Development and Application of a Two-Dimensional Hydrodynamic Model for Riverine Floodplain Environments

By

STEPHEN WAYNE ANDREWS
B.S. (University of Delaware) 2004

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

CIVIL AND ENVIRONMENTAL ENGINEERING

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA,

DAVIS

Approved:

_____ (S.G. Schladow)

_____ (F.A. Bombardelli)

_____ (D.H. Schoellhamer)

Committee in Charge

2007

Abstract

The goals of this work were to: (1) create an accurate and efficient hydrodynamic and scalar transport model for application to riverine floodplain environments, (2) test the model framework with the use of three test cases simulating the relevant characteristics of floodplain flows, and (3) apply the model to a restored floodplain on the lower Cosumnes River to examine the distribution of residence times and implications for phytoplankton exports. In deriving the two-dimensional, depth-averaged governing equations for mass and momentum conservation and scalar transport, care was taken to create a robust model. Drag due to flow through vegetation was incorporated into the parameterization of bottom friction factors, which were calculated during model runtime using only prescribed vegetation heights and flow variables. The governing equations were discretized using a two-level, semi-implicit scheme with an iterative method for bottom friction. An advection algorithm was chosen that conserved energy through flow contractions and momentum through flow expansions. The model framework was found to perform well in all test cases, although specialized procedures to add and remove cells from the computational space were needed for simulation of wetting and drying.

The fully developed model was applied to a leveed floodplain on the lower Cosumnes River, CA. Tracer release simulations were used to approximate the spatial and temporal distributions of residence and flushing times. Mean residence times were found to vary considerably throughout the floodplain. Residence times were on the order of hours along the main flowpaths and days at the periphery. Mean flushing times were short during the rising limb of a floodpulse hydrograph, but increased to several days once inflows ceased. Simulations were run to assess increases in phytoplankton export assuming the speed of the floodplain draining phase could be varied. Decreasing the draining time from its current 42 day period to a 12 day period was found to have little effect on total phytoplankton exports from the floodplain. This implies exports may be maximized by allowing the draining phase to proceed over 12 days and then reflooding the basin.

Acknowledgments

I would like to thank Prof. S. Geoffrey Schladow for his insight, encouragement, and guidance through the graduate school process, and for providing the funding to make this research possible. I thank the professors who, through their courses, taught me the basic principles that have been applied in this work and who have taken time away from their full schedules to help with my work. These include Prof. Fabian Bombardelli, Prof. Bassam Younis, and Prof. David Schoellhamer in the Civil and Environmental Engineering Department, Prof. Jeff Mount in the Geology Department, and Prof. Gerry Puckett in the Mathematics Department at UC Davis. I also thank Prof. Francisco Rueda at the University of Granada, Spain for his help and advice in developing the model code and Prof. Guus Stelling at the Delft University of Technology for his prompt responses to my inquiries.

I would like to thank the other Cosumnes River floodplain researchers whose field data I have utilized in this study. The collection of such a large amount of field measurements as I had access to represented an enormous investment of time and without which this work would not have been possible. These people include floodplain field manager Carson Jeffres, Josh Viers for his help with the lidar data, Dylan Ahearn for the water quality data, and Eric Booth for his analysis of historical floods.

Thanks to current and previous members of the Environmental Dynamics Laboratory at UC Davis, including Laura DiPalermo, Jehan Fugitt, and Bill Fleenor, for their insights and encouragement. Their daily suggestions and support contributed much to this work.

Finally and most importantly, I would like to thank my wife Erin for her patience, love, and support.

Contents

1 Introduction **1**

1.1 Background, motivation, and overview of this work 1

1.2 Review of floodplain hydrodynamic models 6

1.2.1 Two-dimensional models 7

1.2.2 Three-dimensional models 8

1.2.3 Progress on floodplain-specific computational difficulties 8

1.2.4 Present model 10

2 Governing Equations **11**

2.1 Introduction 11

2.2 Mass conservation 11

2.3 Momentum conservation 13

2.3.1 Rotational effects 14

2.3.2 Vegetation effects 15

2.3.3 Boussinesq approximation 16

2.4 Scalar transport 17

2.5 Ensemble-averaged equations 19

2.6 Depth-averaged equations and boundary conditions 22

2.6.1 Continuity equation 23

2.6.2 Momentum equation 27

2.6.3 Scalar transport equation 35

2.6.4 Lateral boundaries 36

2.7 Turbulence closure schemes 36

2.8 Equation summary 39

3 Numerical Methods and Test Problems **41**

3.1 Simplified equations 41

3.2 Discretization methods 41

3.2.1 Continuity and momentum discretizations 42

3.2.2 Advection algorithm 45

3.2.3 Slope limiters . . . 49
3.3 Test Cases for Model Evaluation . . . 51
3.3.1 Flow over rapidly varying topography . . . 51
3.3.2 Domain wetting and drying . . . 59
3.3.3 Hydrograph routing . . . 65
3.4 Two-dimensional approximations . . . 71
3.4.1 Discretization methods . . . 71
3.4.2 Matrix solver . . . 74
3.4.3 Scalar advection algorithm . . . 74

4 Residence Time and Phytoplankton Genesis on a Restored Floodplain 82
4.1 Study Area . . . 82
4.2 Model Calibration and Validation . . . 87
4.3 Residence Time Distribution . . . 95
4.4 Implications for Primary Production . . . 101

5 Summary, Conclusions, and Future Directions 105

Bibliography 108

A List of Notations 115

B Expanded Finite Difference Discretizations 120

C SIFT2D Source Code 125
C.1 SIFT2D.f90 . . . 127
C.2 SIFT2D_advection.f90 . . . 128
C.3 SIFT2D_fd.f90 . . . 136
C.4 SIFT2D_friction.f90 . . . 149
C.5 SIFT2D_initial.f90 . . . 153
C.6 SIFT2D_types.f90 . . . 159
C.7 SIFT2D_update.f90 . . . 161
C.8 interp_hydro.m . . . 167

List of Figures

2.1 Illustration of mechanical diffusion caused by flow through vegetation (modified from Nepf (1999)) 18
2.2 Definition sketch for water depth variables 24
2.3 Schematic of the effects of submerged and emergent vegetation on vertical velocity profiles (modified from Wu *et al.* (2005)) 29
3.1 Staggered 1D grid 42
3.2 Schematic of a sudden bed transition 47
3.3 Relative head losses through a sudden bed contraction as a function of the upstream Froude number 48
3.4 Bottom elevation profile for flow over rapid expansions test case 54
3.5 Momentum conservation principle applied to flow over rapid expansions . . . 54
3.6 Energy head conservation principle applied to flow over rapid expansions . . 55
3.7 Bottom elevation profile for flow over rapid contractions test case 55
3.8 Momentum conservation principle applied to flow over rapid expansions . . . 56
3.9 Energy head conservation principle applied to flow over rapid expansions . . 56
3.10 Momentum conservation principle applied to flow over an obstacle 57
3.11 Energy head conservation principle applied to flow over an obstacle 57
3.12 Dynamic choice method applied to flow over an obstacle 58
3.13 Dynamic choice method with slope limiters applied to flow over an obstacle . 58
3.14 Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin on ebb tide. The bold line indicates the basin boundary. 61
3.15 Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin on flood tide. The bold line indicates the basin boundary. 61
3.16 Water surface profiles calculated using the wetting and drying method of Molinaro *et al.* (1994) for a uniformly sloped basin on ebb tide (from Balzano (1998)) 62

3.17 Water surface profiles calculated using the variable retention surface (VRS) wetting and drying method of Balzano (1998) for a uniformly sloped basin on ebb tide (from Balzano (1998)) 62
3.18 Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin with bottom convexity on ebb tide 63
3.19 Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin with bottom convexity on flood tide 63
3.20 Illustration of incorrectly upwinded depths at the cell edges using (3.4) and slope limiters 64
3.21 Spurious water surface gradient encountered near a wetting front 64
3.22 Time evolution of a hydrograph advected downstream in the hydrograph routing test case 66
3.23 RMS error convergence with number of iterations 68
3.24 RMS error convergence with increasing time and space resolution 68
3.25 Amplitude error convergence with increasing time and space resolution . . . 69
3.26 Phase error convergence with increasing time and space resolution 69
3.27 Mass preservation error convergence with increasing time and space resolution 70
3.28 Comparison of upwind and slope limited solutions with $\Delta x = 1000$ ft and $\Delta t = 25$ s 70
3.29 Staggered 2D grid 71
3.30 Two-dimensional 'natural' ordering method 73
3.31 Upwind method of scalar transport 75
3.32 CTU method of scalar transport 76
3.33 Initial tracer distribution 78
3.34 Tracer distribution at 2.5 seconds for a high resolution simulation 79
3.35 Tracer distribution at 5.0 seconds for a high resolution simulation 79
3.36 Tracer distribution for the upwind method after 5 seconds 80
3.37 Tracer distribution for the second order method at 2.5 seconds 80
3.38 Tracer distribution for the second order method after 5 seconds 81
3.39 Tracer distribution for the CTU method after 5 seconds 81

4.1 Map of Central California hydrology showing relative locations of the Cosumnes River, the Cosumnes Triangle Floodplain (TFP), and the Michigan Bar gauging station (MHB) 83
4.2 Map showing location of Cosumnes Triangle Floodplain field site relative to the Lower Cosumnes River (from Florsheim & Mount (2002)) 84
4.3 Cosumnes floodplain site map showing sampling locations 84
4.4 Cosumnes floodplain topography 86
4.5 Cosumnes floodplain vegetation heights 86
4.6 Vertical temperature structure at a location in the central pond during WY2003 88
4.7 Water surface elevations for the flood event used in model calibration 89
4.8 Modeled and measured water surface elevations at TP for the calibration flood event 91
4.9 Modeled flow paths for the falling limb of the calibration flood event; 0.5 m surface elevation countours are shown 92
4.10 Stage-discharge relationships for TE and TW outlets 93
4.11 Modeled and measured water surface elevations at TP for the validation flood event 93
4.12 Modeled and measured praseodynium concentrations for the validation flood event for sites 1-13 and TE. Concentration units are in pptr. 94
4.13 Spatial distributions of residence time on the Cosumnes River floodplain. The square symbols designate locations of simulated tracer releases. 97
4.14 Chlorophyll-a concentrations at the end of a small flood event in WY2005 (from Ahearn *et al.* (2006)) 98
4.15 Time distribution of flushing times through the calibration flood event. The dotted line indicates the time of disconnection between channel flow at TN and the floodplain. 100
4.16 Transport time dependence on flood magnitude. The numbers above the first set of bars indicate the multiplier used on the base storm magnitude. 100
4.17 Outlet water surface elevations for a draining event during the spring of WY2004 102

List of Tables

2.1 Coefficients for short vegetation friction factor calculation (modified from Mason *et al.* (2003)) 33
2.2 Empirical coefficients for tall vegetation friction factor calculation (modified from Kouwen & Fathi-Moghadam (2000)) 34
2.3 Values for empirical constants in the k-ε turbulence closure scheme 38
3.1 Criteria for the conservation principle to apply to the advection approximation 49
3.2 Slope limiter formulas 50
3.3 Performance of slope limiter methods on the hydraulic jump test case 53
4.1 Algal increase factors for crashing population curves for draining phase simulations 104

1 Introduction

1.1 Background, motivation, and overview of this work

Natural river floodplains are widely regarded as being among the most biologically productive and diverse ecosystems on earth. They owe their biological productivity to periodic enrichment by import and retention of sediment and nutrients and their diversity to a combination of factors, including high primary productivity, their role as a transition region between aquatic and terrestrial ecosystems, and the mosaic of habitat diversity caused by flood disturbance events (Naiman & Decamps, 1997). Floodplains have an estimated worldwide extent of 2×10^6 km^2 and may be defined as 'areas of low lying land that are subject to lateral overflow water from rivers or lakes with which they are associated' (Tockner & Stanford, 2002; Junk & Welcomme, 1990). Floodplain ecosystems may also result from innundation by direct precipitation (such as the Pampa grasslands of Argentina) or groundwater (such as the in-channel 'flow pulse' concept of Tockner *et al.* (2000)) in addition to overland flow. Although floodplains that fringe river systems will be the focus of this study, the model developed herein will be equally applicable to other types of freshwater floodplains.

Floodplains are responsible for numerous ecosystem and human benefits. In particular, floodplains may provide:

- Flood control through the downstream attenuation of peak river discharges,
- Groundwater recharge and local water storage (Tockner & Stanford, 2002),
- Sediment retention and regulation (Mount, 1995),
- Waste treatment and water quality improvement (Tockner & Stanford, 2002; Miller, 2000),
- Nitrogen removal (Tockner *et al.*, 1999; Hubbard & Lowrance, 1996),
- Downstream augmentation of foodwebs through the export of autochthonous carbon (e.g., dissolved organic carbon, algal biomass, leaf litter) (Ahearn *et al.*, 2006),
- Fisheries production (Sommer *et al.*, 2001b),
- Habitat for endangered species,

- Fertile soils for crop production, or
- Recreational opportunities.

Although floodplains are a visible and relatively common component of the landscape, they are among the most threatened ecosystems due to habitat alteration, flow and flood control, species invasion, and pollution (Tockner & Stanford, 2002). Anthropogenic modification of the natural hydrologic regime has led to the loss of an estimated 90% of the original floodplains in North America and Europe (Tockner & Stanford, 2002). Owing to their nutrient rich soils, water supply, relatively flat terrain, and proximity to a navigable waterway, floodplains have been heavily modified for settlement and agriculture (Alexander & Marriott, 1999). The hydrologic regimes of many rivers have been altered by flow diversion and levee and dam construction, all of which can reduce the magnitude, frequency, and duration of peak flows on the floodplain, reduce hydrologic connectivity, and in many cases render the floodplain functionally extinct (Poff *et al.*, 1997).

Much effort has been directed in recent years to restoring and preserving floodplain ecosystems with the hope of regaining their accompanying ecosystem services. There are currently large-scale projects seeking to restore degraded floodplain areas in the Florida Everglades, the Mississippi River Delta, the California Bay-Delta, and the Chesapeake Bay, as well as many smaller scale projects worldwide. Effective restoration and preservation plans usually require knowledge of the hydrologic connectivity between a floodplain and its adjacent river as well as the water circulation and mixing occuring on the floodplain. These physical processes drive most of the ecological processes of interest and dictate the human uses of floodplains.

The driving force behind the development of the model described in this work was the need to quantify exports of autochthonous carbon from a restored floodplain on the Cosumnes River in the Sacramento-San Joaquin Delta region of California. Exports of autochthonous carbon in the form of algal biomass are largely a function of hydrodynamic flow conditions, particularly residence time. In general, floodplain water with relatively long residence times will have slow water velocities, resulting in low suspended sediment concentrations, warmer temperatures, and sufficient time for the floodplain water chemistry to diverge from the river water chemistry (Ahearn *et al.*, 2006; Baranyi *et al.*, 2002; Sommer

et al., 2001a). This creates a more favorable environment for algae growth and thus algal biomass concentrations have been found to increase, to a point, with floodplain residence time (Baranyi *et al.*, 2002).

The genesis of floodplain derived autochthonous carbon has both local and regional consequences. Locally, increased algal biomass has been shown to augment lower trophic levels, enhancing the growth and survival of such secondary consumers as birds, salmon, and other native fish (Sommer *et al.*, 2001b). Periodic connection and isolation of the floodplain with its adjacent channel is responsible for export of algal biomass to downstream ecosystems, supporting regional foodwebs (Ahearn *et al.*, 2006). Recently, there have been severe declines in higher order consumers in the Delta, from zooplankton to native fish (Bennett & Moyle, 1996; Kimmerer & Orsi, 1996). Jassby & Cloern (2000) hypothesize these declines to be the result of a decreased input of phytoplankton and cite the increased innundation of floodplains as a possible remediation method. The model described herein will be used to assess the spatial and temporal distribution of residence time for a restored floodplain on the Cosumnes River, California, USA and the potential of different hydrological regimes to generate algal biomass. It is hoped that the results of these simulations will be used by scientists and regional planners to guide remediation plans for the Delta ecosystem and better allocate water releases and resources.

The application described above is just one example of an ecologically based study that can be performed with the model described in this work. If information is known about the habitat preferences of certain wetland organisms, the model may be used to assess the potential of water releases to create beneficial habitat for species of interest. These species of interest may include certain floodplain plants or plant communities, wintering or native birds, or particular life stages of fish species. A floodplain manager may then be able to prescribe water releases to the floodplain not only to maximize phytoplankton production, but also to optimize, for example, the extent of spawning or rearing grounds available for an endangered fish or the flow conditions needed for vegetative seed dispersal and recruitment. Studies such as these may help answer questions about the tradeoffs involved in ecosystem management for floodplain systems.

The model may also be useful for the classical problem of predicting the extent, mag-

nitude, and duration of flooding in riverside communities in response to an upstream rain event. Flooding represents a major human hazard, and accurate predictions of flood events are essential for preserving human life and reducing economic loss. In particular, simulations may be used in the development and assessment of flood control structures, land use designations, or flood warnings and evacuation plans.

There are several reasons why the creation of an original hydrodynamic model was necessary for this study. Existing free and low-cost hydrodynamic codes often lack efficient computational methods for calculating flow phenomena important to floodplains, such as the influence of submerged vegetation, very shallow water depths, and complex topography. Free-source codes also may have sparse documentation or require much time and effort to be devoted to understanding the code before it can be applied properly. Whereas commercial software generally has better support and a wider range of accurate applicability, it is often very expensive and does not allow viewing or modification of the source code.

Floodplain flows have several computationally complicating attributes that are often not addressed in conventional hydrodynamic models. A major issue concerns the wetting and drying of computational grid cells. When water depths approach zero, many numerical schemes become unstable, and specialized procedures are necessary. A variety of algorithms have been developed to add and remove cells from the computational domain based on their water depth, but these methods also have computational problems associated with them (Balzano, 1998). Another challenge for floodplain models is the need to correctly predict flow variables for both the highly advective channel environment and the less advective floodplain environment. For most natural floodplains, boundary conditions for floodplain flows are very difficult to determine without also simulating flow in the channel. The fast moving flows typically encountered in river channels may result in areas of supercritical flow and consequently discontinuities in the fluid free surface profile. Hydraulic jumps, along with the scale contractions that may be experienced by a scalar variable, necessitate the use of mass conservative methods in order to correctly capture the solution behavior near discontinuities. Other modeling difficulties include effects due to submerged and emergent vegetation and rapidly varying topography. The model presented herein will make use of recent advancements in the treatment of such problems in an attempt to overcome historic

modeling difficulties.

A two-dimensional, depth-averaged model was chosen for this study. At a minimum, two-dimensional models are needed to examine most problems of interest because of the high degree of lateral heterogeneity encountered on floodplains. This heterogeneity includes rapid variations in topography, vegetation, and bottom roughness and is a result of the dynamic and spatially variable nature of overbank flow disturbance events. The choice between two- and three- dimensional models is largely based on the proposed application and the computer resources available. The most common modeling approach for natural river floodplains has been the two-dimensional, depth-averaged type (Nicholas & McLelland, 2004). Depth-averaged models assume the water column is well-mixed in the vertical direction. This is generally an accurate approximation because of the shallow depths. Three-dimensional models will predict information about the structure of flow in the vertical direction, which may be useful in examining fish spawning habitat or sediment transport. However, three-dimensional models are much more computationally intensive and require more detail in specifying model boundary conditions and accounting for the effects of vegetation on flow patterns and turbulence (Nicholas & McLelland, 2004).

This work is organized into five sections. The remainder of Section 1 reviews recent advancements in floodplain hydrodynamic modeling and introduces the methods and improvements used in the present model. Section 2 presents a detailed derivation of the governing equations and boundary conditions with an emphasis on assumptions and approximations that may limit the applicability of the model. Section 3 introduces the spatial and temporal discretization methods used in the numerical solution of the governing equations. The schemes presented are examined for their consistency, accuracy, and stability properties and then evaluated for their performance on test problems designed to simulate different aspects of floodplain flow. In Section 4, the model is applied to a restored floodplain on the Cosumnes River, CA to examine the impact of varying hydrograph inputs on the genesis of phytoplankton biomass. The work closes with a summary and conclusions.

1.2 Review of floodplain hydrodynamic models

The numerical solution methods used in floodplain models are designed to convert the partial differential equations that govern fluid flow into a set of algebraic equations that can be solved efficiently using computers. These methods can be broken into two major classes: grid-point methods (including finite-difference and finite-volume methods) and series-expansion methods (including the finite-element method). Grid-point methods represent the exact solution with a finite set of values defined at discrete intervals (Durran, 1999). Of these, finite-difference methods are developed by simple substitution of discrete operators for continuous ones in the differential governing equations. Finite-volume methods are a variation on finite-difference methods in which the domain is broken into grid cells rather than pointwise approximations. Fluxes into and out of computational cells are computed instead of approximations to the derivatives, and finite-volume methods generate approximations to the average value of the solution over the cell, a property that is not guaranteed by classical finite-difference methods (Durran, 1999). When approximating differential equations with solutions containing discontinuities, classical finite difference methods may break down when the differential equation does not hold. Finite-volume methods avoid this problem by being based on the integral form of the governing equation which will not be violated near a discontinuity in the solution (LeVeque, 2002).

Series-expansion methods are derived by approximating the unknown solution with a linear combination of continuous expansion functions. The solution is represented by a set of the expansion coefficients and the derivatives are calculated analytically. The most popular series expansion used in floodplain models is the finite-element method, which is classified by having an orthogonal set of expansion functions that are nonzero for only a small subset of the computational domain (Durran, 1999).

The choice between grid-point methods and series-expansion methods for floodplain models is largely based on individual philosophy. Numerous examples of both types of models exist in the literature and there have been no extensive studies comparing the two for differences in accuracy, stability, or efficiency in floodplain simulations. Rueda & Schladow (2002) examined differences between a three-dimensional finite-difference method, SI3D (Smith, 1997), and a finite-element method, RMA10 (King, 1993), using analytical test cases

designed to simulate different aspects of lake hydrodynamics. Their results are model and test case specific and are not generalizable to the entire classes of finite-difference and finite-element methods. However, the finite-difference method was found to run approximately two orders of magnitude faster when the same number of computational elements were used. RMA10 was able to represent the boundaries to a high degree of accuracy using fewer computational nodes than SI3D, but the finite-element simulations with fewer nodes still had longer run times, a higher degree of phase error, and poorer mass and energy conservation properties than the finite-difference model (Rueda & Schladow, 2002). The lack of analytical solutions for highly nonlinear floodplain relevant flows and the lack of extensive field measurements during flood events (because of the risks associated with sampling in high flows) have probably hampered comparative studies.

1.2.1 Two-dimensional models

Two-dimensional (2D) hydrodynamic models have been available since the early 1960's and were often used to perform dam break simulations for populated areas at risk for flooding. The earliest codes used explicit finite-difference methods and are still able to produce accurate results in a variety of applications (Sielecki, 1968). One advancement generated during their development was the semi-implicit method. The semi-implicit method treats gravity slope terms in the momentum equation implicitly and all other terms explicitly. In this way the constraint placed on the model time step by propagating surface waves is removed and simulations can be run much more efficiently. The equation for the water surface elevations is effectively decoupled from the velocity equations and less variable storage is required. Although many of the early finite-difference methods have been abandoned for more mass conservative schemes, their efficiency makes them attractive choices for simulations involving large numbers of computational cells. For this reason, accurate finite-difference schemes such as the MacCormack method are still in wide use today.

Finite-element models were developed later, probably because they are more difficult to code and conceptually understand. By discretizing the domain into a non-uniform network of nodes and elements, complex domain boundaries can be better represented. This is a very useful property in complex, topography-driven floodplain environments. Current models

developed for specific application to floodplains in common use include the TELEMAC-2D model of Hervouet & Janin (1994), the RMA2 model (Norton *et al.*, 1973), and the models of Brufau *et al.* (2002) and D'Alpos & Defina (1993). The latter model includes the coupling of one-dimensional (1D) channel segments to the edges of 2D floodplain elements to account for small scale drainage networks that are unresolvable on a large model grid.

Finite-volume methods were the most recent methods to come into use in floodplain simulations, but have since become very popular because of their conservation properties. Specific numerical methods include Riemann solvers, flux-corrected transport, and Godunov methods. Popular 2D finite-volume methods in current use include the hydro2de model of Beffa & Connell (2001) and the SSIIM method of Olsen & Stokseth (1995). Hybrid methods combining aspects from both finite-volume and finite-element methodologies have also been developed by DiGiammarco & Todini (1994) and Bradford & Sanders (2002).

1.2.2 Three-dimensional models

Relatively few three-dimensional (3D) floodplain hydrodynamic investigations have been made because of the complexities inherent in the specification of boundary conditions and the effect of emergent and submerged vegetation (Erduran & Kutija, 2003). One successful approach has been to use a 2D model to obtain boundary conditions for the 3D model (Nicholas & McLelland, 2004). Those 3D studies that have been done have largely focused on isolated flow structures (Nicholas & McLelland, 2004; Fischer-Antze *et al.*, 2001), and have used either the finite-element solver RMA10 or the finite-volume commercial code FLUENT CFD® (Teeter *et al.*, 2001; Nicholas & McLelland, 2004).

1.2.3 Progress on floodplain-specific computational difficulties

Floodplain environments and flow structures present a number of computational difficulties that must be addressed in numerical models. These include but are not limited to: (1) small scale water storage and drainage networks, (2) the wetting and drying of computational cells, (3) flow through vegetation, and (4) complex bottom topography.

Small scale water storage and drainage networks have been addressed using subgrid techniques. These have largely been developed by Defina (2000) and involve modifying the

governing equations in order to account for variations in bottom roughness, water storage, and drainage networks at levels unresolvable on the model grid. The advancements were furthered to included wetting and drying algorithms by Bates & Hervouet (1999) and are used in the finite element schemes applied by Bates *et al.* (1992).

Wetting and drying of computational cells has long been a problem in flooding simulations. In addition to numerical instabilities as water depths approach zero, the propagation of floods over initially dry areas can be problematic and the speed of advancing flood waves may be incorrectly calculated using certain schemes. Many options have been proposed to mitigate these issues. Many models include routines to remove cells from computation when water levels drop below a certain specified tolerance. The first such models were used in the 1970's and are reviewed in detail in Balzano (1998). Other models constrain water depths to remain non-negative through creative discretization of the governing equations or simply reset water levels that fall below a certain small depth to a prescribed value (Stelling & Duinmeijer, 2003; Braschi *et al.*, 1994). Other methods include explicitly modeling the groundwater table (the 'marsh' option) or adapting the mesh to conform to moving boundaries (Nielsen & Apelt, 2003).

Flow through vegetation presents analytical problems in choosing a form of the governing equations to solve as well as numerical difficulties. These issues include the parameterization of the drag force term and effects on turbulence and scalar diffusion. Nepf (1999) showed flow through plant stems causes mechanical diffusion of an advected scalar which can be of the same order as turbulent diffusion in densely vegetated areas. An analytical treatment of the problem led to a successful modeling approach. Work by Kouwen (1988), Fathi-Moghadam & Kouwen (1997), Darby (1999), and Mason *et al.* (2003) was done on the parameterization of bottom frictional values to account for flow through submerged and emergent vegetation. The methods they developed predict roughness coefficients and take into account the deflection of plants during high velocity flows. In the treatment of turbulence, López & García (1998) showed that additional production terms were needed in equations for the k-ε turbulence closure scheme to account for flow through vegetation.

Finally, the correct prescription of bottom topography has hampered the accuracy of floodplain models. Large alluvial rivers can have very large floodplains with only small

differences in topography. The incorrect specification of this topography can cause large changes in the modeled flow patterns and resulting innundation patterns. The recent use of remotely sensed data from GIS (DiGiammarco *et al.*, 1994), laser altimetry (Cobby *et al.*, 2003; Mason *et al.*, 2003) and synthetic apperature radar (Horritt & Bates, 2001) has mitigated this problem.

1.2.4 Present model

The present model seeks to utilize some recent advances in floodplain specific flows in order to overcome historic modeling difficulties. In particular, the semi-implicit method for treatment of terms in the momentum equation will be used so that the surface gravity wave constraint can be avoided and models can be run more efficiently. The finite-volume advection method of Stelling & Duinmeijer (2003) will be used to account for the wetting and drying of computational cells without the use of specialized procedures. The drag term for flow through vegetation will be avoided with the use of the bottom friction parameterization of Darby (1999), and the results of a high-resolution laser altimetry survey will be used to obtain an accurate distribution of ground surface elevations and vegetation heights. A k-ε turbulence closure scheme will be used that takes the turbulent kinetic energy production due to flow through vegetation implicitly into account, and the high-resolution corner transport upwind method of Leveque (1996) will be used to model scalar transport to 'almost second order' accuracy while maintaining positive concentrations.

2 Governing Equations

2.1 Introduction

Chapter 2 presents a derivation of the partial differential equations and associated boundary conditions governing unsteady fluid flow in a freshwater floodplain environment. Throughout the derivation, emphasis will be placed on the assumptions and approximations used to tailor the basic Navier-Stokes equations to a generalized two-dimensional floodplain environment. Derivations of the equations governing conservation of mass, conservation of momentum, and scalar transport are presented in Sections 2.2, 2.3, and 2.4. The governing equations are ensemble-averaged in Section 2.5 and then depth-averaged in Section 2.6. Finally, the equations and theoretical basis for two turbulence closure schemes are presented in Section 2.7. The result will be a set of partial differential equations and their associated boundary conditions governing freshwater flow and scalar transport on a generalized floodplain. Analytic solutions to these non-linear equations have not yet been found, and the numerical methods used to approximate solutions are presented in Section 3.

2.2 Mass conservation

The equation governing the conservation of mass in a fluid can be derived using the Reynolds Transport Theorem. It is given as

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) = 0 \quad , \tag{2.1}$$

where ρ is the density of the fluid, $\mathbf{u} = u\mathbf{i}+v\mathbf{j}+w\mathbf{k}$ is a vector composed of the instantaneous fluid velocity components in the x, y, and z directions, and

$$\boldsymbol{\nabla} \cdot (\rho \mathbf{u}) \equiv \frac{\partial (\rho u)}{\partial x} + \frac{\partial (\rho v)}{\partial y} + \frac{\partial (\rho w)}{\partial z}$$

is the divergence operator applied to the quantity $\rho\mathbf{u}$. Equation (2.1) states that the rate of change of mass within a differential control volume must equal the net rate at which mass passes through the differential surface bounding the control volume. This equation can be algebraically manipulated to

$$\frac{1}{\rho}\left(\frac{D \rho}{Dt}\right) + \boldsymbol{\nabla} \cdot \mathbf{u} = 0 \quad , \tag{2.2}$$

which makes use of the substantial derivative operator, D/Dt,

$$\frac{D\,\rho}{Dt} \equiv \frac{\partial\,\rho}{\partial\,t} + (\mathbf{u}\cdot\boldsymbol{\nabla})\rho \quad , \quad (\mathbf{u}\cdot\boldsymbol{\nabla})\rho \equiv u\frac{\partial\,\rho}{\partial\,x} + v\frac{\partial\,\rho}{\partial\,y} + w\frac{\partial\,\rho}{\partial\,z} \quad .$$

It is common practice to neglect the density derivative term in (2.2) as very small compared to the advective term (Smith, 1997). This assumption can be analyzed by examining the relative magnitude of the density derivative via a scaling analysis. Over a characteristic length scale L, it is assumed that fluid velocity varies by U_c and density varies by $\delta\,\rho$ from a background density, ρ_0. Then

$$\frac{\frac{1}{\rho_0}\left(\frac{D\,\rho}{D\,t}\right)}{\boldsymbol{\nabla}\cdot\mathbf{u}} \sim \frac{u\left(\frac{\partial\,\rho}{\partial\,x}\right)}{\rho_0\left(\frac{\partial\,u}{\partial\,x}\right)} \sim \frac{U_c\left(\frac{\delta\rho}{L}\right)}{\rho_0\frac{U_c}{L}} = \frac{\delta\rho}{\rho_0}$$

and the density derivative term can safely be neglected so long as the density correction term, $\frac{\delta\rho}{\rho_0}$, is much smaller than unity (Kundu & Cohen, 2004).

The density of a given fluid can change with pressure, temperature, or composition (e.g., salinity or sediment concentration) (Kundu & Cohen, 2004). For depths typically encountered in floodplain environments (meters to tens of meters), water is considered incompressible and thus density does not change with pressure (Gill, 1982). Changes in density with temperature can be quantified using the relation

$$\frac{\delta\,\rho}{\rho_0} = \alpha_t\,\delta\,T \quad , \tag{2.3}$$

where $\delta\,\rho$ and $\delta\,T$ are small changes in density and temperature, and $\alpha_t \sim -2\times10^{-4}\,\mathrm{K}^{-1}$ is the thermal expansion coefficient for water (Gill, 1982). A typical rule of thumb is that thermal density corrections are negligible for temperature differences below 20°C, corresponding to $\frac{\delta\rho}{\rho_0} \approx 0.4\%$ (Buscaglia *et al.*, 2002). Maximum changes in floodplain temperatures are almost always less than 20°C and thus changes in density due to temperature can safely be ignored (Tockner *et al.*, 1999; Malard *et al.*, 2001).

Changes in fluid density due to suspended sediment concentration may seem significant. Floodplains, whether natural or engineered, are typically innundated only by high flow events. Since suspended sediment derived from both erosion of the river bed and overland flow is known to increase exponentially with river discharge (Allen, 1997), the high flows resulting in floodplain innundation usually have very high suspended sediment concentrations. However, rapid decreases in fluid velocity for water entering a floodplain will induce

particle settling, and the filtering effect of floodplain vegetation will remove particles from suspension (Mount, 1995). If changes in fluid density as a function of suspended sediment concentration are considered using the equation from Wang *et al.* (2005),

$$\frac{\delta\,\rho}{\rho_0} = \frac{1}{\rho_0}\left(1 - \frac{\rho_0}{\rho_s}\right) c_{sed} \quad ,$$

where $\rho_s \approx 2.65\,\rho_0$ is the density of the sediment and c_{sed} is the sediment concentration, suspended sediment concentrations greater than 6000 mg/L would be needed for the density derivative term in (2.2) to become significant. Even in lowland tropical river basins such as the Amazon, observed floodplain suspended sediment concentrations rarely exceed this threshold, and therefore changes in density due to suspended sediment can be neglected (Mertes, 1997; Mertes *et al.*, 1993).

By disregarding the density derivative term in (2.2), conservation of fluid volume is effectively substituted for conservation of fluid mass. This approximation is the first half of the set of simplifying assumptions known as the Boussinesq approximation and results in the three-dimensional continuity equation shown below.

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0 \tag{2.4}$$

2.3 Momentum conservation

The equations governing the conservation of linear momentum in a moving fluid relative to a non-rotating reference frame can be derived with the Reynolds Transport Theorem. The result is the general Navier-Stokes equation for momentum,

$$\rho\frac{D\,\mathbf{u}}{D\,t} = -\boldsymbol{\nabla} P + \rho\mathbf{g} + \mu\boldsymbol{\nabla}^2\mathbf{u} \quad , \tag{2.5}$$

where P is the instantaneous pressure, $\mathbf{g}$ is the acceleration due to gravity in the x, y, and z directions, μ is the dynamic viscosity of the fluid, and

$$\boldsymbol{\nabla} P \equiv \frac{\partial\,P}{\partial\,x}\mathbf{i} + \frac{\partial\,P}{\partial\,y}\mathbf{j} + \frac{\partial\,P}{\partial\,z}\mathbf{k} \quad , \quad \boldsymbol{\nabla}^2 u \equiv \frac{\partial^{\,2} u}{\partial\,x^2} + \frac{\partial^{\,2} u}{\partial\,y^2} + \frac{\partial^{\,2} u}{\partial\,z^2}$$

are the gradient and Laplacian operators. Equation (2.5) states the time rate of change of the linear momentum in a control volume is equal to the sum of the external forces acting upon the control volume and the net flux of momentum through the control surface. The

first term on the right hand side is the pressure gradient and results from horizontal water surface gradients in a free surface flow or curvature of streamlines in general. The second term is a gravity body-force term and can be simplified to $\rho\mathbf{g} = (0,\, 0,\, -\rho g)$ for a chosen orthogonal coordinate system where the x-axis is directed positive horizontally to the east, the y axis is positive horizontally to the north and the z-axis is directed positive vertically upward along the line of action of gravity (Smith, 1997). The final term in (2.5) is the shear force term and represents internal fluid forces resulting from laminar fluid viscosity (Smith, 1997).

2.3.1 Rotational effects

For very wide floodplains it is necessary to consider the effects of a rotating frame of reference. The rotation of the earth produces an additional term in the momentum equation, the Coriolis force, derived in detail in Kundu & Cohen (2004) and Smith (1997).

$$\rho\frac{D\,\mathbf{u}}{D\,t} = -\boldsymbol{\nabla}P + \rho\mathbf{g} + \mu\boldsymbol{\nabla}^2\mathbf{u} - 2\,\rho\,\boldsymbol{\Omega}\times\mathbf{u} \tag{2.6}$$

The Coriolis force, $-2\boldsymbol{\Omega}\times\mathbf{u}$ where $\boldsymbol{\Omega}$ is the angular velocity vector of the earth about its rotational axis and $\times$ is the cross product operator, is known a pseudo-force because it arises solely from the use of a non-inertial frame of reference. It deflects motion to the right in the Northern hemisphere and to the left in the Southern hemisphere. A common approximation for this term is

$$-2\boldsymbol{\Omega}\times\mathbf{u} \approx f_c v\mathbf{i} - f_c u\mathbf{j} \quad , \quad f_c \equiv 2||\boldsymbol{\Omega}||\sin\phi \quad ,$$

where ϕ is the latitude and $||\boldsymbol{\Omega}|| = 2\pi$ rad/day $\approx 7.3\times10^{-5}$ s^{-1} is the magnitude of the angular velocity of the earth. The parameter f_c is the Coriolis frequency and at $\phi = 45°$ N latitude is approximately 1.0×10^{-4} s^{-1}. The magnitude of the Coriolis force relative to nonlinear fluid acceleration can be examined using the Rossby number

$$Ro \equiv \frac{\text{nonlinear acceleration}}{\text{Coriolis force}} \sim \frac{u\frac{\partial u}{\partial x}}{f_c u} \sim \frac{U_c}{f_c L} \quad ,$$

where L and U_c are representative horizontal length and velocity scales (Kundu & Cohen, 2004). If Ro is less than or approximately equal to one, the Coriolis force is significant and must be included in the governing equations. The majority of floodplains that fringe rivers

in the United States have average widths less than 100 m and representative fluid velocities around 0.2 m/s, resulting in $Ro \sim 20$ (Tockner & Stanford, 2002; Mertes, 1997). However, large tropical lowland rivers such as the Amazon and engineered systems such as California's Yolo Bypass may have length scales above 10 km and thus relatively large Coriolis forces (Tockner & Stanford, 2002; Sommer *et al.*, 2001a). Since evaluation of the extra terms in a numerical simulation requires very few additional computations, the Coriolis terms will be retained and kept in the form $-2\,\rho\,\mathbf{\Omega} \times \mathbf{u}$ for brevity.

2.3.2 Vegetation effects

Vegetation on natural river floodplains can significantly affect flow processes. Submerged and emergent floodplain vegetation produce high flow resistance and thus may have large impacts on circulation patterns and water levels, geomorphic processes, and aquatic habitat (Wu *et al.*, 2005). The high flow resistance encountered on floodplains can significantly reduce maximum in-channel flows for downstream areas and has led to the use of floodplains as flood control structures in areas such as the Yolo Bypass, CA (Sommer *et al.*, 2001a). Nicholas & McLelland (2004) cite the spatial heterogeneity of floodplain vegetation communities and topographic features as the principle determinants of floodplain hydrodynamic patterns.

Although vegetation has a large effect on flow properties, its incorporation into the analytic framework is difficult. Floodplain vegetation will have different effects on the flow depending upon whether it is emergent or submerged. High fluid velocities can cause vegetation to bend in the direction of flow, signficantly reducing its momentum absorbing area. Flow through stems and leaves will affect turbulence by adding terms to the turbulent kinetic energy budget and will increase the diffusivity for scalar transport through mechanical diffusion. Even when vegetation effects can be incorporated in the governing equations, the evaluation of such terms may be hampered by cumbersome data requirements (e.g., spatial distributions of vegetation density or leaf areas, or unique flexural rigidity values for different plants species) or unrealistic assumptions (e.g., all vegetation modeled as emergent rigid cylinders).

The effect of vegetation is typically modeled with the addition of a drag sink term to

the momentum equation. From dimensional analysis, the drag force exerted by an object on the flow can be represented as

$$\mathscr{D} = \frac{1}{2}\,\rho\, C_D\, \mathbf{u}||\mathbf{u}||A \quad ,$$

where $\mathscr{D}$ is the drag force due to both skin friction and form drag from the resulting non-uniform pressure distribution, A is the characteristic area of the object, and C_D is a dimensionless drag coefficient. For vegetation, the drag coefficient is the subject of many studies and has been shown to be a function of flow depth and velocity, and vegetation stem diameter, spacing, flexural rigidity, and leaf area index, to name a few (Fathi-Moghadam & Kouwen, 1997). Vegetative drag is modeled in the momentum equation with the addition of the final term in (2.7)

$$\rho\frac{D\,\mathbf{u}}{D\,t} = -\boldsymbol{\nabla} P + \rho\mathbf{g} + \mu\boldsymbol{\nabla}^2\mathbf{u} - 2\,\rho\,\boldsymbol{\Omega}\times\mathbf{u} - \frac{1}{2}\,\rho\,\lambda\, C_D\,\mathbf{u}||\mathbf{u}|| \quad , \tag{2.7}$$

where λ is the vegetation's momentum absorbing area projected normal to the flow per unit volume (Fathi-Moghadam & Kouwen, 1997). If vegetation is modeled as emergent rigid cylinders of diameter d_c, λ will be given by

$$\lambda = n_c d_c = \frac{d_c h}{(\Delta S)^2 h} = \frac{d_c}{(\Delta S)^2} \quad ,$$

where n_c is the number of cylinders per unit area, ΔS is the mean spacing between cylinders, and h is the total water depth (Nepf, 1999). It is therefore possible for λ to take on values greater than unity and, for floodplain vegetation, λ has been measured in the range of 2-20 (Nicholas & McLelland, 2004; Nepf, 1999).

2.3.3 Boussinesq approximation

A common simplification to the momentum equation also concerns the well-known Boussinesq approximation. The instantaneous pressures and densities are broken into hydrostatic reference components, P_0 and ρ_0, and dynamic components, $\tilde{P}$ and $\tilde{\rho}$, as

$$P = P_0 + \tilde{P} \qquad , \qquad \rho = \rho_0 + \tilde{\rho} \quad .$$

The hydrostatic components are defined by the hypothetical state where density has a constant value of ρ_0 and the corresponding pressures, $P_0(z)$, satisfy the hydrostatic equation,

$$\boldsymbol{\nabla} P_0 = \rho_0\mathbf{g} \quad . \tag{2.8}$$

By subtracting the hydrostatic reference state (2.8) from (2.7) and dividing by ρ_0, the momentum equation becomes

$$\left(1+\frac{\tilde{\rho}}{\rho_0}\right)\frac{D\,\mathbf{u}}{D\,t} = -\frac{1}{\rho_0}\boldsymbol{\nabla}\tilde{P}+\frac{\tilde{\rho}}{\rho_0}\mathbf{g}+\nu\boldsymbol{\nabla}^2\mathbf{u}-2\left(1+\frac{\tilde{\rho}}{\rho_0}\right)\boldsymbol{\Omega}\times\mathbf{u}-\frac{1}{2}\left(1+\frac{\tilde{\rho}}{\rho_0}\right)\lambda\,C_D\,\mathbf{u}||\mathbf{u}|| \quad (2.9)$$

where $\nu \equiv \mu/\rho_0$ is the kinematic viscosity. For floodplain environments where the vertical flow scale is small, departures from the hydrostatic reference state will have only a minor effect on the inertial, Coriolis, and drag terms and can thus be neglected. Density variations multiplied by gravity will be significant and are therefore retained in the equations. Adding (2.8) back to the simplified version of (2.9) yields the three-dimensional momentum equation shown below

$$\rho_0\frac{D\,\mathbf{u}}{D\,t} = -\boldsymbol{\nabla}P+\rho\mathbf{g}+\mu\boldsymbol{\nabla}^2\mathbf{u}-2\,\rho_0\,\boldsymbol{\Omega}\times\mathbf{u}-\frac{1}{2}\,\rho_0\,\lambda\,C_D\,\mathbf{u}||\mathbf{u}|| \quad , \qquad (2.10)$$

which will be further simplified in the following sections.

2.4 Scalar transport

The differential conservation law for the transport of a scalar variable, c, in laminar flow or static water is

$$\frac{D\,c}{D\,t} = D_{molec}\,\boldsymbol{\nabla}^2 c + S_c \quad , \qquad (2.11)$$

where D_{molec} is the molecular diffusivity of the scalar and S_c is a term representing the sum of all scalar sources and sinks. Equation (2.11) states the time rate of change for the concentration of a scalar in a control volume is balanced by the sum of all scalar sources and sinks and the net flux of the scalar through the control surface by advection and molecular diffusion.

When vegetation is present, a term representing mechanical diffusion must also be included in the equation. Mechanical diffusion arises from flow around physical obstructions and is illustrated in Figure 2.1. Particles advected by the mean flow are forced to one side by the obstructions, the vegetation stems in this case. For a sufficiently large number of particles and encounters with obstructions, the process acts similar to Fickian diffusion. Nepf (1999) showed that the mechanical diffusivity D_{mech} for vegetation modeled as rigid emergent cylinders can be quantified by

$$D_{mech} = \frac{\vartheta^2}{2}\lambda||\mathbf{u}||d_c^2 \quad , \qquad (2.12)$$

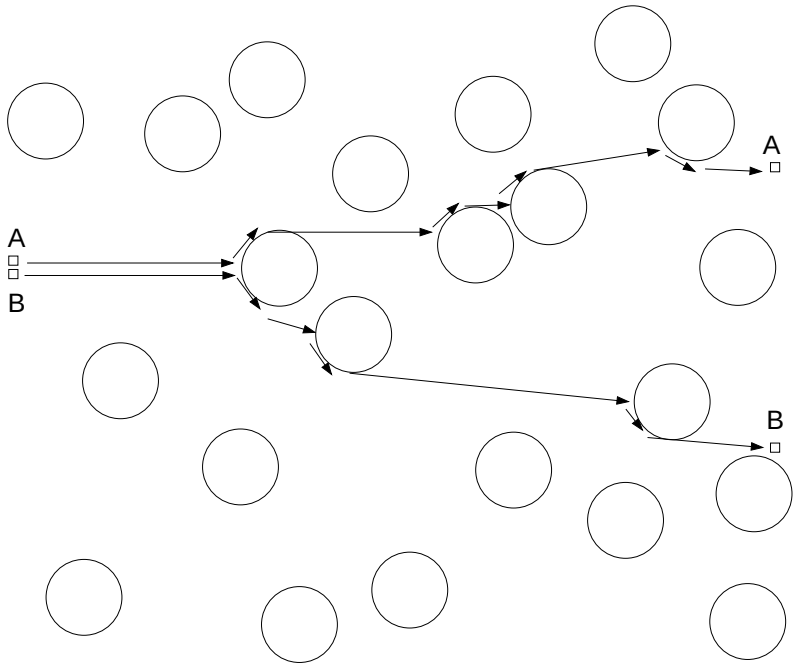

Figure 2.1: Illustration of mechanical diffusion caused by flow through vegetation (modified from Nepf (1999))

where ϑ is an scaling factor on the order of one which represents the departure of the vegetation stems from perfect cylinders. Since the molecular and mechanical diffusion processes act independent of one another, their contributions to the total diffusivity are additive and the scalar transport equation can be rewritten as

$$\frac{D\,c}{D\,t} = (D_{molec} + \frac{\vartheta^2}{2}\lambda||\mathbf{u}||d_c^2)\,\boldsymbol{\nabla}^2 c + S_c \quad . \tag{2.13}$$

In general, c can represent the concentration of any scalar variable that is transported with the flow. If the scalar is conservative, S_c will be identically equal to zero. If c is taken to represent temperature, the source-sink term will be

$$S_c = \frac{1}{\rho_0\,c_p}\frac{\partial\,I}{\partial\,z} \quad ,$$

where c_p is the specific heat of water and I is the downward solar irradiance (Rueda, 2001). The term $\frac{\partial\,I}{\partial\,z}$ is the only non-zero term in the divergence of the downward irradiance and represents the amount of short wave radiation absorbed at each depth layer (Rueda, 2001). Other components of the heat budget such as long wave radiative, evaporative, and sensible

heat fluxes are included in the formulation of the boundary condition at the free surface (Rueda, 2001). The variable c could also be taken to represent a general non-conservative chemical that is degraded by environmental reactions. The source and sink term may be modeled as a net reaction rate assuming first order kinetics

$$S_c = -k_r c \quad ,$$

where k_r represents the sum of all environmental degradation reactions, including photolysis, hydrolysis, oxidation, reduction, and biodegradation (Chapra, 1997). The physical processes of point loading, settling, resuspension, volatilization, and atmospheric deposition would be included in the formulation of the boundary conditions.

2.5 Ensemble-averaged equations

Fluid motion in floodplains, like the vast majority of flows encountered in natural environments, is turbulent. Turbulent flows are classified by irregular and chaotic motions, rapid rates of dissipation, and a wide range of time and length scales for eddy motion. The smallest of these length scales, known as the Kolmogorov microscale, is given by

$$\xi = \left(\frac{\nu^3}{\varepsilon}\right)^{1/4} \quad ,$$

where ε is the rate of turbulent kinetic energy dissipation. For the open ocean, ξ is on the order of millimeters (Kundu & Cohen, 2004). In natural freshwater floodplains, the rate of energy dissipation is expected to be greater than that of the open ocean, and thus the length scale of the smallest eddies should be much smaller. Predicting fluid motions at such small scales is computationally impractical and often unnecessary. For these reasons, equations (2.4), (2.10), and (2.13) are commonly averaged and reformatted in terms of the mean flow variables. For a generalized variable $\Phi(t)$, a time averaged quantity $\dot{\Phi}(t)$ can be defined using

$$\dot{\Phi}(t) \equiv \lim_{t_0 \to \infty} \frac{1}{t_0} \int_t^{t+t_0} \Phi(t)\, dt \quad ,$$

where t_0 is a time scale that is large compared to the time scale of turbulent fluctuations. If Φ varies significantly with time, a practical selection of t_0 is difficult and must additionally be small compared to the time scale at which the mean flow is varying. In these situations,

ergotic flow is often assumed and a discrete version of the previous equation can be written using the ensemble average of $\Phi(t)$ as

$$\overline{\Phi(t)} \equiv \frac{1}{N}\sum_{n=1}^{N}\Phi_n(t) \quad ,$$

where $\Phi_n(t)$ is the value of $\Phi(t)$ for the n^{th} trial in a set of experiments performed under an identical set of environmental conditions and N is a sufficiently large number (Kundu & Cohen, 2004). Using this representation, the instantaneous velocities, pressures, and densities can be decomposed into fluctuating and ensemble-averaged quantities,

$$\mathbf{u} = \overline{\mathbf{u}} + \mathbf{u}' \quad , \quad P = \overline{P} + P' \quad , \quad \rho = \overline{\rho} + \rho' \quad . \tag{2.14}$$

The ensemble average of the fluctuating quantity is identically zero since

$$\overline{\mathbf{u} = \overline{\mathbf{u}} + \mathbf{u}'} \quad \Rightarrow \quad \overline{\mathbf{u}} = \overline{\mathbf{u}} + \overline{\mathbf{u}'} \quad \Rightarrow \quad \overline{\mathbf{u}'} = 0 \quad .$$

Substituting (2.14) into the continuity equation and ensemble-averaging yields

$$\boldsymbol{\nabla}\cdot\overline{\mathbf{u}} = 0 \quad . \tag{2.15}$$

For the momentum equation (2.10), the ensemble-averaging of the individual terms, excluding the drag force, produces

$$\overline{\rho_0\frac{\partial\,\mathbf{u}}{\partial\,t}} = \rho_0\frac{\partial\,\overline{\mathbf{u}}}{\partial\,t} \qquad \overline{\rho_0(\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u}} = \rho_0(\overline{\mathbf{u}}\cdot\boldsymbol{\nabla})\overline{\mathbf{u}} + \rho_0\boldsymbol{\nabla}\cdot(\overline{\mathbf{u}'\mathbf{u}'})$$

$$\overline{-\boldsymbol{\nabla}P} = -\boldsymbol{\nabla}\overline{P} \qquad \overline{\rho\mathbf{g}} = \overline{\rho}\mathbf{g}$$

$$\overline{\mu\boldsymbol{\nabla}^2\mathbf{u}} = \mu\boldsymbol{\nabla}^2\overline{\mathbf{u}} \qquad \overline{-2\,\rho_0\,\boldsymbol{\Omega}\times\mathbf{u}} = -2\,\rho_0\,\boldsymbol{\Omega}\times\overline{\mathbf{u}}$$

where $\overline{\mathbf{u}'\mathbf{u}'}$ is the second order tensor

$$\begin{pmatrix} \overline{u'u'} & \overline{u'v'} & \overline{u'w'} \\ \overline{u'v'} & \overline{v'v'} & \overline{v'w'} \\ \overline{u'w'} & \overline{v'w'} & \overline{w'w'} \end{pmatrix}$$

with entries that are nonzero because of correlations between fluctuating velocity components. When multiplied by density, $\overline{\mathbf{u}'\mathbf{u}'}$ is known as the Reynolds stress tensor (Kundu & Cohen, 2004). Although there are notable discrepancies between eddy behavior and molecular behavior, this tensor is treated analogously to molecular viscosity and represents

a momentum flux associated with the turbulent fluctuations (Smith, 1997). Analytically accounting for these unknown terms is known as turbulence closure and will be addressed in Section 2.7.

Rigorous ensemble-averaging of the drag force term is typically not done because of the many unknown correlations that exist between fluctuating drag force variables. Both the vegetative momentum absorbing area λ and the drag coefficient C_D are known to be functions of velocity and will have fluctuating quantities associated with them (Fathi-Moghadam & Kouwen, 1997). The ensemble-average of the drag force will therefore have the following terms

$$\begin{aligned}\overline{\frac{1}{2}\,\rho_0\,\lambda\,C_D\,\mathbf{u}||\mathbf{u}||} &= \frac{1}{2}\rho_0\,\overline{(\overline{\lambda}+\lambda')(\overline{C_D}+C_D')(\overline{\mathbf{u}}+\mathbf{u}')(\overline{||\mathbf{u}||}+||\mathbf{u}||')} \\ &= \frac{1}{2}\rho_0\Big(\overline{\lambda}\;\overline{C_D}\;\overline{\mathbf{u}}\;\overline{||\mathbf{u}||} \;+\; \overline{\lambda}\;\overline{C_D}\;\overline{\mathbf{u}'||\mathbf{u}||'} \;+\; \overline{\lambda}\;\overline{C_D'\mathbf{u}'}\;\overline{||\mathbf{u}||} \\ &\quad + \overline{\lambda' C_D'}\;\overline{\mathbf{u}}\;\overline{||\mathbf{u}||} \;+\; \cdots \;+\; \overline{\lambda' C_D'\mathbf{u}'||\mathbf{u}||'}\Big) \quad .\end{aligned} \tag{2.16}$$

Little information is known about these higher order correlations and specific closure methods for the terms in (2.16) have not been well developed. These terms may be thought of as dispersive momentum fluxes due to turbulent flow through vegetation and have a structure similar to the Reynolds stresses. The magnitude of one of the turbulent drag terms ($\frac{1}{2}\rho_0\overline{\lambda}\;\overline{C_D}\;\overline{\mathbf{u}'||\mathbf{u}||'}$) relative to the Reynolds stress force can be examined using a scaling analysis.

$$\frac{\frac{1}{2}\rho_0\overline{\lambda}\;\overline{C_D}\;\overline{\mathbf{u}'||\mathbf{u}||'}}{\boldsymbol{\nabla}\cdot(\rho_0\overline{\mathbf{u}'\mathbf{u}'})} \sim \frac{\frac{1}{2}\rho_0\lambda\,C_D\,H\;\overline{U_c'U_c'}}{\frac{H}{L}\rho_0\overline{U_c'U_c'}} = \frac{1}{2}\,\frac{L}{H}\,(C_D\,\lambda\,H)$$

For a range of experiments, Fathi-Moghadam & Kouwen (1997) found the dimensionless parameter $C_D\,\lambda\,H$, where H is a representative vertical length scale, to have a range of approximately 0.4-2.0. Since floodplain horizontal length scales are much greater than the vertical scale, the turbulent drag stress terms will therefore be significant. Because of the similarity of turbulent drag terms to the Reynolds stresses, the effects of vegetation on turbulence are typically absorbed into the turbulence closure scheme for the Reynolds stress terms. Specific methods for this will be addressed in Section 2.7 and the turbulent drag terms are dropped in the remaining analysis for simplicity.

Since turbulent stresses are almost always much larger than laminar stresses in floodplain environments, it is common to neglect the laminar stress terms (Jia *et al.*, 2002; Smith,

1997). The complete ensemble-averaged momentum equation can thus be written as

$$\rho_0 \frac{D\,\overline{\mathbf{u}}}{D\,t} = -\boldsymbol{\nabla}\overline{P} + \overline{\rho}\mathbf{g} - \boldsymbol{\nabla}\cdot(\rho_0\overline{\mathbf{u}'\mathbf{u}'}) - 2\,\rho_0\,\boldsymbol{\Omega}\times\overline{\mathbf{u}} - \frac{1}{2}\,\rho_0\,\lambda\,C_D\,\overline{\mathbf{u}}\,\overline{||\mathbf{u}||} \quad . \tag{2.17}$$

Similarly, ensemble-averaging the scalar transport equation and neglecting molecular and mechanical diffusion terms produces

$$\frac{D\,\overline{c}}{D\,t} = -\boldsymbol{\nabla}\cdot\overline{\mathbf{u}'c'} + S_c \quad , \tag{2.18}$$

where $\overline{\mathbf{u}'c'} = (\overline{u'c'}, \overline{v'c'}, \overline{w'c'})$ represents a mass flux of c due to turbulent fluctuations. Molecular diffusion terms can safely be ignored because they are much smaller than turbulent diffusion terms for most environmental flows (Smith, 1997). The effect of mechanical diffusion in floodplain environments is also very small when compared to turbulent diffusion, provided the vegetation density is not excessively high (Nepf, 1999). Field measurements in estuarine environments by Nepf (1999) have shown accurate simulation of diffusion processes without the inclusion of mechanical diffusion terms. It is also useful to neglect these terms because of the data requirement burden imposed by their evaluation. Computational storage of large arrays of vegetation density and stem diameters will substantially increase the resources required to run model simulations, and hand measurements of these variables for large domains is impractical. Although spatial distributions of vegetation density and stem diameters may be obtained with the use of remote sensing techniques such as lidar, they represent derived quantities, and methods used to convert remotely sensed quantities to vegetation densities may introduce more error into the model than the inclusion of mechanical diffusion will correct.

2.6 Depth-averaged equations and boundary conditions

Floodplain hydrodynamic models can be one-, two-, or three-dimensional. The selection of a specific type is largely based on the expected application. One-dimensional floodplain models are commonly used to predict downstream water elevations in response to an upstream flooding event. If lateral variations in floodplain flow variables such as velocity or residence time are of interest, a two-dimensional depth-averaged floodplain model may be used. If variation in the vertical direction is also of interest, for example in assessing

fish spawning habitat or vegetative seed dispersal, a three-dimensional floodplain model may be the best choice. The most common floodplain model in use today is the two-dimensional, depth-averaged type (Nicholas & McLelland, 2004). On a typical floodplain, horizontal scales of fluid motion exceed vertical scales by at least two orders of magnitude. Vertical velocities and vertical variations in lateral velocities and scalar concentrations are commonly neglected to create simplified equations that can be numerically solved in an efficient manner. Although recent three-dimensional simulations have pointed to inadequacies of two-dimensional models representing surface roughness and turbulence, two-dimensional models do not suffer from the acute difficulties of representing boundary conditions for topographically complex floodplain environments (Nicholas & McLelland, 2004). Additionally, many of the computational issues associated with advancing and retreating wetting fronts and very small water depths on floodplains are absent in two-dimensional models. For these reasons, a two-dimensional depth-averaged floodplain model is considered herein.

2.6.1 Continuity equation

The water surface elevation, ζ (measured positively upward), and the location of the floodplain bottom, d (measured positively downward), are defined relative to an arbitrary datum as shown in Figure 2.2. The total water depth h is defined as the sum of ζ and d. Using these definitions and with overbars dropped for simplicity, (2.15) can be depth-averaged as follows

$$\frac{1}{h}\int_{-d}^{\zeta}\frac{\partial u}{\partial x}\,dz \;+\; \frac{1}{h}\int_{-d}^{\zeta}\frac{\partial v}{\partial y}\,dz \;+\; \frac{1}{h}\int_{-d}^{\zeta}\frac{\partial w}{\partial z}\,dz \;=\; 0 \quad . \tag{2.19}$$

The above equation is simplified by employing Liebnitz' rule, shown below for a generalized function $f(x,z)$

$$\int_{a(x)}^{b(x)}\frac{\partial f}{\partial x}\,dz = \frac{\partial}{\partial x}\int_{a(x)}^{b(x)} f\,dz + [f]_{a(x)}\frac{\partial a(x)}{\partial x} - [f]_{b(x)}\frac{\partial b(x)}{\partial x} \tag{2.20}$$

and the kinematic free surface and bottom boundary conditions which result from a mass balance on a differential element at the boundary

$$\frac{\partial \zeta}{\partial t} + u\frac{\partial \zeta}{\partial x} + v\frac{\partial \zeta}{\partial y} \;=\; w + P_r - ET \qquad \text{at } z=\zeta \tag{2.21}$$

$$u\frac{\partial d}{\partial x} + v\frac{\partial d}{\partial y} \;=\; -w \pm GW \qquad \text{at } z=-d \quad , \tag{2.22}$$

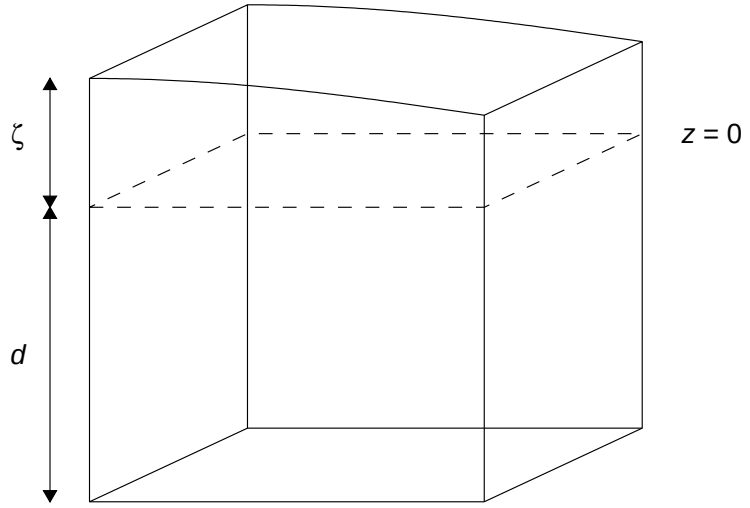

Figure 2.2: Definition sketch for water depth variables

where P_r, ET, and GW represent rates of precipitation, evapotranspiration, and groundwater flow, respectively (Smith, 1997). The $\pm$ in front of the groundwater term in (2.22) is used to indicate the potential loss or gain of water to the floodplain due to groundwater flow. It is common to neglect water fluxes from the free surface and bottom boundaries because simulated flood events typically last a maximum of several days, and the boundary flux terms rarely account for more than a few millimeters of water per day (Bates *et al.*, 1992; Nicholas & Mitchell, 2003; Connell *et al.*, 2001). However, in seasonal flooding of large river systems, water may remain on the floodplain for a few months and evaporation, precipitation, and groundwater flow may substantially influence water levels. These terms thus are retained for model robustness.

Equation (2.19) can now be simplified by use of (2.20)-(2.22) to

$$\frac{\partial \zeta}{\partial t} + \frac{\partial}{\partial x}\int_{-d}^{\zeta} u\, dz + \frac{\partial}{\partial x}\int_{-d}^{\zeta} v\, dz = P_r - ET \pm GW \quad ,$$

and by defining depth-averaged lateral velocities

$$U \equiv \frac{1}{h}\int_{-d}^{\zeta} u\, dz \quad , \quad V \equiv \frac{1}{h}\int_{-d}^{\zeta} v\, dz \quad ,$$

the following depth-averaged continuity equation is obtained.

$$\frac{\partial \zeta}{\partial t} + \frac{\partial (hU)}{\partial x} + \frac{\partial (hV)}{\partial y} = P_r - ET \pm GW \tag{2.23}$$

Of the three terms on the right hand side of (2.23), only the precipitation rate P_r can easily be measured or estimated from meteorological data collected near the floodplain. Evapotranspiration and groundwater flow are much harder to predict. Evapotranspiration is the combination of two physically separate processes: evaporation and transpiration. Evaporation is the process by which water at the free surface is transformed into the vapor state and lost to the atmosphere. It may be roughly estimated by the product of a standard pan evaporation rate and a correction coefficient that is around 0.7 for areas in the United States, or can be more accurately predicted by use of the aerodynamic method

$$E_v = 0.622\, C_{Ev}\, W_z \frac{\rho_{air}}{\rho_0} \left(\frac{e_{sat,w} - e_a}{P_{atm}} \right) \quad , \tag{2.24}$$

where

$$\begin{aligned}
E_v &= \text{evaporation rate in units of length per time,} \\
\rho_{air} &= \text{density of air,} \\
W_z &= \text{wind speed measured at a height } z \text{ from the free surface,} \\
C_{Ev} &= \text{coefficient with values between } 1.15 \times 10^{-3} \text{and } 1.4 \times 10^{-3}, \\
e_{sat,w} &= \text{saturation vapor pressure of air using the surface water temperature,} \\
e_a &= \text{vapor pressure of air at a height } z \text{ from the free surface, and} \\
P_{atm} &= \text{atmospheric pressure.}
\end{aligned}$$

(Bedient & Huber, 2002; Gupta, 2001). The saturation vapor pressure of air in millibars can be calculated for any temperature T with the Magnes-Tetens formula (Tennessee Valley Authority, 1972)

$$e_{sat} = \exp\left(2.3026 \left(\frac{7.5T}{T + 237.3} + 0.7858\right)\right) \quad , \tag{2.25}$$

and the value of e_a can be calculated using relative humidity (RH) and the saturation vapor pressure using the air temperature at a height z from the free surface as

$$e_a = \frac{RH}{100}\, e_{sat,a} \quad . \tag{2.26}$$

The wind speed, air temperature, and relative humidity are common meteorological values that can be directly measured or interpolated from nearby records. However, without knowledge of the spatial distribution of surface water temperatures, the $e_{sat,w}$ parameter cannot be calculated accurately. Because of very shallow water, floodplain water temperatures can have diurnal fluctuations of greater than 10°C, resulting in $e_{sat,w}$ fluctuations of ±50% (Malard *et al.*, 2001). Nevertheless, the aerodynamic approach is favored over the pan evaporation approach because of its dependence on wind speed and humidity and will be adopted as the method for this model.

Transpiration is the process whereby water is taken up by vegetation and lost to the atmosphere via evaporation from plant stomata (Bedient & Huber, 2002). It may play a significant role in the water balance for heavily vegetated floodplains during the growing season. Transpiration is highly variable and depends on vegetation type and density, and the time of year. Since spatial patterns of vegetation type and density are difficult to obtain, transpiration is often assumed to be a constant fraction of evaporation and the two processes are grouped into one term. A small modification to this scheme is proposed here to account for the fact that vegetation will be unable to transpire when it is submerged. If it is assumed that transpiring plant area is distributed evenly across the height of the plant, then total ET may be written as

$$ET = E_v + C_{Tr} \max\left(0, 1 - \frac{h}{h_{veg}}\right) E_v \quad , \tag{2.27}$$

where C_{Tr} is an empirical transpiration coefficient and h_{veg} is the height of the vegetation. This assumption is more likely to be accurate if most of the floodplain vegetation is grasses and small shrubs and not large floodplain trees.

The final term on the right hand side of (2.23) is groundwater flow. Although it is possible to accurately predict this quantity with Darcy's Law for flow through porous media, the use of this equation is constrained by the need for spatial distributions of soil type and hydraulic conductivity values and the location of the groundwater table. In the absence of these values, a constant rate of seepage or groundwater inflow will be prescribed in this model formulation.

Finally, it should be mentioned that the boundary conditions for the continuity equation assume that bottom topography is fixed. In actuality, the large flow rates associated with

flooding events will transport and rework much of the bottom sediment in the channel and may deposit or remove sediment from the floodplain (Florsheim & Mount, 2002). However, simplifications are necessary to create an efficient computer model, and it is assumed here that the extent of sediment transport in a single flood event is not great enough to influence subsequent flow patterns within the event.

2.6.2 Momentum equation

Prior to depth-averaging the momentum equation, it is common to employ the hydrostatic approximation whereby vertical velocities and accelerations are neglected as small compared to the pressure and gravity terms in the z-momentum equation. This assumption is accurate in shallow water environments without large changes in topography over small distances and is commonly applied in floodplain models (Casulli & Stelling, 1998; Nicholas & Mitchell, 2003). Using this simplification, (2.17) can be written in scalar form as

$$\begin{aligned}
\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} &= -\frac{1}{\rho_0}\frac{\partial P}{\partial x} - \left(\frac{\partial \overline{u'u'}}{\partial x} + \frac{\partial \overline{u'v'}}{\partial y} + \frac{\partial \overline{u'w'}}{\partial z}\right) \qquad (2.28)\\
&\quad + f_c v - \frac{1}{2} C_D \lambda u \sqrt{u^2 + v^2 + w^2}\\
\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} &= -\frac{1}{\rho_0}\frac{\partial P}{\partial y} - \left(\frac{\partial \overline{v'u'}}{\partial x} + \frac{\partial \overline{v'v'}}{\partial y} + \frac{\partial \overline{v'w'}}{\partial z}\right) \qquad (2.29)\\
&\quad - f_c u - \frac{1}{2} C_D \lambda v \sqrt{u^2 + v^2 + w^2}\\
\frac{\partial P}{\partial z} &= -\rho_0 g \quad .
\end{aligned}$$

The Coriolis and pressure gradient terms in (2.28) can be easily depth-averaged to yield

$$\frac{1}{h}\int_{-d}^{\zeta} f_c v \, dz = f_c V \qquad \frac{1}{h}\int_{-d}^{\zeta} -\frac{1}{\rho_0}\frac{\partial P}{\partial x} dz = -g\frac{\partial \zeta}{\partial x}$$

where the hydrostatic equation has been utilized to simplify the pressure term. Depth-integration of the turbulent stress terms is accomplished by using the dynamic free surface and bottom boundary conditions

$$\begin{aligned}
\frac{\tau_{xs}}{\rho_0} &= \overline{u'u'}\frac{\partial \zeta}{\partial x} + \overline{u'v'}\frac{\partial \zeta}{\partial y} - \overline{u'w'} \qquad \text{at } z = \zeta\\
\frac{\tau_{xb}}{\rho_0} &= \overline{u'u'}\frac{\partial d}{\partial x} + \overline{u'v'}\frac{\partial d}{\partial y} - \overline{u'w'} \qquad \text{at } z = -d \quad ,
\end{aligned}$$

which are obtained via a differential force balance at the boundaries and where τ_{xs} and τ_{xb} are x-components of the free surface and bottom boundary shear stresses, respectively

(Smith, 1997). The depth-averaged turbulent stress terms can then be reduced to

$$\frac{1}{h}\int_{-d}^{\zeta} -\left(\frac{\partial\,\overline{u'u'}}{\partial\,x}+\frac{\partial\,\overline{u'v'}}{\partial\,y}+\frac{\partial\,\overline{u'w'}}{\partial\,z}\right)\,dz=\frac{\tau_{xs}-\tau_{xb}}{\rho_0 h}+\frac{1}{\rho_0 h}\left(\frac{\partial\,(h\tau_{xx})}{\partial\,x}+\frac{\partial\,(h\tau_{xy})}{\partial\,y}\right)$$

where variables of the form τ_{xx} are defined as the average Reynold stresses over the depth as follows

$$\frac{\tau_{xx}}{\rho_0}\equiv\frac{1}{h}\int_{-d}^{\zeta}-\overline{u'u'}\,dz \quad .$$

The acceleration terms on the left hand side of (2.28) can be similarly depth-averaged with the use of the kinematic free surface and bottom boundary conditions to yield

$$\begin{aligned}\frac{1}{h}\int_{-d}^{\zeta}\left(\frac{\partial\,u}{\partial\,t}+u\,\frac{\partial\,u}{\partial\,x}+v\,\frac{\partial\,u}{\partial\,y}\right)\,dz \;&=\; \frac{\partial\,(hU)}{\partial\,t}+\frac{1}{h}\int_{-d}^{\zeta}u^2\,dz+\frac{1}{h}\int_{-d}^{\zeta}uv\,dz\\ &\quad -\frac{[u]_{\zeta}}{h}\,(P_r-ET)-\frac{[u]_{-d}}{h}\,(\pm GW)\,.\end{aligned}$$

If a universal vertical velocity profile is assumed, the dependence of u and v on z can be removed by multiplying by an appropriate shape factor of the form $f(\varphi)$ where $\varphi = z/h$ is a scaled depth (García, 1996). Using this formulation, the depth-averaged acceleration terms reduce to

$$\begin{aligned}\frac{1}{h}\frac{\partial\,(hU)}{\partial\,t}+\frac{1}{h}\frac{\partial\,(hU^2)}{\partial\,x}\int_{-d}^{\zeta}f(\varphi)^2\,dz \;&+\; \frac{1}{h}\frac{\partial\,(hUV)}{\partial\,y}\int_{-d}^{\zeta}f(\varphi)^2\,dz\\ &\quad -\frac{[u]_{\zeta}}{h}\,(P_r-ET)-\frac{[u]_{-d}}{h}\,(\pm GW)\,.\end{aligned}\tag{2.30}$$

The integral of the squared shape factor, is known as the momentum correction coefficient or Boussinesq coefficient and is commonly denoted β (Chanson, 2004). If the velocity profile is assumed to have the form given by Prandtl's power law, $U/U_{max} = \varphi^{1/N}$, then the momentum coefficient can be calculated with

$$\beta=\frac{(N+1)^2}{N(N+2)}$$

(Chanson, 2004). For common turbulent velocity profiles ($N = 6, 7$) the momentum coefficients only represent 1.6% or 2.1% corrections. However, floodplain velocity profiles are often influenced by wind and flow over vegetation. The effects of vegetation are complicated and change with water depth, as shown in Figure 2.3. Wind is another factor that may change the velocity profile near the free surface (Menendez & Laciana, 2006). It is therefore difficult to prescribe an accurate shape factor for floodplain flows and the momentum

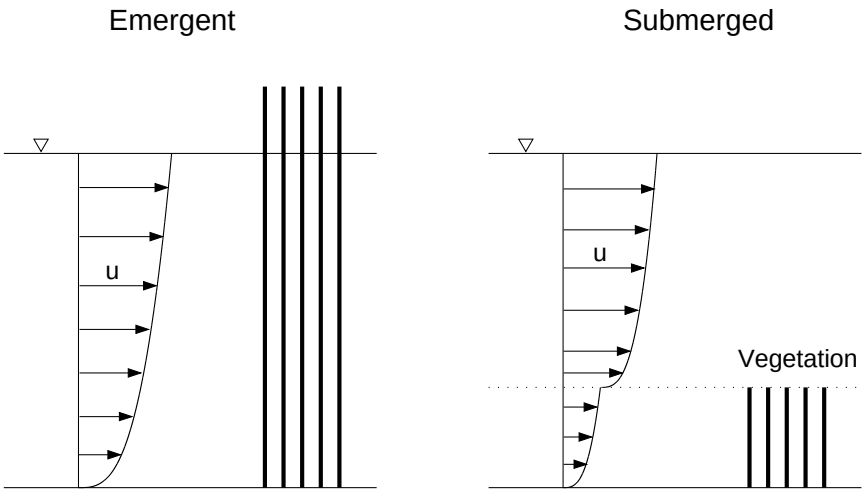

Figure 2.3: Schematic of the effects of submerged and emergent vegetation on vertical velocity profiles (modified from Wu *et al.* (2005))

coefficient is assumed identically equal to one. This simplification is fortunate because it allows the depth-averaged continuity equation (2.23) to be factored out of (2.30) and the acceleration terms can be expressed as

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + V\frac{\partial U}{\partial y} + \underbrace{\left(\frac{U}{h} - \frac{[u]_{\zeta}}{h}\right)(P_r - ET) + \left(\frac{U}{h} - \frac{[u]_{-d}}{h}\right)(\pm GW)} \quad .$$

The terms indicated with an underbrace in the above equation are commonly disregarded as insignificant in comparison to the first three terms. Disregarding these terms is equivalent to ignoring the influence of precipitation, evapotranspiration, and groundwater flow on the mean fluid momentum and is generally a good assumption.

The nonlinear drag force term is depth-integrated in a manner similar to the nonlinear acceleration terms. The drag coefficient, C_D, and the momentum absorbing area, λ, are strictly functions of z. However, little information is available concerning the vertical distributions of either term and they are assumed constant with z for this analysis. A later parameterization of the drag force term below will account for decreases in momentum absorbing area and drag force with increases in depth. As a result of the nonlinear velocity

terms, another momentum correction coefficient will be introduced in the integration and is assumed to be identically equal to one. The depth-averaged drag force term is then

$$\frac{1}{2}\,\lambda\, C_D\, U\,\sqrt{U^2+V^2} \quad . \tag{2.31}$$

Integration of the y-momentum equation (2.29) proceeds similarly, and the depth-averaged momentum equations can finally be written as

$$\begin{aligned}
\frac{\partial U}{\partial t}+U\frac{\partial U}{\partial x}+V\frac{\partial U}{\partial y} &= -g\frac{\partial \zeta}{\partial x}+\frac{\tau_{xs}-\tau_{xb}}{\rho_0 h}+\frac{1}{\rho_0 h}\left(\frac{\partial\,(h\tau_{xx})}{\partial\, x}+\frac{\partial\,(h\tau_{xy})}{\partial\, y}\right) \\
&\quad +f_c V-\frac{1}{2}\,C_D\,\lambda U\sqrt{U^2+V^2} \quad , \qquad (2.32)\\
\frac{\partial V}{\partial t}+U\frac{\partial V}{\partial x}+V\frac{\partial V}{\partial y} &= -g\frac{\partial \zeta}{\partial y}+\frac{\tau_{ys}-\tau_{yb}}{\rho_0 h}+\frac{1}{\rho_0 h}\left(\frac{\partial\,(h\tau_{yx})}{\partial\, x}+\frac{\partial\,(h\tau_{yy})}{\partial\, y}\right) \\
&\quad -f_c U-\frac{1}{2}\,C_D\,\lambda V\sqrt{U^2+V^2} \quad . \qquad (2.33)
\end{aligned}$$

The shear stresses at the free surface due to wind are commonly parameterized using a quadratic stress law (Smith, 1997) as

$$\tau_{xs}=C_w\,\rho_{air}\,(W_{10})^2\,\sin(\Theta) \qquad \tau_{ys}=C_w\,\rho_{air}\,(W_{10})^2\,\cos(\Theta) \tag{2.34}$$

where ρ_{air} is the density of air, W_{10} is the wind speed measured 10 m above the free surface, Θ is the angle between the wind and the positive y-axis (measured in the clockwise direction from the y-axis to the wind vector), and C_w is the wind drag coefficient. Although C_w is a function of wind speed and wave height, it is customarily approximated as 1.5×10^{-3} (Smith, 1997).

The shear stresses at the bottom boundary can be determined using the law of the wall

$$\frac{\sqrt{U^2+V^2}}{u_*}=\frac{1}{\kappa}\ln\left(\frac{z'}{z_0}\right) \quad , \quad u_*\equiv\sqrt{\frac{\tau_b}{\rho_0}} \quad ,$$

where

$$\begin{aligned}
u_* &= \text{shear velocity,}\\
\kappa &= \text{von Karman constant} = 0.41,\\
z' &= \text{vertical distance measured up from the roughness elements, and}\\
z_0 &= \text{average height of the roughness elements.}
\end{aligned}$$

(Kundu & Cohen, 2004; Smith, 1997). This equation can be rearranged for the desired τ_b values

$$\tau_{xb} = \rho_0 \, c_f \, U \sqrt{U^2 + V^2} \quad , \quad \tau_{yb} = \rho_0 \, c_f \, V \sqrt{U^2 + V^2} \quad , \quad c_f \equiv \frac{\kappa^2}{\ln\left(\frac{z'}{z_0}\right)^2} \quad , \tag{2.35}$$

and c_f can be parameterized using either the Chézy coefficient (C_z), the Darcy-Weisbach friction factor (f_D), or the Manning roughness coefficient (n_M). Although the Chézy, Darcy-Weisbach, and Manning parameterizations are strictly valid only for uniform equilibrium flows, it is assumed that the model time step and spatial grid size are small enough so this is approximately true. The bottom shear stress can then be written using any of the three coefficients as

$$\frac{\tau_{xb}}{\rho_0 h} = \frac{g}{C_z^2} \frac{U \sqrt{U^2 + V^2}}{h} = \frac{f_D}{8} \frac{U \sqrt{U^2 + V^2}}{h} = \frac{g \, n_M^2}{k_M^2 \, h^{1/3}} \frac{U \sqrt{U^2 + V^2}}{h} \quad , \tag{2.36}$$

where k_M is a coefficient which will equal 1.0 when using SI units and 1.49 when Imperial units are used. The similarity between the depth-averaged drag force term (2.31) and the parameterization of bottom friction in (2.35) has led to the use of the relationships in (2.36) to parameterize the drag force term. While bottom shear stress and drag due to flow through vegetation are separate physical processes, they are often grouped together because of the similarity of terms and the difficulty in determining the momentum absorbing area, λ, and the vegetation drag coefficient, C_D. If the terms are treated as one, empirical studies can be used to determine the dependence of the effective frictional coefficient on flow depths, velocities, and vegetation types.

In the current model formulation, a menu approach will be taken whereby a different equation set is used to calculate effective frictional coefficients depending on the type of vegetation in the subregion (Mason *et al.*, 2003). The particular method described below was first proposed by Darby (1999) and has been shown to effectively simulate floodplain flows by Mason *et al.* (2003).

Channel

Regions within the channel banks are typically unvegetated, and bottom stress is assumed to be a function only of bottom roughness. There have been many formulas proposed to

estimate roughness coefficients based on channel characteristics. For gravel-bedded streams, Henderson (1996) proposed the relation

$$n_M = 0.038(d_{75})^{1/6} \quad ,$$

where d_{75} is the 75^{th} percentile grain size (in meters) which can be determined with a grain size analysis. These Strickler-type equations are useful, but can be inaccurate above $n_M = 0.02$ (Bombardelli & García, 2003). If information is also available for estimation of the channel's hydraulic radius, $R_h = A_c/P_w$ where A_c and P_w are the channel area and wetted perimeter at bankful depth, the equation proposed by Hey (1979) may be used

$$\frac{1}{\sqrt{f_D}} = 2.03 \log\left(\frac{a_s R_h}{3.5\, d_{84}}\right) \quad , \quad a_s = 11.1\left(\frac{R_h}{h}\right)^{-0.314} \quad , \tag{2.37}$$

where a_s is a dimensionless shape correction factor (Mason *et al.*, 2003).

Short vegetation

Short vegetation is defined as vegetation with heights less than 1.2 m. Vegetation in this class is usually composed of grass and sedge species and is assumed to be flexible (Mason *et al.*, 2003). Kouwen & Li (1980) have shown the deflected heights to be descibed by

$$h^d_{veg} = 0.14\, h_{veg}\left(\frac{\left(ME_yI_z/\tau_b\right)^{0.25}}{h_{veg}}\right)^{1.59} \quad , \tag{2.38}$$

where

$$\begin{aligned} h^d_{veg} &= \text{deflected height of the vegetation,} \\ h_{veg} &= \text{pre-inundation height of the vegetation,} \\ ME_yI_z &= \text{product of the stem density and flexural rigidity of the vegetation.} \end{aligned}$$

Since individual values for the stem density (M), the modulus of elasticity of the stems (E_y), and the stem area's second moment of inertia (I_z) are difficult to measure, the terms are typically grouped together. Work by Kouwen (1988) on grasses showed these values to be correlated with vegetation height and time of year by

$$ME_yI_z = \begin{cases} 319\,(h_{veg})^{3.3} & \text{for growning grasses,} \\ 25.4\,(h_{veg})^{2.26} & \text{for dormant grasses.} \end{cases} \tag{2.39}$$

The Darcy-Weisbach friction factor can then be estimated for short vegetation as

$$\frac{1}{\sqrt{f_D}} = a + b \log\left(\frac{h}{h_{veg}^d}\right) \quad , \tag{2.40}$$

where the coefficients a and b have empirically determined values depending on the ratio of the shear velocity, u_* to the critical shear velocity, u_{*crit}, as shown in Table 2.1. The values for critical shear velocity were shown by Kouwen & Li (1980) to be described by

$$u_{*crit} = \min\left(0.028 + 6.33\, ME_y I_z,\ 0.23\, ME_y I_z^{0.106}\right) \quad . \tag{2.41}$$

Since equations (2.38)-(2.41) are highly nonlinear and include a dependence on τ_b, they must be calculated iteratively or by simply using flow values from the previous time step. Mason *et al.* (2003) presented results using the latter method, with the value of τ_b calculated for the first time step as $\tau_b = \rho_0\, h\, S_0$ where S_0 is the local bottom slope of the floodplain. The above set of equations for short vegetation was derived for flow over submerged grasses. However, the same equation set can be applied to emergent grasses provided the deflected vegetation height, h_{veg}^d, is capped at the water depth, h.

Table 2.1: Coefficients for short vegetation friction factor calculation (modified from Mason *et al.* (2003))

Criteria	a	b
$u_*/u_{*crit} \leq 1.0$	0.15	1.85
$1.0 < u_*/u_{*crit} \leq 1.5$	0.20	2.70
$1.5 < u_*/u_{*crit} \leq 2.5$	0.28	3.08
$2.5 < u_*/u_{*crit}$	0.29	3.50

Tall and intermediate vegetation

Floodplain vegetation with undisturbed heights of greater than 1.2 meters is largely composed of shrub and tree species (Mason *et al.*, 2003). It is modeled with a separate equation which also takes into account vegetation flexibility and a reduction in momentum absorbing area based on flow velocity as

$$f_D = 4.06 \left(\frac{\sqrt{U^2 + V^2}}{\sqrt{\xi_v E_y/\rho_0}}\right)^{-0.46} \left(\frac{h}{h_{veg}}\right) \quad , \tag{2.42}$$

where E_y is the modulus of elasticity of the tall vegetation, and ξ_v is a coefficient to account for vegetation deformation. This equation is strictly valid only for emergent vegetation, but can be used with $h/h_{veg} = 1$ for $h > h_{veg}$. Unfortunately, values for E_y and ξ_v are not available for the entire range of tall and intermediate vegetation encountered on any particular floodplain. Kouwen & Fathi-Moghadam (2000) presented values for four coniferous tree species, as shown in Table 2.2. Mason *et al.* (2003) then justified treating all tall vegetation on the floodplain as having characteristics similar to white pines, for example, by reasoning that this method, which takes into account the flexibility and deflection of trees, will still be more accurate than any method that does not.

Table 2.2: Empirical coefficients for tall vegetation friction factor calculation (modified from Kouwen & Fathi-Moghadam (2000))

Species	$\xi_v E_y$ (N/m^2)
Cedar (*Thuja occidentalis*)	2.07
Spruce (*Picea glauca*)	3.36
White pine (*Pinus strobus*)	2.99
Austrian pine (*pinus palustris*)	4.54

Using this menu-based approach, the bottom shear stress and vegetation drag terms in (2.28) and (2.29) are accounted for using only the flow variables and the vegetation heights, a parameter that can be measured to a high degree of accuracy using such remote sensing techniques as lidar. Although this approach makes some assumptions, such as all tall vegetation being modeled as one species of coniferous trees and all short vegetation being modeled as grasses, it takes into account flow variables such as water velocity and depth and thus provides a more realistic representation of the physics of the flow than a constant frictional value which is either estimated by visual inspection of the field site or calibrated as a model coefficient.

2.6.3 Scalar transport equation

Term by term integration of the scalar transport equation is similar to the analysis already shown for the momentum equation and will not be presented here. In the integration, it is assumed that the system is well-mixed so that the concentration of the scalar is approximately uniform over the depth. Because of the shallow depths and relatively high turbulent mixing occuring on floodplains, this is generally a good assumption. The resulting depth-averaged scalar transport equation is

$$\frac{\partial C}{\partial t} + U\frac{\partial C}{\partial x} + V\frac{\partial C}{\partial y} = \frac{1}{\rho_0 h}\left(\frac{\partial (hJ_x)}{\partial x} + \frac{\partial (hJ_y)}{\partial y}\right) + S_c \quad , \tag{2.43}$$

where J_x and J_y are defined as the depth-averaged turbulent scalar fluxes

$$\frac{J_x}{\rho_0} \equiv \frac{1}{h}\int_{-d}^{\zeta} -\overline{u'c'}\, dz \quad , \quad \frac{J_y}{\rho_0} \equiv \frac{1}{h}\int_{-d}^{\zeta} -\overline{v'c'}\, dz$$

and C is the depth-averaged scalar concentration.

A final physical process that should be included in the scalar transport governing equation is shear dispersion due to the presence of a non-uniform vertical velocity profile. By having generally low water velocities near the bed in comparison to near the water surface, floodplain vertical velocity profiles may smear the lateral distribution of a tracer further than the distortion caused by turbulent diffusion would predict. An analytical treatment of this mechanism is given in detail in Rutherford (1994). By using G.I. Taylor's assumption that the lateral dispersive flux is proportional to the lateral gradient in the depth-averaged concentration, the final depth-averaged scalar transport equation can be written as

$$\frac{\partial C}{\partial t} + U\frac{\partial C}{\partial x} + V\frac{\partial C}{\partial y} = \frac{1}{h}\left(\frac{\partial}{\partial x}\left(h\frac{J_x}{\rho_0} + hk_x\frac{\partial C}{\partial x}\right) + \frac{\partial}{\partial y}\left(h\frac{J_y}{\rho_0} + hk_y\frac{\partial C}{\partial y}\right)\right) + S_c \quad , \tag{2.44}$$

where k_x and k_y are the dispersion coefficients.

It should be noted that integration of the scalar transport equation may have terms introduced due to boundary conditions specific to the particular scalar. If the scalar is taken to represent temperature, there will be an additional term on the right hand side of (2.43) that will have the form $q_s/\rho_0 h$ where q_s is the heat flux through the free surface (Rodi, 1980). If the scalar represents a chemical subjected to the physical processes of settling, resuspension, volatilization, and deposition, there will be terms introduced on the

right hand side that will resemble

$$-k_s C + k_{rs} C - k_v C + k_d C$$

if the processes are modeled as first order rates.

2.6.4 Lateral boundaries

At the lateral domain boundaries, it is assumed that there is no flow normal to the boundary except at defined locations where either the water depth or the fluid velocity is prescribed. Fluid velocities parallel to the boundary may be modeled with a quadratic stress law, for example (Smith, 1997). However for most environmental simulations, the grid discretization is much larger than the boundary layer in which the quadratic stress law is valid, and applying this condition produces distorted lateral velocities (Smith, 1997). In this model, the velocities and scalar transport normal to the side boundaries are assumed zero and no lateral frictional resistance is used in calculating velocities in adjacent computational cells.

2.7 Turbulence closure schemes

A turbulence closure scheme is necessary to relate the depth-averaged Reynolds stresses, τ_{xx}, τ_{xy}, τ_{yx}, and τ_{yy}, and the depth-averaged turbulent scalar fluxes, J_x and J_y, to the flow variables U, V, ζ, and C. In this manner, equations (2.23), (2.28), (2.29), and (2.43) can be formulated in terms of the four dependent variables to close the set of fluid motion equations. By analogy with molecular viscosity in laminar flow, the depth-averaged Reynolds stresses are assumed to be proportional to the mean velocity gradients (Rodi, 1980). This is known as the eddy viscosity concept and is written

$$\frac{\tau_{xx}}{\rho_0} = 2\,\nu_t \frac{\partial U}{\partial x} - \frac{2}{3}\,k \qquad \frac{\tau_{xy}}{\rho_0} = \frac{\tau_{yx}}{\rho_0} = \nu_t \left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} \right) \qquad \frac{\tau_{yy}}{\rho_0} = 2\,\nu_t \frac{\partial V}{\partial y} - \frac{2}{3}\,k \quad (2.45)$$

where

$$k = \frac{1}{2} \left(\overline{(u')^2} + \overline{(v')^2} + \overline{(w')^2} \right)$$

is the turbulent kinetic energy, and ν_t is known as a turbulent or eddy viscosity which, unlike molecular viscosity, is strongly dependent on the state of the flow (Rodi, 1980). Similarly,

the turbulent scalar fluxes are assumed to be proportional to mean concentration gradients and are written

$$\frac{J_x}{\rho_0} = \Gamma \frac{\partial C}{\partial x} \quad , \quad \frac{J_y}{\rho_0} = \Gamma \frac{\partial C}{\partial y} \quad , \tag{2.46}$$

where Γ is the depth-averaged turbulent or eddy diffusivity.

The turbulence problem has now been shifted to determining the distribution of the turbulent viscosities. Options available for this are generally separated into three classes: zero-, one-, and two-equation models. A common zero-equation method used in floodplain flows is the simple prescription of a constant horizontal eddy viscosity throughout the domain. Although this approach is easy to implement and requires few additional computations, it is very simplistic and fails to capture spatial distributions and transport of turbulent flow characteristics. A common two-equation model used in floodplain simulations is the k-ε closure model and uses transport equations to determine spatial distributions of turbulent kinetic energy, k, and turbulent dissipation, ε. The k-ε closure scheme is one of the most commonly used turbulence schemes for depth-averaged floodplain simulations because of its relative ease of use and its generally accurate results (Nicholas & McLelland, 2004). The turbulent viscosities and diffusivities are calculated by

$$\nu_t = c_\mu \frac{k^2}{\varepsilon} \quad , \quad \Gamma = \frac{\nu_t}{\sigma_t} \quad , \tag{2.47}$$

where c_μ is an empirical constant and σ_t is the turbulent Schmidt number. This assumes the turbulent diffusivity is related to turbulent viscosity by an approximately constant proportion. The turbulent Schmidt number is equal to 0.7 when using temperature and is on the order of one for a general scalar tracer (Pope, 2000). The transport equations for use in depth-averaged calculations were derived by Rodi (1980) and are given as

$$\begin{aligned}
\frac{\partial k}{\partial t} + U\frac{\partial k}{\partial x} + V\frac{\partial k}{\partial y} &= \frac{\partial}{\partial x}\left(\frac{\nu_t}{\sigma_k}\frac{\partial k}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_t}{\sigma_k}\frac{\partial k}{\partial y}\right) + \mathscr{P}_h + \mathscr{P}_{kV} - \varepsilon, \qquad (2.48)\\
\frac{\partial \varepsilon}{\partial t} + U\frac{\partial \varepsilon}{\partial x} + V\frac{\partial \varepsilon}{\partial y} &= \frac{\partial}{\partial x}\left(\frac{\nu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{\nu_t}{\sigma_\varepsilon}\frac{\partial \varepsilon}{\partial y}\right) + c_{\varepsilon 1}\frac{\varepsilon}{k}\mathscr{P}_h + \mathscr{P}_{\varepsilon V} - c_{\varepsilon 2}\frac{\varepsilon^2}{k},
\end{aligned}$$

where

$$\begin{aligned}
\mathscr{P}_h &= \text{production of } k \text{ by shear in the mean fluid motion,} \\
&= \nu_t \left(2 \left(\frac{\partial U}{\partial x} \right)^2 + 2 \left(\frac{\partial V}{\partial y} \right)^2 + \left(\frac{\partial U}{\partial y} + \frac{\partial V}{\partial x} \right)^2 \right) \\
\mathscr{P}_{kV} &= \text{production of } k \text{ by bed shear,} \\
&= c_f^{-1/2} \left(\frac{u_*^3}{h} \right) \\
\mathscr{P}_{\varepsilon V} &= \text{increase in } \varepsilon \text{ due to bed shear,} \\
&= c_{\varepsilon\Gamma}\, c_{\varepsilon 2}\, c_\mu^{1/2}\, c_f^{-3/4} \left(\frac{u_*^4}{h^2} \right)
\end{aligned}$$

and $c_{\varepsilon 1}$, $c_{\varepsilon 2}$, $c_{\varepsilon\Gamma}$, σ_k, and σ_ε are empirical constants. Commonly used values for these parameters are given by Rodi (1980) and are shown in Table 2.3. The value for $c_{\varepsilon\Gamma}$ takes a value given by Rodi (1980) as 3.6 for laboratory experiments and a value of 1.8 for natural river flows (Wu *et al.*, 2005). These values are empirical coefficients and are not be expected to apply universally to all situations.

Table 2.3: Values for empirical constants in the k-ε turbulence closure scheme

c_μ	σ_k	σ_ε	$c_{\varepsilon 1}$	$c_{\varepsilon 2}$	$c_{\varepsilon 3}$	$c_{\varepsilon\Gamma}$	c_{vk}
0.09	1.0	1.3	1.44	1.92	1.33	1.8, 3.6	1.0

Equation (2.48) states the time rate of change of turbulent kinetic energy in a two-dimensional differential volume is balanced by advective and diffusive transport through the cell, production by mean fluid motion and bed shear, and destruction by viscous dissipation. The equation for ε is formulated similarly and tracks the spatial distribution of dissipation. This k-ε formulation indirectly accounts for the production of turbulent kinetic energy due to flow through vegetation in the terms $\mathscr{P}_{kV}$ and $\mathscr{P}_{\varepsilon V}$ which involve the effective shear velocity and effective friction coefficient, c_f, which accounts for both bottom roughness and flow through vegetation. When the effects of bottom friction and drag force are not combined, the addition of new terms into the transport equations is needed to account for k production by flow through vegetation. A derivation by López & García (1998) showed the necessity of adding production terms $\mathscr{P}_v$ and $c_{\varepsilon 1} \frac{\varepsilon}{k} c_{\varepsilon 3} \mathscr{P}_v$ to the k and ε transport equations,

respectively. The term $\mathscr{P}_v$ accounts for shear production by submerged vegetation and is given by

$$\mathscr{P}_v = c_{vk}\left(F_{Dx}U + F_{Dy}V\right) \qquad F_{Dx} = \frac{1}{2}\,C_D\,\lambda\,U\,\sqrt{U^2+V^2}$$

where F_D is the drag force and $c_{\varepsilon 3}$ and c_{vk} are empirical constants with commonly used values given by Wu *et al.* (2005) and shown in Table 2.3. Since the bottom stress and drag force terms were combined in the formulation of the momentum equation, the original transport formulas (2.48) will be used in this model.

The lateral boundary conditions for the k-ε model can be broken into three types: boundary conditions at inflows, boundary conditions at outflows, and boundary conditions at sidewalls. At inlet boundaries, a uniform velocity is usually assumed and the inflowing turbulent kinetic energy is set at a small value, such as 0.3%, of the squared inflow velocity (Bernard, 1993). The inflowing dissipation rate is set equal to the turbulent energy production at the inlet. At outflowing boundaries, the normal derivatives of k and ε are set at zero (Bernard, 1993). Near sidewall boundaries, an approach using the logarithmic law of the wall is generally used to calculate values of k and ε close to the wall. This approach will not be used because the generally wide and shallow nature of floodplains produces only small velocity gradients near lateral boundaries. Instead, values of zero are prescribed for both k and ε at the wall boundary.

2.8 Equation summary

In summary, the equations governing freshwater flows in a floodplain environment are:

Continuity equation:

$$\frac{\partial\,\zeta}{\partial\,t} + \frac{\partial\,(hU)}{\partial\,x} + \frac{\partial\,(hV)}{\partial\,y} = P_r - ET \pm GW \tag{2.49}$$

Momentum equations:

$$\frac{\partial\,U}{\partial\,t} + U\frac{\partial\,U}{\partial\,x} + V\frac{\partial\,U}{\partial\,y} = -g\frac{\partial\,\zeta}{\partial\,x} + \frac{\tau_{xs}-\tau_{xb}}{\rho_0 h} + \frac{1}{\rho_0 h}\left(\frac{\partial\,(h\tau_{xx})}{\partial\,x} + \frac{\partial\,(h\tau_{xy})}{\partial\,y}\right) + f_c V \tag{2.50}$$

$$\frac{\partial\,V}{\partial\,t} + U\frac{\partial\,V}{\partial\,x} + V\frac{\partial\,V}{\partial\,y} = -g\frac{\partial\,\zeta}{\partial\,y} + \frac{\tau_{ys}-\tau_{yb}}{\rho_0 h} + \frac{1}{\rho_0 h}\left(\frac{\partial\,(h\tau_{yx})}{\partial\,x} + \frac{\partial\,(h\tau_{yy})}{\partial\,y}\right) - f_c U \tag{2.51}$$

Scalar transport equation:

$$\frac{\partial C}{\partial t} + U\frac{\partial C}{\partial x} + V\frac{\partial C}{\partial y} = \frac{1}{h}\left(\frac{\partial}{\partial x}\left(h(k_x + \Gamma)\frac{\partial C}{\partial x}\right) + \frac{\partial}{\partial y}\left(h(k_y + \Gamma)\frac{\partial C}{\partial y}\right)\right) + S_c \quad (2.52)$$

Expressions for evapotranspiration are given by (2.24)-(2.27), surface shear stresses by (2.34), and bottom shear stresses by (2.35)-(2.42). Expressions for the depth-averaged Reynolds stresses and turbulent scalar fluxes are given in equations (2.45)-(2.48).

3 Numerical Methods and Test Problems

3.1 Simplified equations

In this chapter numerical methods and analysis are presented for solutions of the set of governing equations for mass and momentum conservation. The equations will be solved with a two-level, semi-implicit time integration scheme using the advection discretization proposed by Stelling & Duinmeijer (2003) whereby energy head is conserved through flow contractions and momentum is conserved through expansions. While the advection discretization produces results that are strictly first order convergent overall, the use of slope limiters gives 'almost second order' convergence (Stelling & Duinmeijer, 2003). An iterative method is used for evaluating bottom friction.

The chapter will begin with the presentation of numerical methods and analysis for a set of simplified, one-dimensional equations. The simplified governing equations, shown below, are the one-dimensional versions of (2.49) and (2.50). Source/sink terms in the continuity equation and surface shear and turbulent stresses have been neglected as small in comparison to the other terms. This simplification allows for a more straightforward analysis of the proposed numerical methods.

$$\frac{\partial \zeta}{\partial t} + \frac{\partial (hU)}{\partial x} = 0 \tag{3.1}$$

$$\frac{\partial U}{\partial t} + U\frac{\partial U}{\partial x} + g\frac{\partial \zeta}{\partial x} + \frac{g\, n_M^2}{k_M^2}\frac{U\,|U|}{h^{4/3}} = 0 \tag{3.2}$$

3.2 Discretization methods

The two dependent variables in (3.1) and (3.2) are the water surface elevation, ζ, and the depth-averaged velocity, U. Values for these unknowns are computed at discrete intervals throughout the model domain using a staggered grid such as the one shown in Figure 3.1. The water surface elevation, depth, and bottom roughness values are defined at the centers of the computational cells and fluid velocities are defined at the cell edges. The use of a staggered grid system has several advantages. It allows the straightforward prescriptions of no-flow or no-slip boundary conditions. Schemes using staggered grids also have the advantages of being easily extendable to higher dimensions and allowing the semi-implicit

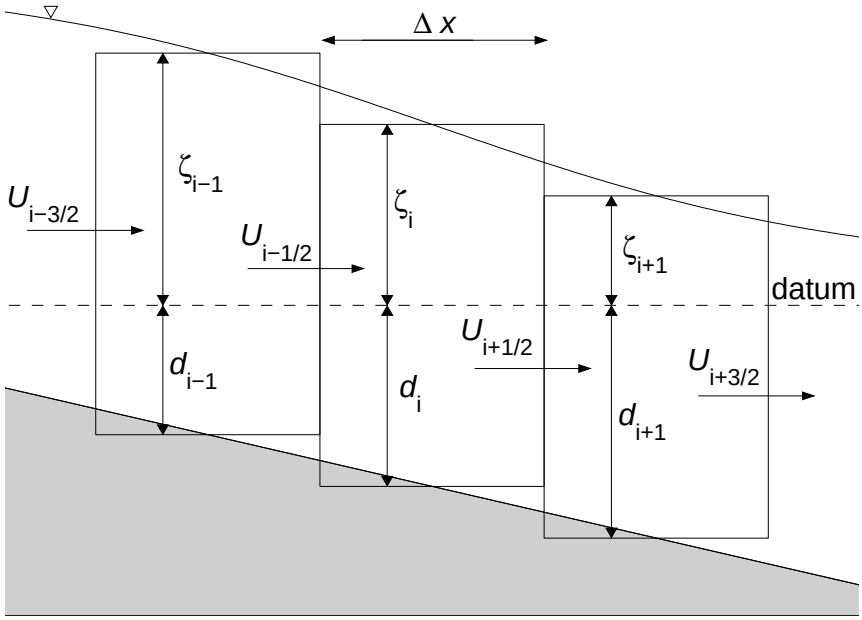

Figure 3.1: Staggered 1D grid

time integration method to be used, producing computationally efficient solutions. The length of each computational cell is Δx, a value which is constant throughout the domain. The time discretization is defined by the set of points $t^n = n\Delta t$, for $n = 0, 1, 2, \ldots, n_{ts}$, where n_{ts} is the number of time steps and Δt is constant through the duration of a simulation. An initial spatial distribution of ζ and U values is specified at $t = 0$; a series of computations then advances the solution from a previously known state at t^n to a new state at time t^{n+1}. Boundary conditions at both ends of the domain are prescribed for each time step.

3.2.1 Continuity and momentum discretizations

Following the scheme presented by Stelling & Duinmeijer (2003), with a few minor corrections and modifications, the continuity equation (3.1) is discretized using the second order accurate trapezoidal method as

$$\frac{\zeta_i^{n+1} - \zeta_i^n}{\Delta t} + \frac{{}^*h_{i+1/2}^n U_{i+1/2}^{n+1/2} - {}^*h_{i-1/2}^n U_{i-1/2}^{n+1/2}}{\Delta x} = 0 \quad , \tag{3.3}$$

where $U_{i+1/2}^{n+1/2} = \frac{1}{2}(U_{i+1/2}^n + U_{i+1/2}^{n+1})$. Since values of h are defined at the cell centers, a method is needed to estimate values of h at the cell edges, which are denoted in the form ${}^*h_{i+1/2}$.

One method of estimation is given by first order upwinding[1]

$$ {}^{*}h_{i+1/2} \;=\; \min(d_i, d_{i+1}) \;+\; \begin{cases} \zeta_i \quad , & \text{if } U_{i+1/2} \geq 0, \\ \zeta_{i+1} \quad , & \text{if } U_{i+1/2} < 0 \; . \end{cases} \tag{3.4} $$

Equation (3.3) can then be rearranged to

$$ \zeta_i^{n+1} = \underbrace{\zeta_i^n + \frac{\Delta t}{\Delta x}\left({}^{*}h_{i-1/2}^{n} U_{i-1/2}^{n} - {}^{*}h_{i+1/2}^{n} U_{i+1/2}^{n} \right)}_{} + \frac{\Delta t}{\Delta x}\left({}^{*}h_{i-1/2}^{n} U_{i-1/2}^{n+1} - {}^{*}h_{i+1/2}^{n} U_{i+1/2}^{n+1} \right) \tag{3.5} $$

where the terms in the underbrace are all known at time level n and are henceforth denoted by the symbol $\widehat{\zeta_i}$.

Since it is assumed the depths, d, do not vary with time, $\frac{\partial \zeta}{\partial t} = \frac{\partial h}{\partial t}$ and assuming locally positive flow, (3.3) can alternatively be rearranged to

$$ h_i^{n+1} = \left(1 - \frac{\Delta t \, U_{i+1/2}^{n+1/2}}{\Delta x} \right) h_i^n + \left(\frac{\Delta t \, U_{i-1/2}^{n+1/2}}{\Delta x} \right) h_{i-1}^n \quad . $$

Non-negative water depths are then guaranteed provided the advection Courant-Friedrich-Lewy (CFL) condition,

$$ \frac{\Delta t \, U_{i+1/2}^{n+1/2}}{\Delta x} \leq 1 \quad , \tag{3.6} $$

is satisfied. A fortunate consequence of fulfilling (3.6), which must be satisfied regardless to ensure model stability, is that no special wetting and drying procedures are required.

The time discretization of the momentum equation (3.2) is based on the semi-implicit method, which is well suited for staggered grids and has been successfully applied in many geophysical fluid dynamics simulations (Backhaus, 1983; Casulli & Cheng, 1992; Smith, 1997). In the semi-implicit method, the water surface slope term and the bottom friction term are treated implicitly while advection and other terms previously removed from the simplified 1D equations (surface shear, Coriolis forces, turbulent stresses) are treated explicitly.

$$ \frac{\partial U}{\partial t} + \underbrace{U \frac{\partial U}{\partial x}}_{\text{explicit}} + \underbrace{g \frac{\partial h}{\partial x}}_{\text{implicit}} + \underbrace{\frac{g \, n_M^2}{k_M^2} \frac{U \, |U|}{h^{4/3}}}_{\text{implicit}} = 0 \tag{3.7} $$

[1]The upwinding definition shown here varies considerably from the definition given in Stelling & Duinmeijer (2003) and was obtained from these authors by personal communication.

By treating the water surface slope term implicitly, the dependence of model stability on the highly restrictive gravity wave CFL condition, $\Delta t\sqrt{gh}/\Delta x \leq 1$, is removed. With this condition removed, stable values of Δt can be much larger than if the term was treated implicitly and long term environmental simulations can be run quickly. The explicit treatment of the advection term requires the model time step to still be limited by the advection CFL condition, (3.6). This constraint is, however, much less restrictive and the explicit treatment of some terms in the momentum equation increases model efficiency.

Using the semi-implict method, the momentum equation is discretized as

$$\frac{U_{i+1/2}^{n+1} - U_{i+1/2}^{n}}{\Delta t} + \left(U\frac{\partial U}{\partial x}\right)_{i+1/2}^{n} + g\,\frac{\zeta_{i+1}^{n+1/2} - \zeta_{i}^{n+1/2}}{\Delta x} + \left(\frac{g\,n_M^2}{k_M^2}\frac{U\,|U|}{h^{4/3}}\right)_{i+1/2}^{n+1/2} = 0 \quad , \tag{3.8}$$

where the friction term is

$$\left(\frac{g\,n_M^2}{k_M^2}\frac{U\,|U|}{h^{4/3}}\right)_{i+1/2}^{n+1/2} = g\,\frac{\overline{n_M}_{i+1/2}}{k_M^2\left(\overline{h}_{i+1/2}^{n+1/2}\right)^{4/3}}\left(\chi_{i+1/2}|U_{i+1/2}^{n}|U_{i+1/2}^{n+1} + \left(1-\chi_{i+1/2}\right)|U_{i+1/2}^{n}|U_{i+1/2}^{n}\right) \tag{3.9}$$

and the advection term is left in the form $\left(U\frac{\partial U}{\partial x}\right)_{i+1/2}^{n}$ to be addressed in the next section. The overbars in (3.9) denote a standard spatial average, $\overline{h}_{i+1/2} = \frac{1}{2}(h_i + h_{i+1})$. The coefficient $\chi_{i+1/2}$ was suggested by Cunge *et al.* (1980) and used successfully by Smith (1997) to improve stability and accuracy in cases where rapid variations and flow reversals are possible. For the first iteration, $\chi_{i+1/2}$ is taken to be 1.0. For each subsequent iteration, it is calculated using

$$\chi_{i+1/2} = \frac{\widetilde{U}_{i+1/2}^{n+1/2}\,\widetilde{U}_{i+1/2}^{n+1/2} - U_{i+1/2}^{n}U_{i+1/2}^{n}}{U_{i+1/2}^{n}\left(\widetilde{U}_{i+1/2}^{n+1} - U_{i+1/2}^{n}\right)} = \frac{\widetilde{U}_{i+1/2}^{n+1}}{4\,U_{i+1/2}^{n}} + \frac{3}{4} \quad , \tag{3.10}$$

where the tilde denotes the value obtained from the last iteration. After some substitutions, an expanded version of (3.8) is obtained

$$\begin{aligned} U_{i+1/2}^{n+1} = {} & U_{i+1/2}^{n} - \Delta t\left(U\frac{\partial U}{\partial x}\right)_{i+1/2}^{n} + \frac{g\Delta t}{2\Delta x}\left(\zeta_i^n - \zeta_{i+1}^n\right) + \frac{\left(1-\chi_{i+1/2}\right)}{\chi_{i+1/2}}\,\gamma_{i+1/2}U_{i+1/2}^{n} \\ & - \gamma_{i+1/2}\,U_{i+1/2}^{n+1} + \frac{g\Delta t}{2\Delta x}\left(\zeta_i^{n+1} - \zeta_{i+1}^{n+1}\right) \\ \gamma_{i+1/2} = {} & \frac{\Delta t\,n_M^2\,g}{k_M^2\left(\overline{h}_{i+1/2}^{n+1/2}\right)^{4/3}}\,\chi_{i+1/2}\,|U_{i+1/2}^{n}| \quad . \end{aligned} \tag{3.11}$$

The first four terms on the right hand side of (3.11) are all known at time level n. In a manner similar to the continuity equation procedure, these explicit terms can be collected

and denoted $\widehat{U}_{i+1/2}$. A final expression for the new U value is then obtained

$$U_{i+1/2}^{n+1} = \frac{1}{1+\gamma_{i+1/2}}\widehat{U}_{i+1/2} + \frac{g\Delta t}{2\Delta x\left(1+\gamma_{i+1/2}\right)}\left(\zeta_i^{n+1} - \zeta_{i+1}^{n+1}\right) \quad . \tag{3.12}$$

An equation for $U_{i-1/2}^{n+1}$ can be derived likewise and substituted back into (3.5) to get an expression of the form

$$\Xi_{i-1/2}\zeta_{i-1}^{n+1} + \left(1 - \Xi_{i-1/2} - \Xi_{i+1/2}\right)\zeta_i^{n+1} + \Xi_{i+1/2}\zeta_{i+1}^{n+1} = (RHS)_i \quad ,$$

where

$$\begin{aligned} \Xi_{i+1/2} &= -\,{}^{*}h_{i+1/2}^{n}\left(\frac{\Delta t}{2\Delta x}\right)\left(\frac{g\Delta t}{2\Delta x\left(1+\gamma_{i+1/2}\right)}\right) \\ (RHS)_i &= \widehat{\zeta}_i + \frac{\Delta t}{2\Delta x}\left(\frac{{}^{*}h_{i-1/2}^{n}}{1+\gamma_{i-1/2}}\widehat{U}_{i-1/2} - \frac{{}^{*}h_{i+1/2}^{n}}{1+\gamma_{i+1/2}}\widehat{U}_{i+1/2}\right) \end{aligned}$$

Applying this procedure to each cell in the computational domain results in a set of equations which form the symmetric tridiagonal system

$$\begin{bmatrix} (1-\Xi_{1/2}-\Xi_{3/2}) & \Xi_{3/2} & 0 & \cdots & 0 \\ \Xi_{3/2} & (1-\Xi_{3/2}-\Xi_{5/2}) & \ddots & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \Xi_{N-1/2} \\ 0 & \cdots & 0 & \Xi_{N-1/2} & (1-\Xi_{N-1/2}-\Xi_{N+1/2}) \end{bmatrix} \begin{bmatrix} \zeta_1^{n+1} \\ \zeta_2^{n+1} \\ \vdots \\ \vdots \\ \zeta_N^{n+1} \end{bmatrix} = \begin{bmatrix} (RHS)_1 \\ (RHS)_2 \\ \vdots \\ \vdots \\ (RHS)_N \end{bmatrix} \quad .$$

This system can be solved efficiently for water surface elevations at the new time level using the Thomas algorithm, a method of Gaussian elimination that accounts for pivot values of zero. An upstream water surface elevation boundary condition, for example, may easily be prescribed by setting $\Xi_{3/2} = 0$ in the first row and $(RHS)_1 =$ the prescribed value. Once all of the ζ values are calculated, the fluid velocities can be determined using (3.12), and the process can be repeated with updated values of χ to improve estimation of the bottom friction term. In Section 3.3.3, it will be shown that one iteration is generally sufficient.

3.2.2 Advection algorithm

The method used to discretize the advection term in (3.7) was proposed by Stelling & Duinmeijer (2003) and uses the general idea of applying a discretization consistent with

the momentum principle during flow expansions and a discretization consistent with the Bernoulli equation during flow contractions. In strong flow expansions, energy is dissipated and conserving energy head would not be correct. In strong flow contractions, it will be shown that applying conservation of momentum leads to increases in fluid energy, an incorrect result that can lead to model instability. There are several benefits of using the method presented here. Floodplain topography, especially near the channel-floodplain interface can vary quickly in comparison to the resolution of the computational mesh. In these areas, the hydrostatic approximation is locally invalid and the conservation properties of the numerical method become crucial to the accuracy and stability of the solution (Stelling & Duinmeijer, 2003). Unlike other finite volume methods, such as Riemann solvers or Godunov methods, the method presented here works very well with staggered grids and the semi-implicit method, is relatively simple to implement, and can easily be extended to higher dimensions. Riemann solvers and Godunov methods usually require explicit time integration, necessitating very small time steps, and display problems near steep bed transitions (Stelling & Duinmeijer, 2003; Brufau *et al.*, 2002).

The idea that applying the momentum principle through a flow contraction can lead to increases in energy head can be shown using the illustration in Figure 3.2. The mass balance and momentum principle applied to this transition are given as

$$\begin{aligned} h_i U_i &= h_{i+1} U_{i+1} \\ \rho_0\, q_{i+1}\, U_{i+1} - \rho_0\, q_i\, U_i &= \frac{1}{2}\,\rho_0\, g\left(h_i^2 - h_{i+1}^2\right) - F_B \\ F_B &= \rho_0\, g\left(\zeta_{i+1/2} + d_{i+1} + \frac{1}{2}(d_i - d_{i+1})\right)(d_i - d_{i+1}) \quad , \end{aligned}$$

where F_B is the bed force term (Chanson, 2004). If it is assumed that $\zeta_{i+1/2}$ can be written as $\zeta_{i+1/2} = \psi\zeta_i + (1-\psi)\zeta_{i+1}$ and that $d_{i+1} = 0$, without loss of generality, then the above equations can be manipulated into a single dimensionless form

$$2Fr^2\left(\frac{\eta + \delta_c - 1}{\eta + \delta_c}\right) + (1-\eta)\left((1+\eta) + 2(1-\psi)\delta_c\right) = 0 \quad , \tag{3.13}$$

where

$$Fr \equiv \frac{U_{i+1}}{\sqrt{g\,\zeta_{i+1}}} \quad , \quad \eta \equiv \frac{\zeta_i}{\zeta_{i+1}} \quad , \quad \delta_c \equiv \frac{d_i}{\zeta_{i+1}}$$

are the downstream Froude number, the relative water surface difference, and the relative strength of the contraction, respectively.

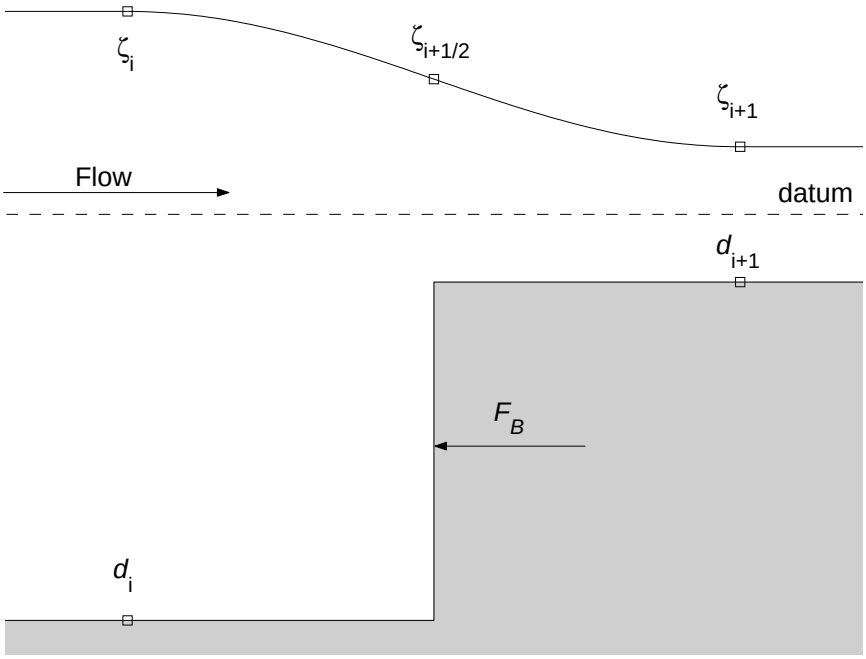

Figure 3.2: Schematic of a sudden bed transition

By defining the systematic energy head, $E = U^2/2g + \zeta$, the relative head loss through the contraction, β_E, can be written as

$$\beta_E \equiv \frac{E_i - E_{i+1}}{E_i} = 1 - \frac{\frac{1}{2}Fr^2 + 1}{\frac{1}{2}\left(\frac{Fr}{\eta+\delta_c}\right)^2 + \eta} \quad . \tag{3.14}$$

For a given contraction strength δ_c and value of ψ, Equation (3.13) can be solved to yield η as a function of Fr, which can subsequently be used in (3.14) to plot β_E as a function of Fr. The results of such an analysis are shown in Figure 3.3 for $\psi = \frac{1}{2}$ and $\delta_c = 2, 4, 8$. Since negative head loss values correspond to increases in energy head, substantial increases in energy head are seen to result from the application of the momentum principle across a strong flow contraction. This idea will be further supported by test case results presented in the next section.

The advection term is approximated in two different manners based on the quantity to be conserved. The so-called momentum conserving form of the momentum equation (3.2) is

$$\frac{\partial}{\partial t}(hU) + \frac{\partial}{\partial x}\left(hU^2 + \frac{1}{2}gh^2\right) + c_f\, U|U| - gh\frac{\partial d}{\partial x} = 0 \quad .$$

The spatial discretization of the advection term that was shown by Stelling & Duinmeijer

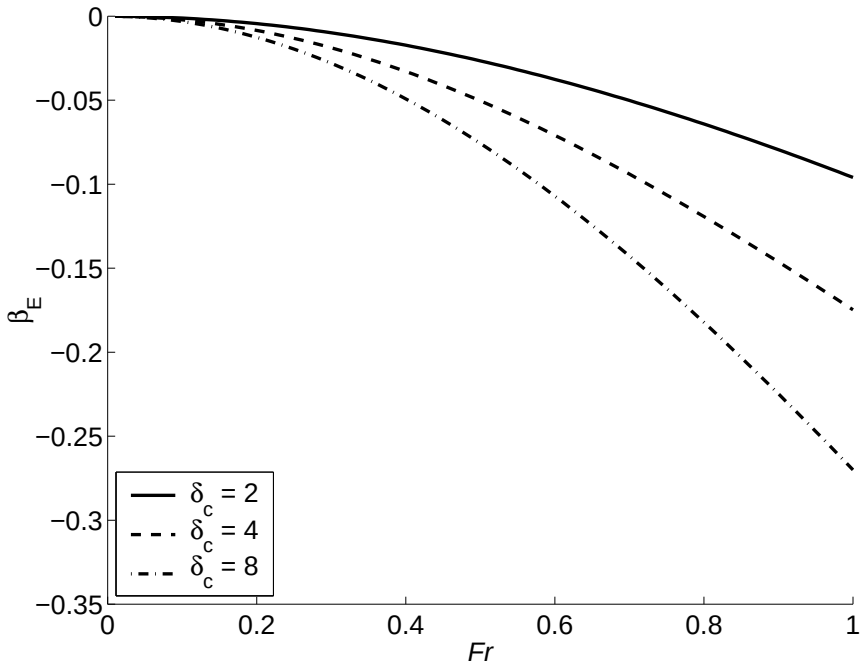

Figure 3.3: Relative head losses through a sudden bed contraction as a function of the upstream Froude number

(2003) to be consistent with this formulation is

$$\left(U\frac{\partial U}{\partial x}\right)^n_{i+1/2} = \frac{1}{\overline{h}^n_{i+1/2}}\left(\frac{\overline{q}^n_{i+1}\,{}^*U^n_{i+1} - \overline{q}^n_i\,{}^*U^n_i}{\Delta x} - U^n_{i+1/2}\frac{\overline{q}^n_{i+1} - \overline{q}^n_i}{\Delta x}\right) \quad , \tag{3.15}$$

where

$$\overline{h}_{i+1/2} \equiv \frac{1}{2}\left(h_i + h_{i+1}\right) \quad , \quad \overline{q}_i \equiv \frac{1}{2}\left(q_{i-1/2} + q_{i+1/2}\right) \quad , \quad q_{i+1/2} \equiv {}^*h_{i+1/2}\,U_{i+1/2}$$

and the approximations for U at the cell centers are upwinded by

$${}^*U_i \;=\; \begin{cases} U_{i-1/2} & , \text{ if } \frac{1}{2}(q_{i-1/2} + q_{i+1/2}) \geq 0 \ , \\ U_{i+1/2} & , \text{ if } \frac{1}{2}(q_{i-1/2} + q_{i+1/2}) < 0 \ . \end{cases}$$

Alternatively, the conservation of energy head form of (3.2) is

$$\frac{\partial U}{\partial t} + \frac{\partial}{\partial x}\left(\frac{1}{2}U^2 + g\zeta\right) + c_f\frac{U|U|}{h} = 0 \quad .$$

The spatial discretization of the advection term that was shown to be consistent with this formulation is

$$\left(U\frac{\partial U}{\partial x}\right)^n_{i+1/2} = \frac{1}{2}\left(\frac{({}^*U^n_{i+1})^2 - ({}^*U^n_i)^2}{\Delta x}\right) \quad , \tag{3.16}$$

where the approximations for U at the cell centers are upwinded by

$$ {}^{*}U_i \;=\; \begin{cases} U_{i-1/2} \quad , \quad \text{if } \frac{1}{2}(U_{i-1/2}+U_{i+1/2}) \geq 0 \ , \\ U_{i+1/2} \quad , \quad \text{if } \frac{1}{2}(U_{i-1/2}+U_{i+1/2}) < 0 \ . \end{cases} $$

The choice between the two advection discretizations can be made dynamically[2] based on the current state of the flow. If the flow is experiencing a strong contraction, the energy head conserving discretization (3.16) should be applied; otherwise the momentum conserving discretization (3.15) is applied. The criteria for applying each principle is given in Table 3.1, where the parameter $\epsilon > 0$ can be varied such that the energy head form is only applied in strong flow contractions.

Table 3.1: Criteria for the conservation principle to apply to the advection approximation

Criteria	Principle to apply
$\frac{1}{\Delta x}\left(U_{i+1/2}-U_{i-1/2}\right) > \epsilon$	conservation of energy head
$\frac{1}{\Delta x}\left(U_{i+1/2}-U_{i-1/2}\right) \leq \epsilon$	conservation of momentum

3.2.3 Slope limiters

The standard upwind method for estimating the ζ values at the cell edges and the U values at the cell centers is a first order approximation. All other discretizations used to this point are second order approximations that are centered in time and space. However, since the overall accuracy of a method is dictated by its least accurate term, the framework so far is only first order accurate. To improve the numerical properties of the method, slope limiters can be used to impart 'almost second order' accuracy (Stelling & Duinmeijer, 2003).

For the positive flow direction, the first and second order upwind approximations, denoted ζ^L for low order and ζ^H for higher order, are

$$ \begin{aligned} \zeta^L_{i+1/2} &= \zeta_i \quad , \\ \zeta^H_{i+1/2} &= \zeta_i + \frac{1}{2}(\zeta_i - \zeta_{i-1}) \quad . \end{aligned} $$

[2]Evaluated during the model runtime

While the higher order approximation offers a more accurate estimate of ζ at the cell edges, it is prone to the generation of spurious oscillations near large gradients in the water surface (Durran, 1999). Such large gradients may occur with rapidly varying flow or near the boundary between wet and dry cells. Slope limiting methods solve the problem by using the higher order method only when the water surface profile is sufficiently smooth. This is accomplished by starting with the low order method and adding in a correction factor that changes the approximation to higher order when the water surface is locally smooth. The use of the correction factor is determined by a coefficient 'switch', $C(r)$, which produces values near one in smooth areas and values near zero otherwise. The slope limited formula for the water levels can thus be written as

$$\begin{aligned} \zeta_{i+1/2} &= \zeta^L_{i+1/2} + C(r_\zeta)\left(\zeta^H_{i+1/2} - \zeta^L_{i+1/2}\right) \\ \zeta_{i+1/2} &= \zeta_i + \frac{C(r_\zeta)}{2}\left(\zeta_i - \zeta_{i-1}\right) \quad , \end{aligned} \tag{3.17}$$

where r_ζ is a parameter indicating the relative smoothness of the water surface slope

$$r_\zeta = \frac{\zeta_{i+1} - \zeta_i}{\zeta_i - \zeta_{i-1}} \tag{3.18}$$

and $C(r)$ approaches zero as r deviates from unity. Four common limiter formulas are given in Durran (1999) and are here listed in Table 3.2. By assuming that rapid variations in the water surface elevation cover only a small percentage of the domain, the second order approximation will be used most of the time and the method is said to be 'almost second order' accurate.

Table 3.2: Slope limiter formulas

Slope limiter	$C(r)$
Minimod	$\max\left(0, \min(1, r)\right)$
Superbee	$\max\left(0, \min(1, 2r), \min(2, r)\right)$
Van Leer	$\left(r + \lvert r\rvert\right) / \left(1 + \lvert r\rvert\right)$
Montonized centered (MC)	$\max\left(0, \min(2r, \frac{1}{2}(1 + r), 2)\right)$

Slope limiters will also be used for the upwind approximations of U at the cell centers.

For flow in the positive direction, these formulations are similar to (3.17) and (3.18)

$$\begin{aligned} {}^{*}U_i &= U_{i-1/2} + \frac{C(r_U)}{2}\left(U_{i-1/2} - U_{i-3/2}\right) \\ r_U &= \frac{U_{i+1/2} - U_{i-1/2}}{U_{i-1/2} - U_{i-3/2}} \quad . \end{aligned}$$

Formulas for flow in the negative x-direction can be derived accordingly and are not presented here.

3.3 Test Cases for Model Evaluation

The use of standardized test simulations for which analytical or numerical solutions are well established can be very useful in determining (a) if a model under development is working correctly, (b) how it performs when compared against other previously developed models, and (c) how its stability and convergence properties behave. In this section, the model described herein is subjected to three sets of test cases, each designed to simulate important aspects of flow in floodplain environments. The first test case examines flow over rapidly varying topography, a feature of many geophysical flooding simulations and a common source of numerical difficulties. The second test case examines floodplain wetting and drying, whereby computational cells are subjected to the innundation and recession of a floodpulse. The last test case is a classic hydrograph routing simulation designed to evaluate how well the model predicts the arrival time and peak discharge of an advected floodpulse. Convergence of error measures is assessed using this final test case.

3.3.1 Flow over rapidly varying topography

The test cases used to examine the performance of the model on flow over rapidly varying topography are very similar to the test cases presented by Stelling & Duinmeijer (2003) in the original description of the advection discretization. Results are reproduced here to illustrate the correct functioning of the model and to highlight conservation properties of the advection discretization. In the first test case, the basin is 100 m long and discretized with $\Delta x = 0.5$ m. The bottom is comprised of three gradual flow contractions and two rapid expansions of height 3.3 m, as shown in Figure 3.4. A flow per unit width of $q = 1$ m^2/s was prescribed at the upstream boundary and $\zeta = 1$ m was set on the downstream end. No

bottom friction was used. The initial water surface elevation was set at $\zeta = 1.0$ m and run until equilibrium is reached, which generally occurred in under two minutes of simulation time.

Results are shown in Figures 3.5 and 3.6 for the momentum and energy head advection discretizations. The energy conservation method shows no decreases in energy through rapid expansions and is clearly not correct from a physical standpoint. Energy was dissipated at each of the expansions with the momentum conserving approximation, and results obtained using the dynamic choice conservation principle were identical to the momentum conservation results.

The second rapidly varying topography test case is similar to the first, but with the bottom topography altered to create rapid flow contractions, as shown in Figure 3.7. Test case results are presented in Figures 3.8 and 3.9 for the momentum and energy head discretizations. It is seen in Figure 3.8 that applying the momentum principle through large flow contractions incorrectly increases the energy head across the basin, a property that could detrimentally affect the numerical stability of the model. The dynamic choice scheme produced results that were similar to the energy head conservation results, with only minor energy losses across the basin.

The final rapidly varying topography test case is the simulation of a hydraulic jump via flow over an obstacle. The obstacle is 1 m high and 20 m wide, with the transitions in bed elevation taking place over one grid cell. The flow per unit width at the upstream end was prescribed at 1 m^2/s, and the water surface elevation at the downstream end was held constant at $\zeta = 0$ m. No bottom friction was used and the initial water surface elevation was set at $\zeta = 0$ m and run until equilibrium was reached. Using principles from open channel flow, the analytical solutions for the water depths upstream and over the obstacle are 0.6828 m and 0.4672 m.

Results for the momentum and energy head conservation schemes are shown in Figures 3.10 and 3.11. The momentum conserving method correctly predicts the downstream water elevation, but underpredicts the upstream water level by about 1.5 cm and shows unphysical increases and spikes in energy at the flow transitions. The energy head conserving method correctly predicts the upstream water depths, but fails to predict the transition from

supercritical to subcritical flow downstream of the obstacle. The dynamic choice scheme, illustrated in Figure 3.12, correctly predicts the upstream and downstream water levels and shows no increases in energy. The use of slope limiters flattens the water surface profile in the region above the obstacle and steepens gradients in the transition areas, as illustrated in Figure 3.13 for the MC limiter. Table 3.3 presents the performance of the four slope limiter methods introduced in the previous section; the MC limiter produced the best results, and the minimod limiter also performed well. The van Leer and superbee limiters under- and over-predicted water depths above the obstacle and had problems with stability.

Table 3.3: Performance of slope limiter methods on the hydraulic jump test case

Slope limiter	Upstream depth	Depth over obstacle
Minimod	0.6827	0.4684
Superbee	0.6835	0.4820
Van Leer	0.6897	0.4240
Montonized centered (MC)	0.6828	0.4673
Exact solution	0.6828	0.4672

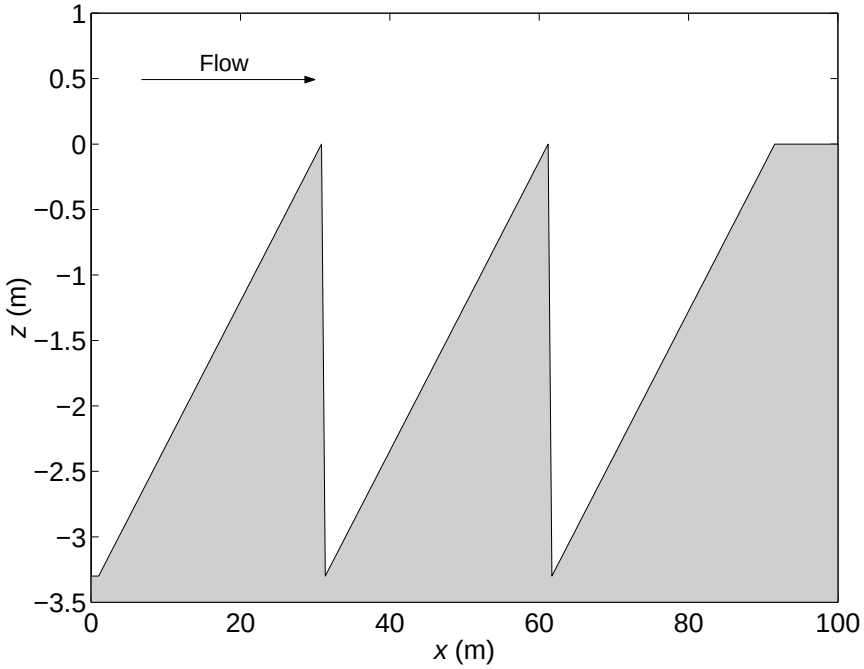

Figure 3.4: Bottom elevation profile for flow over rapid expansions test case

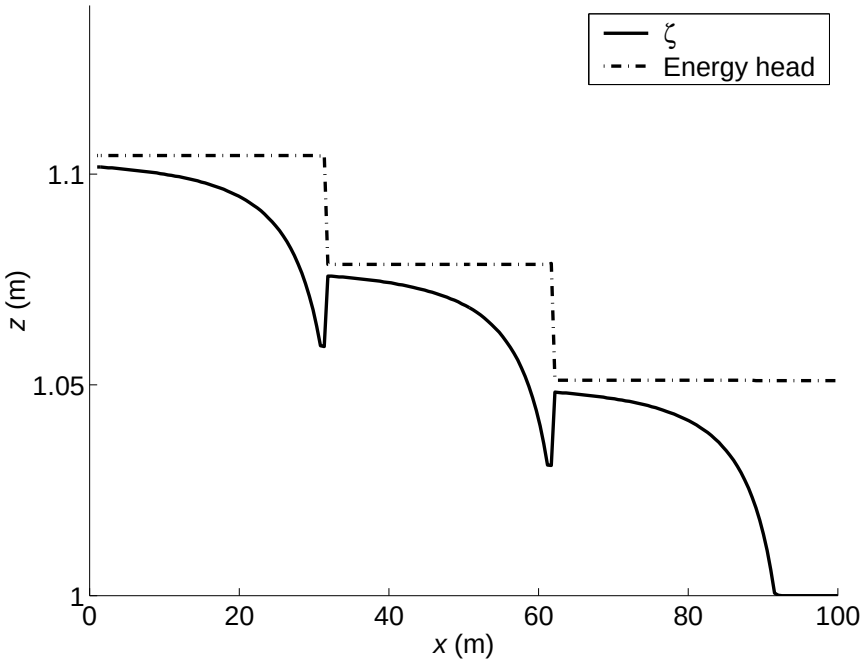

Figure 3.5: Momentum conservation principle applied to flow over rapid expansions

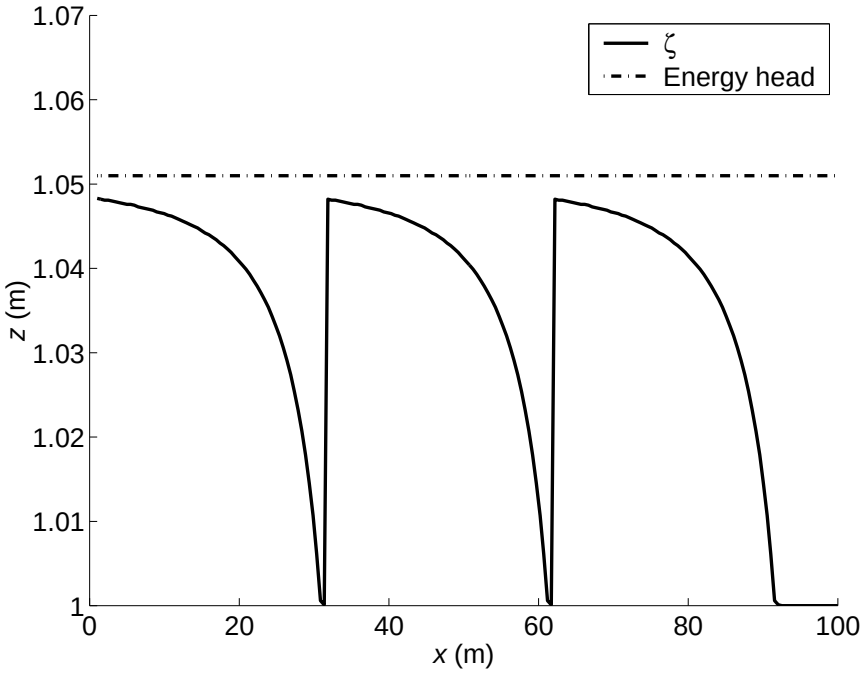

Figure 3.6: Energy head conservation principle applied to flow over rapid expansions

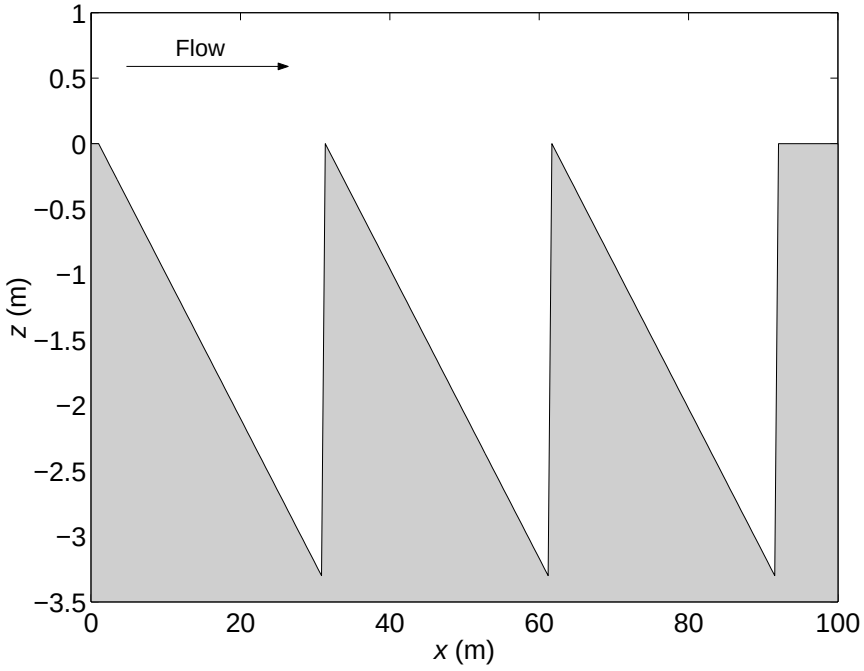

Figure 3.7: Bottom elevation profile for flow over rapid contractions test case

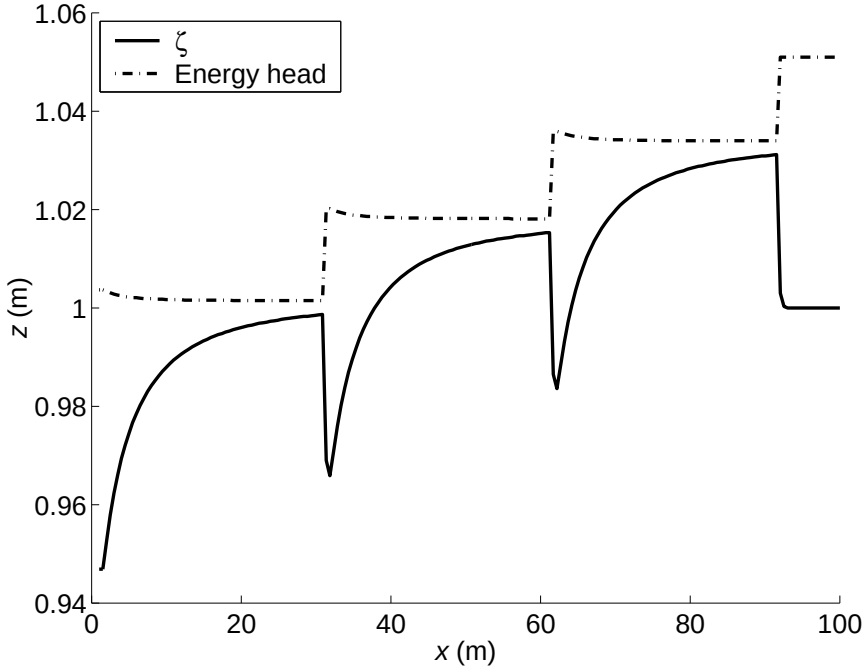

Figure 3.8: Momentum conservation principle applied to flow over rapid expansions

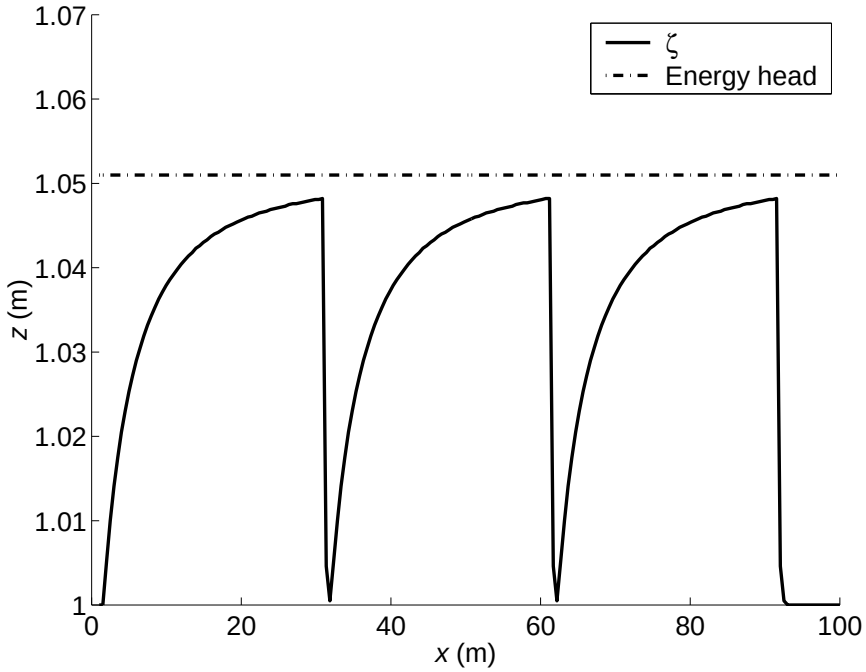

Figure 3.9: Energy head conservation principle applied to flow over rapid expansions

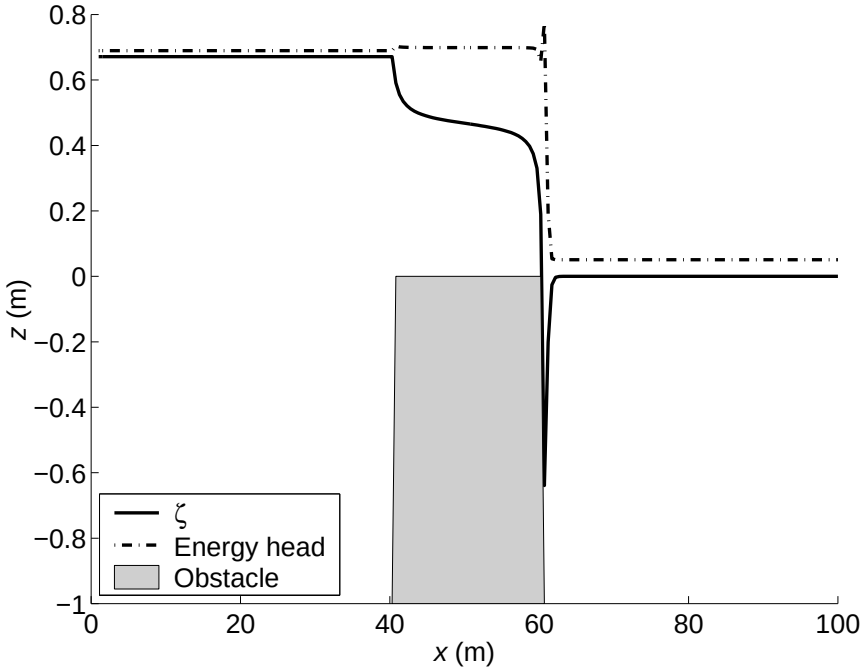

Figure 3.10: Momentum conservation principle applied to flow over an obstacle

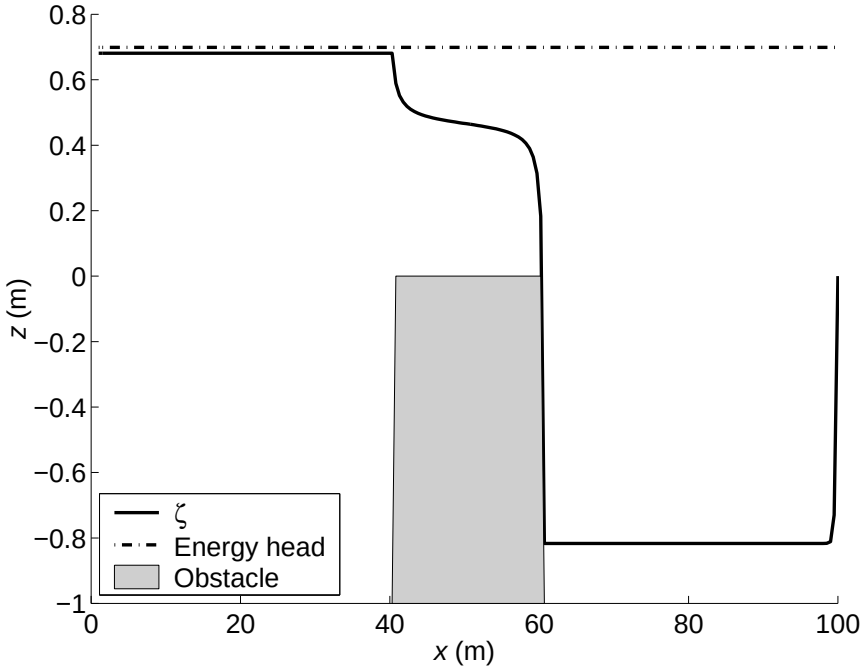

Figure 3.11: Energy head conservation principle applied to flow over an obstacle

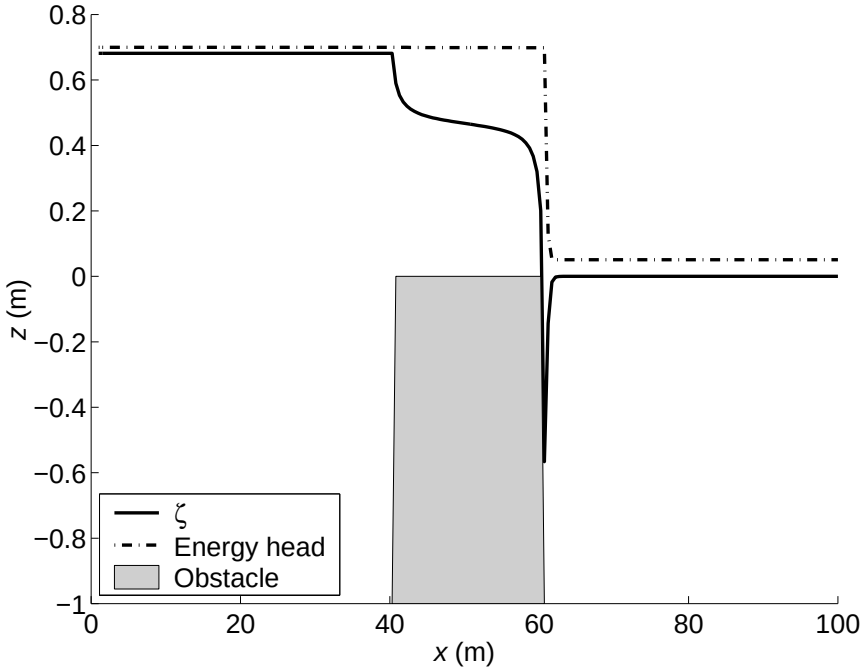

Figure 3.12: Dynamic choice method applied to flow over an obstacle

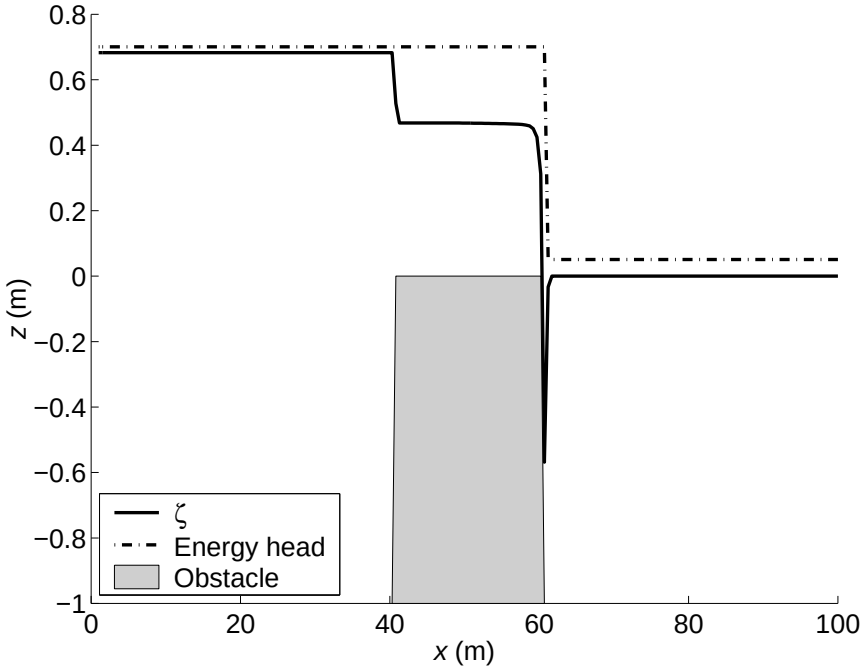

Figure 3.13: Dynamic choice method with slope limiters applied to flow over an obstacle

3.3.2 Domain wetting and drying

The test case used to examine model performance in the presence of wetting and drying fronts was first used by Falconer & Owens (1987) and later by Balzano (1998) to evaluate ten alternative wetting and drying methods. A uniformly sloped basin of length 14,400 m was discretized using 12 cells of length 1200 m. The basin has a depth of 5 m at the open end where tidal oscillations were prescribed and 0 m at the closed end. The unperturbed water surface elevation was located at 0 m and tidal fluctuations of ± 2 m were prescribed with a period of 12 hrs. The Manning roughness coefficient was 0.02, and the time step used for all simulations was $\Delta t = 600$ s. This test case is mainly qualitative and seeks to identify the presence of any unphysical behavior.

Figures 3.14 and 3.15 present water surface profiles at 10 minute intervals for ebb and flood tides using the dynamic choice scheme with slope limiters. No unphysical behavior was observed and the results were qualitatively similar to the majority of the wetting and drying methods tested by Balzano (1998). Examples of unphysical behavior that did not occur with this model are shown in Figure 3.16, where the wetting and drying method of Molinaro *et al.* (1994) overestimates water depths on the ebb tide, and in Figure 3.17, where the variable retention surface (VRS) method proposed by Balzano (1998) produces water surfaces that pass through the bottom boundary. Following the successful completion of this test case, the model was subjected to a more severe circumstance, similar to the first but with the inclusion of a bottom convexity within the intertidal zone. Resulting water surface profiles are shown in Figures 3.18 and 3.19. Again, no unphysical behavior is displayed and the results are qualitatively similar to the most accurate of the ten methods evaluated in the study by Balzano (1998).

Two modifications to the previously presented model framework were required in order to produce the results shown in this section. The first concerns the upwind method used to estimate ζ values at the cell edges. When a second order approximation is used on uniform flow in a sloped basin, the upwinding method of (3.4) produces incorrectly shallow water depths at the cell edges as a result of using the minimum of the adjacent cell depths. This problem is illustrated in Figure 3.20 and was corrected by removing the minimum function

from the upwinded water surface values, redefining them as

$$ {}^{*}h_{i+1/2} = \begin{cases} h_i \quad , & \text{if } U_{i+1/2} \geq 0, \\ h_{i+1} \quad , & \text{if } U_{i+1/2} < 0 \end{cases} $$

and applying the slope limiters to the full water depth rather than the water surface elevation. A second modification was necessary to prevent numerical overflow problems when evaluating the frictional term (3.9) which includes the water depth h in the denominator. Although the method of Stelling & Duinmeijer (2003) guarantees non-negative water depths, h values of zero are still possible and are even necessary when simulating flow over initially dry areas. Other floodplain models have worked around this problem by always keeping a thin layer of water of depth $h = \varepsilon_h$ present over the entire model domain (Braschi *et al.*, 1994). If h values fall below ε_h they are immediately reset. This approach was not chosen for the current model because water of even very small depths before an advancing wave front can substantially effect the calculated wave propagation speed. Instead, a method was used to remove cells from the computations by declaring them dry when the water depth drops below a certain small tolerance value, ε_{shal}. This approach additionally prevents the generation of any spurious velocities due to a fictitious water surface gradient which exists at wetting fronts. The fictitious water surface gradient results from the requirement for non-negative water depths and is shown in Figure 3.21. By removing dry cells from computation, the creation of spurious velocites is prevented. This wetting and drying method is analogous to that proposed by Hervouet & Janin (1994) for the TELEMAC2D model whereby the water surface gradient for cells at a wetting front is reset to zero. The value for ε_{shal} was set at 0.01 mm for this test case and during a later application to the Cosumnes Floodplain was set to 1 mm, although calculated model values of water depths and velocities were not very sensitive to changes in ε_{shal} within the range of 0.001 mm to 10 mm.

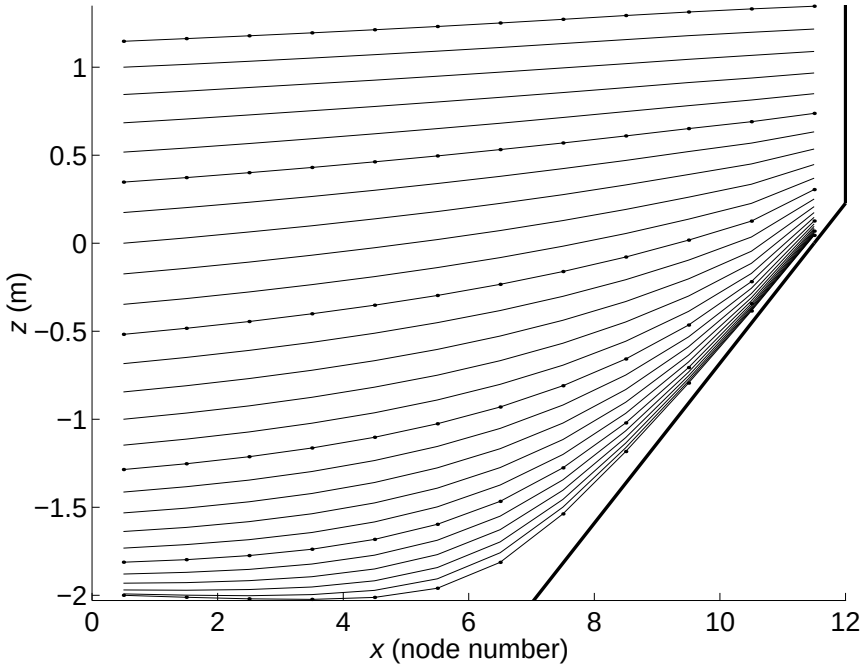

Figure 3.14: Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin on ebb tide. The bold line indicates the basin boundary.

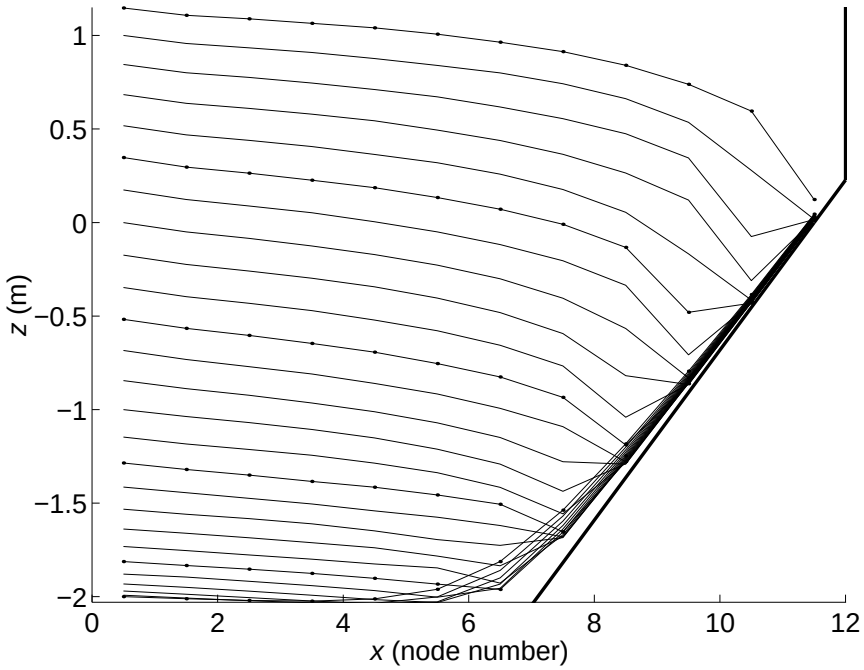

Figure 3.15: Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin on flood tide. The bold line indicates the basin boundary.

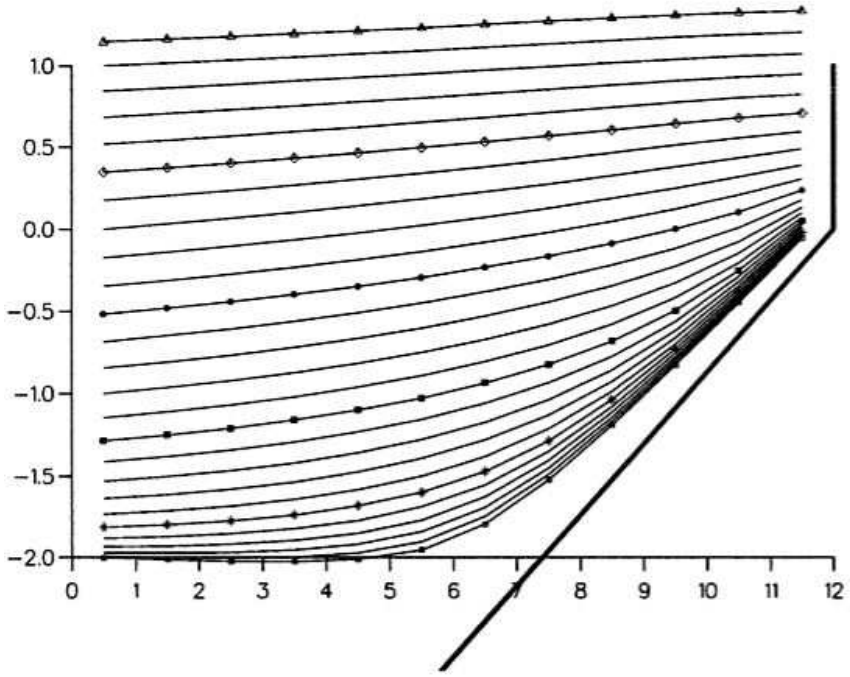

Figure 3.16: Water surface profiles calculated using the wetting and drying method of Molinaro *et al.* (1994) for a uniformly sloped basin on ebb tide (from Balzano (1998))

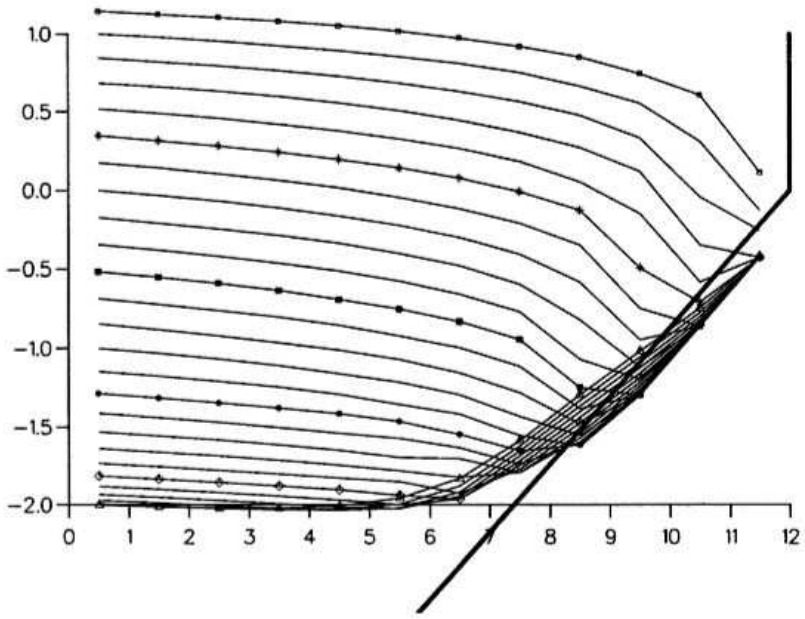

Figure 3.17: Water surface profiles calculated using the variable retention surface (VRS) wetting and drying method of Balzano (1998) for a uniformly sloped basin on ebb tide (from Balzano (1998))

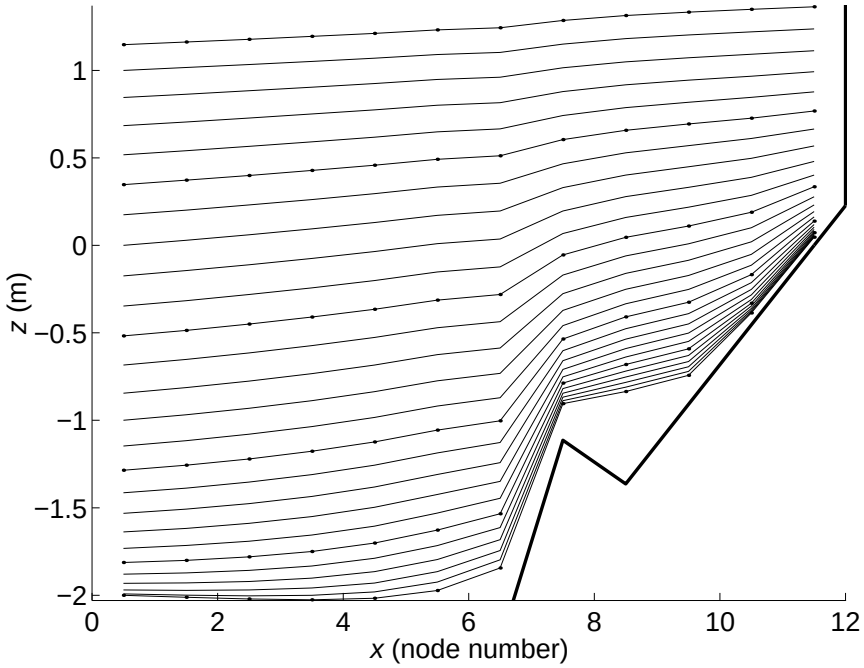

Figure 3.18: Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin with bottom convexity on ebb tide

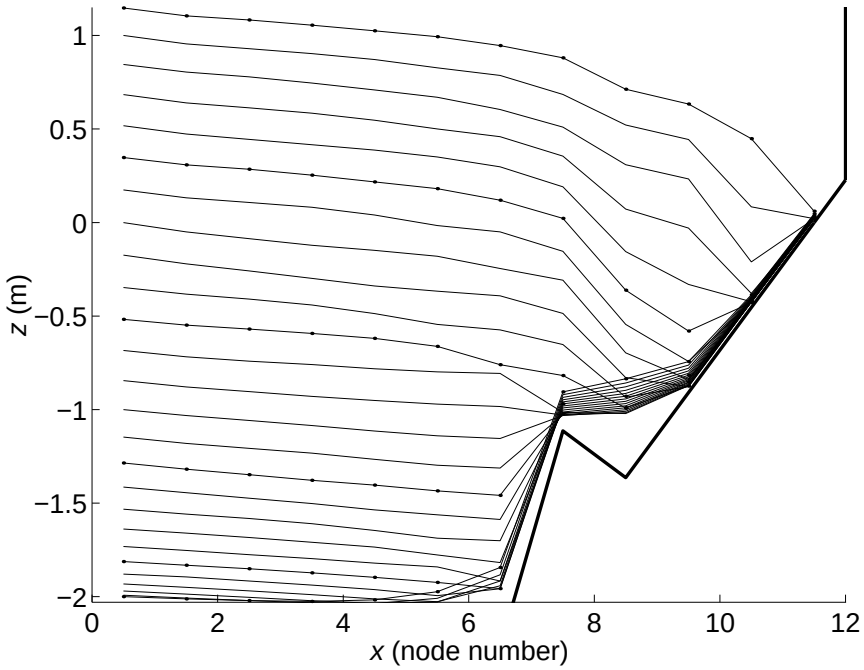

Figure 3.19: Calculated water surface profiles for the wetting and drying test case for a uniformly sloped basin with bottom convexity on flood tide

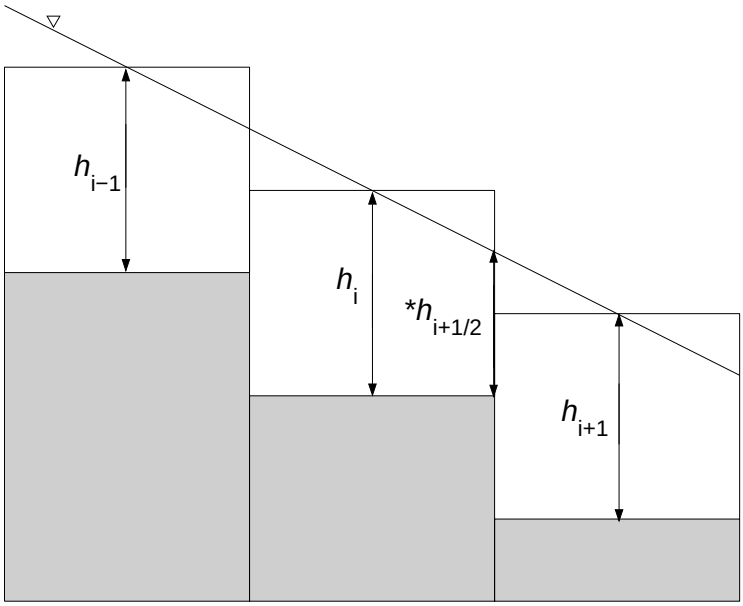

Figure 3.20: Illustration of incorrectly upwinded depths at the cell edges using (3.4) and slope limiters

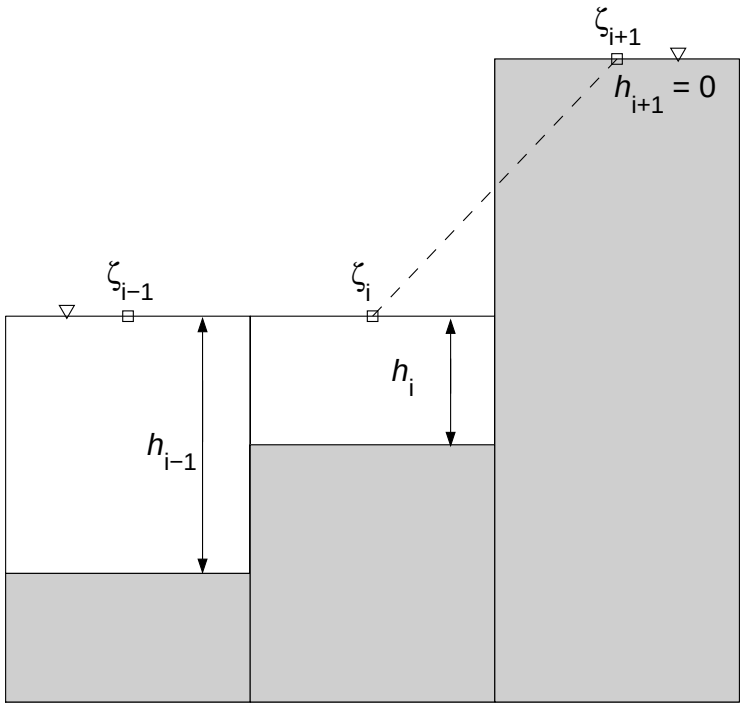

Figure 3.21: Spurious water surface gradient encountered near a wetting front

3.3.3 Hydrograph routing

The final test case involves routing an input hydrograph down a uniformly sloped prismatic channel. The shallow water depths and relatively high frictional resistance create a highly non-linear situation and provide an excellent test for a floodplain model. This test case is also useful to assess the convergence of error measures because of its well known solution and wide use with other flow models. The prismatic channel is 150,000 ft long and 100 ft wide, and has a Manning's n value of 0.045 and a bottom slope of $S_o = 0.001$. Flow at the upstream end of the basin is prescribed as

$$Q(t) = \begin{cases} Q_0 + \frac{3\,Q_0}{\pi}\left(1 - \cos\left(\frac{\pi\,t}{75}\right)\right), & \text{if } t \leq 150 \text{ min}, \\ Q_0, & \text{if } t > 150 \text{ min}, \end{cases}$$

where the initial steady flow is $Q_0 = 250$ ft^3/s. Imperial units are used in order to compare the results to those from previous studies. The initial depth of flow h_0 calculated with the Manning equation is 1.689 ft. Five hundred minutes are simulated, so that the hydrograph is routed some distance downstream but does not interfere with the downstream boundary condition where the flow depth is fixed at h_0. The evolution of the hydrograph as it travels downstream is shown in Figure 3.22. As the initial sinusoidal pulse is advected, bottom friction causes the peak discharge to decrease and the downstream slope to steepen.

Measures of error used to assess the model performance are root mean square (RMS) error, amplitude error, mass preservation error, and phase error, and are defined in Smith (1997) as

$$RMS_e = \left.\frac{100\left(\sum_{n=1}^{n_{ts}}\left[Q^n(x) - Q_b^n(x)\right]^2\right)^{1/2}}{\sqrt{n_{ts}}\,Q_b^{max}(x)}\right|_{x=50{,}000 \text{ ft}} \tag{3.19}$$

$$A_e = \left.\frac{100\left[Q^{max}(x) - Q_b^{max}(x)\right]}{Q_b^{max}}\right|_{x=50{,}000 \text{ ft}} \tag{3.20}$$

$$M_e = \left.100\left(1 - \frac{\int_0^{L_c}\left[h(x,t) - h_0\right]dx}{\int_0^{L_c}\left[h_b(x,t) - h_0\right]dx}\right)\right|_{t=500 \text{ minutes}} \tag{3.21}$$

$$P_e = \left.\frac{100\left[T(x) - T_b(x)\right]}{T_b(x)}\right|_{x=50{,}000 \text{ ft}} \tag{3.22}$$

where

$$T(x) = \frac{\int_0^{T_{end}} t\left[Q(x,t) - Q_0\right]dt}{\int_0^{T_{end}}\left[Q(x,t) - Q_0\right]dt} \tag{3.23}$$

and

$$
\begin{aligned}
L_c &= \text{length of the channel,} \\
T_{end} &= \text{time of the simulation} = 500 \text{ minutes,} \\
Q_b^n(x) &= \text{the discharge of the base solution at } x \text{ at time } t^n, \\
Q_b^{max}(x) &= \text{the maximum discharge of the base solution at discharge at } x, \\
T_b(x) &= \text{the arrival time of the center of gravity of the base solution at } x, \text{ and} \\
h_b(x,t) &= \text{the flow depth of the base solution at } x \text{ at time } t.
\end{aligned}
$$

The integrals in (3.21) and (3.23) were evaluated numerically using Simpson's method. The base solution was computed using $\Delta x = 20$ ft, $\Delta t = 2$ s, and five iterations for the friction term. Resulting values for the maximum flow, Q_b^{max}, and arrival time, T_b, at $x = 50,000$ ft are 510.3 ft^3/s and 363.0 minutes, which are in close agreement to those obtained by Smith (1997) using a similar semi-implicit method.

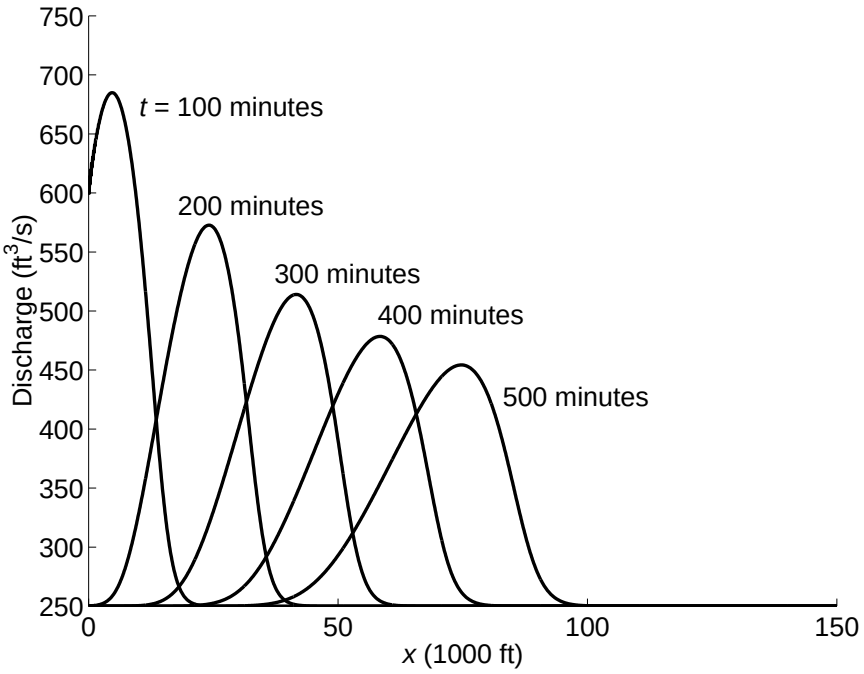

Figure 3.22: Time evolution of a hydrograph advected downstream in the hydrograph routing test case

Before examining the convergence of errors with decreases in Δx and Δt, some preliminary tests were run to examine error convergence with the number of iterations used

to evaluate the frictional term. Results are presented in Figure 3.23 for RMS error using $\Delta x = 1000$ ft, $\Delta t = 200$ s, and the slope limiter method. All runs in this section are made using the dynamic choice for advection discretization. The calculation made without using χ or any iterations was unstable and is not plotted. The use of the parameter χ is seen to greatly reduce oscillations of the solution with successive iterations. Although the method using χ and no iterations produced a lower RMS error than with one or more iterations, this is not the case generally and one iteration was shown to improve error in tests made at different values of Δx and Δt. For all but the coarsest discretizations examined, one iteration was sufficient to achieve convergence, a fortuitous result for model efficiency.

The results of error convergence with decreasing Δx and Δt are shown in Figures 3.24 to 3.27. The maximum advection CFL condition during the simulation was held constant at 0.23 for all runs, and the error measures were plotted as a function of the space discretization, Δx; the associated maximum gravity wave CFL condition was approximately 1.24. In Figure 3.24, the solution obtained with slope limiters is seen to have lower RMS error at every discretization value tested. Since the slope limiters impart almost second order accuracy to the method, this result was expected. In Figure 3.25, the first order upwind method is shown to be highly diffusive at coarse discretizations. This is a well known property of the upwind method. On the other hand, the slope limited method produced positive amplitude errors for medium level discretizations. Although this has potentially negative consequences for model stability, the method was stable for all runs performed here. Figure 3.26 presents the phase error for both methods. The trapezoidal time discretization used by both methods is known to display a phase lag (Durran, 1999). The performance on the test case illustrated here is more complicated, but both methods produce relatively small phase errors. As illustrated in Figure 3.27, the mass preservation error for both methods is very good, with the upwind method losing mass on the order of machine precision for high resolution runs. Upwind and slope limited solutions are plotted with the base solution in Figure 3.28 to visually demonstrate the better fit of the slope limited solution at a given discretization level.

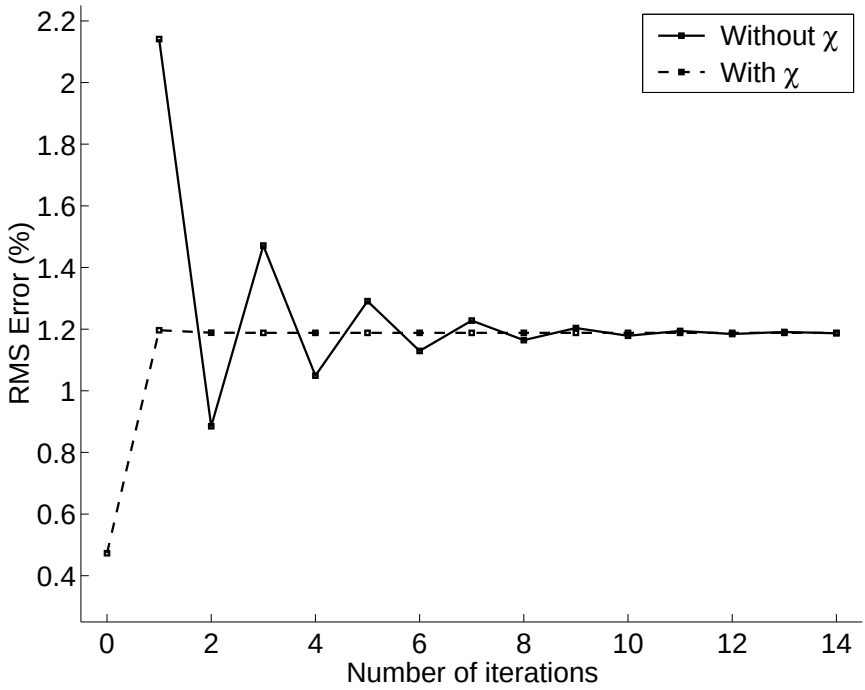

Figure 3.23: RMS error convergence with number of iterations

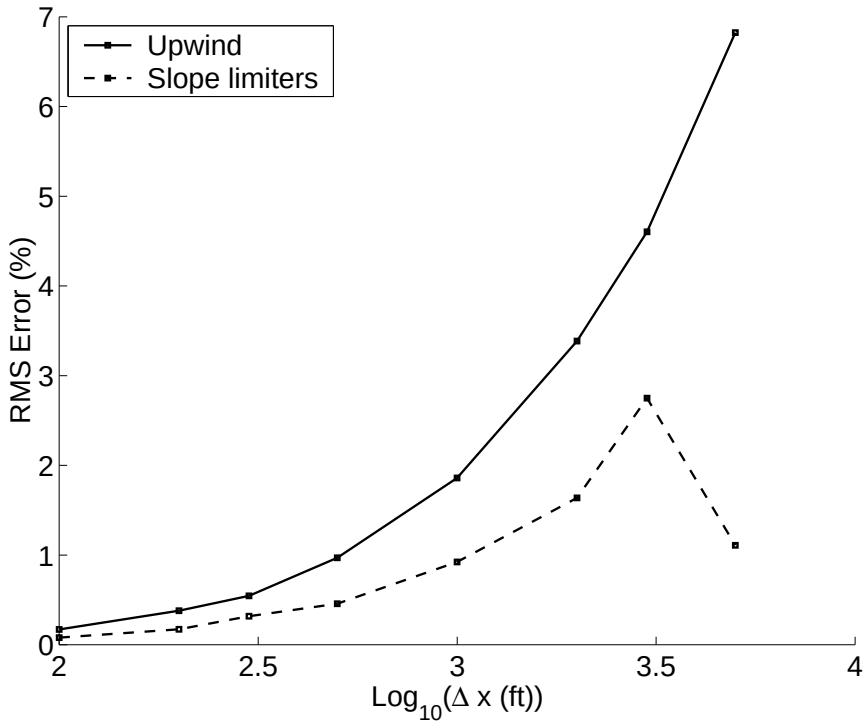

Figure 3.24: RMS error convergence with increasing time and space resolution

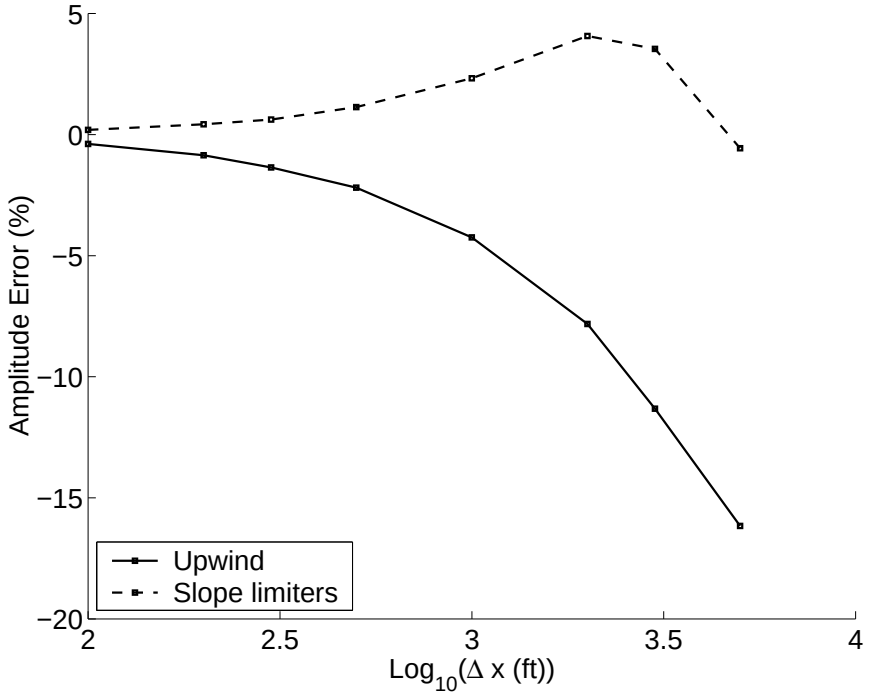

Figure 3.25: Amplitude error convergence with increasing time and space resolution

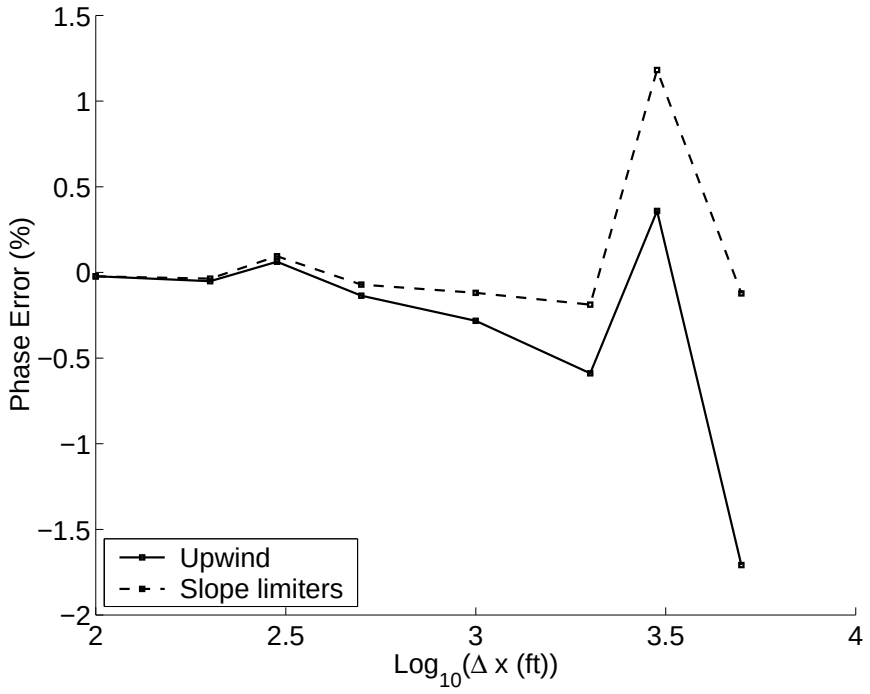

Figure 3.26: Phase error convergence with increasing time and space resolution

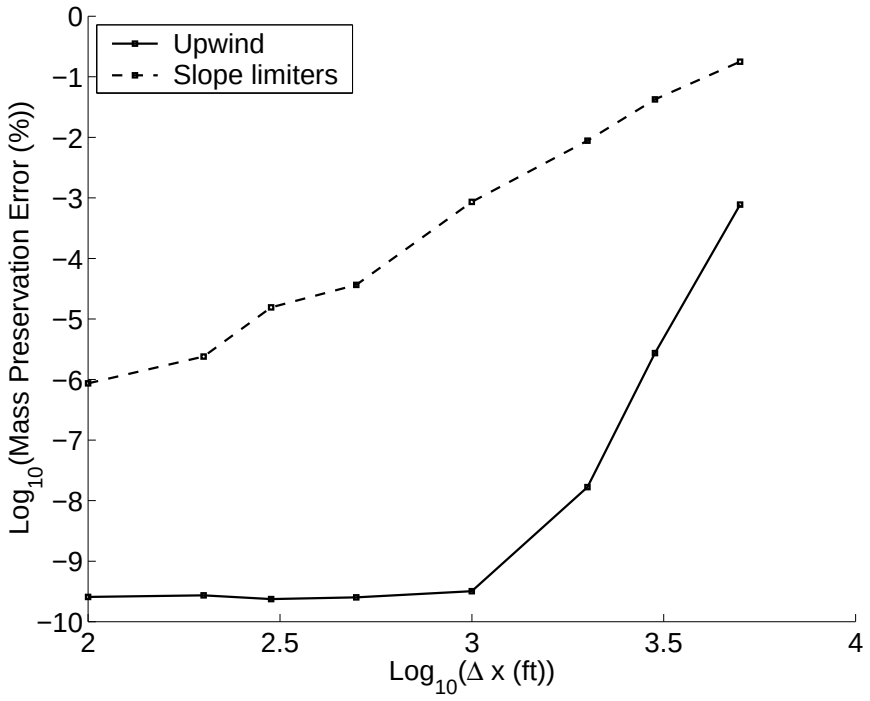

Figure 3.27: Mass preservation error convergence with increasing time and space resolution

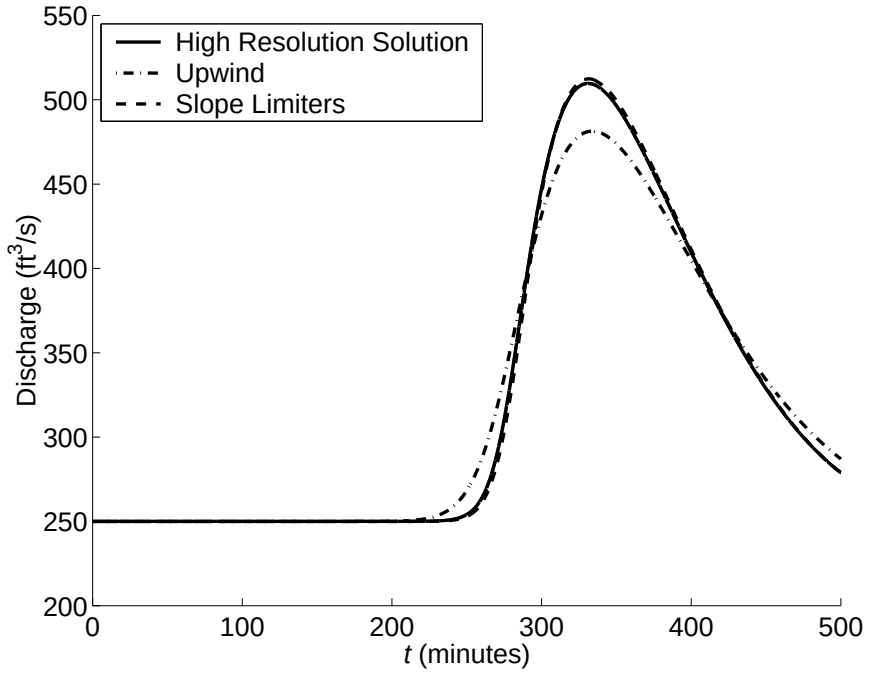

Figure 3.28: Comparison of upwind and slope limited solutions with $\Delta x = 1000$ ft and $\Delta t = 25$ s

3.4 Two-dimensional approximations

3.4.1 Discretization methods

The procedure for discretizing the full 2D mass and momentum equations presented in (2.49) to (2.51) is analogous to the procedure previously outlined for 1D flow. The semi-implicit approach is followed again, whereby the water surface gradient and bottom friction terms in the momentum equation are treated implicitly and all other terms are treated explicitly. The domain is discretized on the 2D staggered grid shown in Figure 3.29, which is analogous to the 1D staggered grid presented previously. Velocities in the x direction are defined at the centers of the East-West cell edges and velocites in the y direction are defined at the centers of the North-South cell edges. The water surface elevations, depths, scalar concentrations, and bottom friction factors are defined at the cell centers.

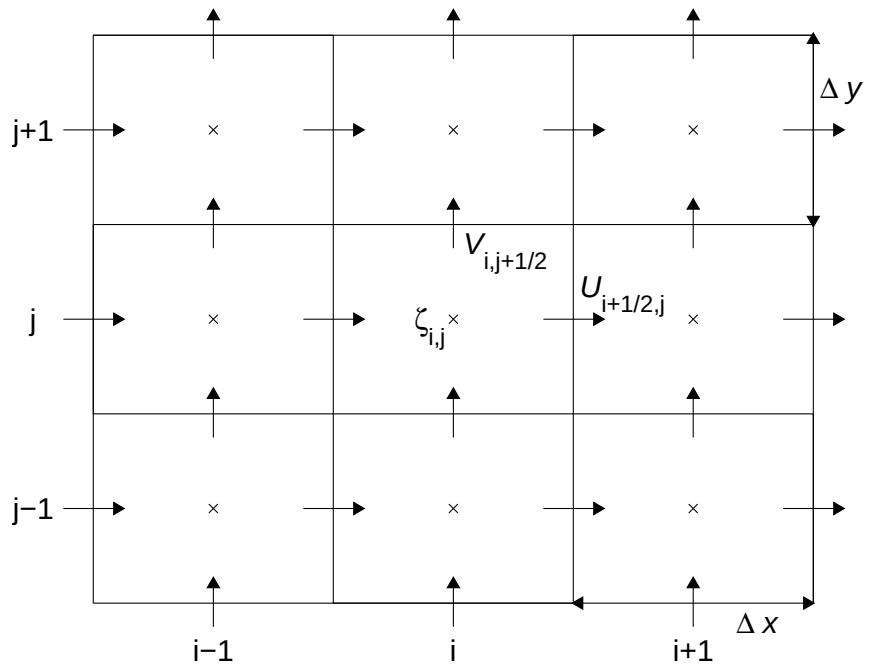

Figure 3.29: Staggered 2D grid

The complete 2D mass and momentum equations are discretized as

$$\begin{aligned} \frac{\zeta_{i,j}^{n+1}-\zeta_{i,j}^{n}}{\Delta t} &+ \frac{{}^{*}h_{i+1/2,j}^{n}U_{i+1/2,j}^{n+1/2}-{}^{*}h_{i-1/2,j}^{n}U_{i-1/2,j}^{n+1/2}}{\Delta x} \\ &+ \frac{{}^{*}h_{i,j+1/2}^{n}V_{i,j+1/2}^{n+1/2}-{}^{*}h_{i,j-1/2}^{n}V_{i,j-1/2}^{n+1/2}}{\Delta y} = (HYDRO)_{i,j}^{n} \end{aligned} \tag{3.24}$$

$$\begin{aligned} \frac{U_{i+1/2,j}^{n+1}-U_{i+1/2,j}^{n}}{\Delta t} &+ (ADV_x)_{i+1/2,j}^{n} = -g\,\frac{\zeta_{i+1,j}^{n+1/2}-\zeta_{i,j}^{n+1/2}}{\Delta x} + (WIND_x)_{i+1/2,j}^{n} \\ &- (FRICT_x)_{i+1/2,j}^{n+1/2} + (HDIFF_x)_{i+1/2,j}^{n} + (COR_x)_{i+1/2,j}^{n} \end{aligned} \tag{3.25}$$

$$\begin{aligned} \frac{V_{i,j+1/2}^{n+1}-V_{i,j+1/2}^{n}}{\Delta t} &+ (ADV_y)_{i,j+1/2}^{n} = -g\,\frac{\zeta_{i,j+1}^{n+1/2}-\zeta_{i,j}^{n+1/2}}{\Delta y} + (WIND_y)_{i,j+1/2}^{n} \\ &- (FRICT_y)_{i,j+1/2}^{n+1/2} + (HDIFF_y)_{i,j+1/2}^{n} + (COR_y)_{i,j+1/2}^{n} \end{aligned} \tag{3.26}$$

where the specific discretizations for the hydrologic term in the continuity equation and advection, the surface, bottom, and turbulent stresses, and the Coriolis force are expanded in Appendix B. Collecting explicit terms and denoting them with an overhat leads to the simple set of equations for the dependent variables at time level $n+1$. The expanded form of the explicit terms is given at the end of Appendix B.

$$\zeta_{i,j}^{n+1} = \widehat{\zeta}_{i,j} + \frac{\Delta t}{2\Delta x}\left({}^{*}h_{i-1/2,j}\,U_{i-1/2,j}^{n+1} - {}^{*}h_{i+1/2,j}\,U_{i+1/2,j}^{n+1}\right) \tag{3.27}$$

$$+ \frac{\Delta t}{2\Delta y}\left({}^{*}h_{i,j-1/2}\,V_{i,j-1/2}^{n+1} - {}^{*}h_{i,j+1/2}\,V_{i,j+1/2}^{n+1}\right) \tag{3.28}$$

$$U_{i+1/2,j}^{n+1} = \widehat{U}_{i+1/2,j} + \frac{g\Delta t}{2\Delta x\left(1+\gamma_{i+1/2,j}\right)}\left(\zeta_{i,j}^{n+1}-\zeta_{i+1,j}^{n+1}\right) \tag{3.29}$$

$$V_{i,j+1/2}^{n+1} = \widehat{V}_{i,j+1/2} + \frac{g\Delta t}{2\Delta y\left(1+\gamma_{i,j+1/2}\right)}\left(\zeta_{i,j}^{n+1}-\zeta_{i,j+1}^{n+1}\right) \tag{3.30}$$

Substituting the velocites into the ζ equation produces

$$-sx_{i-1/2,j}\,\zeta_{i-1,j}^{n+1} - sy_{i,j-1/2}\,\zeta_{i,j-1}^{n+1} + rr_{i,j}\,\zeta_{i,j}^{n+1} - sx_{i+1/2,j}\,\zeta_{i+1,j}^{n+1} - sy_{i,j+1/2}\,\zeta_{i,j+1}^{n+1} = (RHS)_{i,j} \tag{3.31}$$

where

$$\begin{aligned} sx_{i+1/2,j} &= \left(\frac{g\Delta t}{2\Delta x}\right)\left(\frac{1}{1+\gamma_{i+1/2,j}}\right)\left(\frac{\Delta t}{2\Delta x}\right){}^{*}h_{i+1/2,j} \\ sy_{i,j+1/2} &= \left(\frac{g\Delta t}{2\Delta y}\right)\left(\frac{1}{1+\gamma_{i,j+1/2}}\right)\left(\frac{\Delta t}{2\Delta y}\right){}^{*}h_{i,j+1/2} \\ rr_{i,j} &= 1 + sx_{i-1/2,j} + sy_{i,j-1/2} + sx_{i+1/2,j} + sy_{i,j+1/2}. \end{aligned}$$

If cells in the domain are numbered from 1 to N, where N is the total number of cells, equation (3.31) can be written for each cell in the domain to obtain a system of equations. If

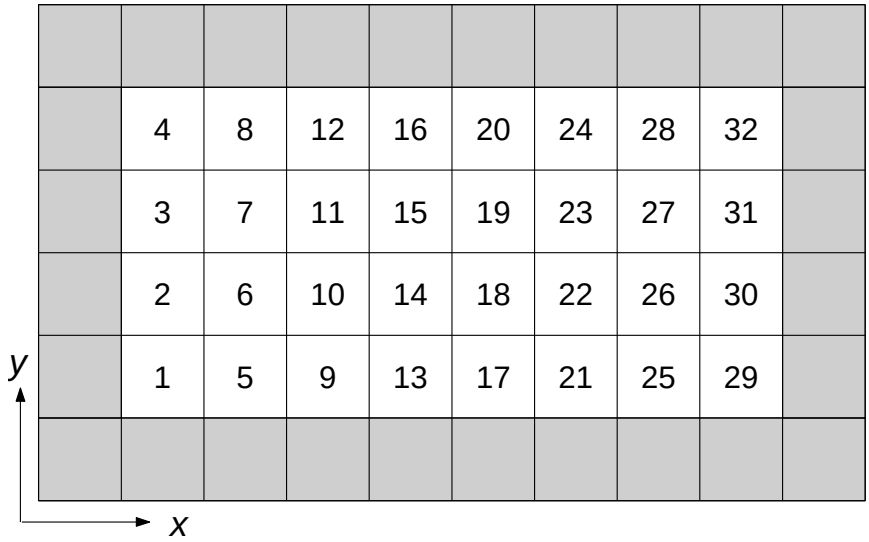

Figure 3.30: Two-dimensional 'natural' ordering method

the 'natural' ordering method is used, where cells are numbered along the shortest dimension of the domain, a penta-diagonal coefficient matrix will result with a tridiagonal band along the main diagonal and two additional diagonal bands displaced above and below the main diagonal by the number of cells in the shortest domain direction (Smith, 1997). An example of such an ordering is shown for a small domain in Figure 3.30 and the resulting coefficient matrix is given in (3.32).

$$\begin{bmatrix} rr_{1,1} & -sy_{1,3/2} & 0 & 0 & -sx_{3/2,1} & 0 & 0 & \cdots \\ -sy_{1,3/2} & rr_{1,2} & -sy_{1,5/2} & 0 & 0 & -sx_{3/2,2} & 0 & \cdots \\ 0 & -sy_{1,5/2} & rr_{1,3} & -sy_{1,7/2} & 0 & 0 & -sx_{3/2,3} & \cdots \\ 0 & 0 & -sy_{2,7/2} & rr_{1,4} & 0 & 0 & 0 & \cdots \\ -sx_{3/2,1} & -sx_{3/2,2} & 0 & 0 & rr_{2,1} & -sy_{2,3/2} & 0 & \cdots \\ 0 & -sx_{3/2,2} & 0 & 0 & -sy_{2,3/2} & rr_{2,2} & -sy_{2,5/2} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{3.32}$$

A realistic model domain is not usually rectangular, but the procedure for solving an irregular domain is the same as for a rectangular one. The irregular domain is enclosed by a bounding rectangular region. Cells that are within the rectangular region but outside

the domain are represented with a value of unity on the main diagonal and zeros for all other elements in the coefficient matrix row. The continuity equation is then reduced to $\zeta_{i,j}^{n+1} = \zeta_{i,j}^{n}$ and an artificial water surface elevation can be maintained for cells outside the domain (Smith, 1997).

3.4.2 Matrix solver

The coefficient matrix (3.32) can be solved efficiently using the preconditioned conjugate-gradient method. The free-source software package, NSPCG, developed at the Center for Numerical Analysis at the University of Texas, Austin implements the preconditioned conjugate-gradient method and allows the user a choice of preconditioners and accelerators (Kincaid *et al.*, 1989). This software package has been sucessfully applied by Smith (1997) in a 3D geophysical flow model, who found the best preconditioner to be the modified incomplete Cholesky and the best accelerator to be the conjugate-gradient accelerator through numerical testing. The initial estimates for the water surface solution at time t^{n+1} are taken from the values calculated at the previous time step. Once the water surface elevations are known, horizontal velocities can be calculated using (3.29) and (3.30). The process is then repeated to improve the treatment of the bottom friction term. Boundary conditions on ζ can be applied easily by modifying the coefficients in (3.32) and the right hand side vector.

3.4.3 Scalar advection algorithm

Once the velocites are known at time t^{n+1}, the advection and diffusion of a scalar tracer can be calculated. The method chosen for this discretization is known as the Corner Transport Upwind (CTU) method, as described by Leveque (1996). This method is known as a high-resolution method; although the CTU method is only first order convergent, it produces results that are more accurate than the regular upwind method, which is also first order convergent, especially in areas where flow is not directed along the coordinate axes. The method of scalar advection in the regular 2D upwind method is shown in Figure 3.31. In this figure, the shaded box represents where the mass of fluid that is in cell (i, j) at time t^{n+1} is at the previous time step. The upwind method calculates fluxes (shown as the barred areas in the figure) through the cell edges assuming that all the fluid in cell (i, j) at time t^{n+1}

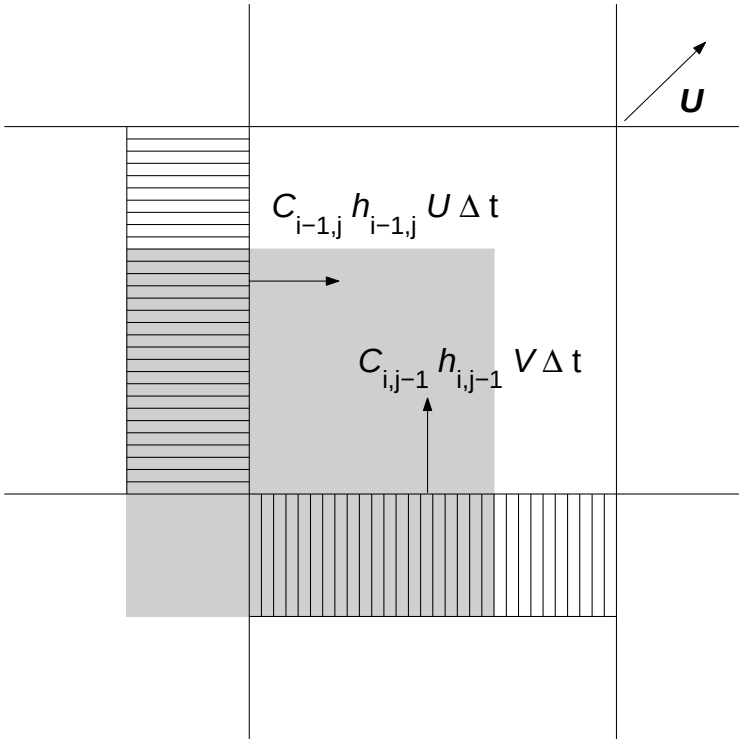

Figure 3.31: Upwind method of scalar transport

came from cells $(i-1, j)$ and $(i, j-1)$. Fluxes out of the cell across the upper boundaries are calculated similarly. A better method to calculate the fluxes into cell (i, j) can be devised by adding the contribution to the flux by cell $(i-1, j-1)$ and subtracting the fluxes from the $(i-1, j)$ and $(i, j-1)$ cells that do not cross the boundaries of cell (i, j). This idea forms the basis for the CTU scheme and is illustrated in Figure 3.32. The areas in triangles labeled **A** are added to the fluxes across the bottom and left boundaries, and the areas labeled **B** are subtracted. These are known as the corrective terms.

To implement the CTU method, the conservative form of the scalar transport equation (2.52) is first considered,

$$\frac{\partial (hC)}{\partial t} + \frac{\partial (hUC)}{\partial x} + \frac{\partial (hVC)}{\partial y} = \frac{\partial}{\partial x}\left(h\,\Gamma \frac{\partial C}{\partial x}\right) + \frac{\partial}{\partial y}\left(h\,\Gamma \frac{\partial C}{\partial y}\right) \quad . \tag{3.33}$$

The conservative form is used because numerical solutions to (3.33) are known to better conserve mass than methods operating on the advective form (2.52). The turbulent diffusion coefficient, Γ, and dispersion coefficients, k_x and k_y, have the same form and are combined into a single parameter for simplicity in the derivation of this numerical method.

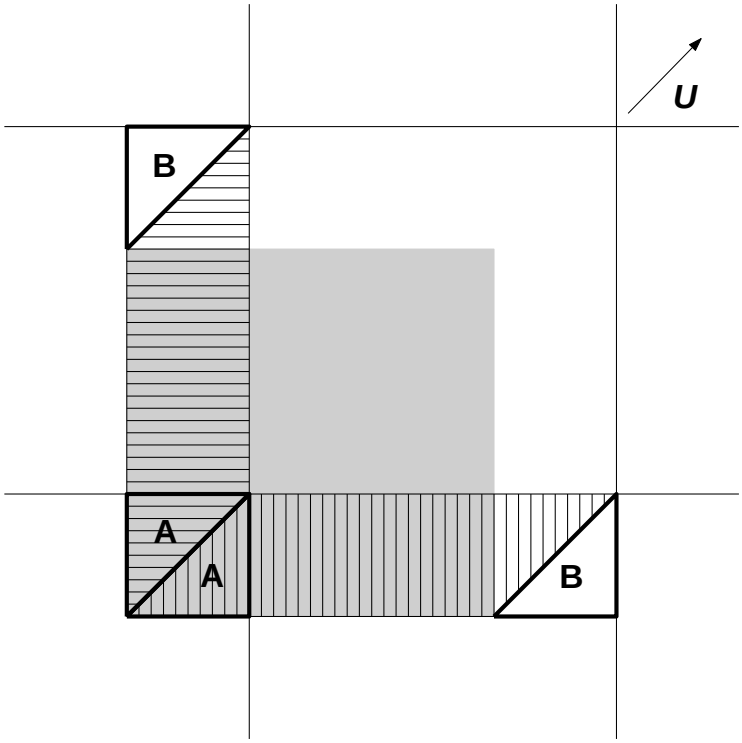

Figure 3.32: CTU method of scalar transport

Equation (3.33) is discretized as

$$h_{i,j}^{n+1} C_{i,j}^{n+1} = h_{i,j}^{n} C_{i,j}^{n} - \frac{\Delta t}{\Delta x}\left(F_{i+1/2,j}^{n} - F_{i-1/2,j}^{n}\right) - \frac{\Delta t}{\Delta y}\left(G_{i,j+1/2}^{n} - G_{i,j-1/2}^{n}\right) + \Delta t\,(SDIFF)_{i,j}^{n}$$

where F and G are the advective fluxes in the x and y directions. For a constant, positive flow field, the advective fluxes are discretized using the standard upwind method as

$$F_{i-1/2,j}^{up} = U\, h_{i-1,j}^{n}\, C_{i-1,j}^{n} \quad G_{i,j-1/2}^{up} = V\, h_{i,j-1}^{n}\, C_{i,j-1}^{n}$$

To obtain the CTU method, the corrective terms are added,

$$\begin{aligned} F_{i-1/2,j}^{ctu} &= F_{i-1/2,j}^{up} - \frac{1}{2} UV \frac{\Delta t}{\Delta y} h_{i-1,j}^{n} C_{i-1,j}^{n} + \frac{1}{2} UV \frac{\Delta t}{\Delta y} h_{i-1,j-1}^{n} C_{i-1,j-1}^{n} \\ G_{i,j-1/2}^{ctu} &= G_{i,j-1/2}^{up} - \frac{1}{2} UV \frac{\Delta t}{\Delta x} h_{i,j-1}^{n} C_{i,j-1}^{n} + \frac{1}{2} UV \frac{\Delta t}{\Delta x} h_{i-1,j-1}^{n} C_{i-1,j-1}^{n} \quad . \end{aligned}$$

This first order CTU method can be extended to 'almost second order' accuracy with the use of slope limiters. A second set of corrective terms is added to the above set of equations

to obtain

$$F^{2nd}_{i-1/2,j} = F^{ctu}_{i-1/2,j} + C((r_{hc})_{i-1/2,j}) \frac{|U|}{2} \left(1 - \frac{\Delta t}{\Delta x}|U|\right) \left(h^n_{i,j} C^n_{i,j} - h^n_{i-1,j} C^n_{i-1,j}\right) \quad (3.34)$$

$$G^{2nd}_{i,j-1/2} = G^{ctu}_{i,j-1/2} + C((r_{hc})_{i,j-1/2}) \frac{|V|}{2} \left(1 - \frac{\Delta t}{\Delta y}|V|\right) \left(h^n_{i,j} C^n_{i,j} - h^n_{i,j-1} C^n_{i,j-1}\right) \quad ,$$

where the higher order terms are switched on in the presence of smooth concentration profiles, and r_{hc} is defined analogous to r_ζ in (3.18) for the product hC. The expanded algorithm for calculating fluxes in non-uniform 2D flow is given in Leveque (1996). It is noted that the full CTU method was shown to be consistent with the continuity equation by Gross *et al.* (2002), a property that is known to improve conservation properties.

To ensure the correct functioning of the coded CTU scheme and to highlight its properties, the method was subjected to a scalar advection test case suggested by Durran (1999). In the test case, a swirling deformation velocity field is used to stretch out an initial blob of tracer. The spatial domain for the test case is $0 \leq x \leq 1$, $0 \leq y \leq 1$, and the initial tracer condition is given by

$$C(x,y) = \frac{1}{2}\left(1 + \cos(\pi R)\right)$$

$$R(x,y) = \min\left(1, 4\sqrt{\left(x - \frac{1}{4}\right)^2 \left(y - \frac{1}{4}\right)^2}\right) \quad .$$

The velocity field of swirling shear flow is

$$U(x,y) = \sin^2(\pi x)\, \sin(2\pi y)\, \cos(\pi t/5)$$

$$V(x,y) = -\sin^2(\pi y)\, \sin(2\pi x)\, \cos(\pi t/5) \quad .$$

The tracer distribution obtained using a high resolution mesh is shown in Figure 3.33 for $t = 0$ and in Figure 3.34 for $t = 2.5$ s, where it is most deformed. At $t = 5$ s, the analytical solution for the tracer distribution is identical to the initial condition. Resulting tracer concentrations for the high resolution mesh after 5 s are shown in Figure 3.35. The mesh chosen for the test case was $\Delta x = \Delta y = 0.02$ and $\Delta t = 0.01$ s. At $t = 2.5$ s, the width of the arc for the analytical solution is approximately 0.05, which is poorly resolved on a grid with spacing 0.02. Solutions are presented below for the upwind method (Figure 3.36), for the second order non-slope limited method, obtained by assuming $C(r) = 1$ in (3.34)

(Figures 3.37 and 3.38), and for the slope limited method (Figure 3.39). In all the graphs, the dashed line indicates the -0.001 contour. The upwind method is shown to be highly diffusive, with a maximum concentration of only 0.1950 at $t = 5$ s. However, the solution never generates negative values, a highly desirable property for scalar transport algorithms. The maximum value at simulation end is 0.7467 for the second order method, but the solution generates highly negative values, with a minimum concentration at $t = 2.5$ s of -0.2249. The CTU solution, obtained using the Van Leer limiter, displays only very slightly negative values and is less diffusive than the first order upwind method.

The turbulent diffusion terms are discretized using

$$\begin{aligned}(SDIFF)_{i,j}^{n} &= \frac{1}{\Delta x^2}\left({}^{*}h_{i+1/2,j}\Gamma_{i+1/2,j}\left(C_{i+1,j}^{n} - C_{i,j}^{n}\right) - {}^{*}h_{i-1/2,j}\Gamma_{i-1/2,j}\left(C_{i,j}^{n} - C_{i-1,j}^{n}\right)\right) \\ &\quad + \frac{\Gamma}{\Delta y^2}\left({}^{*}h_{i,j+1/2}\Gamma_{i,j+1/2}\left(C_{i,j+1}^{n} - C_{i,j}^{n}\right) - {}^{*}h_{i,j-1/2}\Gamma_{i,j-1/2}\left(C_{i,j}^{n} - C_{i,j-1}^{n}\right)\right)\end{aligned}$$

and the concentrations are then calculated by dividing by h. If h drops below the shallow tolerance, ε_{shal}, the cell is considered dry and the tracer concentration is set to a very small, constant value. In numerical experiments, the mass of tracer material lost in this way has never exceeded a few hundreths of a percent of the total mass input.

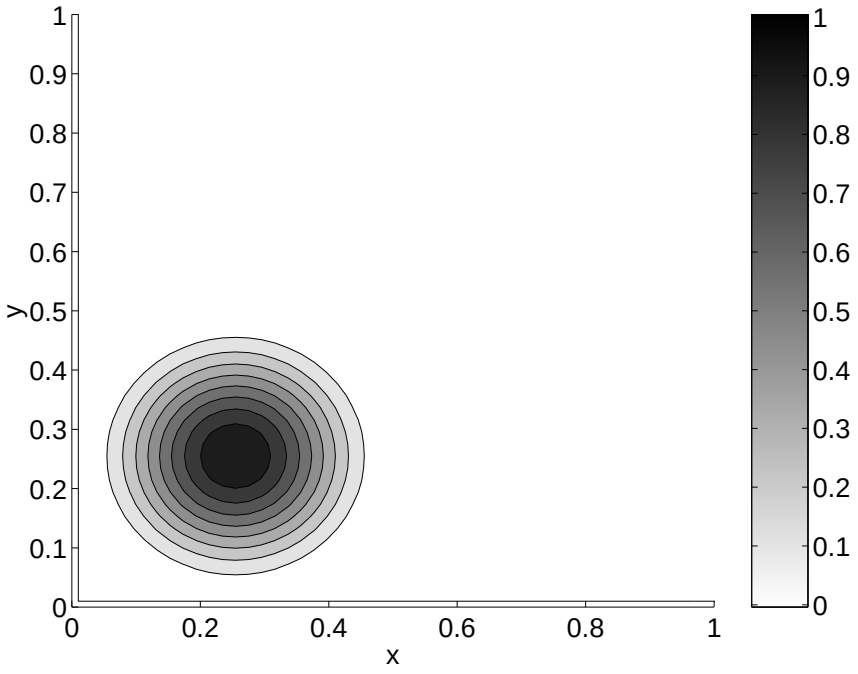

Figure 3.33: Initial tracer distribution

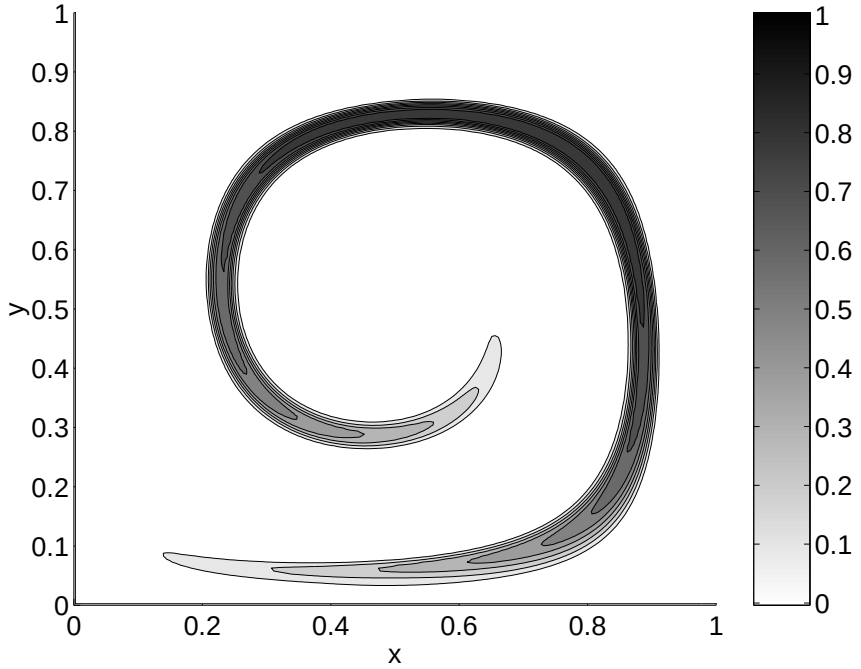

Figure 3.34: Tracer distribution at 2.5 seconds for a high resolution simulation

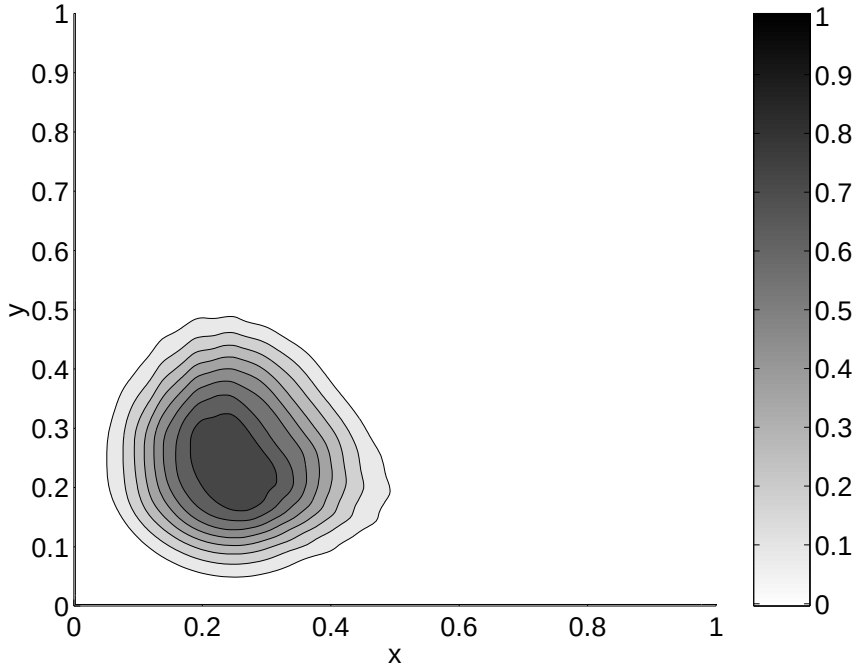

Figure 3.35: Tracer distribution at 5.0 seconds for a high resolution simulation

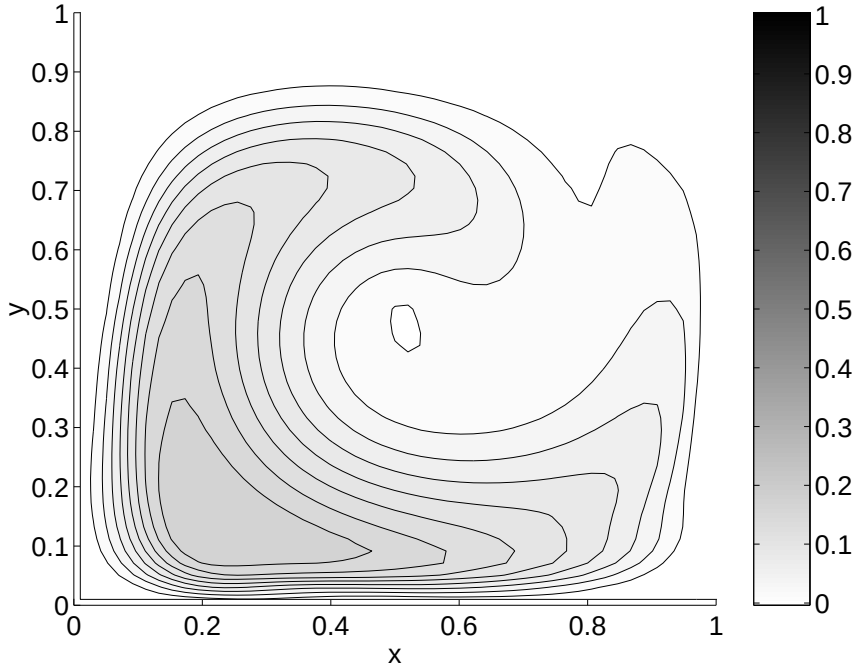

Figure 3.36: Tracer distribution for the upwind method after 5 seconds

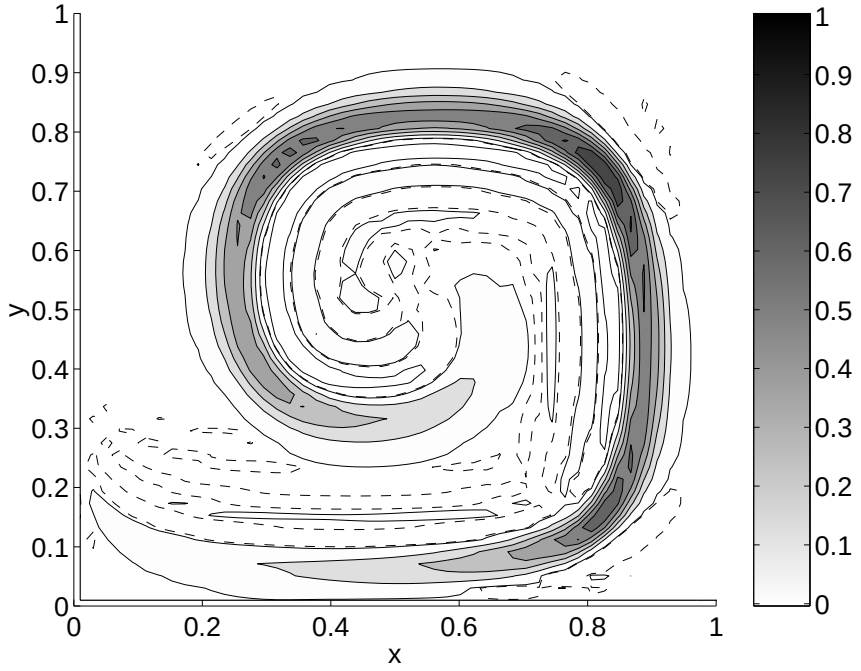

Figure 3.37: Tracer distribution for the second order method at 2.5 seconds

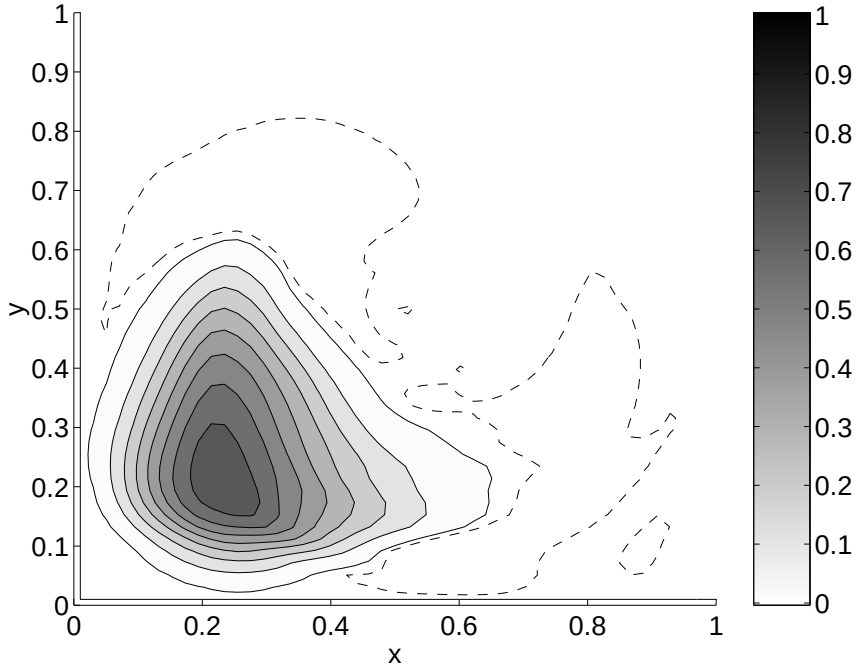

Figure 3.38: Tracer distribution for the second order method after 5 seconds

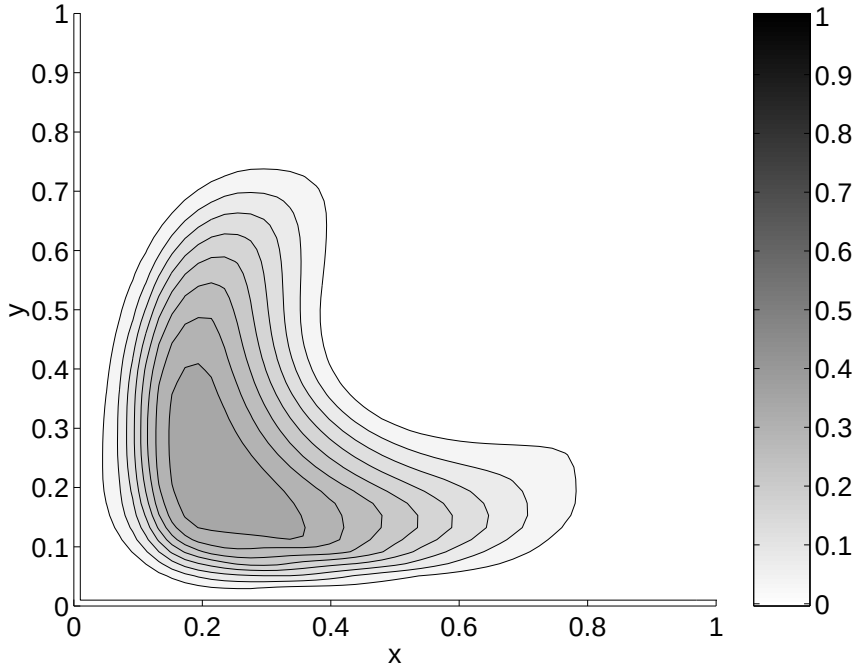

Figure 3.39: Tracer distribution for the CTU method after 5 seconds

4 Residence Time and Phytoplankton Genesis on a Restored Floodplain

In this chapter the previously developed floodplain model will be applied to examine residence times in a restored floodplain adjacent to the Cosumnes River, CA, USA. A brief description of the study site will be followed by model calibration, validation, and a description of model modifications made during these two processes. A series of tracer release experiments will then be used to examine the spatial and temporal distribution of residence times in the Cosumnes floodplain during a flood event. Results from these tests will be compared to observed chlorophyll concentrations. Finally, results from a hypothetical set of floodplain draining simulations will be presented and floodplain management implications discussed.

4.1 Study Area

The Cosumnes River watershed is located southeast of Sacramento, CA and drains approximately 1989 km^2 of land in the western Sierra Nevada mountain range and Central Valley, 84% of which is below 1500 m in elevation (Booth *et al.*, 2006). Because of its relatively low elevation, runoff from winter rainfall plays a more important role in the seasonal hydrologic cycle than for adjacent basins having greater snowpack (Booth *et al.*, 2006). The Cosumnes River maintains a relatively unimpaired hydrograph, and is differentiated from other southern Sierra streams by its conspicuous lack of large dams. The Cosumnes joins the Mokelumne River about 5 km downstream of the floodplain study site, which eventually empties into the San Francisco Bay (Figure 4.1).

Prior to anthropogenic disturbance, the lower Cosumnes River was a low-gradient, anastamosing system with frequent avulsion events and seasonal innundation of much of its floodplain during the winter and spring months (Florsheim & Mount, 2002). Since then the system has been greatly impacted by forest clearing, the filling of floodplain lakes, marshes, and sloughs, gradation of floodplain land for agriculture, and levee construction (Florsheim & Mount, 2002). The result is a straightened and incised main channel without hydrologic connectivity to the majority of its historic floodplain (Booth *et al.*, 2006).

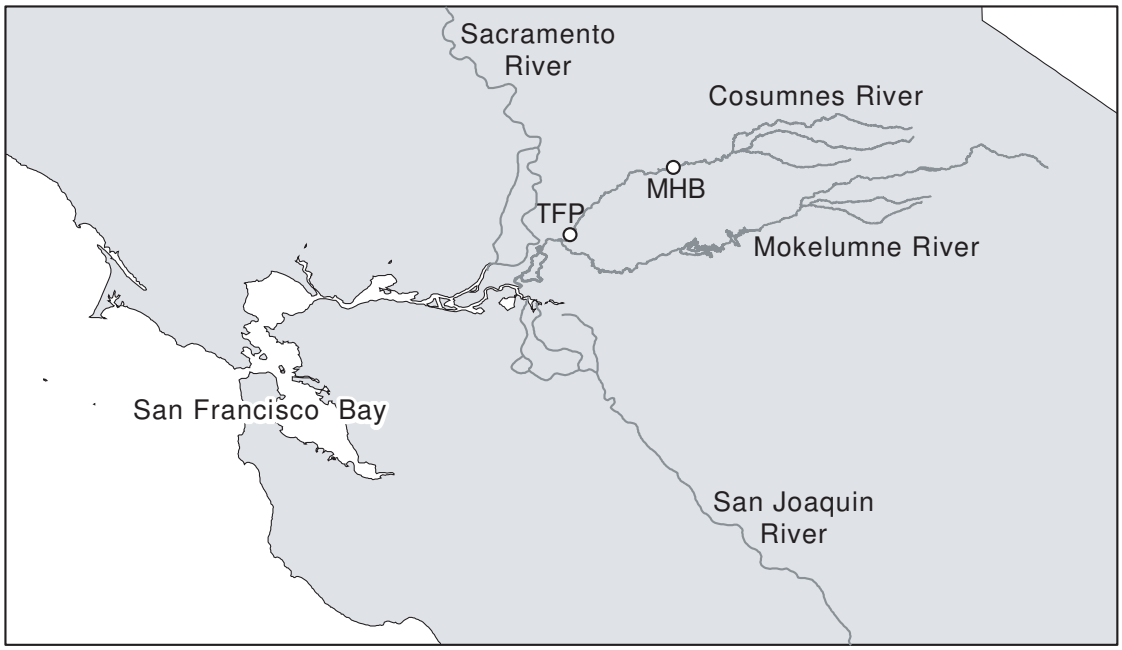

Figure 4.1: Map of Central California hydrology showing relative locations of the Cosumnes River, the Cosumnes Triangle Floodplain (TFP), and the Michigan Bar gauging station (MHB)

The floodplain of interest to this study is located in the Cosumnes River Preserve, about 34 km south of Sacramento, and is collectively managed by a group of federal, state, and non-governmental agencies (Ahearn *et al.*, 2006). It is 0.36 km^2 in extent and bordered by levees on all sides to form the right triangle shown in Figure 4.2. The straightened Cosumnes River flows from the East, makes a 90° turn at the northwest corner of the floodplain, and flows south along the western floodplain edge. In 1997, four levee breaches made by the US Army Corps of Engineers restored connectivity between the channel and the floodplain. Two ponds were also excavated by the Corps to slow water velocities and promote habitat heterogeneity (Ahearn *et al.*, 2006). During high flow events, water flows into the floodplain through two breaches in the western levee, designated Triangle North (TN) and Triangle South (TS). Flow is roughly north to south and leaves the floodplain from two breaches in the southern levee, designated Triangle East (TE) and Triangle West (TW). The locations of these breaches are shown in Figure 4.3. Since reconnection was established, the movement of water through the floodplain has transported sediment, organic material, and large woody debris onto the floodplain and has introduced significant topographic changes and vegetative

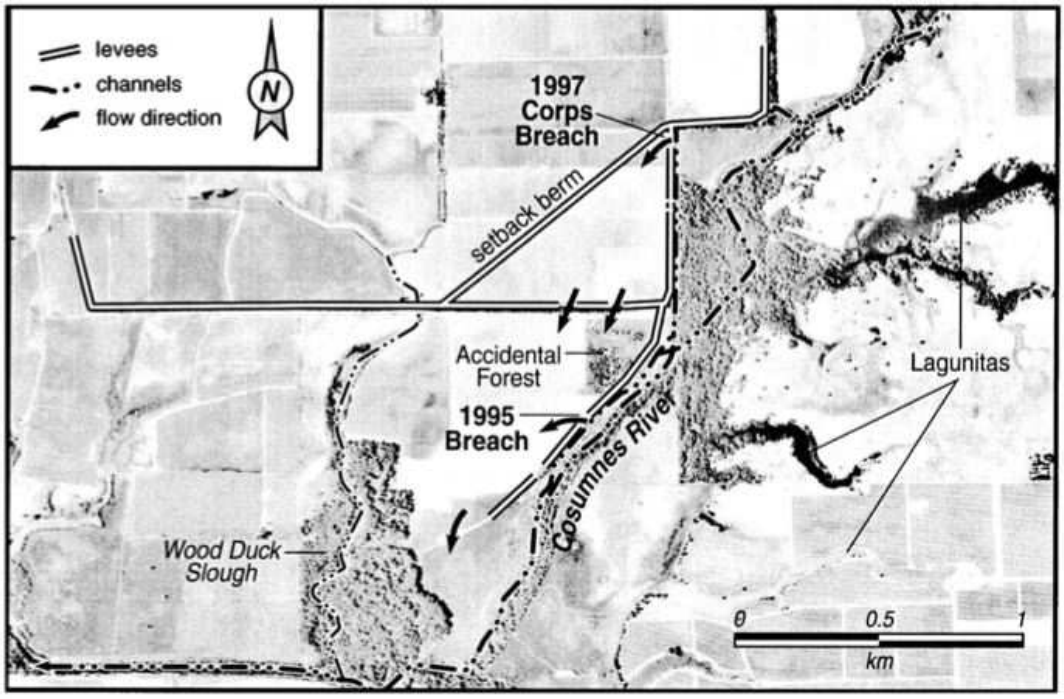

Figure 4.2: Map showing location of Cosumnes Triangle Floodplain field site relative to the Lower Cosumnes River (from Florsheim & Mount (2002))

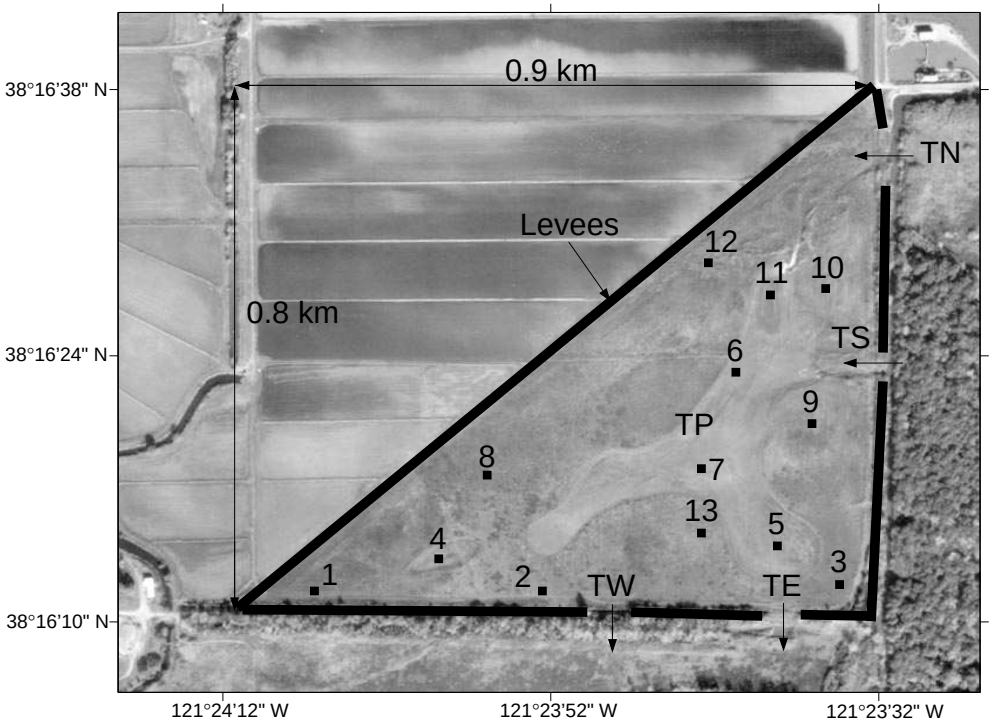

Figure 4.3: Cosumnes floodplain site map showing sampling locations

recruitment of riparian species (Florsheim & Mount, 2002; Ahearn *et al.*, 2006). Flooding typically begins in December, and water levels fluctuate with incoming flood events until late March or April when the floodplain steadily drains and water remaining in the ponds is lost to evaporation or seepage into the subsurface. Water draining from the floodplain rejoins the Cosumnes River a short distance downstream.

From a modeling point of view, the study site has the advantages of easily definable flow boundaries and a large volume of high resolution topographic and vegetation data from a laser altimetry (lidar) survey. The lidar survey was performed in July 2005 and yielded surface elevation and canopy height data with a horizontal spatial resolution of 0.5 m and a vertical accuracy of less than 0.1 m. The resulting values are mapped in Figures 4.4 and 4.5. In Figure 4.4, the ponds dug by the Army Corps can clearly be seen, as can the topographic changes introduced by localized sediment deposition and scour along flow paths from TN to the central pond. In Figure 4.5, much of the aerial extent of the floodplain is observed to be covered by vegetation of less than 1 m, mainly composed of grasses and sedges. Small stands of cottonwood, willow, and oak trees have colonized areas near the TS inlet and parts of the sand splay complex which formed downstream of TN.

As part of a larger project funded by the California Bay-Delta Authority, data have been gathered on the Cosumnes floodplain system over the past several years (Florsheim & Mount, 2002; Booth *et al.*, 2006; Ahearn *et al.*, 2006). Pressure transducers were installed to record water depth at the inlets (TN, TS), outlets (TE, TW), and in the pond (location TP in Figure 4.3) at 10 minute intervals during flood seasons since the late 1990's. The vented Druck® pressure sensors are calibrated to output water depths assuming a hydrostatic pressure distribution. A network of Onset Optic StowAway® thermistors placed throughout the floodplain has recorded water temperature data, and a meteorological station has measured air temperature, humidity, precipitation, and wind speed and direction since water year 2003 (WY2003). Water quality data were collected during WY2005 by Ahearn *et al.* (2006). Spatial distributions of chlorophyll-a, dissolved oxygen, and turbidity were recorded throughout the floodplain during three flood events, and similar parameters were measured hourly at the inlets and outlets for the duration of the flood season. Finally, a tracer study was performed during in March 2004 using the rare earth elements

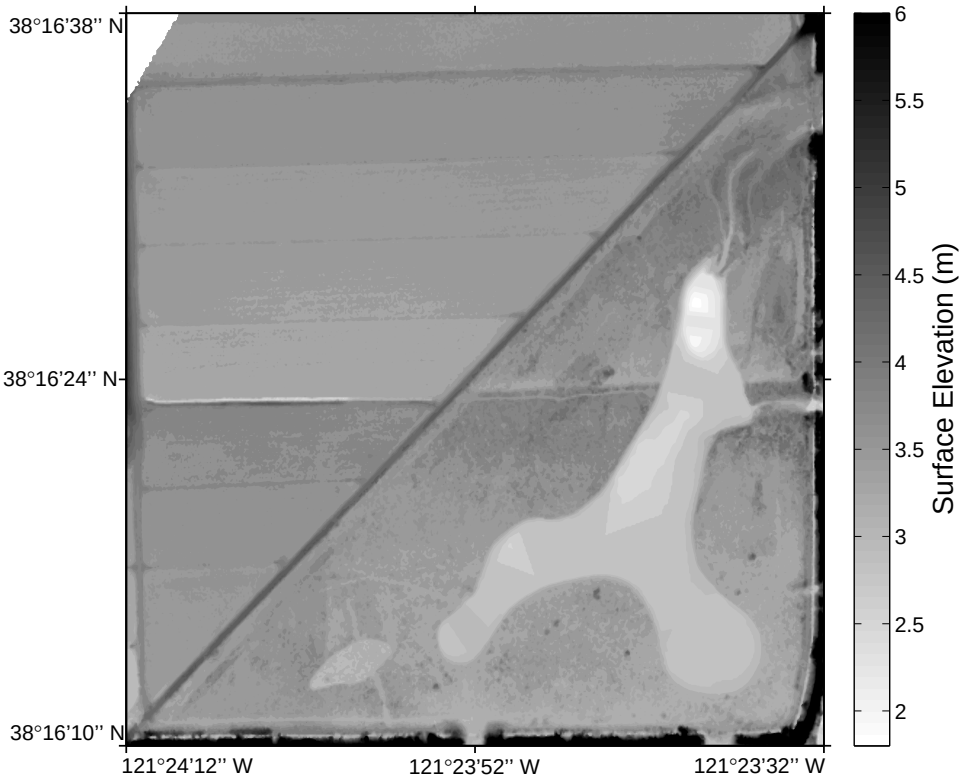

Figure 4.4: Cosumnes floodplain topography

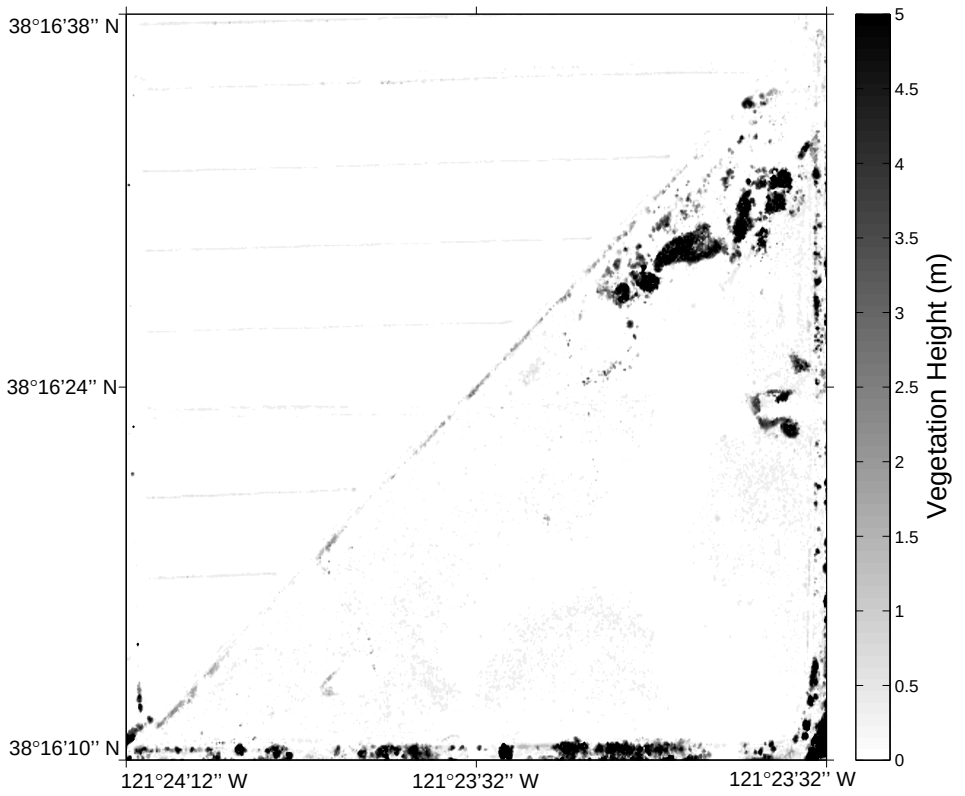

Figure 4.5: Cosumnes floodplain vegetation heights

praseodymium (Pr) and samarium (Sm). Pr and Sm were injected at TN and TS, respectively, and water concentrations were measured at 13 sites in the floodplain for the following 10 days (Schladow, unpublished data). Sampling locations are shown in Figure 4.3.

4.2 Model Calibration and Validation

In this section, the model will be calibrated for specific application to the Cosumnes floodplain using a flood event that occurred in late February 2004. Before these results are presented, however, it is useful examine the validity of using a 2D depth-averaged model for this system. The largest assumption in using a depth-averaged model to predict the transport and fate of a tracer is that the system is well-mixed in the vertical dimension. This assumption was tested during WY2003 with the use of vertical thermistor chains at two locations within the central pond. At both locations, the vertical temperature structure was obtained by securing one thermistor to the bottom while a second thermistor was maintained directly above the first and floating just below the water surface. A time series plot of the surface and bottom temperatures is presented in Figure 4.6 for a thermistor chain deployed near sampling location 11 (see Figure 4.3), the deepest portion of the pond. Short term fluctuations in water temperatures are due to diurnal fluctuations in shortwave radiation, and long term oscillations are due to channel derived inflows from storm events. The surface and bottom temperatures are indistinguishable for the vast majority of the samples. The average difference between the two was 0.08°C, significantly below the smallest temperature difference resolvable by the Onset Optic StowAway® thermistors used. Data from the second location showed similar trends. From these results it is concluded that density stratification due to temperature is not significant in this system and does not hinder turbulent vertical mixing. Flooding in the winter and spring months is fortuitous as greater summer solar radiation may cause stratification but the floodplain is dry.

The flood event chosen for model calibration began at 12:00 PST on February 18 2004 and ended at midnight following February 24. This particular event was chosen because it was a single precipitation event, resulting in a well-defined, non-composite hydrograph. It was also chosen because it was one of the few events during which all four inlet and outlet pressure sensors and the TP sensor were functioning correctly through most of the

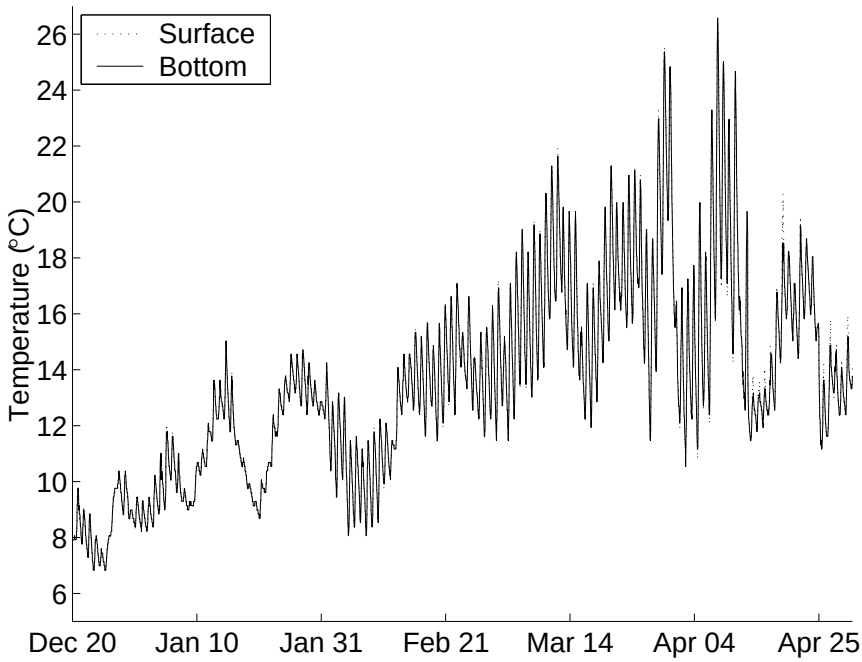

Figure 4.6: Vertical temperature structure at a location in the central pond during WY2003

event. Surface water elevation data are shown for these five gauges along with the channel gauge (denoted CH and installed just upstream of the 90° bend in the channel near TN) in Figure 4.7. Unfortunately, the TP pressure sensor was unable to record accurate data when the water depth in the pond exceeded 6 ft.

To attain a more computationally feasible description of the surface elevation and vegetation heighs, the lidar data was interpolated from 0.5 m resolution to 10 m square grid cells using bilinear interpolation. A grid size of 10 m was chosen because it was the largest cell size that retained predominant features of the topography, including the complex sand splay near the TN inlet, and the inlet jet around TS. Several simulations were run with a 5 m grid for comparison, and no significant improvements in the results were observed. All models were run using a half second time step, corresponding to a maximum CFL number (equation (3.6)) of approximately 0.1. Although this CFL number is well below the required value of 1.0, a half second time step was necessary for model stability. Meteorological data were obtained from the California Irrigation Management Information System (CIMIS) website for a sensor located at Lodi West, about 17 km southeast of the field site. The onsite meteorological station was not functioning correctly during this event. Evapotranspiration

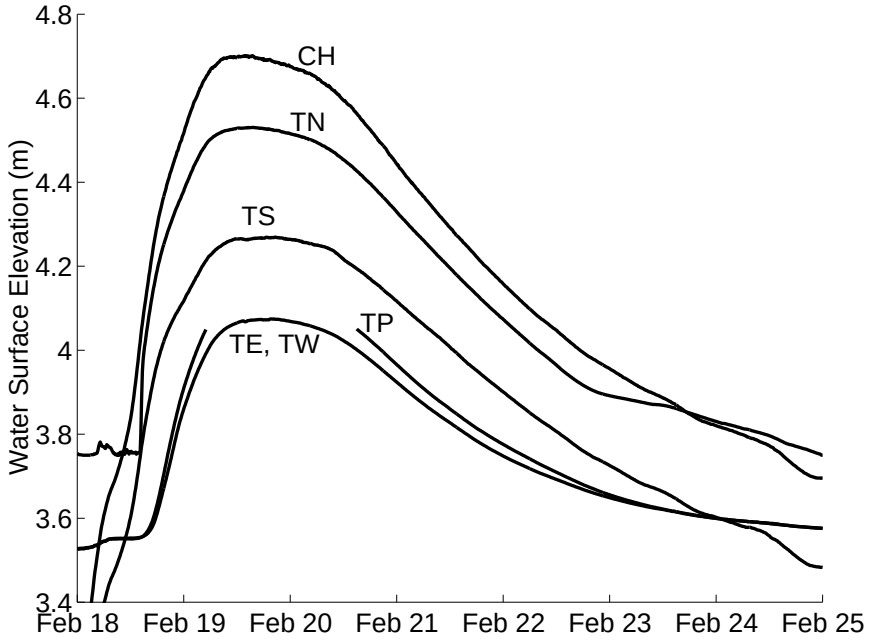

Figure 4.7: Water surface elevations for the flood event used in model calibration

values given by CIMIS were used in the model rather than calculating them using equations (2.24)-(2.27). A constant value of 0.1 m^2/s was used for turbulent viscosity, which is consistent with values used by other researchers (Cobby *et al.*, 2003). Simulations using viscosity values of 0.0 m^2/s and 1.0 m^2/s were also run, and the model proved insensitive to changes in the parameter within this range. A level inital water surface elevation was prescribed with a value equal to the average of the water surface elevations measured at TE, TW, and TP at the start of the event. Initial lateral velocities were set at zero, and the bottom shear stress used in calculation of friction factors for the first model time step was estimated as

$$\tau_b = \rho\, g\, h\, S_0$$

A complete version of the Fortran source code configured to run this event is given in Appendix C. The model took approximately 8 hours to run using a Dell Precision PWS670 workstation with two 2.8 GHz Intel® Xeon processors.

The flood chosen for calibration is historically a small but frequent flood. Booth *et al.* (2006) classified floods on the Cosumnes River for a 98 year period of record based on a USGS gauging station located upstream of the floodplain at Michigan Bar (see Figure 4.1).

Floods were separated into ten groups based on the duration and magnitude of the event. Using this classification system, the calibration flood falls into the group with the smallest magnitude and shortest duration. A flood of this type happened on the Cosumnes once or more in 96% of water years, twice or more in 72% of water years, and three or more times in 54% of the analyzed water years. Although the flood was historically small in magnitude, its large frequency makes it an ecologically relevant flood to examine.

The menu approach to estimating bottom friction factors described in Section 2.6.2 has the benefit that no calibration procedures are necessary for determining Manning's n values. Accurate friction factors should be predicted using only the vegetation heights, a model input, and flow variables such as water depth and velocity, calculated during runtime. Unfortunately, this approach generated Manning's friction factors as high as 0.8 in areas of shallow water, approximately twenty times greater than values used by other researchers in flooding simulations (Horritt & Bates, 2001; Marks & Bates, 2000). The origin of these high values can be seen by examining equation (2.40); for very shallow, slow-moving waters, the Darcy-Weisbach friction factor is $f_{DW} = 1/0.15^2 = 44.4$. Although the menu approach described in 2.6.2 was used successfully in the study by Mason *et al.* (2003), the average water depth at their field site was approximately 1 m. For the simulations run here, the average water depth was closer to 0.3 m. The interim solution adopted for this study was to cut all short vegetation friction values by 75%. This produced more realistic friction values (Manning's n values of only 0.1-0.2 in shallow water), gave a better agreement between the modeled results and the measured data, and preserved relationships between the vegetation height, water depth and velocity, and the friction factors. With this modification, the modeled results at TP matched the sensor recorded values very well (Figure 4.8). The RMS error was 0.30% and the model bias, calculated as the average difference between the predicted and the measured depth, was -0.21% of the reference value. The model is seen to underpredict the water levels on the falling limb of the hydrograph; this could be due to flow from the agricultural field northwest of the floodplain over the southern portion of the setback berm. No gauge was available to estimate flow at this location, but field observations suggest it may be significant.

The spatial distribution of floodplain flow patterns is presented in Figure 4.9. The main

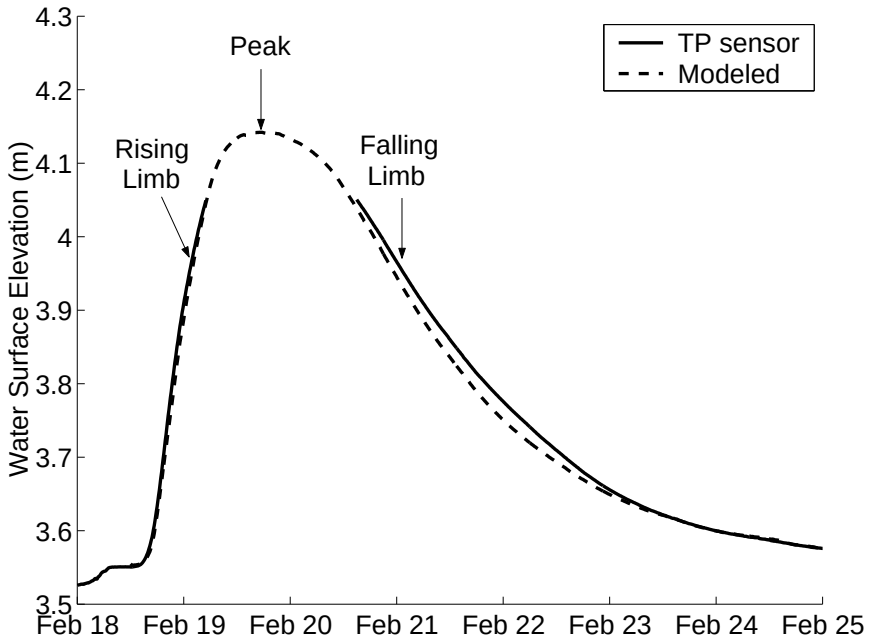

Figure 4.8: Modeled and measured water surface elevations at TP for the calibration flood event

flow paths originate from the inlets, pass through the north-south axis of the pond, and exit more or less equally through the outlets. The resulting flow distribution fits with a general understanding of paths observed at the site by field researchers. Complex topography and vegetation structures near the TN and TS inlets are seen to have large influences on flow paths. The spatially averaged velocities for the floodplain were generally between 5 and 20 cm/s, with local velocities ranging from 0 to 180 cm/s near the inlet jet at TS.

A final check on the validity of the model is made by comparing the measured stage-discharge relationships at the outlets against the calculated ones. Results of this comparison are shown in Figure 4.10. Although the model overpredicts discharges at the TE outlet, the TW results are very close to observed. It is interesting to note that the model results display a hysteresis effect, whereby the rising limb of the hydrograph produces higher discharge values than the falling limb at the same water depths. This occurs because unsteady flow discharge is not only a function of stage, but is also influenced by the water surface gradient and local inertial forces (Henderson, 1996); this hysteresis effect has been observed in natural systems and is not a fictitious result of model procedures (Perumal *et al.*, 2004).

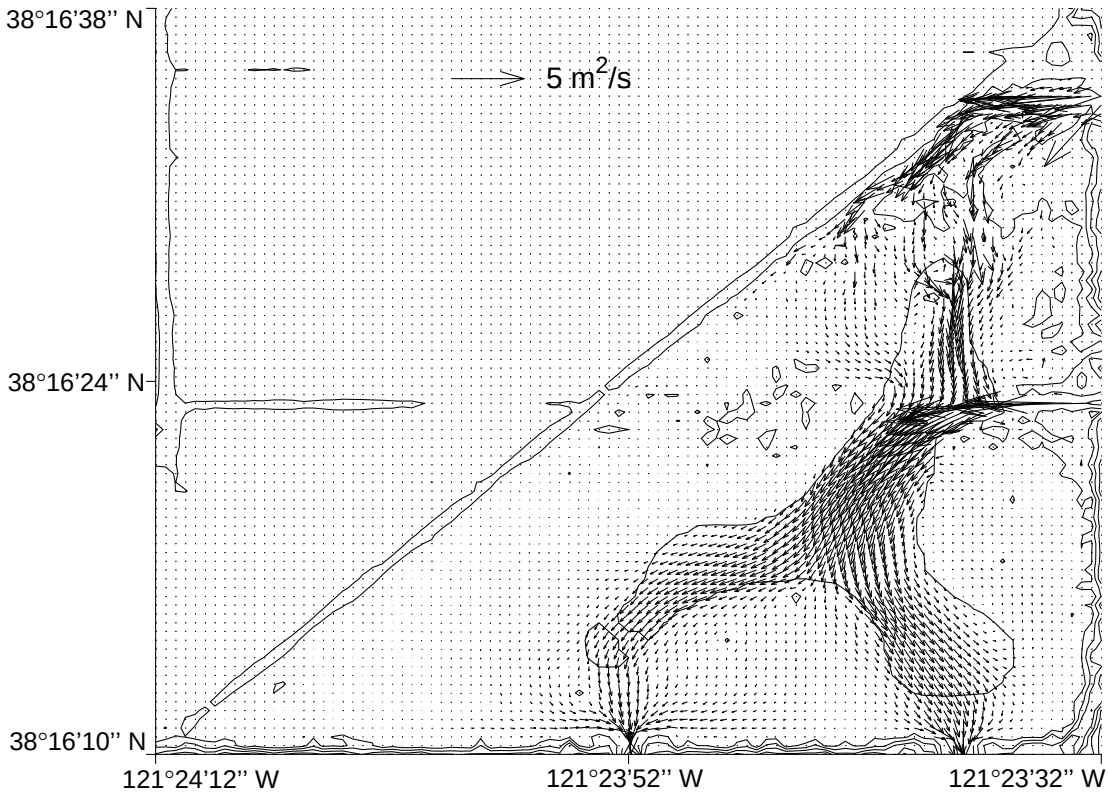

Figure 4.9: Modeled flow paths for the falling limb of the calibration flood event; 0.5 m surface elevation countours are shown

A model validation simulation was run using a flood pulse from later in the WY2004 flood season. Model results are presented along with the measured stage at TP in Figure 4.11. Again, the RMS error is a few tenths of a percent, and the model bias is negative but small. This flood event was additionally used to calibrate the scalar transport algorithm. On March 3 2004, 100 g of the rare earth element salt praseodynium (III) acetate hydrate ($Pr(C_2H_3O_2)_3 \cdot XH_2O$, forumla wt 318.04) were released from the TN inlet. Water concentrations were sampled at 13 locations throughout the floodplain, shown in Figure 4.3, for the following several days. Results for modeled and measured concentrations are presented in Figure 4.12. Unfortunately, only four of the recorded tracer concentrations were significantly above background levels. The accuracy of the scalar transport algorithm was therefore assessed by how well the model represented breakthrough times at each of the locations. A turbulent scalar diffusivity value of 0.1 m^2/s was found to match the data well.

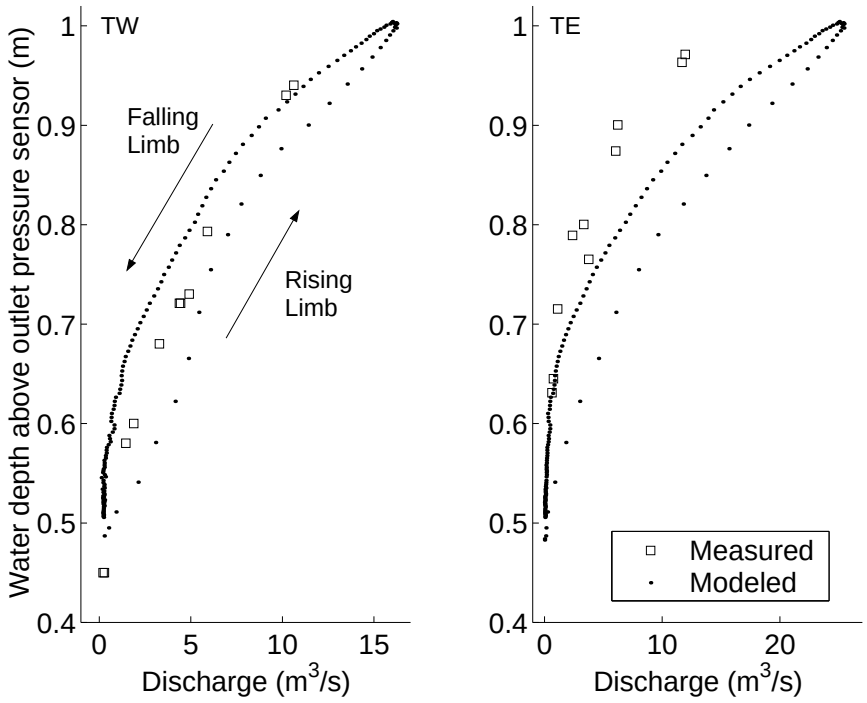

Figure 4.10: Stage-discharge relationships for TE and TW outlets

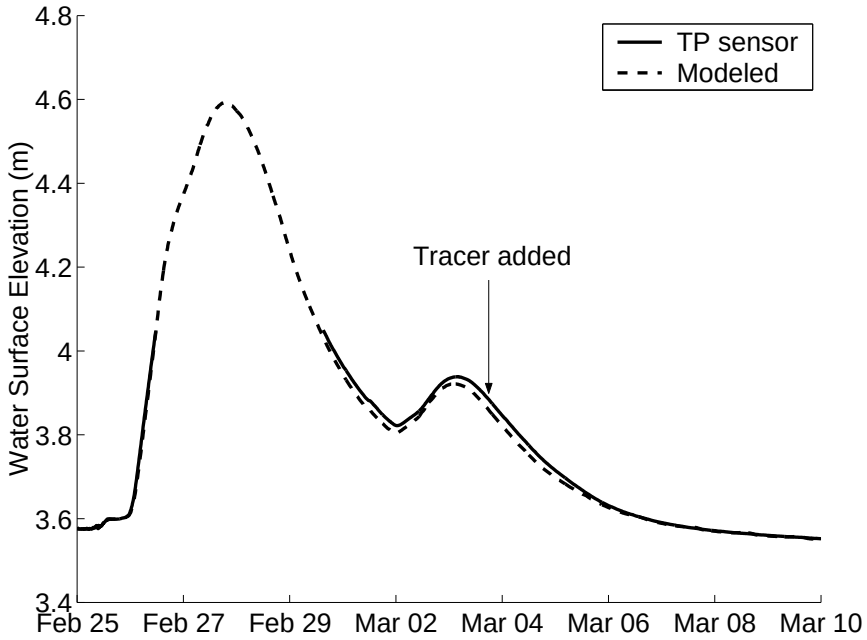

Figure 4.11: Modeled and measured water surface elevations at TP for the validation flood event

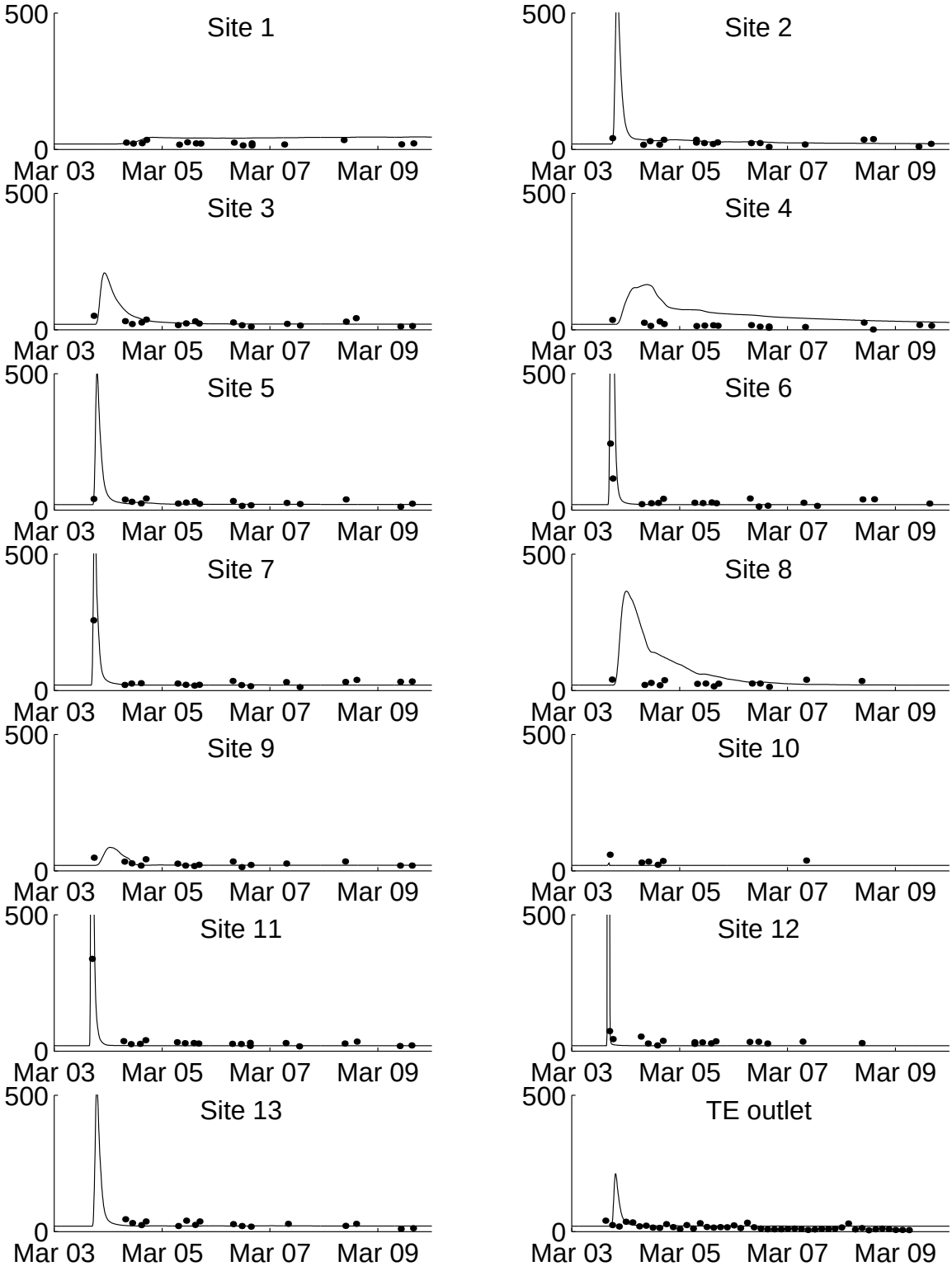

Figure 4.12: Modeled and measured praseodynium concentrations for the validation flood event for sites 1-13 and TE. Concentration units are in pptr.

4.3 Residence Time Distribution

In aquatic systems, nutrients, contaminants, dissolved gases, suspended solids, and most of the living biomass are passively carried with the flow (Monsen *et al.*, 2002). Understanding the hydrodynamics of a system is therefore critical to obtaining an understanding of biogechemical processes, population dynamics, and ecosystem functions. One method of simplifying complex environmental flows to obtain an understandable description is to use the concepts of residence time, flushing time, and mean water age. Each of these concepts has a slightly different connotation and, unfortunately, each can be defined in different manners depending upon the physical system to which it is applied (Monsen *et al.*, 2002). In general, all three concepts attempt to quantify the amount of time a water parcel spends within the boundaries of a defined aquatic system and can be very useful in getting a bulk description of water quality phenomena. In particular, residence time has been linked to numerous processes and water quality observations, including:

- Primary production and eutrophication (Jassby *et al.*, 1990),
- Concentrations of dissolved organic carbon (Christensen *et al.*, 1996),
- Elemental ratios of heavy metals (Hilton *et al.*, 1995),
- Mineralization rates for organic material (denHeyer & Kalff, 1998),
- Occurance of toxic algal blooms (Bricelj & Lonsdale, 1997),
- Export of various zooplankton life stages (Ohman & Wood, 1996), and
- Partitioning of primary production between phytoplankton and macroalgae (Valiela *et al.*, 1997).

Transport time scales are simplified parameters used to examine complex hydrodynamic processes; however, they themselves also vary spatially and temporally. This study will attempt to characterize transport time scales using the approach of Rueda & Cowen (2005), where *mean residence time* is defined as the average time necessary for a particle originating at a specific location (x_0, y_0) at a specific time t_0 to be removed from the system. This definition implies variations in time and space, and the mean residence time, τ_r, can be

defined

$$\tau_r \;=\; \frac{-1}{m_0}\int_0^\infty \frac{dm}{dt}\,t\,dt \;=\; \frac{1}{m_0}\int_0^\infty m(t)\,dt \quad . \tag{4.1}$$

The mean residence time is easily calculated with a hydrodynamic model. An initial mass of conservative tracer, m_0, is released at a specified time and location and the amount of mass remaining in the system at any particular time, $m(t)$, is output from the model at specific intervals. The *mean flushing time* is defined by Rueda & Cowen (2005) as the bulk or integrative description of the transport characteristics of a water body. Instead of tracking an ensemble of particles released at a specific location, an ensemble of particles evenly distributed throughout the system is followed. The mean flushing time therefore does not vary spatially. It can be calculated using (4.1) by prescribing a constant concentration of a conservative tracer in all cells within the computational domain at time t_0 and recording the mass remaining in the system at specified intervals.

It was mentioned above that definitions for transport time scales were often modified for specific application to a particular system. This has been done to cope with complicating factors such as non-steady flow, complex boundaries, and tidal influences (Monsen *et al.*, 2002). In this Cosumnes River floodplain application, a modification of equation (4.1) is necessary in order to account for tracer mass stranded on the floodplain at the end of the simulation. Water levels on the floodplain rise and recede with passing flood events. If residence time on a falling limb of a flood pulse is to be calculated, a certain percent of the initial tracer mass may remain in floodplain lakes and localized depressions. Evaluation of the integral in (4.1) will produce the uninformative value of ∞. The solution adopted here is to calculate the mass of tracer remaining in the floodplain at the end of the simulation and subtract it from the initial mass input. In effect, this procedure calculates the residence or flushing time as if the mass remaining at simulation end was never released at time t_0. The modified residence time equation can be written

$$\tau_r \;=\; \frac{1}{m_0 - m_r}\int_0^\infty \left(m(t) - m_r\right)\,dt \quad , \tag{4.2}$$

where m_r is the mass remaining at simulation end. In most model runs performed in this section the end of the simulation corresponded with total isolation of the floodplain from its channel, and thus the use of (4.2) is consistent with a physical description of the flow. The mass remaining was usually no more than a few of percent of m_0.

To examine the spatial distribution of residence time in the Cosumnes River floodplain, a number of tracer release simulations were performed using the hydrodynamics calculated for the calibration flood pulse. Tracer was released at 26 locations throughout the floodplain, and residence times were calculated using equation (4.2). To remove the dependence of residence time on t, separate releases were made on the rising limb, peak, and falling limb of the hydrograph, as shown in Figure 4.8, and the resulting residence times were averaged. Data was interpolated over the floodplain area using the biharmonic spline interpolation method of Sandwell (1987) to create the map in Figure 4.13. The locations of tracer releases are marked in the figure.

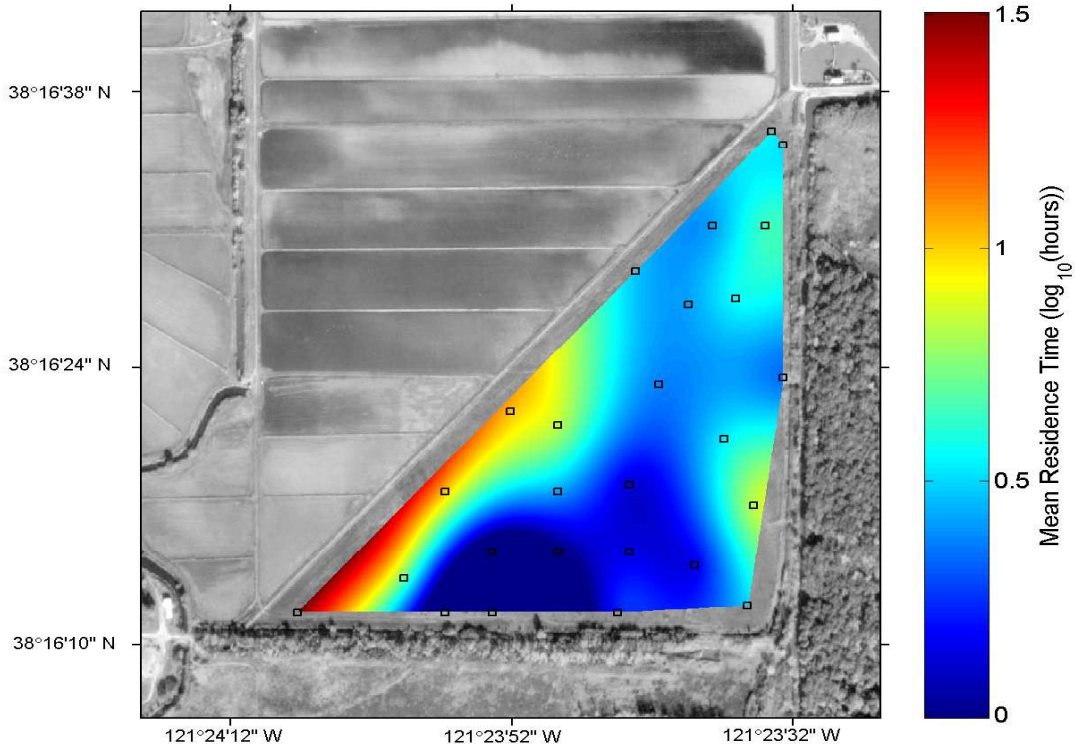

Figure 4.13: Spatial distributions of residence time on the Cosumnes River floodplain. The square symbols designate locations of simulated tracer releases.

In Figure 4.13, a wide distribution in the residence times can be seen. The lowest values are located near the TW outlet and are on the order of minutes. The highest values are found in the southwest corner of the floodplain and are between one and two days. Flow

paths from the TN and TS inlets are clearly visible, and residence times for tracer releases near these points are around 3 to 4 hours. Residence times along the main flow paths for the Cosumnes floodplain are therefore relatively short, even for a small flooding event. Areas of comparitively high residence time are seen adjacent to the eastern levee between TN and TS, and between TS and TE, as well as along the lower half of the setback levee. Chlorophyll-a concentrations measured by Ahearn *et al.* (2006) at the end of a flood event of similar magnitude and duration are shown in Figure 4.14. A correspondence between chlorophyll concentrations and residence time can clearly be seen.

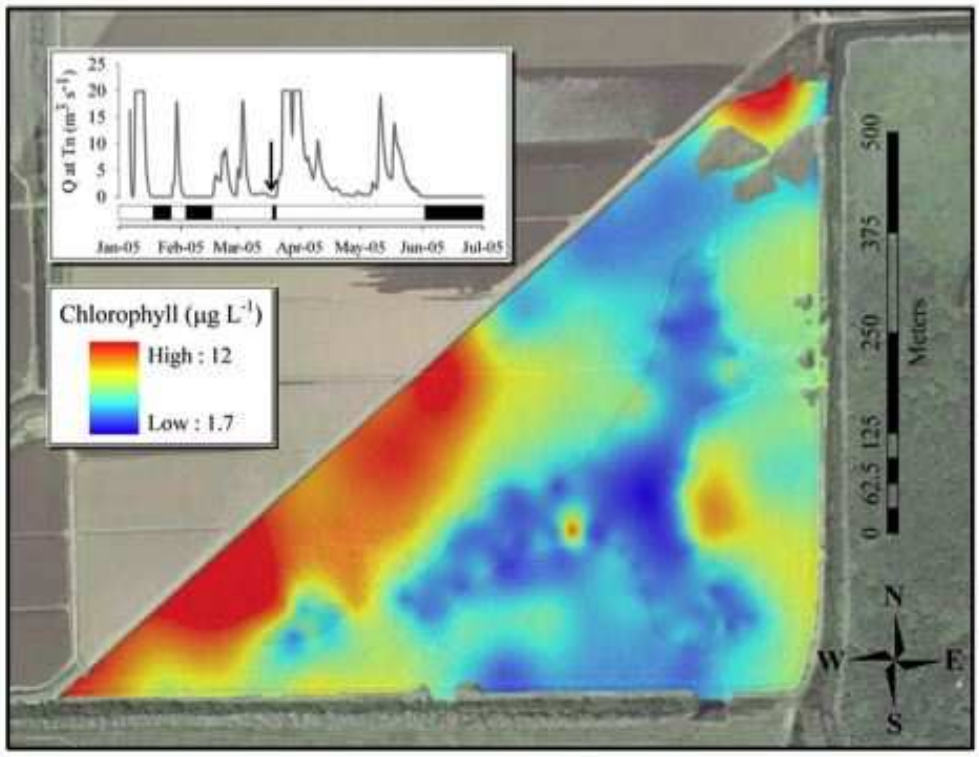

Figure 4.14: Chlorophyll-a concentrations at the end of a small flood event in WY2005 (from Ahearn *et al.* (2006))

The previous discussion and analysis of residence times has focused on the floodplain system during the 'connection' phase, where the floodplain is hydraulically connected with the channel at both inlets and outlets. There are two other relevant phases of floodplain flow: the 'draining' or 'disconnection' phase, where the TN and TS inlets are not flowing but water remaining on the floodplain is steadily draining from TE and TW, and the 'isolation' phase, where any remaining water is totally disconnected from the channel. Although many

ecologically relevant processes occur during the isolation phase, residence times are infinite, no flow is available to tranport matter, and the isolation phase will not be discussed further. The disconnection phase, however, is highly relevant to a number of biogeochemical and ecological processes, including the export of autochthonous carbon downstream. Ahearn *et al.* (2006) showed chlorophyll concentrations to increase dramatically once the disconnection phase began. After inflow from a flood event stops, residence times increase from hours to days and better align with the time scales required for algal growth. The hydrodynamics of this draining phase are used in the following determination of mean flushing times for the calibration flood pulse.

The temporal distribution of transport times through a flood event was assessed with the use of mean flushing time simulations. Separate tracer releases were made at half day intervals throughout the duration of the 6.5 day calibration flood event, and it was assumed that the event was followed by a draining phase unimpaired by subsequent precipitation events and lasting 42 days (an average amount of time based on pressure gauge records over several water years). The results of the flushing time simulations are plotted in Figure 4.15 along with the inflow at TN. A dramatic increase in flushing time is seen beginning on the falling limb and reaching a plateau near the start of disconnection. The mean flushing time at the moment of disconnection was calculated as approximately 3.5 days.

Finally, simulations were run to examine the effects of increasing flood magnitudes on residence times. The boundary conditions shown in Figure 4.7 were artificially increased to create synthetic floods of 2, 3, 5, and 10 times the magnitude of the original calibration flood using measured stage-discharge relationships. Results are graphed in Figure 4.16 for residence time determinations on the rising limb, peak, and falling limb of the hydrograph, and a flushing time determination at the peak. Floods of increasing magnitude showed decreasing residence times as expected. However, the incremental change with Q decreased at higher magnitudes, and the flushing time for a flood with ten times the discharge of the calibration flood was slightly greater than for a flood of only five times the discharge. This result is attibuted to increased transport into high residence time areas of the floodplain at very high discharges.

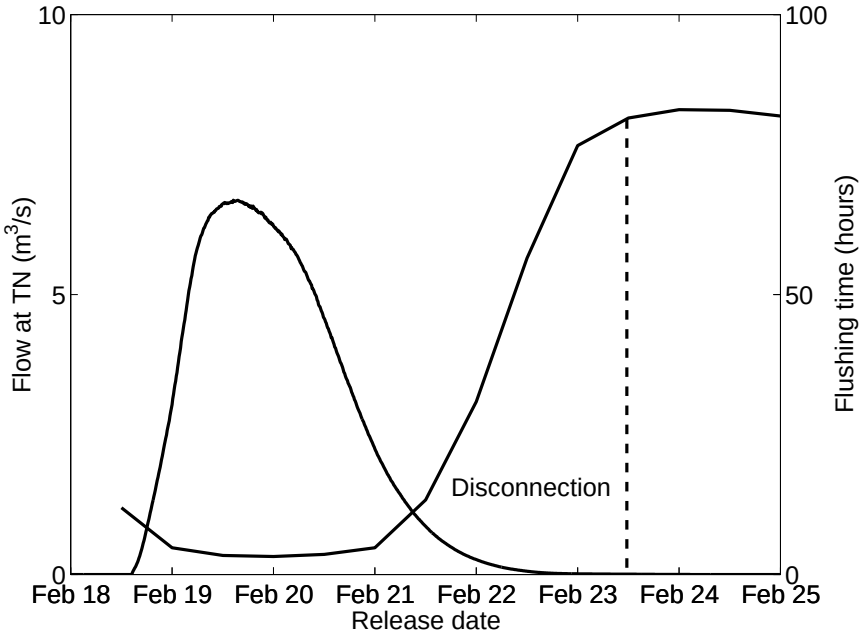

Figure 4.15: Time distribution of flushing times through the calibration flood event. The dotted line indicates the time of disconnection between channel flow at TN and the floodplain.

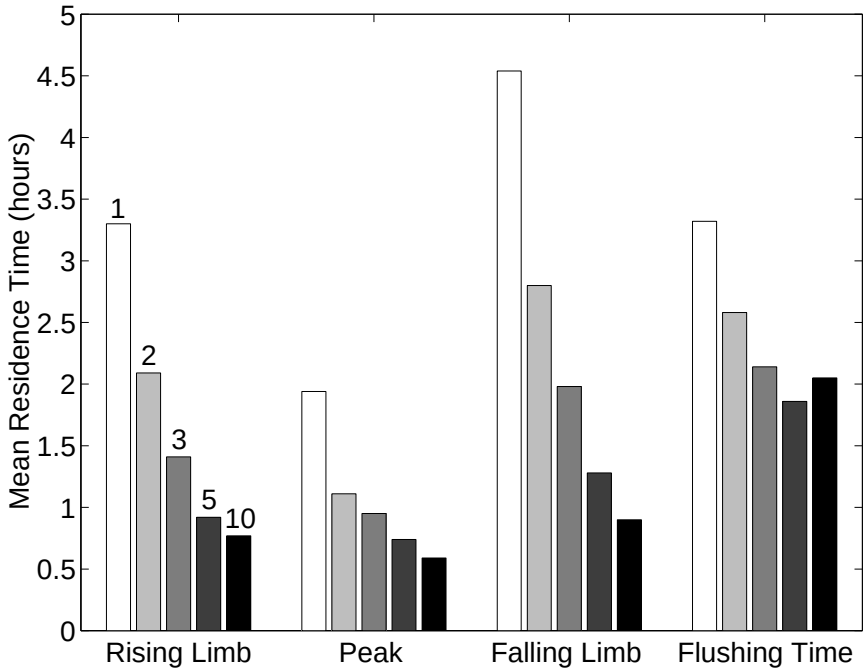

Figure 4.16: Transport time dependence on flood magnitude. The numbers above the first set of bars indicate the multiplier used on the base storm magnitude.

4.4 Implications for Primary Production

Transport time scales and algal growth time scales, which are typically on the order of days to weeks, align during the draining phase of floodplain flow (Ahearn *et al.*, 2006). Residence times calculated for the flood events shown above were on the order of hours for most of the floodplain area. Floods that are smaller in magnitude or longer in duration will have similar transport times during the connection phase. Once the draining phase begins, average transport time scales increase to days and enough time is available for water chemistry to diverge from that of the channel and algal populations to increase significantly. In this section, management scenarios will be investigated assuming that it is possible to control the rate at which water drains from the floodplain. This may be accomplished, for example, by altering floodplain vegetation to influence flow patterns or by installing flow control structures near inlets and outlets. The influence of management scenarios on primary productivity is considered here, with the objective of determining whether floodplain flow may be altered in a way to maximize phytoplankton exports to the downstream San Francisco Bay-Delta ecosystem.

To analyze phytoplankton exports on the draining phase, flushing time simulations were run using boundary conditions and meteorological data from the spring of WY2004, when the floodplain drained without the complication of additional precipitation events in about 42 days (Figure 4.17). Stage boundary conditions for simulations involving faster draining were developed by fractionally increasing the rate of water level drop at TE and TW. Simulations were configured to force the floodplain to drain in 36, 30, 24, 18, and 12 days. The parameter used to estimate algal population growth through the draining phase was calculated using a formula similar to that presented by Rueda & Cowen (2005),

$$\frac{[A]}{[A]_0} = f(t) \qquad \frac{[A]_{out}}{[A]_{in}} = \int_0^\infty f(t)\,\Psi(t)\,dt \quad ,$$

where $[A]$ is the concentration of algae, $[A]_{out}/[A]_{in}$ is denoted the algal increase factor (AIF), $\Psi(t)$ is the residence time distribution function defined by

$$\Psi(t) = -\frac{1}{m_0}\frac{dm}{dt} \quad ,$$

and the function $f(t)$ describes the fractional population increase in algal concentrations. For this study, two general algal growth curves were examined: one reaching a maximum con-

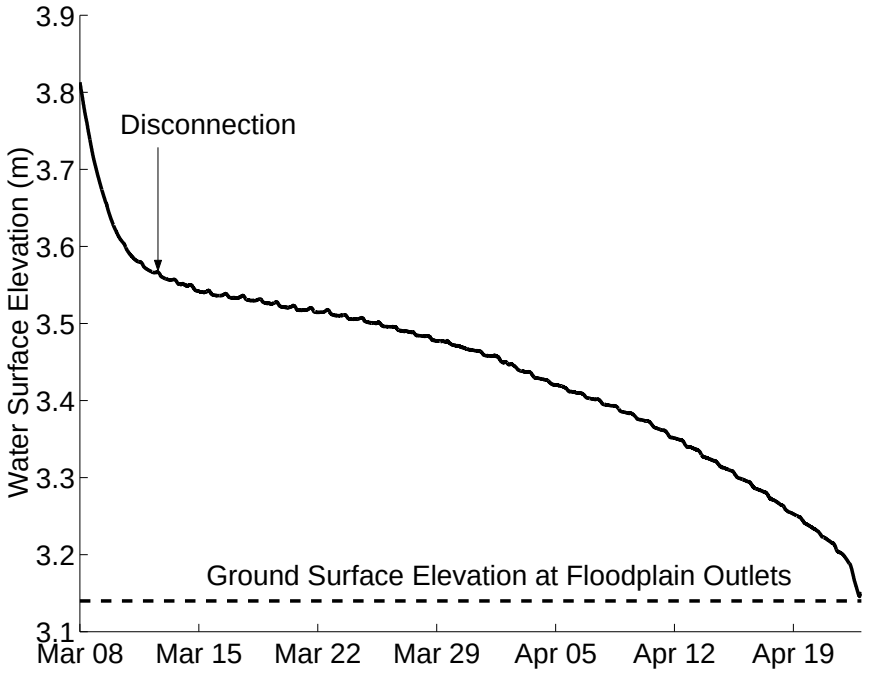

Figure 4.17: Outlet water surface elevations for a draining event during the spring of WY2004

centration at a time t_{max} and remaining constant for all subsequent times (limited growth), and another reaching a maximum at t_{max}, remaining constant for a period of time, and rapidly decreasing to zero thereafter (crashing growth). Time scales for algal growth were taken from data presented by Hein *et al.* (2004) for sidearm channels of the Danube River and Ahearn *et al.* (2006) from this study site. Both studies found chlorophyll-a concentrations to reach a maximum of three times the initial concentration at approximately 10 days. Population size stayed constant or decreased thereafter, probably due to grazing pressures (Ahearn *et al.*, 2006). Sensitivity to the chosen value of t_{max} was estimated by AIF's for draining simulations using values of 6, 10, and 14 days for t_{max}.

The use of the algal increase factor to estimate phytoplankton exports from the floodplain requires several assumptions. The AIF assumes that that an initial algae population is uniformly distributed throughout the floodplain at time t_0. Here t_0 was assumed to be the moment floodplain disconnection from TN and TS occurs. Subsequent algal movements are modeled with the use of the conservative tracer algorithm, and each incremental mass of tracer is multiplied by a factor $f(t)$ as it exits the floodplain. The factor increases as

the algal population would, and each of these products of tracer mass and $f(t)$ factor are summed at simulation end. If the entire mass of tracer exits the floodplain one instant after t_0, the AIF will approximately equal 1.0 and no increase in the algae population took place on the floodplain. If, on the other hand, all the water present at t_0 remained stationary for ten days and then instantaneously exited, the AIF would equal 3.0 based on the assumed time and growth scales given earlier.

The mean flushing times and AIF's calculated with the crashing growth curve for the draining phase simulations are presented in Table 4.1. Increase factors obtained with the limited growth curve were similar. The mean flushing time, which is recalled as a spatially integrated description of the mean residence times, was 3.41 days for the 42 day draining simulation. The simulation with drainage occurring over only 12 days had a mean flushing time of 2.68 days. Thus, decreasing the draining time by a factor of four only decreased the mean flushing time by about 20%. The simulations showed this result occurred because the majority of the water mass that is able to leave the floodplain during the draining phase leaves within the first 7 cm decrease in outlet water surface elevation. This results from the large aerial extent of the floodplain under shallow water and occurred in only a couple of days less time for the shortest draining simulation than for the longest. Since the assumed algal growth time scales were significantly larger than differences in mean flushing times, little effect was seen in the AIF's. If a limiting algal growth curve is assumed, observed chlorophyll concentrations for water exiting the floodplain during the 42 day draining simulation will remain high for a month longer than concentrations in the 12 day simulation. However, these results indicate that total mass of algae exported will be approximately the same.

Table 4.1 shows shorter duration simulations to have higher algal increase factors than the longer simulations. Although the AIF strictly depends on the residence time distribution function, it is unlikely that faster draining simulations have a significant amount of tracer mass leaving the floodplain at a later time than longer draining simulations. However, the differences are relatively small and are probably a result of faster draining simulations stranding of a greater amount of tracer mass coupled with the use of equation (4.2).

These results are important from a floodplain management point of view. It was shown that increasing the floodplain draining rate to one quarter of its present value will not effect

Table 4.1: Algal increase factors for crashing population curves for draining phase simulations

Time to drain	Mean Flushing Time	Algal Increase Factor		
(days)	(hours)	$t_{max} = 6$ days	$t_{max} = 10$ days	$t_{max} = 14$ days
42	81.9	1.86	1.68	1.49
36	80.2	1.87	1.68	1.49
30	78.0	1.88	1.68	1.49
24	75.2	1.88	1.68	1.49
18	71.1	1.92	1.69	1.50
12	64.3	1.98	1.70	1.51

phytoplankton exports to the Delta. This result could be useful if floodplains similar to that in the Cosumnes River Preserve are managed with multi-use principles such as with crop production in the nearby Yolo Bypass (Sommer *et al.*, 2001a). If the floodplain were drained faster, it could be farmed starting earlier in the spring, and might generate higher crop production while retaining the same ecological benefits to the Delta. If managers are every able to control the timing of flood events on the Cosumnes River, the floodplain area could be innundated, allowed to drain for two weeks, and then reflooded in order to maximize the biomass flux downstream.

5 Summary, Conclusions, and Future Directions

The goals of this work were to: (1) create an accurate and efficient hydrodynamic and scalar transport model for application to riverine floodplain environments, (2) test the model framework with the use of three test cases simulating the relevant characteristics of floodplain flows, and (3) apply the model to a restored floodplain on the lower Cosumnes River to examine the distribution of residence times and implications for phytoplankton exports.

Floodplain flows are complex, driven by topography and bottom roughness, and characterized by very shallow water depths and flow through vegetation. In deriving the two-dimensional, depth-averaged governing equations for mass and momentum conservation and scalar transport, care was taken to create a robust model for applications other than only the Cosumnes River floodplain. To this effect, terms that are not typically included in previous floodplain models, such as precipitation, evapotranspiration, seepage, wind stress, and Coriolis force terms were retained in the equations. Drag due to flow through vegetation was incorporated into the parameterization of bottom friction because of the high data acquisition and storage needs for treating it separately. Bottom friction factors were calculated during model runtime using only prescribed vegetation heights and flow variables.

A one dimensional discretization of the governing equations was made using a two-level, semi-implicit scheme and an iterative method for bottom friction. An advection algorithm was chosen that conserved energy through flow contractions and momentum through flow expansions. The use of slope limiters imparted an overall accuracy of 'almost second order.' The model framework was subjected to three sets of test cases designed to simulate: (1) flow over complex and rapidly varying topography, (2) tidal wetting and drying, and (3) the advection of a hydrograph. The model was found to perform well in all cases, although specialized procedures to add and remove cells from the computational space were needed for simulation of wetting and drying. Slope limiters were shown to improve the root mean square error greatly from first order upwinding at similar discretization levels, mainly through the mitigation of amplitude error. Phase and mass preservation error properties of the method were very good. The two-dimensional discretization proceeded similarly, resulting in a large sparse matrix that was solved efficiently using the preconditioned conjugate gradient

method. Corner transport upwind, a high resolution method that is positivity preserving and can be extended to almost second order using slope limiters, was used to approximate the advection of a scalar variable.

The fully developed model was applied to a leveed floodplain on the lower Cosumnes River, CA. Tracer release experiments were used to approximate the spatial and temporal distributions of residence and flushing times. Mean residence times were found to vary considerably throughout the floodplain. Residence times were on the order of hours along the main North-South flowpaths and days at the periphery. Mean flushing times were short during the rising limb and peak portions of a floodpulse hydrograph, but increased to several days once inflows ceased.

Simulations were run to assess increases in phytoplankton export assuming the speed of the floodplain draining phase could be varied. Decreasing the draining time from its current 42 day period to a 12 day period was found to have little effect on total phytoplankton exports from the floodplain. This implies exports may be optimized by allowing the draining phase to proceed over 12 days and then reflood the basin. It is noted here that these management scenarios focused only on primary production exports to the Delta and ignored all other ecosystem and water quality consequences. By way of example, the splittail (*Pogonichthys macrolepidotus*), a native fish species of concern in the Delta, is known to use the floodplain for spawning habitat. Alien fish found on the floodplain favor low flows and high temperatures, while natives prefer colder and deeper water. This observation lead Crain *et al.* (2004) to recommend that 'restoration should emphasize early flooding followed by rapid draining to prevent alien fishes from becoming abundant.' Although the timescale of rapid draining was not defined, draining the floodplain over two days in contrast to 12 days would certainly have consequences for primary production. This highlights some of the ecosystem tradeoffs intrinsic to floodplain management. It is hoped that the model developed in this work can be used to further explore some of these tradeoffs and help guide restoration efforts.

Finally, it should be mentioned that there are a few drawbacks to the method used to predict phytoplankton exports from the floodplain. These include the lack of an explicit treatment of nutrient and light limitation, temperature, self-shading, and grazing. Never-

theless, the algal modeling method used in this study was simple to implement, required few model inputs, and illustrated that results meaningful to floodplain management can be obtained with a residence time analysis. Future work will account for the previously listed limitations and explicitly model algal concentrations with the use of a full water quality model including such constituents as nutrients, suspended sediments, and dissolved oxygen.

References

Ahearn, D. S., Viers, J. H., Mount, J. F., & Dahlgren, R. A. 2006. Priming the productivity pump: Floodpulse driven trends in suspended algal biomass distribution across a restored floodplain. *Freshwater Biology*, **51**, 1417–1433.

Alexander, J., & Marriott, S. B. 1999. Introduction. *Pages 1–13 of:* Marriott, S. B., & Alexander, J. (eds), *Floodplains: Interdisciplinary Approaches*. Bath, UK: The Geological Society of London.

Allen, P. A. 1997. *Earth Surface Processes*. Oxford, UK: Blackwell Sciences Ltd.

Backhaus, J. O. 1983. A semi-implicit scheme for the shallow water equations for application to shelf sea modeling. *Continental Shelf Research*, **2**(4), 243–254.

Balzano, A. 1998. Evaluation of methods for numerical simulation of wetting and drying in shallow water flow models. *Coastal Engineering*, **34**, 83–107.

Baranyi, C., Hein, T., Holarek, C., Keckeis, S., & Schiemer, F. 2002. Zooplankton biomass and community structure in a Danube River floodplain system: effects of hydrology. *Freshwater Biology*, **47**, 473–482.

Bates, P. D., & Hervouet, J. 1999. A new method for moving-boundary hydrodynamic problems in shallow water. *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, **455**, 3107–3128.

Bates, P. D., Anderson, M. G., Baird, L., Walling, D. E., & Simm, D. 1992. Modelling floodplain flows using a two-dimensional finite element model. *Earth Surface Processes and Landforms*, **17**, 575–588.

Bedient, P. B., & Huber, W. C. 2002. *Hydrology and Floodplain Analysis*. Third edn. Upper Saddle River, NJ: Prentice Hall, Inc.

Beffa, C., & Connell, R. J. 2001. Two-dimensional flood plain flow I: Model description. *Journal of Hydrologic Engineering*, **6**(5), 397–405.

Bennett, W. A., & Moyle, P. B. 1996. Where have all the fishes gone? Interactive factors producing fish declines in the Sacramento-San Joaquin estuary. *Pages 519–542 of:* Hollibaugh, J. T. (ed), *San Francisco Bay: The Ecosystem*. San Francisco, CA: Pacific Division of the American Association for the Advancement of Science.

Bernard, R. S. 1993. *STREMR: Numerical Model for Depth-Averaged Incompressible Flow*. Tech. rept. REMR-HY-11. US Army Corps of Engineers Hydraulics Laboratory, Washington, D.C.

Bombardelli, F. A., & García, M. H. 2003. Hydraulic design of large-diameter pipes. *Journal of Hydraulic Engineering*, **129**(1), 839–846.

Booth, E. G., Mount, J. F., & Viers, J. H. 2006. Hydrologic variability of the Cosumnes River Floodplain. *San Francisco Estuary and Watershed Science*, **4**(2).

Bradford, S. F., & Sanders, B. F. 2002. Finite-volume model for shallow-water flooding of arbitrary topography. *Journal of Hydraulic Engineering*, **128**(3), 289–298.

Braschi, G., Dadone, F., & Gallati, M. 1994. Plain flooding: Near field and far field simulations. *Pages 45–59 of:* Molinaro, P., & Natale, L. (eds), *Modeling of Flood Propagation over Initially Dry Areas*. New York, NY: American Society of Civil Engineers.

Bricelj, V. M., & Lonsdale, D. J. 1997. Causes and ecological consequences of brown tides in U.S. mid-Atlantic coastal waters. *Limnology and Oceanography*, **42**, 1023–1038.

Brufau, P., Vázquez-Cendón, M. E., & García-Navarro, P. 2002. A numerical model for the flooding and drying of irregular domains. *International Journal for Numerical Methods in Fluids*, **39**, 247–275.

Buscaglia, G. C., Bombardelli, F. A., & García, M. H. 2002. Numerical modeling of large-scale bubble plumes accounting for mass transfer effects. *International Journal of Multiphase Flow*, **28**, 1763–1785.

Casulli, V., & Cheng, R. T. 1992. Semi-implicit finite difference methods for three-dimensional shallow water flow. *International Journal for Numerical Methods in Fluids*, **15**, 629–648.

Casulli, V., & Stelling, G. S. 1998. Numerical simulation of 3D quasi-hydrostatic, free-surface flows. *Journal of Hydraulic Engineering*, **124**(7), 678–686.

Chanson, H. 2004. *The Hydraulics of Open Channel Flow: An Introduction*. Second edn. Boston, MA: Elsevier, Inc.

Chapra, S. C. 1997. *Surface Water-Quality Modeling*. Boston, MA: McGraw-Hill Companies, Inc.

Christensen, D. L., Carpenter, S. R., Cottingham, K. L., Knight, S. E., LeBouton, J. P., Schindler, D. E., Voichick, N., Cole, J. J., & Pace, M. L. 1996. Pelagic responses to changes in dissolved organic carbon following division of a seepage lake. *Limnology and Oceanography*, **41**, 553–559.

Cobby, D. M., Mason, D. C., Horritt, M. S., & Bates, P. D. 2003. Two-dimensional hydraulic flood modelling using a finite-element mesh decomposed according to vegetation and topographic features derived from airborne scanning laser altimetry. *Hydrological Processes*, **17**, 1979–2000.

Connell, R. J., Painter, D. J., & Beffa, C. 2001. Two-dimensional flood plain flow II: Model validation. *Journal of Hydrologic Engineering*, **6**(5), 406–415.

Crain, P. K., Whitener, K., & Moyle, P. B. 2004. Use of a restored Central California floodplain by larvae of native and alien fishes. *American Fisheries Society Symposium*, **39**, 125–140.

Cunge, J. A., Holly, F. M., & Verwey, A. 1980. *Practical Aspects of Computational River Hydraulics*. Monographs and Surveys in Water Resources Engineering, no. 3. Boston, MA: Pitman Publishing.

D'Alpos, L., & Defina, A. 1993. Venice lagoon hydrodynamics simulation by coupling 2D and 1D finite element models. *In:* Morgan, K., Zienkiewicz, O. C., Onate, E., Peraire, J., & Periaux, J. (eds), *Proceedings of the 8th Conference on Finite Elements in Fluids. New Trends and Applications, Barcelona, Spain*. Swansea, UK: Pineridge Press.

Darby, S. E. 1999. Effect of riparian vegetation on flow resistance and flood potential. *Journal of Hydraulic Engineering*, **125**(5), 443–454.

Defina, A. 2000. Two-dimensional shallow flow equations for partially dry areas. *Water Resources Research*, **36**(11), 3251–3264.

denHeyer, C., & Kalff, J. 1998. Organic matter mineralization rates in sediments: a within- and among-lake study. *Limnology and Oceanography*, **43**, 695–705.

DiGiammarco, P., & Todini, E. 1994. A control volume finite element method for the solution of 2D overland flow problems. *Pages 82–101 of:* Molinaro, P., & Natale, L. (eds), *Modeling of Flood Propagation over Initially Dry Areas*. New York, NY: American Society of Civil Engineers.

DiGiammarco, P., Todini, E., Consuegra, D., Joerin, F., & Vitalini, F. 1994. Combining a 2-D floodplain model with GIS for flood delineation and damage assessment. *Pages 171–185 of:* Molinaro, P., & Natale, L. (eds), *Modeling of Flood Propagation over Initially Dry Areas*. New York, NY: American Society of Civil Engineers.

Durran, D. R. 1999. *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*. Texts in Applied Mathematics, no. 32. New York, NY: Springer-Verlag New York, Inc.

Erduran, K. S., & Kutija, V. 2003. Quasi-three-dimensional numerical model for flow through flexible, rigid, submerged and non-submerged vegetation. *Journal of Hydroinformatics*, **5**(3), 189–202.

Falconer, R. A., & Owens, P. H. 1987. Numerical simulation of flooding and drying in a depth-averaged tidal flow model. *Proceedings of the Institution of Civil Engineers, Part 2*, **83**, 161–180.

Fathi-Moghadam, M., & Kouwen, N. 1997. Nonrigid, nonsubmerged, vegetative roughness on floodplains. *Journal of Hydraulic Engineering*, **123**(1), 51–57.

Fischer-Antze, T., Stoesser, T., Bates, P., & Olsen, N. R. B. 2001. 3D numerical modelling of open-channel flow with submerged vegetation. *Journal of Hydraulic Engineering*, **39**, 303–310.

Florsheim, J. L., & Mount, J. F. 2002. Restoration of floodplain topography by sand-splay complex formation in response to intentional levee breaches, Lower Cosumnes River, California. *Geomorphology*, **44**, 67–94.

García, M. H. 1996. *Hidrodinamica Ambiental*. Colección Ciencia y Técnica, Centro de Publicaciones Universidad Nacional del Litoral. Santa Fe, Argentina: Imprenta Lux S.R.L.

Gill, A. E. 1982. *Atmosphere-Ocean Dynamics*. San Diego, CA: Academic Press.

Gross, E. S., Bonaventura, L., & Rosatti, G. 2002. Consistency with continuity in conservative advection schemes for free-surface models. *International Journal for Numerical Methods in Fluids*, **38**, 307–327.

Gupta, R. S. 2001. *Hydrology and Hydraulic Systems*. Second edn. Prospect Heights, IL: Waveland Press, Inc.

Hein, T., Baranyi, C., Reckendorfer, W., & Schiemer, F. 2004. The impact of surface water exchange on the nutrient and particle dynamics in side-arms along the River Danube, Austria. *Science for the Total Environment*, **328**, 207–218.

Henderson, F. M. 1996. *Open Channel Flow.* New York, NY: MacMillan Company.

Hervouet, J. M., & Janin, J. M. 1994. Finite element algorithms for modelling flood propagation. *Pages 102–113 of:* Molinaro, P., & Natale, L. (eds), *Modeling of Flood Propagation over Initially Dry Areas.* New York, NY: American Society of Civil Engineers.

Hey, R. D. 1979. Flow resistance in gravel-bed rivers. *Journal of the Hydraulic Division, ASCE*, **105**(4), 365–379.

Hilton, J., Rigg, E., Davison, W., Hamilton-Taylor, J., Kelly, M., Livens, F. R., & Singelton, D. L. 1995. Modeling and interpreting element ratios in water and sediments: a sensitivity analysis of post-Chernobyl Ru:Cs ratios. *Limnology and Oceanography*, **40**, 1302–1309.

Horritt, M. S., & Bates, P. D. 2001. Predicting floodplain innundation: raster-based modeling versus the finite element approach. *Hydrological Processes*, **15**, 825–842.

Hubbard, R. K., & Lowrance, R. R. 1996. Solute transport and filtering through a riparian forest. *Transactions of the ASCE*, **39**(2), 477–488.

Jassby, A. D., & Cloern, J. E. 2000. Organic matter sources and rehabilitation of the Sacramento-San Joaquin Delta (California, USA). *Aquatic Conservation: Marine and Freshwater Ecosystems*, **10**, 323–352.

Jassby, A. D., Powell, T. M., & Goldman, C. R. 1990. Interannual fluctuations in primary production: direct physical effects and the trophic cascade at Castle Lake, California. *Limnology and Oceanography*, **35**, 1021–1038.

Jia, Y., Wang, S. S. Y., & Xu, Y. 2002. Validation and application of a 2D model to channels with complex geometry. *International Journal of Computational Engineering Science*, **3**(1), 57–71.

Junk, W. J., & Welcomme, R. L. 1990. Floodplains. *Pages 491–524 of:* Pattern, B. C. (ed), *Wetlands and Shallow Continental Water Bodies.* The Hague, The Netherlands: SPB Academic Publishers.

Kimmerer, W. J., & Orsi, J. J. 1996. Changes in the zooplankton of the San Francisco Bay Estuary since the introduction of the clam *potamocorbula amurensis. Pages 403–424 of:* Hollibaugh, J. T. (ed), *San Francisco Bay: The Ecosystem.* San Francisco, CA: Pacific Division of the American Association for the Advancement of Science.

Kincaid, D. R., Oppe, T. C., & Joubert, W. D. 1989. An introduction to the nspcg software package. *International Journal for Numerical Methods in Engineering*, **27**(3), 589–608.

King, I. P. 1993. *A Finite Element Model for Three-Dimesional Density Stratified Flow. Sidney Deepwater Outfalls Environmental Monitoring Program Post-commission Phase.* Tech. rept. Australian Water and Coastal Studies, Sydney, Australia.

Kouwen, N. 1988. Field estimation of the biomechanical properties of grass. *Journal of Hydraulic Engineering*, **26**(5), 559–568.

Kouwen, N., & Fathi-Moghadam, M. 2000. Friction factors for coniferous trees along rivers. *Journal of Hydraulic Engineering*, **126**(10), 732–740.

Kouwen, N., & Li, R. M. 1980. Biomechanics of vegetative channel linings. *Journal of the Hydraulic Division, ASCE*, **106**(6), 713–728.

Kundu, P. K., & Cohen, I. M. 2004. *Fluid Mechanics*. Third edn. Boston, MA: Elsevier, Inc.

Leveque, R. J. 1996. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis*, **33**(2), 627–665.

LeVeque, R. J. 2002. *Finite Volume Methods for Hyperbolic Problems*. New York, NY: Cambridge University Press.

López, F., & García, M. H. 1998. Open-channel flow through simulated vegetation: Suspended sediment transport modeling. *Water Resources Research*, **34**(9), 2341–2352.

Malard, F., Mangin, A., Uehlinger, U., & Ward, J. V. 2001. Thermal heterogeneity in the hyporheic zone of a glacial floodplain. *Canadian Journal of Fisheries and Aquatic Sciences*, **58**, 1319–1335.

Marks, K., & Bates, P. 2000. Integration of high-resolution topographic data with floodplain flow models. *Hydrological Processes*, **14**, 2109–2122.

Mason, D. C., Cobby, D. M., Horritt, M. S., & Bates, P. D. 2003. Floodplain friction parameterization in two-dimensional river flood models using vegetation heights derived from airborne scanning laster altimetry. *Hydrological Processes*, **17**, 1711–1732.

Menendez, A. N., & Laciana, C. E. 2006. Pollutant dispersion in water currents under wind action. *Journal of Hydraulic Research*, **44**(4), 470–479.

Mertes, L. A. K. 1997. Documentation and significance of the perirheic zone on inundated floodplains. *Water Resources Research*, **33**(7), 1749–1762.

Mertes, L. A. K., Smith, M. O., & Adams, J. B. 1993. Estimating suspended sediment concentrations in surface waters of the Amazon River wetlands from Landsat images. *Remote Sensing of Environment*, **43**, 281–301.

Miller, Jr, G. T. 2000. *Living in the Environment: Principles, Connections, and Solutions*. Eleventh edn. Pacific Grove, CA: Brooks/Cole Publishing Company.

Molinaro, P., DiFillipo, A., & Ferrari, F. 1994. Modelling of flood wave propagation over flat dry areas of complex topography in presence of different infrastructures. *Pages 209–228 of:* Molinaro, P., & Natale, L. (eds), *Modeling of Flood Propagation over Initially Dry Areas*. New York, NY: American Society of Civil Engineers.

Monsen, N. E., Cloern, J. E., Lucas, L. V., & Monismith, S. G. 2002. A comment on the use of flushing time, residence time, and age as transport time scales. *Limnology and Oceanography*, **47**(5), 1545–1553.

Mount, J. F. 1995. *California Rivers and Streams: The Conflict Between Fluvial Processes and Land Use*. Berkeley, CA: University of California Press.

Naiman, R. J., & Decamps, H. 1997. The ecology of interfaces: Riparian zones. *Annual Review of Ecology and Systematics*, **28**, 621–658.

Nepf, H. M. 1999. Drag, turbulence, and diffusion in flow through emergent vegetation. *Water Resources Research*, **35**(2), 479–489.

Nicholas, A. P., & McLelland, S. J. 2004. Computational fluid dynamics modelling of three-dimensional processes on a natural river floodplain. *Journal of Hydraulic Research*, **42**(2), 131–143.

Nicholas, A. P., & Mitchell, C. A. 2003. Numerical simulation of overbank processes in topographically complex floodplain environments. *Hydrological Processes*, **17**, 727–746.

Nielsen, C., & Apelt, C. 2003. Parameters affecting the performance of wetting and drying in a two-dimensional finite element long wave hydrodynamic model. *Journal of Hydraulic Engineering*, **129**(8), 628–636.

Norton, W. R., King, I. P., & Orlob, G. T. 1973. *A Finite Element Model for Lower Granite Reservoir*. Tech. rept. US Army Corps of Engineers, Walla Wall District, Walla Walla, WA.

Ohman, M. D., & Wood, S. N. 1996. Mortality estimation for planktonic copepods: *Pseudocalanus newmani* in a temperate fjord. *Limnology and Oceanography*, **41**, 126–135.

Olsen, N. R. B., & Stokseth, S. 1995. Three-dimensional numerical modeling of water flow in a river with large bed roughness. *Journal de Recherches Hydrauliques*, **33**(4), 571–581.

Perumal, M., Shrestha, K. B., & Chaube, U. C. 2004. Reproduction of hysteresis in rating curves. *Journal of Hydraulic Engineering*, **130**(9), 870–878.

Poff, N. L., Allen, J. D., Bain, M. B., Karr, J. R., Prestegaard, K. L., Richter, B. D., Sparks, R. E., & Stromberg, J. C. 1997. The natural flow regime: A paradigm for river conservation and restoration. *BioScience*, **11**, 769–784.

Pope, S. B. 2000. *Turbulent Flows*. New York, NY: Cambridge University Press.

Rodi, W. 1980. *Turbulence Models and their Application in Hydraulics: A State of the Art Review*. Delft, The Netherlands: International Association for Hydraulic Research.

Rueda, F. 2001. *A Three-Dimensional Hydrodynamic and Transport Model for Lake Environments*. Ph.D. thesis, University of California, Davis, Davis, CA.

Rueda, F. J., & Cowen, E. A. 2005. Residence time of a freshwater embayment connected to a large lake. *Limnology and Oceanography*, **50**(5), 1638–1653.

Rueda, F. J., & Schladow, S. G. 2002. Quantitative comparison of models for barotropic response of homogeneous basins. *Journal of Hydraulic Engineering*, **128**(2), 201–213.

Rutherford, J. C. 1994. *River Mixing*. New York, NY: John Wiley and Sons, Inc.

Sandwell, D. T. 1987. Biharmonic spline interpolation of GOES-3 and SEASAT altimeter data. *Geophysical Research Letters*, **14**(2), 139–142.

Sielecki, A. 1968. An energy-conserving difference scheme for the storm surge equations. *Monthly Weather Review*, **96**(3).

Smith, P. E. 1997. *A Three-Dimensional, Finite-Difference Model for Estuarine Circulation*. Ph.D. thesis, University of California, Davis, Davis, CA.

Sommer, T., Harrell, B., Nobriga, M., Brown, R., Moyle, P., Kimmerer, W., & Schemel, L. 2001a. California's Yolo Bypass: Evidence that flood control can be compatible with fisheries, wetlands, wildlife, and agriculture. *Fisheries*, **26**(8), 6–16.

Sommer, T. R., Nobriga, M. L., Harrell, W. C., Batham, W., & Kimmerer, W. J. 2001b. Floodplain rearing juvenile chinook salmon: evidence of enhanced growth and survival. *Canadian Journal of Fisheries and Aquatic Sciences*, **58**, 325–333.

Stelling, G. S., & 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**, 1329–1354.

Teeter, A. M., Johnson, B. H., Berger, C., Stelling, G., Scheffner, N. W., Garcia, M. H., & Parchure, T. M. 2001. Hydrodynamic and sediment transport modeling with emphasis on shallow-water, vegetated areas (lakes, reservoirs, estuaries and lagoons). *Hydrobiologia*, **444**, 1–23.

Tennessee Valley Authority. 1972. *Heat and Mass Transfer Between a Water Surface and the Atmosphere*. Tech. rept. 0-6803. Tennessee Valley Authority Division of Water Control Planning Engineering Laboratory, Norris, TN.

Tockner, K., & Stanford, J. A. 2002. Riverine flood plains: Present state and future trends. *Environmental Conservation*, **29**(3), 308–330.

Tockner, K., Pennetzdorfer, D., Reiner, N., Schiemer, F., & Ward, J. V. 1999. Hydrologic connectivity, and the exchange of organic matter and nutrients in a dynamic river-floodplain system (Danube, Austria). *Freshwater Biology*, **41**, 521–535.

Tockner, K., Malard, F., & Ward, J. V. 2000. An extension of the flood pulse concept. *Hydrological Processes*, **14**, 2861–2883.

Valiela, I., McClelland, J., Hauxwell, J., Behr, P. J., Hersh, D., & Foreman, K. 1997. Macroalgal blooms in shallow estuaries: controls and ecophysiological and ecosystem consequences. *Limnology and Oceanography*, **42**, 1105–1118.

Wang, X. H., Byun, D. S., Wang, X. L., & Cho, Y. K. 2005. Modelling tidal currents in a sediment statified idealized estuary. *Continental Shelf Research*, **25**, 655–665.

Wu, W., Shield, Jr, F. D., Bennett, S. J., & Wang, S. S. Y. 2005. A depth-averaged two-dimensional model for flow, sediment transport, and bed topography in curved channels with riparian vegetation. *Water Resources Research*, **41**, W03015.

A List of Notations

Character	Description	First Usage
a, b	coefficients for calculation of short vegetation shear stress	33
a_s	shape correction factor for bottom roughness	32
A	characteristic area for object exerting drag force	16
A_c	channel area at bankful depth	32
A_e	amplitude error	65
$[A]$	algal concentration	101
ADV	explicit 2D discretization for advection	72
c	concentration of a generalized scalar variable	18
c_f	frictional coefficient	31
c_p	specific heat of water	18
c_{sed}	suspended sediment concentration	13
c_{vk}	k-ε closure scheme coefficient	39
$c_{\varepsilon 1}$, $c_{\varepsilon 2}$, $c_{\varepsilon 3}$	k-ε closure scheme coefficients	38
c_μ	k-ε closure scheme coefficient	37
C	depth-averaged scalar concentration	35
$C(r)$	slope limiter function of r	50
C_w	wind drag coefficient	30
C_z	Chézy coefficient	31
C_D	drag coefficient	16
C_{Ev}	evaporation coefficient	25
C_{Tr}	transpiration coefficient	26
COR	explicit 2D discretization for Coriolis terms	72
d	location of floodplain bottom	23
d_c	cylinder diameter	16
d_{xx}	$\mathrm{xx}^{\mathrm{th}}$ percentile grain size diameter	32
D_{mech}	mechanical diffusivity	17
D_{molec}	molecular diffusivity	17
$\mathscr{D}$	generalized drag force	16
e_a	vapor pressure of air	25
e_{sat}	saturation vapor pressure of air	25
E	systematic energy head	47
E_v	evaporation rate	25
E_y	vegetation stem modulus of elasticity	32

ET	evapotranspiration rate	24
f	generalized function	23
f_c	Coriolis frequency	14
f_D	Darcy-Weisbach friction factor	31
F	advective scalar flux in the x direction	76
F_B	bed force term	46
F_D	drag force due to vegetation	39
Fr	Froude number	46
$FRICT$	2D discretization for friction	72
g	gravitational acceleration in the vertical direction	14
$\mathbf{g}$	gravitational acceleration vector	13
G	advective scalar flux in the y direction	76
GW	ground water flow rate	24
h	total water depth	23
h_{veg}	vegetation height	26
h_{veg}^d	deflected vegetation height	32
h_0	uniform flow depth	65
${}^*h$	total water depth at a computational cell edge	42
H	vertical length scale	21
$HDIFF$	explicit 2D discretization for turbulent diffusion terms	72
$HYDRO$	explicit 2D discretization for hydrologic terms	72
$\mathbf{i}$, $\mathbf{j}$, $\mathbf{k}$	Cartesian coordinate unit vectors	11
I	downwelling solar irradiance	18
I_z	vegetation stem second moment of inertia	32
J_x, J_y	depth-averaged turbulent scalar fluxes	35
k	turbulent kinetic energy	36
k_d	deposition rate	35
k_r	net reaction rate	19
k_{rs}	resuspension rate	35
k_s	settling rate	35
k_v	volatilization rate	35
k_x, k_y	shear dispersion coefficients	35
k_M	Manning equation coefficient	31
L	characteristic horizontal length scale	12
L_c	length of channel	66
$m(t)$	mass of tracer in the computational domain at time t	96
m_r	mass of tracer remaining in domain at simulation end	96

m_0 initial mass of tracer released 96
M vegetation stem density 32
M_e mass preservation error 65
n_c number of cylinders per unit area 16
n_M Manning roughness coefficient 31
P instantaneous fluid pressure 13
P_{atm} atmospheric pressure 25
P_e phase error 65
P_r precipitation rate 24
P_w channel wetted perimeter at bankful depth 32
P_0 hydrostatic reference pressure 16
$\tilde{P}$ departure from hydrostatic condition in P 16
$\mathscr{P}_h$ production of k by shear in mean fluid motion 38
$\mathscr{P}_{kV}$ production of k by bed shear 38
$\mathscr{P}_v$ production of k by flow over submerged vegetation 39
$\mathscr{P}_{\varepsilon V}$ increase in ε by bed shear 38
q discharge per unit width 48
q_s heat flux at the free surface 35
Q discharge 65
Q_0 steady flow discharge 65
r measure of local smoothness for a dependent variable 50
rr main diagonal coefficients in 2D matrix equation 72
R distance function 77
R_h channel hydraulic radius 32
Ro Rossby number 14
RH relative humidity 25
RHS right hand side vector in matrix equation for ζ 45
RMS_e root mean square error 65
sx, sy off diagonal coefficients in 2D matrix equation 72
S_c sum of all scalar sources and sinks 17
S_0 local bottom slope 33
$SDIFF$ explicit 2D discretization for turbulent scalar fluxes 76
ΔS mean spacing of cylinders 16
t time 11
t_{max} time to reach maximum algal concentrations 102
t_0 time averaging interval 19
T temperature 12

$T(x)$	time of arrival of hydrograph center of mass	65
T_{end}	end of simulation time	66
u, v, w	instantaneous fluid velocity in the x, y, and z directions	11
u_*	shear velocity	30
u_{*crit}	threshold shear velocity	33
$\mathbf{u}$	instantaneous fluid velocity vector	11
U, V	depth-averaged lateral fluid velocities	24
U_c	characteristic horizontal velocity scale	12
${}^*U$, ${}^*V$	depth-averaged fluid velocities at the cell centers	48
W_z	wind speed measured at a height z above the free surface	25
$WIND$	explicit 2D discretization for surface shear stress	72
x, y, z	Cartesian coordinates	11
z_0	average height of roughness elements	30
z'	vertical distance measured from roughness elements	30
α_t	thermal expansion coefficient	12
β	momentum correction coefficient	28
β_E	relative head loss	47
γ	explicit terms in the discretization of bottom friction	44
Γ	depth-averaged turbulent diffusivity	37
$\delta(\ )$	small perturbation in $(\ )$	12
δ_c	relative strength of a flow contraction	46
ϵ	coefficient for dynamic choice of the advection discretization	49
ϵ_h	small water depth constant	60
ϵ_{shal}	wet/dry water depth constant	60
ε	turbulent kinetic energy dissipation rate	19
ζ	water surface elevation	23
ζ^L, ζ^H	low and high order estimations for the water surface elevation	49
η	relative water surface difference	46
ϑ	scaling function for mechanical diffusion	18
Θ	wind direction angle	30
κ	von Karman constant	30
λ	vegetative momentum absorbing area per unit volume	16
μ	laminar dynamic fluid viscosity	13
ν	laminar kinematic fluid viscosity	17
ν_t	turbulent kinematic fluid viscosity	36
ξ	Kolmogorov microscale	19
ξ_v	coefficient for large vegetation deformation	34

Ξ	coefficient in 1D coefficient matrix for ζ	45
ρ	instantaneous fluid density	11
ρ_{air}	air density	25
ρ_s	sediment density	13
ρ_0	hydrostatic reference density	12
$\tilde{\rho}$	departure from hydrostatic condition in ρ	16
σ_k, σ_ε	k-ε closure scheme coefficients	38
σ_t	turbulent Schmidt number	37
τ_r	mean residence time	95
τ_{xb}, τ_{yb}	bottom shear stresses	27
τ_{xs}, τ_{ys}	free surface shear stresses	27
τ_{xx}, τ_{xy}, τ_{yy}	depth averaged Reynolds stresses	28
ϕ	latitudinal angle	14
φ	dimensionless depth variable	28
Φ	generalized variable	19
χ	coefficient used to stabilize frictional iterations	44
ψ	water surface at cell edge coefficient	46
$\Psi(t)$	residence time distribution function	101
$\mathbf{\Omega}$	angular velocity vector of the earth	14
$\times$	cross product operator	14
$\boldsymbol{\nabla} \cdot (\ )$	divergence operator	11
$\boldsymbol{\nabla}(\ )$	gradient operator	13
$\boldsymbol{\nabla}^2(\ )$	Laplacian operator	13
$\frac{D(\ )}{Dt}$	material derivative operator	12
$\dot{}$	time averaged quantity	19
$\bar{}$	ensemble averaged quantity	20
$'$	turbulent fluctuation of a quantity	20
$\sim$	value of a dependent variable from previous iteration	44
$\widehat{}$	collection of explicit terms	45

B Expanded Finite Difference Discretizations

The discretized terms involving hydrology, advection, surface and bottom shear, turbulent momentum diffusion, and the Coriolis force are represented in compact form in equations (3.24)-(3.30). The complete expanded form of these terms is presented here.

Hydrology terms in (3.24):

$$\begin{aligned}(HYDRO)_{i,j}^{n} &= (Pr - ET \pm GW)_{i,j}^{n} \\ &= Pr^{n} - ET^{n} \pm GW_{i,j}^{n}\end{aligned}$$

The precipitation rate at time t^n is determined from meteorological data input, the groundwater rate of loss or gain is prescribed based on existing knowledge of the site hydrogeology, and the evapotranspiration rate can either be determined from direct meteorological data input or calculated using air temperature, relative humidity, and wind speed with equations (2.24)-(2.27). The subscript on the groundwater term is included to indicate the possibility of spatially varying subsurface contributions. For example, cells that are wet at t^n may have a constant seepage rate may be applied. Cells that are dry cells at the start of a precipitation event may be subject to infiltration, in which case a maximum infiltration rate is prescribed.

Advection terms in (3.25) and (3.26):

Momentum conserving form:

$$\begin{aligned}(ADV_x)_{i+1/2,j}^{n} &= \left(U\frac{\partial U}{\partial x}\right)_{i+1/2,j}^{n} + \left(V\frac{\partial U}{\partial y}\right)_{i+1/2,j}^{n} \\ \left(U\frac{\partial U}{\partial x}\right)_{i+1/2,j}^{n} &= \frac{1}{\overline{h}_{i+1/2,j}}\left(\frac{{}_u\overline{q}^{x}_{i+1,j}\,{}^{*}U_{i+1,j} - {}_u\overline{q}^{x}_{i,j}\,{}^{*}U_{i,j}}{\Delta x} - U_{i+1/2,j}\frac{{}_u\overline{q}^{x}_{i+1,j} - {}_u\overline{q}^{x}_{i,j}}{\Delta x}\right) \\ \left(V\frac{\partial U}{\partial y}\right)_{i+1/2,j}^{n} &= \frac{1}{\overline{h}_{i+1/2,j}}\left(\frac{{}_v\overline{q}^{x}_{i+1/2,j+1/2}\,{}^{**}U_{i+1/2,j+1/2} - {}_v\overline{q}^{x}_{i+1/2,j-1/2}\,{}^{**}U_{i+1/2,j-1/2}}{\Delta y}\right. \\ &\quad \left. - U_{i+1/2,j}\frac{{}_v\overline{q}^{x}_{i+1/2,j+1/2} - {}_v\overline{q}^{x}_{i+1/2,j-1/2}}{\Delta y}\right) \quad ,\end{aligned}$$

where

$$
\begin{aligned}
{}_u\overline{q}^x_{i,j} &= \frac{1}{2}\left({}_uq_{i+1/2,j} + {}_uq_{i-1/2,j}\right) \qquad & {}_v\overline{q}^x_{i+1/2,j+1/2} &= \frac{1}{2}\left({}_vq_{i,j+1/2} + {}_vq_{i+1,j+1/2}\right) \\
{}_uq_{i+1/2,j} &= {}^*h_{i+1/2,j}U_{i+1/2,j} & {}_vq_{i,j+1/2} &= {}^*h_{i,j+1/2}V_{i,j+1/2}
\end{aligned}
$$

and where the ${}^{**}U$ values are upwinded by

$${}^{**}U_{i+1/2,j+1/2} = \begin{cases} U_{i+1/2,j}\,, & \text{if } \frac{1}{2}(V_{i,j+1/2} + V_{i+1,j+1/2}) \geq 0\,, \\ U_{i+1/2,j+1}\,, & \text{otherwise.} \end{cases}$$

and all depth-averaged velocity values are evaluated at time level t^n. For the y-momentum equation,

$$
\begin{aligned}
(ADV_y)^n_{i,j+1/2} &= \left(U\frac{\partial V}{\partial x}\right)^n_{i,j+1/2} + \left(V\frac{\partial V}{\partial y}\right)^n_{i,j+1/2} \\
\left(U\frac{\partial V}{\partial x}\right)^n_{i,j+1/2} &= \frac{1}{\overline{h}_{i,j+1/2}}\left(\frac{{}_u\overline{q}^y_{i+1/2,j+1/2}\,{}^{**}V_{i+1/2,j+1/2} - {}_u\overline{q}^y_{i-1/2,j+1/2}\,{}^{**}V_{i-1/2,j+1/2}}{\Delta x}\right. \\
&\qquad \left. - V_{i,j+1/2}\frac{{}_u\overline{q}^y_{i+1/2,j+1/2} - {}_u\overline{q}^y_{i-1/2,j+1/2}}{\Delta x}\right) \\
\left(V\frac{\partial V}{\partial y}\right)^n_{i,j+1/2} &= \frac{1}{\overline{h}_{i,j+1/2}}\left(\frac{{}_v\overline{q}^y_{i,j+1}\,{}^*V_{i,j+1} - {}_v\overline{q}^y_{i,j}\,{}^*V_{i,j}}{\Delta y} - V_{i,j+1/2}\frac{{}_v\overline{q}^y_{i,j+1} - {}_v\overline{q}^y_{i,j}}{\Delta y}\right)
\end{aligned}
$$

and values of ${}_u\overline{q}^y$, ${}_v\overline{q}^y$, and ${}^{**}V$ are defined analogously to those given for the x-momentum equation.

Energy head conserving form:

$$
\begin{aligned}
\left(U\frac{\partial U}{\partial x}\right)^n_{i+1/2,j} &= \frac{1}{2}\left(\frac{{}^*U^2_{i+1,j} - {}^*U^2_{i,j}}{\Delta x}\right) \\
\left(V\frac{\partial U}{\partial y}\right)^n_{i+1/2,j} &= {}^*\overline{V}_{i+1/2,j}\left(\frac{{}^{**}U_{i+1/2,j+1/2} - {}^{**}U_{i+1/2,j-1/2}}{\Delta y}\right)\,,
\end{aligned}
$$

where the ${}^{**}U$ values are defined as before and V values are calculated at the U points using

$${}^*\overline{V}_{i+1/2,j} = \begin{cases} \frac{1}{2}\left(V_{i,j-1/2} + V_{i+1,j-1/2}\right), & \text{if } U_{i+1/2,j} \geq 0 \;\&\; \frac{1}{2}\left(V_{i,j+1/2} + V_{i,j-1/2}\right) \geq 0\,, \\ \frac{1}{2}\left(V_{i,j-1/2} + V_{i+1,j-1/2}\right), & \text{if } U_{i+1/2,j} < 0 \;\&\; \frac{1}{2}\left(V_{i+1,j+1/2} + V_{i+1,j-1/2}\right) \geq 0\,, \\ \frac{1}{2}\left(V_{i,j+1/2} + V_{i+1,j+1/2}\right), & \text{otherwise.} \end{cases}$$

Definitions for the y-direction equation are similar.

$$\left(U\frac{\partial V}{\partial x}\right)^n_{i,j+1/2} = {}^*\overline{U}_{i,j+1/2}\left(\frac{{}^{**}V_{i+1/2,j+1/2} - {}^{**}V_{i-1/2,j+1/2}}{\Delta x}\right)$$

$$\left(V\frac{\partial V}{\partial y}\right)^n_{i,j+1/2} = \frac{1}{2}\left(\frac{{}^*V^2_{i,j+1} - {}^*V^2_{i,j}}{\Delta y}\right)$$

The choice between the two conservation methods is again based on the state of the flow in the relevant direction and can be made using Table 3.1.

Surface shear stress terms:

$$\begin{aligned}(WIND)^n_{i+1/2,j} &= \left(\frac{C_w}{h}\frac{\rho_{air}}{\rho_0}\left(W_{10}\right)^2\sin(\Theta)\right)^n_{i+1/2,j}\\ &= \frac{C_w}{\overline{h}^{n+1/2}_{i+1/2}}\frac{\rho_{air}}{\rho_0}\left(W^n_{10}\right)^2\sin(\Theta^n)\end{aligned}$$

The wind speed and direction values at time t^n are determined from input from a meteorological data file.

Bottom shear stress terms:

$$\begin{aligned}(FRICT)^{n+1/2}_{i+1/2,j} &= \left(\frac{g\,n^2_M}{k^2_M}\frac{U\sqrt{U^2+V^2}}{h^{4/3}}\right)^{n+1/2}_{i+1/2,j}\\ &= \gamma_{i+1/2,j}\,U^{n+1}_{i+1/2,j} + \frac{\left(1-\chi_{i+1/2,j}\right)}{\chi_{i+1/2,j}}U^n_{i+1/2,j}\\ \gamma_{i+1/2,j} &= \frac{g\,n^2_M\Delta t}{\left(\overline{h}^{n+1/2}_{i+1/2,j}\right)^{4/3}}\chi_{i+1/2,j}\sqrt{\left(U^n_{i+1/2,j}\right)^2+\left(\overline{\overline{V}}^n_{i+1/2,j}\right)^2}\end{aligned}$$

where

$$\overline{\overline{V}}_{i+1/2,j} = \frac{1}{4}\left(V_{i,j+1/2}+V_{i+1,j+1/2}+V_{i,j-1/2}+V_{i+1,j-1/2}\right) \quad , \quad \chi_{i+1/2,j} = \frac{\widetilde{U}^{n+1}_{i+1/2,j}}{4U^n_{i+1/2,j}} + \frac{3}{4}.$$

The y-momentum discretization is similar,

$$\begin{aligned}(FRICT)^{n+1/2}_{i,j+1/2} &= \left(\frac{g\,n^2_M}{k^2_M}\frac{V\sqrt{U^2+V^2}}{h^{4/3}}\right)^{n+1/2}_{i,j+1/2}\\ &= \gamma_{i,j+1/2}\,V^{n+1}_{i,j+1/2} + \frac{\left(1-\chi_{i,j+1/2}\right)}{\chi_{i,j+1/2}}V^n_{i,j+1/2}\\ \gamma_{i,j+1/2} &= \frac{g\,n^2_M\Delta t}{\left(\overline{h}^{n+1/2}_{i,j+1/2}\right)^{4/3}}\chi_{i,j+1/2}\sqrt{\left(\overline{\overline{U}}^n_{i,j+1/2}\right)^2+\left(V^n_{i+1/2,j}\right)^2}\end{aligned}$$

The roughness values are calculated from equations (2.37)-(2.42) and then converted from Darcy-Weisbach friction factors to Manning's n values using equation (2.36).

Turbulent momentum diffusion terms:

$$
\begin{aligned}
HDIFF_x &= \frac{1}{h}\left(2\frac{\partial}{\partial x}\left(\nu_t h\frac{\partial U}{\partial x}\right)+\frac{\partial}{\partial y}\left(\nu_t h\frac{\partial U}{\partial y}\right)+\frac{\partial}{\partial y}\left(\nu_t h\frac{\partial V}{\partial x}\right)\right)\\
&= \frac{1}{{}^{*}h_{i+1/2,j}}\left(2\frac{(\nu_t h)_{i+1,j}\left(\frac{U_{i+3/2,j}-U_{i+1/2,j}}{\Delta x}\right)-(\nu_t h)_{i,j}\left(\frac{U_{i+1/2,j}-U_{i-1/2,j}}{\Delta x}\right)}{\Delta x}\right.\\
&\quad +\frac{\nu_t\,{}^{**}h_{i+1/2,j+1/2}\left(\frac{U_{i+1/2,j+1}-U_{i+1/2,j}}{\Delta y}\right)-\nu_t\,{}^{**}h_{i+1/2,j-1/2}\left(\frac{U_{i+1/2,j}-U_{i+1/2,j-1}}{\Delta y}\right)}{\Delta x}\\
&\quad \left.+\frac{\nu_t\,{}^{**}h_{i+1/2,j+1/2}\left(\frac{V_{i+1,j+1/2}-V_{i,j+1/2}}{\Delta x}\right)-\nu_t\,{}^{**}h_{i+1/2,j-1/2}\left(\frac{V_{i+1,j-1/2}-V_{i,j-1/2}}{\Delta x}\right)}{\Delta x}\right)\\
HDIFF_y &= \frac{1}{h}\left(2\frac{\partial}{\partial y}\left(\nu_t h\frac{\partial V}{\partial y}\right)+\frac{\partial}{\partial x}\left(\nu_t h\frac{\partial V}{\partial x}\right)+\frac{\partial}{\partial x}\left(\nu_t h\frac{\partial U}{\partial y}\right)\right)\\
&= \frac{1}{{}^{*}h_{i,j+1/2}}\left(2\frac{(\nu_t h)_{i,j+1}\left(\frac{V_{i,j+3/2}-V_{i,j+1/2}}{\Delta y}\right)-(\nu_t h)_{i,j}\left(\frac{V_{i,j+1/2}-V_{i,j-1/2}}{\Delta y}\right)}{\Delta y}\right.\\
&\quad +\frac{\nu_t\,{}^{**}h_{i+1/2,j+1/2}\left(\frac{V_{i+1,j+1/2}-V_{i,j+1/2}}{\Delta x}\right)-\nu_t\,{}^{**}h_{i-1/2,j+1/2}\left(\frac{V_{i,j+1/2}-V_{i-1,j+1/2}}{\Delta x}\right)}{\Delta x}\\
&\quad \left.+\frac{\nu_t\,{}^{**}h_{i+1/2,j+1/2}\left(\frac{U_{i+1/2,j+1}-U_{i+1/2,j}}{\Delta y}\right)-\nu_t\,{}^{**}h_{i-1/2,j+1/2}\left(\frac{U_{i-1/2,j+1}-U_{i-1/2,j}}{\Delta y}\right)}{\Delta x}\right)
\end{aligned}
$$

The values for ν_t can be determined by numerically solving the set of transport equations (2.48) for turbulent kinetic energy and dissipation using the scalar advection algorithm outlined in Section 3.4.3. This, however, has yet to be implemented in the model and a constant ν_t value is assumed.

Coriolis terms:

$$
\begin{aligned}
(COR_x)^n_{i+1/2,j} &= (f_c V)^n_{i+1/2,j}\\
&= \frac{f_c}{2}\left({}^{*}\overline{V}^{n}_{i+1/2,j}+\widetilde{{}^{*}\overline{V}}^{n+1}_{i+1/2,j}\right)\\
(COR_y)^n_{i,j+1/2} &= (-f_c U)^n_{i,j+1/2}\\
&= \frac{-f_c}{2}\left({}^{*}\overline{U}^{n}_{i,j+1/2}+\widetilde{{}^{*}\overline{U}}^{n+1}_{i,j+1/2}\right)
\end{aligned}
$$

Explicit terms in equations (3.27)-(3.30):

$$
\begin{aligned}
\widehat{\zeta}_{i,j} &= \zeta^n_{i,j} + \frac{\Delta t}{2\Delta x}\left({}^*h_{i-1/2,j}\, U^n_{i-1/2,j} - {}^*h_{i+1/2,j}\, U^n_{i+1/2,j}\right) \\
&\quad + \frac{\Delta t}{2\Delta y}\left({}^*h_{i,j-1/2}\, V^n_{i,j-1/2} - {}^*h_{i,j+1/2}\, V^n_{i,j+1/2}\right) + (HYDRO)^n_{i,j} \\
\widehat{U}_{i+1/2,j} &= \frac{1}{1+\gamma_{i+1/2,j}}\Bigg(U^n_{i+1/2,j} - \Delta t\,(ADV_x)^n_{i+1/2,j} + \frac{g\Delta t}{2\Delta x}\left(\zeta^n_{i,j} - \zeta^n_{i+1,j}\right) \\
&\quad + \Delta t\,(WIND_x)^n_{i+1/2,j} - \gamma_{i+1/2,j}\frac{1-\chi_{i+1/2,j}}{\chi_{i+1/2,j}} U^n_{i+1/2,j} + \Delta t\,(HDIFF_x)^n_{i+1/2,j} \\
&\quad + \Delta t\,(COR_x)^n_{i+1/2,j}\Bigg) \\
\widehat{V}_{i,j+1/2} &= \frac{1}{1+\gamma_{i,j+1/2}}\Bigg(V^n_{i,j+1/2} - \Delta t\,(ADV_y)^n_{i,j+1/2} + \frac{g\Delta t}{2\Delta y}\left(\zeta^n_{i,j} - \zeta^n_{i,j+1}\right) \\
&\quad + \Delta t\,(WIND_y)^n_{i,j+1/2} - \gamma_{i,j+1/2}\frac{1-\chi_{i,j+1/2}}{\chi_{i,j+1/2}} V^n_{i,j+1/2} + \Delta t\,(HDIFF_y)^n_{i,j+1/2} \\
&\quad + \Delta t\,(COR_y)^n_{i,j+1/2}\Bigg)
\end{aligned}
$$

C SIFT2D Source Code

A complete copy of the SIFT2D (Semi-Implicit Floodplain Transport model) source code is presented in this Appendix. The code as given here is set up to run a Cosumnes River floodplain simulation using the calibration flood event as boundary conditions and with a tracer release from location TN on the rising limb of the hydrograph. The model is written in the Fortran 90 computer language using double precision. The model files needed to run the simulation are:

SIFT2D.f90 Governing program for SIFT2D. It utilizes the following six Fortran modules to obtain user-defined inputs, numerically solve the governing equations, update the solution, and output flow variables.

SIFT2D_advection.f90 Module containing subroutines used in the calculation of advection and turbulent diffusion in the momentum equation.

SIFT2D_fd.f90 Module containing the finite difference solver. This is the governing program for the finite difference approximations used to numerically solve the governing equations. It uses subroutines from SIFT2D_advection.f90, SIFT2D_friction.f90, and NSPCG.f90, a free-source coding of the preconditioned conjugate gradient method.

SIFT2D_friction.f90 Module containing subroutines used in the determination of bottom friction factors and hydrologic terms in the continuity equation.

SIFT2D_initial.f90 Module containing user-defined information on input files, initial conditions, boundary conditions, time steps, and cell sizes.

SIFT2D_types.f90 Module containing variable declarations and values for geophysical constants.

SIFT2D_update.f90 Module containing subroutines to update the numerical solution after each time step and output flow variables and model parameters.

interp_hydro.m Matlab routine to interpolate boundary condition data to the chosen model time step.

All of the files listed above follow in their complete form. Additional files describing the time series of stage boundaries and meteorological data, topography, and vegetation heights are needed to run the model. Also needed is a copy of the free-source nonsymmetric preconditioned conjugate gradient solver developed at the University of Texas, Austin.

C.1 SIFT2D.f90

```
!***********************************************************************
 PROGRAM SIFT2D
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Governing program for implementation of semi-implicit scheme
!  for 2D floodplain model using Stelling and Duinmeijer 2003
!  advection discretizations
!  ----------------------------------------------------------------------

    USE SIFT2D_types
    USE SIFT2D_initial
    USE SIFT2D_fd
    USE SIFT2D_update
    IMPLICIT NONE

    REAL(KIND = 8) ::  time0, time1, execution_time

    CALL cpu_time (time0)

    nn = 0

    !  ...  Define input parameters and allocate space
    CALL input
    CALL allocate_space
    CALL met_input
    CALL init

    !  ...  Print out initial conditions
    CALL output

    !  ...  Loop over time steps
    time_seconds = zero
    DO nn = 1, nsteps
       time_seconds = FLOAT(nn) * dt
       CALL solve_fd
       CALL output
       CALL update
    END DO

    !  ...  Write model execution time to screen
    CALL cpu_time (time1)
    execution_time = time1 - time0
    WRITE(6,*) 'Execution Time = ', execution_time, ' seconds'
    CALL end_output (execution_time)

 END PROGRAM SIFT2D
```

C.2 SIFT2D_advection.f90

```
!***********************************************************************
 MODULE SIFT2D_advection
!***********************************************************************

    USE SIFT2D_types
    IMPLICIT NONE
    SAVE

 CONTAINS

!***********************************************************************
 SUBROUTINE setM (min1)
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  May 2006
!  Purpose:  Set index values in order to use same subroutines/code blocks
!  for the x and y direction calculations
!-----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  min1

    IF (min1 == 1) THEN
       a1 = 1; b1 = 0; c1 = 2; d1 = 0; dxy = dx; dxyop = dy
       mop = 2; f1 = 1; g1 = -1; p1 = 1; q1 = -2; v1 = 1; w1 = 2
       dt2dxy = dt / (2.0 * dx); gdt2dxy = g * dt2dxy
    ELSE
       a1 = 0; b1 = 1; c1 = 0; d1 = 2; dxy = dy; dxyop = dx
       mop = 1; f1 = -1; g1 = 1; p1 = -2; q1 = 1; v1 = 2; w1 = 1
       dt2dxy = dt / (2.0 * dy); gdt2dxy = g * dt2dxy
    END IF

 END SUBROUTINE setM


!***********************************************************************
 SUBROUTINE upwind_H2 (vel, hLL, hL, hR, hRR, hout)
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Calculate H values at the u pts by the upwind method and
!  slope limiters
!-----------------------------------------------------------------------

    REAL(KIND = 8), INTENT(IN) ::  vel, hLL, hL, hR, hRR
    REAL(KIND = 8), INTENT(INOUT) ::  hout
    REAL(KIND = 8) ::  r, Cr, s_star

    hout = zero

    IF (vel >= zero) THEN
       IF (ABS(hL - hLL) >= tol) THEN
          r = (hR - hL) / (hL - hLL)
          CALL limiters (ilim, r, Cr)
          s_star = hL + Cr / two * (hL - hLL)
          hout = hout + s_star
       ELSE
          hout = hout + hL
```

```
      END IF
   ELSE
      IF (ABS(hRR - hR) >= tol) THEN
         r = (hR - hL) / (hRR - hR)
         CALL limiters (ilim, r, Cr)
         s_star = hR + Cr / two * (hR - hRR)
         hout = hout + s_star
      ELSE
         hout = hout + hR
      END IF
   END IF

 END SUBROUTINE upwind_H2

!**********************************************************************
 SUBROUTINE energy_head (iin, jin, totADVout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Governing program to calculate advection using E head discret
!----------------------------------------------------------------------

   INTEGER, INTENT(IN) ::  iin, jin
   REAL(KIND = 8), INTENT(OUT) ::  totADVout
   REAL(KIND = 8) ::  ustarL, ustarR, udbstarB, udbstarU, vbar
   REAL(KIND = 8) ::  ududx, vdudy

   !  ...  Calculate upwind u values for ududx
   IF (.NOT. domain(iin-a1,jin-b1)) THEN
      CALL upwind_eh2 (up(iin-a1,jin-b1,m), up(iin-a1,jin-b1,m), &
         up(iin,jin,m), up(iin+a1,jin+b1,m), ustarL)
   ELSE
      CALL upwind_eh2 (up(iin-c1,jin-d1,m), up(iin-a1,jin-b1,m), &
         up(iin,jin,m), up(iin+a1,jin+b1,m), ustarL)
   END IF
   IF (.NOT. domain(iin+c1,jin+d1)) THEN
      CALL upwind_eh2 (up(iin-a1,jin-b1,m), up(iin,jin,m), &
         up(iin+a1,jin+b1,m), up(iin+a1,jin+b1,m),ustarR)
   ELSE
      CALL upwind_eh2 (up(iin-a1,jin-b1,m), up(iin,jin,m), &
         up(iin+a1,jin+b1,m), up(iin+c1,jin+d1,m),ustarR)
   END IF

   !  ...  Calculate ududx
   ududx = (ustarR**two - ustarL**two) / (two * dxy)

   !  ...  Calculate vbar as part of vdudy determination
   CALL vbar_calc (up(iin,jin,m), up(iin,jin,mop), &
      up(iin+a1,jin+b1,mop), up(iin-b1,jin-a1,mop), &
      up(iin+f1,jin+g1,mop), vbar)
   !  ...  Calculate upwind u values for vdudy
   CALL dbstar_calc(iin, jin, udbstarU, udbstarB)

   !  ...  Calculate vdudy
   vdudy = vbar * (udbstarU - udbstarB) / dxyop

   !  ...  Calculate total advection using E head conservation
   totADVout = ududx + vdudy
```

```
 END SUBROUTINE energy_head

!**********************************************************************
 SUBROUTINE dbstar_calc (iin, jin, udbstarUout, udbstarBout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  May 2006
!  Purpose:  Calculate doublestar quantities
!----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  iin, jin
    REAL(KIND = 8), INTENT(OUT) ::  udbstarUout, udbstarBout

    IF ((.NOT. domain(iin-b1,jin-a1)) .OR. &
       (.NOT. domain(iin+f1,jin+g1))) THEN
       udbstarBout = zero
    ELSE IF ((.NOT. domain(iin-d1,jin-c1)) .OR. &
       (.NOT. domain(iin+p1,jin+q1))) THEN
       CALL upwind2_eh2 (up(iin-b1,jin-a1,mop), up(iin+f1,jin+g1,mop), &
          up(iin-b1,jin-a1,m), up(iin-b1,jin-a1,m), up(iin,jin,m), &
          up(iin+b1,jin+a1,m), udbstarBout)
    ELSE IF ((.NOT. domain(iin+b1,jin+a1)) .OR. &
       (.NOT. domain(iin+1,jin+1))) THEN
       CALL upwind2_eh2 (up(iin-b1,jin-a1,mop), up(iin+f1,jin+g1,mop), &
          up(iin-d1,jin-c1,m), up(iin-b1,jin-a1,m), up(iin,jin,m), &
          up(iin,jin,m), udbstarBout)
    ELSE
       CALL upwind2_eh2 (up(iin-b1,jin-a1,mop), up(iin+f1,jin+g1,mop), &
          up(iin-d1,jin-c1,m), up(iin-b1,jin-a1,m), up(iin,jin,m), &
          up(iin+b1,jin+a1,m), udbstarBout)
    END IF
    IF ((.NOT. domain(iin+b1,jin+a1)) .OR. &
       (.NOT. domain(iin+1,jin+1))) THEN
       udbstarUout = zero
    ELSE IF ((.NOT. domain(iin-b1,jin-a1)) .OR. &
       (.NOT. domain(iin+f1,jin+g1))) THEN
       CALL upwind2_eh2 (up(iin,jin,mop), up(iin+a1,jin+b1,mop), &
          up(iin,jin,m), up(iin,jin,m), up(iin+b1,jin+a1,m), &
          up(iin+d1,jin+c1,m), udbstarUout)
    ELSE IF ((.NOT. domain(iin+d1,jin+c1)) .OR. &
       (.NOT. domain(iin+v1,jin+w1))) THEN
       CALL upwind2_eh2 (up(iin,jin,mop), up(iin+a1,jin+b1,mop), &
          up(iin-b1,jin-a1,m), up(iin,jin,m), up(iin+b1,jin+a1,m), &
          up(iin+b1,jin+a1,m), udbstarUout)
    ELSE
       CALL upwind2_eh2 (up(iin,jin,mop), up(iin+a1,jin+b1,mop), &
          up(iin-b1,jin-a1,m), up(iin,jin,m), up(iin+b1,jin+a1,m), &
          up(iin+d1,jin+c1,m), udbstarUout)
    END IF

 END SUBROUTINE dbstar_calc


!**********************************************************************
 SUBROUTINE upwind_eh2 (uLL, uL, uR, uRR, uout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Calculate u or v values at the zeta pts by the upwind method
```

```
!  and slope limiters for the E head cons advection discret
!-----------------------------------------------------------------------

    REAL(KIND = 8), INTENT(IN) ::  uLL, uL, uR, uRR
    REAL(KIND = 8), INTENT(OUT) ::  uout
    REAL(KIND = 8) ::  r, Cr

    IF (half * (uL + uR) >= zero) THEN
       IF (ABS(uL - uLL) >= tol) THEN
          r = (uR - uL) / (uL - uLL)
          CALL limiters (ilim, r, Cr)
          uout = uL + Cr / two * (uL - uLL)
       ELSE
          uout = uL
       END IF
    ELSE
       IF (ABS(uRR - uR) >= tol) THEN
          r = (uR - uL) / (uRR - uR)
          CALL limiters (ilim, r, Cr)
          uout = uR + Cr / two * (uR - uRR)
       ELSE
          uout = uR
       END IF
    END IF

 END SUBROUTINE upwind_eh2


!***********************************************************************
 SUBROUTINE upwind2_eh2 (vL, vR, uBB, uB, uU, uUU, udbstarout)
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Calculate udblstar for use in advection E head discretization
!  using upwind and slope limiters
!-----------------------------------------------------------------------

    REAL(KIND = 8), INTENT(IN) ::  vL, vR, uBB, uB, uU, uUU
    REAL(KIND = 8), INTENT(OUT) ::  udbstarout
    REAL(KIND = 8) ::  r, Cr

    IF (half * (vL + vR) >= zero) THEN
       IF (ABS(uB - uBB) >= tol) THEN
          r = (uU - uB) / (uB - uBB)
          CALL limiters (ilim, r, Cr)
          udbstarout = uB + Cr / two * (uB - uBB)
       ELSE
          udbstarout = uB
       END IF
    ELSE
       IF (ABS(uUU - uU) >= tol) THEN
          r = (uU - uB) / (uUU - uU)
          CALL limiters (ilim, r, Cr)
          udbstarout = uU + Cr / two * (uU - uUU)
       ELSE
          udbstarout = uU
       END IF
    END IF
```

```
 END SUBROUTINE upwind2_eh2


!**********************************************************************
 SUBROUTINE vbar_calc (uin, vUL, vUR, vBL, vBR, vbarout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Calculate vbar for use in advection E head discretization
!----------------------------------------------------------------------

    REAL(KIND = 8), INTENT(IN) ::  uin, vUL, vUR, vBL, vBR
    REAL(KIND = 8), INTENT(OUT) ::  vbarout

    IF (uin >= zero .and.  half * (vUL + vBL) >= zero) THEN
       vbarout = half * (vBL + vBR)
    ELSE IF (uin < zero .and.  half * (vUR + vBR) >= zero) THEN
       vbarout = half * (vBL + vBR)
    ELSE
       vbarout = half * (vUL + vUR)
    END IF

 END SUBROUTINE vbar_calc


!**********************************************************************
 SUBROUTINE momentum (iin, jin, totADVout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Governing program to calculate advection using moment cons.
!----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  iin, jin
    REAL(KIND = 8), INTENT(OUT) ::  totADVout
    REAL(KIND = 8) ::  ustarL, ustarR, uqbarL, uqbarR
    REAL(KIND = 8) ::  udbstarB, udbstarU, vqbarB, vqbarU
    REAL(KIND = 8) ::  ududx, vdudy, Hbar

    !  ...  Calculate upwind u values for ududx
    IF (.NOT. domain(iin-a1,jin-b1)) THEN
       CALL upwind_m2 (up(iin-a1,jin-b1,m), up(iin-a1,jin-b1,m), &
          up(iin,jin,m), up(iin+a1,jin+b1,m), Hstar(iin-a1,jin-b1,m),&
          Hstar(iin,jin,m), ustarL)
    ELSE
       CALL upwind_m2 (up(iin-c1,jin-d1,m), up(iin-a1,jin-b1,m), &
          up(iin,jin,m), up(iin+a1,jin+b1,m), Hstar(iin-a1,jin-b1,m),&
          Hstar(iin,jin,m), ustarL)
    END IF
    IF (.NOT. domain(iin+c1,jin+d1)) THEN
       CALL upwind_m2 (up(iin-a1,jin-b1,m), up(iin,jin,m), &
          up(iin+a1,jin+b1,m), up(iin+a1,jin+b1,m), Hstar(iin,jin,m),&
          Hstar(iin+a1,jin+b1,m), ustarR)
    ELSE
       CALL upwind_m2 (up(iin-a1,jin-b1,m), up(iin,jin,m), &
          up(iin+a1,jin+b1,m), up(iin+c1,jin+d1,m), Hstar(iin,jin,m),&
          Hstar(iin+a1,jin+b1,m), ustarR)
    END IF
```

```
    !  ...  Calculate uqbars
    uqbarL = half * (Hstar(iin-a1,jin-b1,m) * up(iin-a1,jin-b1,m) + &
       Hstar(iin,jin,m) * up(iin,jin,m))
    uqbarR = half * (Hstar(iin,jin,m) * up(iin,jin,m) + &
       Hstar(iin+a1,jin+b1,m) * up(iin+a1,jin+b1,m))

    !  ...  Calculate ududx
    Hbar = half * (H(iin,jin) + H(iin+a1,jin+b1))
    ududx = one / Hbar * ((uqbarR * ustarR - uqbarL * ustarL) / dxy &
       - up(iin,jin,m) * (uqbarR - uqbarL) / dxy)

    !  ...  Calculate upwind u values for vdudy
    CALL dbstar_calc(iin, jin, udbstarU, udbstarB)

    !  ...  Calculate vqbars
    vqbarB = half * (Hstar(iin-b1,jin-a1,mop) * up(iin-b1,jin-a1,mop) &
       + Hstar(iin+f1,jin+g1,mop) * up(iin+f1,jin+g1,mop))
    vqbarU = half * (Hstar(iin,jin,mop) * up(iin,jin,mop) &
       + Hstar(iin+a1,jin+b1,mop) * up(iin+a1,jin+b1,mop))

    !  ...  Calculate vdudy
    vdudy = one / Hbar * ((vqbarU * udbstarU - vqbarB * udbstarB) &
       / dxyop - up(iin,jin,m) * (vqbarU - vqbarB) / dxyop)

    !  ...  Calculate total advection using momentum conservation
    totADVout = ududx + vdudy

 END SUBROUTINE momentum


!**********************************************************************
 SUBROUTINE upwind_m2 (uLL, uL, uR, uRR, hL, hR, uout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Calculate u or v values at the zeta pts by the upwind method
!  and slope limiters for the momentum cons advection discret
!----------------------------------------------------------------------

    REAL(KIND = 8), INTENT(IN) ::  uLL, uL, uR, uRR, hL, hR
    REAL(KIND = 8), INTENT(OUT) ::  uout
    REAL(KIND = 8) ::  r, Cr

    IF (half * (hL * uL + hR * uR) >= zero) THEN
       IF (ABS(uL - uLL) >= tol) THEN
          r = (uR - uL) / (uL - uLL)
          CALL limiters (ilim, r, Cr)
          uout = uL + Cr / two * (uL - uLL)
       ELSE
          uout = uL
       END IF
    ELSE
       IF (ABS(uRR - uR) >= tol) THEN
          r = (uR - uL) / (uRR - uR)
          CALL limiters (ilim, r, Cr)
          uout = uR + Cr / two * (uR - uRR)
       ELSE
          uout = uR
       END IF
```

```
   END IF

 END SUBROUTINE upwind_m2


!**********************************************************************
 SUBROUTINE turbulent_diff (iin, jin, turb_diff_out)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  December 2006
!  Purpose:  Calculate turbulent diffusion term in the momentum eqn
!----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  iin, jin
    REAL(KIND = 8), INTENT(OUT) ::  turb_diff_out
    REAL(KIND = 8) ::  hdbstarT, hdbstarB

    !  ...  First term:  2 d/dx (H dU/dx)
    turb_diff_out = two * (H(iin+a1,jin+b1) * (up(iin+a1,jin+b1,m) - &
       up(iin,jin,m)) / dxy - H(iin,jin) * (up(iin,jin,m) - &
       up(iin-a1,jin-b1,m)) / dxy) / dxy

    !  ...  Calculate H double star values
    IF (half * (up(iin,jin,m) + up(iin+b1,jin+a1,m)) >= zero) THEN
       IF (half * (up(iin,jin,mop) + up(iin+a1,jin+b1,mop)) >=zero) THEN
          hdbstarT = H(iin,jin)
       ELSE
          hdbstarT = H(iin+b1,jin+a1)
       END IF
    ELSE
       IF (half * (up(iin,jin,mop) + up(iin+a1,jin+b1,mop)) >=zero) THEN
          hdbstarT = H(iin+a1,jin+b1)
       ELSE
          hdbstarT = H(iin+1,jin+1)
       END IF
    END IF
    IF (half * (up(iin,jin,m) + up(iin-b1,jin-a1,m)) >= zero) THEN
       IF (half * (up(iin-b1,jin-a1,mop) + up(iin+f1,jin+g1,mop)) >= &
          zero) THEN
          hdbstarB = H(iin-b1,jin-a1)
       ELSE
          hdbstarB = H(iin,jin)
       END IF
    ELSE
       IF (half * (up(iin-b1,jin-a1,mop) + up(iin+f1,jin+g1,mop)) >= &
          zero) THEN
          hdbstarB = H(iin+f1,jin+g1)
       ELSE
          hdbstarB = H(iin+a1,jin+b1)
       END IF
    END IF

    !  ...  Second term:  d/dy (H dU/dy)
    turb_diff_out = turb_diff_out + (hdbstarT * (up(iin+b1,jin+a1,m) - &
       up(iin,jin,m)) / dxyop - hdbstarB * (up(iin,jin,m) - &
       up(iin-b1,jin-a1,m)) / dxyop) / dxyop
    !  ...  Third term:  d/dy (H dV/dx)
    turb_diff_out = turb_diff_out + (hdbstarT * (up(iin+a1,jin+b1,mop) &
```

```
      - up(iin,jin,mop)) / dxy - hdbstarB * (up(iin+f1,jin+g1,mop) - &
      up(iin-b1,jin-a1,mop)) / dxy) / dxyop

   !  ...  If water too shallow, estimate with KH (d2U/dx2 + d2U/dy2)
   IF (Hstar(iin,jin,m) > shtol) THEN
      turb_diff_out = turb_diff_out * KH / Hstar(iin,jin,m)
   ELSE
      turb_diff_out = KH * ((up(iin+a1,jin+b1,m)-two*up(iin,jin,m) + &
         up(iin-a1,jin-b1,m)) / dxy**two + (up(iin+b1,jin+a1,m) - &
         two*up(iin,jin,m) + up(iin-b1,jin-a1,m)) / dxyop**two)
   END IF

 END SUBROUTINE turbulent_diff


!**********************************************************************
 SUBROUTINE limiters (limnum, r, Cr)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Calculate C(r) for 4 different limiters
!----------------------------------------------------------------------

   INTEGER, INTENT(IN) ::  limnum
   REAL(KIND = 8), INTENT(IN) ::  r
   REAL(KIND = 8), INTENT(OUT) ::  Cr

   IF (limnum == 1) THEN
      Cr = (r + ABS(r)) / (1 + ABS(r)) !  van Leer
   ELSE IF (limnum == 2) THEN
      Cr = MAX(zero, MIN(two*r, (one+r)/two, two)) !  MC
   ELSE IF (limnum == 3) THEN
      Cr = MAX(zero, MIN(one, r)) !  minmod
   ELSE IF (limnum == 4) THEN
      Cr = MAX(zero, MIN(one, two*r), MIN(two, r)) !  superbee
   ELSE IF (limnum == 5) THEN
      Cr = one !  2nd order
   END IF

 END SUBROUTINE limiters

 END MODULE SIFT2D_advection
```

C.3 SIFT2D_fd.f90

```
!***********************************************************************
 MODULE SIFT2D_fd
!***********************************************************************

    USE SIFT2D_types
    USE SIFT2D_initial
    USE SIFT2D_advection
    USE SIFT2D_friction
    IMPLICIT NONE
    EXTERNAL MIC2,CG
    SAVE

 CONTAINS

!***********************************************************************
 SUBROUTINE solve_fd
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Solve 2D shallow water eqns using Stelling and Duinmeijer
!  discretizations with the semi-implicit method
!----------------------------------------------------------------------

    !  ...  Allocatable arrays
    REAL(KIND = 8), ALLOCATABLE, DIMENSION (:,:)  ::  exs
    REAL(KIND = 8), ALLOCATABLE, DIMENSION (:,:,:)  ::  exu
    REAL(KIND = 8), ALLOCATABLE, DIMENSION (:,:)  ::  Hnew
    REAL(KIND = 8), ALLOCATABLE, DIMENSION (:,:,:)  ::  gamma
    REAL(KIND = 8), ALLOCATABLE, DIMENSION (:,:)  ::  sx, sy, rr
    REAL(KIND = 8), ALLOCATABLE, DIMENSION (:,:,:)  ::  scflux

    !  ...  Other local variables
    INTEGER ::  istat !  Error code
    INTEGER ::  i, j, niter !  Loop counters
    INTEGER ::  maxnz1, nw1, inw1 !  NSPCG call parameters
    INTEGER ::  hrnum !  For met data
    INTEGER ::  int1, int2, int3 !  Scalar var output
    REAL(KIND = 8) ::  turb_diff, chi, chi2, Man !  Momentum eqn terms
    REAL(KIND = 8) ::  ADV, udbar, hydro !  Momentum eqn terms
    REAL(KIND = 8) ::  cor, UU, VV, wind !  Cor and wind calcs
    REAL(KIND = 8) ::  II, JJ, RRR, SS, r, Cr !  Scalar var calcs
    REAL(KIND = 8) ::  sstarN, sstarS, TSin, TNin !  TN and TS stages
    REAL(KIND = 8) ::  sstarE, sstarW, TEout, TWout !  TE and TW stages
    REAL(KIND = 8) ::  FPmass, treleases, treleasee !  Scalar var calcs
    REAL(KIND = 8) ::  oldSmt, newSmt, scu1 !  Sm tracer values
    REAL(KIND = 8) ::  scH1, scH2, scH3, scH4 !  For scalar calcs
    LOGICAL ::  sc1, sc2, sc3, sc4

    !  ...  Allocate local arrays
    ALLOCATE (exs(1:im1,1:jm1), STAT = istat)
       IF (istat /= 0) CALL allocate_error (istat, 11)
    ALLOCATE (exu(1:im1,1:jm1,2), STAT = istat)
       IF (istat /= 0) CALL allocate_error (istat, 12)
    ALLOCATE (Hnew(1:im1,1:jm1), STAT = istat)
       IF (istat /= 0) CALL allocate_error (istat, 13)
    ALLOCATE (gamma(1:im1,1:jm1,2), STAT = istat)
```

```
   IF (istat /= 0) CALL allocate_error (istat, 14)
ALLOCATE (sx(1:im1,1:jm1), sy(1:im1,1:jm1), STAT = istat)
   IF (istat /= 0) CALL allocate_error (istat, 15)
ALLOCATE (rr(1:im1,1:jm1), STAT = istat)
   IF (istat /= 0) CALL allocate_error (istat, 16)
ALLOCATE (scflux(1:im1,1:jm1,2), STAT = istat)
   IF (istat /= 0) CALL allocate_error (istat, 17)

!  ...  Determine hour number of simulation
hrnum = CEILING(time_seconds/(sixty*sixty))
IF (stflow > 0) THEN
   hrnum = hrnum - stflow
   IF (hrnum < 1) hrnum = 1
END IF

!  ...  Loop over number of iterations for friction
DO niter = 1,maxiter

!  ...  Initialize more working arrays
exs = zero; exu = zero
sx = zero; sy = zero
rr = zero; zeta = zero
aa = zero; ds = zero
H = zero; Hnew = zero
Hstar = zero; gamma = zero
scflux = zero; ManN = zero

!  ============ Calculate variables with upwind ====================

!  ...  Define H, Hnew, and calculate Hstar at u and v pts by upwind
H = D + sp
Hnew = D + s
DO m = 1,2
   CALL setM(m)
DO j = j1-1,jm
DO i = i1-1,im
   !  ...  If before or after wet boundary, set = 0, if on
   !  ...  boundary, set = adjacent cell height, and calculate
   !  ...  using first or second order in interior depending on
   !  ...  proximity to boundaries
   IF ((.NOT. domain(i,j)) .AND. (.NOT. domain(i+a1,j+b1))) THEN
      Hstar(i,j,m) = zero
   ELSE IF ((.NOT. domain(i,j)) .AND. domain(i+a1,j+b1)) THEN
      Hstar(i,j,m) = H(i+a1,j+b1)
   ELSE IF (domain(i,j) .AND. (.NOT. domain(i+a1,j+b1))) THEN
      Hstar(i,j,m) = H(i,j)
   ELSE
      CALL upwind_H2 (up(i,j,m), H(i-a1,j-b1), H(i,j), &
         H(i+a1,j+b1), H(i+c1,j+d1), Hstar(i,j,m))
      IF (up(i,j,m) == zero .AND. (H(i,j)== zero .OR. H(i+a1,j+b1) &
         == zero)) Hstar(i,j,m) = zero
   END IF
END DO
END DO
END DO

!  ============ Momentum explicit terms ============================

!  ...  Contribution from time derivative term
exu(i1:im,j1:jm,:)  = up(i1:im,j1:jm,:)
```

```
!  ...  Contribution from advection, turbulent diffusion, and the
!  ...  explicit parts of friction and barotropic transport for u
DO m = 1,2
   CALL setM(m)
DO j = j1,jm
DO i = i1,im
   !  ...  Conditions for setting the explicit momentum part = 0
   IF ((.NOT. domain(i,j)) .OR. (.NOT. domain(i+a1,j+b1))) THEN
      exu(i,j,m) = zero; gamma(i,j,m) = zero
   ELSE IF (H(i,j) < shtol .AND. H(i+a1,j+b1) < shtol) THEN
      exu(i,j,m) = zero; gamma(i,j,m) = zero
   ELSE IF ((sp(i+a1,j+b1) > sp(i,j) .AND. H(i+a1,j+b1) < shtol) &
      .OR. (sp(i,j) > sp(i+a1,j+b1) .AND. H(i,j) < shtol)) THEN
      exu(i,j,m) = zero; gamma(i,j,m) = zero
   ELSE
   !  ...  IF structure to evaluate ADV terms
   !  ...  Momentum conservation only
   IF (iadv == 1) THEN
      CALL momentum (i, j, ADV)
   !  ...  Energy head conservation only
   ELSE IF (iadv == 2) THEN
      CALL energy_head (i, j, ADV)
   !  ...  Dynamic scheme
   ELSE IF (iadv == 3) THEN
      IF (up(i,j,m) >= zero) THEN
         IF ((up(i,j,m) - up(i-a1,j-b1,m)) / dxy > eps) THEN
            CALL energy_head (i, j, ADV)
         ELSE
            CALL momentum (i, j, ADV)
         END IF
      ELSE
         IF ((up(i+a1,j+b1,m) - up(i,j,m)) / dxy > eps) THEN
            CALL energy_head (i, j, ADV)
         ELSE
            CALL momentum (i, j, ADV)
         END IF
      END IF
   !  ...  No advection
   ELSE
      ADV = zero
   END IF
   !  ...  Coriolis force contribution
   IF (up(i,j,m) >= zero) THEN
      UU = half * (up(i,j,mop) + up(i-b1,j-a1,mop))
   ELSE
      UU = half * (up(i+a1,j+b1,mop) + up(i+f1,j+g1,mop))
   END IF
   IF (u(i,j,m) >= zero) THEN
      VV = half * (u(i,j,mop) + u(i-b1,j-a1,mop))
   ELSE
      VV = half * (u(i+a1,j+b1,mop) + u(i+f1,j+g1,mop))
   END IF
   cor = FLOAT(f1) * fc * (half * (UU + VV))
   !  ...  Wind stress contribution
   !  ...  For constant wind speed simulations
   !wind = wind1
   !  ...  For variable wind speed
   wind = Cw * (rhoair / rho0) * Windsp(hrnum)**two
```

```
   wind = wind / (quart * (H(i,j) + H(i+a1,j+b1) + Hnew(i,j) &
      + Hnew(i+a1,j+b1)))
   !IF (m == 1) wind = wind * SIN(theta/1.8D2*pi)
   !IF (m == 2) wind = wind * COS(theta/1.8D2*pi)
   IF (m == 1) wind = wind * SIN((Winddir(hrnum)+1.8D2)/1.8D2*pi)
   IF (m == 2) wind = wind * COS((Winddir(hrnum)+1.8D2)/1.8D2*pi)
   !  ...  Frictional contribution
   IF (ABS(up(i,j,m)) >= shtol) THEN
      chi = quart * u(i,j,m) / up(i,j,m) + 0.75D0
   ELSE
      chi = one
   END IF
   CALL bottom_friction (i, j, Man)
   udbar = quart * (up(i-b1,j-a1,mop) + up(i+f1,j+g1,mop) + &
      up(i,j,mop) + up(i+a1,j+b1,mop))
   chi2 = dt*g*Man**two / (Km**two * (quart*(H(i,j) + H(i+a1,j+b1) &
      + Hnew(i,j) + Hnew(i+a1,j+b1)))**(4.0D0/3.0D0)) &
      * SQRT(up(i,j,m)**two + udbar**two)
   gamma(i,j,m) = chi2 * chi
   !  ...  Turbulent diffusion contribution
   CALL turbulent_diff (i, j, turb_diff)
   !  ...  Adding everything together
   exu(i,j,m) = exu(i,j,m) - dt * ADV + gdt2dxy * &
      (sp(i,j) - sp(i+a1,j+b1)) + dt * turb_diff - &
      (one - chi) * chi2 * up(i,j,m) + cor + wind
   exu(i,j,m) = exu(i,j,m) / (one + gamma(i,j,m))
   END IF
END DO
END DO
END DO

!  ============ Continuity explicit terms ============================

!  ...  Contribution from time derivative term
exs(i1:im,j1:jm) = sp(i1:im,j1:jm)

!  ...  Explicit terms in the continuity eqn
DO m = 1,2
   CALL setM(m)
DO i = i1,im
DO j = j1,jm
   exs(i,j) = exs(i,j) + dt2dxy * (Hstar(i-a1,j-b1,m) * &
      up(i-a1,j-b1,m) - Hstar(i,j,m) * up(i,j,m))
   IF (domain(i,j)) THEN
      CALL hydrology(i, j, hydro)
      exs(i,j) = exs(i,j) + hydro
   END IF
END DO
END DO
END DO

!  ...  Explicit terms in momentum eqn
DO m = 1,2
   CALL setM(m)
DO i = i1,im
DO j = j1,jm
   exs(i,j) = exs(i,j) + dt2dxy * (Hstar(i-a1,j-b1,m) * &
      exu(i-a1,j-b1,m) - Hstar(i,j,m) * exu(i,j,m))
END DO
```

```
END DO
END DO


!  ============ Solve continuity equation for zeta ==================

!  ...  Define matrix sx
m = 1; CALL setM(m)
DO j = j1,jm
DO i = i1-1,im
   IF ((.NOT. domain(i,j)) .OR. (.NOT. domain(i+a1,j+b1))) THEN
      sx(i,j) = zero
   ELSE
      sx(i,j) = gdt2dxy * dt2dxy / (one+gamma(i,j,m)) * Hstar(i,j,m)
   END IF
END DO
END DO

!  ...  Define matrix sy
m = 2; CALL setM(m)
DO i = i1,im
DO j = j1-1,jm
   IF ((.NOT. domain(i,j)) .OR. (.NOT. domain(i+a1,j+b1))) THEN
      sy(i,j) = zero
   ELSE
      sy(i,j) = gdt2dxy * dt2dxy / (one+gamma(i,j,m)) * Hstar(i,j,m)
   END IF
END DO
END DO

!  ...  Define matrix rr
DO i = i1,im
DO j = j1,jm
   rr(i,j) = one + sx(i-1,j) + sx(i,j) + sy(i,j-1) + sy(i,j)
END DO
END DO

!  ...  Prescribe zetas for wall cells
rr(im,TNb:TNt) = one !  Factor multiplying zeta
rr(im,TSb:TSt) = one
rr(TEl:TEr,j1) = one
rr(TWl:TWr,j1) = one
IF (niter == 1) CALL flow_input(TSin, TNin, TEout, TWout)
sstarN = -TNd + TNin
DO j = TNb,TNt
   IF (sstarN > -D(im,j)) THEN
      exs(im,j) = sstarN
   ELSE
      exs(im,j) = -D(im,j)
   END IF
END DO
sstarS = -TSd + TSin
DO j = TSb,TSt
   IF (sstarS > -D(im,j)) THEN
      exs(im,j) = sstarS
   ELSE
      exs(im,j) = -D(im,j)
   END IF
END DO
```

```
sstarE = -TEd + TEout
DO i = TEl,TEr
   IF (sstarE > -D(i,j1)) THEN
      exs(i,j1) = sstarE
   ELSE
      exs(i,j1) = -D(i,j1)
   END IF
END DO
sstarW = -TWd + TWout
DO i = TWl,TWr
   IF (sstarW > -D(i,j1)) THEN
      exs(i,j1) = sstarW
   ELSE
      exs(i,j1) = -D(i,j1)
   END IF
END DO

!  ...  Put variables into storage form required for NSPCG call
DO i = i1,im
DO j = j1,jm
   m = (i-i1) * (jm-j1+1) + (j-j1) + 1
   aa(m,1) = rr(i,j) !  Center diagonal
   aa(m,2) = -sy(i,j) !  Upper diagonal
   aa(m,3) = -sx(i,j) !  Far upper diagonal
   ds(m) = exs(i,j)
   zeta(m) = sp(i,j)
   IF (i == im-1 .AND. (j >= TNb .AND. j <= TNt)) THEN
      aa(m,3) = zero !  Set other coefficients to zero
   END IF
   IF (i == im .AND. (j >= TNb-1 .AND. j <= TNt)) THEN
      aa(m,2) = zero
   END IF
   IF (i == im-1 .AND. (j >= TSb .AND. j <= TSt)) THEN
      aa(m,3) = zero
   END IF
   IF (i == im .AND. (j >= TSb-1 .AND. j <= TSt)) THEN
      aa(m,2) = zero
   END IF
   IF (j == j1 .AND. (i >= TEl .AND. i <= TEr)) THEN
      aa(m,2) = zero
   END IF
   IF (j == j1 .AND. (i >= TEl-1 .AND. i <= TEr)) THEN
      aa(m,3) = zero
   END IF
   IF (j == j1 .AND. (i >= TWl .AND. i <= TWr)) THEN
      aa(m,2) = zero
   END IF
   IF (j == j1 .AND. (i >= TWl-1 .AND. i <= TWr)) THEN
      aa(m,3) = zero
   END IF
   IF ((i == im-1 .AND. sstarN > -D(i+1,j)) .AND. &
      (j >= TNb .AND. j <= TNt)) THEN
      ds(m) = ds(m) + sx(i,j) * sstarN
   END IF
   IF ((i == im-1 .AND. sstarS > -D(i+1,j)) .AND. &
      (j >= TSb .AND. j <= TSt)) THEN
      ds(m) = ds(m) + sx(i,j) * sstarS
   END IF
   IF ((j == j1+1 .AND. sstarE > -D(i,j-1)) .AND. &
```

```
      (i >= TEl .AND. i <= TEr)) THEN
      ds(m) = ds(m) + sy(i,j-1) * sstarE
   END IF
   IF ((j == j1+1 .AND. sstarW > -D(i,j-1)) .AND. &
      (i >= TWl .AND. i <= TWr)) THEN
      ds(m) = ds(m) + sy(i,j-1) * sstarW
   END IF
   IF ((i == im .AND. sstarN > -D(i,j-1)) .AND. j == TNt+1) THEN
      ds(m) = ds(m) + sy(i,j-1) * sstarN
   END IF
   IF ((i == im .AND. sstarN > -D(i,j+1)) .AND. j == TNb-1) THEN
      ds(m) = ds(m) + sy(i,j) * sstarN
   END IF
   IF ((i == im .AND. sstarS > -D(i,j-1)) .AND. j == TSt+1) THEN
      ds(m) = ds(m) + sy(i,j-1) * sstarS
   END IF
   IF ((i == im .AND. sstarS > -D(i,j+1)) .AND. j == TSb-1) THEN
      ds(m) = ds(m) + sy(i,j) * sstarS
   END IF
   IF ((j == j1 .AND. sstarE > -D(i-1,j)) .AND. i == TEr+1) THEN
      ds(m) = ds(m) + sx(i-1,j) * sstarE
   END IF
   IF ((j == j1 .AND. sstarE > -D(i+1,j)) .AND. i == TEl-1) THEN
      ds(m) = ds(m) + sx(i,j) * sstarE
   END IF
   IF ((j == j1 .AND. sstarW > -D(i-1,j)) .AND. i == TWr+1) THEN
      ds(m) = ds(m) + sx(i-1,j) * sstarW
   END IF
   IF ((j == j1 .AND. sstarW > -D(i+1,j)) .AND. i == TWl-1) THEN
      ds(m) = ds(m) + sx(i,j) * sstarW
   END IF
END DO
END DO

!  ...  Call default parameters and reset some
CALL DFAULT (iparm, rparm)
iparm(2) = 500 !  Maximum of 500 iterations
iparm(3) = 0 !  Output fatal error messages only
rparm(1) = 1.0D-8 !  Relative accuracy (default = 1.0E-6)

!  ...  Set workspace parameters (these may be modified by nspcg)
maxnz1 = maxnz
nw1 = nw
inw1 = inw

!  ...  Call NSPCG
CALL nspcg (MIC2,CG,numcells,maxnz,numcells,maxnz1,aa,jcoef,jp,ip,&
   zeta,ubarp,ds,wksp,iwksp,nw1,inw1,iparm,rparm,ier)

!  ...  Read solution back into array
s = zero
DO m = 1,numcells
   i = (m-1) / (jm-j1+1) + i1
   j = m + (j1-1) - (i-i1) * (jm-j1+1)
   s(i,j) = zeta(m)
END DO
DO j = 1,jm1
   s(1,j) = s(i1,j)
   s(im1,j) = s(im,j)
```

```
END DO
DO i = 1,im1
   s(i,1) = s(i,j1)
   s(i,jm1) = s(i,jm)
END DO

!  ...  Set depths that may have become slightly negative back to zero
Hnew = D + s
DO i = 1,im1
DO j = 1,jm1
   IF (Hnew(i,j) < zero) THEN
      Hnew(i,j) = zero
      s(i,j) = -D(i,j)
   END IF
END DO
END DO

!  ============ Calculate horizontal velocity components ============

u = zero
DO m = 1,2
   CALL setM(m)
DO j = j1-1,jm
DO i = i1-1,im
   IF ((.NOT. domain(i,j)) .OR. (.NOT. domain(i+a1,j+b1))) THEN
      u(i,j,m) = zero
   ELSE IF (Hnew(i,j) < shtol .AND. Hnew(i+a1,j+b1) < shtol) THEN
      u(i,j,m) = zero
   ELSE IF ((s(i+a1,j+b1) > s(i,j) .AND. Hnew(i+a1,j+b1) < shtol) &
      .OR. (s(i,j) > s(i+a1,j+b1) .AND. Hnew(i,j) < shtol)) THEN
      u(i,j,m) = zero
   ELSE
      u(i,j,m) = exu(i,j,m) - gdt2dxy / (one + gamma(i,j,m)) * &
         (s(i+a1,j+b1) - s(i,j))
   END IF
END DO
END DO
END DO

!  ...  Estimate inlet and outlet velocities with adjacent cell vels
j = j1
DO i = TEl,TEr
   u(i,j-1,2) = u(i,j,2)
END DO
DO i = TWl,TWr
   u(i,j-1,2) = u(i,j,2)
END DO
i = im
DO j = TSb,TSt
   u(i,j,1) = u(i-1,j,1)
END DO
!  ...  For TN set initial velocity at 30 degrees to horizontal
DO j = TNb+1,TNt
   u(i,j,1) = u(i-1,j,1) * COS(pi / 6.0D0)
   u(i,j-1,2) = u(i-1,j,1) * SIN(pi / 6.0D0)
END DO
u(i,TNb,1) = u(i-1,TNb,1)

END DO
```

```
!  ============ Calculate tracer concentrations =====================

!  ...  Conservative flux form
cp = cp * (D + sp)

!  ...  Fluxes through left cell interface
DO i = i1+1,im
DO j = j1,jm
   UU = half * (u(i-1,j,1) + up(i-1,j,1))
   RRR = cp(i,j) - cp(i-1,j)
   IF (UU >= zero) THEN
      II = i-1
      IF (ABS(RRR) >= tol) THEN
         r = (cp(i-1,j) - cp(i-2,j)) / RRR
      ELSE
         r = zero
      END IF
   ELSE
      II = i
      IF (ABS(RRR) >= tol) THEN
         r = (cp(i+1,j) - cp(i,j)) / RRR
      ELSE
         r = zero
      END IF
   END IF
   scflux(i-1,j,1) = scflux(i-1,j,1) + UU * cp(II,j)
   CALL limiters (ilim,r,Cr)
   SS = ABS(UU)/two * (one - dt/dx * ABS(UU)) * Cr * RRR
   scflux(i-1,j,1) = scflux(i-1,j,1) + SS
   IF (UU >= zero) THEN
      II = i
   ELSE
      II = i-1
   END IF
   VV = half * (u(i,j,2) + up(i,j,2))
   IF (VV >= zero) THEN
      scflux(II,j,2) = scflux(II,j,2) - dt/(two*dy) * RRR*UU*VV
      scflux(i,j,2) = scflux(i,j,2) + dt/dy * VV*SS
      scflux(i-1,j,2) = scflux(i-1,j,2) - dt/dy *VV*SS
   END IF
   VV = half * (u(i,j-1,2) + up(i,j-1,2))
   IF (VV < zero) THEN
      scflux(II,j-1,2) = scflux(II,j-1,2) - dt/(two*dy) * RRR*UU*VV
      scflux(i,j-1,2) = scflux(i,j-1,2) + dt/dy * VV*SS
      scflux(i-1,j-1,2) = scflux(i-1,j-1,2) - dt/dy * VV*SS
   END IF
END DO
END DO

!  ...  Fluxes through bottom cell interface
DO j = j1+1,jm
DO i = i1,im
   VV = half * (u(i,j-1,2) + up(i,j-1,2))
   RRR = cp(i,j) - cp(i,j-1)
   IF (VV >= zero) THEN
      JJ = j-1
      IF (ABS(RRR) >= tol) THEN
         r = (cp(i,j-1) - cp(i,j-2)) / RRR
      ELSE
```

```
         r = zero
      END IF
   ELSE
      JJ = j IF (ABS(RRR) >= tol) THEN
         r = (cp(i,j+1) - cp(i,j)) / RRR
      ELSE
         r = zero
      END IF
   END IF
   scflux(i,j-1,2) = scflux(i,j-1,2) + VV * cp(i,JJ)
   CALL limiters (ilim,r,Cr)
   SS = ABS(VV)/two * (one - dt/dy * ABS(VV)) * Cr * RRR
   scflux(i,j-1,2) = scflux(i,j-1,2) + SS
   IF (VV >= one) THEN
      JJ = j
   ELSE
      JJ = j-1
   END IF
   UU = half * (u(i,j,1) + up(i,j,1))
   IF (UU >= zero) THEN
      scflux(i,JJ,1) = scflux(i,JJ,1) - dt/(two*dx) * RRR*VV*UU
      scflux(i,j,1) = scflux(i,j,1) + dt/dx * UU*SS
      scflux(i,j-1,1) = scflux(i,j-1,1) - dt/dx * UU*SS
   END IF
   UU = half * (u(i-1,j,1) + up(i-1,j,1))
   IF (UU < zero) THEN
      scflux(i-1,JJ,1) = scflux(i-1,JJ,1) - dt/(two*dx) * RRR*VV*UU
      scflux(i-1,j,1) = scflux(i-1,j,1) + dt/dx * UU*SS
      scflux(i-1,j-1,1) = scflux(i-1,j-1,1) - dt/dx * UU*SS
   END IF
END DO
END DO

!  ...  Boundary conditions
j = j1
DO i = TWl,TWr
   scu1 = half * (up(i,j-1,2) + u(i,j-1,2))
   IF (scu1 < zero) THEN
      scflux(i,j-1,2) = scu1 * cp(i,j)
   ELSE
      scflux(i,j-1,2) = scu1 * minConc
   END IF
END DO
DO i = TEl,TEr
   scu1 = half * (up(i,j-1,2) + u(i,j-1,2))
   IF (scu1 < zero) THEN
      scflux(i,j-1,2) = scu1 * cp(i,j)
   ELSE
      scflux(i,j-1,2) = scu1 * minConc
   END IF
END DO
i = im
DO j = TNb,TNt
   scu1 = half * (up(i,j,1) + u(i,j,1))
   IF (scu1 > zero) THEN
      scflux(i,j,1) = scu1 * cp(i,j)
   ELSE
      scflux(i,j,1) = scu1 * minConc
   END IF
```

```
END DO
DO j = TSb,TSt
   scu1 = half * (up(i,j,1) + u(i,j,1))
   IF (scu1 > zero) THEN
      scflux(i,j,1) = scu1 * cp(i,j)
   ELSE
      scflux(i,j,1) = scu1 * minConc
   END IF
END DO

!  ...  Update tracer concentrations
DO i = i1,im
DO j = j1,jm
   c(i,j) = cp(i,j) - dt/dx * (scflux(i,j,1) - scflux(i-1,j,1)) &
      - dt/dy * (scflux(i,j,2) - scflux(i,j-1,2))
END DO
END DO

!  ...  Convert back to scalar concentration values
DO i = i1,im
DO j = j1,jm
   IF ((D(i,j)+sp(i,j)) > shtol) THEN
      cp(i,j) = cp(i,j) / (D(i,j)+sp(i,j))
   ELSE
      cp(i,j) = minConc
   END IF
END DO
END DO

!  ...  Turbulent diffusion terms
DO i = i1,im
DO j = j1,jm
   IF (domain(i,j)) THEN
      scH1 = half * (s(i+1,j) + sp(i+1,j)) + D(i+1,j)
      sc1 = scH1 > shtol
      scH2 = half * (s(i-1,j) + sp(i-1,j)) + D(i-1,j)
      sc2 = scH2 > shtol
      scH3 = half * (s(i,j+1) + sp(i,j+1)) + D(i,j+1)
      sc3 = scH3 > shtol
      scH4 = half * (s(i,j-1) + sp(i,j-1)) + D(i,j-1)
      sc4 = scH4 > shtol
      IF ((domain(i+1,j) .AND. sc1) .AND. &
         (domain(i-1,j) .AND. sc2)) THEN
         c(i,j) = c(i,j) + dt * DH / dx**two * (Hstar(i,j,1) * &
            (cp(i+1,j) - cp(i,j)) - Hstar(i-1,j,1) * &
            (cp(i,j) - cp(i-1,j)))
      ELSE IF ((.NOT. domain(i+1,j) .OR. .NOT. sc1) .AND. &
         (.NOT. domain(i-1,j) .OR. .NOT. sc2)) THEN
         c(i,j) = c(i,j)
      ELSE IF (.NOT. domain(i+1,j) .OR. .NOT. sc1) THEN
         c(i,j) = c(i,j) + dt*DH / dx**two * Hstar(i-1,j,1) * &
            (-cp(i,j) + cp(i-1,j))
      ELSE
         c(i,j) = c(i,j) + dt*DH / dx**two * Hstar(i,j,1) * &
            (cp(i+1,j) - cp(i,j))
      END IF
      IF ((domain(i,j+1) .AND. sc3) .AND. &
         (domain(i,j-1) .AND. sc4)) THEN
         c(i,j) = c(i,j) + dt * DH / dy**two * (Hstar(i,j,2) * &
```

```
            (cp(i,j+1) - cp(i,j)) - Hstar(i,j-1,2) * &
            (cp(i,j) - cp(i,j-1)))
      ELSE IF ((.NOT. domain(i,j+1) .OR. .NOT. sc3) .AND. &
         (.NOT. domain(i,j-1) .OR. .NOT. sc4)) THEN
         c(i,j) = c(i,j)
      ELSE IF (.NOT. domain(i,j+1) .OR. .NOT. sc3) THEN
         c(i,j) = c(i,j) + dt*DH / dy**two * Hstar(i,j-1,2) * &
            (-cp(i,j) + cp(i,j-1))
      ELSE
         c(i,j) = c(i,j) + dt*DH / dy**two * Hstar(i,j,2) * &
            (cp(i,j+1) - cp(i,j))
      END IF
   ELSE
      c(i,j) = minConc
   END IF
END DO
END DO

!  ...  Convert back to scalar concentration values
DO i = i1,im
DO j = j1,jm
   IF (Hnew(i,j) > shtol) THEN
      c(i,j) = c(i,j) / Hnew(i,j)
   ELSE
      !  Account for mass lost by cell going dry and set to min C
      IF (wetdry(i,j)) FPmassRm = FPmassRm + c(i,j)*dx*dy
      c(i,j) = minConc
   END IF
END DO
END DO

!  ...  Adjust matrix of wet cells
DO i = 1,im
DO j = 1,jm
   wetdry(i,j) = (Hnew(i,j) > shtol)
END DO
END DO

!  ...  Reset tracer values that go below minimum concentrations
DO i = i1,im
DO j = j1,jm
   IF (c(i,j) < minConc) c(i,j) = minConc
END DO
END DO

!  ...  Sm tracer run (values are in pptr or micrograms/m**3)
!  ...  Release tracer from 0:00-0:00 on 2/18/04
treleases = 24.0D0 * 0.0D0 + 60.0D0 + 0.0D0/sixty + FLOAT(stflow)
treleases = duration * treleases
treleasee = 24.0D0 * 0.0D0 + 60.0D0 + 0.0D0/sixty + FLOAT(stflow)
treleasee = duration * treleasee
IF (NINT(time_seconds*1.0D1) >= NINT(treleases*3.6D4) .AND. &
   NINT(time_seconds*1.0D1) <= NINT(treleasee*3.6D4)) THEN
   !oldSmt = c(im,TNb+1) * dx * dy * Hnew(im,TNb+1)
   !  ...  Add new Sm = rate * time * stoich factor * conv.  fact
   !newSmt = oldSmt + (1.0D2 / (sixty*sixty)) * dt * &
   !  (150.36D0 / 327.48D0) * 1.0D6
   !c(im,TNb+1) = newSmt / (dx * dy * Hnew(im,TNb+1))
   c(im,TNb+2) = Cmass / (dx*dy*Hnew(im,TNb+2))
```

```
   END IF
   !  ...  Print time to file when x% of tracer has left the FP
   IF (time_seconds*1.0D1 > treleasee*3.6D4) THEN
      FPmass = zero
      DO i = i1,im
      DO j = j1,jm
         IF (domain(i,j)) FPmass = FPmass + c(i,j)*dx*dy*Hnew(i,j)
      END DO
      END DO
      FPmass = FPmass + FPmassRm
      DO m = 1,19
      IF (cswitch(m) == 0) THEN
      IF (FPmass / Cmass < cswitchvals(m)) THEN
         int1 = FLOOR(time_seconds/(sixty*sixty))
         int2 = FLOOR(time_seconds/sixty-FLOAT(int1)*sixty)
         int3 = NINT(time_seconds-int1*60*60-int2*60)
         WRITE(ifout9,*) (one - cswitchvals(m))*100.0D0, &
            '% of tracer gone', int1, ' hours,', int2, ' minutes,', &
            int3, ' seconds'
         cswitch(m) = 1
      END IF
      END IF
      END DO
   END IF

   !  ...  Deallocate variables
   DEALLOCATE (exs, exu, Hnew, gamma, sx, sy, rr, scflux)

END SUBROUTINE solve_fd

END MODULE SIFT2D_fd
```

C.4 SIFT2D_friction.f90

```
!**********************************************************************
 MODULE SIFT2D_friction
!**********************************************************************

    USE SIFT2D_types
    IMPLICIT NONE
    SAVE

 CONTAINS

!**********************************************************************
 SUBROUTINE bottom_friction (iin, jin, ManNout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  August 2006
!  Purpose:  Governing program for calculation of bottom roughness values
!  based on vegetation heights, water depths, and velocities
!----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  iin, jin
    REAL(KIND = 8), INTENT(OUT) ::  ManNout

    !  ...  Friction factor calculation for cell left of interface
    !  ...  Vegetation heights of 999 meters denote the channel
    IF (ManN(iin,jin) == zero) THEN
       IF (vegH(iin,jin) == 999.0D0) THEN
          ManN(iin,jin) = ManNchan * (half * (H(iin,jin) + &
             H(iin+a1,jin+b1)))**(one/6.0D0)
       ELSE IF (vegH(iin,jin) < 1.2D0) THEN
          CALL shortveg (iin,jin, iin,jin, iin+a1,jin+b1, ManN(iin,jin))
       ELSE
          CALL tallveg (iin, jin, iin+a1, jin+b1, ManN(iin,jin))
       END IF
    END IF

    !  ...  Friction factor calculation for cell right of interface
    IF (ManN(iin+a1,jin+b1) == zero) THEN
       IF (vegH(iin+a1,jin+b1) == 999.0D0) THEN
          ManN(iin+a1,jin+b1) = ManNchan * (half * (H(iin,jin) + &
             H(iin+a1,jin+b1)))**(one/6.0D0)
       ELSE IF (vegH(iin+a1,jin+b1) < 1.2D0) THEN
          CALL shortveg (iin,jin, iin+a1,jin+b1, iin,jin, &
             ManN(iin+a1,jin+b1))
       ELSE
          CALL tallveg (iin+a1, jin+b1, iin, jin, ManN(iin+a1,jin+b1))
       END IF
    END IF

    !  ...  Friction factor for interface is average of adjacent cells
    !ManNout = half * (ManN(iin,jin) + ManN(iin+a1,jin+b1))
    !  ...  Friction factor calculation by upwinding
    IF (up(iin,jin,m) >= zero) THEN
       ManNout = ManN(iin,jin)
    ELSE
       ManNout = ManN(iin+a1,jin+b1)
    END IF
```

```
 END SUBROUTINE bottom_friction


!**********************************************************************
 SUBROUTINE shortveg (iin1, jin1, iin2, jin2, iin3, jin3, ManNsvegout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  August 2006
!  Purpose:  Calculate Manning's n for short vegetation (< 1.2m)
!----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  iin1, jin1, iin2, jin2, iin3, jin3
    REAL(KIND = 8), INTENT(OUT) ::  ManNsvegout
    REAL(KIND = 8) ::  fDsvegout, Havg1
    REAL(KIND = 8) ::  slope, tau, udbar2, vel2, ust, MEI
    REAL(KIND = 8) ::  vegHdefl, ustcrit, ratio1, A2, B2, int1

    !  ...  For initial time step, estimate tau
    Havg1 = half * (H(iin2,jin2) + H(iin3,jin3))
    IF (time_seconds == dt) THEN
       slope = ABS((D(iin1+a1,jin1+b1) - D(iin1,jin1)) / dx)
       tau = g * H(iin2,jin2) * slope
    ELSE
       udbar2 = quart * (up(iin2-b1,jin2-a1,mop) + &
          up(iin2+f1,jin2+g1,mop) + up(iin2,jin2,mop) + &
          up(iin2+a1,jin2+b1,mop))
       vel2 = up(iin2,jin2,m)**two + udbar2**two
       tau = g * ManNp(iin2,jin2)**two * vel2 / (Km**two * Havg1 &
          **(one/3.0D0))
    END IF
    ust = SQRT(tau)

    !  ...  Calculate stem rigidity in bending
    !MEI = 319.0D0 * vegH(iin2,jin2)**3.3D0 !  for growing grasses
    MEI = 25.4D0 * vegH(iin2,jin2)**2.26D0 !  for dormant grasses

    !  ...  Calculate deflected vegetation height
    vegHdefl = 0.14D0 * vegH(iin2,jin2) * ((MEI/(tau*rho0))**quart &
       / vegH(iin2,jin2))**1.59D0
    IF (vegHdefl > vegH(iin2,jin2)) vegHdefl = vegH(iin2,jin2)
    IF (vegHdefl > Havg1) vegHdefl = Havg1

    !  ...  Choose coefficients for log relationship
    ustcrit = MIN(0.028D0 + 6.33D0 * MEI, 0.23D0 * MEI**0.106D0)
    ratio1 = ust / ustcrit
    IF (ratio1 <= 1.0D0) THEN
       A2 = 0.15D0; B2 = 1.85D0
    ELSE IF (ratio1 <= 1.5D0) THEN
       A2 = 0.20D0; B2 = 2.70D0
    ELSE IF (ratio1 <= 2.5D0) THEN
       A2 = 0.28D0; B2 = 3.08D0
    ELSE
       A2 = 0.29D0; B2 = 3.50D0
    END IF
    int1 = A2 + B2 * LOG10(Havg1 / vegHdefl)
    fDsvegout = one / (int1**two)

    !  ...  Convert to Manning's n
```

```
    ManNsvegout = SQRT(fDsvegout * Km**two / (8.0D0 * g))
    ManNsvegout = ManNsvegout * Havg1**(one/6.0D0)

    !  ...  Cut values by % value dictated by parameter svegcut
    ManNsvegout = svegcut * ManNsvegout

    !  ...  Cap Manning's n values at value dictated by parameter svegcap
    IF (ManNsvegout > svegcap) ManNsvegout = svegcap

 END SUBROUTINE shortveg


!**********************************************************************
 SUBROUTINE tallveg (iin2, jin2, iin3, jin3, ManNtvegout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  August 2006
!  Purpose:  Calculate Manning's n for tall vegetation
!----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  iin2, jin2, iin3, jin3
    REAL(KIND = 8), INTENT(OUT) ::  ManNtvegout
    REAL(KIND = 8) ::  xiE, udbar2, vel2, fDtvegout, Havg1

    xiE = 2.99D0 !  All values assumed equivalent to white pine
    udbar2 = 0.25D0 * (up(iin2-b1,jin2-a1,mop) + up(iin2+f1,jin2+g1,mop) &
       + up(iin2,jin2,mop) + up(iin2+a1,jin2+b1,mop))
    vel2 = SQRT(up(iin2,jin2,m)**two + udbar2**two)
    Havg1 = half * (H(iin2,jin2) + H(iin3,jin3))
    fDtvegout = 4.06D0 * (vel2 / SQRT(xiE/rho0))**(-0.46D0) * Havg1 / &
       vegH(iin2,jin2)

    !  ...  Convert to Manning's n
    ManNtvegout = SQRT(fDtvegout * Km**two / (8.0D0 * g))
    ManNtvegout = ManNtvegout * Havg1**(one/6.0D0)

    !  ...  Cut values by % value dictated by parameter tvegcut
    ManNtvegout = tvegcut * ManNtvegout

    !  ...  Cap Manning's n values at value dictated by parameter tvegcap
    IF (ManNtvegout > tvegcap) ManNtvegout = tvegcap

 END SUBROUTINE tallveg


!**********************************************************************
 SUBROUTINE hydrology (iin, jin, hydro_out)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  December 2006
!  Purpose:  Calculate hydrologic sources/sinks for continuity eqn
!----------------------------------------------------------------------

    INTEGER, INTENT(IN) ::  iin, jin
    REAL(KIND = 8), INTENT(OUT) ::  hydro_out
    REAL(KIND = 8) ::  esata, ea, esatw, Evap, ET, vfrac
    REAL(KIND = 8), PARAMETER ::  CEv = 1.15D-3 !  Evaporation coef
    REAL(KIND = 8), PARAMETER ::  CTr = 1.0D0 !  Transpiration coef
    REAL(KIND = 8), PARAMETER ::  Twater = 11.0D0 !  Average water temp
```

```
   INTEGER ::  hrnum

   !  ...  Find hour number
   hrnum = CEILING(time_seconds/(sixty*sixty))
   IF (stflow > 0) THEN
      hrnum = hrnum - stflow
      IF (hrnum < 1) hrnum = 1
   END IF

   hydro_out = zero
   !  ...  Precipitation contribution
   hydro_out = Precip(hrnum) /1.0D3 /(sixty*sixty) * dt

   !  ...  Groundwater contribution
   !  ...  Seepage for cells that are already wet
   IF (wetdry(iin,jin)) THEN
      hydro_out = hydro_out - Seep_rt /1.0D3 /(sixty*sixty*24.0D0) * dt
   ELSE ! Assume infiltration capacity = 1.5 cm/hr
      IF (Precip(hrnum) < 15.0D0) THEN
         hydro_out = zero
      ELSE
         hydro_out = (Precip(hrnum)-15.0D0) /1.0D3 /(sixty*sixty) * dt
      END IF
   END IF

   !  ...  Evapotranspiration contribution
   !  ...  Use ET calculated by CIMIS or other met site, if available
   IF (EvapoTransp(hrnum) > -1.0D1) THEN
      hydro_out = hydro_out - EvapoTransp(hrnum)/1.0D3/(sixty*sixty)*dt
   ELSE ! Calculate using aerodynamic method
      esata = EXP(2.3026D0 * ((7.5D0*AirTemp(hrnum))/(AirTemp(hrnum) &
         + 237.3D0)) + 0.7858D0)
      esatw = EXP(2.3026D0*((7.5D0*Twater)/(Twater+237.3D0))+0.7858D0)
      ea = esata * (RelHum(hrnum)/1.0D2)
      Evap = 0.622D0 * CEv * Windsp(hrnum) * (rhoair / rho0) * &
         ((esatw - ea) / Patm)
      vfrac = 1.0D0 - (H(iin,jin) / vegH(iin,jin))
      ET = (Evap + CTr * Evap * DMAX1(zero, vfrac)) * dt
      hydro_out = hydro_out - ET
   END IF

END SUBROUTINE hydrology

END MODULE SIFT2D_friction
```

C.5 SIFT2D_initial.f90

```
!***********************************************************************
 MODULE SIFT2D_initial
!***********************************************************************

    USE SIFT2D_types
    IMPLICIT NONE
    SAVE

 CONTAINS

!***********************************************************************
 SUBROUTINE input
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Define input parameters
!-----------------------------------------------------------------------

    !  ...  Spatial dimensions of the problem
    i1 = 2 !  Index of first wet x cell in domain
    im = 98 !  Index of last wet x cell in domain
    im1 = im+1 !  Index of last x cell in domain
    j1 = 2 !  Index of first wet y cell in domain
    jm = 87 !  Index of last wet y cell in domain
    jm1 = jm+1 !  Index of last y cell in domain
    TNb = 75; TNt = 80 !  Triangle north boundaries
    TSb = 42; TSt = 43 !  Triangle south boundaries
    TWl = 50; TWr = 51 !  Triangle west boundaries
    TEl = 83; TEr = 85 !  Triangle east boundaries
    TNd = 5.25D0; TSd = 6.17D0 !  Inlet pressure sensor depths
    TEd = 5.86D0; TWd = 5.81D0 !  Outlet pressure sensor depths
    dx = 10.0D0
    dy = 10.0D0

    !  ...  Total number of cells in the rectangular domain
    numcells = (im-i1+1) * (jm-j1+1)

    !  ...  Advection discretization choice
    !  ...  1 = moment cons, 2 = energy cons, 3 = dynamic choice
    !  ...  4 = no advection
    iadv = 3
    !  ...  Dynamic choice parameter
    eps = 1.0D-2
    !  ...  Limiter choice
    !  ...  1 = van Leer, 2 = MC, 3 = minmod, 4 = superbee, 5 = 2nd
    !  ...  order accurate without use of limiters
    ilim = 2
    !  ...  Limiter tolerance - 1.0E-5 = apply limiters with reckless
    !  ...  abandon, 1.0E5 = don't apply limiters
    tol = 1.0D-5
    !  ...  Shallow water tolerance
    shtol = 1.0D-3 !  Below this value cells declared dry

    !  ...  Triangle floodplain flood duration simulations
    duration = 1.0D0

    !  ...  Time step and no.  of steps in simulation
```

```
    dt = 0.50D0
    Tend = duration * 6.5D0 * 24.0D0 * sixty * sixty !  6.5 days
    stflow = 0 !  Hours of steady flow
    Tend = Tend + FLOAT(stflow) * sixty * sixty !  Steady flow start
    nsteps = NINT(Tend/dt)

    !  ...  Number of iterations for friction term
    maxiter = 2

    !  ...  Output types (1 = create file)
    iout1 = 1 !  Water surface map
    iout2 = 1 !  Water surface time series
    iout3 = 0 !  u time series
    iout4 = 0 !  v time series
    iout5 = 1 !  Velocity map
    iout6 = 1 !  Flow map
    iout7 = 1 !  Scalar concentration map
    iout8 = 0 !  Manning's n map
    iout9 = 1 !  Run parameters
    iout10 = 0 !  Outlet scalar concentration time series
    iout11 = 1 !  Total tracer mass remaining in the FP time series

    !  ...  Output parameters - number of node for which time series
    !  ...  of zeta, u, and v will be output
    xnode_no = 68
    ynode_no = 29

 END SUBROUTINE input


!**********************************************************************
 SUBROUTINE allocate_space
!**********************************************************************
!  Author :  Francisco Rueda
!  Date :  April 2005
!  Purpose:  Dimensionalize allocatable arrays
!----------------------------------------------------------------------

    !  ...  Local variables
    INTEGER ::  istat

    ALLOCATE (u(1:im1,1:jm1,2),up(1:im1,1:jm1,2), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 1)

    ALLOCATE (s(1:im1,1:jm1),sp(1:im1,1:jm1), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 2)

    ALLOCATE (c(1:im1,1:jm1),cp(1:im1,1:jm1), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 3)

    ALLOCATE (D(1:im1,1:jm1),vegH(1:im1,1:jm1), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 4)

    ALLOCATE (ManN(1:im1,1:jm1), ManNp(1:im1,1:jm1), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 5)

    ALLOCATE (domain(1:im1,1:jm1), wetdry(1:im1,1:jm1), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 6)

    ALLOCATE (H(1:im1,1:jm1), Hstar(1:im1,1:jm1,2), STAT = istat)
```

```
    IF (istat /= 0) CALL allocate_error (istat, 7)

    ALLOCATE (zeta(1:numcells),ds(1:numcells), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 8)

    ALLOCATE (aa(1:numcells,1:maxnz), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 9)

 END SUBROUTINE allocate_space

!**********************************************************************
 SUBROUTINE allocate_error (istat, ierror_code)
!**********************************************************************
!  Author :  Francisco Rueda
!  Date :  April 2005
!  Purpose:  Print a flag on the screen and stop when allocation error
!  is detected
!----------------------------------------------------------------------

    !  ...  Arguments
    INTEGER, INTENT(IN) ::  istat, ierror_code
    !  ...  Print error messages
    WRITE(6,*) "Error allocating space"
    WRITE(6,*) '("istat =, I4, " error code ", = I3)', istat, &
       ierror_code
    STOP

 END SUBROUTINE allocate_error

!**********************************************************************
 SUBROUTINE flow_input (TSinout, TNinout, TEinout, TWinout)
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  August 2006
!  Purpose:  Read in text file of input hydrographs
!----------------------------------------------------------------------

    INTEGER ::  i, iocode
    REAL(KIND = 8), INTENT(OUT) ::  TNinout, TSinout, TEinout, TWinout

    IF (nn == 1) THEN
       OPEN(UNIT = ifin3, FILE = 'wave1_interp.txt')
       READ(UNIT = ifin3, FMT = *, IOSTAT = iocode) TSst,TNst,TEst,TWst
       IF (iocode /= 0) PAUSE 'Error reading hydrograph'
       TSinout = TSst; TNinout = TNst
       TEinout = TEst; TWinout = TWst
    END IF

    IF (NINT(time_seconds*1.0D1) <= stflow*3.6D4) THEN
       TSinout = TSst; TNinout = TNst
       TEinout = TEst; TWinout = TWst
    ELSE
    READ(UNIT = ifin3, FMT = *, IOSTAT = iocode) TSinout, TNinout, &
       TEinout, TWinout
    IF (iocode /= 0) PAUSE 'Error reading hydrograph'
    END IF

    IF (nn == nsteps) CLOSE(UNIT = ifin3)

 END SUBROUTINE flow_input
```

```
!**********************************************************************
 SUBROUTINE met_input
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  August 2006
!  Purpose:  Read in text file of meteorological variables
!----------------------------------------------------------------------

    INTEGER ::  npts, istat, i, iocode

    !  ...  Allocate space for met variables
    npts = NINT(Tend/(sixty*sixty))+1
    ALLOCATE (Precip(1:npts) , STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 21)
    ALLOCATE (AirTemp(1:npts) , STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 22)
    ALLOCATE (RelHum(1:npts) , STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 23)
    ALLOCATE (Windsp(1:npts) , STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 24)
    ALLOCATE (Winddir(1:npts) , STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 25)
    ALLOCATE (EvapoTransp(1:npts), STAT = istat)
    IF (istat /= 0) CALL allocate_error (istat, 26)

    !  ...  Read in hourly met data
    !  ...  Precipitation in mm of rain per hour
    !  ...  Air Temperature in degrees C
    !  ...  Relative humidity in %
    !  ...  Wind speed in m/s
    !  ...  Wind direction in degrees
    !  ...  Evapotranspiration in mm per hour - if not available, value
    !  ...  set to -9999
    OPEN(UNIT = ifin4, FILE = 'met_data.txt')
    DO i = 1,npts
       READ(UNIT = ifin4, FMT = *, IOSTAT = iocode) Precip(i), &
          AirTemp(i), RelHum(i), Windsp(i), Winddir(i), EvapoTransp(i)
       IF (iocode /= 0) PAUSE 'Error reading met data'
    END DO

    CLOSE(UNIT = ifin4)

 END SUBROUTINE met_input


!**********************************************************************
 SUBROUTINE init
!**********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Set initial conditions
!----------------------------------------------------------------------

    CHARACTER ::  input_file*19
    INTEGER ::  i, j, numj, ios
    REAL(KIND = 8) ::  latitude, omega_earth !  for Coriolis freq
    REAL(KIND = 8) ::  aS, int1 !  for channel friction
```

```
!  ...  Define domain boundaries with logical variable
!  ...  if domain = .TRUE. then in the domain
DO i = 1,im1
DO j = 1,jm1
   !  ...  Cut off domain at setback levee
   domain(i,j) = (i >= i1 .AND. i <= im) .AND. (j >= j1 &
      .AND. j <= jm)
   IF (i >= 2 .AND. i <=11) domain(i,j) = domain(i,j) .AND. j <= i
   IF (i >=12 .AND. i <=21) domain(i,j) = domain(i,j) .AND. j <= i-1
   IF (i >=22 .AND. i <=30) domain(i,j) = domain(i,j) .AND. j <= i-2
   IF (i >=31 .AND. i <=40) domain(i,j) = domain(i,j) .AND. j <= i-3
   IF (i >=41 .AND. i <=51) domain(i,j) = domain(i,j) .AND. j <= i-4
   IF (i >=52 .AND. i <=60) domain(i,j) = domain(i,j) .AND. j <= i-5
   IF (i >=61 .AND. i <=69) domain(i,j) = domain(i,j) .AND. j <= i-6
   IF (i >=70 .AND. i <=79) domain(i,j) = domain(i,j) .AND. j <= i-7
   IF (i >=80 .AND. i <=88) domain(i,j) = domain(i,j) .AND. j <= i-8
   IF (i >=89 .AND. i <=96) domain(i,j) = domain(i,j) .AND. j <= i-9
END DO
END DO

!  ...  Location of diagonal rows for use with NSPCG
numj = jm - j1 + 1
jcoef = (/ 0, 1, numj /)

!  ...  Horizontal eddy viscosity in (m**2/s) and diffusivity
KH = 0.1D0 !  0.1 m**2/s = average for floodplain
DH = 0.1D0

!  ...  Coriolis frequency calculation
latitude = 38.0D0 + 16.0D0/sixty !  in degrees (approx triangle fp)
omega_earth = 7.3D-5 !  in s**(-1)
fc = two * omega_earth * SIN(latitude/180.0D0 * pi)

!  ...  Wind speed and direction
!W10 = zero
W10 = 2.3D0 !  in m/s
!W10 = 80.0D0 !  Category 5 hurricane winds
theta = 45.0D0 !  Direction of wind vector
!  in degrees measured (+) cw from pos y axis
wind1 = Cw * (rhoair / rho0) * W10**two !  for use in momentum eqn

!  ...  Vegetation heights and channel friction factor
!  ...  in meters (999.0 m denotes channel)
!  ...  Open file and read vegetation heights
vegH = 0.3D0
input_file = "tri_vegetate.txt"
OPEN (UNIT = ifin, FILE = input_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening input file 1 = ",ios; STOP
END IF
DO j = j1,jm
   READ(UNIT = ifin, FMT = *) (vegH(i,j), i=i1,im)
END DO
CLOSE (UNIT = ifin)
!  ...  Values for cells in the channel
aS = 11.1D0 * (Rh / Dmax)**(-0.314D0)
int1 = 2.03D0 * LOG10(aS * Rh / (3.5D0 * d84))
ManNchan = one / (int1**two)
```

```
   ManNchan = SQRT(ManNchan) * SQRT(Km**two / (8.0D0 * g))
   !  ...  Fractions of routine calculated ManN values to use
   svegcut = 0.25D0; tvegcut = 1.0D0
   svegcap = 0.05D0; tvegcap = 0.12D0
   ManN = zero; ManNp = ManN

   !  ...  Open file and read depth values
   D = zero
   input_file = "tri_depths.txt"
   OPEN (UNIT = ifin2, FILE = input_file, IOSTAT = ios)
   IF (ios /= 0) THEN
      WRITE(6,*) "Error opening input file 2 = ",ios; STOP
   END IF
   DO j = j1,jm
      READ(UNIT = ifin2, FMT = *) (D(i,j), i=i1,im)
   END DO
   CLOSE (UNIT = ifin2)
   DO j = 1,jm1
      D(1,j) = D(i1,j)
      D(im1,j) = D(im,j)
   END DO
   DO i = 1,im1
      D(i,1) = D(i,j1)
      D(i,jm1) = D(i,jm)
   END DO

   !  ...  Initial conditions for water surface elevation (pond full)
   initial_s = 5.445D0 !  Depth measured positive down from datum
   DO j = 1,jm1
   DO i = 1,im1
      IF (D(i,j) >= initial_s .AND. domain(i,j)) THEN
         sp(i,j) = -initial_s
         wetdry(i,j) = .TRUE. !  True if cell is wet
      ELSE
         sp(i,j) = -D(i,j)
         wetdry(i,j) = .FALSE.
      END IF
   END DO
   END DO
   s = sp

   !  ...  Initial conditions for velocities
   up = zero
   u = up

   !  ...  Initial conditions for scalar concentrations
   minConc = 1.0D-6
   !Cmass = 1.0D2 * (150.36D0/327.48D0) * 1.0D6 !  initial Sm in ug
   Cmass = 1.0D6
   FPmassRm = zero !  Total mass lost by cells going dry
   !  ...  Fractional values for tracer left in floodplain for scalars
   cswitchvals = (/ (FLOAT(i)*0.05D0, i = 1,19) /)
   cswitch = (/ (0, i = 1,19) /)
   cp = minConc
   c = cp

END SUBROUTINE init

END MODULE SIFT2D_initial
```

C.6 SIFT2D_types.f90

```
!**********************************************************************
 MODULE SIFT2D_types
!**********************************************************************

    IMPLICIT NONE
    SAVE

    !  ...  I/O file numbers and node numbers
    INTEGER, PARAMETER ::  ifout = 101, ifout2 = 102, ifout3 = 103
    INTEGER, PARAMETER ::  ifout4 = 104, ifout5 = 105, ifout6 = 106
    INTEGER, PARAMETER ::  ifout7 = 107, ifout8 = 108, ifout9 = 109
    INTEGER, PARAMETER ::  ifout10 = 110, ifout11 = 111
    INTEGER, PARAMETER ::  ifin = 201, ifin2 = 202, ifin3 = 203
    INTEGER, PARAMETER ::  ifin4 = 204
    INTEGER ::  xnode_no, ynode_no
    INTEGER ::  iout1, iout2, iout3, iout4, iout5, iout6
    INTEGER ::  iout7, iout8, iout9, iout10, iout11

    !  ...  State variable arrays
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:,:)  ::  u, up !  Flow vel
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:  )  ::  s, sp !  Sfc elev
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:  )  ::  c, cp !  Scalar

    !  ...  Depth, total water height, and roughness arrays
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:  )  ::  D, vegH
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:  )  ::  ManN, ManNp
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:  )  ::  H
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:,:)  ::  Hstar

    !  ...  Solution scheme arrays
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:  )  ::  zeta
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:)  ::  aa
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:  )  ::  ds

    !  ...  Geophysical constants
    REAL(KIND = 8), PARAMETER ::  g = 9.8061600D0
    REAL(KIND = 8), PARAMETER ::  pi = 3.1415926D0
    REAL(KIND = 8), PARAMETER ::  rhoair = 1.2D0 !  In kg/m**3
    REAL(KIND = 8), PARAMETER ::  rho0 = 998.2D0 !  In kg/m**3
    REAL(KIND = 8), PARAMETER ::  Km = 1.0D0 !  Manning const
    REAL(KIND = 8), PARAMETER ::  Cw = 1.5D-3 !  Water sfc drag

    !  ...  Commonly used numbers in double precision
    REAL(KIND = 8), PARAMETER ::  zero = 0.0D0, half = 0.5D0
    REAL(KIND = 8), PARAMETER ::  two = 2.0D0, sixty = 60.0D0
    REAL(KIND = 8), PARAMETER ::  one = 1.0D0, quart = 0.25D0

    !  ...  Channel and floodplain parameters
    LOGICAL, ALLOCATABLE, DIMENSION(:,:)  ::  domain
    LOGICAL, ALLOCATABLE, DIMENSION(:,:)  ::  wetdry
    REAL(KIND = 8), PARAMETER ::  d84 = 0.01D0 !  d84 grain size
    REAL(KIND = 8), PARAMETER ::  Rh = 0.71D0 !  Estimated hyd rad
    REAL(KIND = 8), PARAMETER ::  Dmax = 2.0D0 !  Max depth flow
    REAL(KIND = 8) ::  ManNchan !  Channel friction
    REAL(KIND = 8) ::  TNd, TSd, TWd, TEd !  Sensor depths
    REAL(KIND = 8) ::  KH ! Eddy viscosity
```

```
    REAL(KIND = 8) ::  DH ! Eddy diffusivity
    REAL(KIND = 8) ::  fc !  Coriolis frequency
    REAL(KIND = 8) ::  W10, theta !  Wind speed and direction
    REAL(KIND = 8) ::  wind1 !  For constant wind speed
    REAL(KIND = 8) ::  initial_s !  Initial water surface elevation
    REAL(KIND = 8) ::  TNst, TSst, TEst, TWst !  Steady flow vals
    REAL(KIND = 8) ::  duration !  For varying flood duration runs

    !  ...  Meteorological and hydrological parameters
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:)  ::  Precip
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:)  ::  AirTemp
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:)  ::  RelHum
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:)  ::  Windsp
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:)  ::  Winddir
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:)  ::  EvapoTransp
    REAL(KIND = 8), PARAMETER ::  Seep_rt = 0.5D0 !  GW seepage rt, mm/d
    REAL(KIND = 8), PARAMETER ::  Patm = 1021.0D0 !  Atm pressure (mb)

    !  ...  Discretization parameters
    INTEGER ::  i1, im, im1 !  Boundary cells x direction
    INTEGER ::  j1, jm, jm1 !  Boundary cells y direction
    INTEGER ::  numcells !  Total number of cells
    INTEGER ::  iadv, ilim !  Advection and limiter choices
    INTEGER ::  maxiter !  Number of iterations for friction
    REAL(KIND = 8) ::  eps, tol !  Advection and limiter parameters
    REAL(KIND = 8) ::  shtol !  For propagation over dry areas
    REAL(KIND = 8) ::  minConc !  Background scalar concentration
    REAL(KIND = 8) ::  Cmass !  Initial tracer mass input
    REAL(KIND = 8) ::  FPmassRm !  Mass lost by cells going dry
    REAL(KIND = 8) ::  svegcut, tvegcut !  % of calc ManN values to use
    REAL(KIND = 8) ::  svegcap, tvegcap !  Upper cap on calculated ManN
    REAL(KIND = 8) ::  dx, dy !  Spatial grid size
    INTEGER ::  a1, b1, c1, d1 !  Coefficients for indices
    INTEGER ::  f1, g1, mop !  Coefficients for indices
    INTEGER ::  p1, q1, v1, w1 !  Coefficients for indices
    INTEGER ::  m !  1 for x direction, 2 for y direction
    REAL(KIND = 8) ::  dxy, dxyop !  Constants for index routines
    REAL(KIND = 8) ::  dt2dxy, gdt2dxy !  Constants for index routines
    INTEGER ::  TNt,TNb,TSt,TSb !  Indices for TN,TS boundaries
    INTEGER ::  TWl,TWr,TEl,TEr !  Indices for TW,TE boundaries
    INTEGER ::  stflow !  Hours of steady flow to start run
    INTEGER, DIMENSION(19) ::  cswitch !  For tracer result output
    REAL(KIND = 8), DIMENSION(19) ::  cswitchvals

    !  ...  NSPCG parameters
    INTEGER, PARAMETER ::  nw = 300000, inw = 100000, maxnz = 3
    INTEGER, DIMENSION(:)  ::  jp(1), ip(1)
    INTEGER ::  ier
    INTEGER, DIMENSION(:)  ::  iparm(25), jcoef(maxnz), iwksp(inw)
    REAL(KIND = 8), DIMENSION(:)  ::  rparm(16), wksp(nw), ubarp(1)

    !  ...  Time stepping information
    INTEGER ::  nn, nsteps
    REAL(KIND = 8) ::  dt, time_seconds, Tend

END MODULE SIFT2D_types
```

C.7 SIFT2D_update.f90

```
!***********************************************************************
 MODULE SIFT2D_update
!***********************************************************************

    USE SIFT2D_types
    USE SIFT2D_advection
    IMPLICIT NONE
    SAVE

 CONTAINS

!***********************************************************************
 SUBROUTINE update
!***********************************************************************

    sp = s; up = u
    cp = c; ManNp = ManN

 END SUBROUTINE update


!***********************************************************************
 SUBROUTINE output
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  September 2005 (Revised May 2006)
!  Purpose:  Output time series of zeta and u for a particular node
!  and output all zetas at specified intervals
!----------------------------------------------------------------------

    !  ...  Local variables
    CHARACTER ::  output_file*25, fmt*12, date1*8, time1*10
    INTEGER ::  i, j, ios
    REAL(KIND = 8) ::  interval, Caverage, FPmass
    REAL(KIND = 8), ALLOCATABLE, DIMENSION(:,:,:)  ::  uquiv

    !  ...  Open output files on first entry
    IF (nn == 0) THEN
       !  ...  Water surface elevations everywhere throughout FP
       IF (iout1 == 1) THEN
       output_file = "s_intervals.txt"
       OPEN (UNIT = ifout, FILE = output_file, IOSTAT = ios)
       IF (ios /= 0) THEN
          WRITE(6,*) "Error opening solution file 1 = ",ios; STOP
       END IF
       WRITE (UNIT=fmt, FMT='("(",I3,"E15.6)")') im-i1+1
       DO j = j1,jm
          WRITE (UNIT=ifout, FMT = fmt) (-D(i,j), i=i1,im)
       END DO
       END IF
       !  ...  Timeseries of water surface elevations at a given cell
       IF (iout2 == 1) THEN
       output_file = "s_timeseries.txt"
       OPEN (UNIT = ifout2, FILE = output_file, IOSTAT = ios)
       IF (ios /= 0) THEN
          WRITE(6,*) "Error opening solution file 2 = ",ios; STOP
       END IF
```

```
END IF
!  ...  Timeseries of u velocity at a given cell
IF (iout3 == 1) THEN
output_file = "u_timeseries.txt"
OPEN (UNIT = ifout3, FILE = output_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening solution file 3 = ",ios; STOP
END IF
END IF
!  ...  Timeseries of v velocity at a given cell
IF (iout4 == 1) THEN
output_file = "v_timeseries.txt"
OPEN (UNIT = ifout4, FILE = output_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening solution file 4 = ",ios; STOP
END IF
END IF
!  ...  u and v velocities everywhere throughout FP
IF (iout5 == 1) THEN
output_file = "u_intervals.txt"
OPEN (UNIT = ifout5, FILE = output_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening solution file 5 = ",ios; STOP
END IF
END IF
!  ...  x and y direction flows everywhere throughout FP
IF (iout6 == 1) THEN
output_file = "q_intervals.txt"
OPEN (UNIT = ifout6, FILE = output_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening solution file 6 = ",ios; STOP
END IF
END IF
!  ...  Scalar concentration values everywhere throughout FP
IF (iout7 == 1) THEN
output_file = "c_intervals.txt"
OPEN (UNIT = ifout7, FILE = output_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening solution file 7 = ",ios; STOP
END IF
END IF
!  ...  Manning's n values everywhere throughout the FP
IF (iout8 == 1) THEN
output_file = "ManN_intervals.txt"
OPEN (UNIT = ifout8, FILE = output_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening solution file 8 = ",ios; STOP
END IF
END IF
!  ...  Run parameters
IF (iout9 == 1) THEN
output_file = "run_parameters.txt"
OPEN (UNIT = ifout9, FILE = output_file, IOSTAT = ios)
IF (ios /= 0) THEN
   WRITE(6,*) "Error opening solution file 9 = ",ios; STOP
END IF
CALL date_and_time(date1, time1)
!  ...  Write run parameters to file
WRITE(UNIT = ifout9, FMT = *) 'Date (YYYYMMDD) = ', date1
```

```
   WRITE(UNIT = ifout9, FMT = *) 'TIME (HHMMSS.SSS) = ', time1
   WRITE(UNIT = ifout9, FMT = *) 'Delta x = ', dx, ' m'
   WRITE(UNIT = ifout9, FMT = *) 'Delta y = ', dy, ' m'
   WRITE(UNIT = ifout9, FMT = *) 'Delta t = ', dt, ' m'
   WRITE(UNIT = ifout9, FMT = *) 'T end = ', Tend/3600.0D0, ' hr'
   WRITE(UNIT = ifout9, FMT = *) 'Advection method = ', iadv
   WRITE(UNIT = ifout9, FMT = *) 'Limiter tolerence = ', tol
   WRITE(UNIT = ifout9, FMT = *) 'Limiter method = ', ilim
   WRITE(UNIT = ifout9, FMT = *) 'Shallow water tolerance = ', shtol
   WRITE(UNIT = ifout9, FMT = *) 'Number of iterations = ', maxiter
   WRITE(UNIT = ifout9, FMT = *) 'Time series x node = ', xnode_no
   WRITE(UNIT = ifout9, FMT = *) 'Time series y node = ', ynode_no
   WRITE(UNIT = ifout9, FMT = *) 'Constant eddy viscosity = ', &
      KH, ' m**2/s'
   WRITE(UNIT = ifout9, FMT = *) 'Constant eddy diffusivity = ', &
      DH, ' m**2/s'
   WRITE(UNIT = ifout9, FMT = *) 'Coriolis frequency = ', fc, ' 1/s'
   WRITE(UNIT = ifout9, FMT = *) 'Wind speed = ', W10, ' m/s'
   WRITE(UNIT = ifout9, FMT = *) 'Short veg percentage used = ', &
      svegcut*1.0D2, ' %'
   WRITE(UNIT = ifout9, FMT = *) 'Tall veg percentage used = ', &
      tvegcut*1.0D2, ' %'
   WRITE(UNIT = ifout9, FMT = *) 'Short veg ManN capped at = ', &
      svegcap
   WRITE(UNIT = ifout9, FMT = *) 'Tall veg ManN capped at = ', &
      tvegcap
   WRITE(UNIT = ifout9, FMT = *) 'Initial water surface depth = ',&
      initial_s
   WRITE(UNIT = ifout9, FMT = *) 'Background scalar conc.  = ', &
      minConc
   WRITE(UNIT = ifout9, FMT = *) 'Hours of steady flow = ', stflow
   END IF
   !  ...  Outlet scalar concentration time series
   IF (iout10 == 1) THEN
   output_file = "Coutlet_timeseries.txt"
   OPEN (UNIT = ifout10, FILE = output_file, IOSTAT = ios)
   IF (ios /= 0) THEN
      WRITE(6,*) "Error opening solution file 10 = ",ios; STOP
   END IF
   END IF
   !  ...  Total tracer mass remaining in the FP time series
   IF (iout11 == 1) THEN
   output_file = "FPmass_timeseries.txt"
   OPEN (UNIT = ifout11, FILE = output_file, IOSTAT = ios)
   IF (ios /= 0) THEN
      WRITE(6,*) "Error opening solution file 11 = ",ios; STOP
   END IF
   END IF
END IF

!  ...  Print elapsed model time to screen after each interval
!  ...  Print state variables to file after each interval
!  ...  Ten minute intervals
interval = 10.0D0 * sixty !  Ten minutes
IF (MOD(NINT(time_seconds*1000),NINT(interval*1000)) == 0) THEN
IF (NINT(time_seconds) < stflow*3600) THEN
   WRITE(6,*) 'Steady flow', NINT(time_seconds/sixty), ' minutes'
ELSE
   IF (iout2 == 1) &
```

```
   WRITE(UNIT = ifout2, FMT='(E13.6)') s(xnode_no,ynode_no)
   IF (iout3 == 1) &
   WRITE(UNIT = ifout3, FMT='(E13.6)') u(xnode_no,ynode_no,1)
   IF (iout4 == 1) &
   WRITE(UNIT = ifout4, FMT='(E13.6)') u(xnode_no,ynode_no,2)
   IF (iout10 == 1) THEN
   Caverage = c(83,2)*H(83,2) + c(84,2)*H(84,2) + &
      c(85,2)*H(85,2) + c(50,2)*H(50,2) + &
      c(51,2)*H(51,2)
   Caverage = Caverage/(H(83,2) + H(84,2) + H(85,2) + &
      H(50,2) + H(51,2))
   WRITE(UNIT = ifout10, FMT='(E13.6)') Caverage
   END IF
   IF (iout11 == 1) THEN
      FPmass = zero
      DO i = i1,im
      DO j = j1,jm
         IF (domain(i,j)) FPmass = FPmass + c(i,j)*dx*dy*H(i,j)
      END DO
      END DO
      FPmass = FPmass + FPmassRm
      WRITE(ifout11, FMT='(E13.6)') FPmass
   END IF
   WRITE(6,*) NINT(time_seconds/sixty)-stflow*60, ' minutes'
END IF
END IF

!  ...  One hour intervals
interval = sixty * sixty !  1 hour
IF (MOD(NINT(time_seconds*1000),NINT(interval*1000)) == 0) THEN
IF (NINT(time_seconds) < stflow*3600) THEN
   WRITE(6,*) 'Steady flow', NINT(time_seconds/(sixty*sixty)), &
      ' hours'
ELSE
   WRITE (UNIT=fmt, FMT='("(",I3,"E15.6)")') im-i1+1
   DO j = j1,jm
      IF (iout1 == 1) &
      WRITE (UNIT=ifout, FMT = fmt) ( s(i,j), i=i1,im)
      IF (iout7 == 1) &
      WRITE (UNIT=ifout7, FMT = fmt) ( c(i,j), i=i1,im)
      IF (iout8 == 1) &
      WRITE (UNIT=ifout8, FMT = fmt) (ManN(i,j), i=i1,im)
   END DO
   IF (iout5 == 1 .OR. iout6 == 1) THEN
   ALLOCATE (uquiv(i1:im,j1:jm,2))
   DO m = 1,2
      CALL setM(m)
   DO i = i1,im
   DO j = j1,jm
      !  ...  Calculate upwind u values
      !IF (.NOT. domain(i-a1,j-b1)) THEN
      !  CALL upwind_m2 (u(i-a1,j-b1,m), u(i-a1,j-b1,m), &
      !  u(i,j,m), u(i+a1,j+b1,m), Hstar(i-a1,j-b1,m),&
      !  Hstar(i,j,m), uquiv(i,j,m))
      !ELSE
      !  CALL upwind_m2 (u(i-c1,j-d1,m), u(i-a1,j-b1,m), &
      !  u(i,j,m), u(i+a1,j+b1,m), Hstar(i-a1,j-b1,m),&
      !  Hstar(i,j,m), uquiv(i,j,m))
      !END IF
```

```
            !  Calculate average u values over the cell
               uquiv(i,j,1) = half * (u(i-1,j,1) + u(i,j,1))
               uquiv(i,j,2) = half * (u(i,j-1,2) + u(i,j,2))
            END DO
            END DO
            END DO
            END IF
            IF (iout5 == 1) THEN
            DO j = j1,jm
               WRITE (UNIT=ifout5, FMT = fmt) (uquiv(i,j,1), i=i1,im)
               WRITE (UNIT=ifout5, FMT = fmt) (uquiv(i,j,2), i=i1,im)
            END DO
            END IF
            !  ...  Convert u values to flow values
            IF (iout6 == 1) THEN
            DO i = i1,im
            DO j = j1,jm
               uquiv(i,j,1) = uquiv(i,j,1) * H(i,j) * dy
               uquiv(i,j,2) = uquiv(i,j,2) * H(i,j) * dx
            END DO
            END DO
            DO j = j1,jm
               WRITE (UNIT=ifout6, FMT = fmt) (uquiv(i,j,1), i=i1,im)
               WRITE (UNIT=ifout6, FMT = fmt) (uquiv(i,j,2), i=i1,im)
            END DO
            END IF
            WRITE(6,*) NINT(time_seconds/(sixty*sixty))-stflow, ' hours'
            DEALLOCATE (uquiv)
         END IF
         END IF

         !  ...  Close files when run finished
         IF (nn == nsteps) THEN
            CLOSE(UNIT = ifout)
            CLOSE(UNIT = ifout2)
            CLOSE(UNIT = ifout3)
            CLOSE(UNIT = ifout4)
            CLOSE(UNIT = ifout5)
            CLOSE(UNIT = ifout6)
            CLOSE(UNIT = ifout7)
            CLOSE(UNIT = ifout8)
            CLOSE(UNIT = ifout10)
            CLOSE(UNIT = ifout11)
         END IF

      END SUBROUTINE output


!***********************************************************************
      SUBROUTINE end_output (execution_time1)
!***********************************************************************
!  Author :  Steve Andrews
!  Date :  November 2006
!  Purpose:  Output variables at end of the run
!----------------------------------------------------------------------

         REAL(KIND = 8), INTENT(IN) ::  execution_time1
         REAL(KIND = 8) ::  FPmass, percent_left
         INTEGER ::  i, j
```

```
   !  ...  Calculate mass of tracer remaining in the floodplain
   FPmass = zero
   DO i = i1,im
   DO j = j1,jm
      IF (domain(i,j)) FPmass = FPmass + c(i,j)*dx*dy*H(i,j)
   END DO
   END DO
   FPmass = FPmass + FPmassRm
   percent_left = FPmass / Cmass * 100.0D0
   WRITE(UNIT = ifout9, FMT = *) 'Total Tracer mass remaining = ', &
      FPmass, ' from remaining wet cells and cells that went dry'
   WRITE(UNIT = ifout9, FMT = *) 'Percent of tracer mass left = ', &
      percent_left
   WRITE(UNIT = ifout9, FMT = *) 'Tracer mass lost to dry cells = ',&
      FPmassRm

   !  ...  Write execution time to file
   WRITE(UNIT = ifout9, FMT = *) 'Run ended successfully'
   WRITE(UNIT = ifout9, FMT = *) 'Execution Time = ', &
      execution_time1/(sixty*sixty), ' hours'
   CLOSE(UNIT = ifout9)

END SUBROUTINE end_output

END MODULE SIFT2D_update
```

C.8 interp_hydro.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script file to read in stage data and interpolate between points
% data was given on 10 minute intervals - program reads Excel time
% and stage (in a text file) and outputs a similar text file with
% interpolated stages for dt intervals
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
format long

% input the number of time-stage data points that are present
numpoints = 937;

%create strings to output file
str1 = 'wave1_in'; str2 = '.txt'; str3 = 'terp.txt';
str = [str1 '' str2];
outstr = [str1 '' str3];

% open the text file containing the original points
% and read them into a matrix
% matrix A has column 1 = excel time, column 2 = water depth TS,
% 3 = water depth TN, 4 = TE, 5 = TW
fid = fopen(str,'rt');
A = fscanf(fid,'%g %g %g %g %g',[5,numpoints])';
fclose(fid);

% create an array of the new times
dt = input('Enter dt value in seconds ');
newhours = linspace(A(1,1), A(numpoints,1), ...
   round(600/dt*(numpoints-1)+1));

for i = 2:5
   newstage(:,(i-1)) = interp1(A(:,1), A(:,i), newhours');
end

% print the new data to a file with _interp at the end
fid = fopen(outstr,'w');
fprintf(fid,'%9.5f /t %9.5f /t %9.5f /t %9.5f /n', newstage');
fclose(fid);
```