Figure A.5.

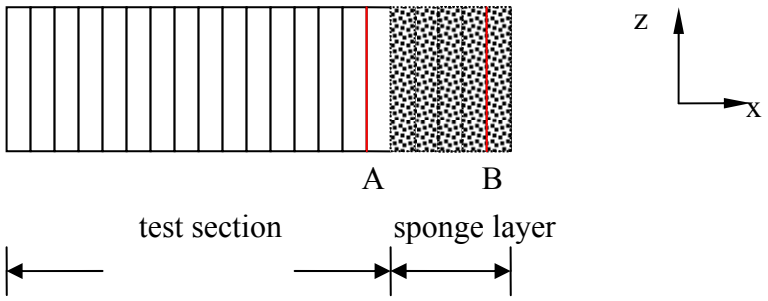


Figure A.5 Schematic illustrations of grids

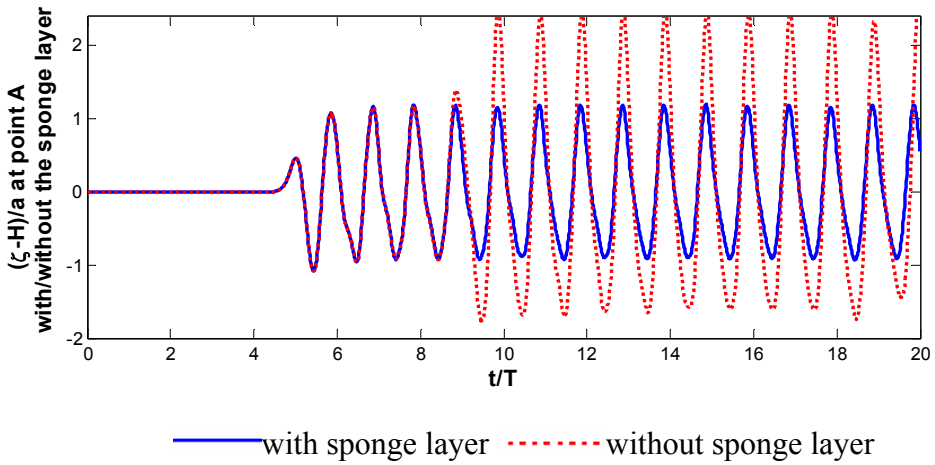


Figure A.6 Water level at point A with/without the sponge layer

Figure A.6 shows the evolution of the nondimensionalized water depth ($(\zeta - H)/a$) at point A with and without a sponge layer. Without the sponge layer, when the wave reaches the solid wall where zero flux is prescribed, reflection occurs and its amplitude is increased to about two times of the original one. However, with a sponge layer condition, the wave amplitude stays the same. We also compare the wave at point A and B in Figure A.7. The wave inside the sponge layer is damped about 10% of the wave in the test section. Thus the reflection effect is minimized.

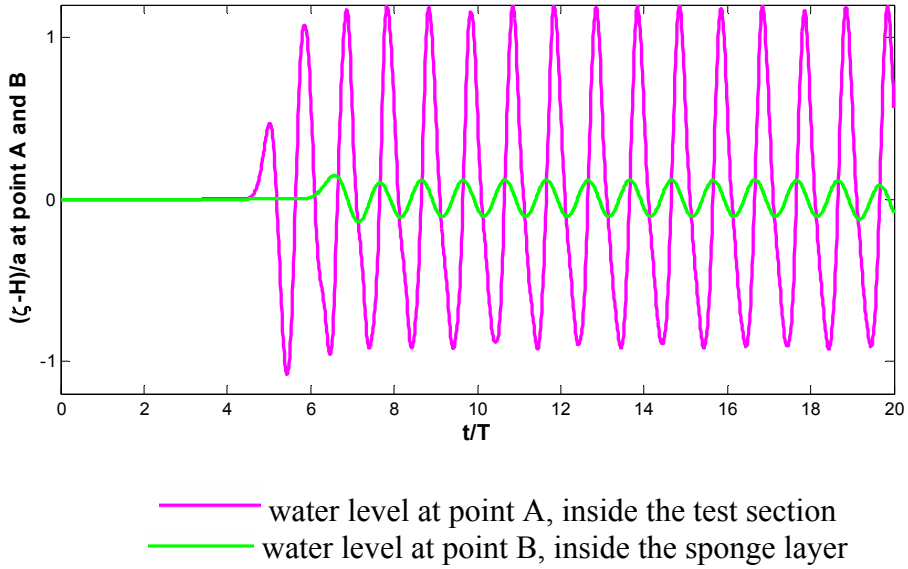


Figure A.7 Water level evolutions inside the test section and sponge layer.

### A.3 RESULTS AND DISCUSSION

Figure A.8 compares the free surface shape in the entire domain (including test section and sponge layer) at 7T and 14T for TCS solution 1 and 2 respectively. At 7T, the progressive wave has traveled through the domain. When we double the simulation time to 14T, the free surface shape remains almost the same. This indicates that we successfully simulated a wave progress through an open boundary system, and our numerical results model the wave characteristics such as wave speed and period well. Both TCS solutions 1 and 2 give us similar results.

To further compare the two TCS solutions in this progressive wave test case, we compare the free surface shape calculated using both TCS discretizations in the entire domain at 7T and 14 T, respectively, in Figure A.9. The results simulated using each TCS discretization overlapped at 7T. After a longer simulation time, this similarity between two solutions still holds true at 14T.

The time evolution of free surface at different periods in the entire domain for TCS solutions 1 and 2 are shown in Figure A.10 and Figure A.11 respectively.

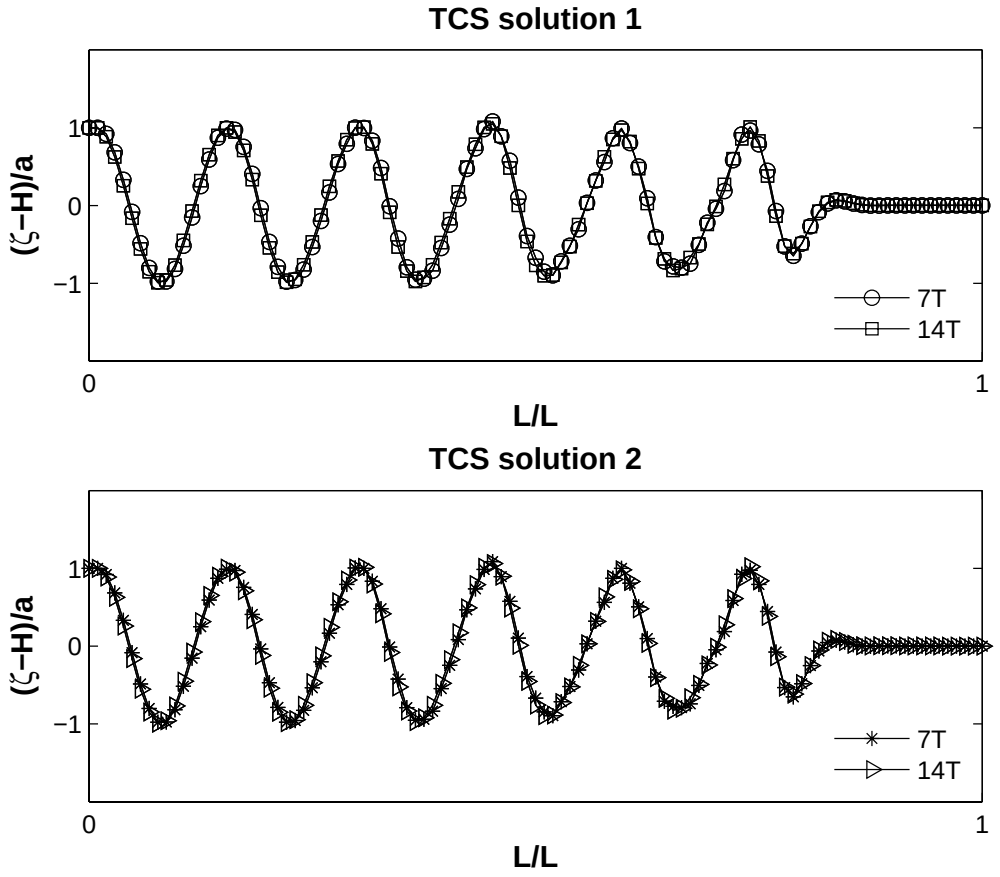


Figure A.8 Comparison of wave shape at time 7T and 14T for TCS solution1 and TCS solution 2 respectively

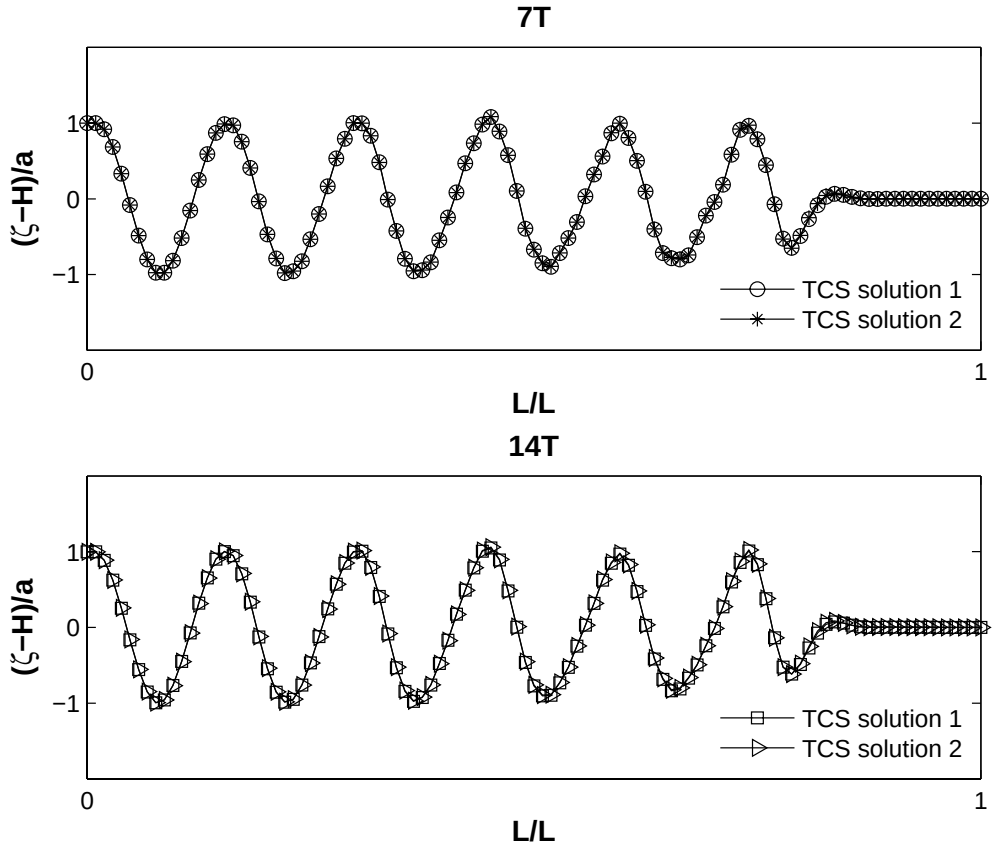


Figure A.9 Comparison of wave shape simulated from TCS solution 1 and 2 at time 7T and 14T respectively

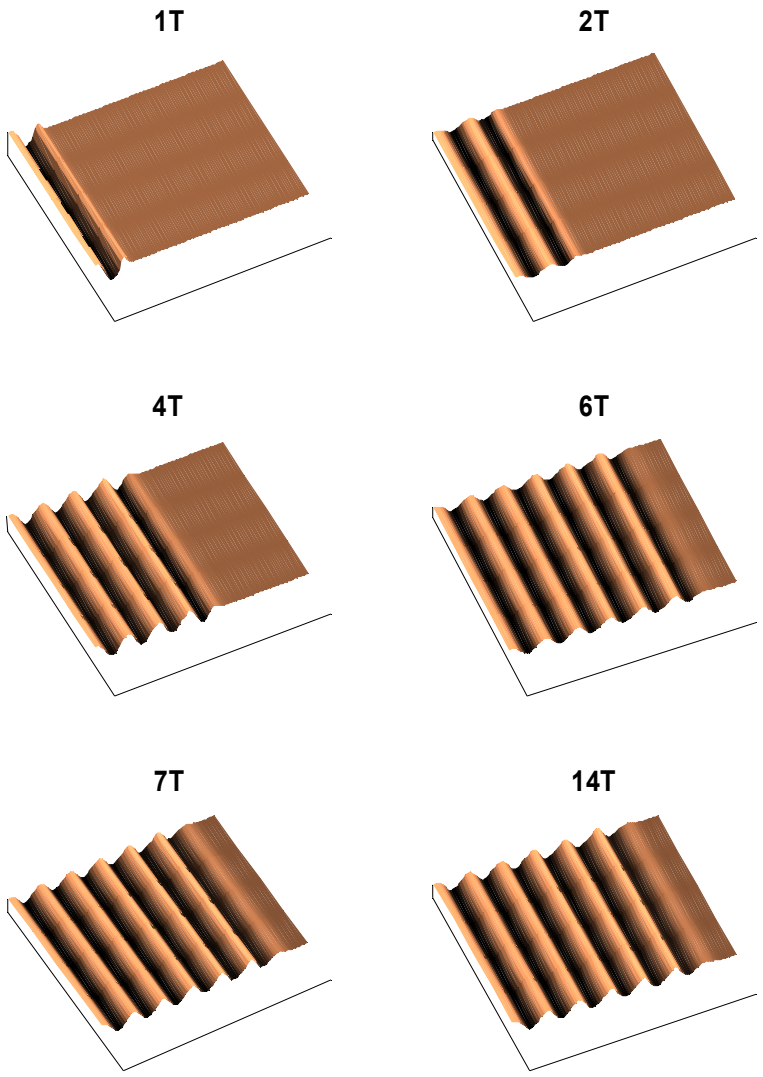


Figure A.10 Time evolution of a progressive wave at different wave periods from TCS solution 1.

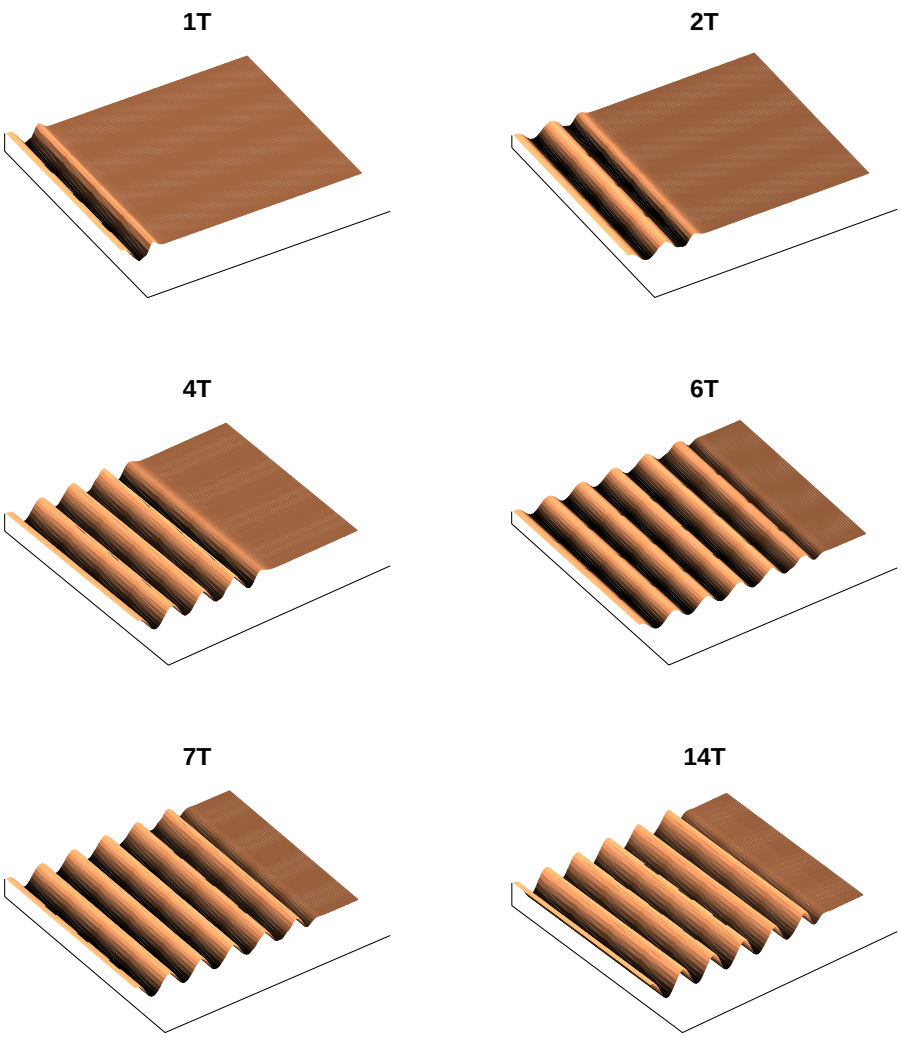


Figure A.11 Time evolution of a progressive wave at different wave periods from TCS solution 2.

# References


Agoshkov, V., Ovchinnikov, E., Quarteroni, A., and Saleri, F. (1994). "Recent Developments in the Numerical Simulation of Shallow Water Equations. II. Temporal Discretization." *Mathematical Models and Methods in Applied Sciences*, 4(4), 533-566.

Amat, S., Busquier, S., and Gutierrez, J. M. (2003). "Geometric constructions of iterative functions to solve nonlinear equations." *Journal of Computational and Applied Mathematics*, 157(1), 197-205.

Anderson, D. A., Tannehill, J. C. and Pletcher, R. H. (1997). Computational Fluid Mechanics and Heat Transfer. 2nd ed. Hemisphere Publishing, MacGraw-Hill.

Andrus, C. W., Long, B. A., and Froehlich. H. A. (1988). "Woody debris and its contribution to pool formation in a coastal stream 50 years after logging." *Can. J. of Fisheries and Aquatic Sci.*, 45, 2080-2086.

Ancey, C., Iverson, R. M., Rentschler, M., and Denlinger, R. P. (2008). "An exact solution for ideal dam-break floods on steep slopes." *Water Resources Research*, 44(1).

Arakawa, A., and Lamb, V.R. (1977). "Computational design of the basic dynamical processes of the UCLA general circulation model." *Methods in Computational Physics*, 17, 173-265.

Arega, F., Lee, J. H. W., and Tang, H. W. (2008). "Hydraulic jet control for river junction design of Yuen Long Bypass Floodway, Hong Kong." *Journal of Hydraulic Engineering-Asce*, 134(1), 23-33.

Bagatur, T. (2007). "Modified Newton-Raphson solution for dispersion equation of transition water waves." *Journal of Coastal Research*, 23(6), 1588-1592.

Bajo, M., Zampato, L., Umgiesser, G., Cucco, A., and Canestrelli, P. (2007). "A finite element operational model for storm surge predicition in Venice." *Estuarine Coastal and Shelf Science*, 75(1-2), 236-249.

Baldwin, B.S., Lomax H. (1978). "Thin-layer approximation and algebraic model for separated turbulent flows." *AISS paper*, No. 78-257.

Baldwin, B.S., Barth T.J. (1990). "A one-equation turbulence transport model for high Reynolds number wall-bounded flows." NASA TM-102847.

Barros, S. R. M., and Garcia, C. I. (2007). "A global finite-difference semi-Lagrangian model for the adiabatic primitive equations." *Journal of Computational Physics*, 226(2), 1645-1667.

Bisson, P.A., et al. (1987). Large woody debris in forested streams in the Pacific Northwest, past, present and future; Contribution No.57. in Streamside management, forestry and fishery interactions. E. O. Salo and T. W. Cundy. Eds., Coll. Of Forest Resour, Univ. of Washington, Seattle, Wash., 143-190.

Blayo, E. and Debreu, L. (2005). "Revisting open boundary conditions from the point of view of characteristic variables." *Ocean Modeling*, 9, pp. 231-252.

Blumberg, A. F., and Mellor, G. L. (1987). *A description of a three-dimensional coastal ocean circulation model, In Three-Dimensional Coastal Ocean Models*, American Geophysical Union, Washington, DC.

Bonaventura, L., and Rosatti, G. (2002). "A cascadic conjugate gradient algorithm for mass conservative, semi-implicit discretization of the shallow water equations on

locally refined structured grids." *International Journal for Numerical Methods in Fluids*, 40(1-2), 217-230.

Bourchtein, A., and Bourchtein, L. (2006). "Modified Time Splitting Scheme for Shallow Water Equations." *Mathematics and Computers in Simulation*, 73, 52-64.

Bourchtein, A., and Bourchtein, L. (2007). "Semi-Lagrangian semi-implicit time-splitting scheme for the shallow water equations." *International Journal for Numerical Methods in Fluids*, 54(4), 453-471.

Brown, P. N., and Saad, Y. (1990). "Hybrid Krylov methods for nonlinear systems of equations." *SIAM J. Sci. Stat. Comput.*, 11, 450-481.

Burguete, J., and Garcia-Navarro, P. (2004). "Implicit schemes with large time step for non-linear equations: application to river flow hydraulics." *International Journal for Numerical Methods in Fluids*, 46(6), 607-636.

Burguete, J., Garcia-Navarro, P., and Murillo, J. (2006). "Numerical boundary conditions for globally mass conservative methods to solve the shallow-water equations and applied to river flow." *International Journal for Numerical Methods in Fluids*, 51(6), 585-615.

Cao, J. W., and Sun, J. C. (2005). "An efficient and effective nonlinear solver in a parallel software for large scale petroleum reservoir simulation." *International Journal of Numerical Analysis and Modeling*, 2, 15-27.

Cao, Z., and Carling, P. A. (2002). "Mathematical modelling of alluvial rivers: reality and myth. Part 2: Special issues." *Proceedings of the Institution of Civil Engineers-Water and Maritime Engineering*, 154(4), 297-307.

Casulli, V. (1990). "Semi-implicit Finite Difference Methods for the Two-Dimensional Shallow Water Equations." *Journal of Computational Physics*, 86, 56-74.

Casulli, V., and Cattani, E. (1994). "Stability, Accuracy and Efficiency of a Semiimplicit Method for 3-Dimensional Shallow-Water Flow." *Computers & Mathematics with Applications*, 27(4), 99-112.

Casulli, V., and Cheng, R. T. (1992). "Semiimplicit Finite-Difference Methods for 3-Dimensional Shallow-Water Flow." *International Journal for Numerical Methods in Fluids*, 15(6), 629-648.

Celia, M. A., Ahuja, L. R., and Pinder, G. F. (1987). "Orthogonal collocation and alternating-direction procedures for unsaturated flow problems " *Adv. Water Resour.*, 10, 178-187.

Chan, T. F., and Jackson, K. R. (1984). "Nonlinearly preconditioned Krylov subspace methods for discrete Newton algorithms." *SIAM J. Sci. Stat. Comput.*, 5, 533-542.

Chen, S. C., and Peng, S. H. (2006). "Two-dimensional numerical model of two-layer shallow water equations for confluence simulation." *Advances in Water Resources*, 29(11), 1608-1617.

Cho, Y.-S., and Yoon, S. B. (1998). "A Modified Leap-Frog Scheme for Linear Shallow-Water Equations." *Costal Engineering Journal, JSCE*, 40(2), 191-205.

Clement, T. P., Wise, W. R., and Molz, F. J. (1994). "A Physically-Based, 2-Dimensional, Finite-Difference Algorithm for Modeling Variably Saturated Flow." *Journal of Hydrology*, 161(1-4), 71-90.

Cullen, M. J. P. (2001). "Alternative implementations of the semi-Lagrangian semi-implicit schemes in the ECMWF model." *Quarterly Journal of the Royal Meteorological Society*, 127(578), 2787-2802.

Daily, C., and Imberger, J. (2003). "Modelling solitons under the hydrostatic and Boussinesq approximations." *International Journal for Numerical Methods in Fluids*, 43(3), 231-252.

Davis, J. A. (1986). Boundary Layers, Flow Microenvironments and Stream Benthos. Limnology in Australia (eds De Deck and Williams), pp. 293-312. CSIRO Australia, Melbourne. Dr. W. Junk Publishers, Doedercht.

Dean, R. G. and R. A. Dalrymple (1998). Water Wave Mechanics for Engineering and Scientists. World Scientific Publishing Co. Pte. Ltd., New Jersey.

Dedong, L., Zhongbo, Y., Zhenchun, H., Chuanguo, Y., and Qin, J. (2007). "Groundwater simulation in the Yangtze River basin with a coupled climate-hydrologic model." *Journal of China University of Geosciences*, 18, 155-157.

Delis, A. I., and Katsaounis, T. (2005). "Numerical solution of the two-dimensional shallow water equations by the application of relaxation methods." *Applied Mathematical Modelling*, 29(8), 754-783.

Dettmer, W. G., and Peric, D. (2007). "A fully implicit computational strategy for strongly coupled fluid-solid interaction." *Archives of Computational Methods in Engineering*, 14(3), 205-247.

Dubois, T., Jauberteau, F., Temam, R. M., and Tribbia, J. (2005). "Multilevel schemes for the shallow water equations." *Journal of Computational Physics*, 207, 660-694.

Durran, D. R. (1999). Numerical Methods for Wave Equations in Geophysical Fluid Dynamics. Springer-Verlag, New York.

Dutta, D., Alam, J., Umeda, K., Hayashi, M., and Hironaka, S. (2007). "A two-dimensional hydrodynamic model for flood inundation simulation: a case study in the lower Mekong river basin." *Hydrological Processes*, 21(9), 1223-1237.

Fennema, R. J., and Chaudhry, M. H. (1990). "Explicit Methods for 2-D Transient Free-Surface Flows." *Journal of Hydraulic Engineering*, 116(8), 1013-1035.

Ferziger J.H. (1996). Simulation and Modeling of Turbulent Flows: Large eddy simulation. M.Y. Hussaini, T. Gatski (eds.), Cambridge Univ. Press, New York. 1996

Ferziger J. H., Koseff, J.R., Monismith, S.G. (2002). "Numerical simulation of geophysical turbulence." *Computers & Fluids*, 31, 557-568.

Ferziger, J. H., and Peric, M. (1999). *Computational Methods for Fluid Dynamics*, Springer, New York.Fleckenstein, J. H., Niswonger, R. G., and Fogg, G. E. (2006). "River-aquifer interactions, geologic heterogeneity, and low-flow management." *Ground Water*, 44(6), 837-852.

Fu, S. and Hodges, B. R. (2005). "Grid-scale dependency of subgrid-scale structure effects in hydraulic models of rivers and streams." Proc., 2005 Mechanics and Materials Conference (McMat 2005), Louisiana State University, Baton Rouge.

Fujima, K. and Shigemura, T (2000). "Determination of Grid Size for Leap-Frog Finite Difference Model to Simulate Tsunamis around A Conical Island." *Costal Engineering***,** 42(2), 197-210.

Garcia, R., and Kahawita, R. A. (1986). "Numerical Solution of the St. Venant Equations with the MacCormack Finite-Difference Scheme." *International Journal for Numerical Methods in Fluids*, 6, 259-274.

Gejadze, I. Y., and Monnier, J. (2007). "On a 2D 'zoom' for the 1D shallow water model: Coupling and data assimilation." *Computer Methods in Applied Mechanics and Engineering*, 196(45-48), 4628-4643.

Gippel, C. J.(1995). "Environmental hydraulics of large woody debris in streams and rivers." *J. Environmental Engineering*, 121, 388-395.

Giri, S., and Shimizu, Y. (2006). "Numerical computation of sand dune migration with free surface flow." *Water Resources Research*, 42(10).

Grötzner, A. et al. (1996). "The Impact of Sub-grid Scale Sea-ice Inhomogeneities on the Performance of the Atmospheric General Circulation Model ECHAM3." *Climate Dynamics*,12,477-496.

Grossman, G.D., Rincon, P. A., Farr, M.D., and Ratajczak, R. E. (2002). "A new optimal foraging model predicts habitat use by drift-feeding stream minnows." *Ecol. Freshwat. Fish*, 11,2-10.

Hamrick, J. M. (1992). "A three-dimensional environmental fluid dynamics computer code: theoretical and computarional aspects." Virginia Institute of Marine Science, School of Marine Science, The College of William and Mary, Special Report 317., Gloucester Point, VA 23062.

Harmon, M. E., et al. (1986). "Ecology of coarse woody debris in temperature ecosystems." *Adv. in Ecological Res*, 15, 133-302.

Hirose, N., Asai, K., Ikawa, K., and Kawamura, R. (1991). "Euler Flow-Analysis of Turbine Powered Simulator and Fanjet Engine." *Journal of Propulsion and Power*, 7(6), 1015-1022.

Hodges, B. R. (1997). "Numerical simulation of nonlinear free-surface waves on a turbulent open-channel flow," Ph.D. dissertation, Department of Civil Engineering, Stanford University.

Hodges, B. R. (2000). "Numerical Techniques in CWR-ELCOM (code release v.1)." Centre Water Res., Nedlands, Western Australia, Australia.

Hodges, B. R., B. Laval, Wadzuk, B. (2006). "Numerical error assessment and temporal horizon for internal waves in a hydrostatic model." *Ocean Modeling*, 13(1), 44-64.

Hunter, N. M., Bates, P. D., Neelz, S., Pender, G., Villanueva, I., Wright, N. G., Liang, D., Falconer, R. A., Lin, B., Waller, S., Crossley, A. J., and Mason, D. C. (2008). "Benchmarking 2D hydraulic models for urban flooding." *Proceedings of the Institution of Civil Engineers-Water Management*, 161(1), 13-30.

Iskandarani, M., Levin, J. C., Choi, B. J., and Haidvogel, D. B. (2005). "Comparison of advection schemes for high-order h-p finite element and finite volume methods." *Ocean Modelling*, 10(1-2), 233-252.

Kadalbajoo, M. K., and Awasthi, A. (2006). "A numerical method based on Crank-Nicolson Scheme for Burgers' Equation." *Applied Mathematics and Computation*, 182, 1430-1442.

Kar, S. K. (2006). "A Semi-Implicit Runge-Kutta Time-Difference Scheme for the Two-Dimensional Shallow Water Equations." *Monthly Weather Review*, 134(October), 2916-2926.

Kazezyılmaz-Alhana, C. M., Medina, M. A., and Raob, P. (2005). "On numerical modeling of overland flow." *Applied Mathematics and Computation* 166(3), 724-740.

Kutluay, S., Bahadir, A. R., and Ozdes, A. (1999). "Numerical solution of one-dimensional Burgers equation: explicit and exact-explicit finite difference methods." *Journal of Computational and Applied Mathematics*, 103, 251-261.

Keller, E.A., and Swanson, F.J. (1979). "Effect of large organic material on channel form and fluvial processes." Earth Surface Processes, 4, 361-380.

Keshari, A. K., and Koo, M. H. (2007). "A numerical model for estimating groundwater flux from subsurface temperature profiles." *Hydrological Processes*, 21(25), 3440-3448.

Knoll, D. A., and Keyes, D. E. (2004). "Jacobian-free Newton-Krylov methods: a survey of approaches and applications." *Journal of Computational Physics*, 193(2), 357-397.

Kolmogorov, AN. (1942). "The equations of turbulent motion in an incompressible fluid." *Isv. Acad. Sci., USSR, Phys*. 6, 56-58.

Kurbatskii, K. A., and Mankbadi, R. R. (2004). "Review of computational aeroacoustics algorithms." *International Journal of Computational Fluid Dynamics*, 18(6), 533-546.

Kwag, S. H. (2000). "Computation of water and air flow with submerged hydrofoil by interface capturing method." *Ksme International Journal*, 14(7), 789-795.

Kwok, F., and Tchelepi, H. (2007). "Potential-based reduced Newton algorithm for nonlinear multiphase flow in porous media." *Journal of Computational Physics*, 227(1), 706-727.

Lomax, H., Pulliam, T. H., and Zingg, D. W. (1999). *Fundamentals of Computational Fluid Dynamics*, Springer, New York.

Le, V. S., Yamashita, T., Okunishi, T., Shinohara, R., and Miyatake, M. (2006). "Characteristics of suspended sediment material transport in the Ishikari Bay in snowmelt season." *Applied Ocean Research*, 28(4), 275-289.

Lehmann, F., and Ackerer, P. (1998). "Comparison of Iterative Methods for Improved Solutions of the Fluid Flow Equation in Partially Saturated Porous Media." *Transport in Porous Media*, 31, 275-292.

Li, C. W. (1993). "A simplified Newton iteration method with linear finite elements for transient unsaturated flow." *Water Resour. Res.*, 29(4), 965-971.

Li, G. Y., and Jackson, C. R. (2007). "Simple, accurate, and efficient revisions to MacCormack and Saulyev schemes: High Peclet numbers." *Applied Mathematics and Computation*, 186(1), 610-622.

Liao, C. B., Wu, M. S., and Liang, S. J. (2007). "Numerical simulation of a dam break for an actual river terrain environment." *Hydrological Processes*, 21(4), 447-460.

Lomax, H., Pulliam, T. H. and Zingg, D.W. (1996) Fundamentals of Computational Fluid Dynamics. Springer, New York Inc.

Luo, Q. (2007). "A distributed surface flow model for watersheds with large water bodies and channel loops." *Journal of Hydrology*, 337(1-2), 172-186.

MacCormack, R. W. (1969). "The effect of viscosity in hypervelocity impact cratering." *AIAA-1969-354*.

Marzolf, G. R. (1978). The potential effects of clearing and snagging on stream ecosystems. FWS/OBS-78-14, U.S. Dept of Interior. Fish and Wildlife Service, Nat. Stream Alteration Team, Washington, D.C.

McMahon, T. E., and Hartman, G. F. (1989). "Influence of cover complexity and current velocity on winter habitat use by juvenile Coho Salmon (Oncorhynchus kisutch)." *Can. J. of Fisheries and Aquatic Sci.*, 46, 1551-1557.

Mendez-Nunez, L. R., and Carroll, J. J. (1993). "Comparison of Leapfrog, Smolarkiewicz, and MacCormack Schemes Applied to Nonlinear Equations." *Mon. Weather Rev.*, 121(February), 565-578.

Miropol'sky, Y. Z. (2001). *Dynamics of internal gravity waves in the ocean*, Kluwer.

Mohebalhojeh, A. R., and Dritschel, D. G. (2007). "Assessing the numerical accuracy of complex spherical shallow-water flows." *Monthly Weather Review*, 135(11), 3876-3894.

Moin, P. (2001). Fundamentals of Engineering Numerical Analysis. Cambridge, U.K., Cambridge University Press.

Mousseau, V. A., Knoll, D. A., and Reisner, J. M. (2002). "An implicit nonlinearly consistent method for the two-dimensional shallow-water equations with coriolis force." *Monthly Weather Review*, 130(11), 2611-2625.

Murillo, J., Garcia-Navarro, P., and Burguete, J. (2008). "Analysis of a second-order upwind method for the simulation of solute transport in 2D shallow water flow." *International Journal for Numerical Methods in Fluids*, 56(6), 661-686.

Nguyen, D. K., Shi, Y. E., Wang, S. S. Y., and Nguyen, H. (2006). "2D shallow-water model using unstructured finite-volumes methods." *Journal of Hydraulic Engineering-Asce*, 132(3), 258-269.

Niet, A. d., Wubs, F., Scheltinga, A. T. v., and Dijkstra, H. A. (2007). "A tailored solver for bifurcation analysis of ocean-climate models." *Journal of Computational Physics*, 227 (1), 654-679.

Paniconi, C., Aldama, A. A., and Wood, E. F. (1991). "Numerical Evaluation of Iterative and Noniterative Methods for the Solution of the Nonlinear Richards Equation." *Water Resources Research*, 27(6), 1147-1163.

Paniconi, C., and Putti, M. (1994). "A Comparison of Picard and Newton Iteration in the Numerical Solution of Multidimensional Varibly Saturated Flow Problems." *Water Resources Research*, 30(12), 3357-3374.

Peyret, R., and Taylor, T. D. (1983). *Computational Methods for Fluid Flow*, Springer, New York.

Pozrikidis, C. (2005). *Introduction to finite and spectral element methods using MATLAB*, Chapman & Hall/CRC, Boca Raton.

Prandtl L. (1925). Uber die ausgebildete turbulenz. ZAMM 5, pp. 331-340.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1992). *Numerical recipes in FORTRAN : the art of scientific computing*, Cambridge University Press, New York.Peyret R. and Taylor, T. D. (1983). Computational Method s for Fluid Flow. Springer-Verlag, New York Inc.

Rabeni. C. F., and Jacobson, R. B. (1993). “The importance of fluvial hydraulics to fish-habitat restoration in low-gradient alluvial streams.” *Freshwater Biology*, 29, 211-220.

Reisner, J. M., Mousseau, V. A., Wyszogrodzki, A. A., and Knoll, D. A. (2005). "An implicitly balanced hurricane model with physics-based preconditioning." *Monthly Weather Review*, 133(4), 1003-1022.

Robert, L. W. (1992). *Oceanographical Enginereing*, Dover Publications, Inc., Mineola, New York.

Rosatti, G., Cesari, D., and Bonaventura, L. (2005). "Semi-implicit, semi-Lagrangian modelling for environmental problems on staggered Cartesian grids with cut cells." *Journal of Computational Physics*, 204(1), 353-377.

Rueda, F. J., and Schladow, G. S. (2002). "Quantitative Comparison of Models for Barotropic Response of Homogeneous Basins." *Journal of Hydraulic Engineering*, 128(2), 201-213.

Salvetti, MV, Banerjee, S. (1995). “A priori tests of a new dynamic subgrid-scale model for finite- difference large-eddy simulations.” *Phys. Fluids*, 7(11), 2831-2847.

Schoenstadt, A. (1980). "A transfer analysis of numerical schemes used to simulate geostrophic adjustment." *Mon. Weather Rev.*, 108, 1248-1295.

Shaw, D., Martz, L. W., and Pietroniro, A. (2005). "Flow routing in large-scale models using vector addition." *Journal of Hydrology*, 307(1-4), 38-47.

Shields, F.D. Jr., and Nunnally, N.R. (1984). “Environmental aspects of clearing and snagging.” *Journal of Environmental Engineering*, 110 (1), 152-165.

Spalart, P.R, Allmaras, S.R. (1992). “A one-equation turbulence model for aerodynamic flow.” *AIAA J*. 29, 1819-1835.

Spitaleri, R. M., and Corinaldesi, L. (1997). "A multigrid semi-implicit finite difference method for the two-dimensional shallow water equations." *International Journal for Numerical Methods in Fluids*, 25(11), 1229 - 1240.

Stelling, G. S., and Duinmeijer, S. P. A. (2003). "A staggered conservative scheme for every Froude number in rapidly varied shallow water flows." *International Journal for Numerical Methods in Fluids*, 43(12), 1329-1354.

Steppeler, J. (2006). "HLI, a direct method suitable for partial and fully implicit time integration of primitive equation meteorological models." *Computers & Mathematics with Applications*, 52(8-9), 1357-1372.

Sullivan, K., Lisle, T. E., Dolloff, C. A., Grant, G. E., and Reid, L. M. (1987). “Stream channels: the link between forest and fishes.” Streamside management, forestry and fishery interactions. E. O. Salo and T. W. Cundy. Eds., Coll. Of Forest Resour., Univ. of Washington, Seattle, Wash., 39-97.

Szymkiewicz, R. (1992). "A Mathematical-Model of Storm-Surge in the Vistula Lagoon, Poland." *Coastal Engineering*, 16(2), 181-203.

Tannehill, J. C., Anderson, D. A., and Pletcher, R. H. (1997). *Computational fluid mechanics and heat transfer*, Taylor & Francis, Washington, DC.

Thomee, V. (2001). "From finite differences to finite elements - A short history of numerical analysis of partial differential equations." *Journal of Computational and Applied Mathematics*, 128(1-2), 1-54.

Thouvenin, B., Gonzalez, J. L., Chiffoleau, J. F., Boutier, B., and Le Hir, P. (2007). "Modelling Pb and Cd dynamics in the Seine estuary." *Hydrobiologia*, 588, 109-124.

Tsynkov, S. V. (1998). "Numerical solution of problems on unbounded domains. A review." *Applied Numerical Mathematics*, 27, 465-532.

Turek, S. (1996). "A comparative study of time-stepping techniques for the incompressible Navier-Stokes equations: From fully implicit non-linear schemes to semi-implicit projection methods." *International Journal for Numerical Methods in Fluids*, 22(10), 987-1011.

Vinayan, V. (2003). Boundary-Integral Analysis of Nonlinear Diffraction Forces on a Submerged Body. Master thesis, Department of Ocean Engineering, Florida Atlantic University.

Vreugdenhil, C. B. (1994). *Numerical Methods For Shallow-Water Flow*, Kluwer Academic, Dordrecht, The Netherlands.

Wadzuk, B. M. (2004). "Hydrostatic and non-hydrostatic internerwave models," Ph.D dissertation, The Univeristy of Texas at Austin.

Weerakoon, S. B., Tamai, N., and Kawahara, Y. (2003). "Depth-averaged flow computation at a river confluence." *Proceedings of the Institution of Civil Engineers-Water and Maritime Engineering*, 156(1), 73-83.

Webster, R. (2007). "Algebraic multigrid and incompressible fluid flow." *International Journal for Numerical Methods in Fluids*, 53(4), 669-690.

Wiegel, R. L. (1964). Oceanographical Engineering. Dover Publication, Inc., Mineola, New York.

Wilders, P., Van Stijn, T. L., Stelling, G. S., and Fokkema, G. A. (1988). "A Fully Implicit Splitting Method for Accuracy Tidal Computations." *International Journal for Numerical Methods in Engineering*, 26, 2707-2721.

Wubs, F. W., de Niet, A. C., and Dijkstra, H. A. (2006). "The performance of implicit ocean models on B- and C-grids." *Journal of Computational Physics*, 211(1), 210-228.

Yuan, H., and Wu, C. (2004). "An implicit three-dimensional fully non-hydrostatic model for free-surface flows." *International Journal for Numerical Methods in Fluids*, 44(8), 811-835

Zhang, Y., Baptista, A. M., and III, E. P. M. (2004). "A cross-scale model for 3D baroclinic circulation in estuary–plume–shelf systems: I. Formulation and skill assessment." *Continental Shelf Research*, 24(18), 2187-2214.

Zhou, J. T., Lin, J. G., and Xie, Z. H. (2007). "A compact explicit difference scheme of high accuracy for extended Boussinesq equations." *China Ocean Engineering*, 21(3), 507-514.

Zhou, W. (2002). "An Alternative Leapfrog Scheme for Surface Gravity Wave Equations." *Journal of Atmospheric and Oceanic Technology*, 19(September), 1415-1423.