ตัวแบบและการจำลองของปัญหาการพังทลายของเขื่อน

นางสาวกนกนุช แจ่มศรี

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาคณิตศาสตร์ ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2547

ISBN 974-17-6641-6

MODEL AND SIMULATION OF DAM-BREAK PROBLEM

Miss Kanoknuch Chamsri

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science in Mathematics

Department of Mathematics

Faculty of Science

Chulalongkorn University

Academic Year 2004

ISBN 974-17-6641-6

Thesis Title	Model and Simulation of Dam-Break Problem
By	Miss. Kanoknuch Chamsri
Field of Study	Mathematics
Thesis Advisor	Associate Professor Jack Asavanant, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

..... Dean of the Faculty of Science
(Professor Piamsak Menasveta, Ph.D.)

Thesis Committee

..... Chairman
(Assistant Professor Pornchai Satravaha, Ph.D.)

..... Thesis Advisor
(Associate Professor Jack Asavanant, Ph.D.)

..... Member
(Assistant Professor Anusorn Chonwerayuth, Ph.D.)

กนกนุช แจ่มศรี : ตัวแบบและการจำลองของปัญหาการพังทลายของเขื่อน (MODEL AND SIMULATION OF DAM-BREAK PROBLEM) อ.ที่ปรึกษา : รองศาสตราจารย์ ดร. จักษ์ อัศวานันท์, จำนวนหน้า 98 หน้า. ISBN 974-17-6641-6

ปัญหาการพังทลายของเขื่อนเป็นปัญหาหนึ่งที่ดึงดูดความสนใจของนักวิจัยในสาขาคณิตศาสตร์ประยุกต์และกลศาสตร์ของไหลเป็นอย่างมาก ดังนั้น วิธีการทางคณิตศาสตร์ที่เหมาะสมที่จะช่วยในการจำลองการเคลื่อนที่ของน้ำอย่างฉับพลันหลังจากการพังทลายของเขื่อน จึงเป็นปัจจัยหนึ่งที่สำคัญ

ระบบสมการเชิงอนุพันธ์บางส่วนที่เรียกว่า Hyperbolic Systems ถูกใช้อย่างกว้างขวางในปรากฏการณ์ที่เกี่ยวข้องกับการเคลื่อนที่ของคลื่น หรือ การถ่ายทอดของสสาร สำหรับปัญหาการพังทลายของเขื่อน ระบบสมการที่ใช้โดยทั่วไปเป็นระบบสมการไม่เชิงเส้นและยากที่จะหาคำตอบที่แท้จริงโดยวิธีการทางคณิตศาสตร์ ดังนั้นการประมาณเชิงตัวเลขจึงถูกนำมาใช้โดยทั่วไป ปัจจุบันนี้มีวิธีการประมาณเชิงตัวเลขมากมายที่ถูกใช้ในการแก้ปัญหาระบบสมการนี้ และวิธีการทาง Finite Volume เป็นวิธีการหนึ่งที่มีการถูกใช้อย่างกว้างขวางและมีความถูกต้องและมีประสิทธิภาพ นอกจากนี้วิธีการที่เรียกว่า Adaptive Mesh Refinement (AMR) ยังถูกนำมาใช้ในการแก้ปัญหาเพื่อเพิ่มประสิทธิภาพในการคำนวณมากขึ้น

ภาควิชา**คณิตศาสตร์** ลายมือชื่อนิสิต . . . . . .
สาขาวิชา**คณิตศาสตร์** ลายมือชื่ออาจารย์ที่ปรึกษา . . . . . .
ปีการศึกษา **2547** ลายมือชื่ออาจารย์ที่ปรึกษาร่วม -

##4572201223 : MAJOR MATHEMATICS KEY WORDS : DAM-BREAK PROBLEM/ SAINT-VENANT EQUATION / FINITE VOLUME / ADAPTIVE MESH REFINEMENT METHOD

KANOKNUCH CHAMSRI : MODEL AND SIMULATION OF DAM-BREAK PROBLEM. THESIS ADVISOR : ASSOCIATE PROFESSOR JACK ASAVANANT, Ph.D., 98 pp. ISBN 974-17-6641-6

The dam-break analysis is one of the problems that has greatly attracted the interest of researchers in the field of applied mathematics and fluid mechanics. One of the main issues is the suitability of numerical treatment to capture the sudden movement of water after the breaking of the dam.

Hyperbolic systems of partial differential equations can be used to model a wide variety of phenomena that involve wave motion or advective transport of substances. For dam-break problem, such a system of equation is generally nonlinear which makes the problem too difficult to solve exactly. Numerical approximations are generally sought. Many numerical methods are now at hand to solve the model equations. Among these the finite volume formulation has been widely accepted in modeling strategy for its accuracy and robustness. Here, we shall improve its efficiency by including the adaptive mesh refinement method in the calculation of dam-break solution.

Department **Mathematics**
Field of study **Mathematics**
Academic year **2004**

Student's signature.....
Advisor's signature.....

ACKNOWLEDGEMENTS

I am greatly indebted to Associate Professor Dr. Jack Asavanant, my thesis advisor, for his untired offering me helpful advice during preparation and writing this thesis. I would like to thank Assistant Professor Dr. Pornchai Satravaha, and Assistant Professor Dr. Anusorn Chonverayut, the chairman and member of this comittee, respectively, for their suggestions to this thesis. Besides, I would like to thank all of my teachers, who have taught me, for my knowledge and skills. Thanks to all my friends for giving me warmth and happiness including Acting Sec. Lt. Panat Guayjarernpanishk, Mr. Pairin Suwanasri and Mr. Surachai Chroensri.

Last but not least, I would like to express my deep gratitude to my family especially my parents for their love and encouragement throughout my graduate study.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE page

CHAPTER I

Introduction

The study of free-surface flow of water in open channels has many important applications particularly in the area of river modeling. With major river engineering projects, such as flood prevention measures, becoming ever more common and ambitious, there is an increasing need to be able to model and predict the far ranging consequences on the environment as a whole of any potential project. A major part of this process is to predict the new hydraulic characteristics of the system. For example constricting the river at some point may result in an increased risk of flooding at a point upstream. The basic equations expressing hydraulic principles were formulated in the $19th$ century by de St. Venant and Boussinesq [1]. Properties of these relationships were studied in the first half of this century, but application to real river engineering projects awaited the advent of electronic computers. The hydraulic equations are also of great importance in the modeling and design of networks of artificial channals, as for example may occur in industrial plants or sewage systems [2].

The original hydraulic model of de St. Venant is written in the form of a system of two partial differential equations, known as the Saint-Venant equations. There are derived under the hypothesis that the flow is one-dimensional. One-dimensional flows do not actually exist in nature, but the equations remain valid provided the flow is approximately one-dimensional. Until recently, two or three-dimensional models have been too computationally expensive to be practical. Even now it is often prohibitively expensive to obtain the amount of survey data for a river network necessary to make use of the added realism of a higher dimensional model. For this reason the bulk of river modeling still makes use of a one-dimensional model.

The subject of this thesis is the unsteady flow problem. If discontinuous solutions are likely to occur, it is then plausible that there may be more than one solution. One way to distinguish a physically possible unsteady solution from the others is to use the entropy condition [3]. This is discussed in chapter 2. In the numerical calculations, it is important to choose numerical schemes that is not only stable but also capable of capturing the presence of discontinuities in the computation domain. This is generally known as the shock capturing schemes. In doing so, a more general integral form of the governing equations is required. This is discussed in chapter 3.

The discussion of shock capturing schemes particularly the *Roe Linearization* [3] is given in chapter 3. The Roe scheme has been applied to the Saint-Venant equations by many authors, for example in Godunov's Method and High-Resolution Method. Despite its effectiveness in modeling problem when discontinuous solutions are expected, there are still some difficulties in the calculations.

In Chapter 4 we presents a numerical method for efficiently solving a class of partial differential equations in one dimension. The method we use is called the adaptive mesh refinement [4]. It is very well suited for those equations in which the solution is highly discontinuous (sharp gradients) in only a small portion of the domain. For any given computation, one would like to set up a grid that minimizes the discretization error and maximizes the computational effifiency of the solution method. In this chapter we present a numerical method based on local grid refinements in both space and time for efficiently solving equations whose solutions are locally irregular. Our approach is to generate as many independent, refined subgrids as needed in the irregular region of the domain. The solution on each subgrid can then be approximated by standard finite volume techniques, as done on the original coarse grid. Since we are solving a time dependent problem, the irregular regions will change in time, and

thus our grids must adapt in time in response to the solution. Comparison of computational dam-break solutions from various techniques are given and discussed in chapter 5.

CHAPTER II

The Saint-Venant Equations

In this chapter the Saint-Venant equations and the characteristics of its solution are introduced. The model assumes that the flow is strictly one-dimension. To derive the governing equations for such a wave movement, we use the principles of conservation of mass and conservation of momentum for the unsteady open channal flow. Solution obtained by perturbation method [5] is also presented at the end of this chapter.

2.1. Derivation of Conservation Laws

To see how conservation laws arise from physical principles, we shall begin by deriving the equation for conservation of mass in a one-dimensional problem. Let x denote the location along the tube and $q(x,t)$ be a function defined in such a way that the total mass of fluid in any given section from x_1 to x_2 say, is given by the integral:

$$\text{mass in } [x_1, x_2] \text{ at time } t = \int_{x_1}^{x_2} q(x,t)dx. \tag{2.1.1}$$

If we assume that the wall of the tube are impermeable and that mass is neither created nor destroyed, then the mass in this one section can change only because of fluid flowing across the endpoints x_1 and x_2.

Now let $u(x,t)$ be the velocity of the fluid at the point x at time t. Then the rate of flow, or flux of fluid past this point is given by

$$\text{mass flux at } (x,t) = q(x,t)u(x,t). \tag{2.1.2}$$

From above, the rate of change of mass in $[x_1, x_2]$ is given by the difference of fluxes at x_1 and x_2:

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)dx = q(x_1,t)u(x_1,t) - q(x_2,t)u(x_2,t) \tag{2.1.3}$$

Integrating (2.1.3) in time from t_1 to t_2, we obtain an expression for the mass in $[x_1, x_2]$ at time $t_2 > t_1$ in terms of the mass at time t_1 and the total flux at each boundary during this time period:

$$\int_{x_1}^{x_2} q(x,t_2)dx = \int_{x_1}^{x_2} q(x,t_1)dx + \int_{t_1}^{t_2} q(x_1,t)u(x_1,t)dt - \int_{t_1}^{t_2} q(x_2,t)u(x_2,t)dt \tag{2.1.4}$$

To derive the differential form of the conservation law, we must now assume that $q(x,t)$ and $u(x,t)$ are differentiable functions. Then using

$$q(x,t_2) - q(x,t_1) = \int_{t_1}^{t_2} \frac{\partial}{\partial t} q(x,t)dt \tag{2.1.5}$$

and

$$q(x_2,t)u(x_2,t) - q(x_1,t)u(x_1,t) = \int_{x_1}^{x_2} \frac{\partial}{\partial x}(q(x,t)u(x,t))dx \tag{2.1.6}$$

in (2.1.4) gives

$$\int_{t_1}^{t_2}\int_{x_1}^{x_2} \left\{ \frac{\partial}{\partial t} q(x,t) + \frac{\partial}{\partial x}(q(x,t)u(x,t)) \right\} dxdt = 0. \tag{2.1.7}$$

Since this must hold for any section $[x_1, x_2]$ and over any time interval $[t_1, t_2]$, we conclude that in fact the integrand in (2.1.7) must be identically zero, i.e.,

$$q_t + (qu)_x = 0 \tag{2.1.8}$$

This is the desired differential form of the conservation law for the conservation of mass.

The conservation law (2.1.8) can be solved in isolation only if the velocity $u(x,t)$ is known a priori or as a function of $q(x,t)$. If it is, then qu is a function of q alone, say $qu = f(q)$, and the conservation of mass equation (2.1.8) becomes a scalar conservation law for q,

$$q_t + f_x(q) = 0 \tag{2.1.9}$$

More typically the equation (2.1.8) must be solved in conjuction with equations for the conservation of momentum. The product $q(x,t)u(x,t)$ gives the density of momentum, in the sense that the integral of qu between any two points x_1 and x_2 yields the total momentum in this interval, and this can change only due to the flux of momentum through the endpoints of the interval. For any density function qu this flux has the form $(qu)u$, as we have already seen at the beginning of this chapter, and so for the momentum qu the contribution to the flux is $(qu)u = qu^2$.

In general the pressure gives the microscopic momentum flux that must be added to the advective flux qu^2 to obtain the total momentum flux,

$$\text{momentum flux } = qu^2 + P, \tag{2.1.10}$$

where P is the fluid pressure (as described below), leading to the integral conservation law

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)u(x,t)dx = -[qu^2 + P]_{x_1}^{x_2}.$$

Following the similar derivation of mass conservation, we arrive at the differential form of the momentum equation

$$(qu)_t + (qu^2 + P)_x = 0. \tag{2.1.11}$$

2.2. The Saint-Venant Equations

To derive the one-dimensional Saint-Venant equations, we consider fluid flow along a channel of unit width.

We now assume that the fluid is incompressible (the density ρ is constant). Let the fluid depth at each cross section be denoted by $h(x,t)$. The total mass in $[x_1, x_2]$ at time t is

$$\int_{x_1}^{x_2} \rho h(x,t)dx,$$

and the mass flux at (x,t) is $\rho u(x,t)h(x,t)$. The constant ρ drops out of the conservation of mass equation, which then takes the familiar form $\big($compare (2.1.8)$\big)$

$$h_t + (hu)_x = 0. \tag{2.2.1}$$

The quantity hu is often called the discharge Q which measures the flow rate of water at a certain point in space.

The conservation of momentum equation also takes the same form $\big($see (2.1.11)$\big)$,

$$(\rho hu)_t + (\rho hu^2 + P)_x = 0, \tag{2.2.2}$$

but now P is determined from a hydrostatic law [3], stating that the pressure at distance $(h-y)$ below the surface is $\rho g(h-y)$, where g is the gravitational constant. Integrating this vertically from $y = 0$ to $y = h(x,t)$ yields the total pressure at (x,t).

The form of pressure term in the momentum flux is

$$P = \frac{1}{2}\rho g h^2. \tag{2.2.3}$$

Using this in (2.2.2) and cancelling ρ gives

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x = 0. \tag{2.2.4}$$

The equations (2.2.1) and (2.2.4) are the differential form of the Saint-Venant equations.

2.3. Source Terms

In some situations the change of $\int_{x_1}^{x_2} q(x,t)dx$ could be due to source or sink contained within the section in addition to the endpoint fluxes. Let us denote the density function for such a source by $\psi(q,x,t)$. (Negative values of ψ correspond to a sink). Then the equation (2.2.1) and (2.2.4) becomes

$$\frac{d}{dt}\int_{x_1}^{x_2} q(x,t)dx = -\int_{x_1}^{x_2} \frac{\partial}{\partial x} f(q(x,t))dx + \int_{x_1}^{x_2} \psi(q(x,t),x,t)dx.$$

This leads to the partial differential equation (PDE)

$$q_t(x,t) + f_x(q(x,t)) = \psi(q(x,t),x,t). \tag{2.3.1}$$

In general, the external force term constitutes of the bottom friction (F) and the bottom slope (G). Here

$$F = \rho g h S_f$$

and S_f is associated with the bed friction and can be represented by the empirical Manning law [6]:

$$S_f = \frac{u\,|u|\,n^2}{h^{4/3}} \tag{2.3.2}$$

where n is a roughness coefficient. The function G is given by

$$G = \rho g h S_0$$

where S_0 is the bed slope. Therefore, ψ is composed of the friction force and the gravity force. The function ψ is given by

$$\psi = gh(S_0 - S_f).$$

Finally, the complete form of the momentum equation can be written as

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x = gh(S_0 - S_f). \tag{2.3.3}$$

2.4. Characteristic Speeds

The Saint-Venant equations are classified as a hyperbolic system of partial differential equations. To see this we write the system in the vector form

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{q})}{\partial x} = 0, \tag{2.4.1}$$

where

$$\mathbf{q} = \begin{bmatrix} h \\ hu \end{bmatrix} \quad \text{and} \quad \mathbf{f} = \begin{bmatrix} hu \\ hu^2 + 1/2gh^2 \end{bmatrix}.$$

Let q^1 and q^2 be the two components of $\mathbf{q}$ and rewrite the flux $\mathbf{f}$ as

$$\mathbf{q} = \begin{bmatrix} h \\ hu \end{bmatrix} = \begin{bmatrix} q^1 \\ q^2 \end{bmatrix},$$

$$\mathbf{f} = \begin{bmatrix} hu \\ hu^2 + 1/2gh^2 \end{bmatrix} = \begin{bmatrix} q^2 \\ (q^2)^2/q^1 + 1/2g(q^1)^2 \end{bmatrix}.$$

For smooth solution, these equations can equivalently be written in the quasilinear form

$$\mathbf{q}_t + \mathbf{f}'(\mathbf{q})\mathbf{q}_\mathbf{x} = 0,$$

where the Jacobian matrix $\mathbf{f}'(\mathbf{q})$ is

$$\mathbf{f}'(\mathbf{q}) = \begin{bmatrix} 0 & 1 \\ -\left(\frac{q^2}{q^1}\right)^2 + gq^1 & 2\frac{q^2}{q^1} \end{bmatrix} \tag{2.4.2}$$

$$= \begin{bmatrix} 0 & 1 \\ -u^2 + gh & 2u \end{bmatrix} \tag{2.4.3}$$

$$= \begin{bmatrix} 0 & 1 \\ c^2 - u^2 & 2u \end{bmatrix}, \tag{2.4.4}$$

where $c = \sqrt{gh}$. The eigenvalues of $\mathbf{f}'(\mathbf{q})$ are

$$\lambda^1 = u - \sqrt{gh} \quad \text{and} \quad \lambda^2 = u + \sqrt{gh}, \tag{2.4.5}$$

with the corresponding eigenvectors

$$\mathbf{r^1} = \begin{bmatrix} 1 \\ u - \sqrt{gh} \end{bmatrix} \quad \text{and} \quad \mathbf{r^2} = \begin{bmatrix} 1 \\ u + \sqrt{gh} \end{bmatrix}.$$

The λ^i, $i = 1, 2$, denotes the characteristic speeds [3] of the solution.

2.5. Discontinuous Solution

Though the differential form (2.4.1) is not as general as the integral form, it posseses some popularity in practical use. However, The differential form can break down due to the formation of a shock. When this happens one must return to the integral form of the Saint-Vanant equations so that appropriate conditions can be used to describe the shock. These conditions are known as the Rankine-Hugoniot conditions (see [3]),

$$h_r u_r - h_l u_l = s(h_r - h_l) \tag{2.5.1}$$

$$(h_r u_r^2 + 1/2gh_r^2) - (h_l u_l^2 + 1/2gh_l^2) = s(h_r u_r - h_l u_l), \tag{2.5.2}$$

where the subsripts l and r denote values of the quantities on the left and right of the discontinuity respectively (e.g. $h_l = h(x-, t)$ and $h_r = h(x+, t)$), and s is the shock speed.

This is to ensure that the problem has a unique weak solution that is physically correct. However, these conditions are too difficult to implement in the system. There are other alternative conditions that can be applied directly to weak solutions of hyperbolic equations to examine if they are physically admissible. Such conditions are sometimes calles *entropy conditions*.

Entropy Conditions (Lax) [3] A discontinuity separating states q_l and q_r, propagating at speed s, satisfies the Lax entropy condition if there is an index p such that

$$\lambda^p(q_l) > s > \lambda^p(q_r), \tag{2.5.3}$$

so that p-characteristics are impinging on the discontinuity, while the other characteristics are crossing the discontinuity,

$$\lambda^j(q_l) < s \quad \text{and} \quad \lambda^j(q_r) < s \quad \text{for} \quad j<p,$$

$$\lambda^j(q_l) > s \quad \text{and} \quad \lambda^j(q_r) > s \quad \text{for} \quad j>p.$$

In this definition we assume the eigenvalues are ordered so that $\lambda^1 < \lambda^2 < ... < \lambda^m$ in each state.

2.6. Riemann Problem

In this section we will further explore the characteristic structure of linear hyperbolic systems of equations. In particular, we will study solutions to the *Riemann problem* [3], consisting of a piecewise constant function with a single jump discontinuity. This simple problem plays a very important role in understanding the structure of more general solutions. It is also a fundamental building block for the finite volume methods discussed in this thesis.

Consider a linear hyperbolic systems of the form

$$\mathbf{q}_t + \mathbf{A}\mathbf{q}_x = 0. \tag{2.6.1}$$

The problem is hyperbolic if $\mathbf{A} \in \mathbb{R}^{m\times m}$ is diagonalizable with real eigenvalues. We write

$$\mathbf{A} = \mathbf{R}\mathbf{\Lambda}\mathbf{R^{-1}}, \tag{2.6.2}$$

where $\mathbf{R}$ is the matrix of eigenvectors. Then introducing the new variables

$$\mathbf{w} = \mathbf{R}^{-1}\mathbf{q}$$

allows us to reduce the system (2.6.1) to

$$\mathbf{w}_t + \mathbf{\Lambda}\mathbf{w}_x = 0, \tag{2.6.3}$$

which is a set of m decoupled advection equations. Here $\mathbf{A}$ is assumed to be constant.

Consider the Cauchy problem for the constant-coefficient system (2.6.1), in which we are given the data

$$\mathbf{q}(x,0) = \mathbf{q}^0(x) \qquad \text{for} \qquad -\infty < x < \infty.$$

From this data we can compute

$$\mathbf{w}^0(x) \equiv \mathbf{R}^{-1}\mathbf{q}^0(x)$$

for the system (2.6.3). The pth equation of (2.6.3) is the advection equation

$$w_t^p + \lambda^p w_x^p = 0, \tag{2.6.4}$$

with solution

$$w^p(x,t) = w^p(x - \lambda^p t, 0) = (w^p)^0(x - \lambda^p t).$$

Having computed all components $w^p(x,t)$, we can write

$$\mathbf{q}(x,t) = \mathbf{R}\mathbf{w}(x,t) \tag{2.6.5}$$

which gives the solution to the original problem. Note that we can write (2.6.5) as

$$\mathbf{q}(x,t) = \sum_{p=1}^{m} w^p(x,t)\mathbf{r}^p, \tag{2.6.6}$$

so that the vector $\mathbf{q}(x,t)$ can be viewed as some linear combination of the eigenvectors $\mathbf{r}^1, ..., \mathbf{r}^m$ at each point in space-time, and hence as a superposition of waves propagating at different velocities λ^p. The requirements of hyperbolicity insure that these m vectors are linearly independent and hence every vector $\mathbf{q}$ has a unique representation in this form.

The *Riemann problem* consists of the hyperbolic equation together with special initial data that is piecewise constant with a single jump discontinuity,

$$\mathbf{q}^0(x) = \begin{cases} \mathbf{q}_l, & \text{if } x < 0 \ , \\ \mathbf{q}_r, & \text{if } x > 0 \ . \end{cases}$$

Let us take the scalar advection equation $q_t - \bar{u}q_x = 0$, as an example. The coefficient matrix is the scalar value $\bar{u}$. The single eigenvalue is $\lambda^1 = \bar{u}$, and we can choose the eigenvector to be $r^1 = 1$. The solution to the Riemann problem consists of the discontinuity $q_r - q_l$ propagating at speed $\bar{u}$, along the characteristic, and the solution is $q(x,t) = q^0(x - \bar{u}t)$.

For the Riemann problem we can simplify the notation if we decompose $\mathbf{q}_l$ and $\mathbf{q}_r$ as

$$\mathbf{q}_l = \sum_{p=1}^{m} w_l^p \mathbf{r}^p \quad \text{and} \quad \mathbf{q}_r = \sum_{p=1}^{m} w_r^p \mathbf{r}^p. \tag{2.6.7}$$

The pth advection equation (2.6.4) has Riemann data

$$(w^p)^0(x) = \begin{cases} w_l^p, & \text{if } x < 0 \ , \\ w_r^p, & \text{if } x > 0 \ , \end{cases} \tag{2.6.8}$$

and this discontinuity simply propagates with speed λ^p. So

$$w^p(x,t) = \begin{cases} w_l^p, & \text{if } x - \lambda^p t < 0 \ , \\ w_r^p, & \text{if } x - \lambda^p t > 0 \ , \end{cases} \tag{2.6.9}$$

If we let $P(x,t)$ be the maximum value of p for which $x - \lambda^p t > 0$, then

$$\mathbf{q}(x,t) = \sum_{p=1}^{P(x,t)} w_r^p \mathbf{r}^p + \sum_{p=P(x,t)+1}^{m} w_l^p \mathbf{r}^p, \tag{2.6.10}$$

which we will write more concisely as

$$\mathbf{q}(x,t) = \sum_{p:\lambda^p < x/t} w_r^p \mathbf{r}^p + \sum_{p:\lambda^p > x/t} w_l^p \mathbf{r}^p. \tag{2.6.11}$$

The determination of $\mathbf{q}(x,t)$ at a given point (X,T) is illustrated in Figure 2.1. In the case shown, $w^1 = w_r^1$ while $w^2 = w_l^2$ and $w^3 = w_l^3$. The solution at the point illustrated is thus

$$\mathbf{q}(X,T) = w_r^1 \mathbf{r}^1 + w_l^2 \mathbf{r}^2 + w_l^3 \mathbf{r}^3. \tag{2.6.12}$$

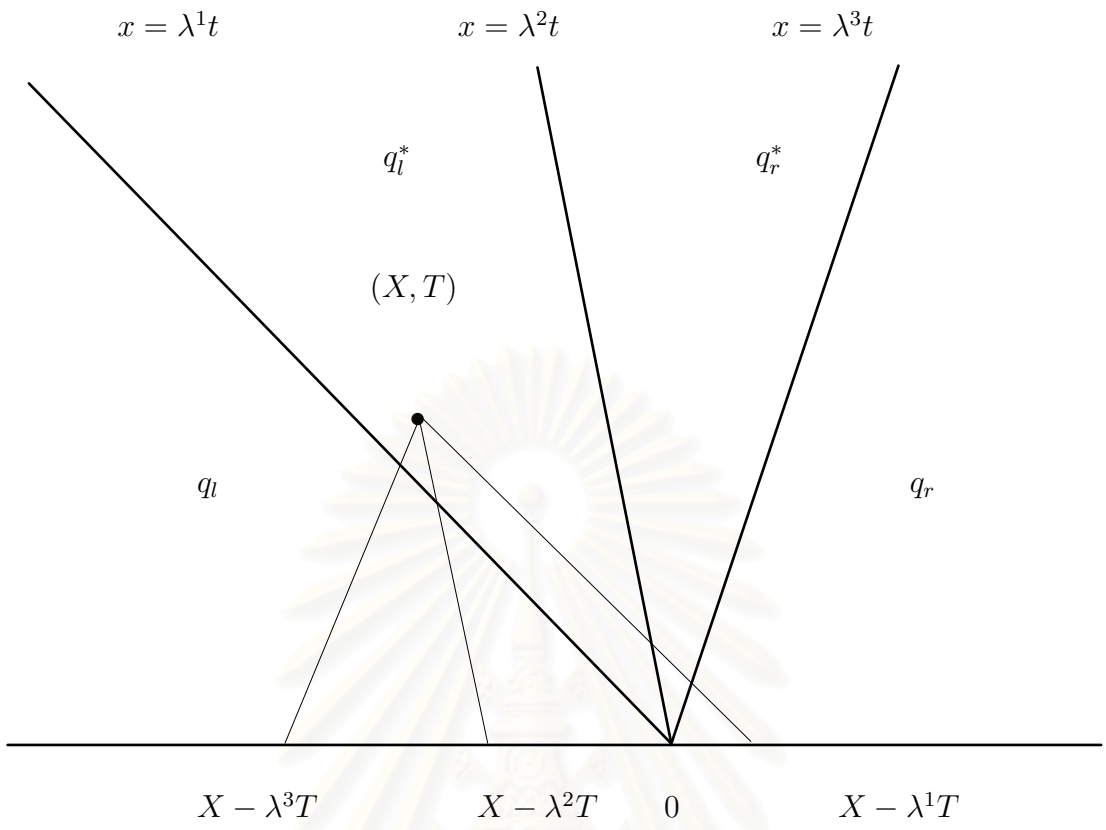

Fig 2.1 Construction of the solution to the Riemann problem at (X, T).

The solution is constant in each of the wedges as shown in Figure 2.1. Across the pth characteristic the solution jumps with the jump in $\mathbf{q}$ given by

$$(w_r^p - w_l^p)\mathbf{r}^p \equiv \alpha^p \mathbf{r}^p. \tag{2.6.13}$$

Note that this jump in $\mathbf{q}$ is an eigenvector of the matrix A (being a scalar multiple of $\mathbf{r}^p$). This is an extremely important fact, and a generalization of this statement is what will allow us to solve the Riemann problem for nonlinear systems of equations. This condition, called the Rankine-Hugoniot jump condition, will be derived from the integral form of the conservation law and will hold across any propagating

discontinuity. Since

$$\mathbf{q}_r - \mathbf{q}_l = (\mathbf{q}^*{}_l - \mathbf{q}_l) + (\mathbf{q}^*{}_r - \mathbf{q}^*{}_l) + (\mathbf{q}_r - \mathbf{q}^*{}_r) \tag{2.6.14}$$

$$= \alpha^1\mathbf{r}^1 + \alpha^2\mathbf{r}^2 + \alpha^3\mathbf{r}^3, \tag{2.6.15}$$

$$\mathbf{R}\alpha = \mathbf{q}_r - \mathbf{q}_l \quad \text{for the vector } \alpha. \tag{2.6.16}$$

Then $\alpha = \mathbf{R}^{-1}(\mathbf{q}_r - \mathbf{q}_l)$.

Hence the vector α has components $\alpha^p = \mathbf{l}^p(\mathbf{q}_r - \mathbf{q}_l)$, where $\mathbf{l}^p$ is the pth row of $\mathbf{R}^{-1}$. Let $\mathcal{W}^p = \alpha^p\mathbf{r}^p$. Then

$$\mathbf{q}(x,t) = \mathbf{q}_l + \sum_{p:\lambda^p<x/t} \mathcal{W}^p \tag{2.6.17}$$

$$= \mathbf{q}_r - \sum_{p:\lambda^p\geq x/t} \mathcal{W}^p. \tag{2.6.18}$$

2.7. Perturbation Method

Nowaday, there are various mathematical techniques that can be used to solve for approximate solutions to a class of initial and boundary value problems for partial differential equations. However, when we expect the dependence of solutions on a small parameter, perturbation method is often one of the choices to speaking, this method is used when a small parameter occurs in the given equation or data for the problem. The solution is represented by a power of the small parameter. This expansion is then substituted into the governing equations and the boundary conditions.

Perturbation techniques can also be used to replace given equations by simpler ones whose solutions contain many features of the solutions of the original problem. This is especially important for nonlinear equations where perturbation methods are

used to linearize the problem or may still be possible to replace the given equation or system by a simpler nonlinear equation.

The type of problem considered in this chapter belongs in the category of problems concerned with motions in their early stages after initial impulses have been applied. A typical example is the motion of the water in a dam when the dam is suddenly broken. Consequently we choose the quantities a, b, and t as independent variables, with a and b representing Cartesian coordinates of the initial positions of fluid particle at the time $t = 0$. We denote $X(a, b; t)$ and $Y(a, b; t)$, as the displacement of fluid particle and $p(a, b; t)$ as the pressure.

From Newton's second law, the equations of motion are

$$X_{tt} = -\frac{1}{\rho} p_X \tag{2.7.1}$$

$$Y_{tt} = -\frac{1}{\rho} p_Y - g. \tag{2.7.2}$$

All subscripts here denote the differentiation. We assume gravity to be the only external force.

The results (see Appendix B) for the specific case of a dam 70 meters high are shown in Figure 2.2.

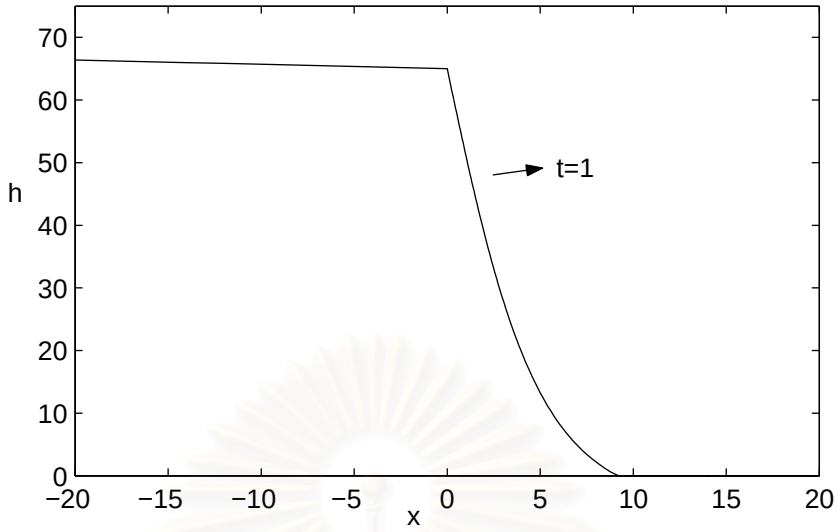

Figure 2.2 Free water surface after the breaking of a dam

CHAPTER III

Shock Capturing Methods

The Saint-Vanant equations form a system of conservation laws of hyperbolic type. Such systems of equations occur frequently in applied mathematics and so much effort has been put into developing numerical methods for their solutions.

For a numerical scheme to be of practical use, the equation must be written in what is known as conservation form and such schemes are known as conservative. We start this chapter by defining this property which leads to the approach of Godunov [3]. In addition to requiring that a scheme be conservative, we require that a scheme be stable in the presence of discontinuities, and such schemes are known as shock capturing schemes. Later we discuss the subject of stability and the extension of shock capturing schemes to higher order accuracy by the use of limiter functions.

Consider the system of equations

$$\frac{\partial \mathbf{q}}{\partial t} + \frac{\partial \mathbf{f}(\mathbf{q})}{\partial x} = 0. \tag{3.0.3}$$

We shall consider numerical approximations on a uniform grid in $x - t$ space, with Δx and Δt denoting the grid spacing in space and time respectively. The nodes are labelled by the indices i and n with positions given by $(x_i, t_n) = (i\Delta x, n\Delta t)$. It is normal in the theory of numerical methods to let $\mathbf{Q}_i^n$ denote the value of the approximation to the exact solution at the particular grid point, i.e. $\mathbf{Q}_i^n \approx \mathbf{q}(x_i, t_n)$. However, for modeling the systems of conservation laws, it is more appropriate for $\mathbf{q}_i^n$ to denote the value of the approximation to the cell average of the exact solution.

We therefore have

$$\mathbf{Q}_i^n \approx \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{q}(x,t_n)dx. \tag{3.0.4}$$

3.1. The Conservation Form

The system (3.0.3) is of hyperbolic type if the Jacobian of the function $\mathbf{f}$ has all real eigenvalues and has a full set of linearly independent eigenvectors. The early attempts in numerically modelling this type of equation were for the linear case and simply involved replacing the derivatives by finite difference formula. This led to many classical schemes such as Lax-Wendroff [3]. Such schemes were successful for solving problems with smooth solutions but failed miserably for discontinuous solutions. Certain schemes such as the one-sided second order scheme (Beam-Warming) could compute discontinuous solutions successfully, but now difficulties were encountered when generalising these to the nonlinear problem. Although convincing discontinuous solutions were obtained, more often than not at further inspection the discontinuities were found to move at the wrong speed. The fact that a numerical solution does not satisfy the appropriate Rankine-Hugoniot conditions indicates that the scheme does not approximate consistently the underlying conservation law. Examples of this behavior can be found in [3]

This requirement is that the scheme can be written in conservation form. A scheme is in *conservation form* if it is written in the form

$$\frac{\mathbf{Q}_i^{n+1}-\mathbf{Q}_i^n}{\Delta t} + \frac{\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n}{\Delta x} = 0, \tag{3.1.1}$$

where

$$\mathbf{F}_{i+1/2}^n = \mathbf{F}(\mathbf{q}_{i+k}^n, ..., \mathbf{q}_{i+1}^n, \mathbf{q}_i^n, ..., \mathbf{q}_{i-k+1}^n)$$

for some k. For this scheme to be consistent with the differential equation (3.0.3) the function $\mathbf{F}$ is required to satisfy the consistency condition

$$\mathbf{F}(\mathbf{q}) = \mathbf{f}(\mathbf{q}), \tag{3.1.2}$$

for all $\mathbf{q}$.

The function $\mathbf{F}$ is called the numerical flux function since it approximates the time average flux across the cell interfaces, i.e.

$$\mathbf{F}^n_{i+1/2} \approx \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{q}(x_{i+1/2}, t))dt. \tag{3.1.3}$$

To see why this is true, we substitute

$$\mathbf{Q}^n_i = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{q}(x, t_n)dx, \tag{3.1.4}$$

$$\mathbf{F}^n_{i-1/2} = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{q}(x_{i-1/2}, t))dt, \tag{3.1.5}$$

into the scheme (3.1.1) and multiply by a factor $\Delta x \Delta t$ to obtain:

$$\int_{x_{i-1/2}}^{x_{i+1/2}} [\mathbf{q}]_{t_n}^{t_{n+1}} dx + \int_{t_n}^{t_{n+1}} [\mathbf{f}(\mathbf{q})]_{x_{i-1/2}}^{x_{i+1/2}} dt = 0. \tag{3.1.6}$$

This is the integral form of the conservation law over the rectangle $[x_{i-1/2}, x_{i+1/2}] \times [t_n, t_{n+1}]$.

3.2. The Godunov's Method

We observe that the numerical solution satisfies the integral form of the conservation law exactly if (3.1.4) and (3.1.5) hold. This constitutes the so-called Godunov's method and can be explained in the following *algorithm*:

(1) Replace the initial data $\mathbf{q^0}$ by a piecewise constant function where the constant value in each cell is given by the cell average.

(2) Use the piecewise constant function and the formula (3.1.5) to compute the numerical fluxes across each cell interface.

(3) Use the numerical scheme to compute the cell averages at the next time level, hence defining a new piecewise constant function.

(4) Repeat from step 2.

This appears to be a relatively straightforward strategy for stepping forward the numerical solution in time. The crucial step is (2) which involves computing the time average flux at each cell interface. Recall the formula (3.1.5),

$$\mathbf{F}^n_{i-1/2} = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{q}(x_{i-1/2}, t))dt.$$

In general the function $\mathbf{q}(x_{i-1/2}, t)$ varies with t, and we certainly do not know this variation of the exact solution. However, we can compute this integral exactly if we replace $\mathbf{q}(x, t)$ by the function $\tilde{\mathbf{q}}^n(x, t)$. For example, $\tilde{\mathbf{q}}^n(x_{i-1/2}, t)$ is constant over the time interval $t_n < t < t_{n+1}$. Denote this value by $\mathbf{Q}^{\downarrow}_{i-1/2} = \mathbf{q}^{\downarrow}(\mathbf{Q}^n_{i-1}, \mathbf{Q}^n_i)$. This suggests us to define the numerical flux $\mathbf{F}^n_{i-1/2}$ by

$$\mathbf{F}^n_{i-1/2} = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{f}(\mathbf{q}^{\downarrow}(\mathbf{Q}^n_{i-1}, \mathbf{Q}^n_i))dt \tag{3.2.1}$$

$$= \mathbf{f}(\mathbf{q}^{\downarrow}(\mathbf{Q}^n_{i-1}, \mathbf{Q}^n_i)). \tag{3.2.2}$$

As usual, $\mathbf{q}^{\downarrow}(\mathbf{q}_l, \mathbf{q}_r)$ denotes the solution to the Riemann problem between states $\mathbf{q}_l$ and $\mathbf{q}_r$, evaluated along $x/t = 0$.

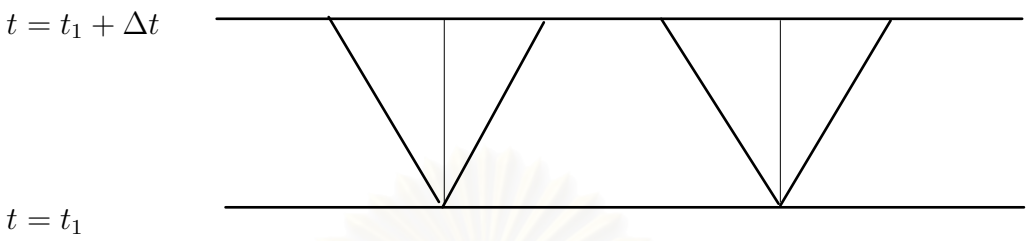

Figure 3.1 Solving the Riemann problems at each interface (vertical lines) for the Godunov's method with Courant number less than 1/2 (there is no interaction of waves).

In Figure 3.1 the time step is taken to be small enough that there is no interaction of waves from the neighboring Riemann problems. In order to use the flux $\mathbf{F}^n_{i-1/2} = \mathbf{f}(\mathbf{q}^{\downarrow}(\mathbf{Q}^n_{i-1}, \mathbf{Q}^n_i))$, it is only necessary that the edge value $\tilde{\mathbf{q}}^n(x_{i-1/2}, t)$ remains constant in time over the entire time step. If s_{max} represents the largest wave speed that is encountered, then on a uniform grid with the cell interfaces distance Δx apart, we must require

$$\frac{s_{max}\Delta t}{\Delta x} \leq 1 \tag{3.2.3}$$

in order to insure that the formula $\mathbf{F}^n_{i-1/2} = \mathbf{f}(\mathbf{q}^{\downarrow}(\mathbf{Q}^n_{i-1}, \mathbf{Q}^n_i))$ is valid. Note that this is precisely the CFL (Courant-Friedrichs Laws) condition required for stability of this three-point method. In general $\frac{s_{max}\Delta t}{\Delta x}$ is called the *Courant number*.

Godunov's method can be implemented in the wave propagation form as

$$\mathbf{Q}^{n+1}_i = \mathbf{Q}^n_i - \frac{\Delta t}{\Delta x}(\mathcal{A}^+\Delta\mathbf{Q}_{i-1/2} + \mathcal{A}^-\Delta\mathbf{Q}_{i+1/2}). \tag{3.2.4}$$

and we might take one of the two different fluctuations by

(1) Define

$$\mathcal{A}^{-}\Delta\mathbf{Q}_{i-1/2} = \mathbf{f}(\mathbf{Q}^{\downarrow}_{i-1/2}) - \mathbf{f}(\mathbf{Q}_{i-1}), \tag{3.2.5}$$

$$\mathcal{A}^{+}\Delta\mathbf{Q}_{i-1/2} = \mathbf{f}(\mathbf{Q}_{i}) - \mathbf{f}(\mathbf{Q}^{\downarrow}_{i-1/2}), \tag{3.2.6}$$

where

$$\mathbf{Q}^{\downarrow}_{i-1/2} = \mathbf{Q}_{i-1/2}(0) = \mathbf{Q}_{i-1} + \sum_{p:s^{p}_{i-1/2}<0} \mathcal{W}^{p}_{i-1/2} \tag{3.2.7}$$

is the value along the cell interface. Or

(2) Use the waves and speeds from the approximate Riemann solution to define

$$\mathcal{A}^{-}\Delta\mathbf{Q}_{i-1/2} = \sum_{p=1}^{M_w} (s^{p}_{i-1/2})^{-}\mathcal{W}^{p}_{i-1/2}, \tag{3.2.8}$$

$$\mathcal{A}^{+}\Delta\mathbf{Q}_{i-1/2} = \sum_{p=1}^{M_w} (s^{p}_{i-1/2})^{+}\mathcal{W}^{p}_{i-1/2}, \tag{3.2.9}$$

where M_w is the maximum value of p and s^p is a characteristic speed obtained from Linearized Riemann Solvers. The wave $\mathcal{W}^{p}_{i-1/2}$ is discussed in section 3.4.

These two approaches yield the same result if special care is taken in defining the approximate solution. (The first approach is always conservative, since it is based on an interface flux.)

3.3. Linearized Riemann Solvers

One natural approach to defining an approximate Riemann solution is to replace the nonlinear problem $\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0$ by some linearized problem defined locally at each cell interface,

$$\hat{\mathbf{q}}_t + \hat{\mathbf{A}}_{i-1/2}\hat{\mathbf{q}}_x = 0. \tag{3.3.1}$$

The matrix $\hat{\mathbf{A}}_{i-1/2}$ is chosen to be some approximation to $\mathbf{f}'(\mathbf{q})$ valid in a neighborhood of the data $\mathbf{Q}_{i-1}$ and $\mathbf{Q}_i$. The matrix $\hat{\mathbf{A}}_{i-1/2}$ should satisfy the following condition:

$$\hat{\mathbf{A}}_{i-1/2} \quad \text{is diagonalizable with real eigenvalues,} \tag{3.3.2}$$

so that (3.3.1) is hyperbolic, and

$$\hat{\mathbf{A}}_{i-1/2} \longrightarrow \mathbf{f}'(\bar{\mathbf{q}}) \quad \text{as} \quad \mathbf{Q}_{i-1} \quad \text{and} \quad \mathbf{Q}_i \longrightarrow \bar{\mathbf{q}}, \tag{3.3.3}$$

so that the method is consistent with the original conservation law. The approximate Riemann solution then consists of m waves proportional to the eigenvectors $\hat{\mathbf{r}}^p_{i-1/2}$ of $\hat{\mathbf{A}}_{i-1/2}$, propagating with speeds $s^p_{i-1/2} = \hat{\lambda}^p_{i-1/2}$ given by the eigenvalues. Since this is a linear problem, the Riemann problem can generally be solved more easily than the original nonlinear problem. Often, there are simple closed-form expressions for the eigenvectors and hence for the solution, which is obtained by solving the linear system

$$\mathbf{Q}_i - \mathbf{Q}_{i-1} = \sum_{p=1}^{m} \alpha^p_{i-1/2}\hat{\mathbf{r}}^p_{i-1/2} \tag{3.3.4}$$

for the coefficients $\alpha^p_{i-1/2}$ with $\mathcal{W}^p_{i-1/2} = \alpha^p_{i-1/2}\hat{\mathbf{r}}^p_{i-1/2}$.

We might take, for example,

$$\hat{\mathbf{A}}_{i-1/2} = \mathbf{f}'(\hat{\mathbf{Q}}_{i-1/2}),$$

where $\hat{\mathbf{Q}}_{i-1/2}$ is some average of $\mathbf{Q}_{i-1}$ and $\mathbf{Q}_i$. One of the most accepted methods is called the Roe linearization described in the next section.

3.3.1. Roe Linearization. If $\mathbf{Q}_{i-1}$ and $\mathbf{Q}_i$ are connected by a single wave (shock or contact discontinuity), then

$$\mathbf{f}(\mathbf{Q}_i) - \mathbf{f}(\mathbf{Q}_{i-1}) = s(\mathbf{Q}_i - \mathbf{Q}_{i-1}),$$

where s is the wave speed. If this is also to be a solution to the linearized Riemann problem, then we must have

$$\hat{\mathbf{A}}_{i-1/2}(\mathbf{Q}_i - \mathbf{Q}_{i-1}) = s(\mathbf{Q}_i - \mathbf{Q}_{i-1}).$$

Combining these, we obtain the condition

$$\hat{\mathbf{A}}_{i-1/2}(\mathbf{Q}_i - \mathbf{Q}_{i-1}) = \mathbf{f}(\mathbf{Q}_i) - \mathbf{f}(\mathbf{Q}_{i-1}). \tag{3.3.5}$$

Roe introduced a parameter vector $\mathbf{z}(\mathbf{q})$, a change of variables that leads to integrals that are easy to evaluate. We assume that this mapping is invertible and hence $\mathbf{q}(\mathbf{z})$ is known. Using this mapping, we can also view $\mathbf{f}$ as a function of $\mathbf{z}$, and write $\mathbf{f}(\mathbf{z})$ as a shorthand for $\mathbf{f}(\mathbf{q}(\mathbf{z}))$. Let

$$\mathbf{z}(\xi) = \mathbf{Z}_{i-1} + (\mathbf{Z}_i - \mathbf{Z}_{i-1})\xi, \tag{3.3.6}$$

where $\mathbf{Z}_j = \mathbf{z}(\mathbf{Q}_j)$ for $j = i-1, i$. It follows that $\mathbf{z}'(\xi) = \mathbf{Z}_i - \mathbf{Z}_{i-1}$ is independent of ξ. By the invertibity of $\mathbf{f}$, we can write

$$\begin{aligned}\mathbf{f}(\mathbf{Q}_i) - \mathbf{f}(\mathbf{Q}_{i-1}) &= \int_0^1 \frac{d\mathbf{f}(\mathbf{z}(\xi))}{d\xi} d\xi \\ &= \int_0^1 \frac{d\mathbf{f}(\mathbf{z}(\xi))}{d\mathbf{z}} \mathbf{z}'(\xi) d\xi\end{aligned}$$

$$= \left[\int_0^1 \frac{d\mathbf{f}(\mathbf{z}(\xi))}{d\mathbf{z}} d\xi\right] (\mathbf{Z}_i - \mathbf{Z}_{i-1}). \tag{3.3.7}$$

Moreover,

$$\begin{aligned}\mathbf{Q}_i - \mathbf{Q}_{i-1} &= \int_0^1 \frac{d\mathbf{q}(\mathbf{z}(\xi))}{d\xi} d\xi \\ &= \int_0^1 \frac{d\mathbf{q}(\mathbf{z}(\xi))}{d\mathbf{z}} \mathbf{z}'(\xi) d\xi \\ &= \left[\int_0^1 \frac{d\mathbf{q}(\mathbf{z}(\xi))}{d\mathbf{z}} d\xi\right] (\mathbf{Z}_i - \mathbf{Z}_{i-1}). \end{aligned} \tag{3.3.8}$$

From (3.3.7) and (3.3.8), we have

$$\mathbf{f}(\mathbf{Q}_i) - \mathbf{f}(\mathbf{Q}_{i-1}) = \hat{\mathbf{C}}_{i-1/2}(\mathbf{Z}_i - \mathbf{Z}_{i-1}),$$

$$\mathbf{Q}_i - \mathbf{Q}_{i-1} = \hat{\mathbf{B}}_{i-1/2}(\mathbf{Z}_i - \mathbf{Z}_{i-1}),$$

where $\hat{\mathbf{C}}_{i-1/2}$ and $\hat{\mathbf{B}}_{i-1/2}$ defines the integrals in (3.3.7) and (3.3.8) respectively. From these we can obtain the desired relation (3.3.5) by using

$$\hat{\mathbf{A}}_{i-1/2} = \hat{\mathbf{C}}_{i-1/2}\hat{\mathbf{B}}^{-1}_{i-1/2}. \tag{3.3.9}$$

3.3.2. Roe Solver for the Saint-Venant Equations. For the Saint-Vanant equations we have

$$\mathbf{q} = \begin{bmatrix} h \\ hu \end{bmatrix} = \begin{bmatrix} q^1 \\ q^2 \end{bmatrix}, \qquad \mathbf{f}(\mathbf{q}) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix} = \begin{bmatrix} q^2 \\ (q^2)^2/q^1 + \frac{1}{2}g(q^1)^2 \end{bmatrix},$$

and

$$\mathbf{f}'(\mathbf{q}) = \begin{bmatrix} 0 & 1 \\ -(q^2/q^1)^2 + gq^1 & 2q^2/q^1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -u^2 + gh & 2u \end{bmatrix}.$$

As a parameter vector we choose

$$\mathbf{z} = h^{-1/2}\mathbf{q}, \qquad \text{so that} \qquad \begin{bmatrix} z^1 \\ z^2 \end{bmatrix} = \begin{bmatrix} \sqrt{h} \\ \sqrt{h}u \end{bmatrix}. \tag{3.3.10}$$

We find that

$$\mathbf{q}(\mathbf{z}) = \begin{bmatrix} (z^1)^2 \\ z^1 z^2 \end{bmatrix} \qquad \Rightarrow \qquad \frac{\partial \mathbf{q}}{\partial \mathbf{z}} = \begin{bmatrix} 2z^1 & 0 \\ z^2 & z^1 \end{bmatrix} \tag{3.3.11}$$

and

$$\mathbf{f}(\mathbf{z}) = \begin{bmatrix} z^1 z^2 \\ (z^2)^2 + 1/2g(z^1)^4 \end{bmatrix} \qquad \Rightarrow \qquad \frac{\partial \mathbf{f}}{\partial \mathbf{z}} = \begin{bmatrix} z^2 & z^1 \\ 2g(z^1)^3 & 2z^2 \end{bmatrix}. \tag{3.3.12}$$

We now set

$$z^p = Z^p_{i-1} + (Z^p_i - Z^p_{i-1})\xi \quad \text{for} \quad p = 1, 2$$

and integrate each element of these matrices from $\xi = 0$ to $\xi = 1$. All elements are linear in ξ except the $(2,1)^{th}$ element of $\frac{\partial \mathbf{f}}{\partial \mathbf{z}}$, which is cubic. Integrating the linear terms $z^p(\xi)$ yields

$$\int_0^1 z^p(\xi) d\xi = \frac{1}{2}(Z^p_{i-1} + Z^p_i) \equiv \bar{Z}^p,$$

simply the average between the endpoints. For the cubic term we obtain

$$\begin{aligned}\int_0^1 (z^1(\xi))^3 d\xi &= \frac{1}{4}\left(\frac{(Z_i^1)^4 - (Z_{i-1}^1)^4}{Z_i^1 - Z_{i-1}^1}\right) \\ &= \frac{1}{2}(Z_{i-1}^1 + Z_i^1)\cdot\frac{1}{2}((Z_{i-1}^1)^2 + (Z_i^1)^2) \\ &= \bar{Z}^1[\frac{1}{2}(h_{i-1} + h_i)] \\ &= \bar{Z}^1\bar{h},\end{aligned}$$

where

$$\bar{h} = \frac{1}{2}(h_{i-1} + h_i). \tag{3.3.13}$$

Hence we obtain

$$\hat{\mathbf{B}}_{i-1/2} = \begin{bmatrix} 2\bar{Z}^1 & 0 \\ \bar{Z}^2 & \bar{Z}^1 \end{bmatrix}, \hat{\mathbf{C}}_{i-1/2} = \begin{bmatrix} \bar{Z}^2 & \bar{Z}^1 \\ 2g\bar{Z}^1\bar{h} & 2\bar{Z}^2 \end{bmatrix}$$

and so

$$\hat{\mathbf{A}}_{i-1/2} = \hat{\mathbf{C}}_{i-1/2}\hat{\mathbf{B}}_{i-1/2} = \begin{bmatrix} 0 & 1 \\ \left(\bar{Z}^2/\bar{Z}^1\right)^2 + g\bar{h} & 2\bar{Z}^2/\bar{Z}^1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\hat{u}^2 + g\bar{h} & 2\bar{u} \end{bmatrix}. \tag{3.3.14}$$

Here $\bar{h}$ is the arithmetic average of h_{i-1} and h_i given in (3.3.13), but $\hat{u}$ is a different sort of average of the velocities known as the *Roe average*:

$$\hat{u} = \frac{\bar{Z}^2}{\bar{Z}^1} = \frac{\sqrt{h_{i-1}}u_{i-1} + \sqrt{h_i}u_i}{\sqrt{h_{i-1}} + \sqrt{h_i}} \tag{3.3.15}$$

Note that the matrix $\hat{\mathbf{A}}_{i-1/2}$ in (3.3.14) is the Jacobian matrix $\mathbf{f}'(\hat{\mathbf{q}})$ evaluated at the special state $\hat{\mathbf{q}} = (\bar{h}, \bar{h}\hat{u})^T$. In particular, if $\mathbf{Q}_{i-1} = \mathbf{Q}_i$ then $\hat{\mathbf{A}}_{i-1/2}$ reduces to

$\mathbf{f}'(\mathbf{Q}_i)$. The eigenvalues and eigenvectors of $\hat{\mathbf{A}}_{i-1/2}$ are known from section (2.4):

$$\hat{\lambda}^1 = \hat{u} - \hat{c}, \quad \hat{\lambda}^2 = \hat{u} + \hat{c}, \tag{3.3.16}$$

and

$$\hat{\mathbf{r}}^1 = \begin{bmatrix} 1 \\ \hat{u} - \hat{c} \end{bmatrix}, \quad \hat{\mathbf{r}}^1 = \begin{bmatrix} 1 \\ \hat{u} + \hat{c} \end{bmatrix}, \tag{3.3.17}$$

where $\hat{c} = \sqrt{g\bar{h}}$. To use the approximate Riemann solver (section 2.6) we decompose $\mathbf{Q}_i - \mathbf{Q}_{i-1}$ as in (3.3.4),

$$\mathbf{Q}_i - \mathbf{Q}_{i-1} = \alpha^1_{i-1/2}\hat{\mathbf{r}}^1 + \alpha^2_{i-1/2}\hat{\mathbf{r}}^2 \tag{3.3.18}$$

$$\equiv \mathcal{W}^1_{i-1/2} + \mathcal{W}^2_{i-1/2}. \tag{3.3.19}$$

The coefficients $\alpha^p_{i-1/2}$ are computed by solving this linear system, which can be done explicitly by inverting the matrix $\hat{\mathbf{R}}$ of the eigenvectors to obtain

$$\hat{\mathbf{L}} = \hat{\mathbf{R}}^{-1} = \frac{1}{2\hat{c}} \begin{bmatrix} \hat{u} + \hat{c} & -1 \\ -(\hat{u} - \hat{c}) & 1 \end{bmatrix}. \tag{3.3.20}$$

Multiplying this by the vector $\delta \equiv [\,\delta^1, \delta^2\,]^T \equiv \mathbf{Q}_i - \mathbf{Q}_{i-1}$ gives the vector of α-coefficients, and hence

$$\alpha^1_{i-1/2} = \frac{(\hat{u} + \hat{c})\delta^1 - \delta^2}{2\hat{c}}, \tag{3.3.21}$$

$$\alpha^2_{i-1/2} = -\frac{(\hat{u} - \hat{c})\delta^1 + \delta^2}{2\hat{c}}. \tag{3.3.22}$$

The fluctuations (3.2.8) and (3.2.9) are then used in Godunov's method, with the speeds s given by the eigenvalues λ of (3.3.16)

3.4. High-Resolution Methods

In section (3.2) we developed the basic ideas of Godunov's method which is only first-order accurate and possesses some numerical diffusion.

In this section we shall see how this method can be greatly improved by introducing correction terms into (3.2.4)

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n - \frac{\Delta t}{\Delta x}(\mathcal{A}^+\Delta\mathbf{Q}_{i-1/2} + \mathcal{A}^-\Delta\mathbf{Q}_{i+1/2}) - \frac{\Delta t}{\Delta x}(\tilde{\mathbf{F}}_{i+1/2} - \tilde{\mathbf{F}}_{i-1/2}). \qquad (3.4.1)$$

The fluxes $\tilde{\mathbf{F}}_{i-1/2}$ are based on the waves resulting from the Riemann solution, which have already been computed in the process of determining the fluctuations $\mathcal{A}^{\pm}\Delta\mathbf{Q}_{i-1/2}$. The basic form of these correction terms is motivated by the *Lax-Wendroff method*, a standard second-order accurate method described in the next subsection. The addition of crucial *limiters* [3] leads to great improvement, as discussed later in this section.

3.4.1. The Lax-Wendroff Method. The Lax-Wendroff method for the linear system $\mathbf{q}_t + \mathbf{A}\mathbf{q}_x = 0$ is based on the Taylor series expansion

$$\mathbf{q}(x, t_{n+1}) = \mathbf{q}(x, t_n) + \Delta t\mathbf{q}_t(x, t_n) + \frac{1}{2}(\Delta t)^2\mathbf{q}_{tt}(x, t_n) + ... \qquad (3.4.2)$$

From differentiating $\mathbf{q}_t = -\mathbf{A}\mathbf{q}_x$, we have

$$\mathbf{q}_{tt} = -\mathbf{A}\mathbf{q}_{xt} = \mathbf{A}^2\mathbf{q}_{xx},$$

where we have used $\mathbf{q}_{xt} = (-\mathbf{A}\mathbf{q}_x)_x$. From (3.4.2), it follows that

$$\mathbf{q}(x, t_{n+1}) = \mathbf{q}(x, t_n) - \Delta t\mathbf{A}\mathbf{q}_x(x, t_n) + \frac{1}{2}(\Delta t)^2\mathbf{A}^2\mathbf{q}_{xx}(x, t_n) + ... \qquad (3.4.3)$$

Using central finite difference approximations, we obtain the *Lax-Wendroft method*

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n - \frac{\Delta t}{2\Delta x}\mathbf{A}(\mathbf{Q}_{i+1}^n - \mathbf{Q}_{i-1}^n) + 1/2\left(\frac{\Delta t}{\Delta x}\right)^2 \mathbf{A}^2(\mathbf{Q}_{i-1}^n - 2\mathbf{Q}_i^n + \mathbf{Q}_{i+1}^n). \quad (3.4.4)$$

This is a second-order accurate method. The derivation of this method is based on a finite difference interpretation, with $\mathbf{Q}_i^n$ approximating the pointwise value $\mathbf{q}(x_i, t_n)$. However, we can reinterpret (3.4.4) using a finite volume method as

$$\mathbf{Q}_i^{n+1} = \mathbf{Q}_i^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i+1/2}^n - \mathbf{F}_{i-1/2}^n\right) \quad (3.4.5)$$

with the flux function

$$\mathbf{F}_{i-1/2}^n = \frac{1}{2}\mathbf{A}(\mathbf{Q}_{i-1}^n + \mathbf{Q}_i^n) - \frac{1}{2}\frac{\Delta t}{\Delta x}\mathbf{A}^2(\mathbf{Q}_i^n - \mathbf{Q}_{i-1}^n).$$

Since $\mathbf{A}^+ - \frac{1}{2}|A| = \frac{1}{2}\mathbf{A} = \mathbf{A}^- + \frac{1}{2}|\mathbf{A}|$, where $\mathbf{A}^+ = \mathbf{R}\mathbf{\Lambda}^+\mathbf{R}^{-1}$, $\mathbf{A}^- = \mathbf{R}\mathbf{\Lambda}^-\mathbf{R}^{-1}$ and

$$\mathbf{\Lambda}^\pm = \begin{pmatrix} (\lambda^1)^\pm & & \\ & \ddots & \\ & & (\lambda^m)^\pm \end{pmatrix},$$

$$\mathbf{F}_{i-1/2}^n = \mathbf{A}^-\mathbf{Q}_i^n + \mathbf{A}^+\mathbf{Q}_{i-1}^n) + \frac{1}{2}|\mathbf{A}|\left(I - \frac{\Delta t}{\Delta x}|\mathbf{A}|\right)(\mathbf{Q}_i^n - \mathbf{Q}_{i-1}^n).$$

We can rewrite the correction term as

$$\frac{1}{2}|\mathbf{A}|\left(\mathbf{I} - \frac{\Delta t}{\Delta x}|\mathbf{A}|\right)(\mathbf{Q}_i - \mathbf{Q}_{i-1}) = \frac{1}{2}|\mathbf{A}|\left(\mathbf{I} - \frac{\Delta t}{\Delta x}|\mathbf{A}|\right)\sum_{p=1}^{m}\alpha_{i-1/2}^p\mathbf{r}^p,$$

where $\mathbf{r}^p$ denotes the eigenvectors of $\mathbf{A}$ and the coefficients $\alpha^p_{i-1/2}$ are given by

$$\mathbf{Q}_i - \mathbf{Q}_{i-1} = \sum_{p=1}^{m} \alpha^p_{i-1/2}\mathbf{r}^p = \sum_{p=1}^{m} \mathcal{W}^p_{i-1/2}.$$

The flux-limiter method is defined by replacing the scalar coefficient $\alpha^p_{i-1/2}$ by a limited version. We set

$$\tilde{\alpha}^p_{i-1/2} = \alpha^p_{i-1/2}\phi(\theta^p_{i-1/2}), \tag{3.4.6}$$

where

$$\theta^p_{i-1/2} = \frac{\alpha^p_{I-1/2}}{\alpha^p_{i-1/2}}$$

with

$$I = \begin{cases} i-1, & \text{if } \lambda^p > 0\ , \\ i+1, & \text{if } \lambda^p < 0\ , \end{cases}$$

and

$$\text{minmod} : \phi(\theta) = \text{minmod}(1, \theta),$$

$$\text{superbee} : \phi(\theta) = \max(0, \min(1, 2\theta), \min(2, \theta)),$$

$$\text{MC} : \phi(\theta) = \max(0, \min((1+\theta)/2, 2, 2\theta)),$$

$$\text{van Leer} : \phi(\theta) = \frac{\theta + |\theta|}{1 + |\theta|},$$

where

$$\text{minmod}(a,b) = \begin{cases} a & \text{if } |a| < |b| \text{ and } ab > 0, \\ b & \text{if } |b| < |a| \text{ and } ab > 0, \\ 0 & \text{if } ab \leq 0\ . \end{cases}$$

The flux function for the flux-limiter method is then

$$\mathbf{F}_{i-1/2} = \mathbf{A}^{+}\mathbf{Q}_{i-1} + \mathbf{A}^{-}\mathbf{Q}_{i} + \tilde{\mathbf{F}}_{i-1/2}, \tag{3.4.7}$$

and the correction flux $\tilde{\mathbf{F}}_{i-1/2}$ is defined by

$$\tilde{\mathbf{F}}_{i-1/2} = \frac{1}{2}\left|\mathbf{A}\right|\left(I - \frac{\Delta t}{\Delta x}\left|\mathbf{A}\right|\right)\sum_{p=1}^{m}\tilde{\alpha}_{i-1/2}^{p}\mathbf{r}^{p}. \tag{3.4.8}$$

We can rewrite (3.4.8) as

$$\tilde{\mathbf{F}}_{i-1/2} = \frac{1}{2}\sum_{p=1}^{m}\left|\lambda^{p}\right|\left(1 - \frac{\Delta t}{\Delta x}\left|\lambda^{p}\right|\right)\tilde{\mathcal{W}}_{i-1/2}^{p}, \tag{3.4.9}$$

where

$$\tilde{\mathcal{W}}_{i-1/2}^{p} = \tilde{\alpha}_{i-1/2}^{p}\mathbf{r}^{p}.$$

To accomplish this most easily, note that if we use the flux (3.4.7) in the flux-differencing formula (3.4.5) and then rearrange some terms, we can write the formula for $\mathbf{Q}_{i}^{n+1}$ as

$$\mathbf{Q}_{i}^{n+1} = \mathbf{Q}_{i}^{n} - \frac{\Delta t}{\Delta x}\left(\mathbf{A}^{+}\Delta\mathbf{Q}_{i-1/2} + \mathbf{A}^{-}\Delta\mathbf{Q}_{i+1/2}\right) - \frac{\Delta t}{\Delta x}\left(\tilde{\mathbf{F}}_{i+1/2} - \tilde{\mathbf{F}}_{i-1/2}\right),$$

where we drop the superscript n from the current time step because we will need to use superscript p below to denote the wave family. Each term in this expression can

be written as the waves $\mathcal{W}^p_{i-1/2} = \alpha^p_{i-1/2}\mathbf{r}^p$ and wave speeds α^p:

$$\mathbf{A}^+\Delta\mathbf{Q}_{i-1/2} = \sum_{p=1}^{m}(\lambda^p)^+\mathcal{W}^p_{i-1/2},$$

$$\mathbf{A}^-\Delta\mathbf{Q}_{i-1/2} = \sum_{p=1}^{m}(\lambda^p)^-\mathcal{W}^p_{i-1/2},$$

$$\tilde{\mathbf{F}}_{i-1/2} = \frac{1}{2}\sum_{p=1}^{m}|\lambda^p|\left(1-\frac{\Delta t}{\Delta x}|\lambda^p|\right)\tilde{\mathcal{W}}^p_{i-1/2}. \tag{3.4.10}$$

3.4.2. Extension to Nonlinear Systems. The quantities $\mathbf{A}^+\Delta\mathbf{Q}_{i-1/2}$ and $\mathbf{A}^-\Delta\mathbf{Q}_{i-1/2}$ can be generalized to fluctuations in nonlinear systems and will be denoted by $\mathcal{A}^+\Delta\mathbf{Q}_{i-1/2}$ and $\mathcal{A}^-\Delta\mathbf{Q}_{i-1/2}$. In general we can think of setting

$$\mathcal{A}^+\Delta\mathbf{Q}_{i-1/2} = \sum_{p=1}^{m}(s^p_{i-1/2})^+\mathcal{W}^p_{i-1/2}, \tag{3.4.11}$$

$$\mathcal{A}^-\Delta\mathbf{Q}_{i-1/2} = \sum_{p=1}^{m}(s^p_{i-1/2})^-\mathcal{W}^p_{i-1/2}, \tag{3.4.12}$$

as a direct extension of (3.4.10). Here, $s^p_{i-1/2}$ denotes the characteristic speed of the nonlinear solution. Moreover, we set

$$\mathbf{Q}^{n+1}_i = \mathbf{Q}^n_i - \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+\Delta\mathbf{Q}_{i-1/2} + \mathcal{A}^-\Delta\mathbf{Q}_{i+1/2}\right) - \frac{\Delta t}{\Delta x}\left(\tilde{\mathbf{F}}_{i+1/2} - \tilde{\mathbf{F}}_{i-1/2}\right), \tag{3.4.13}$$

where

$$\tilde{\mathbf{F}}_{i-1/2} = \frac{1}{2}\sum_{p=1}^{m}\left|s^p_{i-1/2}\right|\left(1-\frac{\Delta t}{\Delta x}\left|s^p_{i-1/2}\right|\right)\tilde{\mathcal{W}}^p_{i-1/2},$$

where

$$\tilde{\mathcal{W}}^p_{i-1/2} = \phi(\theta^p_{i-1/2})\mathcal{W}^p_{i-1/2},$$

where

$$\theta^p_{i-1/2} = \frac{\alpha^p_{I-1/2}}{\alpha^p_{i-1/2}} = \frac{\mathcal{W}^p_{I-1/2} \cdot \mathcal{W}^p_{i-1/2}}{\mathcal{W}^p_{i-1/2} \cdot \mathcal{W}^p_{i-1/2}}$$

(see [1],chapter 9).

3.5. The HLL and HLLE Solvers

For dry bed case, the HLL and HLLE solvers [3] give more accurate numerical solutions than the Godunov's and High-resolution methods. A simple approximate Riemann solver can be based on estimating the smallest and largest wave speeds arising in the Riemann solution and on defining $\hat{\mathbf{Q}}(x/t)$ by using the waves propagation speeds $s^1_{i-1/2}$ and $s^2_{i-1/2}$. There will then be a single new state $\hat{\mathbf{Q}}_{i-1/2}$ in between, and as waves we use

$$\mathcal{W}^1_{i-1/2} = \hat{\mathbf{Q}}_{i-1/2} - \mathbf{Q}_{i-1} \text{ and } \mathcal{W}^2_{i-1/2} = \mathbf{Q}_i - \hat{\mathbf{Q}}_{i-1/2}.$$

We can determine the state $\hat{\mathbf{Q}}_{i-1/2}$ by requiring that the approximate solution be conservative, which requires

$$s^1_{i-1/2}(\hat{\mathbf{Q}}_{i-1/2} - \mathbf{Q}_{i-1}) + s^2_{i-1/2}(\mathbf{Q}_i - \hat{\mathbf{Q}}_{i-1/2}) = \mathbf{f}(\mathbf{Q}_i) - \mathbf{f}(\mathbf{Q}_{i-1}) \tag{3.5.1}$$

and so

$$\hat{\mathbf{Q}}_{i-1/2} = \frac{\mathbf{f}(\mathbf{Q}_i) - \mathbf{f}(\mathbf{Q}_{i-1}) - s^2_{i-1/2}\mathbf{Q}_i + s^1_{i-1/2}\mathbf{Q}_{i-1}}{s^1_{i-1/2} - s^2_{i-1/2}}. \tag{3.5.2}$$

Approximate Riemann solvers of this type were studied by Harten, Lax, and van Lear and further developed by Einfeldt, who suggested a choice of s^1 and s^2 in the context of gas dynamics that can be generalized to

$$s^1_{i-1/2} = \min{}_p(\min(\lambda^p_i, \hat{\lambda}^p_{i-1/2})), \tag{3.5.3}$$

$$s^2_{i-1/2} = \max{}_p(\max(\lambda^p_{i+1}, \hat{\lambda}^p_{i-1/2})). \tag{3.5.4}$$

Here λ^p_j is the pth eigenvalue of the Jacobian $\mathbf{f}'(\mathbf{Q}_j)$, and $\hat{\lambda}^p_{i-1/2}$ is the pth eigenvalue of the Roe average.

3.6. Boundary Conditions

Suppose the problem is on the physical domain $[a, b]$, which is subdivided into cells $C_1, C_2, ..., C_N$ with $x_1 = a$ and $x_{N+1} = b$, so that $\Delta x = (b - a)/N$. If we use a method for which $\mathbf{F}^n_{i-1/2}$ depends only on $\mathbf{Q}^n_{i-1}$ and $\mathbf{Q}^n_i$, then we need only one ghost cell on either end. The ghost cell $C_0 = (a - \Delta x, a)$ allows us to calculate the flux $\mathbf{F}_{1/2}$ at the left boundary while the ghost cell $C_{N+1} = (b, b + \Delta x)$ is used to calculate $\mathbf{F}^n_{N+1/2}$ at $x = b$. With a flux-limiter method of the type developed in section 3.4, we will generally need two ghost cells at each boundary, since, for example, the jump $\mathbf{Q}_0 - \mathbf{Q}_{-1}$ will be needed in limiting the flux correction in $\mathbf{F}_{1/2}$.

We can obtain the ghost cell by extrapolation from the interior solution. If the ghost-cell value $\mathbf{Q}^n_{N+1}$ is determined from $\mathbf{Q}^n_N, \mathbf{Q}^n_{N-1}, ...$, then the new value $\mathbf{Q}^{n+1}_N$ will effectively be computed on the basis of values to the left alone, even if the formula depends on $\mathbf{Q}^n_{N+1}$. The simplest approach is to use a *zero-order* extrapolation [3], meaning the extrapolation by a constant function. We simply set

$$\mathbf{Q}^n_{N+1} = \mathbf{Q}^n_N, \quad \mathbf{Q}^n_{N+2} = \mathbf{Q}^n_N \tag{3.6.1}$$

at the start of each time step. In the same way, we set

$$\mathbf{Q}_0^n = \mathbf{Q}_1^n, \quad \mathbf{Q}_{-1}^n = \mathbf{Q}_1^n \tag{3.6.2}$$

in each time step.

Note that by setting $\mathbf{Q}_0 = \mathbf{Q}_1$ we insure that the solution to the Riemann problem at the interface $x_{1/2}$ consists of no waves, or more properly that the wave strengths $\alpha_{1/2}^p$ are all zero. So in particular there are no waves generated at the boundary regardless of what is happening in the interior.

3.7. Sonic Entropy Fixes

For the Saint-Venant equations, there is a single intermediate state $\hat{\mathbf{Q}}_m$ in the approximate Riemann solution between $\mathbf{Q}_{i-1}$ and $\mathbf{Q}_i$. We can compute the characteristic speeds in each state as

$$\lambda_{i-1}^1 = u_{i-1} - \sqrt{gh_{i-1}}, \ \lambda_m^1 = \hat{u}_m - \sqrt{g\hat{h}_m},$$
$$\lambda_m^2 = \hat{u}_m + \sqrt{g\hat{h}_m}, \qquad \lambda_i^2 = u_i + \sqrt{gh_i}.$$

If $\lambda_{i-1}^1 < 0 < \lambda_m^1$, then we should suspect that the 1-wave is actually a transonic rarefaction and make some adjustment to the flux, i.e., to $\mathcal{A}^-\Delta\mathbf{Q}_{i-1/2}$ and $\mathcal{A}^+\Delta\mathbf{Q}_{i-1/2}$, in this case. Similarly, if $\lambda_m^2 < 0 < \lambda_i^2$, then we should fix the flux to incorporate a 2-rarefaction. Note that at most one of these situations can hold, since $\lambda_m^1 < \lambda_m^2$.

The Harten-Hyman Entropy Fix

A more general procedure was taken by Harten and Hyman [3]. Suppose there appears to be a transonic rarefaction in the k-wave, i.e., $\lambda_l^k < 0 < \lambda_r^k$, where $\lambda_{l,r}^k$ represents the kth eigenvalue of the matrix $\mathbf{f}'(\mathbf{q})$ computed in the states $\mathbf{q}_{l,r}^k$ just to

the left and right of the k-wave in the approximate Riemann solution, i.e.,

$$\mathbf{q}_l^k = \mathbf{Q}_{i-1} + \sum_{p=1}^{k-1} \mathcal{W}^p, \quad \mathbf{q}_r^k = \mathbf{q}_l^k + \mathcal{W}^k.$$

(We suppress the subscripts $i - 1/2$ here and below for clarity, since we need to add subscripts l and r.) Then we replace the single wave $\mathcal{W}^k$ propagating at speed $\hat{\lambda}^k$ by a pair of waves $\mathcal{W}_l^k = \beta\mathcal{W}^k$ and $\mathcal{W}_r^k = (1-\beta)\mathcal{W}^k$ propagating at speeds λ_l^k and λ_r^k. To maintain conservation we require that

$$\lambda_l^k \mathcal{W}_l^k + \lambda_r^k \mathcal{W}_r^k = \hat{\lambda}^k \mathcal{W}^k.$$

Hence

$$\lambda_l^k (\beta\mathcal{W}^k) + \lambda_r^k (1-\beta)\mathcal{W}^k = \hat{\lambda}^k \mathcal{W}^k.$$

Thus

$$\beta = \frac{\lambda_r^k - \hat{\lambda}^k}{\lambda_r^k - \lambda_l^k}.$$

In practice it is simpler to leave the wave $\mathcal{W}^k$ alone (and continue to use this single wave in the high-resolution correction terms; see Section 3.4) and instead modify the values $(\hat{\lambda}^k)^{\pm}$ used in defining $\mathcal{A}^{\pm}\Delta\mathbf{Q}_{i-1/2}$ via (3.4.11) and (3.4.12). The formula (3.4.11) and (3.4.12) can still be used (with $\hat{s}^k = \hat{\lambda}^k$) if, instead of the position and negative parts of $\hat{\lambda}^k$, we use the values

$$(\hat{\lambda}^k)^- \equiv \beta\lambda_l^k,$$

$$(\hat{\lambda}^k)^+ \equiv (1-\beta)\lambda_r^k$$

in the kth field.

3.8. Fractional Step Method

So far, we have only considered homogeneous conservation laws of the form $\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0$. There are many situations in which *source terms* also appear in the equations, so that we wish to solve the system

$$\mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = \psi(\mathbf{q}). \tag{3.8.1}$$

A fractional-step method for (3.8.1) is applied by first splitting the equation into two *subproblems* that we would take these to be:

$$\text{Problem A: } \mathbf{q}_t + \mathbf{f}(\mathbf{q})_x = 0, \tag{3.8.2}$$

$$\text{Problem B: } \mathbf{q}_t = \psi(\mathbf{q}). \tag{3.8.3}$$

For the advection-reaction problem

$$q_t + \bar{u}q_x = -\beta q$$

we would take these to be

$$\text{Problem A: } q_t + \bar{u}q_x = 0, \tag{3.8.4}$$

$$\text{Problem B: } q_t = -\beta q. \tag{3.8.5}$$

As a simple example of the fractional-step procedure, suppose we use the upwind method for the A-step and the forward Euler for the ODE in the B-step for the advection-reaction problem. Then the simplest fractional-step method over one time

step would consist of the following two stages:

$$\text{A-step:}\quad Q_i^* = Q_i^n - \bar{u}\frac{\Delta t}{\Delta x}(Q_i^n - Q_{i-1}^n) \tag{3.8.6}$$

$$\text{B-step:}\quad Q_i^{n+1} = Q_i^* - \beta \Delta t Q_i^*. \tag{3.8.7}$$

CHAPTER IV

Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations

In many time dependent simulations, the solution on most part of the domain will be fairly smooth, with discontinuities or highly oscillatory phenomena occurring over only a small part of the domain. For such problems, a mesh refinement approach can be the most efficient. Refined grids with smaller and smaller mesh spacing are placed only where they are needed. Since we are solving a time dependent problem, the regions needing refinement will change, and therefore our grids must adapt with time as well.

This thesis presents a method based on the idea of multiple, component grids for the solution of hyperbolic partial differential equations using explicit finite volume methods. Based upon Richardson type estimates of the local truncation error, proposed by Berger [4], refined grids are created or existing ones removed to attain a given accuracy for a minimum amount of work. In addition, this approach is recursive in that fine grids can themselves contain even finer subgrids. Those grids with finer mesh width in space can also have a smaller mesh width in time. This constitutes a mesh refinement algorithm in time and space.

4.1. Grid Description

At the start of a computation, the coarsest or base grid is specified by the user. This base grid denoted by G_0 will remain fixed for the duration of the computation.

In the computation, refined subgrids will be created adaptively in response to some feature in the solution or the appearence of shock fronts.

Our goal is to generate the subgrids to best fit the area of the domain where they are needed.

We define the *level* of a grid to be the depth of this nesting, that is, the number of coarser grids the fine grid is contained in. We say that the coarse grid G_0 is at *level* 0 in the grid hierarchy. The subgrids of G_0 are part of G_1 and they are said to be level 1 refinement. Refined grids within the G_1 grids are at level 2, denoted by G_2, and so on. In this way, a nested sequence of grids with finer and finer discretizations may be created over some portion of the spatial domain. Each such grid is one grid component, denoted by $G_{l,j}$, of G_l, where G_l consists of those grids at level l in the hierarchy having mesh width h_l.

In practice, we assume that a set of possible mesh discretizations $h_0, h_1, h_2, ..., h_{max}$ has been specified in advance. Each h_l is an integral multiple of h_{l+1}. In this thesis, we presents a calculation using $h_{l+1} = h_l/\gamma$ with a refinement ratio $\gamma = 2$.

4.2. Integration Algorithm

In this section we describe the integration algorithm that we use to solve a hyperbolic pde using mesh refinement. There are three main components in this algorithm. They are

(i) the actual time integration using finite volume that is done on each grid,

(ii) the error estimation and subsequent grid generation, and

(iii) the special grid-to-grid operations that must be done every time step during the integration that arise because of mesh refinement itself.

Recall from section (4.1) that in our grid formulation, the mesh widths h_l of grids at level l are an integral factor γ of the mesh width h_{l-1} of the next coarser level. We use the same factor to set the time step on the level l grids, $k_l = k_{l-1}/\gamma$, This is an appropriate time step for hyperbolic equations. In this way we keep the mesh ratio λ of time step to space step constant on all grids. One of the main reasons our

method is efficient because the overly restrictive small time step of the finest grid is not applied over the entire domain.

This constant mesh ratio λ makes it easy to detemine the order of grid integration. If γ is the refinement ratio, for every time step taken on level 0 grids, take γ time steps on level 1 grids, γ^2 on level 2 grids, etc. These steps are interleaved so that before advancing a grid, all its subgrids are integrated to the same time. At every coarse grid step, all grids should be at the same time. One coarse grid cycle is then the basic unit of the algorithm.

Figure 4.1 illustrates this in one space dimension and time. We depict one coarse grid, $G_{0,1}$, one fine grid $G_{1,1}$ at level 1, and a finer grid at level 2, $G_{2,1}$, all with the refinement ratio $\gamma = 2$. The order of their integration, from coarsest to first, for one coarse grid cycle is

$$t = 0 \xrightarrow{(1)} t = \Delta t$$

$$t = 0 \xrightarrow{(2)} t = \Delta t/2 \xrightarrow{(5)} t = \Delta t$$

$$t = 0 \xrightarrow{(3)} t = \Delta t/4 \xrightarrow{(4)} t = \Delta t/2 \xrightarrow{(6)} t = 3\Delta t/4 \xrightarrow{(7)} t = \Delta t$$

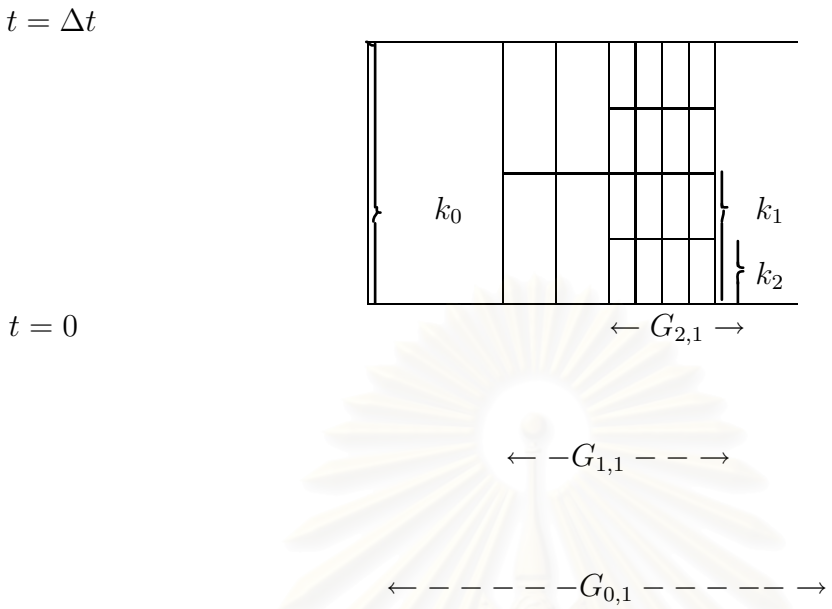

Figure 4.1 Time integration on three-level grid structure.

Error estimation and the subsequent regridding operation is the second major task of the mesh refinement algorithm. In the next section the algorithm used to estimate the error will be presented. In section (4.4) we will discuss the method of grid generation, which uses the results of the error estimation.

The last major component of the mesh refinement algorithm concerns the grid communication routines. These are two tasks which fit under this heading. The first of these deals with boundaries. The boundary values are calculated using values from coarse grids. We use interpolation to get the boundary values. The second item of inter-grid communication is updating. If a fine grid is nested in a coarse grid, then when they are integrated to the same point in time the coarse grid values are updated by injecting the fine grid solution values onto the coarse grid points.

The overall mesh refinement algorithm is presented in figure (4.2) in outline form. It can be written quite simply as a recursive procedure. Of course, in writing the mesh refinement program in Fortran, we convert it to an iterative procedure.

Recursive Procedure INTEGRATE(l)

If $l = 0$ **Then** $nbstep = 1$

Else $nbstep = \gamma$

Endif

Repeat $nbstep$ times

Step on all grids at level l

If level $l + 1$ exists **Then**

Compute boundary conditions at level $l + 1$

INTEGRATE($l + 1$)

update level l

Endif

End Repeat

End Procedure INTEGRATE

Figure 4.2 Coarse Grid Integration Cycle

4.3. Error Estimation

Subgrids are placed over regions that need refinement. As started earlier, a grid is refined where the truncation error is large. A variation of Richardson extrapolation [4] is used to estimate the truncation error. In this section we show how the truncation error estimate is calculated and point out the advantages of this technique.

We begin the discussion by first introducing some notation. Consider a hyperbolic differential equation,

$$u_t = L[u] \tag{4.3.1}$$

where L is the spatial differential operator. A simple explicit finite difference method for this equation is:

$$\frac{u(x,t+k)-u(x,t)}{k} = L_h[u(x,t)] + O(k) \tag{4.3.2}$$

Here, L_h is the spatial finite difference operator for a grid with mesh size h. This can be rewritten in a compact form as:

$$u(x,t+k) = Q_h[u(x,t)]. \tag{4.3.3}$$

The truncation error is obtained by substituting the exact solution into (4.3.3). If the exact solution is smooth in space and time, the truncation error is:

$$\begin{aligned} u(x,t+k) - Q_h[u(x,t)] &= [u(x,t) + u_t(x,t)k + O(k^2)] - [kL_h(u(x,t)) + u(x,t) + O(k^2)] \\ &= k[u_t(x,t) - L_h(u(x,t))] \end{aligned}$$

If the truncation error of $u_t(x,t) - L_h(u(x,t))$ is $O(h^q, k^q)$, then

$$\begin{aligned} u(x,t+k) - Q_h[u(x,t)] &= k[k^q a(x,t) + h^q b(x,t)] + kO(k^{q+1}, h^{q+1}) \\ &= \tau + kO(k^{q+1}, h^{q+1}), \end{aligned}$$

where τ is the leading order term. Taking two consecutive time steps with the method gives

$$u(x,t+2k) - Q_h^2[u(x,t)] = 2\tau + kO(k^{q+1}, h^{q+1}). \tag{4.3.4}$$

Let Q_{2h} represent the same difference method as Q_h except with mesh size $2h$ and time step $2k$. The truncation error for the $2h - 2k$ method is:

$$u(x, t + 2k) - Q_{2h}[u(x, t)] = 2k[(2k)^q a(x, t) + (2h)^q b(x, t)] + kO(k^{q+1}, h^{q+1})$$

$$= 2^q(2\tau) + kO(k^{q+1}, h^{q+1}). \tag{4.3.5}$$

Neglecting the higher-order terms in (4.3.4), subtracting (4.3.5) and dividing by $2(2^q - 1)$ gives:

$$\frac{Q_h^2[u(x, t)] - Q_{2h}[u(x, t)]}{2(2^q - 1)} = \tau + kO(k^{q+1}, h^{q+1}) \tag{4.3.6}$$

(4.3.6) provides an estimate of the leading term in the truncation error.

This is equivalent to advancing the solution two steps from time t with the standard method and comparing it with the solution obtained by taking one double-step on a $2h$ mesh.

A major advantage of this technique is that the exact form of the truncation error does not need to be known. When it is time to estimate the error, (4.3.6) is evaluated at every point in the grid. If the pointwise truncation error estimate is greater than a prescribed value, the point is "flagged" to denote that refinement is needed in its vicinity. Once all the local error estimates have been calculated and checked, the collection of flagged points is then processed to generate the next level of refined subgrids.

4.4. Clustering and Grid Generation

Much of the success of this adaptive mesh refinement algorithm lies in the generation of efficient subgrids. The procedure is to estimate the error at all grid points in

level l grids, and flag those points where the error (or some other measure for determining the need for refinement) exceeds a tolerance ϵ. The grid generation procedure creates a new level of grids with mesh width h_{l+1} so that every flagged point is in the interior of a fine grid.

Thus, for each existing level of grids, we apply the same procedure to generate the next fine level. This regridding procedure consists of 4 steps:

1) flag points needing refinement

2) cluster the flagged points

3) generate a grid for each cluster

4) evaluate, possibly repeat.

The first step in the algorithm is to identify those grid points at level l which need to be in a finer grid at level $l+1$. In section (4.3), we discussed the use of local truncation error estimates in deciding where these refined meshes are needed. Using the procedure described there, we estimate the error at all grid points at level l , flagging those points $\tilde{x}$ where $e(\tilde{x}) > \epsilon$.

The second step of the algorithm is the separation of flagged points into distinct clusters. In step three, each cluster will be fitted with a fine grid containing all the flagged points of the cluster. Possibly, if a long enough gap of unflagged points is found, two or more separate subgrids may be formed instead.

Figure (4.3) illustrates the regridding procedure in one dimension. The x's are the grid points which have been flagged with high error estimates. We have used a buffer width of one coarse grid point in this illustration.

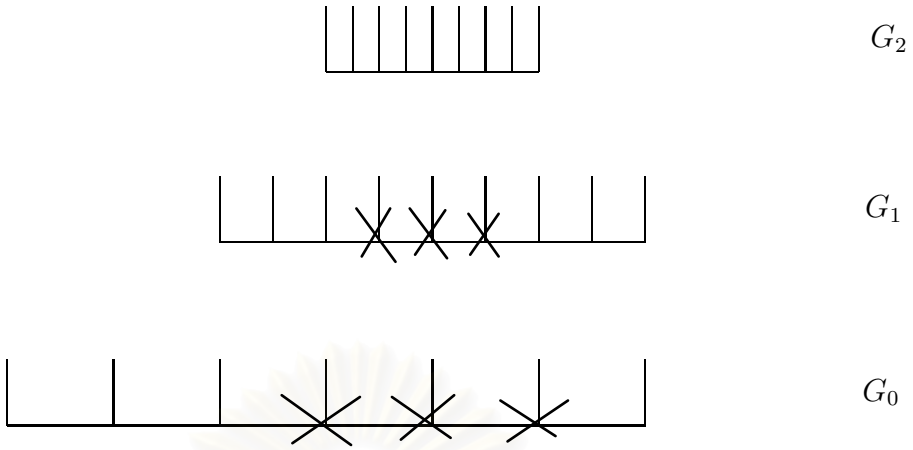

Figure 4.3 1D Regridding Algorithm

CHAPTER V

Numerical Results and Discussion

In this chapter we will present some numerical solutions to illustrate how the numerical methods and the mesh refinement algorithm work.

Let the initial data be

$$h(x,0) = \begin{cases} h_L, & \text{if } x < 0 \ , \\ h_R, & \text{if } x > 0 \ . \end{cases} \tag{5.0.1}$$

The wet bed case is now refered to the problem when $h_R \neq 0$. When $h_R = 0$, the problem is characterized as the dry-bed case. Next, we will give the exact solution for each case which we discuss them in Appendix A.

The exact solution for dry-bed case $h_L = 1$ and $h_R = 0$ is

$$h(x,t) = \begin{cases} 1, & \text{if } x \leq -t \ , \\ \frac{1}{9}(2 - \frac{x}{t})^2, & \text{if } -t \leq x \leq 2t \ , \\ 0, & \text{if } 2t \leq x \ . \end{cases} \tag{5.0.2}$$

For the wet-bed case $h_L = 1,\ h_R = a$, where $a < 1$, we have

$$h(x,t) = \begin{cases} 1, & \text{if } x \leq -t \ , \\ \frac{1}{9}(2 - \frac{x}{t})^2, & \text{if } -t \leq x \leq (u_2 - \sqrt{h_2})t \ , \\ h_2, & \text{if } (u_2 - \sqrt{h_2})t \leq x \leq Vt \ , \\ a, & \text{if } Vt \leq x \ , \end{cases} \tag{5.0.3}$$

where h_2 is compute by using Newton's method for a range of value of a. If $a = 0.6$, then $h_2 = 0.78661, u_2 = 0.22618$ and $V = 0.95340$. Details of (5.0.2) and (5.0.3) are presented in Appendix A.

Figure 5.1 shows the comparison between the Godunov's method and AMR method. The asteric symbols represent numerical solution obtained from the Godunov's method and the solid line represent solution obtained from the AMR method at time $t = 0.8$ with the initial height $h_L = 1$ and $h_R = 0.6$.

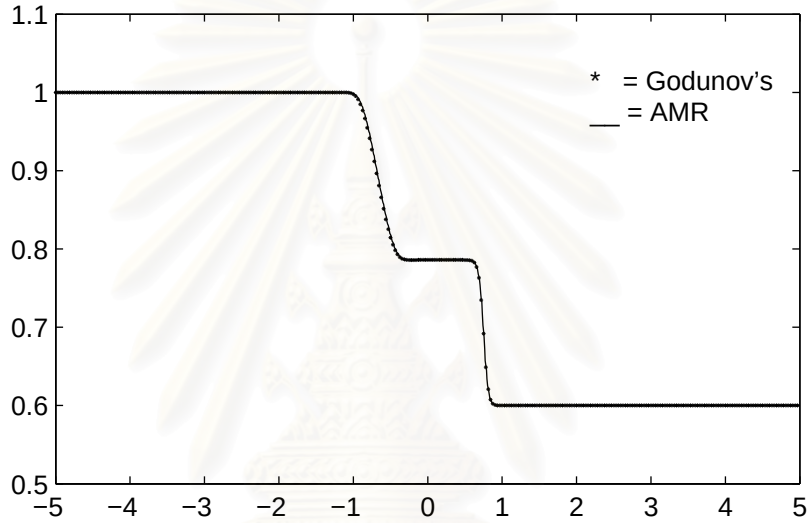

Figure 5.1 The comparison between the Godunov's method and AMR method

To measure the error from the calculation, we compute the l_2 norm of the error at only the coarse grid points. In one dimension, we have

$$\|error\|_2 = \sqrt{\frac{1}{n}\sum_{i=1}^{n} error(x = ih_c)^2}, \tag{5.0.4}$$

where $h_c = \Delta x$ of course grids. The computation time and the $\|\cdot\|_2$ error for the three computations at time $t = 0.5$(secs) with error tolerance 5×10^{-9} are given in Table 5.2.

method	h	$\|\cdot\|_2$	time(secs)
coarse	1/16	1.07×10^{-2}	0.02
AMR 3 level	1/64	0.91×10^{-2}	0.08
fine	1/64	0.56×10^{-2}	0.14

Table 5.2 The computation time and error at time $t = 0.5$(sec)

In Table 5.3 we show the computation time and error at time $t = 0.8$(secs) with error tolerance 5×10^{-9}.

method	h	$\|\cdot\|_2$	time(secs)
coarse	1/32	9.1×10^{-3}	0.15
AMR 3 level	1/128	8.8×10^{-3}	0.77
fine	1/128	4.4×10^{-3}	1.35

Table 5.3 The computation time and error at time $t = 0.8$(sec)

In the following sections, we will compare the numerical solutions calculated by the method in Chapter 3 with the exact solutions.

5.1. Wet-Bed Case

Figures 5.5 and 5.6 show typical profiles of the solution for $h_L = 1$ and $h_R = 0.6$. They represent the profiles of the water surface at $t = 2$ seconds after the dam break on the grid of 400 cells. The analytic solution is given in solid line. The dotted line represents the numerical solutions obtained from the Godunov's method in Figure 5.5 and the High-Resolution method in Figure 5.6. In Figure 5.5, the excessive numerical diffusion introduced by the first order scheme are shown. The High-Resolution method provides the necessary numerical viscosity to cope well both with the shock front and with the critical point at the dam location. Moreover, Figures 5.7 and 5.8 show typical profiles of the solution obtained from the Godunov's and the High-Resolution method respectively with source term. The roughness coefficient is set to

$n = 0.05$ and the bed slope is 0.05

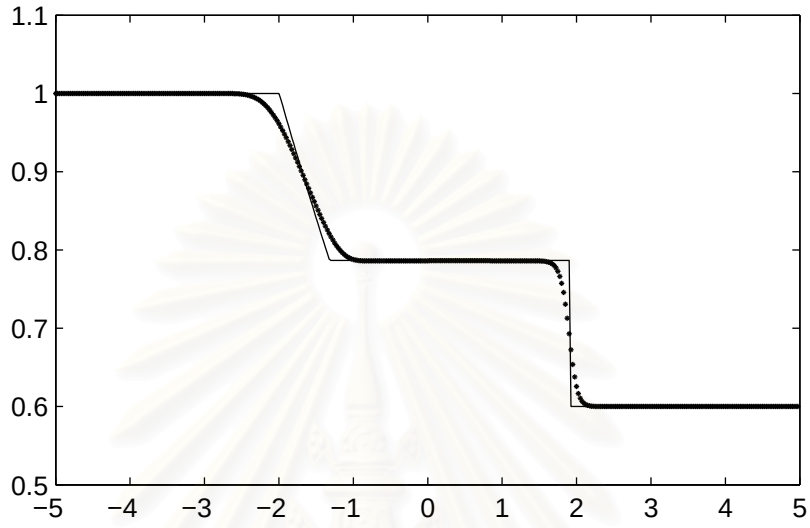

Figure 5.5 The solution solved by Godunov's method

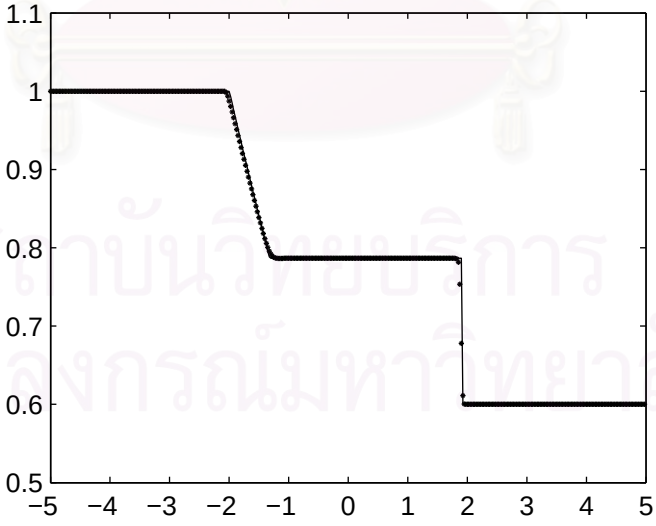

Figure 5.6 The solution solved by High-Resolution method

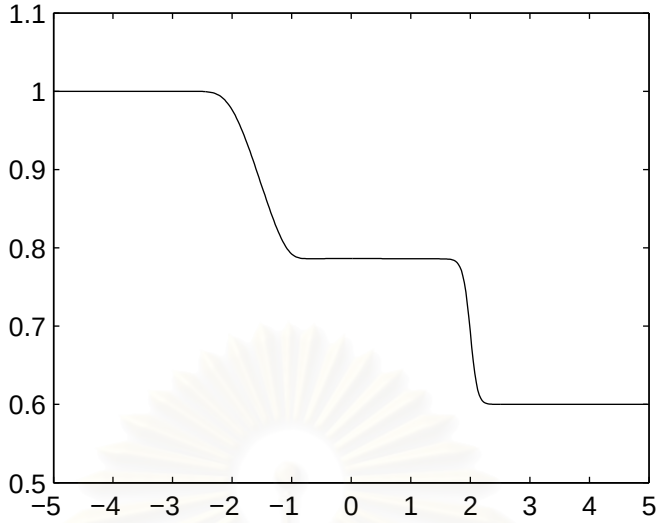

Figure 5.7 The solution solved by Godunov's method with source term

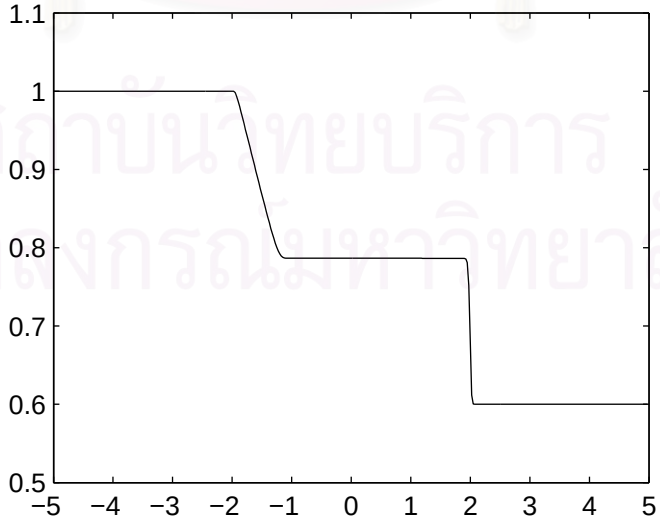

Figure 5.8 The solution solved by High-Resolution method with source term

5.2. Dry-Bed Case

Figure 5.9 depicts the analytic solution when $h_L = 1$ and $h_R = 0$ and Figure 5.10 shows the numerical solutions obtained by the HLL and HLLE solvers. Moreover, when we combine Figure 5.9 and Figure 5.10 into figure 5.11, we will see that the HLL and HLLE solvers can handle the solution.

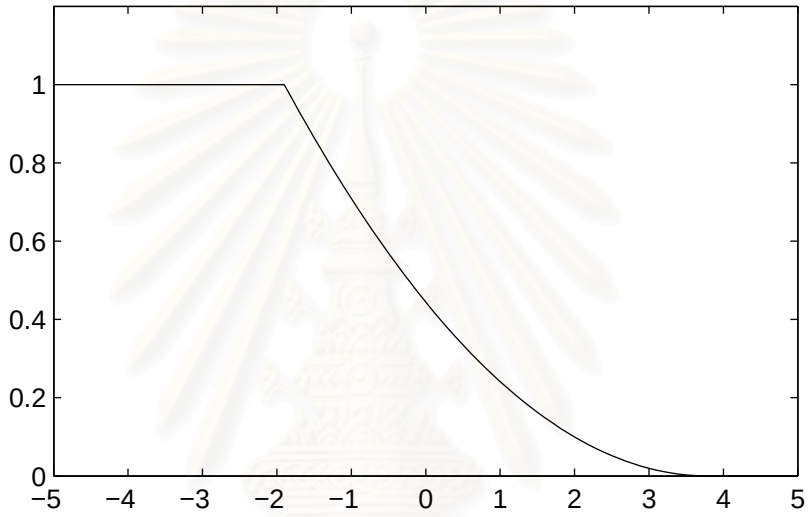

Figure 5.9 The analytic solution

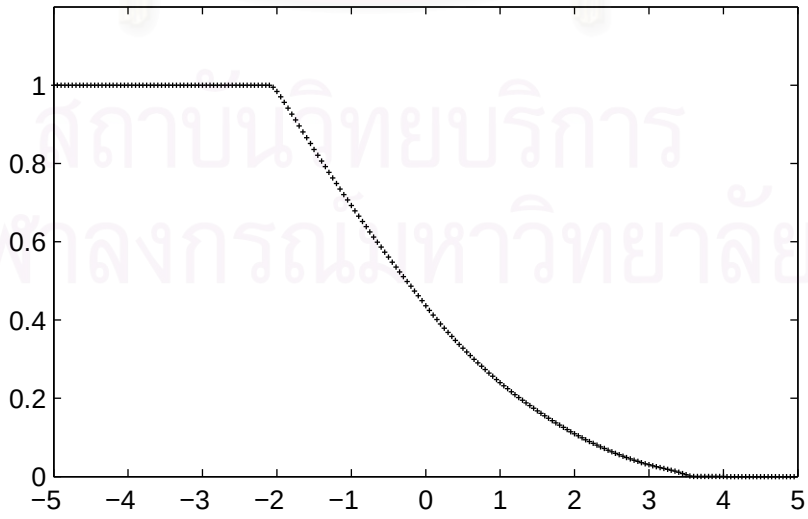

Figure 5.10 The solution solved by HLL and HLLE solvers

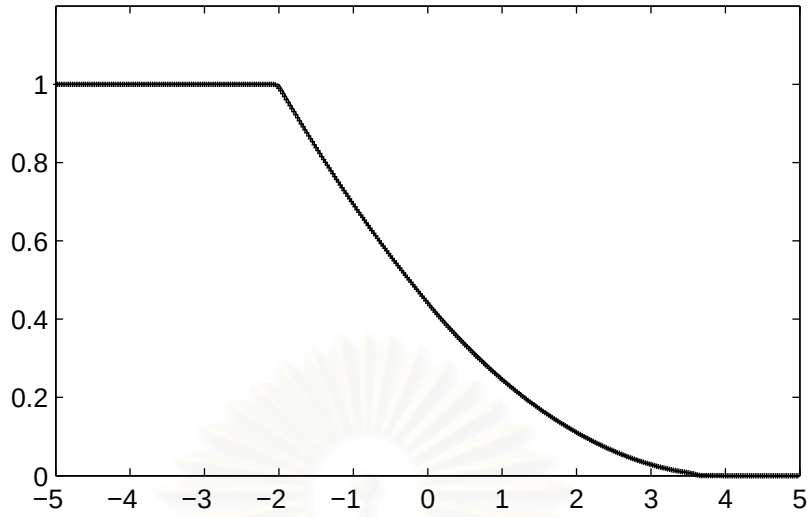

Figure 5.11 The analytic solution and the solution solved by HLL and HLLE solvers

CHAPTER VI

Conclusions

In this thesis we have considered numerical aspects of the unsteady flows governed by the Saint-Venant equations with particular emphasis on the discontinuous solutions.

We applied the Godunov's method, High-Resolution method and the HLL and HLLE solvers to solve the dam-break problem and compared the results with the exact solutions. The High-Resolution scheme gives better representations of the solutions in wet-bed case than the others. This is because the scheme is second order accurate unlike the others that are only first order accurate. Moreover, we use the HLL and HLLE solvers to solve the system of equation in dry-bed case because Godunov's and High-Resolution methods cannot handle the solutions in this case. In the HLL and HLLE solvers, the waves propagation in the fluctuation and correction terms of the High-Resolution method are modified. In addition, a mesh refinement approach can be an atternative to improve the efficiency in the numerical calculation. We conclude with a brief summary of the main features of our adaptive mesh refinement algorithm, and discuss the results.

We have presented an algorithm for the numerical solution of partial differential equations using automatic grid refinements. An automatic procedure which estimates the local truncation error determines the grid region to be included in the finer subgrids. We cluster the parts of the domain needing refinement by using the nearest neighborhood. Next, we have implemented the mesh refinement algorithm for one dimensional problem. The time spent in the calculations to solve with mesh refinement method is much less than those with uniform grid method (Godunov's method). Since

some regions in the AMR method need not be refined, the error that we obtained by using AMR method was dominated by the error that we obtained by using uniform grid (coarse grid) method.

REFERENCES

[1] P. P. Mujumdar. *Flood wave propagation the Saint Venant equations.* India: RESONANCE, 2001.

[2] I. Macdonald. *Analysis and computation of steady open channel flow.* University of Reading, 1996.

[3] R. J. Leveque. *Finite volume methods for hyperbolic problems.* New York: Cambridge University Press, 2002.

[4] M. J. Berger. *Adaptive mesh refinement for hyperbolic partial differential equations.* Ph.D. thesis. Michigan: UMI, 1982.

[5] J. J. Stoker. *Water waves.* New York: Interscience Publishers, 1957.

[6] Z. Christopher and R. Stephen. *Explicit schemes for Dam-Break simulation sanks.* ASCE J. Hydraulic Engineering, 2000.

[7] J. Kevorkian. *Partial Differential equations.* California: Wadsworth Brooks/Cole Advanced Books Software Pacific Grove, 1990.

[8] R. V. Churchill and J. W. Brown. *Complex variables and applications.* New York: McGraw-Hill, 1990.

APPENDICES

APPENDIX A

In this appendix we illustrate the results derived in Chapter 2 for the one-dimensional flow of Saint-Venant equations. The Saint-Venant equations are

$$h_t + (hu)_x = 0 \tag{1}$$

and

$$(hu)_t + (hu^2 + 1/2gh^2)_x = 0. \tag{2}$$

For smooth solutions equation (1) and (2) imply

$$h_t + uh_x + hu_x = 0 \tag{3}$$

and

$$u_t + uu_x + gh_x = 0 \tag{4}$$

respectively. Since the wave speed is $c = \sqrt{gh}$, $c_t = 1/2\sqrt{g/h}h_t$ and $c_x = 1/2\sqrt{g/h}h_x$. So (3) and (4) become

$$2c_t + 2uc_x + cu_x = 0 \tag{5}$$

and

$$u_t + uu_x + 2cc_x = 0 \tag{6}$$

$(6) \pm (5)$;

$$(u \pm 2c)_t + (u \pm c)(u \pm 2c)_x = 0 \tag{7}$$

We have the function $R_\pm = u \pm 2c$ are constant on the two set of characteristic curves $X_\pm(t)$, when

$$\lambda^\pm = \frac{d}{dt} X_\pm = u \pm c = u \pm \sqrt{gh}. \tag{8}$$

Here $R_\pm(u, c)$ are called the Riemann invariants of the system. That is,

On the C_+ characteristic, given by $\lambda^+ = u + c = u + \sqrt{gh}$, the C_+ invariant, $R_+ = u + 2\sqrt{gh}$, is constant

On the C_- characteristic, given by $\lambda^- = u - c = u - \sqrt{gh}$, the C_- invariant, $R_- = u - 2\sqrt{gh}$, is constant.

Dam-Break Problem

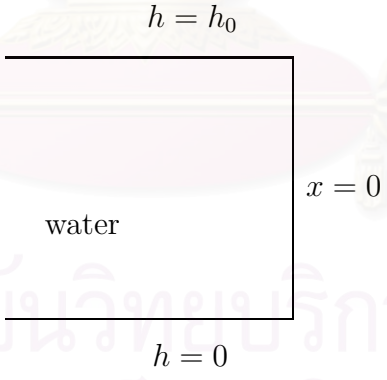

Figure A.1 Initial depth

Initial condition

$$u(x, 0) = 0$$

$$h(x,0) = \begin{cases} h_0, & \text{if } x < 0 \ , \\ 0, & \text{if } x > 0 \ . \end{cases} \tag{9}$$

On the characteristics that originate at $t = 0$, for $x < 0$,

$$R_{\pm} = u \pm 2\sqrt{gh} = \pm 2\sqrt{gh_0} = \pm 2c_0 \tag{10}$$

when $c_0 = \sqrt{gh_0}$ is the initial wave speed. Therefore, if C_+ and C_- characteristics from this region intersect, $u + 2\sqrt{gh} = 2c_0$ and $u - 2\sqrt{gh} = -2c_0$. Then $u = 0$ and $h = h_0$. In other words, the water is undisturbed at that point. In addition, $\lambda^{\pm} = u \pm \sqrt{gh} = \pm c_0$. That is the characteristics are straight lines. Such characteristics must lie in the region $x < -c_0 t$ and hence $u = 0, h = h_0$ in this region.

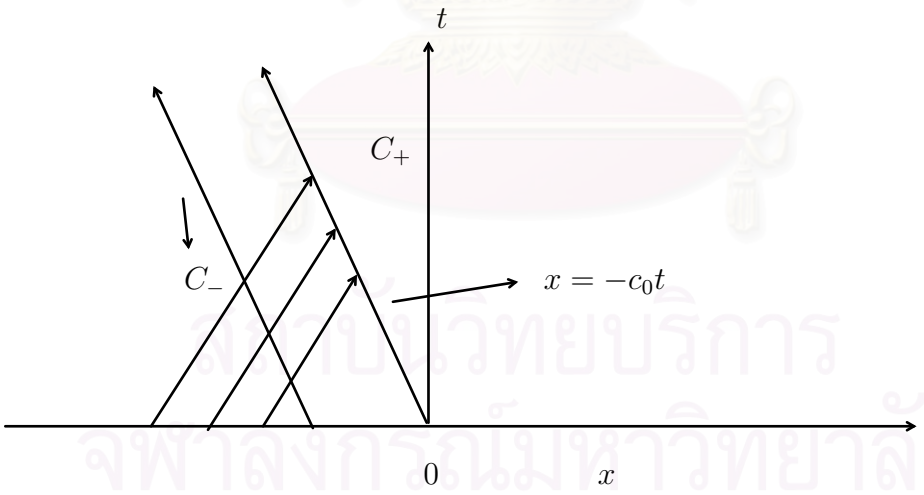

Figure A.2 The characteristics sketched in the region $x < -c_0 t$

Notice that the C_+ characteristics leave this region and enter the domain $x > -c_0 t$. For $x > -c_0 t$, the C_- characteristics satisfy

$$\lambda^- = u - \sqrt{gh} \tag{11}$$

and an each curve $R_- = u - 2\sqrt{gh}$ is constant. However, since we are assume that this region is filled by C_+ characteristics with $R_+ = u + 2\sqrt{gh} = 2c_0$, u and h must be constant on each C_- characteristics. Equation (9) then shows that λ^- is constant on each C_- characteristics, which must therefore be a straight line. Since the fluid only occupies the region $x < 0$ when $t = 0$, these C_- characteristics must start at the origin with $X_-(t) = (u - \sqrt{gh})t$

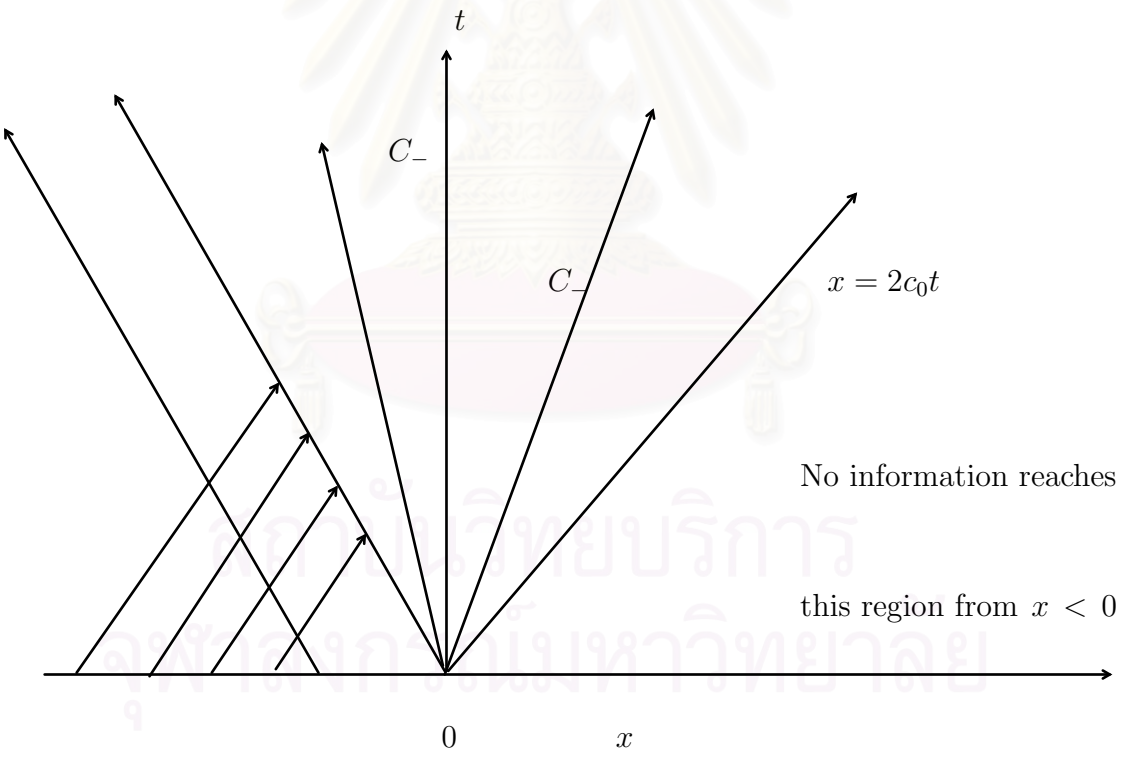

Figure A.3 The characteristics sketched in the region $x < 2c_0 t$

We are therefore shown that $R_+ = u + 2\sqrt{gh} = 2c_0$ and $u - \sqrt{gh} = \frac{x}{t}$ at each point in the domain $x > -c_0 t$. If we now solve these two equations for u and h, we find that

$$h = \frac{h_0}{9}(2 - \frac{x}{c_0 t})^2 \tag{12}$$

and

$$u = \frac{2}{3}(c_0 + \frac{x}{t}) \tag{13}$$

Note that this gives $h = 0$ when $x = 2c_0 t$ suggesting that no C_+ characteristics reach the region $x > 2c_0 t$ so that $u = h = 0$ there. Hence, the solution for u and h are

$$h(x,t) = \begin{cases} h_0, & \text{if } x \leq -c_0 t \ , \\ \frac{h_0}{9}(2 - \frac{x}{c_0 t})^2, & \text{if } -c_0 t \leq x \leq 2c_0 t \ , \\ 0, & \text{if } x \geq 2c_0 t \ , \end{cases} \tag{14}$$

and

$$u(x,t) = \begin{cases} 0, & \text{if } x \leq -c_0 t \ , \\ \frac{2}{3}(c_0 + \frac{x}{t}), & \text{if } -c_0 t \leq x \leq 2c_0 t \ , \\ 0, & \text{if } x \geq 2c_0 t \ . \end{cases} \tag{15}$$

For $h_0 = 1$, we have

$$h(x,t) = \begin{cases} 1, & \text{if } x \leq -t \ , \\ \frac{1}{9}(2 - \frac{x}{t})^2, & \text{if } -t \leq x \leq 2t \ , \\ 0, & \text{if } x \geq 2t \ . \end{cases} \tag{16}$$

that was used in chapter 6.

Dam-Break Problem over Water Downstream

The dam-breaking problem discussed above must be modified when there is a body of quiescent water of height $a < 1$ downstream $(x > 0)$. The initial conditions that replace (9) are now

$$u(x,0) = 0$$

$$h(x,0) = \begin{cases} 1, & \text{if } x < 0\ , \\ a < 1, & \text{if } x > 0\ . \end{cases} \tag{17}$$

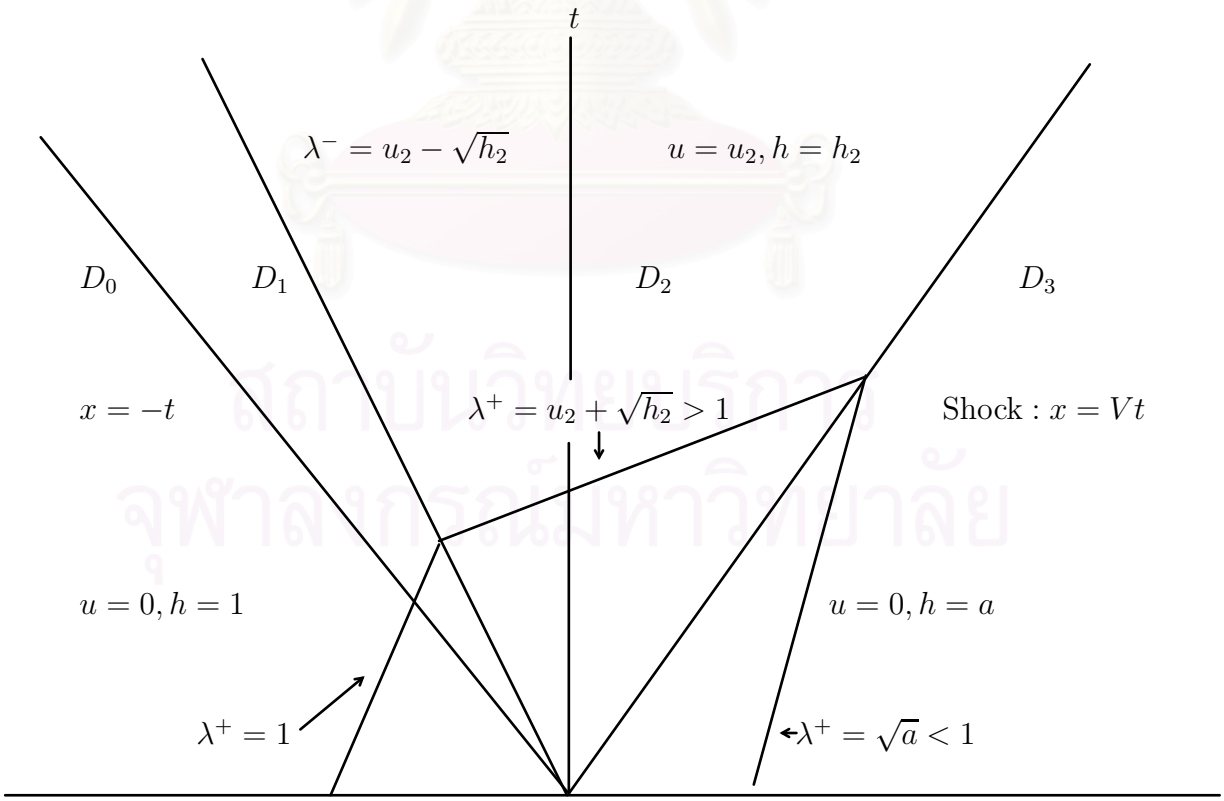

Fig A.4 The characteristics sketched in the domain for wet-bed case

We recall that for $a \to 0$, we had a discontinuity in u propagating along the bounding λ^+ characteristics, but h was continuous across this characteristic. Now, for $t = 0$, $\lambda^+ = \sqrt{a} < 1$ for $x > 0$, and $\lambda^+ = 1$ for $x = 0^-$, where $g = 1$. Then there are u_2 and h_2 such that

$$\lambda^+ = u_2 + \sqrt{h_2} \text{ and } \lambda^- = u_2 - \sqrt{h_2} \text{ in } D_2. \tag{18}$$

Since $\lambda^+ = u + \sqrt{h}$ and $\lambda^- = u - \sqrt{h}$, $u = u_2$ and $h = h_2$ in D_2. In D_0 and D_1, the solution is exactly the one given by (16). Actually, we shall see from our results that $\lambda^+ = u_2 + \sqrt{h_2} > 1$ in D_2 and $\lambda^+ = \sqrt{a} < 1$ in D_3 emerging from either side of the origin immediately cross. Then, a shock must start out from there (see figure A.1). Next, we will find the shock speed V. We do by combining the information provided by the Rankine-Huginiot condition in section 2.5 which must persist into D_2. For our case, the shock conditions are:

$$V = \frac{u_2 h_2}{h_2 - a}$$
$$V = \frac{u_2^2 h_2 + h_2^2/2 - a^2/2}{u_2 h_2}.$$

Eliminating V gives

$$\frac{u_2 h_2}{h_2 - a} = \frac{u_2^2 h_2 + h_2^2/2 - a^2/2}{u_2 h_2}$$

$$h_2^3 - a h_2^2 - (a^2 + 2a u_2^2) h_2 + a^3 = 0. \tag{19}$$

As in the case $a = 0$, the Reimann invariant [7]

$$u + 2\sqrt{h} = 2 \tag{20}$$

must hold in the entire domain D_0, D_1, D_2 to the left of the bore, covered by the λ^+ characteristic emerging from $t = 0,\ x < 0$. Now, using (20) to express u_2 in terms of h_2 in (19) gives

$$h_2^3 - 9ah_2^2 + 16ah_2^{3/2} - (a^2 + 8a)h_2 + a^3 = 0. \tag{21}$$

We compute h_2 using Newton's method for a range of values of a. Having h_2, we obtain u_2 from (20) and the slope from

$$\lambda^- = u_2 - \sqrt{h_2}. \tag{22}$$

This results are tabulated next for a range of values of a.

a	h_2	u_2	V	λ^-
0.9	0.94933	0.05132	0.98763	-0.92302
0.8	0.89715	0.10564	0.97555	-0.84154
0.7	0.84309	0.16360	0.96394	-0.67949
0.6	0.78661	0.22618	0.95340	-0.56043

Finally, the solution of u and h are

$$h(x,t) = \begin{cases} 1, & \text{if } x \le -t\ , \\ \frac{1}{9}(2 - \frac{x}{t})^2, & \text{if } -t \le x \le (u_2 - \sqrt{h_2})t\ , \\ h_2, & \text{if } (u_2 - \sqrt{h_2})t \le x \le Vt\ , \\ a & \text{if } Vt \le x\ , \end{cases} \tag{23}$$

and

$$u(x,t) = \begin{cases} 0, & \text{if } x \leq -t \ , \\ \frac{2}{3}(1+\frac{x}{t}), & \text{if } -t \leq x \leq (u_2 - \sqrt{h_2})t \ , \\ u_2, & \text{if } (u_2 - \sqrt{h_2})t \leq x \leq Vt \ , \\ 0 & \text{if } Vt \leq x \ . \end{cases} \tag{24}$$

APPENDIX B

Here, we illustrate the perturbation method introduced in Chapter 2. The type of problem considered in this chapter belongs in the category of problems concerned with motions in their early stages after initial impulses have been applied. A typical example is the motion of the water in a dam when the dam is suddenly broken. Consequently we choose the quantities a, b, and t as independent variables, with a and b representing Cartesian coordinates of the initial positions of fluid particle at the time $t = 0$. We denote $X(a, b; t)$ and $Y(a, b; t)$, as the displacement of fluid particle and $p(a, b; t)$ as the pressure.

From Newton's second law, the equations of motion are

$$X_{tt} = -\frac{1}{\rho} p_X \tag{25}$$

$$Y_{tt} = -\frac{1}{\rho} p_Y - g. \tag{26}$$

All subscripts here denote the differentiation.We assume gravity to be the only external force.

To eliminate the pressure we multiply (25) and (26) by X_a and Y_a, respectively, and add, then also by X_b, Y_b and add. Hence,

$$X_{tt}X_a + (Y_{tt} + g)Y_a + \frac{1}{\rho}[p_X X_a + p_Y Y_a] = 0.$$

Therefore,

$$X_{tt}X_a + (Y_{tt} + g)Y_a + \frac{1}{\rho}p_a = 0, \tag{27}$$

and

$$X_{tt}X_b + (Y_{tt} + g)Y_b + \frac{1}{\rho}[p_X X_b + p_Y Y_b] = 0.$$

Thus

$$X_{tt}X_b + (Y_{tt} + g)Y_b + \frac{1}{\rho}p_b = 0. \tag{28}$$

From the condition of continuity, we have

$$X_a Y_b - X_b Y_a = 1. \tag{29}$$

If the pressure p is eliminated from (27) and (28) by differentiation the result is

$$X_{ttb}X_a + X_{ab}X_{tt} + (Y_{tt}Y_a)_b + gY_{ab} + \frac{1}{\rho}p_{ab} = 0, \tag{30}$$

and

$$X_{tta}X_b + X_{ab}X_{tt} + (Y_{tt}Y_b)_a + gY_{ab} + \frac{1}{\rho}p_{ab} = 0, \tag{31}$$

substract (30) by (31);

$$X_{ttb}X_a + (Y_{tt}Y_a)_b = X_{tta}X_b + (Y_{tt}Y_b)_a,$$

$$(X_{tb}X_a + Y_{tb}Y_a)_t = (X_{ta}X_b + Y_{ta}Y_b)_t. \tag{32}$$

Integration (32) with respect to t leads to

$$(X_{tb}X_a + Y_{tb}Y_a) - (X_{ta}X_b + Y_{ta}Y_b) = f(a,b) \tag{33}$$

where f is an arbitrary function. If the fluid starts from rest, or from any other state with vanishing vorticity, the function $f(a,b)$ would be zero.

Assume that solution exist in the form of power series, with coefficients which depend on a and b:

$$X(a,b;t) = a + X^{(1)}(a,b)t + X^{(2)}(a,b)t^2 + ...$$

$$Y(a,b;t) = b + Y^{(1)}(a,b)t + Y^{(2)}(a,b)t^2 + ...$$

$$p(a,b;t) = p^{(0)}(a,b) + p^{(1)}(a,b)t + p^{(2)}(a,b)t^2 + ... \tag{34}$$

Substituting (34) into equation (29) and the coefficient of each power of t is equated to zero with the following result for the first two terms:

$$\Big[(1 + x_a^{(1)}t + x_a^{(2)}t^2 + ...)(x_b^{(1)}t + 2x_b^{(2)}t + ...) + (y_a^{(1)}t + y_a^{(2)}t^2 + ...)(y_b^{(1)}t + 2y_b^{(2)}t + ...)\Big] -$$
$$\Big[(x_b^{(1)}t + x_b^{(2)}t^2 + ...)(x_a^{(1)} + 2x_a^{(2)}t + ...) + (1 + y_b^{(1)}t + y_b^{(2)}t^2 + ...)(y_a^{(1)} + 2y_a^{(2)}t + ...)\Big] = 0,$$

$$t^0 : \ X_a^{(1)} + Y_b^{(1)} = 0,$$
$$t^1 : \ X_a^{(2)} + Y_b^{(2)} = -(X_a^{(1)}Y_b^{(1)} - X_b^{(1)}Y_a^{(1)}).$$

Since $X^{(1)}$ and $Y^{(1)}$ have to satisfy the above relation, they cannot be prescribed arbitrarily. However, in the case of dam-break problem the fluid starts from rest and

$X^{(1)} = Y^{(1)} = 0$. Consequently, we have

$$X_a^{(2)} + Y_b^{(2)} = 0. \tag{35}$$

In general, $X^{(n)}$ and $Y^{(n)}$ would satisfy an equation of the form

$$X_a^{(n)} + Y_a^{(n)} = F(X^{(1)}, Y^{(1)}, X^{(2)}, Y^{(2)}, ..., X^{(n-1)}, Y^{(n-1)}),$$

with F a nonlinear function in

$$X^{(i)}, Y^{(i)}, i = 1, 2, ..., n-1.$$

In the following we shall consider only motions starting from rest. Thus,

$$X^{(1)} = Y^{(1)} = 0,$$

and equation (33) holds with $f \equiv 0$; a substitution of (34) in (33) yields (for the lowest order term):

$$X_b^{(2)} - Y_a^{(2)} = 0. \tag{36}$$

Initial Condition

We assume that the region occupied initially by the water is the half-strip $-\infty < a \leq 0, 0 \leq b \leq$ h.The dam is of course located at $a = 0$. Since we assume that the water is initially at rest, when filling the half-strip we have the conditions

$$X(a, b; 0) = a, \quad Y(a, b; 0) = b, \tag{37}$$

$$X_t(a, b; 0) = 0, \quad Y_t(a, b; 0) = 0. \tag{38}$$

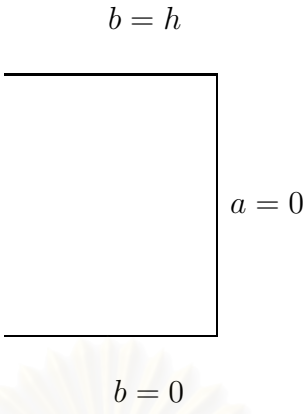

Figure B.1 Fluid domain at $t = 0$

Boundary Condition

The boundary condition at the bottom $b = 0$ results from the assumption that the water particles originally at the bottom remain in contact with it; as a result we have the boundary condition

$$Y(a, 0; t) = 0, \quad -\infty < \ a \leq 0, \ t > 0. \tag{39}$$

Since $0 = Y(a, 0; t) = Y^{(1)}(a, 0)t + Y^{(2)}(a, 0)t^2 + ..., \ \ t > 0,$ then $Y^{(n)}(a, 0) = 0$ for all n.

When the dam is broken, the pressure along it will be changed suddenly from hygrostatic pressure to zero; it will of course be prescribed to be zero on the free surface. This leads to the following boundary conditions for the pressure:

$$p(a, h; t) = 0, \quad 0 \leq \ a < \infty, t > 0, \tag{40}$$

$$p(0, b; t) = 0, \quad 0 \leq \ b \leq h, t > 0. \tag{41}$$

From (27), insertion of the series (34) and use of the boundary conditions for $b = h$ yields

$$X^{(2)}(a, h) = 0, \tag{42}$$

and using the equation (28), then

$$Y^{(2)}(0, b) = -\frac{g}{2}. \tag{43}$$

From (35) and (36), $X^{(2)}$ is a harmonic conjugate of $Y^{(2)}$. Then $Z(z) = Y^{(2)} + iX^{(2)}$ is an analytic function of the complex variable $z = a+ib$ in the half-strip, and we now have prescribed values for either its real or its imaginary part on each of the three sides of the strip; it follows that the function Z can be determined by the standard method, namely conformal mappings. In fact, the solution can be given in closed form, as follows: Since $X^{(2)}(a, h) = 0$, we see that $X^{(2)}_a(a, h) = 0$, and hence that $Y^{(2)}_b(a, h) = 0$ since $X^{(2)}$ and $Y^{(2)}$ are harmonic conjugates. Therefore the hormonic function $Y^{(2)}(a, b)$ can be continued over the line $b = h$ by reflection into a strip of width $2h$, as indicated in Figure B.2; the boundary values for $Y^{(2)}$ are also shown.

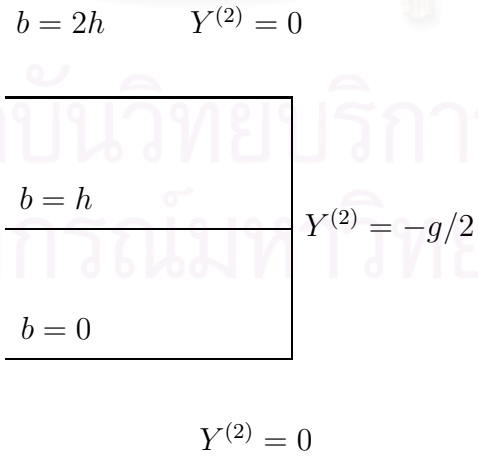

Figure B.2 Boundary value problem for $Y^{(2)}(a, b)$

Thus a completely formulated boundary value problem for $Y^{(2)}(a,b)$ in a half-strip has been derived. To solve this problem we map the half-stip on the upper half of a w-plane by means of the Schwarz-Chistoffel mapping formula and observe that the vertices $z=0$ and $z=2h$ of the half- strip map into the points $w=\mp 1$ of the $w-$ plane, as indicated in Figure B.3. The appropriate boundary values for $Y^{(2)}(w)$ on the real axis of the $w-$ plane are indicated.

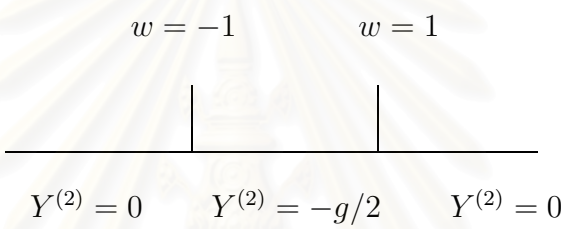

Figure B.3 Mapping on the $w-$ plane

Then

$$z = A\int \frac{dw}{\sqrt{w^2-1}} + B$$
$$= A\cosh^{-1} w + B$$

Since $w=1$ at point $z=2hi, A\cosh^{-1}(1)+B=2hi$. Then $B=2hi$, and $w=-1$ at point $z=0$. Hence $A\cosh^{-1}(-1)+B=0$. Then $A=-2h/\pi$. Therefore, we have

$$z=\frac{-2h}{\pi}\cosh^{-1} w + 2hi,$$
$$w=\cosh(\pi i - \frac{z\pi}{2h}).$$

Since

$$w = u + vi = \cosh(\pi i - \frac{(a+bi)\pi}{2h}),$$
$$u = -\cosh(\frac{\pi a}{2h})\cos(\frac{\pi b}{2h}) \text{ and } v = -\sinh(\frac{\pi a}{2h})\sin(\frac{\pi b}{2h}).$$

Using Poisson's formula [8] for the half plane, we have

$$\begin{aligned} X^{(2)}(u,v) &= \frac{1}{\pi}\int_{-1}^{1} \frac{(u-\eta)(-g/2)d\eta}{v^2+(\eta-u)^2} \\ &= \frac{-g}{4\pi} ln\left[\frac{(u+1)^2+v^2}{(u-1)^2+v^2}\right] \\ &= \frac{-g}{4\pi} ln\left[\frac{(-\cosh(\frac{\pi a}{2h})\cos(\frac{\pi b}{2h})+1)^2+\sinh(\frac{\pi a}{2h})\sin(\frac{\pi b}{2h})^2}{(-\cosh(\frac{\pi a}{2h})\cos(\frac{\pi b}{2h})-1)^2+\sinh(\frac{\pi a}{2h})\sin(\frac{\pi b}{2h})^2}\right], \end{aligned}$$

and

$$\begin{aligned} Y^{(2)}(u,v) &= \frac{1}{\pi}\int_{-1}^{1} \frac{v(-g/2)d\eta}{v^2+(u-\eta)^2} \\ &= \frac{-g}{2\pi}\left[\arctan\left(\frac{1-u}{v}\right) - \arctan\left(\frac{-1-u}{v}\right)\right] \\ &= \frac{-g}{2\pi}\left[\arctan\left(\frac{1+\cosh(\frac{\pi a}{2h})\cos(\frac{\pi b}{2h})}{-\sinh(\frac{\pi a}{2h})\sin(\frac{\pi b}{2h})}\right) - \arctan\left(\frac{-1+\cosh(\frac{\pi a}{2h})\cos(\frac{\pi b}{2h})}{-\sinh(\frac{\pi a}{2h})\sin(\frac{\pi b}{2h})}\right)\right] \\ &= \frac{-g}{\pi}\arctan\left(\frac{-\sin(\frac{\pi b}{2h})}{\sinh(\frac{\pi a}{2h})}\right). \end{aligned}$$

Therefore, the shape of the free surface of the water can be obtained for small times from the equations

$$X = a + X^{(2)}t^2, \qquad Y = b + Y^{(2)}t^2 \tag{44}$$

evaluated for $a = 0$ (for the particles at the face of the dam) and for $b = h$ on the upper free surface. The results of such a calculation for the specific case of a dam 70

meters high are shown in Figure B.4.

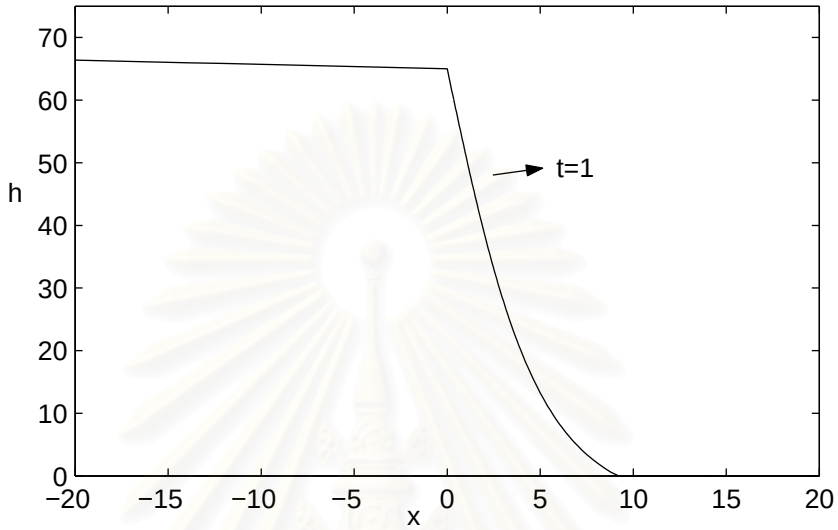

Figure B.4 Free water surface after the breaking of a dam

APPENDIX C

In this appendix, we implement the adaptive mesh refinement algorithm for solving the Saint-Venant equations. In this program, we have only the main program and exclude all other subroutines. We separate the program into two main parts. Firstly, we have initial conditions prescribed in the *boundary* subroutines. Then, we call the *calculate* subroutine for receiving the solutions. After that, we check the error by using the Richardson extrapolation. Secondly, if the error is less than the tolerence, the program stops. If not, we will refine grid only at the mesh that the error is more than tolerance and so on until the error is less than the tolerance for all mesh points or until it reaches the third level.

```
program adaptive mesh refinement

implicit double precision (a-h,o-z)
external boundaryleft,boundaryrigh,calculate,count,calregrid,grid
parameter (maxmx = 20000)
parameter (maxregrid = 20)
parameter (mbc = 2)
parameter (meqn = 2)
parameter (mwaves = 2)
parameter (maxlevel=6)

dimension q(1-mbc:maxmx+mbc, meqn)
dimension qh1(1-mbc:maxmx+mbc, meqn,0:maxlevel)
dimension qh2(1-mbc:maxmx+mbc, meqn)
dimension qu(1-mbc:maxmx+mbc, meqn)
dimension qu2(1-mbc:maxmx+mbc, meqn)
dimension qb(1-mbc:maxmx+mbc, meqn)
dimension qb1(1-mbc:maxmx+mbc, meqn)
dimension qre(1-mbc:maxmx+mbc, meqn,0:maxlevel)
dimension qre1(1-mbc:maxmx+mbc, meqn,0:maxlevel)
dimension iqerr(maxmx)
dimension ii(maxmx)
dimension x(maxmx,0:maxlevel)
dimension iwtemp1(maxmx)
dimension iwtemp2(maxmx)
dimension ikeep1(maxmx,0:maxlevel)
dimension xleft(maxregrid)
dimension xright(maxregrid)
dimension xkeep(maxregrid)
dimension it1(maxregrid)
dimension dxl(0:maxlevel)
dimension dtl(0:maxlevel)
dimension mxl(0:maxlevel)
dimension inumre(maxlevel)
dimension mxre(maxlevel,maxregrid)
dimension itemp11(maxlevel,maxregrid)
dimension itemp22(maxlevel,maxregrid)
dimension ixleft(maxlevel,maxregrid)
dimension ixright(maxlevel,maxregrid)
dimension id(maxmx)
dimension iid(maxmx)
dimension xx(maxmx)
dimension numjj(maxregrid)
dimension num2(0:maxregrid)
dimension ibound(maxregrid)
dimension iibound(maxregrid)
dimension iidd(maxregrid,maxregrid)
dimension iimx(maxregrid,maxregrid)
dimension mthlim(mwaves),mthbc(2)
dimension method(7)
dimension    f(1-mbc:maxmx+mbc, meqn)
dimension    s(1-mbc:maxmx+mbc, mwaves)
dimension wave(1-mbc:maxmx+mbc, meqn, mwaves)
dimension amdq(1-mbc:maxmx+mbc, meqn)
dimension apdq(1-mbc:maxmx+mbc, meqn)
dimension dtdx(1-mbc:maxmx+mbc)
dimension delta(2)
character*10 fname1
logical limit
logical efix
data efix /.true./    !# use entropy fix for transonic rarefactions
data limit /.true./   !# use limiter
do i=1,mwaves
   mthlim(i)=4        !# choose limiter
```

```
                              !# 1= minmod
                              !# 2= superbee
                              !# 3= van leer
                              !# 4= monotinized centered
                              !# 5= Beam-Warming
      enddo
      mthbc(1)=1              !# choose boundary condition
      mthbc(2)=1              !# 0= define boundary condition by yourself
                              !# 1= zero-order extrapolation
                              !# 2= periodic
                              !# 3= solid wall
      method(2)=1             !# 1= Godunov (first order)
                              !# 2= High-resolution method (second order)
                              !# if method(2)=2 and mthlim(i)=0 then the scheme is
      if ((mthbc(1).eq.2 .and. mthbc(2).ne.2) .or.
     &    (mthbc(2).eq.2 .and. mthbc(1).ne.2)) then
         write(*,*) '*** ERROR ***  periodic boundary conditions'
         write(*,*) ' require mthbc(1) and mthbc(2) BOTH be set to 2'
         stop
         endif

      xlower=-5.0d0           !# left domain of x
      xupper = 5.0d0          !# right domain of x
      mx=320                  !# number of the grid points
      t0=0.0d0                !# initial time
      tfinal=1.0d0            !# final time
      mt=500.0d0              !# number of loops of time
      grav= 1.e0              !# gravity

c     # grid spacing
      dx = (xupper - xlower) / float(mx)
      dt =  (tfinal- t0)/float(mt)

c     used in amr
      nlevel=0
      level=0
      sloc=0.0e0              !# stay in the middle and set mx is an even number
      refineratio=2           !# refinement ratio for each fined grid
      tol=0.0005
      ibadpt=2
      iorder=method(2)        !# order of accuracy of method
      itemp=1
      nn=0
      icheck=0
      icheck1=0
      icheck2=0
      iicheck=0
      icheck3=0
      iup=0
      do j=1,maxlevel
      do i=1,maxregrid
        ibound(i)=0
        iibound(i)=0
        ixleft(j,i)=-5
        ixright(j,i)=-5
      enddo
      enddo
      mlevel=0
      lelevel=1               !# number of level
      mxll=-4

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     # set initial conditions:
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      sloc= 0.0e0
      hl = 3.0e0
      ul = 0.0e0
      hr = 1.0e0
      ur = 0.0e0

c      # data in left state:
       hul = hl*ul

c      # data in right state:
       hur = hr*ur
c
      do 1 i=1,mx
           xedge = xlower + (i-1)*dx
           if (xedge .lt. sloc) then
           q(i,1) = hl
           q(i,2) = hul
          else
           q(i,1) = hr
           q(i,2) = hur
          endif
  1        continue

      do m=1,meqn
          do i=1,mx
              qre(i,m,level)=q(i,m)
          enddo
          enddo

c     save for new round
       mx1=mx
       dx1=dx
       dt1=dt

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c       boundary cells:
      do  n=1,mt


      do m=1,meqn
          do i=1,mx
              q(i,m)=qre(i,m,0)
          enddo
      enddo

  3     continue  !# use for regrid

         nn=1

      do while (nn.le.mt)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      call boundaryleft(mthbc,meqn,mbc,q,maxmx,mx)

      call boundaryrigh(mthbc,meqn,mbc,q,maxmx,mx)

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      call calculate(maxmx,mbc,meqn,mwaves,q,mthlim,mthbc,
     $                       method,efix,limit,mx,t0,tfinal,
     $                  mt,grav,dx,dt)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      if (itemp.eq.3) then
          nn=mt+1
          itemp=1
          go to 4
      endif

      if (nn.eq.1) then
          do m=1,meqn
              do i=1,mx
                  qh1(i,m,0)=q(i,m)       !# q with 1 iteration
              enddo
          enddo
          nn=nn+1
      else
          do m=1,meqn
              do i=1,mx
                  qh2(i,m)=q(i,m)
              enddo
          enddo
          nn=mt+1
      endif


ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      enddo       !# loop do while (nn.le.mt)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c     set dx and dt for coarser grids

      dx=2.0d0*dx
      dt=2.0d0*dt
      mx=mx/2
      ddx=dx
      ddt=dt
      dmx=mx

c     initial for coarser grids
         do m=1,meqn
          do i=1,mx
            q(i,m)=(qre(2*i-1,m,level)
     $                 +qre(2*i,m,level))/2.0d0
          enddo
         enddo
      itemp=3
      go to 3
 4      continue

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     check error

      order= float(2**(iorder+1)-2)
      nlevel=0
      do m=1,meqn
          do i=1,mx
              iqerr(i)=0
              aver=(qh2(2*i-1,m)+qh2(2*i,m))/2.0d0
              err = dabs((aver-q(i,m))/order)
              if (err.gt.tol) then
                  iqerr(i)=ibadpt
                  nlevel=1
              endif
          enddo
      enddo
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      do i=1,mx
          ikeep1(i,level)=iqerr(i)
      enddo

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c        set  for level 0
          dxl(level)=dx/2.0d0
          dtl(level)=dt/2.0d0
          mxl(level)=mx*2

c      set for all level
         do i=1,maxlevel+1
          dxl(level+i)=dxl(level+i-1)/2.0d0
          dtl(level+i)=dtl(level+i-1)/2.0d0
          mxl(level+i)=mxl(level+i-1)*2
         enddo
         do j=1,maxlevel
           do m=1,meqn
            do i=1,mxl(j-1)
              qre(2*i-1,m,j)=qre(i,m,j-1)
              qre(2*i,m,j)=qre(i,m,j-1)

            enddo
           enddo
         enddo

         do i=1,mxl(maxlevel)
            id(i)=0
         enddo

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c-----------------------------
c     count number of level
c-----------------------------

  71    continue
        if ((nlevel.eq. 1).and.(level.lt.lelevel)) then

          call count(mx,level,iqerr,ibadpt,
     $               ii,numre,xleft,xlower,xkeep,xright,
     $               dx,maxmx,maxregrid,maxlevel,icheck3,inumre)

          call calregrid(numre,xright,xleft,xkeep,meqn,qre,xlower,
     $                   xupper,mbc,iorder,qh1,tol,ibadpt,iqerr,
     $                   nlevel,maxregrid,maxmx,mthbc,mwaves,
     $                   mthlim,method,efix,limit,t0,tfinal,mt,grav,
     $                   dxl(level),dtl(level),level,maxlevel,
     $                   iicheck,ikeep1,icheck1,itemp11,itemp22,mxre)

          do j=1,inumre(level)
             ixleft(level,j)=itemp11(level,j)
             ixright(level,j)=itemp11(level,j)+mxre(level,j)-1
          enddo

          if(level.gt.1) then
           do jj=1,inumre(1)
            do j=1,inumre(level)
             if(itemp11(level,j).eq.itemp11(level-1,jj)*2-1) then
                ibound(j)=2                !# left boundary regrid
             endif

             ia=itemp11(level,j)+mxre(level,j)-1
```

```
                  ib=(itemp11(level-1,jj)+mxre(level-1,jj)-1)*2
            if(ia.eq.ib) iibound(j)=2     !#  right boundary regrid
          enddo
         enddo
         endif

         dx=dxl(level-1)
         dt=dtl(level-1)
         mx=mxl(level-1)
         go to 71
       endif

      mlevel=level
      mmlevel=mxl(mlevel)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

         do j=0,mlevel
          do i=1,mxl(j)+1
                  x(i,j)=xlower+(i-1)*dxl(j)
          enddo
         enddo
         do i=1,mxl(mlevel)
            iwtemp1(i)=0
            iwtemp2(i)=0
         enddo
            do le=1,mlevel
              do j=1,inumre(le)
               if(le.eq.1) then
               do i=1,mxl(mlevel)
                      if(i.eq.itemp11(le,j)*2-1) then
                        do i1=1,mxre(le,j)*2
                          iwtemp1((itemp11(le,j)*2-1)+i1-1)=1
                        enddo
                      endif
               enddo
               else
                 do i=1,mxl(mlevel)
                  if(i.eq.itemp11(le,j)) then
                   do i2=1,mxre(le,j)
                    iwtemp2(itemp11(le,j)+i2-1)=2
                   enddo
                  endif
                 enddo
               endif
              enddo
             enddo

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      levellevel=2**mlevel
      do m=1,meqn
          im=0
          do i=1,mxl(0)
              do j=1,levellevel
                 qu(im+j,m)=qh1(i,m,0)
              enddo
             im=im+2**mlevel
            enddo
      enddo

      if(1.le.mlevel) then
       do m=1,meqn
            do i=1,mxl(0)
               qb1(2*i-1,m)=(qre(i,m,0)+qh1(i,m,0))/2.0d0
```

```
               qb1(2*i,m)=(qre(i,m,0)+qh1(i,m,0))/2.0d0
             enddo
          enddo
         endif
c        get qb for level 1

         if(2.le.mlevel) then
                do m=1,meqn
                 do i=1,mxl(1)
                  qb(2*i-1,m)=(qb1(i,m)+qre(i,m,1))/2.0d0
                  qb(2*i,m)=(qb1(i,m)+qre(i,m,1))/2.0d0
                 enddo
                enddo
         endif

         do i=1,mxl(mlevel)
            iqerr(i)=0
         enddo

c        -------------------------------------
c        calculate for regrid
c        -------------------------------------

         level=1

         if(level.le.mlevel) then
            do nb1=1,refineratio
              if((nb1.eq.1).or.((nb1.eq.2).and.(2.gt.mlevel))) then
               do j= 1,inumre(level)
                  imxre=mxre(level,j)
                  iitemp11=itemp11(level,j)
                  iitemp22=itemp22(level,j)
                  if(nb1.eq.2) then
                    iicheck=4
                    icheck1=5
                  endif
                  call grid(level,meqn,imxre,q,qre,iitemp11,
     $                 iicheck,qh1,ixleft,xlower,mbc,maxmx,ixright,xupper,
     $                 maxregrid,maxlevel,icheck1,icheck2,iitemp22,icheck3,
     $                 mwaves,mthbc,method,efix,limit,t0,tfinal,mt,grav,
     $                 dxl(level),dtl(level),j,qre1,ibound,iibound,qb,n,nb1,
     $                 mthlim,mxl(level))

c                 update
                  if((nb1.eq.2).and.(2.gt.mlevel)) then
                  do m=1,meqn
                     do i=1,mxre(level,j)
                        qu(iitemp11+i-1,m)=qh1(iitemp11+i-1,m,1)
                     enddo
                  enddo
                  endif
                enddo    !# loop j
              endif      !!!!!!!!

                if((nb1.eq.2).and.(2.le.mlevel)) then
                  iup=3
                  do j=1,inumre(level)
                    do jj=1,numjj(j)
                        imxre=iimx(jj,j)/2
                        iitemp11=iidd(jj,j)
                        iitemp22=(iitemp11+1)/2

c        set initial
                        do m=1,meqn
```

```
                    do i=1,imxre
                      q(i,m)=qh1(iitemp22+i-1,m,level)
                    enddo
                  enddo

c     set boundary


            if(iitemp22.eq.itemp11(level,j)) then
          do m=1,meqn
            q(0,m)=(qre(itemp22(level,j)-1,m,level-1)+
     $              qh1(itemp22(level,j)-1,m,level-1))/2.0d0
            q(-1,m)=q(0,m)
            q(imxre+1,m)=qh1(iitemp22+imxre,m,level)
            q(imxre+2,m)=qh1(iitemp22+imxre+1,m,level)
           if(level+1.le.mlevel) then
            q(imxre+1,m)=qh1(iitemp11+iimx(jj,j),m,level+1)
c     $                  qh1(iitemp11+iimx(jj,j)+1,m,level+1))/2.0d0
            q(imxre+2,m)=qh1(iitemp11+iimx(jj,j)+2,m,level+1)
c     $                  qh1(iitemp11+iimx(jj,j)+3,m,level+1))/2.0d0

           endif
          enddo
          if(iitemp22+imxre-1.eq.itemp11(level,j)+mxre(level,j)-1)
     $       then
              do m=1,meqn
                  q(imxre+1,m)=(qre(itemp22(level,j)+
     $                        (mxre(level,j)/2),m,level-1)+
     $                         qh1(itemp22(level,j)+(mxre(level,j)/2),
     $                         m,level-1))/2.0d0
                  q(imxre+2,m)= q(imxre+1,m)
               enddo
          endif

          else
          do m=1,meqn
            q(-1,m)=qh1(iitemp22-2,m,level)
            q(0,m)=qh1(iitemp22-1,m,level)
            q(imxre+1,m)=qh1(iitemp22+imxre,m,level)
            q(imxre+2,m)=qh1(iitemp22+imxre+1,m,level)
             if(level+1.le.mlevel) then
              q(-1,m)=qh1(iitemp11-4,m,level+1)
c     $                qh1(iitemp11-3,m,level+1))/2.0d0
              q(0,m) =qh1(iitemp11-2,m,level+1)
c     $                qh1(iitemp11-1,m,level+1))/2.0d0
              q(imxre+1,m)=qh1(iitemp11+iimx(jj,j),m,level+1)
c     $                     qh1(iitemp11+iimx(jj,j)+1,m,level+1))/2.0d0
              q(imxre+2,m)=qh1(iitemp11+iimx(jj,j)+2,m,level+1)
c     $                     qh1(iitemp11+iimx(jj,j)+3,m,level+1))/2.0d0
             endif
          enddo
           if(iitemp22+imxre-1.eq.itemp11(level,j)+mxre(level,j)-1)
     $       then
                do m=1,meqn
                  q(imxre+1,m)=(qre(itemp22(level,j)+
     $                        (mxre(level,j)/2),m,level-1)+
     $                         qh1(itemp22(level,j)+(mxre(level,j)/2),
     $                         m,level-1))/2.0d0
                  q(imxre+2,m)= q(imxre+1,m)
             enddo
            endif
           endif
           if(iitemp11.eq.1) then
              do m=1,meqn
```

```
                do ibc=1,mbc
                   q(1-ibc,m) = q(1,m)
                enddo
             enddo
          endif
          if(iitemp22+imxre-1.eq.mxl(level)) then
             do m=1,meqn
                  do ibc=1,mbc
                     q(imxre+ibc,m) = q(imxre,m)
                  enddo
             enddo
          endif

                    call calculate(maxmx,mbc,meqn,mwaves,q,mthlim,
     $                   mthbc,method,efix,limit,imxre,t0,tfinal,
     $                   mt,grav,dxl(level),dtl(level))
             do m=1,meqn
                do i=1,imxre
                 qh1(iitemp22+i-1,m,level)=q(i,m)
                 qu(iitemp11+2*i-2,m)=q(i,m) !#update
                 qu(iitemp11+2*i-1,m)=q(i,m) !#update
                enddo
             enddo
            enddo                    !# loop j
           enddo                    !# loop jj
         endif
         level=level+1

c-------------------------------------------------------------------------------

c     level=2

         if(level.le.mlevel) then
            if(icheck3.eq.1) icheck3=6
                    do nb2=1,refineratio
                      do j= 1,inumre(level)
                         imxre=mxre(level,j)
                         iitemp11=itemp11(level,j)
                         iitemp22=itemp22(level,j)
              if(ibound(j).eq.2) ibound(j)=3
              if(iibound(j).eq.2) iibound(j)=3
                    call grid(level,meqn,imxre,q,qre,iitemp11,
     $                   iicheck,qh1,ixleft,xlower,mbc,maxmx,ixright,xupper,
     $                   maxregrid,maxlevel,icheck1,icheck2,iitemp22,icheck3,
     $                   mwaves,mthbc,method,efix,limit,t0,tfinal,mt,grav,
     $                   dxl(level),dtl(level),j,qre1,ibound,iibound,qb,n,nb1,
     $                   mthlim,mxl(level))
              if(ibound(j).eq.3)  ibound(j)=2
              if(iibound(j).eq.3)  iibound(j)=2
            enddo

c-------------------------------------------------------------------
              iicheck=4          !# set initial for second iteration
              icheck1=5      !# set for boundary

              if(nb1.eq.2) then
                icheck3=1
                do m=1,meqn
                      do i=1,mxl(level-1)
                        qre(i,m,level-1)= qre1(i,m,level-1)      !# keep old qh1
                      enddo
                 enddo
              endif
```

```
          enddo                    !!!!!!!!!!!!!2
          icheck=0

c-----------------------------------------------------------------------
c     update level 1

          icheck3=1
          do m=1,meqn
             do i=1,mx1(level-1)
                qre1(i,m,level-1)= qh1(i,m,level-1)  !# keep old qh1
             enddo
          enddo
          do j=1,inumre(level)
              call update(meqn,mxre(level,j),itemp22(level,j),
     $            level,qh1,itemp11(level,j),mbc,maxmx,maxlevel,qu,
     $            mmlevel,mlevel,j,iup,id)
             enddo

c     level=2
c       count points that weren't flaged for level 1
          if(nb1.eq.1) then
             do j=1,inumre(level-1)
                jj=0
                imx=0
                i=1
                do while (i.le.mxre(level-1,j)*2)
                   if(id((itemp11(level-1,j)*2-1)+(i-1)).ne.3) then
                       imx=imx+1
                   iid(imx)=(itemp11(level-1,j)*2-1)+(i-1)
                   if(id((itemp11(level-1,j)*2-1)+i).ne.3) then
                  if(i.eq.mxre(level-1,j)*2) go to 17
                     goto 12
                   else
  17                 continue
                     jj=jj+1
                     iimx(jj,j)=imx
                     iidd(jj,j)=iid(1)
                     numjj(j)=jj
                     imx=0
                   endif
                endif
  12            continue
                i=i+1
                enddo
             enddo
          endif
        endif                              !!!!!!!!!!!!!!!2
         level=level-1
       enddo                       !!!!!!!!!!!!1
       iup=0
       endif                      !!!!!!!!!!!!1

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     before begin new time

       do j=1,maxlevel
         do m=1,meqn
           do i=1,mxl(j)
              qre(i,m,j)=0.0d0
              qre1(i,m,j)=0.0d0
              qb(i,m)=0.0d0
              iqerr(i)=0
              ii(i)=0
              id(i)=0
```

```
            iid(i)=0
          enddo
        enddo
      enddo
c     set initial for new time

        do m=1,meqn
          do i=1,mxl(0)
             qre(i,m,0)=qh1(i,m,0)
          enddo
       enddo


c     plot graph

      do j=0,mlevel
           do i=1,mxl(j)+1
                   x(i,j)=xlower+(i-1)*dxl(j)
           enddo
          enddo
        if(0.le.mlevel) then
          do m=1,meqn
           do i=1,mxl(0)
            xx(i)=x(i,0)
            qu2(i,m)=qu(i,m)
           enddo
          enddo
         endif

        if(1.le.mlevel) then
         do m=1,meqn
          do i=1,mxl(0)
           xx(i)=x(i,0)
           qu2(i,m)=qu(2*i-1,m)
          enddo
         enddo
        endif

        if(2.le.mlevel) then
         do m=1,meqn
           do i=1,mxl(0)
            xx(i)=x(i,0)
            qu2(i,m)=qu((2*i-1)*2-1,m)
           enddo
         enddo
        endif

      if(1.le.mlevel) then
        do j=1,inumre(1)
         it1(j)=(itemp11(1,j)+1)/2
        enddo
      it11=it1(1)
      it12=it11
      jj=0
      do j=1,inumre(1)
         jj=jj+1
         i2=0
  64       continue
           if(xx(it11).lt.x(itemp11(1,j)+mxre(1,j),1)) then
            it11=it11+1
            i2=i2+1
            xx(it11)=x(itemp11(1,j)+i2,1)
            if(2.le.mlevel) then
             do m=1,meqn
```

```
            qu2(it11-1,m)=qu((itemp11(1,j)+i2-1)*2-1,m)
           enddo
          else
           do m=1,meqn
             qu2(it11-1,m)=qu(itemp11(1,j)+i2-1,m)
           enddo
          endif
          if(it11/2.eq.(it11+1)/2) then
            do m=1,meqn
              qre(it12,m,0)=(qh1((itemp11(1,j)+i2-1),m,1)+
     $        qh1((itemp11(1,j)+i2),m,1))/2.0d0
            enddo
            it12=it12+1
          endif
          if(2.le.mlevel) then
          if(jj.le.inumre(2)) then
           if(xx(it11).eq.x(itemp11(2,jj),2)) then
            do i1=1,mxre(2,jj)
               xx(it11+i1)=x(itemp11(2,jj)+i1,2)
              do m=1,meqn
               qu2(it11+i1-1,m)=qu(itemp11(2,jj)+i1-1,m)
              enddo
            ia=it11+i1
            enddo
        do i1=1,mxre(2,jj)/4
           do m=1,meqn
              qre(it12,m,0)=(qh1(itemp11(2,jj)+(i1-1)*4,m,2)
     $             +qh1(itemp11(2,jj)+(i1-1)*4+1,m,2)
     $             +qh1(itemp11(2,jj)+(i1-1)*4+2,m,2)
     $             +qh1(itemp11(2,jj)+(i1-1)*4+3,m,2))/4.0d0
           enddo
           it12=it12+1
         enddo
         it11=ia
         i2=i2+mxre(2,jj)/2
         jj=jj+1
           endif
           endif
           endif
           goto 64
        endif
        jj=jj-1
        i2=0
65        continue
        if(j+1.le.inumre(1)) then
         if(xx(it11).lt.x(itemp11(1,j+1),1)) then
          it11=it11+1
          i2=i2+1
          xx(it11)=x(it1(j)+mxre(1,j)/2+i2,0)
          iaa=it1(j)+mxre(1,j)/2+i2-1
           if(2.le.mlevel) then
            do m=1,meqn
              qu2(it11-1,m)=qu((iaa*2-1)*2-1,m)
              qre(it12,m,0)=qh1(iaa,m,0)
            enddo
           else
             do m=1,meqn
                qu2(it11-1,m)=qu(iaa*2-1,m)
                qre(it12,m,0)=qh1(iaa,m,0)
             enddo
           endif
           it12=it12+1
          goto 65
         endif
```

```
         else
  66     continue
          if(xx(it11).lt.xupper) then
           it11=it11+1
           i2=i2+1
           xx(it11)=x(it1(j)+mxre(1,j)/2+i2,0)
            iaa=it1(j)+mxre(1,j)/2+i2-1
            if(2.le.mlevel) then
             do m=1,meqn
               qu2(it11-1,m)=qu((iaa*2-1)*2-1,m)
               qre(it12,m,0)=qh1(iaa,m,0)
             enddo
            else
             do m=1,meqn
               qu2(it11-1,m)=qu(iaa*2-1,m)
               qre(it12,m,0)=qh1(iaa,m,0)
             enddo
            endif
            it12=it12+1
           goto 66
          endif
         endif
        enddo
       endif
       if(mlevel.eq.0) it11=mxl(0)+1
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

c           fname1 = 'fort.txxxx'
c           nstp = n
c           do 55 ipos = 10, 7, -1
c           idigit = mod(nstp,10)
c            fname1(ipos:ipos) = char(ichar('0') + idigit)
c           nstp = nstp / 10
c 55        continue
c         open(unit=5,file=fname1,status='unknown',form='formatted')
cc    write(*,*) ' value,h= , hu= '
c       do i=1,it11-1
cc        write(1,18) dt1,i, q(i,1),q(i,2)
cc  18         format(f5.1,5x,i5, 5x,f10.6,5x,f10.6)
c          write(5,18) xx(i)
c   18       format(f15.8)
c       enddo
c        close(5)

c-----------------------------------------------------------------------
c           fname1 = 'fort.qxxxx'
c           nstp = n
c             do 57 ipos = 10, 7, -1
c            idigit = mod(nstp,10)
c             fname1(ipos:ipos) = char(ichar('0') + idigit)
c            nstp = nstp / 10
c 57        continue
c            open(unit=6,file=fname1,status='unknown',form='formatted')
cc    write(*,*) ' value,h= , hu= '
c         do i=1,it11-1
cc        write(1,18) dt1,i, q(i,1),q(i,2)
cc  18         format(f5.1,5x,i5, 5x,f10.6,5x,f10.6)
c            write(6,19) qu2(i,1)
c   19       format(f15.8)
c        enddo
c           close(6)

c-----------------------------------------------------------------------
c           fname1 = 'fort.uxxxx'
```

```
c          nstp = n
c          do 59 ipos = 10, 7, -1
c          idigit = mod(nstp,10)
c            fname1(ipos:ipos) = char(ichar('0') + idigit)
c           nstp = nstp / 10
c 59        continue
c           open(unit=7,file=fname1,status='unknown',form='formatted')
cc    write(*,*) ' value,h= , hu= '

c          if((n.eq.50).or.(n.eq.100).or.(n.eq.200)) then
c            do i=1,mxl(0)
c            write(*,19),i,x(i,1),qre(i,1,0),qh1(i,1,0),mlevel,n
c  19           format(i5,2x,f10.6,5x,f10.6,5x,f10.6,2x,i5,2x,i5)
c         enddo
c         read(*,*) kkk

         do i=1,it11-1
cc        write(1,18) dt1,i, q(i,1),q(i,2)
cc  18        format(f5.1,5x,i5, 5x,f10.6,5x,f10.6)
          write(*,81) i,xx(i),qu2(i,1),qre(i,1,0),n
   81      format(i5,f15.8,f15.8,f15.8,
     $            i5)
        enddo
        read(*,*) kkk
c       endif
c         close(7)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

      do j=1,maxlevel
         do m=1,meqn
          do i=1,mxl(j)
                  qhl(i,m,j)=0.0d0
          enddo
         enddo
      enddo

      do m=1,meqn
         do i=1,mxl(maxlevel)
            xx(i)=0.0d0
            qu(i,m)=0.0d0
            qu2(i,m)=0.0d0
            qb(i,m)=0.0d0
         enddo
      enddo


      do j=1,maxlevel
         inumre(j)=0
         num2(j)=0
         itl(j)=0
        do i=1,maxregrid
         xleft(i)=0.0d0
         xright(i)=0.0d0
         xkeep(i)=0.0d0
         numjj(i)=0
         iibound(i)=0
         ibound(i)=0
         mxre(j,i)=0
         itemp11(j,i)=0
         itemp22(j,i)=0
         ixleft(j,i)=-5
         ixright(j,i)=-5
         iidd(j,j)=0
         iimx(j,j)=0
```

```
        enddo
      enddo


      mx=mx1
      dx=dx1
      dt=dt1
      nlevel=0
      level=0
      itemp=1
      nn=0
      icheck=0
      iicheck=0
      icheck1=0
      icheck2=0
      icheck3=0
      iup=0
      mlevel=0

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      enddo       !# loop do n=1,mt
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

        stop
        end

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

VITA

Miss Kanoknuch Chamsri was born on December 5, 1980 in Ratchaburi, Thailand. She graduated with a Bachelor Degree of Science in Mathematics with second class honours from Chulalongkorn University in 2001. For her Master degree, he has studied Mathematics at the Department of Mathematics, Faculty of Science, Chulalongkorn University.