University of Southern Queensland

Faculty of Engineering and Surveying

Finite Volume Solution of the Unsteady Free Surface Flow Equations

A dissertation submitted by

Adam Mark Gould

in fulfilment of the requirements of

Courses ENG4111 and 4112 Research Project

towards the degree of

Bachelor of Engineering (Civil)

Submitted: October, 2010

UNIVERSITY OF SOUTHERN QUEENSLAND

Faculty of Engineering and Surveying

Abstract

Water is a precious resource in much of Australia. Because of this, it is very important to be able to understand, control and manage this resource. One particular water intensive industry is the irrigation industry. Because of a very wide variety of irrigation methods it is hard to pick which is the most efficient or which can be developed into the most efficient. In aid to doing this, an unsteady free surface flow model is required. Many difficulties have been endured in developing a model that is accurate, versatile and robust enough to simulate many irrigation systems. Here the Finite Volume Method (FVM) gets investigated to find if it is a suitable method to be applied to these irrigation systems.

The FVM is investigated by applying a pre-existing MatLab FVM model to a number of case studies. In doing this a series of adaption's need to take place to make the model more generalized so then it can be used to simulate a wider range of systems. The case studies look at the inclusion of differed boundary conditions, friction and bed slope, channel geometry, and dry channel bed conditions.

It was found that the model was successful and proved versatile in modelling a very wide range of conditions throughout each case study. Comparing the valid steady state model outputs to results from the Mannings equation show the model has an average error of -0.019% for water flowing in downhill applications. To find the region where the model becomes inaccurate due to restrictions with the Mannings equation other situations were tested. It was found that for very low heights the comparison yielded very large errors. The threshold for model accuracy compared to the Mannings equation found inaccuracies at

about 0.002m for a Mannings n value of 0.03, a bed slope of 2/1000, and a low flow rate. Other Mannings n values and bed slopes could be used to find other thresholds. This affects the size of the threshold that can be used and conflicts against the value recommended of 0.0001m in Bradford & Sanders (2002b). This model has been successful within the case studies although some limitations have been proven.

UNIVERSITY OF SOUTHERN QUEENSLAND

Faculty of Engineering and Surveying

ENG4111 & ENG4112 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the University of Southern Queensland, its faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof Frank Bullen

Dean

Faculty of Engineering and Surveying

UNIVERSITY OF SOUTHERN QUEENSLAND

Faculty of Engineering and Surveying

Certification

I certify that the ideas, designs, and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Adam Mark Gould

Student Number: w0065087

_____

Signature

_____

Date

Acknowledgment

This research has been taken out under the supervision of Prof Rod Smith and Dr Malcolm Gillies.

Thanks must be given to Prof Brett Sanders from Irvine University in California for scripts and ongoing assistance. Prof Brett Sanders acknowledges Scott Bradford for helping with his scripts.

Table of Contents

List of Figures

List of Tables

Nomenclature

α	Wave speed	$m.s^{-1}$
b	Channel bottom width	m
A	Cross-sectional area	m^2
A_l	Cross-sectional area on the left side of the boundary	m^2
A_r	Cross-sectional area on the right side of the boundary	m^2
$\widehat{A_U}$	Jacobian matrix	
C_R	Courant number	
F	Flux vector	
F_c	Force exerted by channel walls	N
F_r	Froude Number	
g	Gravitational constant	$m.s^{-2}$
h	Depth of flow	m
h_l	Height on the left side of the boundary	m
h_r	Height on the right side of the boundary	m
j	Subscript for the spatial dimension	m
k	Subscript for the time dimension	s
L	Length of solution domain	m
m	mass	kg
n	Mannings n	
N	Number of computational cells	
P	Wetted Perimeter	m
p	Pressure	$N.m^{-2}$
R	Hydraulic radius	m
$\widehat{R}$	Riemann variable	
Q	Discharge	$m^3.s^{-1}$
So	Bedslope	
Sf	Friction slope	m
S	Source term vector	

t	Time variable	s
U	Vector of conservative variables	$m.s^{-1}$
u	Velocity	$m.s^{-1}$
u_l	Velocity on the left side of the boundary	$m.s^{-1}$
u_r	Velocity on the right side of the boundary	$m.s^{-1}$
V	Cross-sectionally averaged velocity	$m.s^{-1}$
x	Distance along channel	m
$\bar{y}$	Depth from free surface to centroid of wetted area	m
y	Depth	m
z	Elevation of the bed	m
$\widehat{\Lambda}$	Matrix of eigenvalues of AU	
ρ	Fluid density	$kg.m^{-3}$
Ω	Spatial domain of computational cell	m
ε	Dry bed tolerance	m
η	Height from datum	m

1. Introduction

1.1 Outline

Finite Volume Methods have been used successfully for simulating many different hyperbolic equations and have many advantages over other solution techniques. One form of hyperbolic equation is the shallow water equations which are otherwise known as the Saint Venant equations. These equations are used to solve gradually varying fluid flow problems like dam breaks and other similar problems. Other applications can also be with unsteady open channel flows, which is of our particular interest. This dissertation will investigate the finite volume balance for possible applications to irrigation channel modelling which have been difficult to model in the past for various reasons. The main difficulties have been with irregular beds and dry initial conditions, and have lead to various simplifications and adaptations which have taken away from the validity of the solution. This investigation has been set out to prove the validity of using the Finite Volume Method for irrigation channels and to investigate its ability to improve the accuracy and robustness of current irrigation modelling software.

1.2 Aim

The aim of this dissertation is to investigate, test, and prove the validity of the Finite Volume Solution for the use within irrigation modelling. This will be done via MatLab modelling with various case studies which have been designed to improve the usefulness of a pre-existing program by making the model more generic and better suited to a wider range of circumstances. The case studies also have the task of setting up opportunities to test its accuracy and robustness via simple tests discussed later within this dissertation.

1.3 Objectives

The objectives of this study are to:

1. Review Finite Volume Solution Techniques used with the unsteady free surface flow equations including the history, attributes, and applications of the techniques.
2. Describe in detail the algorithms and working of an appropriate example of a Finite Volume technique
3. Apply the MatLab codes of Sanders to a range of simple case studies to gain understanding of techniques and boundary conditions.
4. Modify the MatLab code to handle more complex cases.

5. Investigate application of Finite Volume techniques to flow over a dry bed by further modification of Sanders' code or development of original codes

1.4 Background

Water flow has been the topic of much research over the past couple of centuries. This is because it is important to understand how fluid will respond in set circumstances so that it is possible to utilize it efficiently. It has been difficult to mathematically describe many phenomena which has led to empirical methods and simplifications of some circumstances. The dynamic shallow water unsteady flow equations were introduced around 1871 by Saint Venant and are considered as a full mathematical description of shallow water flow. The Saint Venant equations do not rely on empirical methods or simplifications, but is a hyperbolic equation. Hyperbolic equations are tedious to solve exactly, but generally a good estimation can be achieved with the inclusion of computers. Before computers were available, it was often necessary to neglect many terms within these equations and work with much simpler equations which were solvable by hand.

Irrigation models need the full set of Saint Venant equations because often there are circumstances which the simplified versions would not accurately predict. Irrigation systems also have various physical reasons which make it hard to replicate real situations in a numerical model. This leads to the use of advanced techniques such as the Finite Volume Method as the method to solve these equations.

The Finite Volume Method is an established method of solving hyperbolic equations. The attractiveness of this method comes from being adequately robust, highly accurate and is not restricted to rectangular solution grids. The grids can be irregularly shaped and are generally fitted with a software package because manually fitting grids can be very tedious. Irregular grids are appealing for irregularly shaped channel geometries which have large variations in profile. If this profile were fitted with the standard rectangular grids, it can be very hard to fit grids which cover all of the channel bed. This leads to various other shaped cells, e.g. triangular, which can be made to fit almost any channel bed no matter the complexity. This will be covered in detail later within this dissertation

2. Hydraulics Theory

2.1 Introduction

Here we introduce the necessary theory to be able to understand the following chapters. First we shall look at some fluid flow concepts and then follow that through to the affects that these concepts will have on the numerical model.

2.2 Fluid flow concepts

2.2.1 Subcritical and Supercritical Flows

When fluid is flowing, it can take on different characteristics depending on velocity, height, bed shear and bed slope. A fast flowing fluid with relatively shallow height is termed supercritical while a slower fluid with relatively deep height is termed subcritical. A simple way to test this is to drop a small object into the flowing water and observe how these ripples propagate as shown in Figure 2.1.

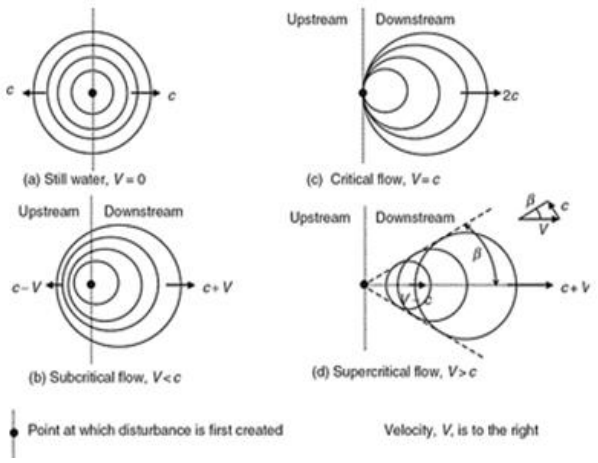


Figure 2.1 Effect of Flow on Propagation of a Disturbance Akan(2006)

The numerical way to differentiate between these two conditions is called the Froude number. It is shown in Eq 2-1 Chadwick et al(2004).

$$F_r = \frac{V}{\sqrt{gy}} \qquad \text{Eq 2-1}$$

The numerical value of the Froude Number can then be interpreted by the following system from Chadwick et al(2004).

If $F_r > 1$ then:

- The flow is considered supercritical
- A disturbance will travel downstream only
- Upstream levels are unaffected by downstream controls

If $F_r < 1$ then:

- The flow is considered subcritical
- A disturbance will travel downstream and upstream
- Upstream levels are affected by downstream controls

The effect this has on the modelling process is detailed in Chapter 2.4.2.

2.2.2 Laminar and Turbulent Flows

This section addresses how laminar and turbulent flows affect the bed shear stress that acts on a channel. As shown in Figure 2.2, there is a region called the laminar boundary layer. The thickness of this layer depends on the viscosity of the water and the velocity of the water outside of the boundary layer. For laminar flow where velocities are slow and where the flow stays fairly straight, the laminar boundary layer is quiet thick. As the water velocity speeds up, this laminar boundary layer will get quiet small as the flow roughens. There is a layer known as the transitional layer which will develop where the flow is somewhere between laminar and turbulent. Finally, there will be a fully turbulent or rough turbulent layer which extends into the middle of the channel as shown the turbulent zone in Figure 2.2.

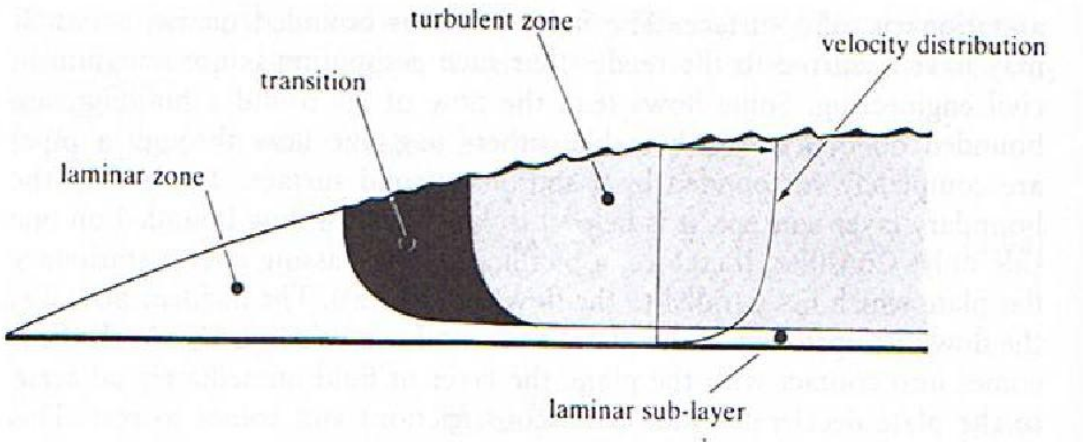


Figure 2.2 Laminar and Turbulent Boundary Layers Chadwick et al(2004)

If we examine a particle of fluid moving, it will follow something called a streamline, as shown in Figure 2.3.

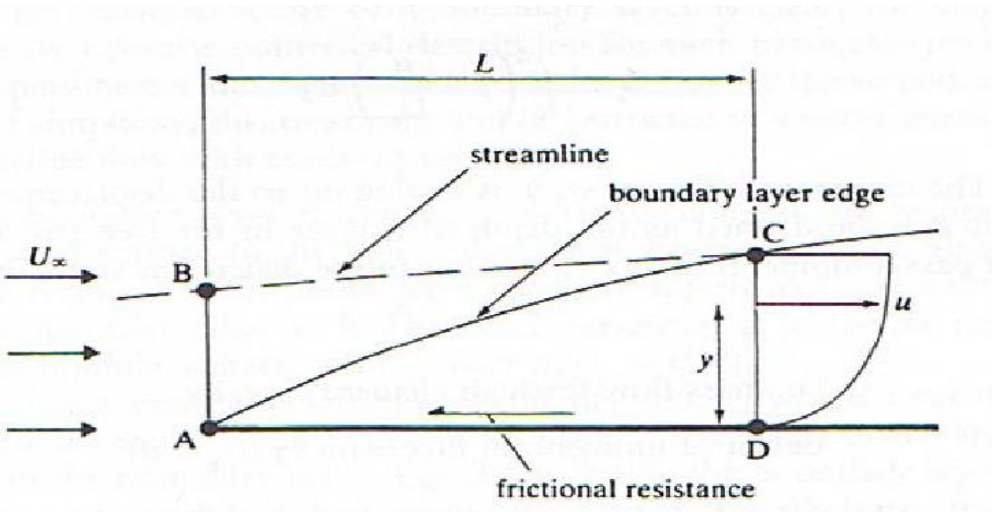


Figure 2.3 Longitudinal Section through a Boundary Layer Chadwick et al(2004)

When the fluid velocity is laminar, the streamlines will be straight. As the fluid moves into the transitional stage, the streamlines will become wavy. When turbulent flows are

achieved, the streamlines movement will become unpredictable. This is shown in Figure 2.4.

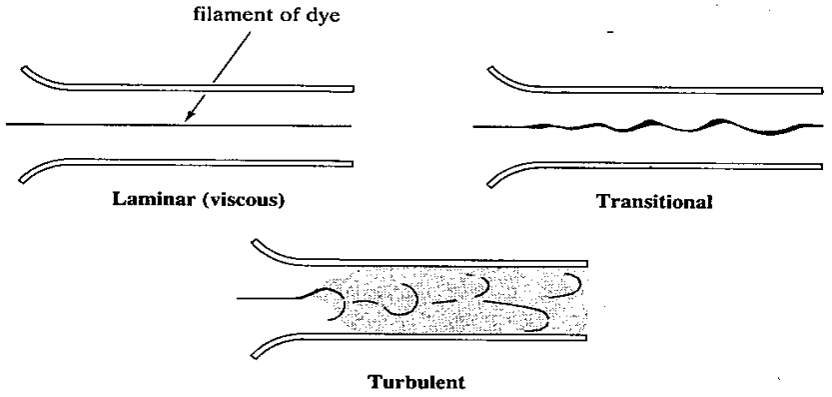


Figure 2.4 Reynolds' Experiment Chadwick et al(2004)

The energy transferred to the boundary walls via heat is different depending on boundary layer conditions. A laminar flow will have less shear stress then a rough turbulent flow. In open channels, turbulent flows are far more common then laminar flows which is why the equations to find the energy lost via bed shear are all suited to transitional or turbulent flows. The Mannings equation, Eq 2-2, is one such equation which is suited to rough turbulent flows.

$$S_f = n^2 \frac{|V|V}{R^{\frac{4}{3}}} = n^2 \frac{|Q|Q}{A^2 R^{\frac{4}{3}}} \qquad \text{Eq 2-2}$$

$$R = \frac{A}{P} \qquad \text{Eq 2-3}$$

The Mannings equation uses a roughness value which accounts for the roughness of the bed. Channels in common use have roughness values between 0.013-0.03 and between 0.03-0.08 for most natural systems Akan(2006).

2.3 1 Dimensional Saint Venant Equations

The dynamic wave approximation of the Saint Venant equations is presented below including the full derivation of the equations as presented by Novak et al. (2010).

To develop the equations which describe a flow, it is assumed that fluids adhere to the fundamental laws of physics Chadwick et al (2004). As stated in Chadwick et al.(2004) The pertinent laws are:

1. Conservation of Mass (Continuity)
2. Conservation of Energy
3. Conservation of Momentum

Here we are only concerned with continuity and the conservation of momentum. These laws will be governing in the derivation of the Saint Venant equations. The derivation of these equations are stated throughout (Akan 2006; Chadwick et al. 2004; Novak et al. 2010; Szymkiewicz 2010; Yen 1973 as cited in Akan 2006). This derivation of the continuity and momentum equation is based on that presented in Novak et al.(2010)

2.3.1 The continuity Equation

Continuity stipulates that matter can neither be created nor destroyed. It can only be transformed. Chadwick et al (2004). As transformations due to chemical changes are of no use in this instance, the law then reduces to the conservation of mass throughout time Chadwick et al (2004). Eq 2-4 which is Saint Venants version of the continuity equation represents this conservation of mass.

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \qquad \text{Eq 2-4}$$

For open-channel flow, the continuity equation is derived by writing the conservation of mass law for a control volume extending from x to x + dx, See Figure 2.5, between times t and t + dt Novak et al.(2010).

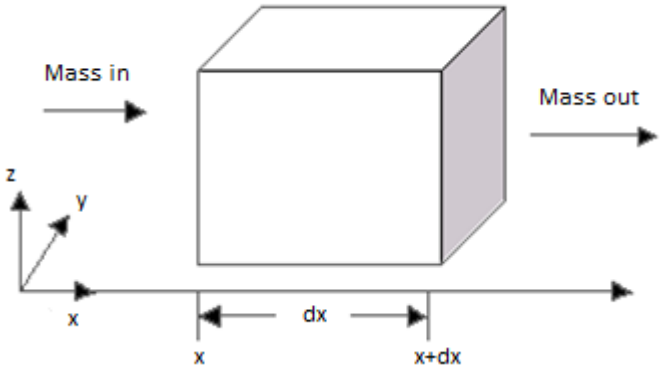


Figure 2.5 Control Volume

Consider the control volume in Figure 2.5. Length being dx, and the average cross-sectional area is A. The principle of conservation of mass states;

Rate of change of mass of water into the element = Net Rate of mass into element Akan (2006).

This principle can be written mathematically Novak et al.(2010) as;

$$m(t + dt) = F(x) - F(x + dx) \qquad \text{Eq 2-5}$$

Where m(t), Eq 2-6, is the mass contained within the control volume at time t, and F(x), Eq 2-7, is the mass that passes x, between t and t + dt (Flux) Novak et al.(2010).

$$m(t) = \rho A(t)dx \qquad \text{Eq 2-6}$$

$$F(x) = \rho Q(x)dt \qquad \text{Eq 2-7}$$

Substitution of these equations into Eq 2-5 results in

$$[\rho A(t + dt) - \rho A(t)]dx = [\rho Q(x + dx)]dt \qquad \text{Eq 2-8}$$

Noting that

$$A(t + dt) - A(t) = \frac{\partial A}{\partial t} dt \qquad \text{Eq 2-9}$$

And

$$Q(x) - Q(x - dx) = -\frac{\partial Q}{\partial x} dx \qquad \text{Eq 2-10}$$

Simplifying by dt, dx, ρ leads to

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0 \qquad \text{Eq 2-11}$$

2.3.2 The momentum equation

The law of the momentum states that a body in motion cannot gain or lose momentum without some external force being applied to it Chadwick et al (2004). The momentum equation of Saint Venant is stated as Eq 2-12.

$$\frac{\partial VA}{\partial t} + \frac{\partial}{\partial x}(V^2 A + gA\bar{y}) = F_c gA(S_0 - S_f) \qquad \text{Eq 2-12}$$

The momentum equation is obtained by applying the law of momentum to the same slice of length dx as the continuity equation Novak et al.(2010). The law of momentum can be written as;

Time rate of change of the momentum accumulated within the element = net rate of momentum transfer into the element + sum of external forces in the flow direction Akan (2006).

Thus the momentum balance can be written as Eq 2-13, Novak et al.(2010).

$$M(t + dt) - M(t) = F(x) - F(x - dx) + Sdt \qquad \text{Eq 2-13}$$

Where M(t) is the momentum of the fluid contained within the control volume at the time t, F(x), (also known as the momentum flux) Novak et al.(2010), is the amount of the momentum transferred into the control volume by the flow over the time interval dt at the point x, Akan (2006), and S is the sum of the external forces in the flow direction which are exerted on the control volume between t and t+dt (Akan 2006; Novak et al.2010).

The momentum M(t) is the product of the mass contained within the control volume and the average flow velocity Novak et al.(2010).

$$M(t) = \rho AVdx = \rho Qdx \qquad \text{Eq 2-14}$$

Where V is the average flow velocity over the cross-sectional. The momentum flux can then be defined by Eq 2-15 from Novak et al.(2010).

$$F(x) = \int_{t}^{t+dt} \int_{A} \rho \tilde{u}^2 dA dt = \rho dt \int_{A} \tilde{u}^2 dA \qquad \text{Eq 2-15}$$

Where ũ is the point value of the flow velocity. If the flow velocity is uniform over the entire cross-sectional area, then ũ=V and the equation becomes

$$F(x) = \rho dt \int_{A} V^2 dA = \rho dt A V^2 = \rho \frac{Q^2}{A} dt \qquad \text{Eq 2-16}$$

In practice, the non-uniform character of the velocity distribution over the cross-section is accounted for by a coefficient β, Novak et al.(2010), which generally gets assumed to be uniform thus β =1, but can be more if the velocity distribution is non-uniform .

$$F(x) = \beta \rho \frac{Q^2}{A} dt \qquad \text{Eq 2-17}$$

The external forces applied to the control volume are the following:

1. Acting on the upstream and downstream faces of the control volume will be pressure forces. The sum of these forces are stated in Eq 2-18

$$P(x) - P(x + dx) = -\frac{\partial P}{\partial x} dx \qquad \text{Eq 2-18}$$

With an assumption that the pressure forces acting on the upstream face of the control volume are hydrostatic in nature then Eq 2-19 holds true:

$$P(x) = \int_A p(x,z)dA = \int_{z_b}^{\eta} p(x,z)b(x,z)dz = \int_{z_b}^{\eta} \rho g(n-z)b(x,z)dz \qquad \text{Eq 2-19}$$

Because of the assumption of a prismatic cross section the equation can be simplified as follows. The auxiliary variable $\xi = z - z_b$ is introduced allowing the equation to be re-written as: Novak et al.(2010).

$$P(x) = \rho g \int_0^Y (Y-\xi)b(x,\xi+z_b)d\xi \qquad \text{Eq 2-20}$$

2. There are two main vertical components acting on the control volume. The first is the weight of the control volume acting on the channel bed. The other is the reaction of the bed due to bedslope. Since the reaction of the bottom is exerted in the direction orthogonal to the bed of the channel, the following equity holds Novak et al.(2010).

$$R_x = R_z S_o \qquad \text{Eq 2-21}$$

Where R_x is the x component of the bottom reaction. As the sum of the vertical forces is zero Novak et al.(2010).

$$R_z - mg = 0 \qquad \text{Eq 2-22}$$

Where g is the gravitational acceleration and R_z is the vertical component of the bed reaction. Consequently the x-component of the reaction is given by Novak et al.(2010) as:

$$R_x = mgS_o = \rho gAS_o dx \qquad \text{Eq 2-23}$$

3. For purposes of friction, the assumption of a nearly horizontal bed means that the frictional force is seen as only acting in the x direction only. This friction force is proportional to length of the control volume and is usually written as Eq 2-24 Novak et al.(2010).

$$F_b = -mgS_f = -\rho gAS_f dx \qquad \text{Eq 2-24}$$

Where S_f is defined as the slope of the energy line. (Friction induced head loss per unit distance) Novak et al.(2010). As stated earlier Sf will be found using the Mannings equation for rough turbulent flows.

Now this derivation is lengthy, but important when looking at the background of the different forms of the Saint Venant equations and the history of shallow water solvers. Looking at the momentum equation

$$\frac{\partial VA}{\partial t} + \frac{\partial}{\partial x}(V^2A + gA\bar{y}) = F_c gA(S_0 - S_f) \qquad \text{Eq 2-25}$$

This equation when rearranged for S_f can be simplified depending on your intended use. This is shown as follows.

$$S_f \cong S_o - \frac{\partial h}{\partial x} - \frac{V}{g}\frac{\partial V}{\partial x} - \frac{\partial V}{g\,\partial t} \qquad \text{Eq 2-26}$$

Represented as

$$1 \cong 2 - 3 - 4 - 5$$

Where Julien(2002) explains;

1 = friction slope

2 = bed slope

3 = pressure gradient, downstream change in flow depth

4 = velocity head gradient, downstream change in flow velocity

5 = local acceleration for unsteady flow

By neglecting one or more of the above terms it is possible to obtain solution techniques such as, Kinematic wave approximation, Diffusive, Quasi-steady, or the Dynamic shallow water equations.

Kinematic wave approximation;

$$S_f \cong S_o \qquad \text{Eq 2-27}$$

Diffusive approximation;

$$S_f \cong S_o - \frac{\partial h}{\partial x} \qquad \text{Eq 2-28}$$

Quasi-steady approximation;

$$S_f \cong S_o - \frac{\partial h}{\partial x} - \frac{V}{g}\frac{\partial V}{\partial x} \qquad \text{Eq 2-29}$$

Dynamic wave approximation;

$$S_f \cong S_o - \frac{\partial h}{\partial x} - \frac{V}{g}\frac{\partial V}{\partial x} - \frac{\partial V}{g\,\partial t} \qquad \text{Eq 2-30}$$

Sturm(2001) presents a version of the equations that includes lateral inflows but is not pursued further here.

2.4 Applying the Saint Venant Equations

Applying the Saint Venant equations to a real situation posses many difficulties which are looked at within this dissertation. A simple system will have few issues, but once a dry bed, channel geometry, and junctions are introduced it brings in issues of instability, flux formulation difficulties, and boundary condition problems. In the next couple of chapters

these will be introduced and techniques of solving for each will be investigated in some depth.

2.4.1 Dry Bed Treatment

The term bed refers to the ground or channel surface itself and is a common term used within river hydraulics. A situation where there is a length of channel bed where there is no water is known as being dry. This situation is commonly encountered when there is a channel being filled causing a wave to propagate over a dry bed. This also works in reverse where the water flow is retreating, draining the bed and subsequently leaving it dry. For numerical modelling the channel is split up into many cells and when the wave front is passing through a cell it is defined as partially wet. See Figure 2.6.

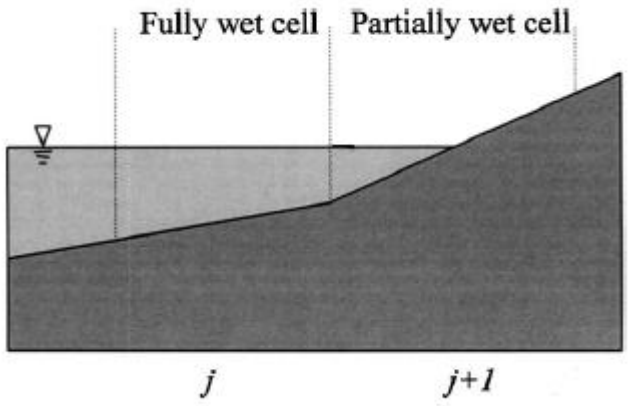


Figure 2.6 Partially Wetter Cell Bradford & Sanders (2002a)

Partially filled rectangular cells are defined in Bradford and Sanders (2002a) as a cell having at least one but no more than three corners dry. Indescrependicies come into the treatment of drybed in these partially wet cells which occur along the wave front where the height of water aproaches 0. When this depth is then cell averaged, i.e. altered to an equivalent thin layer of water over the entire cell, it can lead to an artificial propagation of the wave front which leads to artificial spreading into adjacent cells Bradford and Sanders (2002a). In the case of averaging the depths of a partially filled cell produces small depths throughout the entire cell which causes instability and inaccuracy when the momentum fluxes terms are divided by the height (Bradford and Sanders (2002a); Garcia-Navarro & Toro(2007); Novak et al(2010)). Commonly a minimum area or depth tolerance, ε, needs to be set in order to maintain stability in the solution. (Bergundelli, Bradford & Sanders (2008); Bergundelli & Sanders (2006 & 2007); Bradford and Sanders (2002a &2002b)). Only if the height of water is above this tolerance is the cell deemed to be wet and the momentum fluxes computed Bradford and Sanders (2002b). Sensitivity to the value of ε is low in frictionless simulations but for models that include friction, via the Manning expression Eq 2-2, it brings in an increased sensitivity because there is a height term on the bottom of the Mannings equation Bradford and Sanders (2002a). This will result in unrealistically high predictions of shear stress in shallow regions near wet/dry interfaces which makes the momentum equations become stiff and the solution unstable Bradford and Sanders (2002a). This requires ε to be increased by one or two orders of magnitude to values above 1^{-6}m. It has been reported that this value can be decreased to as much as 1^{-30}m providing a much higher level of accuracy Bradford and Sanders (2002b). This tolerance layer suggests that there is actually a very thin layer of fluid within the model that is needed

to hold stability. If the real fluid surface prediction falls below this tolerance, the cell is perceived to be dry and no water is within the cell. Bradford and Sanders (2002b) justifies this because the model accurately predicts wave propagation. Whereas with other models (Anastasiaduo-Partheniou, Banti & Aissis 2010; Zhang and Cundy 1989; Playan et al. 1994) which also assume a thin layer of water result in the incorrect propagation of waves. For example, bores that reach the shoreline will collapse and intrude on a dry beach as a depression wave but where there is a thin layer of water on the shore, a model will wrongly predict that the bore will travel up the beach Bradford and Sanders (2002b). In addition, such models will have instability when grid cells become dry during a simulation Bradford and Sanders (2002b).

2.4.2 Boundary Conditions

The boundary conditions are used as a way to provide the program with a set of parameters which it will use to predict the flow characteristics throughout the rest of the channel. These vary depending on user requirements, model geometry, and flow conditions. With model geometry, it is possible to set up inflows, outflows, or reflective boundaries depending on the particular situation. Some applications require a change from subcritical to supercritical or vice-versa which varies the number of boundary conditions required. For instance, as a channel sloping uphill fills, it can have a subcritical profile, but when the wave runs out of momentum or draining is taking place the water is then flowing downhill and it is then

possible to see a change to critical flow at the outlet. For super-critical inflow boundaries both height and velocity are stated in the boundary conditions and the gradients of both are set to zero (Bradford and Sanders 2002b). For subcritical flows, the inflow boundary condition can have specified the flow rate and the gradient of the velocity is set to zero (Bradford and Sanders 2002b). For outflow conditions, the velocity and height are extrapolated to the boundary conditions. The gradients of these boundary conditions are kept the same as the interior cell values. The height will then be extrapolated from the first interior cell and then the area and velocity are easily determinable (Bradford and Sanders 2002b).

2.4.3 Lateral Flows

Lateral flows are flows which enter or leave a channel at a junction. An example of which is the branching system shown in Figure 2.7. Other lateral flows can be more regularly spaced like furrows leaving a head ditch for irrigation purposes. These junctions can complicate modelling a system e.g. the proportion of flow flowing out of each junction must be known or a relationship for this must be obtainable.

Figure 2.7 Branching Channel System Novak et al. (2010)

Junctions, similar to a branched canal network, should be treated by isolating each channel as separate channels by splitting the channel at the junction. Therefore for the simple T section junction shown in Figure 2.7 there will be three separate channels as shown in Figure 2.8. These separate channels are defined as downstream of the junction, upstream of the junction, and the offshoot channel (Novak et al 2010). From an algorithmic point of view, the classical-flow equation is still applicable within each reach and the structure or junction is considered as a boundary condition for each of the elementary channels, hence the term ‘internal boundary’(Novak et al 2010).

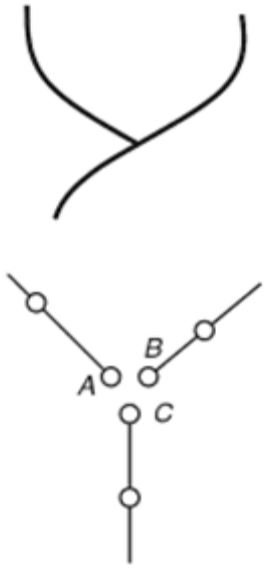


Figure 2.8 Junction Splitting Novak et al. (2010)

2.5 Applications of the Saint Venant Equations

The shallow water equations are used to model wave propagation, lake hydrodynamics, surface irrigation, overland flow, estuarine and tidal flows. The Saint Venant equations can also be used for structures within wave zones to analyse the conditions exerted on the structure. These equations are accurate but are computationally expensive which means that at least desktop computers are needed to calculate to any form of accuracy. The shallow water equations have been used extensively of recent times in programs like HEC-RAS which is an open channel solver for designing civil structures. HEC-RAS can be used to

solve the simplified shallow water equations and provisions have been made for solving the full set of shallow water equations. If accuracy is not essential, it would be beneficial to consider using one of the less complete forms of the equations, such as the kinematic equation or similar.

2.6 Solution Techniques for the Saint Venant Equations

The Saint Venant equations are an example of hyperbolic partial differential equations. Hyperbolic partial differential equations have been solved using a number of methods which have varying forms of accuracy and robustness. These solutions include both implicit and explicit techniques which rely on the finite difference approximation. The finite difference approximation is not suitable for the Saint Venant equations as the accuracy of the solution, and robustness, does not reflect the standard of accuracy given by using the full set of the Saint Venant equations. Also, complications within these methods would arise when considering irregular geometries which require irregular grid patterns. Alternatively, there is the finite volume approximation which is accurate enough, robust enough and can handle irregular topographies. The FVM is covered in Chapter 3.

3. Godunov-Type Algorithms: The Finite Volume Method (FVM)

3.1 Introduction

This dissertation examines a Matlab code provided by Sanders as described in Bradford and Sanders (2002a &2002b). This code uses a Godunov algorithm to solve an abbreviated form of the shallow water equations. Within this chapter, we will analyze the Godunov algorithm as a way of building an understanding of the theory behind the model. The model follows a version of the MUSCL-Hancock approach to the FVM which incorporates a Roe type Riemann invariant solver.

The FVM falls into the family of Godunov-type algorithms and is a technique for solving a system of hyperbolic equations. This method is considered very accurate as it conserves mass at every time step Bradford and Sanders (2002b). It operates by updating the solution within some control volume and includes all the intercell mass and momentum flux contributions in a single step (Toro 2009).

3.2 The Godunov type algorithm

Before getting into the details of the Godunov algorithm, we must first build some background knowledge so then we can understand the process. We will do this by going through some definitions.

3.2.1 Definitions

3.2.1.1 Subscripts

Table 3.1 Subscript meanings

Subscript	Definition
j	Spatial coordinate
k	Time

3.2.1.2 Cartesian and Non-Cartesian Grids

Cartesian grids are grids that follow the x, y and z planes. See Figure 3.1. They are not restricted to rectangles but every component of the grid must follow one of the Cartesian planes.

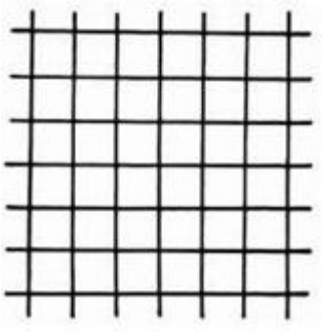


Figure 3.1 Cartesian Grid

Non-Cartesian Grids are those which have parts that do not follow the x y z planes. Figure 3.2 is a representation of a Non-Cartesian grid but Non-Cartesian grids are not limited to just this shape.

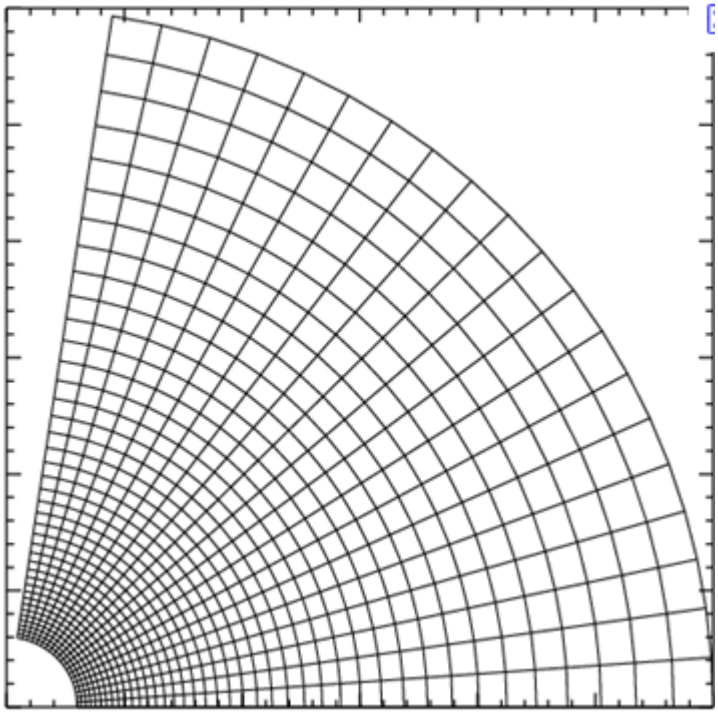


Figure 3.2 Non-Cartesian Grid

3.2.1.3 Channel Geometry

The channel geometry refers to the shape of the channel where the water flows. With increased complexity in the channel shape comes increased difficulty in estimation of the area and finding the wetted perimeter. The process we are using to find the fluxes can find the fluxes for arbitrary topography which means that little accuracy is sacrificed to find the fluxes. Roe's numerical flux scheme can be used as a good estimate of the interface flux. Although not an exact solution, it doesn't have a large influence on the model as cell

averaging occurs at every time step (Bradford and Sanders 2002b). Roe's numerical flux scheme will be discussed further in this chapter.

Complex channel geometry can complicate the process of forming finite volumes especially when non-uniform grid patterns are introduced. Often software is used to create irregular grid patterns because manually creating grids by hand can be exceptionally tedious.

3.2.1.4 The Riemann Problem

The Riemann problem involves solving what is known as a piecewise constant problem. Figure 3.3 demonstrates both the Riemann problem and the piecewise constant problem. Something that is piecewise constant has a constant value for each part or cell in the field. For example, within Figure 3.3 between 0 and x_o the cell has one value U_L and the cell after x_o has another constant value U_R. For the Riemann problem, we must examine the point x_o. At x_o there are then two values for U namely U_L & U_R. The solution to the Riemann problem is being able to find a singular value that represents both values. This value is an average of the left and right values and is the solution to the Equivalent Riemann Problem (ERP) and is known as data or variable reconstruction.

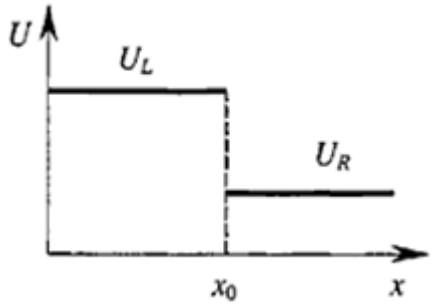


Figure 3.3 The Riemann Problem Guinot(2003)

3.2.1.5 The Equivalent Riemann Problem and the Generalized Riemann Problem

The Generalized Riemann problem (GRP) is a problem where the relationships on the left and right sides of the point in examination are not constant. See Figure 3.4. This means at point x_0 the 2 values of U do not represent their entire cell. This requires that the values of U be converted into values that are representative of the entire cell by averaging the non-constant relationship into an equivalent piecewise constant problem. The process of converting nonlinear functions to piece-wise linear functions is the process of turning the GRP to the Equivalent Riemann Problem (ERP).

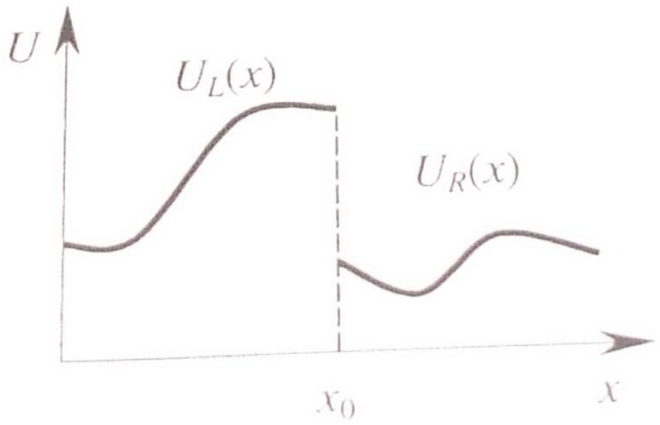


Figure 3.4 The Generalized Riemann Problem Guinot(2003)

3.2.1.6 Source Terms

Source terms is the term given to the affects that are not taken into account via the evolution in time formula provided as Eq 3-24 or Eq 3-25. Source term possibilities include bed slope, friction head loss (Bergundelli, Bradford & Sanders (2008); Bergundelli & Sanders (2006 & 2007); Bradford and Sanders (2002a &2002b)), infiltration Walker & Kasilingam (2004), wind forces Garcia-Navarro & Toro (2007), eddy diffusion Bradford & Sanders (2006), and sediment transport Julien(2002).

3.2.1.7 Intercell Fluxes

Mass and momentum fluxes are calculated to determine how much water from the last cell is being transferred into the next cell and the speed at which it is travelling. These are also stated as flux invariants in some texts.

3.2.1.8 The Finite Volume Method (FVM)

The variation of the Godunov algorithm we are using is the FVM. Instead of breaking the model into discrete cells, we break the model into discrete volumes. The main advantages of this method are that it can handle Non-Cartesian geometries which are required for most natural circumstances. It does not need to generate and remove cells around wetting and drying boundaries Bradford Sanders (2002b). It can handle subcritical and supercritical boundaries with only minor adjustments.The FVM appears very promising and will get investigated further within this chapter.

3.2.1.9 Limiters

Limiters are used to prevent spurious oscillations at discontinuities within the model. These spurious oscillations are encountered in the higher order of accuracy schemes Bradford & Sanders (2006). Commonly used limiters are MinMod, Double MinMod, and Superbee Bradford & Sanders (2006).

3.2.2 The Process of the Godunov-type Algorithm (Toro 2009)

The Godunov-type algorithm has six main steps which are detailed in (Guinot 2003;Toro 2009)

1. Involves the discretisation of space into finite volumes or computational cells where the solution of U, is sought over each cell. (Guinot 2003). See Figure 3.5

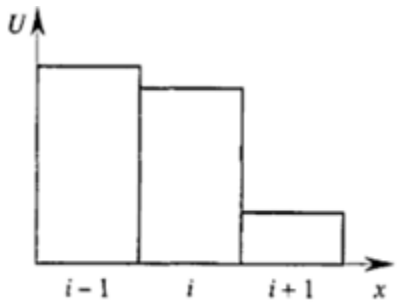


Figure 3.5 Discretion into Finite Volumes Guinot(2003)

2. Finding the GRP in each cell by reconstruction of each cells distribution of U. (Guinot 2003). See Figure 3.6. This is done using cell limited gradients and is the predictor step of the MUSCL-Hancock approach covered later in this chapter.

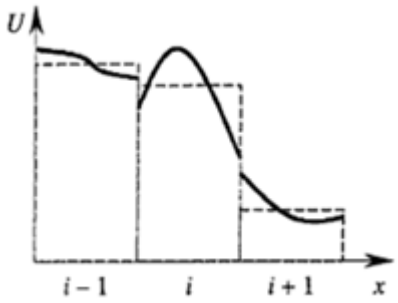


Figure 3.6 Redistribution into Cell Averages Guinot(2003)

3. Convert GRP to ERP by converting the piecewise constant values into a single value at each node.

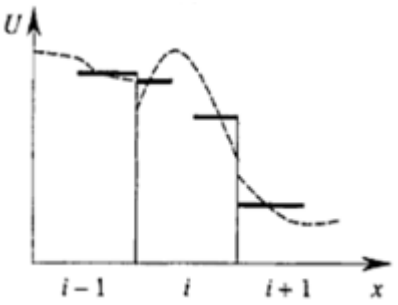


Figure 3.7 Conversion to ERP Guinot(2003)

4. Determine the fluxes. Determining the fluxes manually is a lengthy process that requires trial and error and iteration and should be avoided by using Riemann solvers (Guinot 2003). Within our model we use Roe's Reimann solver as a part of

the Flux Invariants step of the MUSCL-Hancock. Figure 3.8 is a representation of this.

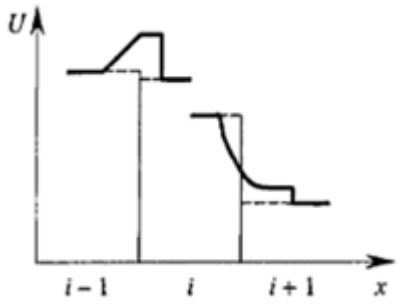


Figure 3.8 Flux Determination Guinot(2003)

5. Computation of the value of U at the next time step via flux balance over the cells (Guinot 2003). This represented in Figure 3.9 and is the corrector step of the MUSCL-Hancock scheme.

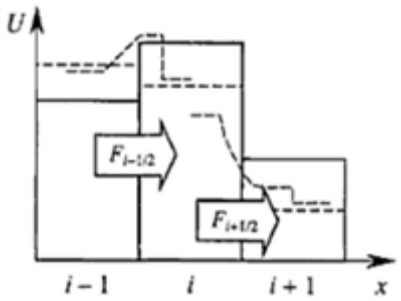


Figure 3.9 Flux Balancing Guinot(2003)

6. Source terms (If Any). This is represented by Guinot(2003) in Figure 3.10 as an end adjustment to the results but it is quiet often included as an extra part within the

fifth step (Bergundelli, Bradford & Sanders (2008); Bergundelli & Sanders (2006 & 2007); Bradford and Sanders (2002a &2002b), Julien(2002), Strelkoff(1970)).

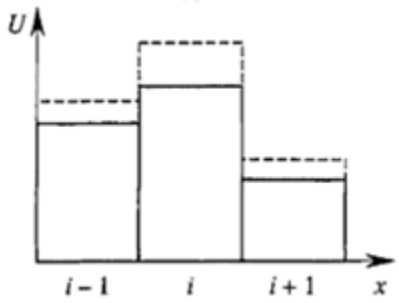


Figure 3.10 Source Term Adjustment Guinot(2003)

These six steps will be detailed within this chapter.

3.3 Structure of the Finite Volume Method

The finite volume method is structured according to several techniques which have been successfully used to solve hyperbolic equations and originated with the Euler equations and the aerospace industry. These techniques follow the Godunov type algorithm detailed in Chapter 3.2 and have incorporated a Predictor-Corrector solution format in combination with several other techniques. The layout of the finite volume method generally has the following steps;

1. Prediction Step (Godunov step 2 and 3)

2. Determine Fluxes (Godunov step 4)
3. Correction Step (Godunov step 5)

3.3.1 Prediction Step

The predictor step progresses A (Area), and V (velocity) to a halfway point in the time-step. The result is accurate except where there are discontinuities, e.g. hydraulic jumps, to which the gradients of the variables have to be limited to prevent over-predicting and under-predicting oscillations. The prediction step uses a rearranged version of the Saint Venant equations of Eq 2-11 and Eq 2-12. The momentum equation, Eq 2-12 can be rearranged and simplified as shown in Eq 3-1 through to Eq 3-8 to result in Eq 3-10 which is used for the predictor step to find velocity.

$$\frac{\partial VA}{\partial t} + \frac{\partial}{\partial x}(V^2A + gA\bar{y}) = F_c gA(S_0 - S_f) \qquad \text{Eq 3-1}$$

Expanding leads to

$$\frac{\partial VA}{\partial t} + \frac{\partial V^2A}{\partial x} + \frac{\partial gA\bar{y}}{\partial x} = F_c gA(S_0 - S_f) \qquad \text{Eq 3-2}$$

Division by A results in

$$\frac{\partial V}{\partial t} + \frac{\partial V^2}{\partial x} + \frac{\partial g\bar{y}}{\partial x} = F_c g(S_0 - S_f) \qquad \text{Eq 3-3}$$

Multiplied by dt results in

$$\partial V + \frac{\partial V^2}{\partial x}dt + \frac{\partial g\bar{y}}{\partial x}dt = F_c\, g\, dt(S_0 - S_f) \qquad \text{Eq 3-4}$$

Simplifying

$$\partial \mathrm{V} + \frac{\mathrm{dt}}{\mathrm{dx}}(\mathrm{V}\,\partial \mathrm{V} + \mathrm{g}\,\partial \bar{\mathrm{y}}) = \mathrm{F_c}\,\mathrm{g}\,\mathrm{dt}(\mathrm{S_0} - \mathrm{S_f}) \qquad \text{Eq 3-5}$$

Rearranging for ∂V results in

$$\partial \mathrm{V} = -\frac{\mathrm{dt}}{\mathrm{dx}}(\mathrm{V}\,\partial \mathrm{V} + \mathrm{g}\,\partial \bar{\mathrm{y}}) + \mathrm{F_c}\,\mathrm{g}\,\mathrm{dt}(\mathrm{S_0} - \mathrm{S_f}) \qquad \text{Eq 3-6}$$

Where

$$V_j^{k+1/2} = V_j^k + \partial \mathrm{V} \qquad \text{Eq 3-7}$$

And where subscript k+1/2 results in dt=dt/2

$$V_j^{k+1/2} = V_j^k - \frac{\mathrm{dt}}{2\mathrm{dx}}(\mathrm{V}\,\partial \mathrm{V} + \mathrm{g}\,\partial \bar{\mathrm{y}}) + \mathrm{F_c}\,\mathrm{g}\,\frac{\mathrm{dt}}{2}(\mathrm{S_0} - \mathrm{S_f}) \qquad \text{Eq 3-8}$$

In a very similar way the continuity equation Eq 2-11 can be used to produced Eq 3-9 where we solve for ∂A instead of ∂V. Eq 3-9 is the equation used to determine the cross-sectional area at a half time step and equation Eq 3-10 is

$$A_j^{k+1/2} = A_j^k - \frac{\Delta t}{2\mathrm{dx}}(V\overline{\Delta A} + \mathrm{A}\overline{\Delta \mathrm{V}})_j^k \qquad \text{Eq 3-9}$$

$$V_j^{k+1/2} = V_j^k - \frac{\Delta t}{2\mathrm{dx}}\left(g\overline{\Delta h} + \mathrm{V}\overline{\Delta \mathrm{V}}\right)_j^k + \frac{g\Delta t}{2}\left((S_0)_j^k - \frac{1}{2}\left(S_f\right)_j^k - \frac{1}{2}\left(S_f\right)_j^{k+1/2}\right) \qquad \text{Eq 3-10}$$

The values calculated within the predictor step are then used for the calculation of the flux terms which are determined in the following section using a variation of the MUSCL-Hancock technique.

3.3.2 The MUSCL-Hancock Scheme – Data Reconstruction

Once the prediction step finds the cell averaged values, ERP needs to be solved. The MUSCL approach is used here and three has three main steps.

1. Data Reconstruction
2. The Riemann Problem
3. Evolution in Time

3.3.2.1 Data Reconstruction

In the data reconstruction step, data cell averaged values from the prediction step are locally replaced by piece-wise linear functions in each cell (Toro 2009). Examples from (Sanders B 2001) are shown below for height, h, velocity, V, and area, A.

$$h_L = h_j + \frac{1}{2}\left(\overline{\Delta h_j}\right) \quad \text{Eq 3-11}$$

$$h_R = h_{j+1} - \frac{1}{2}\left(\overline{\Delta h_{j+1}}\right) \quad \text{Eq 3-12}$$

$$V_L = V_j + \frac{1}{2}\left(\overline{\Delta V_j}\right) \quad \text{Eq 3-13}$$

$$V_R = V_{j+1} - \frac{1}{2}\left(\overline{\Delta V_{j+1}}\right) \quad \text{Eq 3-14}$$

The area A on the left and right side of each cell face is then computed using the extrapolated depths and the depth-area relationship defined at the cell face. Hence,

$$A_L = A(h_L) \quad \text{Eq 3-15}$$

$$A_R = A(h_R) \quad \text{Eq 3-16}$$

3.3.2.2 The Riemann Problem

Once the variables are calculated at the cell boundary the mass and momentum fluxes can be calculated via a Reimann problem. For the rectangular or triangular case this is simple, but for arbitrary topography an approximate solution needs to be formulated. The following Roe scheme is used Sanders (2001).

$$F_l = \frac{1}{2}\left(F_L + F_R - \left|\widehat{A_u}\right|\Delta U\right) \quad \text{Eq 3-17}$$

Where F_L and F_R are fluxes evaluated with MUSCLE reconstructed data

Where A_U is the Jacobian matrix given by dF/dU and the matrix is defined as,

$$\left|\widehat{A_u}\right| = \hat{R}\left|\hat{\Lambda}\right|\hat{L} \quad \text{Eq 3-18}$$

For one-dimensional flow, these matrices are given as,

$$\hat{\Lambda} = \begin{pmatrix} \hat{V} - \hat{a} & 0 \\ 0 & \hat{V} + \hat{a} \end{pmatrix} \quad \text{Eq 3-19}$$

$$\hat{R} = \begin{pmatrix} 1 & 1 \\ \hat{V} - \hat{a} & \hat{V} + \hat{a} \end{pmatrix} \quad \text{Eq 3-20}$$

$$\hat{L} = \frac{1}{2\hat{a}}\begin{pmatrix} \hat{V} + \hat{a} & -1 \\ -\hat{V} + \hat{a} & 1 \end{pmatrix} \quad \text{Eq 3-21}$$

Where $a=(gA/T)^{1/2}$. The quantities denoted with a hat are known as Roe averages. Roe average for velocity is found by Eq 3-22

$$\hat{V} = \frac{\sqrt{A_L}V_L + \sqrt{A_R}V_R}{\sqrt{A_L} + \sqrt{A_R}} \quad \text{Eq 3-22}$$

ΔU within Eq 3-17 is evaluated as

$$\Delta\hat{U} = \begin{pmatrix} \frac{1}{2}\left(\Delta h - \frac{\hat{h}\Delta u}{\hat{a}}\right) \\ 0 \\ \frac{1}{2}\left(\Delta h + \frac{\hat{h}\Delta u}{\hat{a}}\right) \end{pmatrix} \quad \text{Eq 3-23}$$

3.3.2.3 Evolution in Time

The extrapolated boundary values can then be used to evolve the solution to t+Δt according to the following MUSCL-Hancock formula

$$\bar{U}_j = U_j + \frac{1}{2}\frac{\Delta t}{\Delta x}\left[F(U)_j^L - F(U)_j^R\right] \quad \text{Eq 3-24}$$

Note that the evolution step is contained entirely within each cell as the intercell fluxes are evaluated with the boundary extrapolated values of each cell. At each intercell position j+1/2 there are two distinct fluxes namely $F(U_j^R)$ and $F(U_{j+1}^L)$ (Toro 2009).

As there are many variations to the MUSCL-Hancock method an appropriate one must be selected. It is documented that Green's theorem should be used (Bradford & Sanders 2002b) which yields the exact formula by integrating the governing equations with space and time over the control volume (Toro 2006). This yields

$$U_j^{k+1} = U_j^k - \frac{\Delta t}{\Delta x}\left[F_{j+\frac{1}{2}} - F_{j-\frac{1}{2}}\right] + \Delta t S_i \quad \text{Eq 3-25}$$

Comparing this to the corrector step formula of (Bradford & Sanders 2002b) a direct relevance can be seen with the addition of some extra source terms for bed slope and friction.

$$U_j^{k+1} = U_j^k - \frac{\Delta t}{\Delta x}\left(F_{j+1/2}^{k+1/2} - F_{j-1/2}^{k+1/2}\right) + \Delta t(S_i)_j^{k+1/2} + \frac{\Delta t}{2}\left((S_S)_j^k + (S_S)_j^{k+1}\right) \quad \text{Eq 3-26}$$

Where

$$S_j = \begin{pmatrix} 0 \\ F_c \end{pmatrix} \quad \text{Eq 3-27}$$

And

$$S_s = \begin{pmatrix} 0 \\ gA(S_{o-}S_f) \end{pmatrix} \quad \text{Eq 3-28}$$

3.3.3 Slope Limiters

According to Godunov's theorem spurious oscillations will occur where strong gradients are located. To avoid this, slopes Δ_j are replaced by limited slopes $\overline{\Delta}_j$. This extension to non-linear systems is somewhat empirical but will lead to satisfactory results (Toro 2009). This fits into the prediction step of the Finite Volume Method as it alters the ΔV and Δh terms before the initial prediction takes place. This is why the $\overline{\Delta V}$ and $\overline{\Delta h}$ are cell limited in the prediction step.

Slope limiters are used to calculate cell-average gradients of the variables to preserve monotonicity of the solution at discontinuities (Bradford & Sanders 2002a). The limiters become first-order accurate at solution extrema in order to suppress numerical oscillations (Bradford and Sanders 2002b). The choice of limiters becomes an issue of accuracy and in

many journals the Superbee limiter is chosen. The Superbee limiter may be the best with poorly resolved grids but in some situations it would have poor performance (Bradford & Sanders 2006). (Bradford & Sanders 2006) make a strong case for the use of Double Minimum Modulus over the other limiters examined. The accuracy of Double Minimum Modulus predictions was in nearly all cases superior to the other predictions (Bradford & Sanders 2006). The Double MinMod scheme is presented from (Bradford & Sanders 2006).

Double Minimum Modulus $$R(r) = \max\left[0, \max\left(1 \quad \frac{4\mathrm{r}}{\mathrm{r}+1} \quad \frac{4}{\mathrm{r}+1}\right)\right]$$ Eq 3-29

Where $$r = \frac{\delta c_j^+}{\delta c_j^-}$$ Eq 3-30

Where $$\delta c_j^- = c_j^n - c_{j-1}^n$$ Eq 3-31

And $$\delta c_j^+ = c_{j+1}^n - c_j^n$$ Eq 3-32

Where c represents concentration of a scalar moving in a fluid with velocity u and v in the x and y directions, respectively (Bradford & Sanders 2006).

3.4 Alterations of the FVM

There are several alterations that need to occur to the FVM so then it is stable during certain events. This includes alterations for dry bed treatment and alterations for irregular bed profiles.

3.4.1 Dry bed treatment

To handle dry bed, several checks must be added to the program. (Bradford and Sanders 2002b) recommends the following changes. "First, in the event that a depth less than εh is computed, the velocity and discharge are set to zero in the cell. Second, in the event that the extrapolated depth at a cell face (hL or hR) is less than zero, it should be set to zero prior to applying the numerical flux function. Third, in the event that the depth on both sides of a cell face are less than εh, the mass and momentum fluxes for the face should be set to zero, and the flux function need not be called. Note: the mass conservation properties are not a function of εh. That is, εh can be large and *O*(10-14) mass conservation will still be achieved". The MUSCL reconstruction must also be altered to prevent leakage into the surrounding cells (Bradford and Sanders 2002a). Where a wet cell the cell is bounded by a partially wet cell, h should be extrapolated from the wet neighbour (Bradford and Sanders 2002a). This is because when the height of water in the partially wetted cell is averaged over the whole cell, not just the wet area of cell, it becomes a layer of water that is much thinner than the real layer. Another alteration to the MUSCL process is in a partially wetted cell, the height extrapolation is done from the wet side only (Bradford and Sanders 2002a). This prevents wave propagation that is faster than realistic values.

3.4.2 Irregular Bed Treatment

In our particular model irregular bed will not be treated. Channels will take standard geometries, but this topic still deserves a brief mention. It is claimed that any geometry can be modelled with this FVM technique as long as a grid of any sort can be fitted to the terrain. Furthermore there needs to be a height of water to cross sectional area relationship for every component of the governing equations to be found. On top of this, there needs to be a way to calculate hydraulic radius for friction terms within the equation. For real applications it would require surveyed data of the channel to determine accurate wetted perimeters. In the past it was impossible to find the flux invariants for irregular shapes and thus the solution was limited to standard geometric shapes that didn't change spatially. For this shape, an equation could be derived for the flux invariants but would only be useable for that particular model. When (Roe 1981) found an approximate method for solving arbitrary Riemann invariants, which was accurate enough to use with the shallow water equations, it became feasible to solve for arbitrary topography.

3.4.3 Boundary Conditions and the Ghost Cell

For the Godunov scheme to be applied over the entire domain the boundary conditions are placed within ghost cells. These ghost cells are immediately outside of the model boundary

and contain values that are extrapolated from within the model as seen in Figure 3.11 and Figure 3.12. Because the ghost cells are set up the same as a conventional cell, momentum and mass fluxes can be determined via the normal procedure applied to the rest of the cells. An outflow ghost cell is specified only from extrapolated values from inside the model. An inflow boundary needs to be set up in different ways depending on if it is a supercritical or subcritical flow. This is detailed in Chapter 2.4.2.

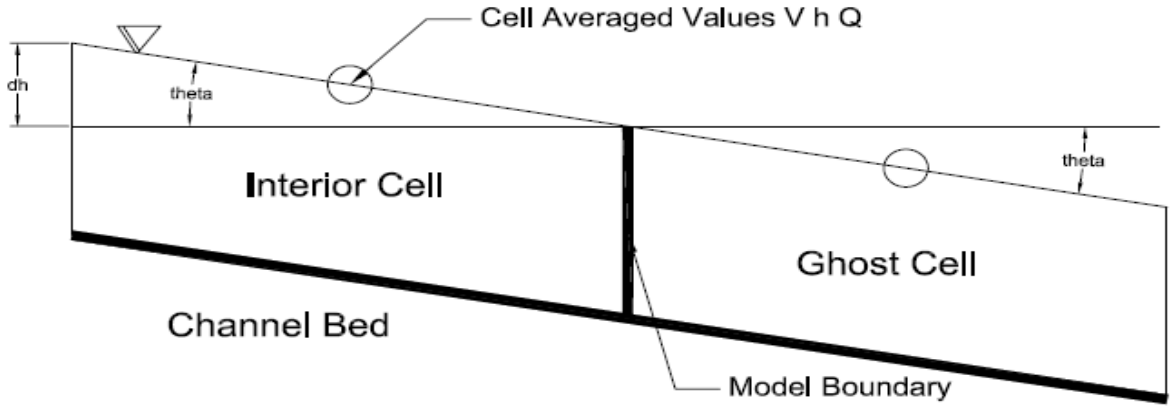


Figure 3.11 Boundary Conditions and the Ghost Cell

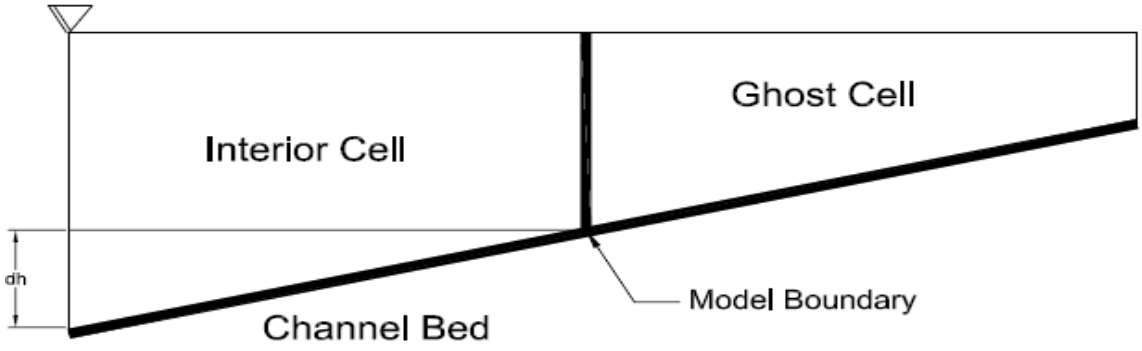


Figure 3.12 Uphill Ghost Cell

3.4.4 Solution Stability

The solution is said stable to be stable if the Courant number is below one (Bradford and Sanders (2002a); Bradford and Sanders (2002b); Toro(2009); Guinot(2003)). The Courant number is defined as Eq 3-33.

$$Cr = \alpha \frac{dt}{dx} \qquad \text{Eq 3-33}$$

It can be seen that for faster flow rates the solution can be kept stable by decreasing the time step or increasing the cell sizes.

4. Existing MatLab Models

4.1 Introduction

An existing MatLab model is available from Professor Brett Sanders. This model is provided to his masters students as a homework project with the task of changing it to dry bed. This model is one dimensional and is simplistic as it is predicting planar flows with no friction, no bed slope and the boundary conditions have been set to solid walls. This model uses an abbreviated version of the shallow water equations with no source terms. This means that friction and bed slope has been neglected.

This then leaves several areas that would be desirable to change to make the model into something that is much more realistic. These changes will be detailed at the end of this chapter.

It is now a good opportunity to introduce some more definitions to help the reader understand the terminology that will be used.

4.1.1 Accuracy

Accuracy refers to the how close to real data, theoretical results, or experimental data that the model can predict. This accuracy can be with two main dimensions. These dimensions include the physical properties of the water flow and the accuracy throughout time. This dimension is measurable through the Mannings equation where the height to flow rate relationship can be examined knowing characteristics of the model, i.e. bedslope and Mannings n. This examines if the number achieved is realistic. The other dimension that the model can be inaccurate with is time. This looks at how long the model has taken to achieve steady state or how long a wave has taken to propagate downstream. This dimension is only comparable against data from a known source or experiment. This dissertation has only set out to measure against the theoretically obtainable results.

4.1.2 Robustness

Robustness refers to the set of circumstances that the model can successfully function with. The more circumstances that the model can function with, the more robust the model is. Robustness can be much more sensitive to particular parameters than to others and to test it fully, these must be found.

4.2 Potential Applications for the Model

The program has no real applications other than modeling a dam break wave over a flat bed, with no friction, in a fixed boundary channel. Sanders freely states, that this program would be completely useless in any real life situation.

4.3 Program (FVM1D) – Original Wet Bed Sanders Model

This program comprises of seven scripts that together fulfill the first five steps of the Godunov algorithm. The sixth step, relating to source terms, has been ignored within this model. The seven scripts are referred to in Table 4.1.

Table 4.1 Location of Scripts

File Name	**Location**
fvm1d.m	Appendix A.1
limiter.m	Appendix A.2
limit.m	Appendix A.3
predictor.m	Appendix A.4
fluxes.m	Appendix A.5
solver.m	Appendix A.6
corrector.m	Appendix A.7

The program has outputs which have not been changed in any way, that help to show the usefulness of the program. Figure 4.1 through to Figure 4.4 are some of the outputs of the program. As the initial Sanders model outputs have no units on the graphs, some housekeeping must be taken care of first to help get an understanding of the graphs. The top sub-plot of the figures is a plot of the free surface profile and energy. This is the height of water in meters on the y-axis and the channel bed is a length in meters. The length on the x-axis holds true for all the subplots. The second sub-plot is the velocity of water movement which has the units of (m/s) on the y axis. The third and bottom subplot is the discharge of water which has the units of (m^3/s) again on the y-axis.

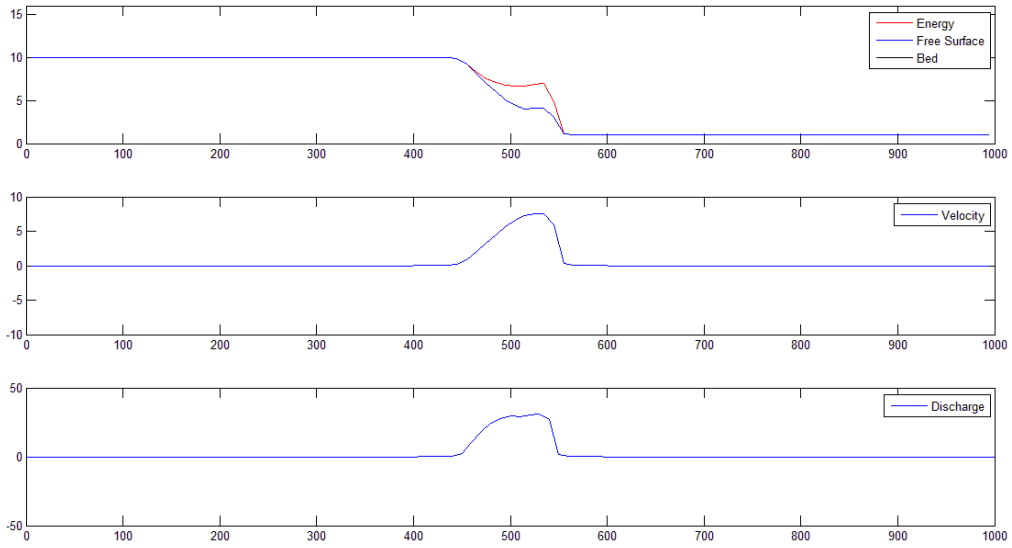


Figure 4.1 Original Sanders model output – Initial conditions

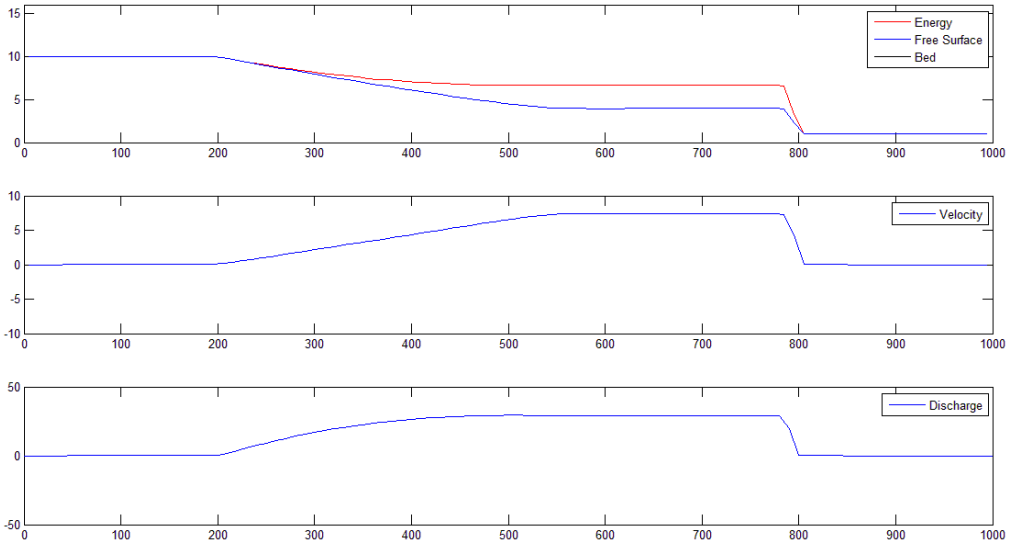


Figure 4.2 Original Sanders model output – Wave propagating

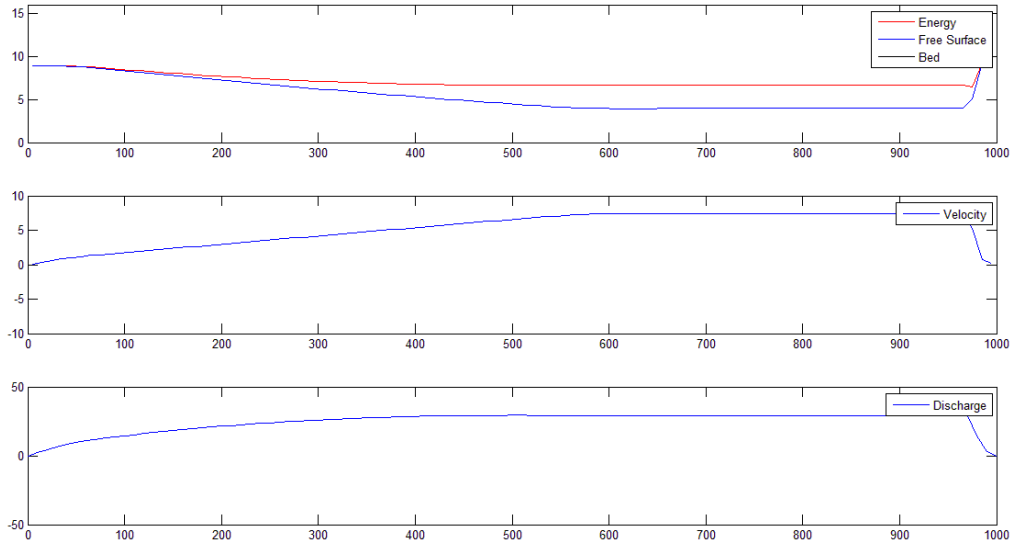


Figure 4.3 Original Sanders model output – Wave reflecting

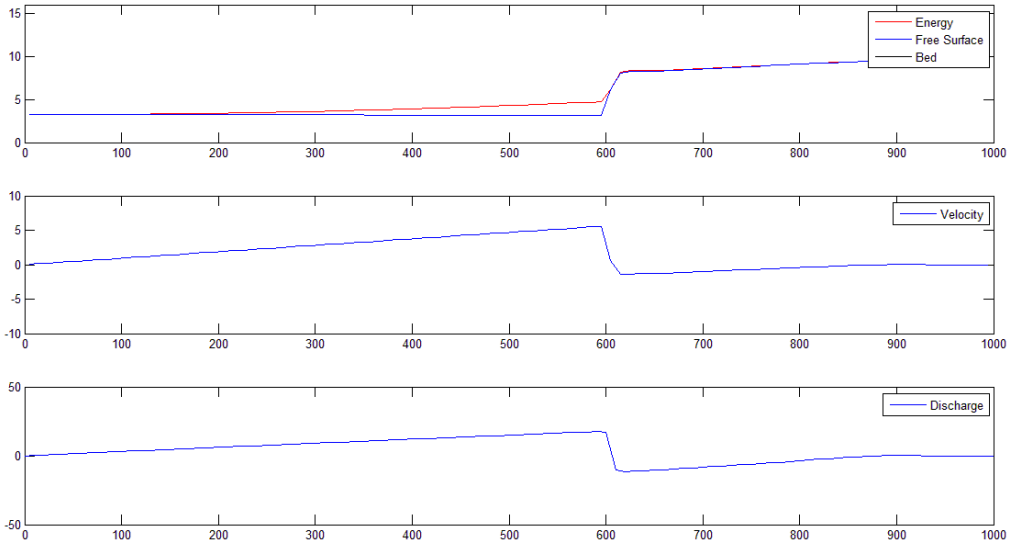


Figure 4.4 Original Sanders model output – Wave reflection

4.3.1 Discussion of Results

The initial conditions of the model have the model starting with a fixed volume of water set up somewhat like a dam that has just failed and the surge wave is propagating over a lake or water surface of some type. Once it hits the end of the model it is reflected back along the channel as if it has hit a solid wall. This model allows for no inflow or outflow, does not allow for dry bed situations, and does not allow for friction. It is hard to judge how accurate or robust the program is at this stage. Further changes will have to be done to the model before a judgment can be made on the accuracy or robustness and will be addressed in a later section in this dissertation.

4.4 Program (FVM1D) – Original Dry Bed Sanders Model

Another script was obtained from Prof. Brett Sanders to deal with dry bed situations. This is shown in Appendix B. Although this script was functioning, it was found that the script would spiral out of control spontaneously for no apparent reason. Also this script, was found to be inaccurate and it was decided that this script would be of little use other than for guidance purposes only on how to deal with dry bed situations. The scripts locations are shown below.

Table 4.2 Location of Scripts

File Name	**Location**
fvm1d.m	Appendix B.1
limiter.m	Appendix B.2
limit.m	Appendix B.3
predictor.m	Appendix B.4
fluxes.m	Appendix B.5
solver.m	Appendix B.6
corrector.m	Appendix B.7
sourceterm.m	Appendix B.8

4.5 Program Coding

Although both codes achieve different results there are a lot of similar characteristics of both. Here we will introduce and discuss the original Sanders wet bed model. All dry bed alterations will be discussed in Chapter 8. The reader is referred to *Appendix C* where the fully commented wet bed code is presented. This commenting is ample for a person who has experience within hydraulic modeling and has read above. Below is a brief explanation of what the code does and the order in which it occurs. It is assumed that the reader has sufficient Matlab knowledge and experience to understand basic Matlab codes and if the reader doesn't have sufficient coding experience the reader is referred to Palm III(2005).

4.5.1 FVM1D.m

This is the main program file. It contains all variables, sets up the initial conditions, calls the function files that follow the predictor corrector scheme proposed within Chapter 3. It also plots the graphs that represent the physical variables of the model. It is provided in Appendix B. Here we step through the code section by section.

Firstly the program defines some variables to set up the model. The definitions of these variables are contained in Table 4.3

Table 4.3 Variable Declaration

Variable	Definition
L	Length in meters of the channel
nc	Number of cells the channel will be divided into
nf	Number of cell faces
dx	Length of each cell
x	An array holding all the edge coordinates along the channel bottom
xc	An array holding all the cell center coordinates along the channel bottom
dt	Time step in seconds
nt	How many time steps will the model run for
ntplot	How often will the graphs be updated with current results
z	Defines the height of each cell edge. This is equivocal to the term So which is common within hydraulics and which will be converted to later
dz	Is the difference in height between two neighbor cells at the cell edges
zc	Computes the height of the bed at each cell center
grav	Gravity acceleration constant at 9.806
iorder	Refers to the order of the finite volume scheme. Here we are only concerned with second order finite volume schemes
beta	Defines which limiter is to be used. Double Minimum Modulus is recommended

Table 4.4 Variable Declaration Cont...

Variable	Definition
xo	Position in meters from the left hand boundary that the wave originates
etalo	Height of water on the left hand side of the wave.
etaro	Height of water on the right hand side of the wave
ulo	Is the initial condition velocity of the water on the left hand side of the wave
uro	Is the initial condition velocity of the water on the right hand side of the wave

These are the variables that are used mainly to set up the initial conditions. Setting up initial conditions is done within a loop which creates in every cell the following terms.

Table 4.5 Initial Conditions

Initial Conditions	Left of wave	Right of wave	Comment
eta	etalo	etaro	Height from datum
h	eta-zc	eta-zc	Height of water
u	ulo	uro	Velocity of water
uh	uh	uh	Flow rate of water

Now that the initial conditions of the model have been set up, the model is ready to start simulating by progressing throughout time. For Matlab to be able to do this we must initialize some arrays by creating a matrix of the correct size. The numerical value of these values in this matrix will not matter because MatLab will be writing directly over the top of them. Thus the Matlab zeros function has been utilized here.

Table 4.6 Initialized Variables

Initialized variable	**Value**
deta	A matrix the same size as eta of zeros
du	A matrix the same size as u of zeros
t	0

The following section is what progresses the model throughout time. It is essentially what implements the Godunov finite volume scheme presented earlier. The following will be called for every time interval.

Table 4.7 Time Stepping Code

1st layer	**2nd layer of program**	**3rd layer of program**	**Comments**
For 2nd order problems only			
Call limiter to find deta			Limiter is a function file that limits the gradients of certain variables
deta = rate of change of eta			
➔	eta=f df=deta		
	df(cell 1)=0		Ghost cell water gradient
	df(nf)=0		Ghost cell water gradient
	For all interior cells		
	Take a forward difference (df1)		
	Take a backward difference (df2)		
	Call the Limit function		The limit function applies a limiting function to the gradient of a variable. Here we will use DoubleMinMod
	➔	If df1*df2 < 0 f=0;	If the sign of the forward difference and backward difference oppose each other. The change in water height is zero

Table 4.8 Time Stepping Code Cont...

1st layer	**2nd layer of program**	**3rd layer of program**	**Comments**
		If not	
		f = the sign of df1 and df2 * the minimum of (2*absolute(df1), (2*absolute(df2), (0.5*2*absolute(df1)+ 2*absolute(df2))	
	←	←	
Call limiter to find du	→		
du=rate of change of eta	Same as deta		
	←		
Call the predictor step			The predictor step makes a prediction on values of velocity height and thus flow rate. These values are then used to find the flux terms which are required within the corrector step
→	dh=deta-dz;		Difference in height can be found from deta and dz
	etap		For every cell apredicted eta can be calculated
	up(i)		A predicted velocity can also be found
hp=etap-zc			

4.9 Time Stepping Code Cont...

1st layer	**2nd layer of program**	**3rd layer of program**	**Comments**
S = grav*hp.*dz/dx;			
Call fluxes function			
➔	Sn=0 cn=1		For a 1D program. Note V is for the second dimension
	Vl= 0 Vr=0		
	hr(1)		Boundary conditions
	ur(1)		Boundary conditions
	Call Solver		Roe Averages for hr and ur
	➔	Hhat	
		Uhat	
		Chat	
		dU	
		R	
		$\hat{\Lambda}$	
		F_L	
		F_R	
		F	
		amax	
	F(1)	➔	
	hl(nf)		
	ul(nf)		
	Call solver		Roe Averages for hr and ur
	➔		Same as hl and ul
		➔	
	F(nf)		

Table 4.10 Time Stepping Code Cont...

1st layer	**2nd layer of program**	**3rd layer of program**	**Comments**
			For all internal cells
	hl		
	ul		
	hr		
	ur		
	Call solver		
	➔		
	□	←	
	F1		Fluxes for cell face 2 to through to nf-1
	amax		celerity
	←		
S			Source term. In the case of the original Sanders model it is for bed slope treatment
Call Corrector function			
➔			For all cells
	h		
	q		
	u		
	←		
eta			
e			
cr			
Plotting			

4.6 Proposed Program Adaptations

To facilitate the changes that are required to make the model more useful, there were several case studies created. These case studies increase in difficulty, to step up the usefulness of the model and as a way to determine the source of errors.

Before any case studies were created, it was necessary to comment the entire Sanders program as there was very little commenting within the program which proved a hindrance in determining what each line of code was doing. After this task was completed, the case studies were addressed.

The model needs many adaptations to become useful for the application required. These adaptations are introduced here.

1. Boundary Conditions – Changed from reflective to free flowing conditions in Chapter 5
2. Friction and Bed slope – Introducing friction and bed slope in Chapter 6
3. Channel Geometry – Applying Channel Geometries in Chapter 7
4. Dry Bed – Introduction of dry beds and partially wetted cells in Chapter 8

5. Boundary Condition Case Study

5.1 Introduction

The first case study changes the reflective boundary walls into free flowing inflow and outflow boundary conditions. Other changes will be with initial conditions but these changes will occur in every case study.

5.2 Alterations

The first case study is targeted at turning the enclosed channel into a free flowing channel where both entry and exit boundary condition allow flows. This turns it into something very similar to an exceptionally wide, frictionless, rectangular channel or a planar flow. This alteration is only in the fluxes function file where the boundary cell equations change. The following boundary conditions, shown in Eq 5-1 to Eq 5-4, were implemented.

No inflow q(1)=0; h(1)=h+1/2dh; v(1)=v+1/2dv; Eq 5-1

F(1,1) = q F(1,2)=q^2/h(1)+0.5*grav*h(1)^2

Inflow q=number; h=h+1/2dh; v=q/h; Eq 5-2

F(1,1) = q F(1,2)=q^2/h(1)+0.5*grav*h(1)^2

No outflow q=0; h=h+1/2dh; v= - (v+1/2dv); Eq 5-3

F(nf,1) = q F(nf,2)=q^2/h(nf)+0.5*grav*h(nf)^2

Outflow q=q+1/2dq; h=h+1/2dh; v=v+1/2dv; Eq 5-4

F(nf,1) = q F(nf,2)=q^2/h(nf)+0.5*grav*h(nf)^2

5.3 Results and Discussion

The results from this case study are simplistic, but are an important step in being able to convert this model into a useful irrigation model. Figure 5.1 through to Figure 5.3 show a system which starts with set initial conditions. When the system begins these figures show the faster moving inflow wave propagates over the top of the slower moving initial conditions wave and eventually the system seems to steady state. Conservation of mass and momentum appear in to be operating correctly. The simple Q=AV or in our case Q=hV shows that nothing appears wrong with this system. This system cannot be verified as there are no realistic situations or theoretical equations that apply. This is ok as verification of the models that include friction and bed slope will be enough to verify this case study.

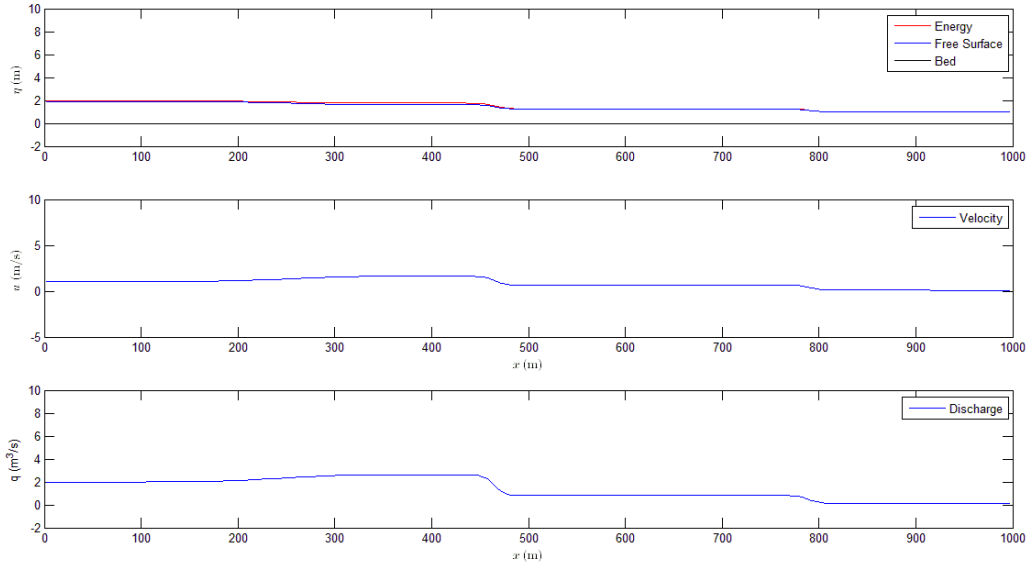


Figure 5.1 Boundary Conditions Test – Part 1

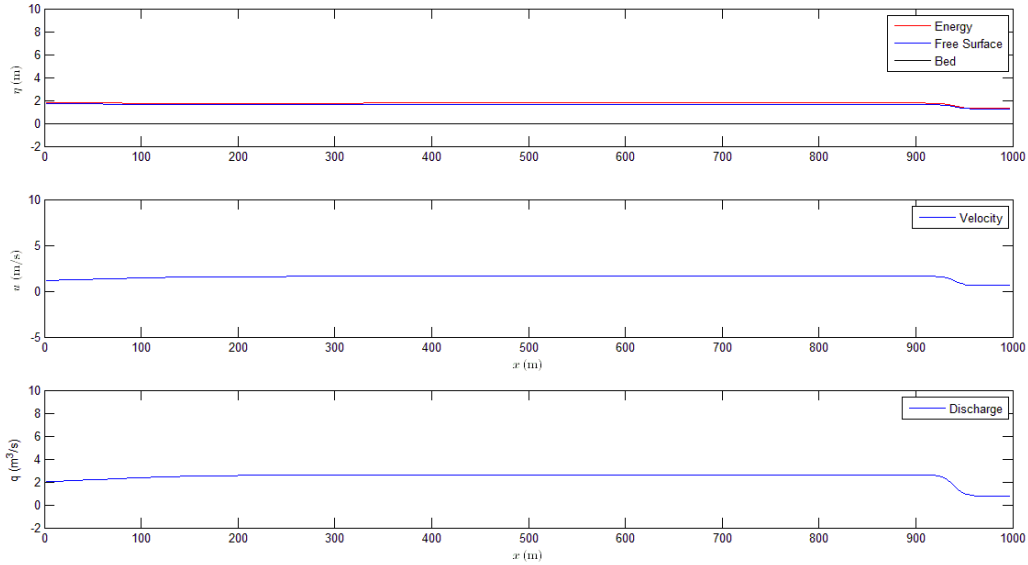


Figure 5.2 Boundary Conditions Test – Part 2

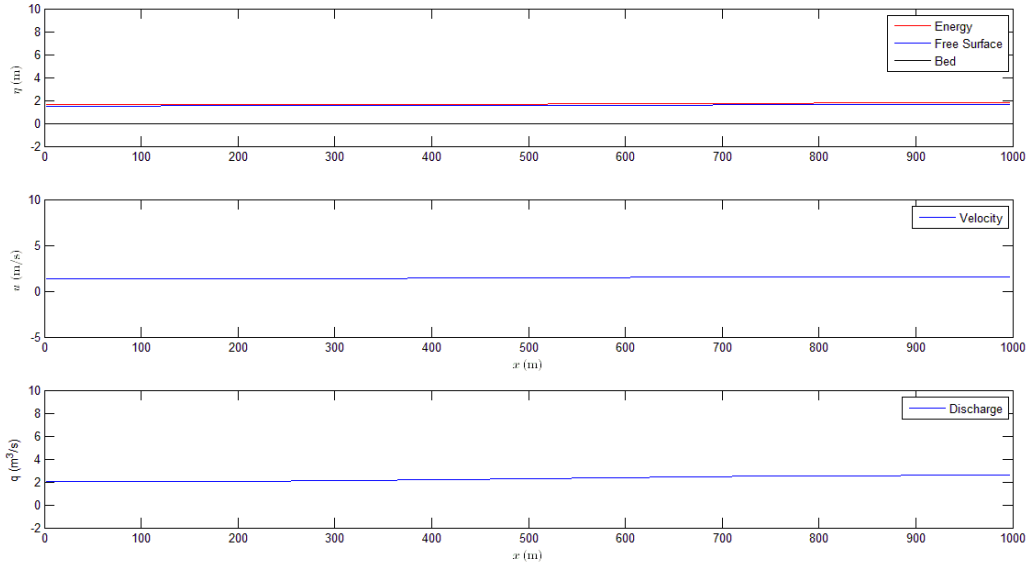


Figure 5.3 Boundary Conditions Test – Part 3

6. Friction and Bed Slope Case Study

6.1 Introduction

The second case study is concerned mainly with introducing friction and bed slope to this planar flow by alteration of the momentum equations. Friction is an important loss to take into account and is widely done in all hydraulics. However within this model care must be taken that the flow height doesn't get too small or it will cause errors because of excessive head losses. This will be examined further within this chapter. To be able to test this, the original Sanders model will have to be altered because it defines the height of flow on either side of the wave and extrapolates that with a gradient of zero to the left and right of the wave up until the boundary conditions. This can be altered by giving an extrapolation angle. In this case an angle of bed slope will be used as the angle of extrapolation

6.2 Alterations

The equations from the original Sanders model were abbreviated versions of the full Saint Venant equations and because of this there are alterations required. The momentum equations need to be altered to include the source terms of friction and dry bed. The abbreviated form of the momentum equations are shown as Eq 6-1 and Eq 6-2.

Predictor $$V_j^{k+1/2} = V_j^k - \frac{\Delta t}{2\Omega}\left(g\overline{\Delta h} + V\overline{\Delta V}\right)_j^k$$ Eq 6-1

Corrector $$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}\left[F_{j+\frac{1}{2}} - F_{j-\frac{1}{2}}\right] + \Delta t S_i$$ Eq 6-2

Where

$$S_i = gh\frac{\partial z}{\partial x}$$ Eq 6-3

To allow for friction and bed slope Eq 6-1 and Eq 6-2 must be changed to Eq 6-4 and Eq 6-5 (Bradford and Sanders 2002b).

Predictor $$V_j^{k+1/2} = V_j^k - \frac{\Delta t}{2\Omega}\left(g\overline{\Delta h} + V\overline{\Delta V}\right)_j^k + \frac{g\Delta t}{2}\left((S_0)_j^k - \frac{1}{2}(S_f)_j^k - \frac{1}{2}(S_f)_j^{k+1/2}\right)$$ Eq 6-4

Corrector $$U_j^{k+1} = U_j^k - \frac{\Delta t}{\Omega}\left(F_{j+1/2}^{k+1/2} - F_{j-1/2}^{k+1/2}\right) + \Delta t(S_i)_j^{k+1/2} + \frac{\Delta t}{2}\left(\left(gA\left(S_o - S_f\right)\right)_j^k + \left(gA\left(S_o - S_f\right)\right)_j^{k+1}\right)$$ Eq 6-5

Where for spatially uniform topography.

$$S_i = 0$$ Eq 6-6

S_o = Bedslope

S_f = Friction losses via the Manning equation

Within Eq 6-5 there is two frictions terms. These friction terms are an average of friction at the current time step and friction at the next time. There is no found documented explanation as to why this is the case. It has been theorized that this is to achieve uniformity within the equation. All other variables have a k+1/2 subscript which translates to t+1/2dt. It could be justified that for a linear relationship an average between k and k+dt would equal k+1/2dt and uniformity in the equations would be achieved.

The next alteration to occur is within the ghost cells. The original Sanders model assumed that the change in the height from the datum, deta, and the change in velocity, du, would be 0 in the ghost cells. For a sloping channel it would be desirable to extrapolate values from the cells immediately inside the channel. This will alter the limiter function as shown in Table 6.1. Within the results section, the before and after effects of the alteration to the limiter will be shown to help show why this change is necessary.

Table 6.1 Limiter Alterations

Limiter function	
Original code	Altered code
`df(1)=0;`	`df(1)=f(2)-f(1);`
`df(nc)=0;`	`df(nc)=f(nc)-f(nc-1);`

These changes equate to a physical change in the ghost cell. See Figure 6.1 for the original code and Figure 6.2 for the altered change.

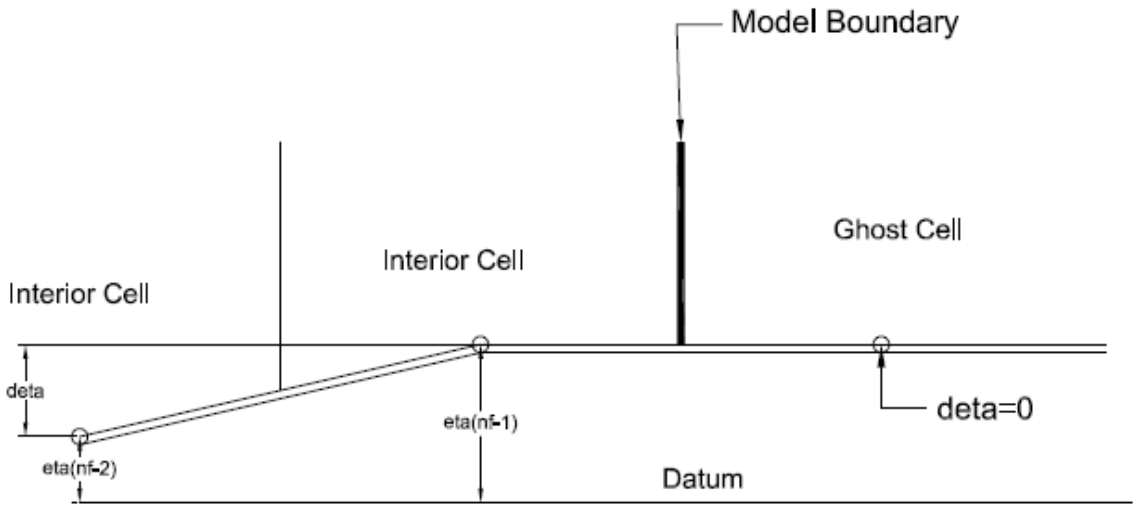


Figure 6.1 Assumption of deta = 0

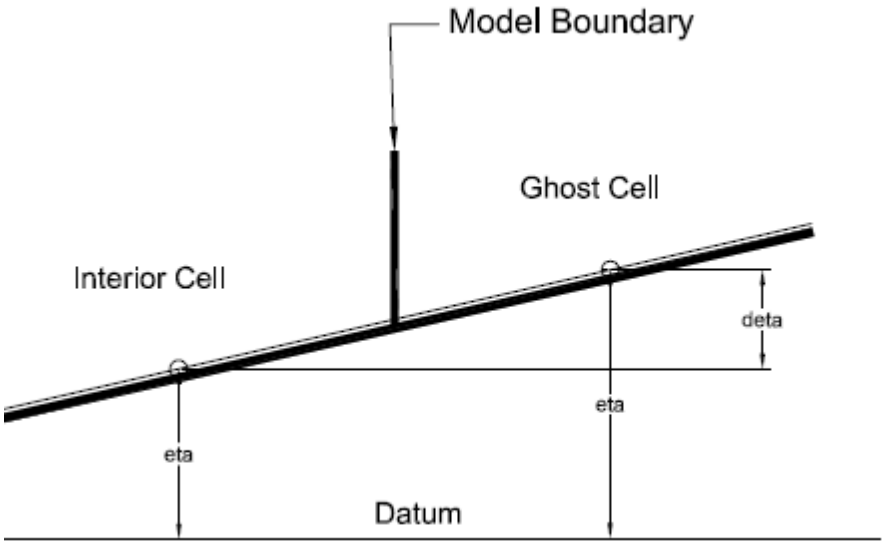


Figure 6.2 Deta Extrapolation

6.3 Results and Discussion

Test runs were done on the model and some of them will be shown here to give examples of how the model performs. These examples represent a sample of the test runs that where done on the model and are a representation of the main situations that can be modelled. A full representation of all the test runs would be of unrealistic size and thus will not be presented here.

For a system with no dry cells the model can handle many situations. The most common wet situation is one of water flowing downhill or of water slowing propagating uphill. Here both scenarios are looked at and the variation of initial conditions has little effect on the accuracy of the system, unless of course, the initial conditions have small heights which yield unrealistic frictional losses (addressed in Chapter 7 and Chapter 8). Another way the initial conditions can hinder the model is if the initial conditions produce very fast flows. This can be done in a variety of ways which include large bed slope angles, small Mannings n values or high flow rates through a small height. A typical downhill system is shown in Figure 6.3 through to Figure 6.7.

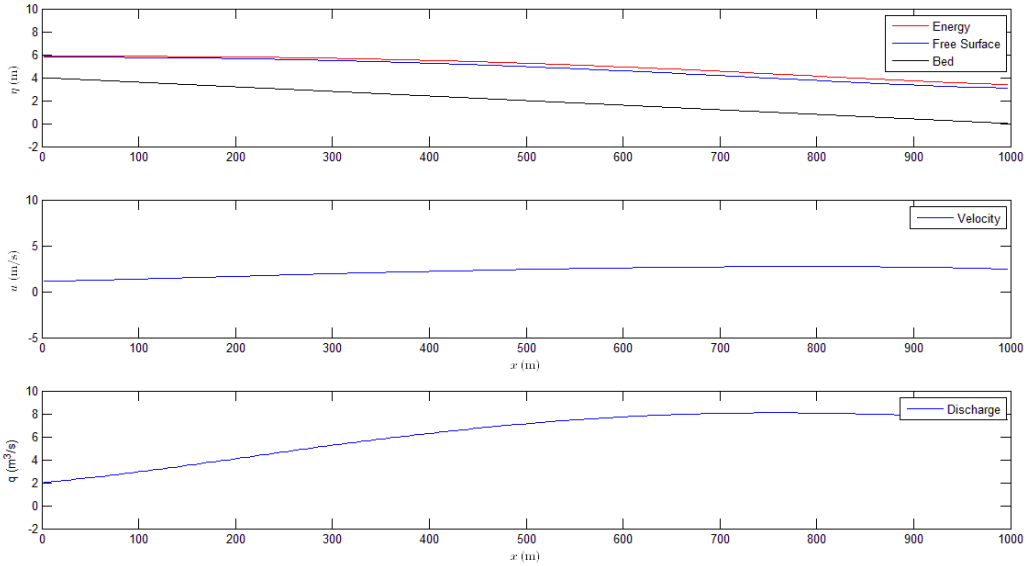


Figure 6.3 Friction and Bed Slope Test 1 – Part 1

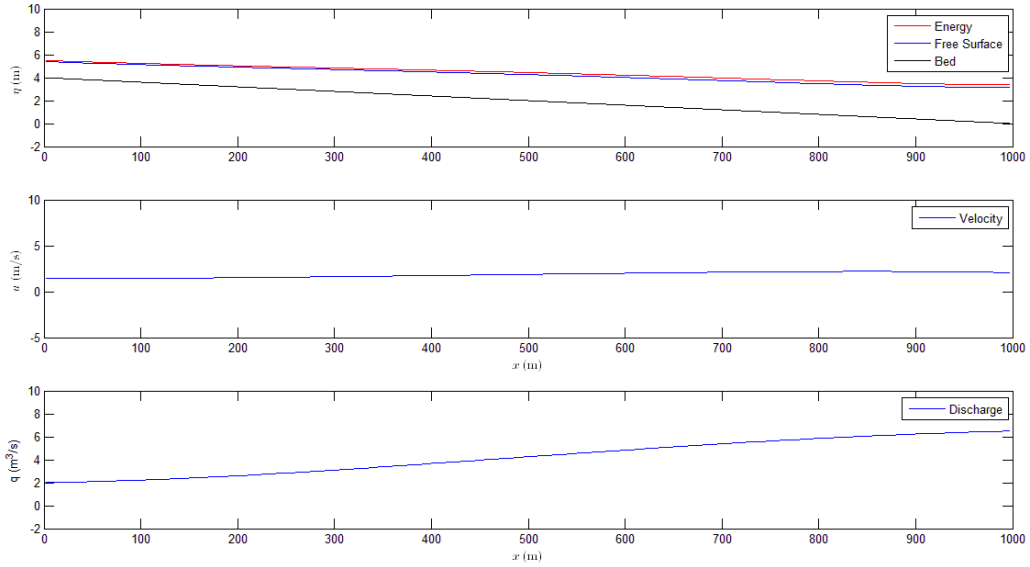


Figure 6.4 Friction and Bed Slope Test 1 – Part 2

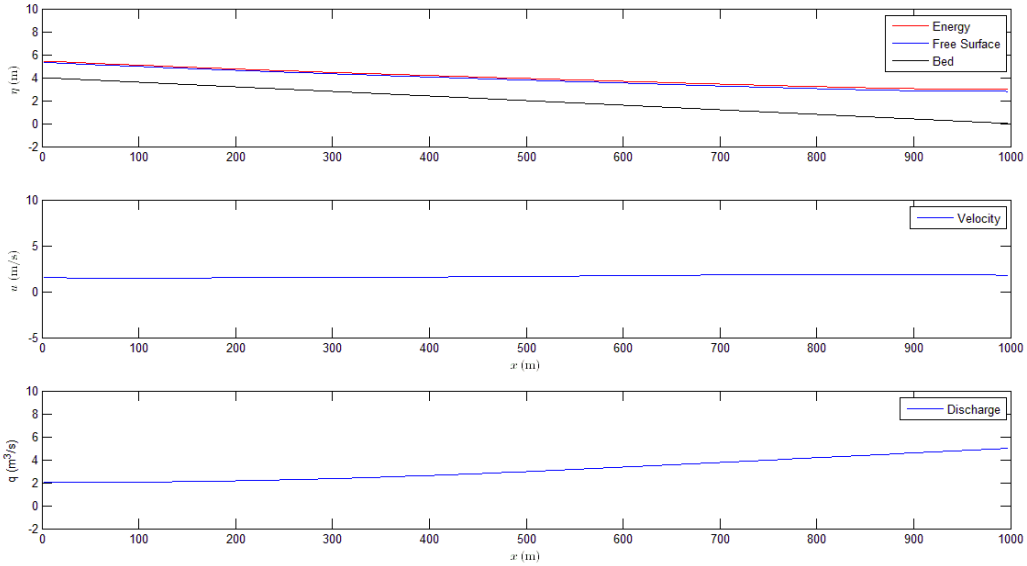


Figure 6.5 Friction and Bed Slope Test 1 – Part 3

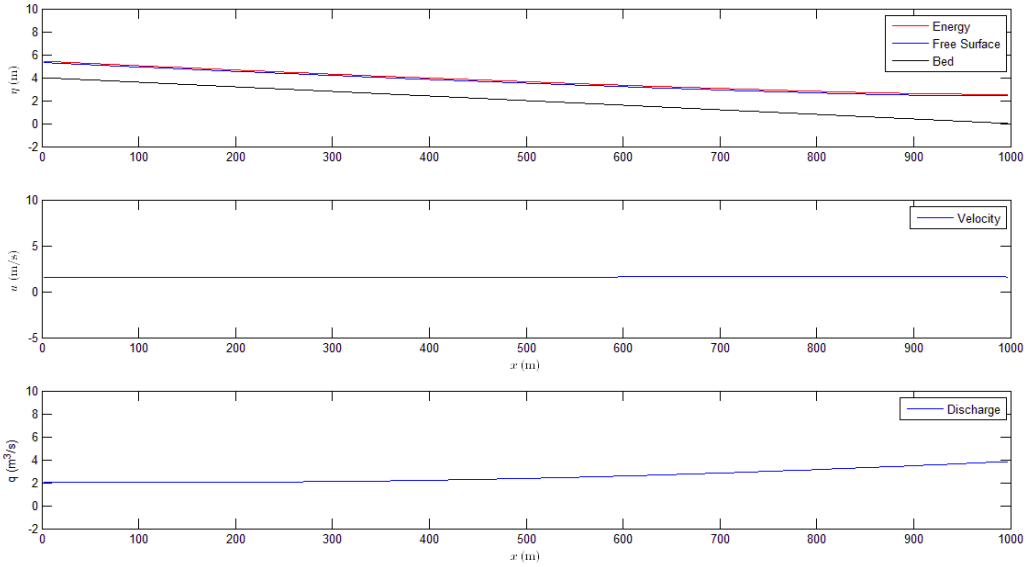


Figure 6.6 Friction and Bed Slope Test 1 – Part 4

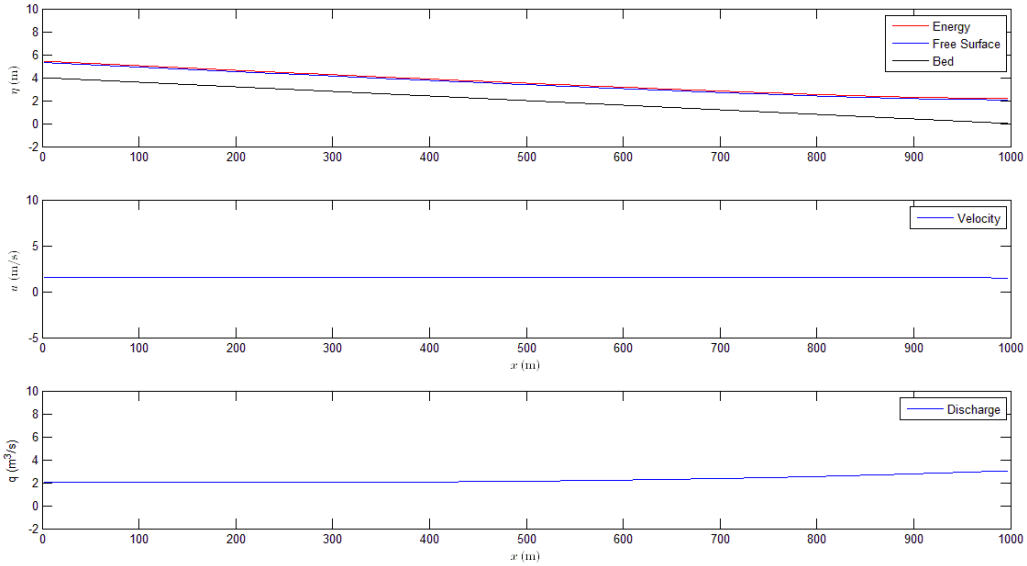


Figure 6.7 Friction and Bed Slope Test 1 – Part 5

Within figures Figure 6.3 through to Figure 6.7 it can seen that a set of initial conditions then flow out of the system and eventually are being replaced by a steady state flow condition. This is realistic in irrigation channels which run over a long period of time and the above system took approximately 30 to 40 minutes to approach a steady state condition which seems realistic for a system of this size and flow rate. This type of system can be verified when it is in the steady state condition by comparing modelled results to those achieved by using the Mannings equation. This is done within Chapter 7.

Next we will look at a pair of uphill scenarios, one with the original limiter function, and the other with the altered limiter function. See Figure 6.8 through to Figure 6.15.

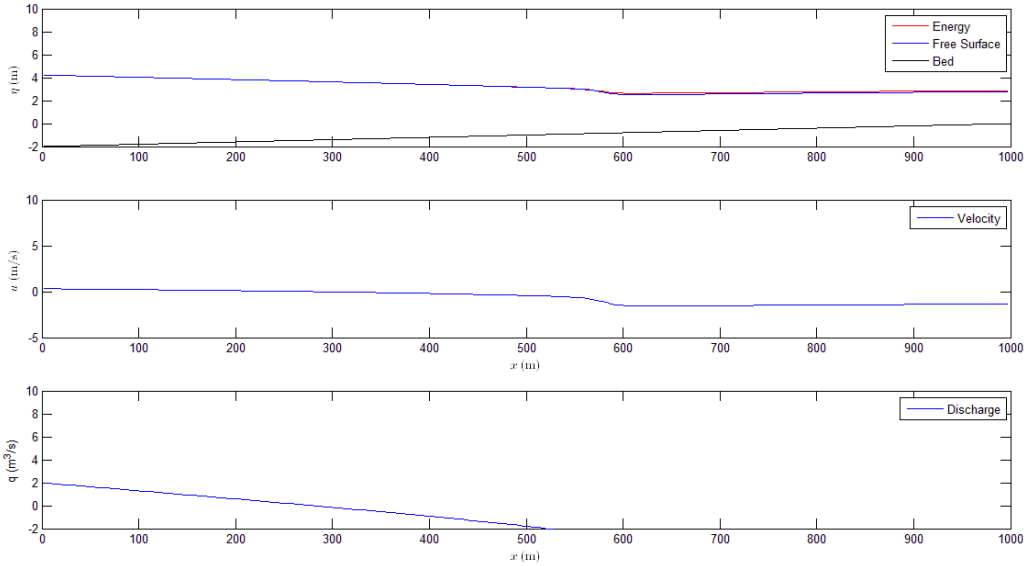


Figure 6.8 Friction and Bed Slope Test 2 – Part 1

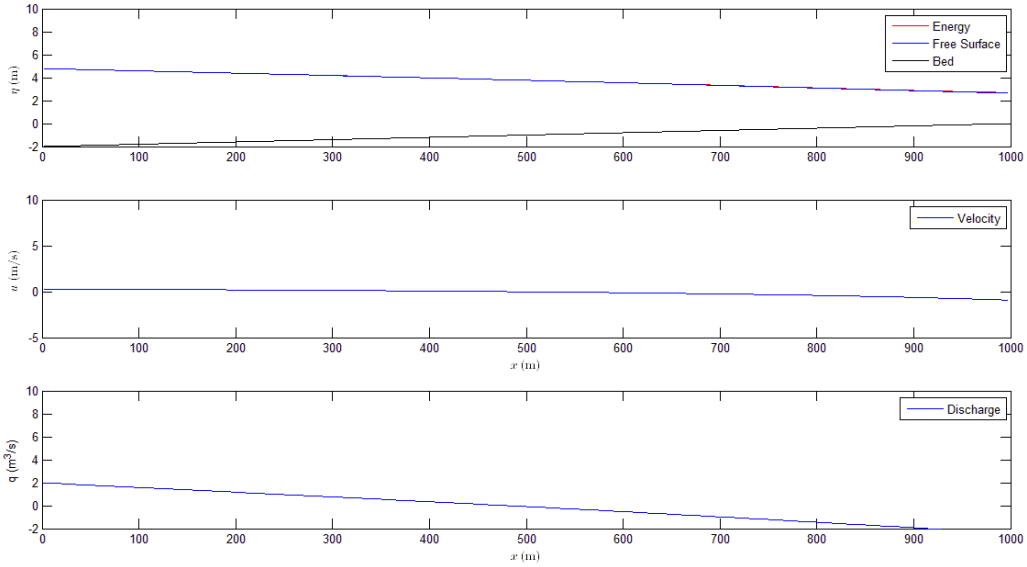


Figure 6.9 Friction and Bed Slope Test 2 – Part 2

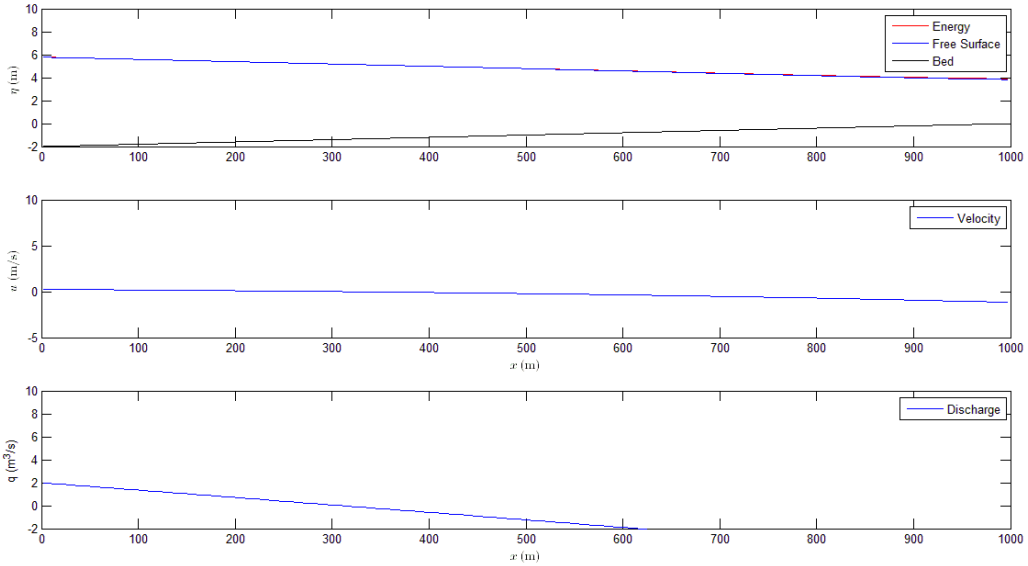


Figure 6.10 Friction and Bed Slope Test 2 – Part 3

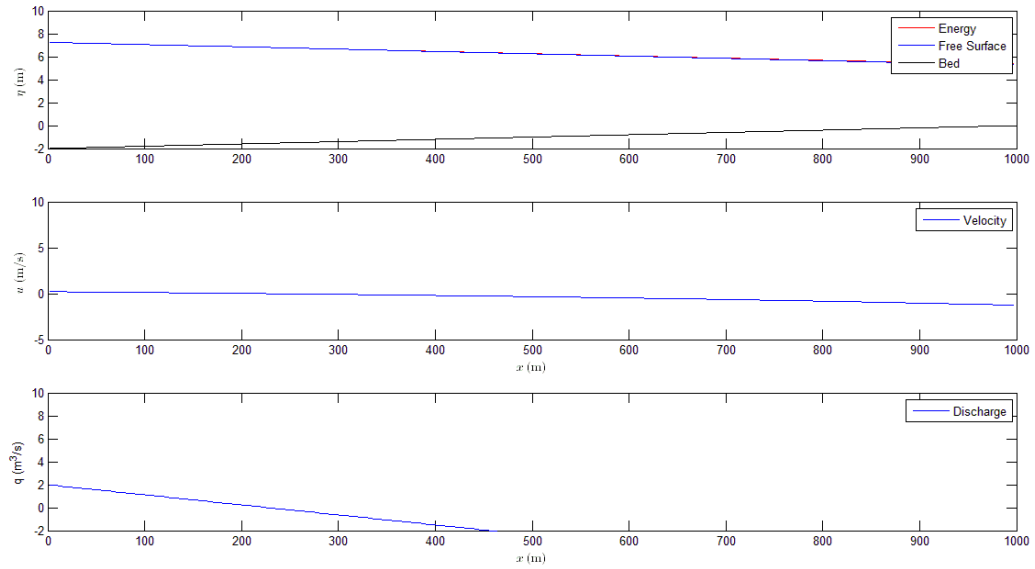


Figure 6.11 Friction and Bed Slope Test 2 – Part 4

Figure 6.8 through to Figure 6.11 show an uphill system before the limiter had been altered. These figures show a set of initial conditions slowly flowing uphill, with the water from the

upstream end of the system flowing back with a negative flow rate. This is ok but the positive flow rate should prevail eventually. Instead with this system the negative flow rate takes over suggesting that the upstream boundary conditions now have inflow conditions rather than outflow conditions. This condition is looked at again in Chapter 8 where what is happening is examined in detail.

Within the next section of this case study we now look at the situation where the limiting function has the boundary conditions altered. This is presented in Figure 6.12 through to Figure 6.15.

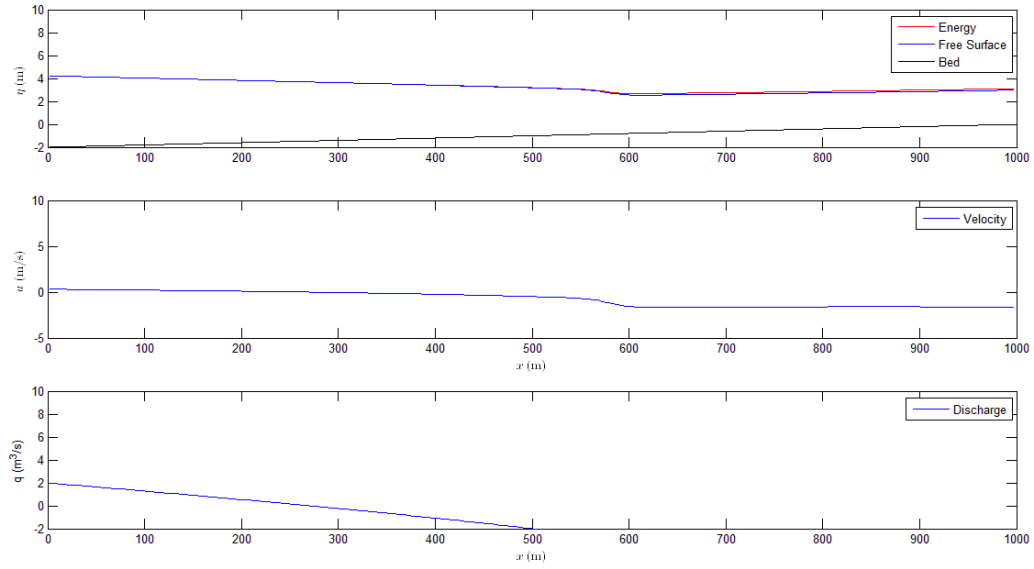


Figure 6.12 Changed Limiter Conditions – Part 1

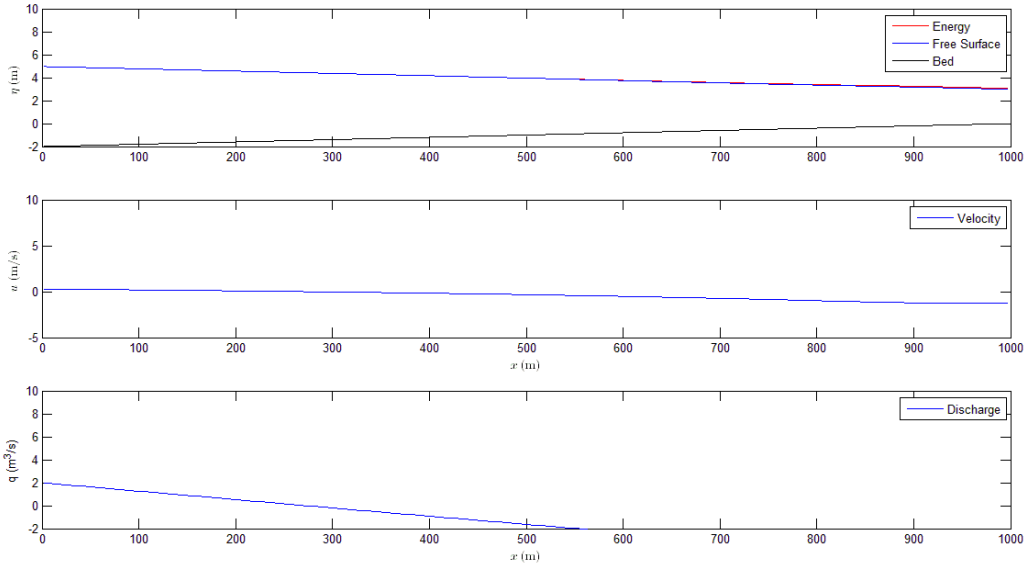


Figure 6.13 Changed Limiter Conditions – Part 2

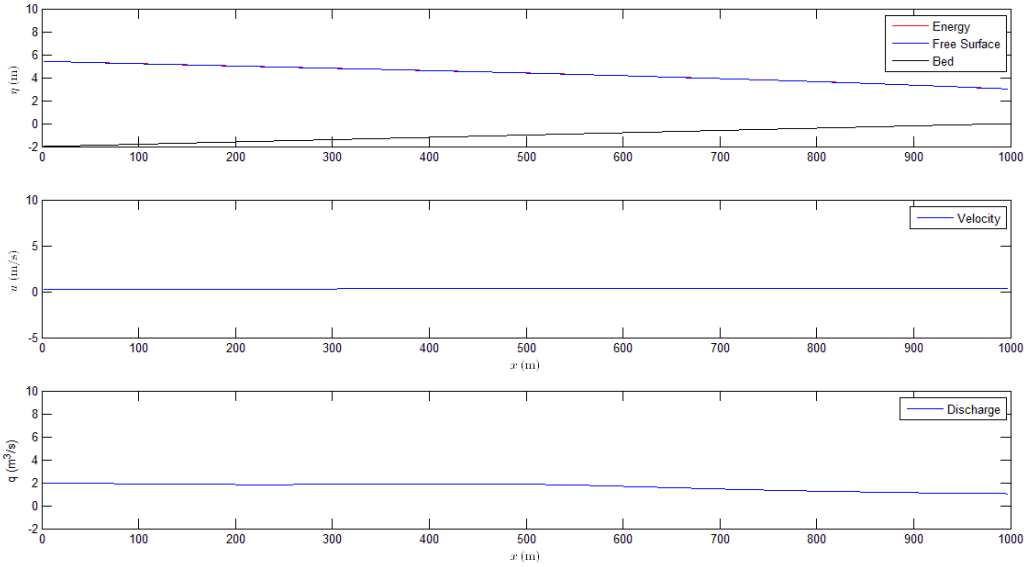


Figure 6.14 Changed Limiter Conditions – Part 3

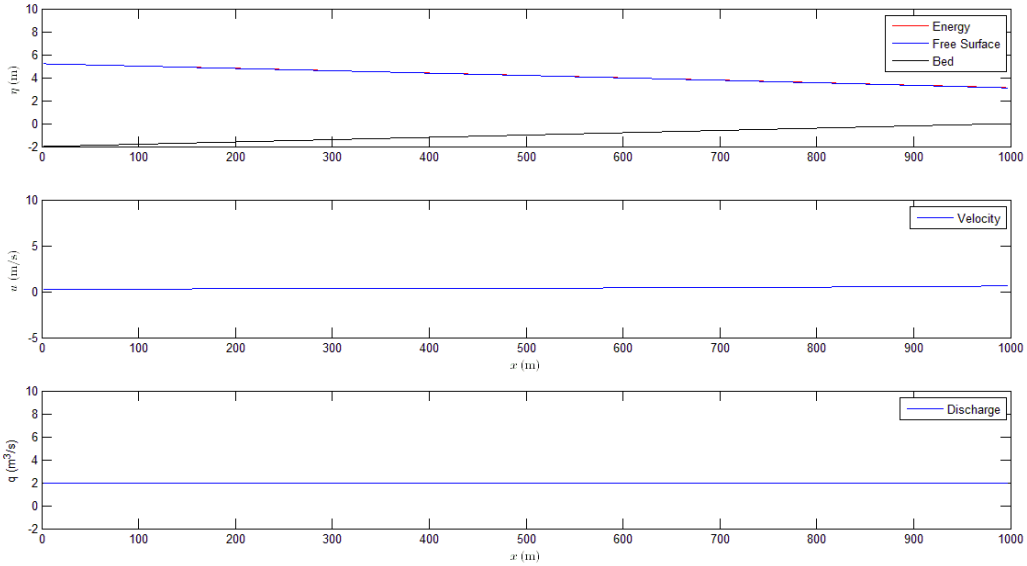


Figure 6.15 Changed Limiter Conditions – Part 4

This section has identical initial conditions as the model examined with the different limiting function. It can be seen here that the inflow at the downstream end eventually takes over the rate of negative flow produced from the bed slope. Eventually the flow rate will reach a steady state. This means that the volume entering the system is the same as the volume exiting the system and thus the volume within the system is not changing. This system cannot be checked using the Mannings equation as the flow profile doesn't approach parallel with the bed slope. It has been assumed that if the system shows no problems, and is appropriately verified within the downhill system, that this uphill system is seen as verified.

7. Channel Geometry Case study

7.1 Introduction

The fourth case study will change the model so then it becomes easy to alter the channel from planar, to rectangular or trapezoidal cross section. In fact, with this model type any geometry could be modelled as long as a height to area/wetted perimeter relationship is known. This generalization will be done by transforming equations previously used with height of the fluid surface into one which uses hydraulic radius.

7.2 Alterations

Until now the model has been operating with planar flows. To alter the model to incorporate geometry, alterations need to be made to the Mannings equation for friction head loss. The alterations turn the height on the bottom of Eq 7-1 to hydraulic radius, which means the side walls of the channel, will be incorporated in the frictional affects. This is shown mathematically in Eq 7-1.

$$S_f = n^2 \frac{|V|V}{h^{\frac{4}{3}}} = n^2 \frac{|V|V}{R^{\frac{4}{3}}} \qquad \text{Eq 7-1}$$

Where $$R = \frac{A}{P}$$ Eq 7-2

Because the height to area relationship is known and the area is directly available from the height it is only necessary to alter the Mannings equation and one term within the corrector step. Using the height instead of area within all other parts of the program is acceptable because there is a direct relationship between height and area. Here will only examine a rectangular channel but it is noted that to change to a trapezoidal channel type is as simple as altering the area, perimeter, and hydraulic radius equations within both the predictor and corrector steps.

7.3 Results and Discussion

The results within this section are very similar to those presented in Chapter 6. Verification is done in the same way where the results are compared to theoretically obtainable results of the Mannings equation. Figure 7.1 through to Figure 7.4 are an example of the results. Further results are examined to verify the results.

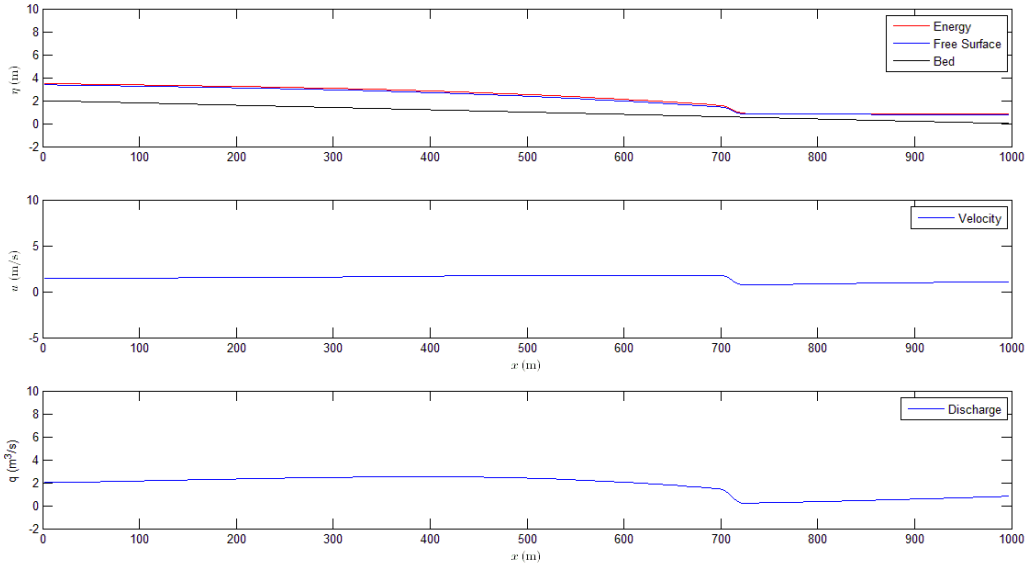


Figure 7.1 Channel Geometry Test – Part 1

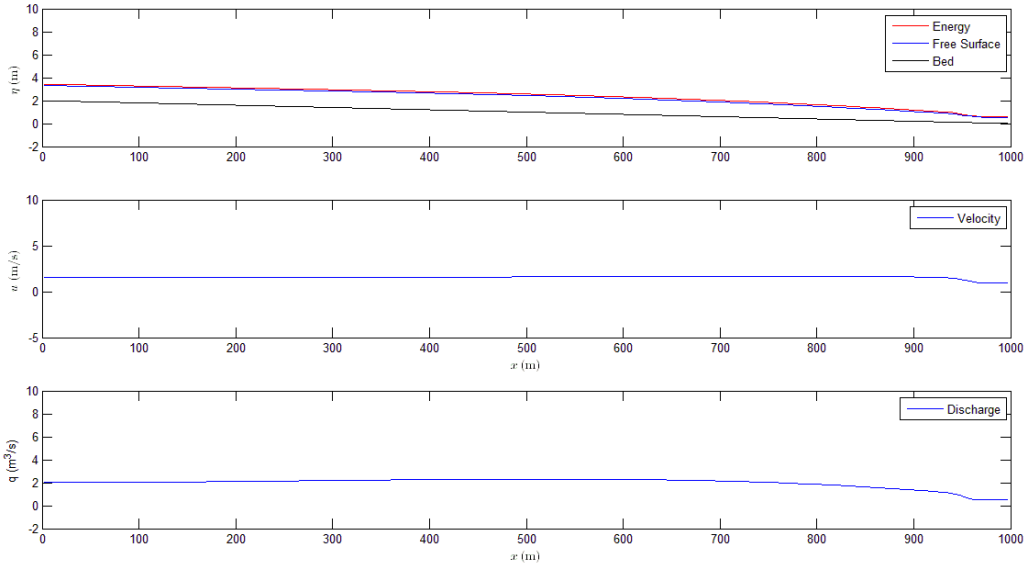


Figure 7.2 Channel Geometry Test – Part 2

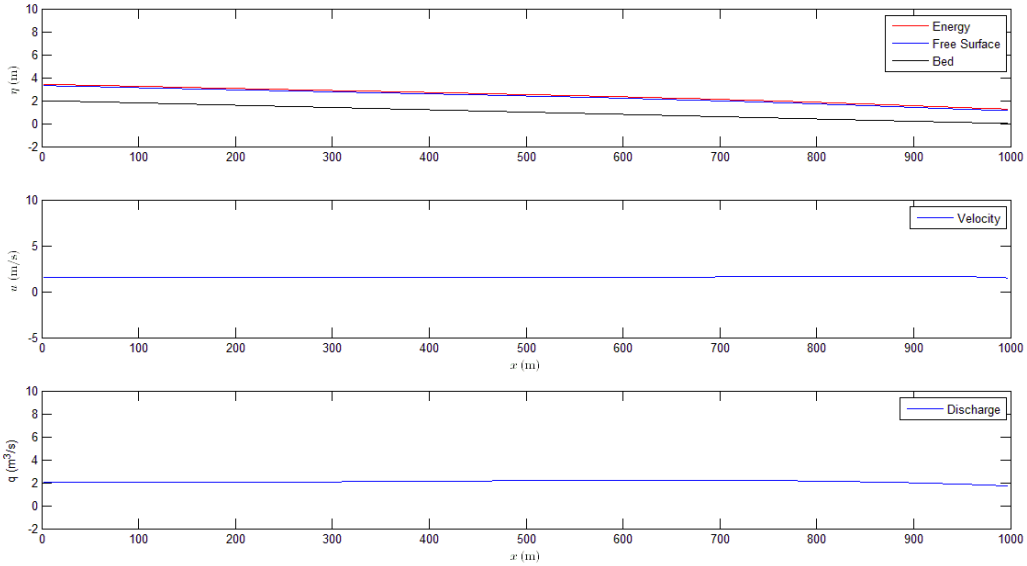


Figure 7.3 Channel Geometry Test – Part 3

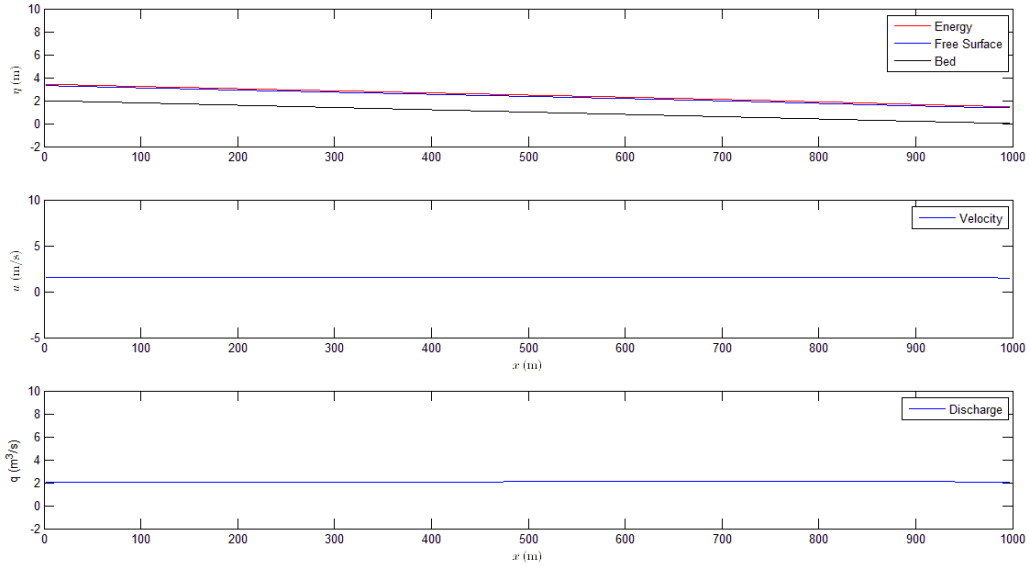


Figure 7.4 Channel Geometry Test – Part 4

Verification of results obtained can be done by comparing the model obtained surface profile height, with known discharge, with the discharge obtained from that height with the Mannings equation. It was found over 9 tests that the average error for comparing systems with different flow rates, different bed slopes, and different Mannings n, was -0.035%. This is a very good result and it is believed that this result could be improved even further if the models were allowed to run for a longer time period. These results are summarised in

Table 7.1

Table 7.1 Mannings Verification

Model Flow rate Q (cumecs)	n	height of water (m)	b(m)	A(m^2)	P(m)	So	Calculated Q	Percentage Error (%)
2.0000	0.0500	1.5564	2.0000	3.1128	5.1128	0.0020	2.0000	0.0000
3.0000	0.0500	2.1606	2.0000	4.3213	6.3213	0.0020	2.9994	-0.0209
4.0000	0.0500	2.7490	2.0000	5.4980	7.4980	0.0020	3.9987	-0.0314
5.0000	0.0500	3.3239	2.0000	6.6477	8.6477	0.0020	4.9896	-0.2078
6.0000	0.0500	3.9039	2.0000	7.8078	9.8078	0.0020	5.9986	-0.0241
2.0000	0.1000	2.3187	2.0000	4.6374	6.6374	0.0030	2.0000	-0.0022
2.0000	0.2000	3.7069	2.0000	7.4137	9.4137	0.0040	1.9993	-0.0329
2.0000	0.0200	0.7765	2.0000	1.5530	3.5530	0.0020	2.0000	-0.0004
2.0000	0.0100	0.6057	2.0000	1.2114	3.2114	0.0010	2.0000	0.0000
							Average error	-0.0355

8. Dry Bed Case Study

8.1 Introduction

The fourth case study is aimed to transform the model to be able to incorporate a dry bed as an initial condition. This will increase the versatility of the model as it would be able to track a wave propagating uphill along a channel, and draining out of a channel. Both are desirable in modelling furrow irrigation systems.

8.2 Alterations

Dry bed alterations have been discussed in Chapter 2.4.1 and Chapter 3.4.1. Here we look at what that means within the program and mathematically. Initially the tolerance that was discussed within Chapter 3.4.1 is introduced into the MatLab model. Once this tolerance is set up, a series of if statements must be set up to determine whether the water is above or below the tolerance. The action taken by the program will then depend on the if statement's outcome. These if statements are situated at various locations within the program and will be discussed further in this section.

Within the initial conditions an alteration needs to occur in setting up partially wet cells. The original Sanders method was programmed such that a cell could never be dry. A cell is examined and the highest edged is deemed zmax, the lowest zmin. See Figure 8.1.

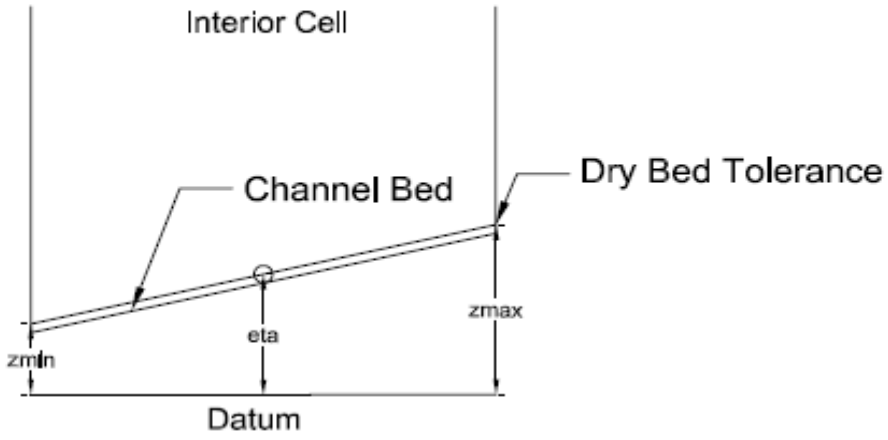


Figure 8.1 Typical Dry Cell

The original model treated the cell as dry if eta was smaller than zmin. However, it is not physically possible for eta to ever be below zmin. Therefore the statement was changed and is described here. The main change is that the height must be examined instead of eta. If the height of water is above zmax, the cell is wet and will be treated the normal way. If the height of the water is below zmin, the cell is dry, thus height is set to zero, velocity is set to zero, and thus the flow rate is set to zero. If the height of water is between these two limits, the cell is deemed to be partially wet and the water level is set with Eq 8-1. If this water level is below the set tolerance the cell is then deemed to be dry and the height and velocity is set to zero.

$$h(j) = 0.5 \times (eta(j) - zmin(j)) \qquad \text{Eq 8-1}$$

Then within the predictor and corrector steps, the area, hydraulic radius, and frictional loss are only found when height of the water is above the threshold. If height is below the threshold, these terms are set to zero before the new velocities and new heights are found. If these new heights are under the threshold, they are set to zero along with the corresponding outflow and velocity. Within the fluxes step, at every cell interface if the height on the left or right side is below the threshold the perpetrating height is set to zero, if both are below the threshold they are both set to zero and the fluxes are not determined.

The codes for this are presented in Appendix E

File Name	**Location**
fvm1d.m	Appendix E.1
limiter.m	Appendix E.2
limit.m	Appendix E.3
predictor.m	Appendix E.4
fluxes.m	Appendix E.5
solver.m	Appendix E.6
corrector.m	Appendix E.7

8.3 Results and Discussion

Firstly a height threshold was set up. If the water height is below this height threshold then the cell is deemed to be dry. It was found that situations with a wave propagating along a downhill slope, with friction, that this threshold didn't affect the accuracy of the program greatly but would stop suddenly for thresholds smaller than the point where the Mannings equation became unstable. This is shown in Figure 8.2 and Figure 8.3. It is possible to step over this by making the dry bed water threshold limit one order higher than the Mannings equations limit.

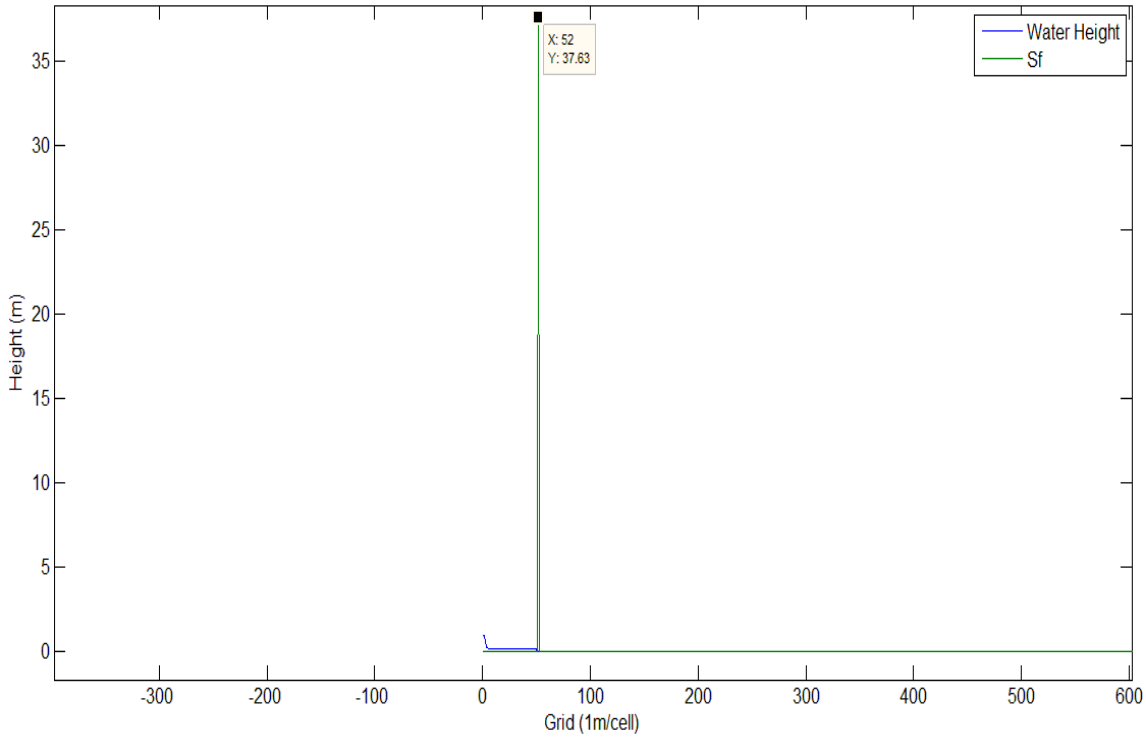


Figure 8.2 Dry Bed Program Mannings Threshold

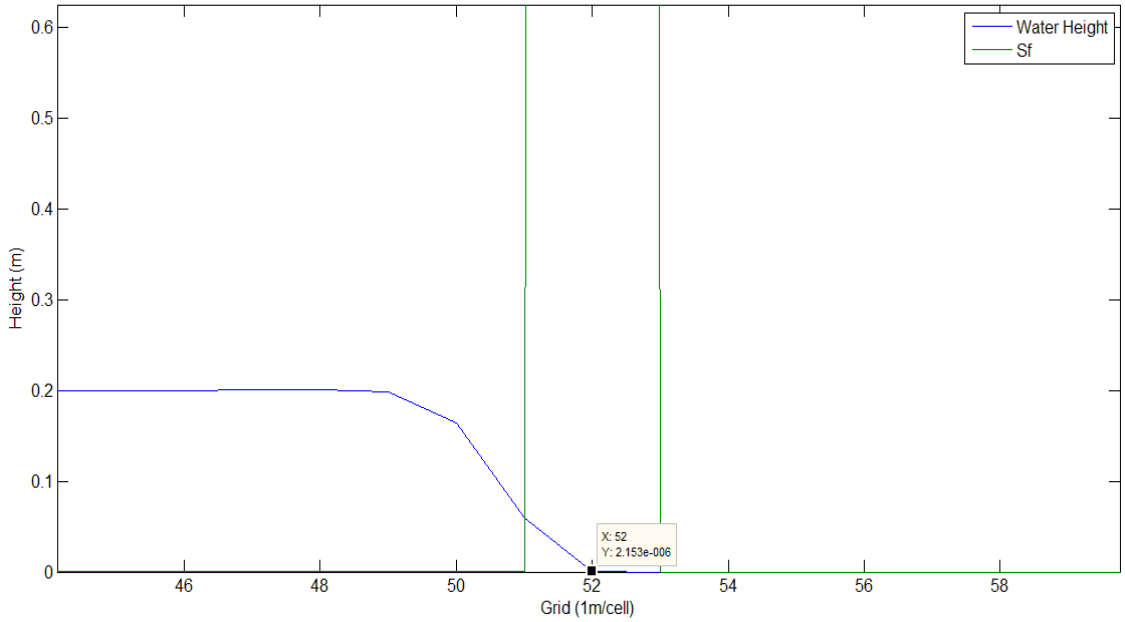


Figure 8.3 Dry Bed Program Mannings Threshold Zoomed

The Mannings limit was examined further by altering the inflow of the water for certain velocities and noting the height the model reads at steady state. It appears that for any height below 0.002m, for this particular circumstance, that the model and the mannings equation do not yield accurate results. So for this particular system the dry bed tolerance should be set to 0.002m to stop any errors accumulating. Other bed slope and Mannings n values would lead to a different tolerance being acceptable.

Model Flow rate Q	n	height of water	b	A	P	So	Calculated Q	Percentage Error
0.00008	0.03	0.001676	2	0.0034	2.0034	0.002	0.0001	-11.9439
0.00006	0.03	0	2	0.0000	2.0000	0.002	0.0000	-100.0000
0.00004	0.03	0	2	0.0000	2.0000	0.002	0.0000	-100.0000
0.00002	0.03	0	2	0.0000	2.0000	0.002	0.0000	-100.0000

Another issue found was water spontaneously appearing near the upstream boundary where the conditions were dry. See Figure 8.4. It is possible to see that within the ghost cell there is an error within the Sanders model. This is an error with the extrapolation within the ghost cell. The extrapolation was using a wrong point to define change in bed slope within the fluxes script which needed to be rectified.

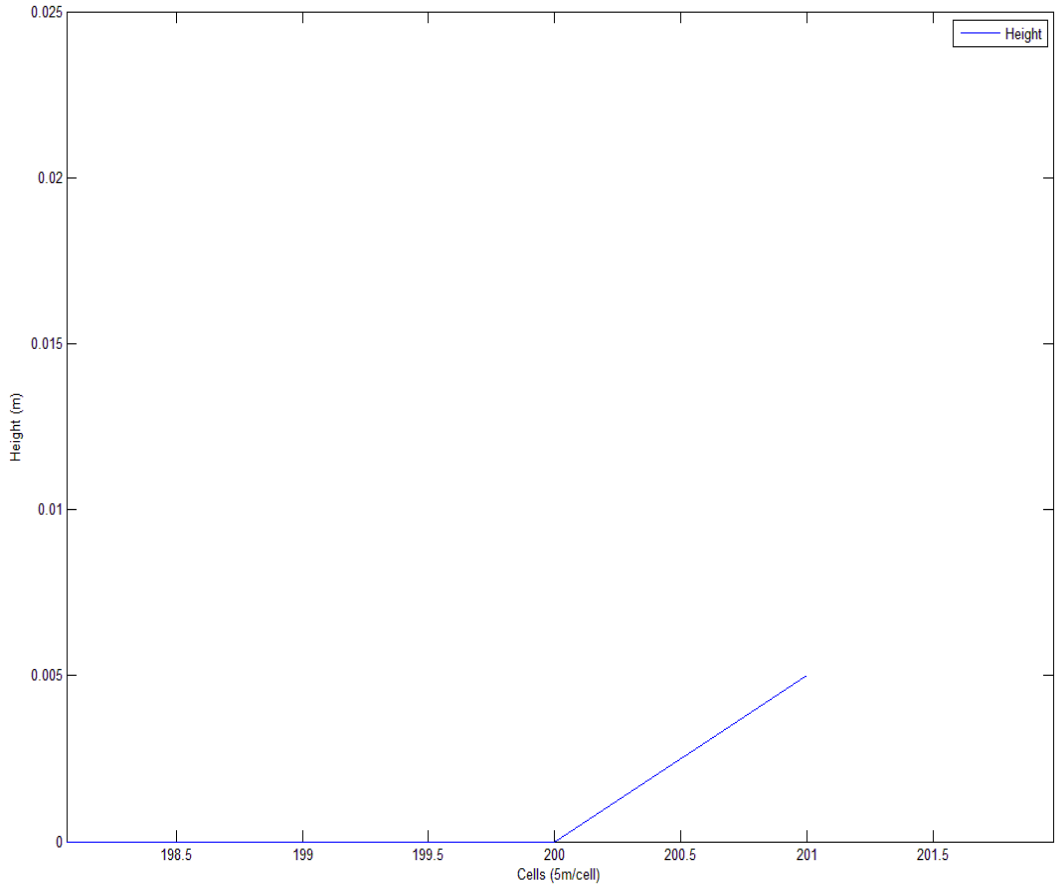


Figure 8.4 Zoomed to Failure Point

Another reason the water was appearing in the ghost cell is because the angle of change in eta over the ghost cell is defined in the Sanders script as zero, see Figure 6.1. This means

that when the angle is extrapolated from an uphill sloped dry cell, momentum and mass fluxes are created. When water appears in the ghost cell the angle of the bed slope allows this water to then drain into the model's cells creating a negative flow rate which adds some mass into the model, which is unaccounted for. This also creates large issues with model stability as the Courant number approaches 0.6. See Figure 8.5 to see that the stability occurs at the downstream end where the velocity is the greatest. These instability issues are largely removed once the issues detailed above were fixed.

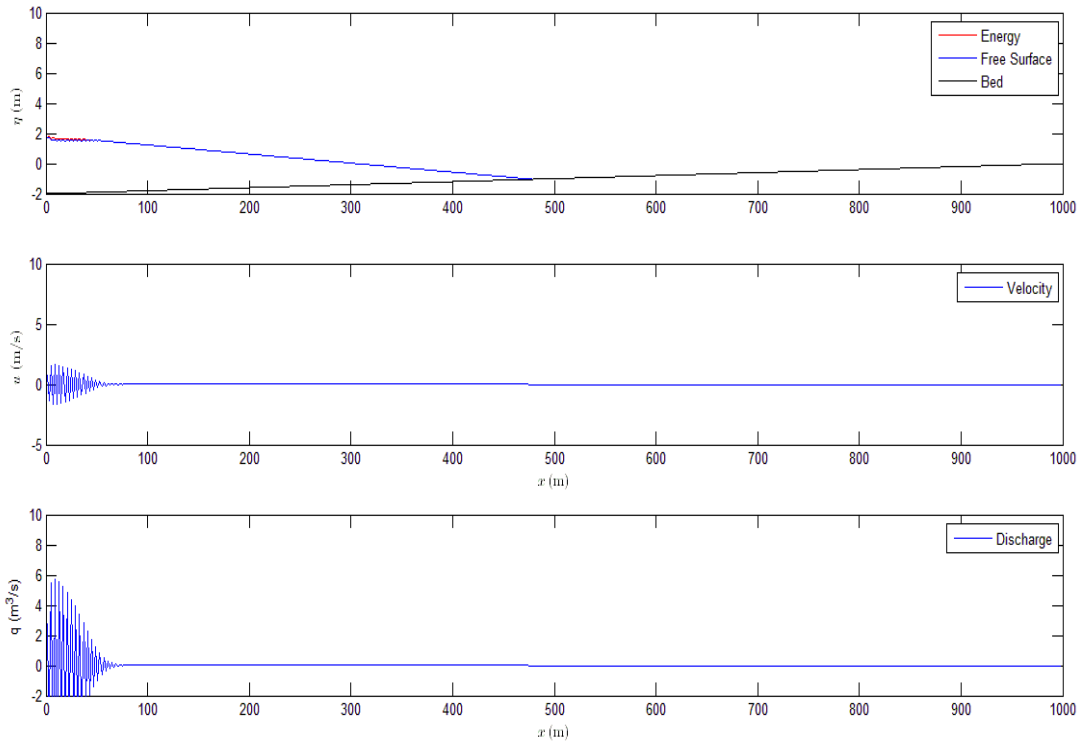


Figure 8.5 Downstream Stability Issues

By setting up the extrapolation angle equal of deta and du to the angle of deta and du respectively from the immediately interior cells, no water is added into the model. This change however causes issues when an uphill, dry bed situation occurs. The issue is faced when the water level reaches the final cell. As seen in Figure 8.6 if the angle of the deta, (height from datum), is extrapolated into the right hand side ghost cell, and the angle of the bed is extrapolated, there will be a negative water level, which triggers the dry bed clause.

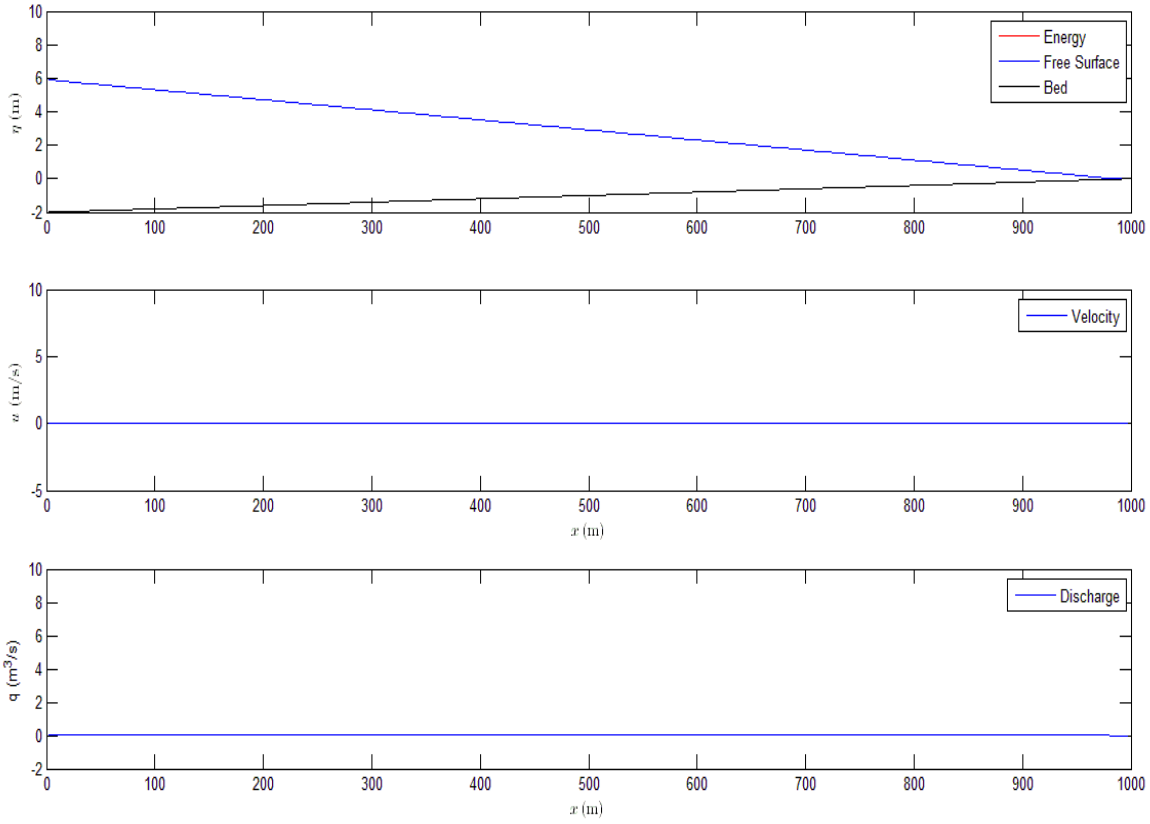


Figure 8.6 Uphill, Dry Bed, Free Flowing Boundary Condition Issues

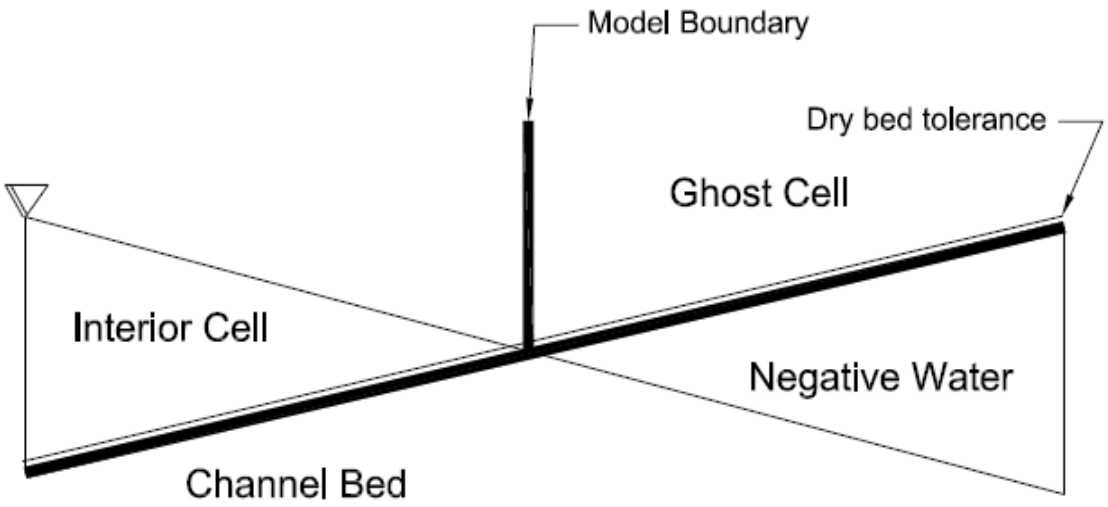


Figure 8.7 Extrapolation in the Ghost Cell

Because this height is then turned to zero, see Figure 8.8, an issue occurs with water level not being able to rise inside the model which causes an infinite friction situation. See Figure 8.9.

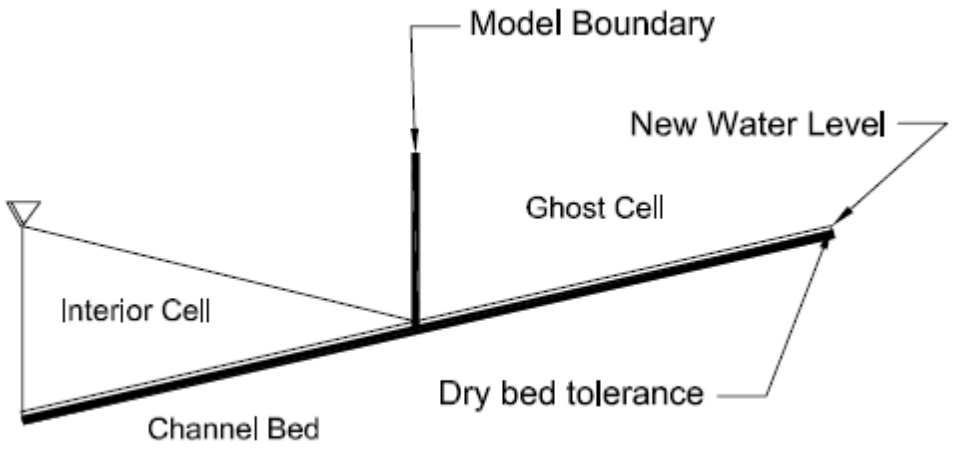


Figure 8.8 Water Level Adjustment

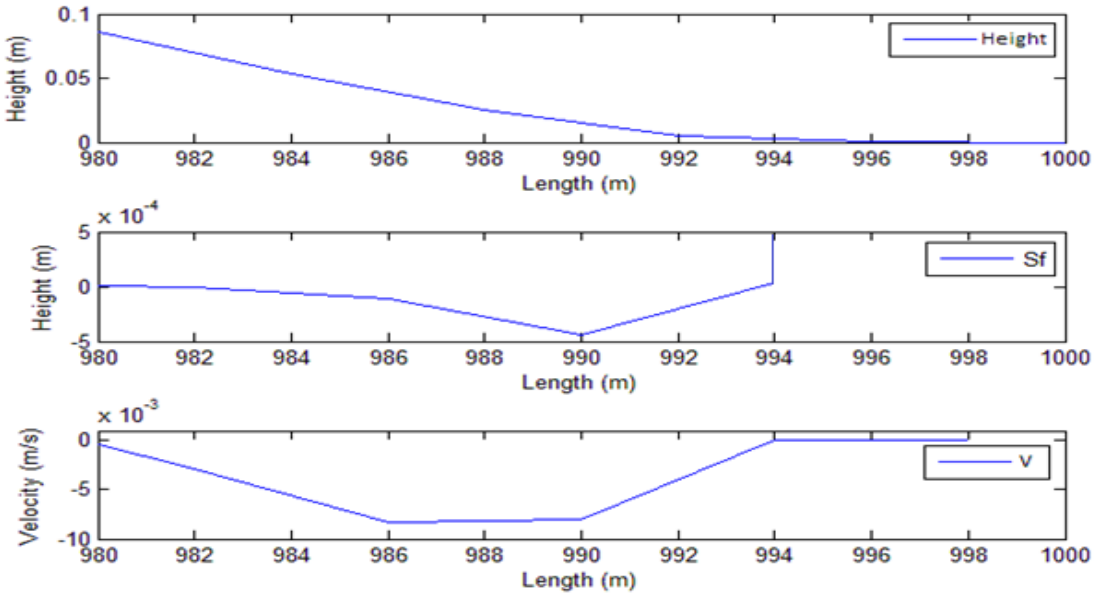


Figure 8.9 Infinite Friction

Now this doesn't appear good for uphill modelling, but the reality is in most cases that there will be no drainage ditch at that end. Instead, a system was modelled with a reflective

wall at the upstream end. Several of these systems were tried in the uphill case including adding clauses to switch between extrapolating the angle for dry bed situations and then setting the angle to zero for situations where the water is approaching the upstream boundary. This was to no avail as the system still encountered the same issue with upstream cases. Shown in Figure 8.10. A possible fix for this situation is to know the volume of water in the ghost cell by tracking water entering and leaving the ghost cell. From this volume it is then possible to find the height of water knowing the wave shape, which is covered later in this chapter. Other restrictions on uphill modelling were encountered with runtimes often being five or six hours. This is for two reasons. To find friction losses the height used is cell averaged. It was found when using bigger cells sizes the model would become unstable i.e. the courant number below 0.6, for small dry bed tolerances. This is because the big cell sizes create smaller water heights when averaging a partially wet cell. The smaller the water height the worse the errors are within the Mannings equation. In order to utilize the high accuracy of this model the cell sizes were then reduced which allows testing of the point where the Mannings equation becomes unstable. The dt must then also be set to around 0.05-0.1 seconds. The model time for the model that produced Figure 8.10 was 186 000 seconds which equates to a runtime of about 6 hours. The model runtime is greatly affected by the amount of computational cells. The large run times obviously puts restrictions on the amount of situations that could be tested for uphill cases.

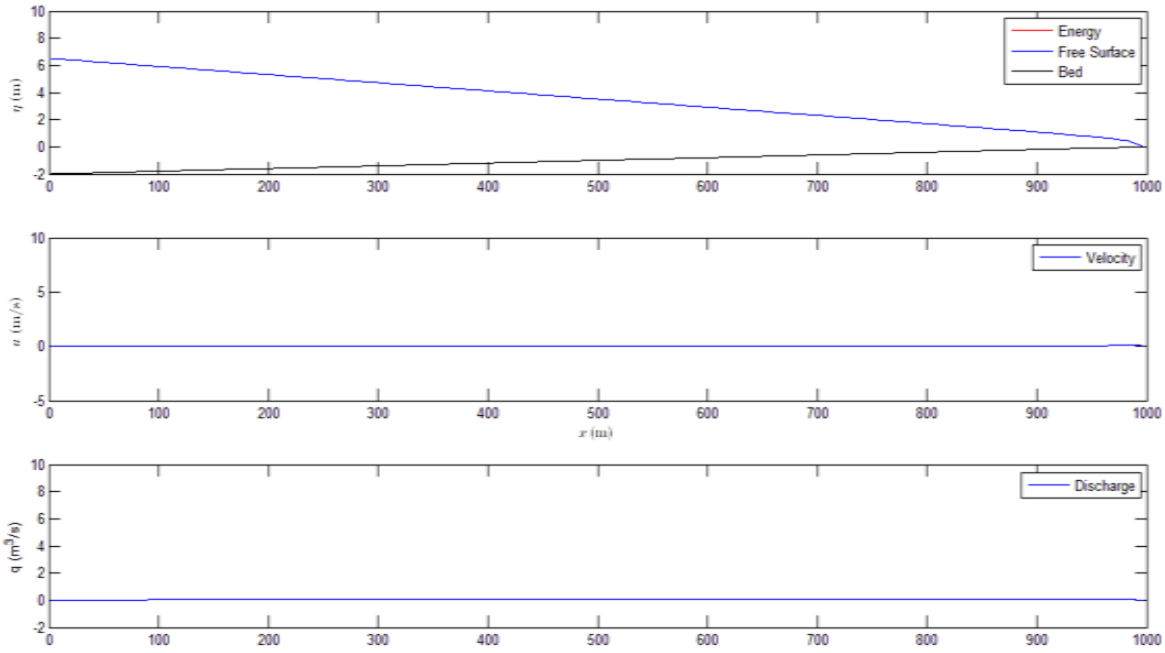


Figure 8.10 Last Output before Failure for Changed Boundary Conditions

Smaller cells sizes are also advantageous because the wave front is a direct interpolation between points within the cells. See Figure 8.11. So by decreasing the cell size the wave front becomes steep and the average of water height over the cell remains above the threshold. See Figure 8.13 and Figure 8.14. The question then becomes, at what angle does the front edge of the wave normally travel in dry conditions, and does the cell size replicate this. The same then goes for wet conditions. This will then come back to the modellers experience as a judgement will have to be made on cell sizes for channel length and wave height.

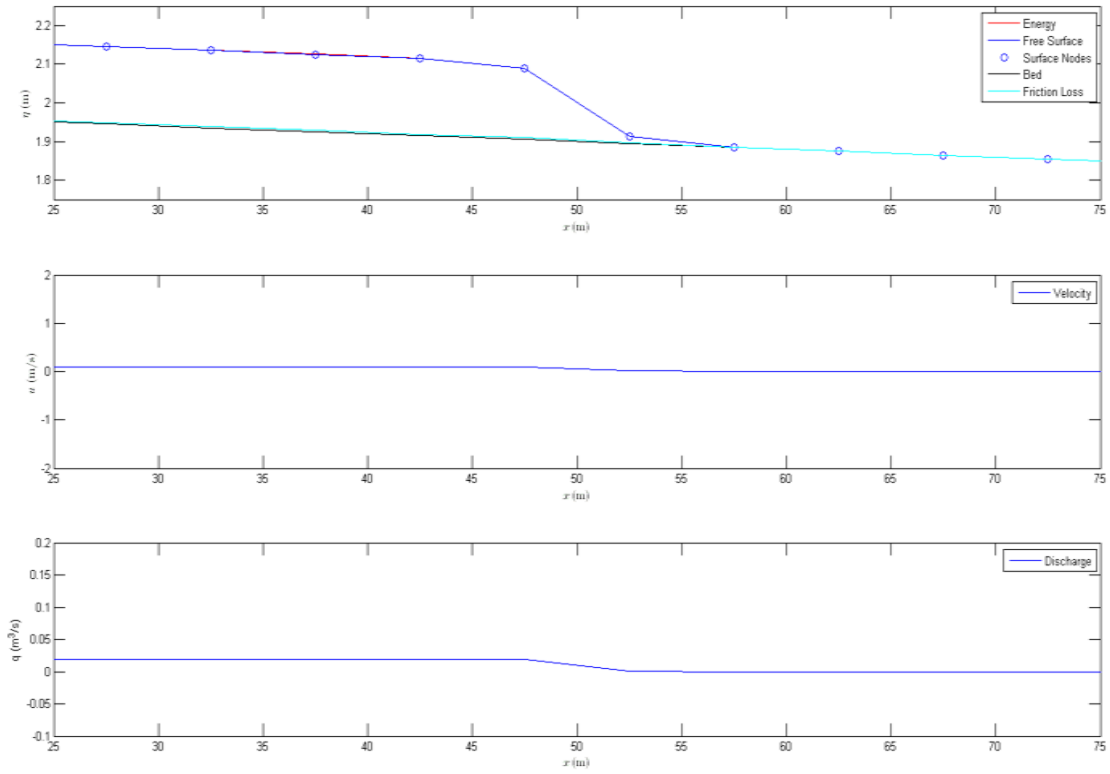


Figure 8.11 Direct Interpolation – Part 1

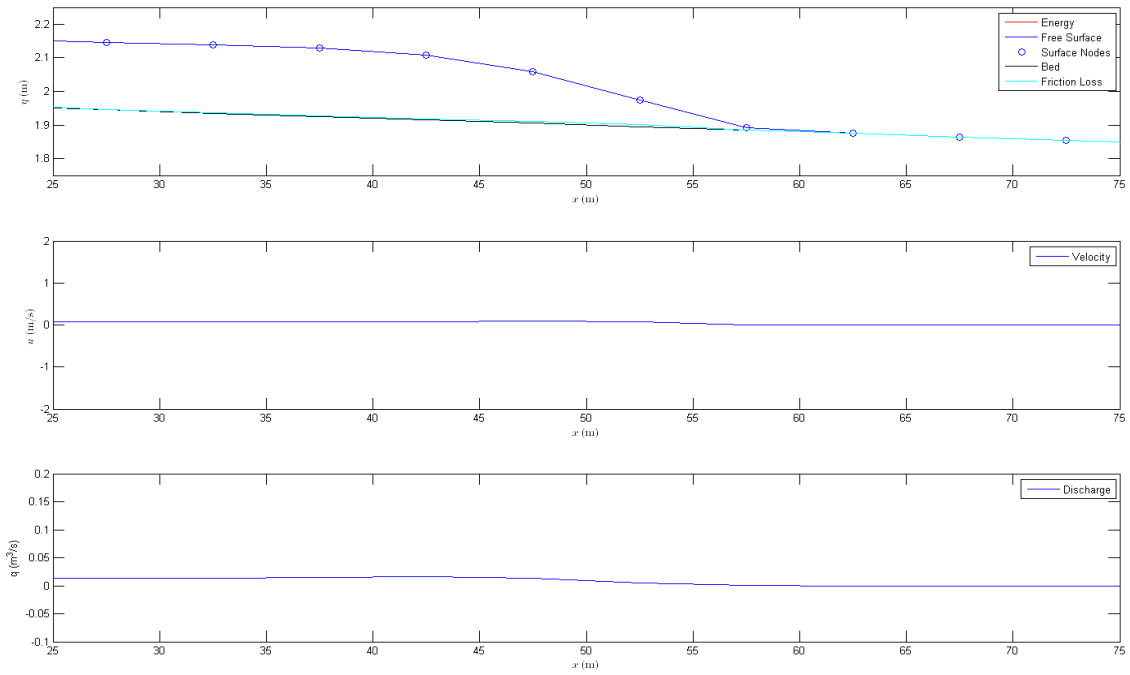


Figure 8.12 Direct Interpolation - Part 2

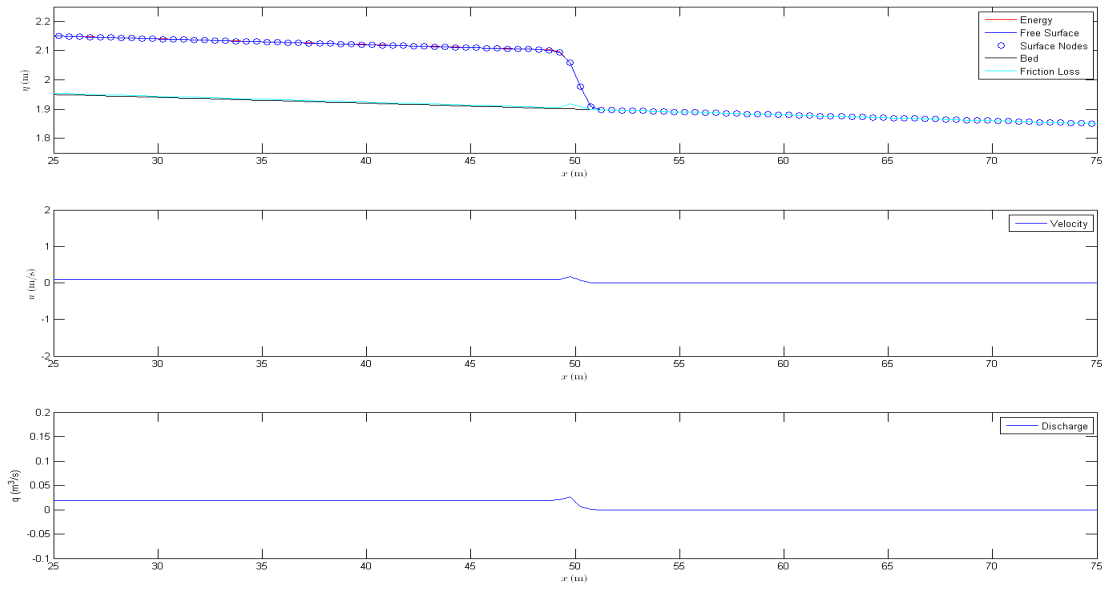


Figure 8.13 Decreased Cell Size and Time Step - Part 1

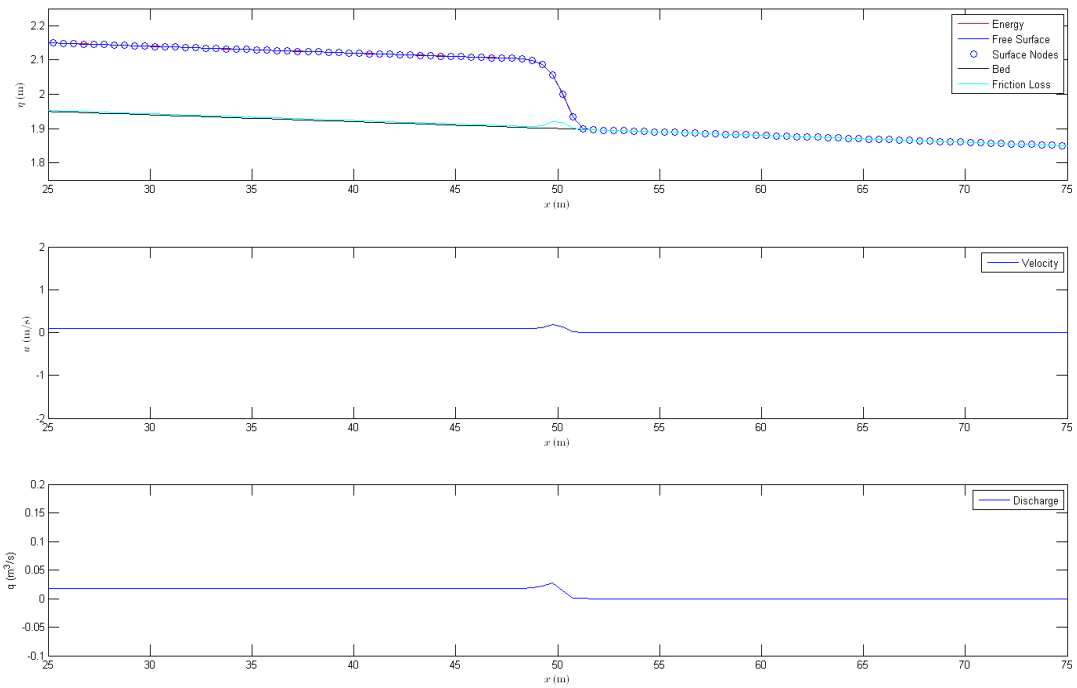
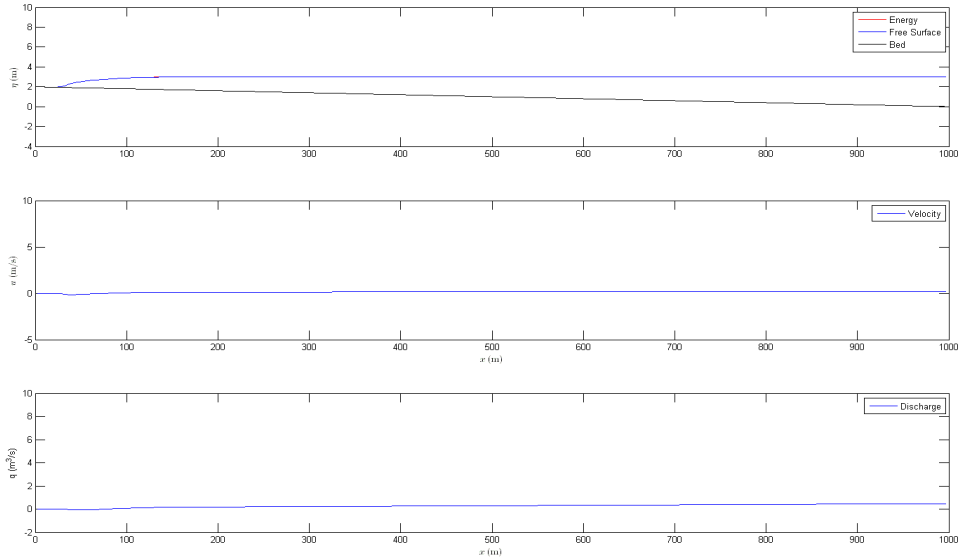
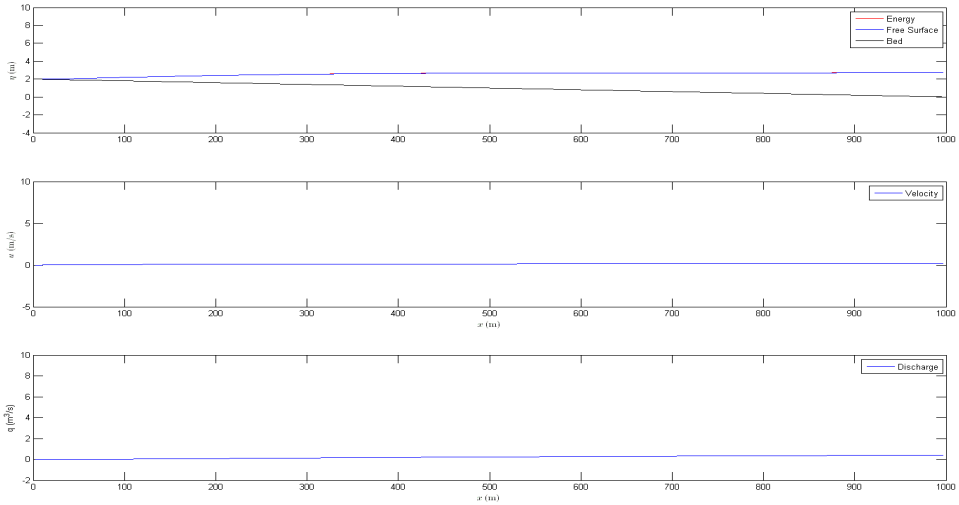


Figure 8.14 Decreased Cell Size and Time Step - Part 2

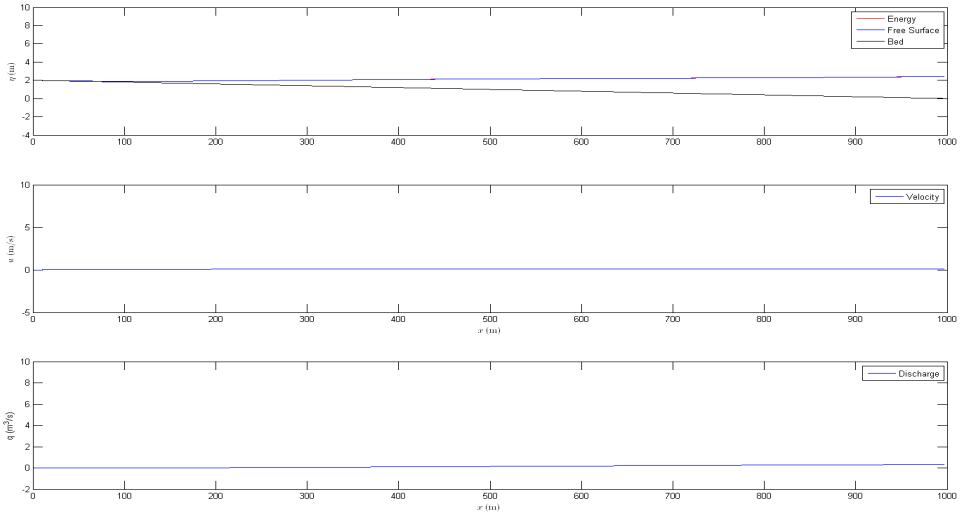
Typical draining situations can also be modelled. This is shown 8.15 through to 8.18. Again with verification of draining models the only way of truly testing the accuracy of the model is to compare it against real data. Verification of such a case is not required in this dissertation. Visual inspection of the draining situation shows no major errors in the model. Because the model has also been verified for steady state downhill flows it is assumed that this model will also run correctly.



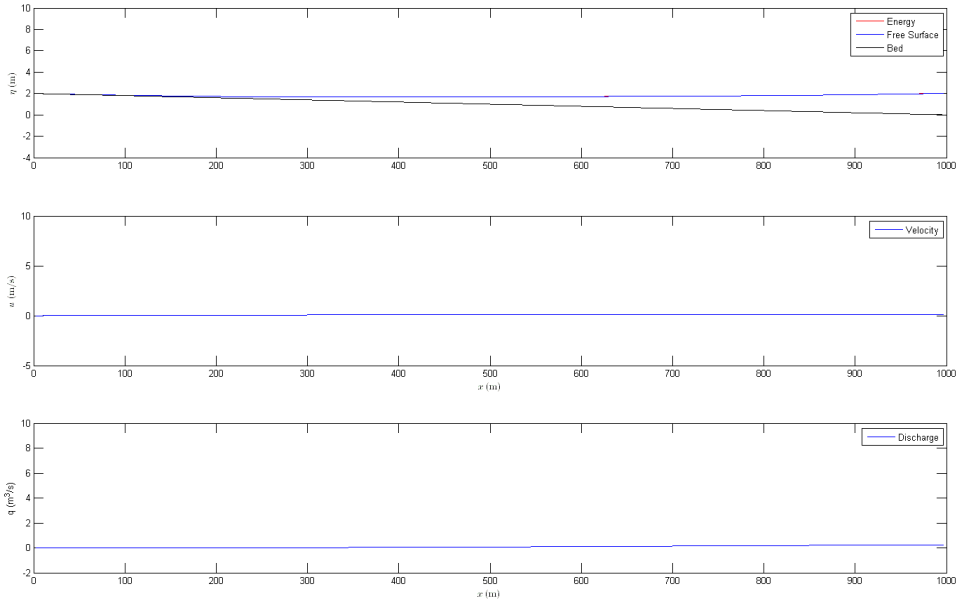
8.15 Draining Situations – Part 1



8.16 Draining Situations – Part 2



8.17 Draining Situations – Part 3



8.18 Draining Situations – Part 4

As can be seen in this chapter the model does have some limitations that will require further investigation. However, the model shows promising signs with its ability to handle a wide range of flows with a wide range of circumstances. This model has various limitations with smaller heights of flows, small velocities, and thus smaller flow rates. This appears to be linked mainly to the Manning's equations limitations and as there are few alternatives, this problem would plague many numerical solvers of this type.

9. Conclusion

9.1 Limitations of Use and Future Work

This model has been proven to be versatile but isn't without its limitations. These limitations are those that would be present in all forms of irrigation modelling and would have to be faced using any method for numerical simulation. An uphill, upstream, dry bed boundary condition is an issue that needs further attention. A suggested remedy is tracking the flow rate into the cell (nf-1) immediately interior of the ghost cell and interpolating the rate of change of flow rate from the cell (nf – 2). By doing this it could be possible to determine the volume of water leaving the model and because we have assumed shape of the water within the ghost cell it is possible to determine the height of the boundary condition.

Other limitations occur when the flows are fast which makes the model unstable. When there are very fast flows, smaller time steps and larger cells sizes must be used to keep the Courant number below 0.6. This can then affect the accuracy of the Mannings equation and a tolerance must then be implemented to stop the model from failing. This has been documented as a valid fix for this issue as waves propagate with correct characteristics.

Other areas that require attention are looking at lateral inflows and outflows. This would greatly increase the usefulness of the program and could be added as discrete flows or over the entire length of the channel. Infiltration is another possible addition to the model which

would increase its usefulness. Work should be done to extend the model into 2 dimensions. This would make the model more versatile, and would improve the modelling of discrete lateral flows and structures.

Future work could be done on verifying the model type in the time dimension. This can be done with data from experiments or from measured field data. A positive result from this verification would be useful in fully validating the use of this model type.

9.2 Modelling Conclusions

The model is accurate for testing many different situations. Situations which do push the limits of the models ability are not common within the irrigation environment. For steady state flows it has been verified that the model can achieve a height within a limit of 0.05% and thus proven that the finite volume method is a good way of predicting water flows. For uphill situations the model cannot handle a dry bed for the final computational cell. But it does appear to model the water flows correctly and can model well if the initial conditions are wet.

Draining situations model as expected but cannot be further verified here. Rigorous verifications could be done by comparing a simple model to a set of lab results, or field data.

Although not verified fully, the model shows many promising signs that could lead to further investigations being undertaken. The advantageous of using this modelling scheme make in ideal for modelling a range of flows with various source terms and make it useful to be incorporated into irrigation modelling.

REFERENCES

Akan A, 2006, Open Channel Hydraulics, Butterworth-Heinemann, Canada

Akanbi A & Katopodes N, 1987, Model for Flood Propagation on Initially Dry Land, Journal of Hydraulic Engineering, ASCE, Vol.114, 689-706

Begnudelli L, Bradford S and Sanders B, 2007, Adaptive Godunov-Based Model for Flood Simulation, Journal of Hydraulic Engineering, ASCE, Vol.134, 714-725

Begnudelli L and Sanders B, 2007, Conservative Wetting and Drying Methodology for Quadrilateral Grid Finite-Volume Models, Journal of Hydraulic Engineering, ASCE, Vol.133, 312-322

Begnudelli L and Sanders B, 2006, Unstructured Grid Finite-Volume Algorithm for Shallow-Water Flow and Scalar Transport with Wetting and Drying, Journal of Hydraulic Engineering, ASCE, Vol.132, 371-384

Bradford S and Sanders B, 2006, Impact of Limiters on Accuracy of High-Resolution Flow and Transport Models, Department of Civil and Environmental Engineering, University of California, Irvine, CA 92697

Bradford S and Sanders B, 2002a, Finite-Volume Model for Shallow-Water Flooding of Arbitrary Topography, Department of Civil and Environmental Engineering, University of California, Irvine, CA 92697

Bradford S and Sanders B, 2002b, High-resolution, monotone solution of the adjoint shallow-water equations, Department of Civil and Environmental Engineering, University of California, Irvine, CA 92697

Chadwick, A, Morfett, J, Borthwick M 2004, Hydraulics *in Civil and Environmental Engineering*, Spon Press, London.

GARCIA-NAVARRO P & Toro E, 2007, Godunov-type methods for free-surface shallow flows: A review, Journal of the Hydraulic Research, IAHR, Vol.45, 736-751

Guinot V, Godunov-type schemes: an introduction for engineers, 2003, Elsevier, Amsterdam

Julien P 2002, River Mechanics, Cambridge University Press, Cambridge.

Katopodes, N & Sanders B (2000), ADJOINT SENSITIVITY ANALYSIS FOR SHALLOW-WATER WAVE CONTROL, *Department of Civil and Environmental Engineering, University of California, Irvine, CA 92697*

Milne-Thomson L, Theoretical Hydrodynamics, 1968, Macmillan & Co Ltd, London.

Palm III W J, Introduction to MatLab 7 for Engineers, 2005, McGraw-Hill, New York

Playan E, Walker W, & Merkley, G, 1994. ‘‘Two-dimensional simulation of basin irrigation. I: Theory.’’ *J. Irrig. Drainage*, 120,837–855.

Roe, P. L. (1981). ‘‘Approximate Riemann solvers, parameter vectors, and difference schemes.’’ *J. Comput. Phys.*, 43, 357–372.

Sanders B (2001), High-resolution and non-oscillatory solution of the St.Venant equations in non-rectangular and non-prismatic channels, Department of Civil and Environmental Engineering, University of California, Irvine, CA 92697

Strelkoff, T, 1970, Numerical Solution of the Saint-Venant Equations, Journal of the Hydraulics Division, ASCE, Vol.96, 223-225

Szymkiewicz R, Numerical Modeling in Open Channel Hydraulics, 2010, Water Science and Technology Library, Springer, London

Toro E, Riemann Solvers and Numerical Methods for Fluid Dynamics – A Practical Introduction 3rd Edition, 2009, Springer, London

Zhang, W., and Cundy, T. W. 1989 ‘‘Modeling of two-dimensional overland flow.’’ *Water Resour. Res.*, 25, 2019–2035.

Appendices

Appendix A

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project

PROJECT SPECIFICATION

FOR: ADAM GOULD

TOPIC: 'Finite volume solution of the unsteady free surface flow equations'

SUPERVISORS: Prof. Rod Smith, Dr Malcolm Gillies

PROJECT AIM: To understand finite volume (conservative) techniques for solution of the unsteady free surface flow equations and to apply them to simple case studies.

PROGRAMME:

1. Review Finite Volume Solution Techniques used with the unsteady free surface flow equations, including the history, attributes and applications of the techniques.
2. Describe in detail the algorithms and working of an appropriate example of a Finite Volume technique.
3. Apply the MatLab codes of Sanders to a range of simple case studies to gain understanding of techniques and boundary conditions.
4. Modify the MatLab code to handle more complex cases.
5. Investigate application of Finite Volume techniques to flow over a dry bed by further modification of Sanders' code or development of original codes.

AGREED:

_____	_____	_____
(Student)	(Supervisors)	(Supervisors)
___/___/___	___/___/___	___/___/___

Appendix B

B.1 FVM1D

```
%FVM1D - FOR WET BED PROBLEMS ONLY
%by Brett F. Sanders (and pieces of code from Scott F.
Bradford)
%
%This is a very simple 1D solver of the shallow-water
equations
%that uses the Hancock predictor-corrector time-stepping
scheme,
%the MUSCL method of slope limiting and variable
reconstruction
%and Roe's approximate Riemann solver to compute fluxes.
%
%The code is kept as simple as possible to emphasize the
basic
%flow of logic. To account for problems involving a dry bed,
%one needs to add a number of "if" statements to avoid
%division by zero. This makes the code pretty messy and
%therefore these lines have been omitted.
%
%Note that the code can either be run in a first order or
%second order accurate mode. The user can select from
%several limiters to see how these impact the solution.
%
%To run this program, copy all the .m files into a directory
%and run "fvm1d.m" by either typing "fvm1d" at the matlab
%command prompt or pushing the execute button in the matlab
%text editor.

clear
close all
format long
```

```
global grav

%Set up grid
nc=100; %number of cells
nf=nc+1; %number of edges
L=1000; %length of channel
dx=L/nc; %length of cell
x=0:dx:L; %array of edge coordinates
xc=dx/2:dx:L-dx/2; %array of cell center coordinates

%Set up time marching and output interval
dt=0.5; %time step (s)
nt=100; %number of time steps
ntplot=10; %plot interval (number of time steps)

%Define bed elevation at faces z=f(x)
z=zeros(size(x)); %flat bed - can enter own function here,
z=f(x)

%Compute bed slope
for i=1:nc,
    dz(i)=z(i+1)-z(i); %dimensions of length
    zc(i)=0.5d0*(z(i+1)+z(i)); %elevation of cell center
end

%Set parameter values
grav=9.806;

%Set attributes of solver
iorder=2; %1=first order scheme, 2=second order scheme
beta=2; %controls limiter used by model
%Notes on limiters
%beta=1 => Minmod
%beta=2 => Superbee
%beta=3 => Fromm scheme, predicts oscillations at sharp
fronts
%beta=4 => Van Leer
%beta=5 => Van Albada
%beta=6 => Double Minmod
```

```
%Set up initial condition
xo=L/2;
etalo=10;
etaro=1;
ulo=0;
uro=0;
for i=1:nc,
    if (xc(i) < xo),
        eta(i)=etalo;
        h(i)=eta(i)-zc(i);
        u(i)=ulo;
    else
        eta(i)=etaro;
        h(i)=eta(i)-zc(i);
        u(i)=uro;
    end
end

%Initialize arrays
uh=h.*u;

deta=zeros(size(eta));
du=zeros(size(u));

t=0; %start time
for n=1:nt, %Begin time-marching loop
    if (iorder == 2), %for second order accuracy only
        deta = limiter(nc,beta,eta);
        du = limiter(nc,beta,u);
        [etap, up]=predictor(nc,eta,h,u,deta,du,dz,zc,dt,dx);
        hp=etap-zc; %Update for dry bed cases
        S = grav*hp.*dz/dx; %Source term treatment
        [F, amax] = fluxes(grav,nf,etap,up,z,dz,deta,du);
%Compute fluxes
    else
        [F, amax] = fluxes(grav,nf,eta,u,z,dz,deta,du);
        S = grav*h.*dz/dx; %Source term treatment
    end
    [uh, h, u] = corrector(nc,h,uh,F,S,dx,dt);
    eta=h+zc; %compute new free surface height
```

```
    e=eta+0.5*u.^2/grav; %compute energy in units of length
(head)
    t=t+dt;
    cr=amax*dt/dx; %compute Courant number
    fprintf(1,'%g %d\n',n,cr)
    if (cr > 1) %Stops program if Courant number exceeds one.
        break
    end
    if (mod(n,ntplot) == 0),
        subplot(3,1,1)
        plot(xc,e,'r-',xc,h+zc,'b-',xc,zc,'k-')
        axis([0 L 0 16])
        legend('Energy','Free Surface','Bed')
        subplot(3,1,2)
        plot(xc,u,'b-')
        axis([0 L -1 10])
        legend('Velocity')
        subplot(3,1,3)
        plot(x,F(:,1),'b-')
        axis([0 L -1 50])
        legend('Discharge')
        pause(0.1)
    end
end
```

B.2 limiter

```
function df = limiter(nc,beta,f)

df(1)=0;
df(nc)=0;
for i=2:nc-1,
    df1=f(i+1)-f(i);
    df2=f(i)-f(i-1);
    df(i)=limit(df1,df2,beta);
end
```

B.3 limit

```
function f=limit(d1,d2,beta)
% 0 = first order, 1-2 = beta, 3= fromm, 4=vanleer,
5=vanalbada, 6=double
% minmod

if beta==0, %first order
    f=0;
elseif beta >= 1 & beta <= 2%beta: minmod (beta=1) and
superbee (beta=2)
    if(d1*d2 < 0),
        f=0;
    else
        s=sign(d1);
        a=abs(d1);
        b=abs(d2);
        f=s*min(max([a b]),beta*min([a b]));
    end
elseif beta == 3 %Fromm
    f=0.5*(d1+d2);
elseif beta == 4 %vanleer
    if(d1*d2 <= 0),
        f=0;
    else
        f=2*d1*d2/(d1+d2);
    end
elseif beta ==  5 %vanalbada
     eps=1.e-20;
     f=(d1*(d2*d2+eps)+d2*(d1*d1+eps))/(d1*d1+d2*d2+2*eps);
elseif beta == 6 %double minmod
    if(d1*d2 < 0),
        f=0;
    else
        s=sign(d1);
        a=abs(d1);
        b=abs(d2);
```

```
        c=0.5*(a+b);
        f=s*min([2*a 2*b c]);
    end
end
```

B.4 predictor

```
function [etap,
up]=predictor(nc,eta,h,u,deta,du,dz,zc,dt,dx);

global grav

dh=deta-dz;
for i=1:nc,
    etap(i)=eta(i)-0.5*dt/dx*(h(i)*du(i)+u(i)*dh(i));
    up(i)=u(i)-0.5*dt/dx*(u(i)*du(i) + grav*deta(i));
end
```

B.5 fluxes

```
function [F, amax0] = fluxes(grav,nf,eta,u,z,dz,deta,du)

%I'm using a 2d solver in 1d, so I'm setting sn=0 and cn=1
for all cases
sn=0;
cn=1;
vl=0;
vr=0;

%Left Boundary : model as wall
hr=eta(1)-0.5*deta(1)-z(1);
ur=u(1)-0.5*du(1);
[fdum, amax]=solver(hr,hr,-ur,ur,vl,vr,sn,cn);
F(1,1)=fdum(1);
F(1,2)=fdum(2);
```

```
%Right Boundary : model as wall
hl=eta(nf-1)+0.5*deta(nf-1)-z(nf);
ul=u(nf-1)+0.5*du(nf-1);
[fdum, amax]=solver(hl,hl,ul,-ul,vl,vr,sn,cn);
F(nf,1)=fdum(1);
F(nf,2)=fdum(2);

%Other bc options could be used
%q=12.0;
%hr=2;
%F(1,1)=q;
%F(1,2)=q^2/hr+0.5*grav*hr^2;

%F(nf,1)=hl*ul;
%F(nf,2)=ul^2*hl+0.5*grav*hl^2;



amax0=0;
for i=2:nf-1, %Sweep over faces
    %Variable reconstruction
    hl=eta(i-1)+0.5*deta(i-1)-z(i);
    ul=u(i-1)+0.5*du(i-1);
    hr=eta(i)-0.5*deta(i)-z(i);
    ur=u(i)-0.5*du(i);
    %Call solver
    [fdum, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn);
    F(i,1)=fdum(1);
    F(i,2)=fdum(2);
    amax0=max([amax0 amax]); %Keep track of max wave speed to
check CFL
end
```

B.6 solver

```
function [F, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn)

global grav

%Compute Roe averages
duml=hl^0.5;
dumr=hr^0.5;
cl=(grav*hl)^0.5;
cr=(grav*hr)^0.5;
hhat=duml*dumr;
uhat=(duml*ul + dumr*ur)/(duml+dumr);
vhat=(duml*vl + dumr*vr)/(duml+dumr);
chat=(0.5*grav*(hl+hr))^0.5;
uperp=uhat*cn+vhat*sn;
dh=hr-hl;
du=ur-ul;
dv=vr-vl;
dupar=-du*sn+dv*cn;
duperp=du*cn+dv*sn;
dW=[0.5*(dh-hhat*duperp/chat); hhat*dupar;
0.5*(dh+hhat*duperp/chat)];

uperpl=ul*cn+vl*sn;
uperpr=ur*cn+vr*sn;
al1=uperpl-cl;
al3=uperpl+cl;
ar1=uperpr-cr;
ar3=uperpr+cr;
R=[1 0 1;
    uhat-chat*cn -sn uhat+chat*cn;
    vhat-chat*sn cn vhat+chat*sn];
da1=max([0 2*(ar1-al1)]);
da3=max([0 2*(ar3-al3)]);
a1=abs(uperp-chat);
a2=abs(uperp);
a3=abs(uperp+chat);
```

```
%Critical flow fix
if a1 < da1,
     a1=0.5*(a1*a1/da1+da1);
end
if a3 < da3,
     a3=0.5*(a3*a3/da3+da3);
end

%Compute interface flux
A=diag([a1 a2 a3]);
FL=[uperpl*hl; ul*uperpl*hl + 0.5*grav*hl*hl*cn; vl*uperpl*hl
+ 0.5*grav*hl*hl*sn];
FR=[uperpr*hr; ur*uperpr*hr + 0.5*grav*hr*hr*cn; vr*uperpr*hr
+ 0.5*grav*hr*hr*sn];
F=0.5*(FL + FR - R*A*dW);
amax=chat+abs(uperp);
```

B.7 corrector

```
function [uhnew, hnew,
unew]=corrector(nc,hold,uhold,F,S,dx,dt)

for i=1:nc,
    hnew(i)=hold(i)-dt/dx*(F(i+1,1)-F(i,1));
    uhnew(i)=uhold(i)-dt/dx*(F(i+1,2)-F(i,2))-dt*S(i);
    unew(i)=uhnew(i)/hnew(i); %update for dry bed cases
    %Add vfr for dry bed cases to compute eta from h
end
```

Appendix C – Sanders Dry bed Model

C.1 FVM1D

```
clear
close all
format long

global tol_h

%This model adopts a linearly sloping structure for the bed.

nc=100;
nf=nc+1;
L=100;
So=0.01;
nm=0.0; %Manning n
dx=L/nc;
x=0:dx:L; %cell edges
xc=dx/2:dx:L-dx/2; %cell centers

xo=60;
etalo=0;
etaro=0.8;
ulo=0;
uro=0;
grav=9.806;
beta=6;
iorder=1;
tol_h=1.d-4;
anim_onoff=0;
save_onoff=1;
nsave=0;



z=So*(L-x);
zc=So*(L-xc);

for i=1:nc,
    zmin(i)=min([z(i) z(i+1)]);
    zmax(i)=max([z(i) z(i+1)]);
    hcrit(i)=0.5*(zmax(i)-zmin(i));
    dz(i)=z(i+1)-z(i);
end

t=0;
dt=0.05;
nt=2000;
```

```
if (anim_onoff == 1),
    aviobj=avifile('linearz.avi');
end
hf=figure(1)

%Initial Condition
for i=1:nc,
    if (xc(i) < xo || xc(i) > 80),
        eta(i)=etalo;
        if (eta(i) > zmax(i)),
            h(i)=eta(i)-zc(i);
        elseif (eta(i) > zmin(i));
            h(i)=0.5*(eta(i)-zmin(i));
        else
            h(i)=0;
        end
        if (h(i) > tol_h),
            u(i)=ulo;
        else
            u(i)=0;
        end
    else
        eta(i)=etaro;
        if (eta(i) > zmax(i)),
            h(i)=eta(i)-zc(i);
        elseif (eta(i) > zmin(i));
            h(i)=0.5*(eta(i)-zmin(i));
        else
            h(i)=0;
        end
        if (h(i) > tol_h),
            u(i)=ulo;
        else
            u(i)=0;
        end
    end
    if eta(i) < zmin(i),
        eta(i) = zmin(i);
    end
    if (h(i) > tol_h),
        cd(i)=grav*nm*nm/h(i);
    else
        cd(i)=0;
    end
end
uh=h.*u;

deta=zeros(size(eta));
du=zeros(size(u));
duh=zeros(size(uh));

t1=cputime;
```

```
%plot initial condition
subplot(2,1,1)
plot(xc,zc,xc,eta)
ylabel('$\eta$ (m)','Interpreter','Latex')
title('Linear z model')
axis([0 L 0 1])
subplot(2,1,2)
plot(xc,u)
xlabel('$x$ (m)','Interpreter','Latex')
ylabel('$u$ (m/s)','Interpreter','Latex')
axis([0 L -10 10])
pause(0.1)
if (save_onoff == 1),
    nsave=nsave+1;
    usave(:,nsave)=u;
    etasave(:,nsave)=eta;
    tsave(nsave)=0;
end

if (anim_onoff == 1),
    aviobj=addframe(aviobj,hf); %adds frames to the AVI file
end

%Begin time loop
for n=1:nt,
    %eta = limiter(nc,beta,eta);
    %if (iorder == 2),
    %    deta = limiter(nc,beta,eta);
    %    duh = limiter(nc,beta,uh);
    %    [etap, hp, up,
uhp]=predictor(grav,nc,eta,h,uh,deta,duh,zc,dz,dt,dx);
    %    [F, amax] = fluxes(grav,nf,etap,up,uhp,zc,deta,du,duh);
    %    S = sourceterm(grav,nc,etap,deta,zc);
    %else
        [F, amax] = fluxes(grav,nf,eta,u,uh,z,deta,du,duh);
        S = -grav*h.*dz;
    %end;
    [uh, h, u] = corrector(nc,h,uh,F,S,cd,dx,dt);
    for i=1:nc,
        %Compute new free surface height (VFR method)
        if (h(i) > hcrit(i)),
            eta(i)=zc(i)+h(i);
        else
            eta(i)=zmin(i)+(2*h(i)*(zmax(i)-zmin(i)))^0.5;
        end
        %Compute new drag coefficient (for resistance)
        if (h(i) > tol_h),
            cd(i)=grav*nm*nm/h(i);
        else
            cd(i)=0;
        end
    end
    %e=zc+h+0.5*u.^2/grav;
    t=t+dt;
    cr=amax*dt/dx;
```

```
    fprintf(1,'%g %d %d %d\n',n,cr,h(50),eta(50))
    if (cr > 1)
        break
    end
    if (mod(n,20) == 0),
        ttext=['t=' num2str(t)];
        subplot(2,1,1)
        plot(xc,zc,xc,eta)
        ylabel('$\eta$ (m)','Interpreter','Latex')
        title('Linear z model')
        text(80,0.8,ttext)
        axis([0 L 0 1])
        subplot(2,1,2)
        plot(xc,u)
        xlabel('$x$ (m)','Interpreter','Latex')
        ylabel('$u$ (m/s)','Interpreter','Latex')
        axis([0 L -1 1])
        pause(0.1)
        if (anim_onoff == 1),
            aviobj=addframe(aviobj,hf); %adds frames to the AVI file
        end
        if (save_onoff == 1),
            nsave=nsave+1;
            usave(:,nsave)=u;
            etasave(:,nsave)=eta;
            tsave(nsave)=t;
        end
    end
end

if (anim_onoff == 1),
    aviobj=close(aviobj); %closes the AVI file
end
close(hf); %closes the handle to invisible figure

trun=cputime-t1

if (save_onoff == 1),
    save linearz xc zc etasave usave tsave
end
```

C.2 Limiter

```
function df = limiter(nc,beta,f)

df(1)=0;
df(nc)=0;
for i=2:nc-1,
```

```
    df1=f(i+1)-f(i);
    df2=f(i)-f(i-1);
    df(i)=limit(df1,df2,beta);
end
```

C.3 Limit

```
function f=limit(d1,d2,beta)
% 0 = first order, 1-2 = beta, 3= fromm, 4=vanleer, 5=vanalbada, 6=double
% minmod

if beta==0, %first order
    f=0;
elseif beta >= 1 & beta <= 2%beta
    if(d1*d2 < 0),
        f=0;
    else
        s=sign(d1);
        a=abs(d1);
        b=abs(d2);
        f=s*min(max([a b]),beta*min([a b]));
    end
elseif beta == 3 %Fromm
    f=0.5*(d1+d2);
elseif beta == 4 %vanleer
    if(d1*d2 < 0),
        f=0;
    else
        f=2*d1*d1/(d1+d2);
    end
elseif beta ==  5 %vanalbada
     eps=1.e-20;
     f=(d1*(d2*d2+eps)+d2*(d1*d1+eps))/(d1*d1+d2*d2+2*eps);
elseif beta == 6 %double minmod
    if(d1*d2 < 0),
        f=0;
    else
        s=sign(d1);
        a=abs(d1);
        b=abs(d2);
        c=0.5*(a+b);
        f=s*min([2*a 2*b c]);
    end
end
```

C.4 Predictor

```
function [etap, hp, up,
uhp]=predictor(grav,nc,eta,h,uh,deta,duh,zc,dz,dt,dx)

dh=deta-dz;
for i=1:nc,
    %etap(i)=eta(i)-0.5*dt/dx*(h(i)*du(i)+u(i)*dh(i));
    etap(i)=eta(i)-0.5*dt/dx*duh(i);
    hp(i)=etap(i)-zc(i);

    %up(i)=u(i)-0.5*dt/dx*(u(i)*du(i) + grav*deta(i));
    %uhp(i)=up(i)*hp(i);

    uhp(i)=uh(i)-0.5*dt/dx*(2*uh(i)*duh(i)/h(i) + (grav*h(i)-
uh(i)^2/h(i)^2)*dh(i) + grav*h(i)*dz(i));
    up(i)=uhp(i)/hp(i);
end
```

C.5 Fluxes

```
function [F, amax0] = fluxes(grav,nf,eta,u,uh,z,deta,du,duh)

global tol_h

hr=eta(1)-0.5*deta(1)-z(1);
if hr < 0,
    hr=0;
end
F(1,1)=0;
F(1,2)=0.5*grav*hr^2;

hl=eta(nf-1)+0.5*deta(nf-1)-z(nf);
if hl < 0,
    hl=0;
end
%ul=u(nf-1)+0.5*du(nf-1);
%ul=(uh(nf-1)+0.5*duh(nf-1))/hl;
F(nf,1)=0;
F(nf,2)=0.5*grav*hl^2;

%q=2;
%hr=2;
%F(1,1)=q;
%F(1,2)=q^2/hr+0.5*grav*hr^2;

%F(nf,1)=hl*ul;
%F(nf,2)=ul^2*hl+0.5*grav*hl^2;
```

```
sn=0;
cn=1;
vl=0;
vr=0;

amax0=0;
for i=2:nf-1,
    hl=eta(i-1)+0.5*deta(i-1)-z(i);
    if hl > tol_h,
        ul=(uh(i-1)+0.5*duh(i-1))/hl;
        if (ul*ul/(grav*hl)) > 1,
            ul=u(i-1)+0.5*du(i-1);
        end
    else
        hl = max([hl 0]);
        ul = 0;
    end
    hr=eta(i)-0.5*deta(i)-z(i);
    if hr > tol_h,
        ur=(uh(i)-0.5*duh(i))/hr;
        if (ur*ur/(grav*hr)) > 1,
            ur=u(i)-0.5*du(i);
        end
    else
        hr = max([hr 0]);
        ur = 0;
    end
    if (hl > tol_h || hr > tol_h),
        [fdum, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn);
        F(i,1)=fdum(1);
        F(i,2)=fdum(2);
        amax0=max([amax0 amax]);
    else
        F(i,1)=0;
        F(i,2)=0;
    end
end
```

C.6 Solver

```
function [F, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn)
g=9.806;
duml=hl^0.5;
dumr=hr^0.5;
cl=(g*hl)^0.5;
cr=(g*hr)^0.5;
hhat=duml*dumr;
uhat=(duml*ul + dumr*ur)/(duml+dumr);
vhat=(duml*vl + dumr*vr)/(duml+dumr);
chat=(0.5*g*(hl+hr))^0.5;
```

```
uperp=uhat*cn+vhat*sn;
dh=hr-hl;
du=ur-ul;
dv=vr-vl;
dupar=-du*sn+dv*cn;
duperp=du*cn+dv*sn;
dW=[0.5*(dh-hhat*duperp/chat); hhat*dupar; 0.5*(dh+hhat*duperp/chat)];

uperpl=ul*cn+vl*sn;
uperpr=ur*cn+vr*sn;
al1=uperpl-cl;
al3=uperpl+cl;
ar1=uperpr-cr;
ar3=uperpr+cr;
R=[1 0 1;
    uhat-chat*cn -sn uhat+chat*cn;
    vhat-chat*sn cn vhat+chat*sn];
da1=max([0 2*(ar1-al1)]);
da3=max([0 2*(ar3-al3)]);
a1=abs(uperp-chat);
a2=abs(uperp);
a3=abs(uperp+chat);

if a1 < da1,
     a1=0.5*(a1*a1/da1+da1);
end
if a3 < da3,
     a3=0.5*(a3*a3/da3+da3);
end
A=diag([a1 a2 a3]);
FL=[uperpl*hl; ul*uperpl*hl + 0.5*g*hl*hl*cn; vl*uperpl*hl +
0.5*g*hl*hl*sn];
FR=[uperpr*hr; ur*uperpr*hr + 0.5*g*hr*hr*cn; vr*uperpr*hr +
0.5*g*hr*hr*sn];
F=0.5*(FL + FR - R*A*dW);
amax=chat+abs(uperp);
%dU=[hr-hl; ur*hr-ul*hl; vr*hr-vl*hl];
%amax1=max([a1 a3]);
%amax2=abs(uperp)+chat*cn;
%amax3=abs(uperp)+chat*sn;
%A=diag([amax1 amax2 amax3]);
%F=0.5*(FL+FR-A*dU);
%F=0.5*(FL+FR);
```

C.7 Corrector

```
function [uhnew, hnew, unew]=corrector(nc,hold,uhold,F,S,cd,dx,dt)

global tol_h
```

```
for i=1:nc,
    hnew(i)=hold(i)-dt/dx*(F(i+1,1)-F(i,1));
    uhnew(i)=uhold(i)-dt/dx*(F(i+1,2)-F(i,2))+dt*S(i)/dx;
    if (hnew(i) > tol_h && hold(i) > tol_h), %account for friction
        uhnew(i)=uhnew(i)/(1+dt*abs(uhold(i))/(hold(i)*hnew(i)));
    end
    if (hnew(i) > tol_h ),
        unew(i)=uhnew(i)/hnew(i);
    else
        unew(i)=0;
    end
    if (hnew(i) < 0),
        hnew(i)=0;
    end
end
```

C.8 Sourceterm

```
function S = sourceterm(grav,nc,eta,deta,z)

for i=1:nc,
    if z(i)>z(i+1),
        if eta(i) > z(i),
            h1=eta(i)-z(i+1);
            h2=eta(i)-z(i);
            Sl=0.5*grav*(h1*h1-h2*h2);
        elseif eta(i) > z(i+1),
            h1=eta(i)-z(i+1);
            Sl=0.5*grav*h1*h1;
        else
            Sl=0;
        end
    else
        Sl=0;
    end
    if z(i+1)>z(i),
        if eta(i) > z(i+1),
            h1=etar-z(i);
            h2=etar-z(i+1);
            Sr=-0.5*grav*(h1*h1-h2*h2);
        elseif etar > z(i),
            h1=etar-z(i);
            Sr=0.5*grav*h1*h1;
        else
            Sr=0;
        end
    else
        Sr=0;
    end
    S(i)=Sl+Sr;
```

```
end
```

Appendix D – Commented code

D.1 FVM1D

```
%FVM1D - FOR WET BED PROBLEMS ONLY
%by Brett F. Sanders (and pieces of code from Scott F. Bradford)
%
%This is a very simple 1D solver of the shallow-water equations
%that uses the Hancock predictor-corrector time-stepping scheme,
%the MUSCL method of slope limiting and variable reconstruction
%and Roe's approximate Riemann solver to compute fluxes.
%
%The code is kept as simple as possible to emphasize the basic
%flow of logic. To account for problems involving a dry bed,
%one needs to add a number of "if" statements to avoid
%division by zero. This makes the code pretty messy and
%therefore these lines have been omitted.
%
%Note that the code can either be run in a first order or
%second order accurate mode. The user can select from
%several limiters to see how these impact the solution.
%
%To run this program, copy all the .m files into a directory
%and run "fvm1d.m" by either typing "fvm1d" at the matlab
%command prompt or pushing the execute button in the matlab
%text editor.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%------------------Script edited by Adam M Gould USQ------------------%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%--------------------------Initial Setup-------------------------------%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clc
close all
format long

global grav

%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Set up grid   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nc=100;              %number of cells
nf=nc+1;             %number of edges
L=1000;              %length of channel
dx=L/nc;             %length of cell
x=0:dx:L;            %array of edge coordinates
xc=dx/2:dx:L-dx/2;   %array of cell center coordinates

%%%%%%%%%%%%    Set up time marching and output interval   %%%%%%%%%%%%%%%
dt=0.5;              %time step (s)
nt=25000;              %number of time steps
```

```
ntplot=250;          %plot interval (number of time steps)

%%%%%%%%%%%%%     Define bed elevation at faces z=f(x)    %%%%%%%%%%%%%%
%z=[2:-.04:0 0:.04:2]
%z=zeros(size(x));   %flat bed - can enter own function here, z=f(x)
So=1/1000;
z=So*(L-x);
%%%%%%%%%%%%%            Compute bed slope            %%%%%%%%%%%%%%%%%


for i=1:nc,             %for cell 1 through to number of cells
    dz(i)=z(i+1)-z(i);  %dimensions of length
    zc(i)=0.5d0*(z(i+1)+z(i)); %elevation of cell center
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%----------------------Set parameter values-----------------------------
-%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
grav=9.806;

%%%%%%----------------------Set attributes of solver------------------
%%%%%%%

iorder=2;           %1=first order scheme, 2=second order scheme
beta=2;             %controls limiter used by model
                    %Notes on limiters
                    %beta=1 => Minmod
                    %beta=2 => Superbee
                    %beta=3 => Fromm scheme, predicts oscillations at
                           %sharp fronts
                    %beta=4 => Van Leer
                    %beta=5 => Van Albada
                    %beta=6 => Double Minmod
xo=L/2;                 %position along channel where wave starts
etalo=2;               %height on left hand side of the wave
etaro=1.5;                %height on right hand side of the wave
ulo=1;                   %initial velocity left side of wave (U/S)
uro=1;                   %initial velocity right side of wave (D/S)


for i=1:nc,
    if (xc(i) < xo),            %%If cell is before wave then
        eta(i)=etalo;           %%height from datum = initial height left
                                %%of wave
        h(i)=eta(i)-zc(i);      %%height of water = initial height from
                                %%datum - height of bed
        u(i)=ulo;               %%velocity = velocity of left side of
wave
    else                        %%if cell is after wave then
        eta(i)=etaro;           %%height from datum = initial height on
                                %%right side of wave
        h(i)=eta(i)-zc(i);      %%height of water = initial height from
```

```
                                %%datum - height of bed
        u(i)=uro;               %%velocity = velocity of right side of
wave
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%----Initialize arrays---%%%%%%%%%%%%%%%%%%%%%%%%%
uh=h.*u;                    %%height of water * velocity ????flow rate
matrix

deta=zeros(size(eta));      %%Initialize change in height from datum
du=zeros(size(u));          %%Initialize change in velocity

t=0; %start time

for n=1:nt, %Begin time-marching loop
    if (iorder == 2), %for second order accuracy only
        deta = limiter(nc,beta,eta);
        du = limiter(nc,beta,u);
                                %%we now have found the
                                    %%difference in velocity and
                                    %%difference in
                                    %%height at the current moment in
time
        [etap, up]=predictor(nc,eta,h,u,deta,du,dz,zc,dt,dx);
                                %%we now have height and velocity
                                    %%at t+(1/2dt)
        hp=etap-zc; %Update for dry bed cases
        S = grav*hp.*dz/dx; %Source term treatment
        [F, amax] = fluxes(grav,nf,etap,up,z,dz,deta,du); %Compute fluxes
                                %%these fluxes are the momentum and mass
                                %%fluxes at each cell interface

    else
        [F, amax] = fluxes(grav,nf,eta,u,z,dz,deta,du);
        S = grav*h.*dz/dx; %Source term treatment
    end
    [uh, h, u] = corrector(nc,h,uh,F,S,dx,dt);
    eta=h+zc; %compute new free surface height
    e=eta+0.5*u.^2/grav; %compute energy in units of length (head)
    t=t+dt;
  %%%%%%%%%%%----------Check the courant number----------%%%%%%%%%%%%
    cr=amax*dt/dx; %compute Courant number
    fprintf(1,'%g %d\n',n,cr)

    if (cr > 1) %Stops program if Courant number exceeds one.
        break
    end

  %%%%%%%%%%%%%%---------------plotting loop---------------%%%%%%%%%%%%%%

    if (mod(n,ntplot) == 0),
        figure('Position',get(0,'ScreenSize'))
```

```
        subplot(3,1,1)
        plot(xc,e,'r-',xc,h+zc,'b-',xc,zc,'k-')
        axis([0 L 0 5])
        legend('Energy','Free Surface','Bed')
        subplot(3,1,2)
        plot(xc,u,'b-')
        axis([0 L -2 10])
        legend('Velocity')
        subplot(3,1,3)
        plot(x,F(:,1),'b-')
        axis([0 L -5 10])
        legend('Discharge')
        pause(0.005)
        saveas(gcf, num2str(n), 'bmp')
    end
end
```

D.2 limiter

```
function df = limiter(nc,beta,f)  %%deta or du = (nc,beta,eta) or
(nc,beta,u)

df(1)=0;         %% Ghost cell boundary condition
df(nc)=0;        %% Ghost cell boundary condition
for i=2:nc-1,    %%for rest of cells
    df1=f(i+1)-f(i);         %%forward difference
    df2=f(i)-f(i-1);         %%backwards difference
    df(i)=limit(df1,df2,beta);      %%Call limit function
end
```

D.3 limit

```
%%Only really concerned with beta = 2
function f=limit(d1,d2,beta)
%
% 0 = first order, 1-2 = beta, 3= fromm, 4=vanleer, 5=vanalbada, 6=double
% minmod

if beta==0, %first order
    f=0;
elseif beta >= 1 & beta <= 2%beta: minmod (beta=1) and superbee (beta=2)
    if(d1*d2 < 0),
```

```
        f=0;
    else
        s=sign(d1);
        %%sign means>0=1 0=0 <0=-1
        a=abs(d1);
        %%absolute values
        b=abs(d2);
        %%absolute values
        f=s*min(max([a b]),beta*min([a b]));
        %%where f = df which is deta or du
        %%df = sign(forward
        %%diff)*the min of(max(forward diff, back diff),
        %beta*min(foward diff,back diff))
    end
elseif beta == 3 %Fromm
    f=0.5*(d1+d2);
elseif beta == 4 %vanleer
    if(d1*d2 <= 0),
        f=0;
    else
        f=2*d1*d2/(d1+d2);
    end
elseif beta ==  5 %vanalbada
     eps=1.e-20;
     f=(d1*(d2*d2+eps)+d2*(d1*d1+eps))/(d1*d1+d2*d2+2*eps);
elseif beta == 6 %double minmod
    if(d1*d2 < 0),
        f=0;
    else
        s=sign(d1);
        a=abs(d1);
        b=abs(d2);
        c=0.5*(a+b);
        f=s*min([2*a 2*b c]);
    end
end
```

D.4 predictor

```
%%The prediction step is used to predict future velocity and future
height
%the future is dt/2 or in this case .25 of a second
%%This is know as the Hancock's method described on pg323 of
%%high resolution and non-oscillatory
%%note these equations are eq12(A) and eq13(v)without friction and
bedslope
```

```
%%terms
function [etap, up]=predictor(nc,eta,h,u,deta,du,dz,zc,dt,dx);

global grav

dh=deta-dz;
for i=1:nc,
    etap(i)=eta(i)-0.5*dt/dx*(h(i)*du(i)+u(i)*dh(i));
%%eq12(refered above)
    up(i)=u(i)-0.5*dt/dx*(u(i)*du(i) + grav*deta(i));       %%eq13
without bedslope or friction terms
end
```

D.5 fluxes

```
function [F, amax0] = fluxes(grav,nf,eta,u,z,dz,deta,du)

%I'm using a 2d solver in 1d, so I'm setting sn=0 and cn=1 for all cases

sn=0;
cn=1;
vl=0;           %%%these are zero because 1D solver
vr=0;           %%%These are zero because 1D solver
q=2;

%Left Boundary
hr=eta(1)-0.5*deta(1)-z(1);          %%eq18 pg324
ur=u(1)-0.5*du(1);                   %%eq19 pg324
[fdum, amax]=solver(hr,hr,-ur,ur,vl,vr,sn,cn);

F(1,1)=q;               %% inflow boundary
F(1,2)=q^2/hr+0.5*grav*hr^2;            %%momentum boundary


%Right Boundary
hl=eta(nf-1)+0.5*deta(nf-1)-z(nf);  %%eq18 pg324
ul=u(nf-1)+0.5*du(nf-1);            %%eq19 pg324
[fdum, amax]=solver(hl,hl,ul,-ul,vl,vr,sn,cn);
F(nf,1)=hl*ul;                      %%boundary flow rate
F(nf,2)=ul^2*hl+0.5*grav*hl^2;      %%boundary momentum



amax0=0;
%%Same as above repeated for internal cells
for i=2:nf-1, %Sweep over faces
    %Variable reconstruction
    hl=eta(i-1)+0.5*deta(i-1)-z(i);
    ul=u(i-1)+0.5*du(i-1);
```

```
    hr=eta(i)-0.5*deta(i)-z(i);
    ur=u(i)-0.5*du(i);
    %Call solver
    [fdum, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn);
    F(i,1)=fdum(1);
    F(i,2)=fdum(2);
    amax0=max([amax0 amax]); %Keep track of max wave speed to check CFL
end
```

D.6 solver

```
%%fluxes are calculated as described within pg290 and 291 of Finite
volume
%%model for shallow water flooding of arbitrary topography

function [F, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn)

global grav

%Compute Roe averages
duml=hl^0.5;
dumr=hr^0.5;
cl=(grav*hl)^0.5;
cr=(grav*hr)^0.5;
hhat=duml*dumr;
uhat=(duml*ul + dumr*ur)/(duml+dumr);
vhat=(duml*vl + dumr*vr)/(duml+dumr);
chat=(0.5*grav*(hl+hr))^0.5;
uperp=uhat*cn+vhat*sn;
dh=hr-hl;
du=ur-ul;
dv=vr-vl;
dupar=-du*sn+dv*cn;
duperp=du*cn+dv*sn;
dW=[0.5*(dh-hhat*duperp/chat); hhat*dupar; 0.5*(dh+hhat*duperp/chat)];
%%eq12

uperpl=ul*cn+vl*sn;
uperpr=ur*cn+vr*sn;
al1=uperpl-cl;
al3=uperpl+cl;
ar1=uperpr-cr;
ar3=uperpr+cr;
R=[1 0 1;
%%eq11
    uhat-chat*cn -sn uhat+chat*cn;
    vhat-chat*sn cn vhat+chat*sn];
da1=max([0 2*(ar1-al1)]);
da3=max([0 2*(ar3-al3)]);
a1=abs(uperp-chat);
```

```
a2=abs(uperp);
a3=abs(uperp+chat);

%Critical flow fix
if a1 < da1,
     a1=0.5*(a1*a1/da1+da1);
end
if a3 < da3,
     a3=0.5*(a3*a3/da3+da3);
end

%Compute interface flux
A=diag([a1 a2 a3]);
%%eq9
FL=[uperpl*hl; ul*uperpl*hl + 0.5*grav*hl*hl*cn; vl*uperpl*hl +
0.5*grav*hl*hl*sn];%%eq7
FR=[uperpr*hr; ur*uperpr*hr + 0.5*grav*hr*hr*cn; vr*uperpr*hr +
0.5*grav*hr*hr*sn];%%eq7
F=0.5*(FL + FR - R*A*dW);
%%eq8
amax=chat+abs(uperp);
%%just below eq9
```

D.7 corrector

```
function [uhnew, hnew, unew]=corrector(nc,hold,uhold,F,S,dx,dt)

for i=1:nc,
    hnew(i)=hold(i)-dt/dx*(F(i+1,1)-F(i,1));%%Unsure where
    uhnew(i)=uhold(i)-dt/dx*(F(i+1,2)-F(i,2))-dt*S(i);%%eq9 of high res
and non-osc or eq36 for dry bed
    unew(i)=uhnew(i)/hnew(i); %update for dry bed cases
    %Add vfr for dry bed cases to compute eta from h
end
```

Appendix E – End Product

E.1 FVM1D

```
%FVM1D - FOR WET BED PROBLEMS ONLY
%by Brett F. Sanders (and pieces of code from Scott F. Bradford)
%
%This is a very simple 1D solver of the shallow-water equations
%that uses the Hancock predictor-corrector time-stepping scheme,
%the MUSCL method of slope limiting and variable reconstruction
%and Roe's approximate Riemann solver to compute fluxes.
%
%The code is kept as simple as possible to emphasize the basic
%flow of logic. To account for problems involving a dry bed,
%one needs to add a number of "if" statements to avoid
%division by zero. This makes the code pretty messy and
%therefore these lines have been omitted.
%
%Note that the code can either be run in a first order or
%second order accurate mode. The user can select from
%several limiters to see how these impact the solution.
%
%To run this program, copy all the .m files into a directory
%and run "fvm1d.m" by either typing "fvm1d" at the matlab
%command prompt or pushing the execute button in the matlab
%text editor.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%------------------Script edited by Adam M Gould USQ------------------%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%--------------------------Initial Setup-----------------------------%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear
clc
close all
format long

global grav
global tol_h

%%%%%%%%%%%%%%%%%%%%%%%%%%%%   Set up grid   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nc=200;              %number of cells
nf=nc+1;             %number of edges
L=1000;              %length of channel
dx=L/nc;             %length of cell
x=0:dx:L;            %array of edge coordinates
xc=dx/2:dx:L-dx/2;   %array of cell center coordinates
nm=0.2;             %Manning n
```

```
    %can include bed slope So
%%%%%%%%%%%%    Set up time marching and output interval   %%%%%%%%%%%%%%

dt=0.05;             %time step (s)
nt=2000000;             %number of time steps
ntplot=5000;          %plot interval (number of time steps)
So=2/1000;

%%%%%%%%%%%%%     Define bed elevation at faces z=f(x)    %%%%%%%%%%%%%%%
    %if using So t need this section
%z=[2:-.04:0 0:.04:2]
%z=zeros(size(x));   %flat bed - can enter own function here, z=f(x)
z=So*(L-x);
%zc=So*(L-xc);
%%%%%%%%%%%%%%            Compute bed slope            %%%%%%%%%%%%%%%%%%
%j = jth cell
for j=1:nc,

    zc(j)=0.5d0*(z(j+1)+z(j));  %elevation of cell center

    zmin(j)=min([z(j) z(j+1)]);%lowest point in the cell
    zmax(j)=max([z(j) z(j+1)]);%highest point in the cell
    hcrit(j)=0.5*(zmax(j)-zmin(j));
    dz(j)=z(j+1)-z(j);%change in elavation at the cell
    %So=dz/dx;

end

% for i=1:nc,                     %for cell 1 through to number of cells
%     dz(i)=z(i+1)-z(i);          %dimensions of length
%     zc(i)=0.5d0*(z(i+1)+z(i));  %elevation of cell center
% end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%----------------------Set parameter values-----------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
grav=9.806;
tol_h=1.d-4; %height to define whether cell is wet or dry
%%%%%--------------------Set attributes of solver------------------%%%%%%

iorder=2;           %1=first order scheme, 2=second order scheme
beta=6;             %controls limiter used by model
                    %Notes on limiters
                    %beta=1 => Minmod
                    %beta=2 => Superbee
                    %beta=3 => Fromm scheme, predicts oscillations at
                              %sharp fronts
                    %beta=4 => Van Leer
                    %beta=5 => Van Albada
                    %beta=6 => Double Minmod
xo=L/20;                %position along channel where wave starts
etalo=.2;              %height on left hand side of the wave
etaro=-3;               %height on right hand side of the wave
```

```
ulo=.1;                    %initial velocity left side of wave (U/S)
uro=.1;                    %initial velocity right side of wave (D/S)


for j=1:nc,

    if (xc(j) < xo),             %%If cell is before wave then
        %eta(j)=etalo;            %%height from datum = initial height
left
                                 %%of wave
        eta(j)=etalo+zc(j);
        if (eta(j) > zmax(j)),   %%if height from datum is higher
                                 %%than the highest bed elevation
            h(j)=eta(j)-zc(j);       %%height of water = initial height
from
                                 %%datum - height of bed
        elseif (eta(j) > zmin(j));%if height from datum is smaller than
                                     %%highest cell point but bigger than
                                     %%the lowest cell point
         h(j)=0.5*(eta(j)-zmin(j));%h = 1/2 height of water from datum
                                          %%- bed elevation
         else
            h(j)=0; %else water = 0
        end
        if (h(j) > tol_h), %if water height is bigger than threshold
            u(j)=ulo;%%velocity = velocity of left side of wave
        else
            u(j)=0;%if not velocity is zero
            h(j)=0;
        end
    else                         %%if cell is after wave then
        eta(j)=etaro;            %%height from datum = initial height on
         if (eta(j)-zc(j))<0;
             eta(j)=zc(j);
             h(j)=0;
         end
                                 %%right side of wave
        if (eta(j) > zmax(j)),   %%if height from datum is higher than
                                 %%the highest bed elevation
            h(j)=eta(j)-zc(j);   %%height of water = initial height from
                                 %%datum - height of bed
        elseif ((eta(j)-zc(j)) > 0);%zmin(j));%if height
                                     %from datum is smaller than
                                     %%highest cell point but bigger than
                                     %%the lowest cell point
            h(j)=0.5*(eta(j)-zmin(j));%h = 1/2 height of water from datum
                                          %- bed elevation
        else
            h(j)=0;               %%if not water height is zero
        end
        if (h(j) > tol_h),    %if water height is bigger than threshold
            u(j)=ulo;%%velocity = velocity of left side of wave%I believe
                     %%this should be u(i)=uro;
        else
```

```
            u(j)=0;
            h(j)=0;%if not water height is zero
        end
    end
    if eta(j) < zmin(j), %if water height from datum is smaller than the
        eta(j) = zmin(j);   %%bed height there is no water at this point
                            %%and eta=bed height


    end

end

%%%%%%%%%%%%%%%%%%%%%%%%----Initialize arrays----%%%%%%%%%%%%%%%%%%%%%%%%
uh=h.*u;                    %%height of water * velocity ????flow rate
matrix

deta=zeros(size(eta));    %%Initialize change in height from datum
du=zeros(size(u));        %%Initialize change in velocity
dh=zeros(size(h));
duh=zeros(size(uh));
t=0; %start time

for i=1:nt, %Begin time-marching loop
    if (iorder == 2), %for second order accuracy only

        deta = limiter(nc,beta,eta,h,tol_h);
        du = limiter(nc,beta,u,h,tol_h);
                                %%we now have found the
                                    %%difference in velocity and
                                    %%difference in
                                    %%height at the current moment in
time

        [etap, hp, up, Uhp, Sfnew]=predictor_d...
            (nc,eta,h,uh,deta,du,dz,zc,dt,dx,nm,So,u);
                                %%we now have height and velocity
                                    %%at t+(1/2dt)



        [F, amax, Hplot] = fluxes_d...
            (grav,nf,etap,up,Uhp,z,deta,du,duh,dz,zc); %Compute fluxes
                                %%these fluxes are the momentum and mass
                                %%fluxes at each cell interface


    else

        [F, amax, Hplot] =
fluxes_d(grav,nf,eta,u,uh,z,deta,du,duh,dz,zc);
    end
    [uh, h, u] = corrector_d(nc,h,uh,F,S,dx,dt,So,nm,h,u);
    eta=h+zc; %compute new free surface height
    e=eta+0.5*u.^2/grav; %compute energy in units of length (head)
    t=t+dt;
```

```
    %%%%%%%%%%------------Check the courant number----------%%%%%%%%%%%

    cr=amax*dt/dx; %compute Courant number
    fprintf(1,'%g %d\n',i,cr)

    if (cr > 1) %Stops program if Courant number exceeds one.
        break
    end

    %%%%%%%%%%%%---------------plotting loop---------------%%%%%%%%%%%%
    if (mod(i,ntplot) == 0),
        hold on
        figure('Position',get(0,'ScreenSize'))
        subplot(3,1,1)
        plot(xc,e,'r-',xc,h+zc,'b-',xc,zc,'k-')
        ylabel('$\eta$ (m)','Interpreter','Latex')
        axis([0 L -2 10])
        legend('Energy','Free Surface','Bed')
        subplot(3,1,2)
        plot(xc,u,'b-')
        xlabel('$x$ (m)','Interpreter','Latex')
        ylabel('$u$ (m/s)','Interpreter','Latex')
        axis([0 L -5 10])
        legend('Velocity')
        subplot(3,1,3)
        plot(xc,uh,'b-')
        xlabel('$x$ (m)','Interpreter','Latex')
        ylabel('q (m^3/s)')
        axis([0 L -2 10])
        legend('Discharge')
        %pause(0.005)
        saveas(gcf, num2str(t), 'bmp')
        close
    end

end
```

E.2 limiter

```
function df = limiter(nc,beta,f,h,tol_h)  %%deta or du = (nc,beta,eta) or
(nc,beta,u)

df(nc)=f(nc)-f(nc-1);   %%Ghost cell boundary Extrapolation

df(1)=f(2)-f(1);        %%Ghost cell boundary extrapolations

for i=2:nc-1,   %%for rest of cells
    df1=f(i+1)-f(i);        %%forward difference
    df2=f(i)-f(i-1);        %%backwards difference
```

```
    df(i)=limit(df1,df2,beta);      %%Call limit function
end
```

E.3 limit

```
%%Only really concerned with beta = 2
function f=limit(d1,d2,beta)
%
% 0 = first order, 1-2 = beta, 3= fromm, 4=vanleer, 5=vanalbada, 6=double
% minmod

if beta==0, %first order
    f=0;
elseif beta >= 1 & beta <= 2%beta: minmod (beta=1) and superbee (beta=2)
    if(d1*d2 < 0),
        f=0;
    else
        s=sign(d1);
        %%sign means>0=1 0=0 <0=-1
        a=abs(d1);
        %%absolute values
        b=abs(d2);
        %%absolute values
        f=s*min(max([a b]),beta*min([a b]));
        %%where f = df which is deta or du
        %%df = sign(forward
        %%diff)*the min of(max(forward diff, back diff),
        %beta*min(foward diff,back diff))
    end
elseif beta == 3 %Fromm
    f=0.5*(d1+d2);
elseif beta == 4 %vanleer
    if(d1*d2 <= 0),
        f=0;
    else
        f=2*d1*d2/(d1+d2);
    end
elseif beta ==  5 %vanalbada
     eps=1.e-20;
     f=(d1*(d2*d2+eps)+d2*(d1*d1+eps))/(d1*d1+d2*d2+2*eps);
elseif beta == 6 %double minmod
    if(d1*d2 < 0),
        f=0;
    else
        s=sign(d1);
        a=abs(d1);
```

```
        b=abs(d2);
        c=0.5*(a+b);
        f=s*min([2*a 2*b c]);
    end
end
```

E.4 predictor

```
%%The prediction step is used to predict future velocity and future
height
%the future is dt/2
%%This is know as the Hancock's method described on pg323 of
%%high resolution and non-oscillatory
%%note these equations are eq12(A) and eq13(v)without friction and
bedslope
%%terms
function [etap, hp, up, Uhp,Sfnew]=predictor_d...
    (nc,eta,h,Uh,deta,dU,dz,zc,dt,dx,nm,So,u);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%----------------------Variable Declaration--------------------------%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b=2;%width of rectangular channel
global grav
global tol_h

dh=deta-dz;

a=1;
Sfnew=zeros(nc,1);

for a = 1:10;

for j=1:1:nc,

etap(j)=eta(j)-0.5*dt/dx*dU(j); %find the eta at t+1/2
hp(j)=etap(j)-zc(j);%find the height at t+1/2


%Sfold
 if (h(j) > tol_h),%if water height is bigger than threshold
     A(j)=h(j)*b;%Area for rectangle
     P(j)=h(j)*2+b;%Perimeter for rectangle
     R(j)=A(j)/P(j);% Hydraulic radius
       Sfold(j)=nm*nm*u(j)*abs(u(j))/((R(j))^(4/3));%sf at current time
       %than g*(manning n)^2 /h(at cell j)
 else%if dry
        Sfold(j)=0; %if not water height is zero
        h(j)=0;
```

```
        A(j)=0;
        P(j)=0;
        R(j)=0;
 end

%find Velocity @ t+1/2

U_temp(j)=u(j)-((0.5*dt/dx)*((grav*dh(j))-(u(j)*dU(j))))...
    + (0.5*grav*dt*(So-(0.5*Sfold(j))-(0.5*Sfnew(j))));

if (hp(j) > tol_h),%if water height is bigger than threshold
     Ap(j)=hp(j)*b;
     Pp(j)=hp(j)*2+b;
     Rp(j)=Ap(j)/Pp(j);
       Sfnew(j)=(nm*nm*U_temp(j)*abs(U_temp(j)))/((Rp(j))^(4/3));
       %than g*(manning n)^2 /h(at cell j)

 else
        Snew(j)=0; %if not water height is zero
        hp(j)=0;
        Ap(j)=0;
        Pp(j)=0;
        Rp(j)=0;
 end


%refind velocity at t+1/2

up(j)=u(j)-(0.5*dt/dx)*(grav*dh(j)-u(j)*dU(j))...
    + 0.5*grav*dt*(So-(0.5*Sfold(j))-(0.5*Sfnew(j)));
Uhp(j)=up(j)*hp(j);%flow rate

if (hp(j) < tol_h),%if water height is smaller than threshold
       up(j)=0;
       Uhp(j)=0;
    end
end

 a=a+1;

end
```

E.5 fluxes

```
function [F, amax0,Hplot] =
fluxes_d(grav,nf,eta,u,uh,z,deta,du,duh,dz,zc)
global tol_h %dry bed threshhold
```

```
%I'm using a 2d solver in 1d, so I'm setting sn=0 and cn=1 for all cases

sn=0;
cn=1;
vl=0;           %%%these are zero because 1D solver
vr=0;           %%%These are zero because 1D solver

q=0.05;     %flow rate

hr=eta(1)-0.5*deta(1)-(z(1)); %extrapolation for the ghost cell
if hr < tol_h,  %if height is under tol_h set to 0.
    hr=0;
end



F(1,1)=q; %mass flux
F(1,2)=q^2/hr+0.5*grav*hr^2; %momentum Flux
        %%eq18 pg324
%Right Boundary : model as wall
hl=eta(nf-1)+0.5*deta(nf-1)-z(nf); %extrapolation for the ghost cell
if hl < tol_h,%if height is under tol_h set to 0.
    hl=0;
end
ul=u(nf-1)+0.5*du(nf-1); %extrapolation for the ghost cell
[fdum, amax]=solver(hl,hl,ul,-ul,vl,vr,sn,cn); %solid wall boundary
F(nf,1)=fdum(1);
F(nf,2)=fdum(2);

% ul=((u(nf-1))+0.5*du(nf-1));  Free flowing boundary conditions
% F(nf,1)=hl*ul;
% F(nf,2)=ul^2*hl+0.5*grav*hl^2;

amax0=0;
%same as above just for every internal cell
for i=2:nf-1,

hl=eta(i-1)+0.5*deta(i-1)-(z(i));
    if hl < tol_h, %if calculated h is below tol it is set to 0
    hl=0;
    end
    if hl > tol_h,
        ul=(uh(i-1)+0.5*duh(i-1))/hl; %find subcritical velocity for cell
                                      %with atleast hl wet
        if (ul*ul/(grav*hl)) > 1,     %Froude number
            ul=u(i-1)+0.5*du(i-1);    %Super critical
        end
    else
        hl = max([hl 0]);
        ul = 0;
    end
    hr=eta(i)-0.5*deta(i)-(z(i));
    if hr < tol_h,%if calculated h is below tol it is set to 0
    hr=0;
```

```
    end
    if hr > tol_h,
        ur=(uh(i)-0.5*duh(i))/hr;%find subcritical velocity for cell
                                          %with atleast hr wet
        if (ur*ur/(grav*hr)) > 1, %Froude number
            ur=u(i)-0.5*du(i);      %Super critical
        end
    else
        hr = max([hr 0]);
        ur = 0;
    end
    if (hl > tol_h || hr > tol_h),  %If the water is above both call
solver
        [fdum, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn);%find fluxes
        F(i,1)=fdum(1);%Mass Flux
        F(i,2)=fdum(2);%Momentum flux
        amax0=max([amax0 amax]);
    else
        F(i,1)=0;%Dry Bed
        F(i,2)=0;%Dry Bed
    end

end
```

E.6 solver

```
%%fluxes are calculated as described within pg290 and 291 of Finite
volume
%%model for shallow water flooding of arbitrary topography

function [F, amax]=solver(hl,hr,ul,ur,vl,vr,sn,cn)

global grav

%Compute Roe averages
%the majority of this code is used to set up variables to simplify the
end
%equations that documented below
duml=hl^0.5;
dumr=hr^0.5;
cl=(grav*hl)^0.5;
cr=(grav*hr)^0.5;
hhat=duml*dumr;
uhat=(duml*ul + dumr*ur)/(duml+dumr);

vhat=(duml*vl + dumr*vr)/(duml+dumr);
chat=(0.5*grav*(hl+hr))^0.5;
uperp=uhat*cn+vhat*sn;
dh=hr-hl;
```

```
du=ur-ul;
dv=vr-vl;
dupar=-du*sn+dv*cn;
duperp=du*cn+dv*sn;
dW=[0.5*(dh-hhat*duperp/chat); hhat*dupar; 0.5*(dh+hhat*duperp/chat)];
%%eq12

uperpl=ul*cn+vl*sn;
uperpr=ur*cn+vr*sn;
al1=uperpl-cl;
al3=uperpl+cl;
ar1=uperpr-cr;
ar3=uperpr+cr;
R=[1 0 1;
    uhat-chat*cn -sn uhat+chat*cn;
    vhat-chat*sn cn vhat+chat*sn];
%%eq11
da1=max([0 2*(ar1-al1)]);
da3=max([0 2*(ar3-al3)]);
a1=abs(uperp-chat);
a2=abs(uperp);
a3=abs(uperp+chat);

%Critical flow fix
if a1 < da1,
     a1=0.5*(a1*a1/da1+da1);
end
if a3 < da3,
     a3=0.5*(a3*a3/da3+da3);
end

%Compute interface flux
A=diag([a1 a2 a3]);
%%eq9
FL=[uperpl*hl; ul*uperpl*hl + 0.5*grav*hl*hl*cn;...
    vl*uperpl*hl + 0.5*grav*hl*hl*sn];
%%eq7
FR=[uperpr*hr; ur*uperpr*hr + 0.5*grav*hr*hr*cn;...
    vr*uperpr*hr + 0.5*grav*hr*hr*sn];
%%eq7
F=0.5*(FL + FR - R*A*dW);
%%eq8
amax=chat+abs(uperp);
%%just below eq9
```

E.7 corrector

```
function [uhnew, hnew, unew]=corrector_d...
```

```
    (nc,hold,uhold,F,dx,dt,So,nm,h,uold)
%%$----------------------Variable Declaration--------------------------%%
b=2;%channel width
global tol_h
global grav
%%$--------------------------------------------------------------------%%
a=0;
Sfnew=zeros(50);
for a = 1:10

for i=1:1:nc,
     hnew(i)=hold(i)-dt/dx*(F(i+1,1)-F(i,1));%determine new height at t+1
     Anew(i)=hnew(i)*b;%Area
     if hnew(i)<tol_h%dry bed check
         hnew(i)=0;
         Anew(i)=0;
     end

%find Friction
%Sfold
if (hold(i) > tol_h),%if water height is bigger than threshold
     Aold(i)=hold(i)*b;%find area
     P(i)=hold(i)*2+b;%find perimeter
     Rold(i)=Aold(i)/P(i);%find hydraulic radius
       Sfold(i)=nm*nm*uold(i)*abs(uold(i))/((Rold(i))^(4/3));
       %than g*(manning n)^2 /h(at cell j)
    else
        Sfold(i)=0; %if not water height is zero
        hold(i)=0;
        Aold(i)=0;
        P(i)=0;
        Rold(i)=0;
 end

 F_=F(i+1,2)-F(i,2);
         u_temp(i)=uold(i)-(dt/(2*dx)*(F_))+dt...
             /2*(grav*Aold(i)*(So-Sfold(i))+ grav*Anew(i)*(So-Sfnew(i)));

%find friction
%Sfnew
if (hnew(i) > tol_h),%if water height is bigger than threshold
     Anew(i)=hnew(i)*b;
     Pc(i)=hnew(i)*2+b;
     Rnew(i)=Anew(i)/Pc(i);
       Sfnew(i)=nm*nm*u_temp(i)*abs(u_temp(i))...
           /((Rnew(i))^(4/3));%than g*(manning n)^2 /h(at cell j)
    else
        Sfold(i)=0; %if not water height is zero
        hnew(i)=0;
        Anew(i)=0;
        Pc(i)=0;
        Rnew(i)=0;
 end
```

```
%refind velocity at t+1/2 with friction
F_=F(i+1,2)-F(i,2);
%Find U at the next time step
unew(i)=uold(i)-((dt/(2*dx))*(F_))+dt...
    /2*(grav*Aold(i)*(So-Sfold(i))+ grav*Anew(i)*(So-Sfnew(i)));

uhnew(i)=unew(i)*hnew(i);%flow rate

if hnew(i)<tol_h%dry bed check
          uhnew(i)=0;
          unew(i)=0;
          hnew(i)=0;
end
end
 a = a+1;
end
```

E.8 Dry Bed Tests

Initial Conditions											
Run No.	So	n	dt	dx	q	hl	ul	hr	ur	ε	Result
1	0.002	0.02	0.2	200	2	0.2	0.1	-3	0.1	1^{-10}	Fail
2	0.002	0.02	0.2	200	2	.2	.1	-3	.1	1^{-9}	Fail
3	0.002	0.02	0.2	200	2	.2	.1	-3	.1	1^{-8}	Fail
4	0.002	0.02	0.2	200	2	.2	.1	-3	.1	1^{-7}	Fail
5	0.002	0.02	0.2	200	2	.2	.1	-3	.1	1^{-6}	Fail
6	0.002	0.02	0.005	1000	2	.2	.1	-3	.1	1^{-6}	Pass -

7	0.002	0.02	0.025	1000	2	.2	.1	-3	.1	1^{-6}	Pass
8	0.002	0.02	0.02	1000	2	.2	.1	-3	.1	1^{-6}	Pass
9	0.002	0.02	0.2	200	2	.2	.1	-3	.1	1^{-5}	Fail
10	0.002	0.02	0.2	200	2	.2	.1	-3	.1	1^{-4}	Pass
11	0.002	0.02	0.05	1000	2	.2	.1	-3	.1	1^{-6}	Fail -
12	-0.002	0.02	0.01	1000	0.5	.2	.1	-3	.1	1^{-6}	

Initial Conditions cont…											
Run No.	So	n	dt	dx	q	hl	ul	hr	ur	ε	Result
13	-0.002	0.02	0.01	1000	0.5	0.2	0.1	-3	0.1	13	

initial v	so	t	cell	n	
0.01	2/1000	0.8	70	0.02	
readings	v	h	sf	e	z
1	0.0947	0.0005	0.0993	1.9315	1.9305
2	0.0967	0.00075	0.058	1.9318	1.9305
3	0.0976	0.001	0.0396	1.932	1.9305
4	0.0981	0.00125	0.0294	1.9323	1.9305
5	0.0985	0.0015	0.0231	1.9325	1.9305
6	0.0987	0.00175	0.0188	1.9328	1.9305
7	0.0989	0.002	0.0158	1.9333	1.9305

8	0.099	0.00225	0.0135	1.9333	1.9305
9	0.0991	0.0025	0.0117	1.9335	1.9305
10	0.0992	0.00275	0.0103	1.9338	1.9305
11	0.0993	0.003	0.0092	1.934	1.9305
12	0.0995	0.004	0.0063	1.935	1.9305
13	0.0997	0.005	0.0047	1.936	1.9305
14	0.0998	0.006	0.0037	1.937	1.9305
15	0.0998	0.007	0.003	1.938	1.9305
16	0.0999	0.008	0.0025	1.938	1.9305
17	0.1	0.009	0.0022	1.939	1.9305
18	0.1	0.01	0.0019	1.941	1.9305

initial v	so	t	cell	n
0.15	2/1000	0.8	70	0.02
readings	v	h	sf	z
1	0.1382	0.0005	0.2217	1.9305
2	0.1425	0.00075	0.1295	1.9305
3	0.1445	0.001	0.0885	1.9305
4	0.1457	0.00125	0.0658	1.9305
5	0.1464	0.0015	0.0517	1.9305
6	0.147	0.00175	0.0421	1.9305
7	0.1473	0.002	0.0353	1.9305
8	0.1473	0.00225	0.0302	1.9305
9	0.1479	0.0025	0.0263	1.9305
10	0.1481	0.00275	0.0232	1.9305
11	0.1482	0.003	0.0206	1.9305
12	0.1487	0.004	0.0141	1.9305
13	0.1489	0.005	0.0105	1.9305
14	0.1491	0.006	0.0083	1.9305
15	0.1493	0.007	0.0067	1.9305
16	0.1494	0.008	0.0057	1.9305
17	0.1495	0.009	0.0048	1.9305
18	0.1495	0.01	0.0042	1.9305

initial v	so	t	cell	n
0.05	2/1000	0.8	70	0.02
readings	v	h	sf	z
1	0.0487	0.0005	0.025	1.9305
2	0.0492	0.00075	0.0146	1.9305
3	0.0495	0.001	0.01	1.9305
4	0.0496	0.00125	0.0074	1.9305
5	0.0497	0.0015	0.0058	1.9305
6	0.0498	0.00175	0.0047	1.9305
7	0.0498	0.002	0.004	1.9305
8	0.0499	0.00225	0.0034	1.9305
9	0.0499	0.0025	0.003	1.9305
10	0.0499	0.00275	0.0026	1.9305
11	0.05	0.003	0.0023	1.9305
12	0.05	0.004	0.0016	1.9305
13	0.0501	0.005	0.0012	1.9305
14	0.0502	0.006	0.00092	1.9305
15	0.0502	0.007	0.00075	1.9305
16	0.0502	0.008	0.00063	1.9305
17	0.0503	0.009	0.00054	1.9305
18	0.0503	0.01	0.000474	1.9305