A comparative study on the impact of different fluxes in a discontinuous Galerkin scheme for the 2D shallow water equations

by

Faraniaina Rasolofoson

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Applied Mathematics at Stellenbosch University

Department of Mathematical Sciences,
Faculty of Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Supervisor: Dr. Sehun Chun

December 2013

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signature: .....
F. Rasolofoson

Date: 2013/12/20 .....

Abstract

A comparative study on the impact of different fluxes in a discontinuous Galerkin scheme for the 2D shallow water equations

F. Rasolofoson

Department of Mathematical Sciences,
Faculty of Sciences,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MSc

December 2013

Shallow water equations (SWEs) are a set of hyperbolic partial differential equations that describe the flow below a pressure surface in a fluid. They are widely applicable in the domain of fluid dynamics. To meet the needs of engineers working on the area of fluid dynamics, a method known as spectral/hp element method has been developed which is a scheme that can be used with complicated geometry. The use of discontinuous Galerkin (DG) discretisation permits discontinuity of the numerical solution to exist at inter-element surfaces. In the DG method, the solution within each element is not reconstructed by looking to neighbouring elements, thus the transfer information between elements will be ensured through the numerical fluxes. As a consequence, the accuracy of the method depends largely on the definition of the numerical fluxes. There are many different type of numerical fluxes computed from Riemann solvers. Four of them will be applied here respectively for comparison through a 2D Rossby wave test case.

Uittreksel

Vergelykende studies oor die impak van die verskillende strome in diskontinue Galerkin metodes vir 2D vlak water vergelykings

("A comparative study on the impact of different fluxes in a discontinuous Galerkin scheme for the 2D shallow water equations")

F. Rasolofoson

Departement van Wiskunde,
Fakulteit van Wetenskap,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MSc

Desember 2013

Vlakwatervergelykings (SWEs) is 'n stel hiperboliese parsiële differensiaalvergelykings wat die vloei onder 'n oppervlak wat druk op 'n vloeistof uitoefen beskryf. Hulle het wye toepassing op die gebied van vloeidinamika. Om aan die behoeftes van ingenieurs wat werk op die gebied van vloeidinamika te voldoen is 'n metode bekend as die spektraal /*hp* element metode ontwikkel. Hierdie metode kan gebruik word selfs wanneer die probleem ingewikkelde grenskondisies het. Die Diskontinue Galerkin (DG) diskretisering wat gebruik word laat diskontinuïteit van die numeriese oplossing toe om te bestaan by tussenelement oppervlakke. In die DG metode word die oplossing binne elke element nie gerekonstrueer deur te kyk na die naburige elemente nie. Dus word die oordrag van informasie tussen elemente verseker deur die numeriese stroomterme. Die akkuraatheid van hierdie metode hang dus grootliks af van die definisie van die numeriese stroomterme. Daar is baie verskillende tipe numeriese strometerme wat bereken kan word uit Riemann oplossers. Vier van hulle sal hier gebruik en vergelyk word op 'n 2D Rossby golf toets geval.

Acknowledgements

I would like to thank my supervisor, Dr. Sehun Chun, for his guidance, support and encouragement during the course of this research. Many thanks are to the AIMS family for giving me the opportunity to study and do research as well as experience through numerous conferences. Finally, I thank my parents and friends for their continuous support during this exceedingly long project.

Dedications

Ek wil graag my studieleier, Dr Sehun Chun, bedank vir sy leiding, ondersteuning en aanmoediging deur die loop van hierdie navorsing. Baie dankie ook aan my AIMS familie vir die geleentheid wat hulle my gebied het om te studeer en navorsing te doen asook om ervaring by talle konferensies te kry. Ten slotte, bedank ek my ouers en vriende vir hul volgehoue ondersteuning gedurende hierdie uiters lang projek.

Contents

List of Figures

List of Tables

Nomenclature

Space

Ω	Spatial domain
$S = \partial\Omega$	Boundary of the domain Ω
$E(\Omega), H^1$	Energy space
χ	Trial space
Ω^e	Element mesh e
χ^e	Mapping
Ω_{st}	Standard domain
$\mathcal{P}_p(\Omega)$	Space of all polynomials of degree at most p defined on Ω

Constants

g =	$9.81\,\mathrm{m/s^2}$

Variables

ρ	Density of the fluid	$[\mathrm{kg/m^3}]$
t	Time variable	$[\mathrm{s}]$
f	Coriolis parameter	$[\mathrm{rad/s}]$
P	Pressure	$[\mathrm{Pa}]$
η	Free surface elevation	$[\mathrm{m}]$
d	Still water depth	$[\mathrm{m}]$
H	Total water depth	$[\mathrm{m}]$
$\bar{u}$	Depth-averaged for velocity u	$[\mathrm{m/s}]$
h	Element size	[ ]
p, P	Degree of the polynomial expansion	[ ]
N_{dof}	Number of degree of freedom	[ ]
$\hat{u}$	Coefficient function	[ ]
Φ_i	Trial function	[ ]
R	Residual	[ ]
v_j	Weight function	[ ]

N_{el}	Number of element	[ ]
ξ	Local coordinate	[ ]
$L_p(x)$	Legendre polynomial	[ ]
$H_p(x)$	Lagrange polynomial	[ ]

Vectors, Matrices and Tensors

$\mathbf{U}$ or $\mathbf{v} = (u, v, w)^T$	Velocity vector of fluid
$\mathbf{h} = (u, v)^T$	Velocity vector in $\mathbb{R}^2$
$\mathbf{n}$	Outward normal vector
$\mathbf{b}$	Body force density
$\mathbf{T}$	Cauchy stress tensor matrix
$\mathbf{g}$	Gravity force
$\mathbf{f}$	Coriolis force
$\mathbf{\Omega}$	Angular velocity vector
$\hat{\mathbf{z}}$	Vertical unit vector
$\mathbf{I}$	Identity matrix
$\bar{\mathbf{T}}$	Matrix of stress terms
$\mathbf{0}$	Zero matrix
$\mathbf{F}$	Flux vector
$\hat{\mathbf{F}}$	Numerical flux vector
$\mathbf{S}$	Source vector

Subscripts and superscripts

$\delta = \delta(h, p)$	Approximation
L	Left state
R	Right state

Operators

$\mathbb{L}$	Differential operator
$\nabla\cdot$	Divergence operator
∇	Gradient operator

Chapter 1

Introduction

1.1 General introduction

The humans are always faced with natural challenges. The impacts in their life are good or bad, and they need to understand the science behind them for the seek of issues and/or profits. Scientists and engineers have provided us with ways of understand the workings of these physical facts.

For instance, mathematical modelling involves in creating a model that described the problem. The problem becomes represented by a differential equation, and thus becomes a mathematical problem.

Solving differential equations is another challenge. Most of them are difficult to solve analytically, and some do not have solution at all. Thus one uses numerical methods for the resolution. The numerical methods are used to provide approximate solutions to those differential equations, and they have to assure the rate of convergence, the accuracy, and the completeness of the solution. The adapted method therefore can be different from one to another problem depending on the features and conditions required. Considering a problem consisting on solving differential equations, numerically, one uses a finite number of values of a given function. So the algorithm is designed to compute the solution of the given differential equation from these finite number of values. Since this is just an approximation, no matter how accurate the method is, it will never provide the exact solution. An error is always generated from the computation, this is called the truncation error. In any given method, the maximum truncation error is proportional to the constant time step Δt to the power p, where p is known as the order of the method. A high-order method, that is a method with great value of p, is usually regarded more efficient and accurate than a low-order method for the same computation resources. But sometimes, the best method is of lower order, depending on the error tolerance of the problem. It may run faster and give the same accuracy as the higher order one.

In this work, the shallow water equations (SWEs) are going to be solved

numerically using the spectral/*hp* element discontinuous Galerkin (DG) methods, and the results for different types of fluxes are going to be compared. The SWEs are a set of nonlinear hyperbolic partial differential equations (PDEs) describing the flow below a pressure surface in a fluid. The SWEs can be derived from the Euler equations for the motion of fluid. The general characteristic of the SWEs is that the vertical dimension is much smaller than the horizontal scale; thus, from the conservation of mass, we can average over the depth to delete the vertical dimension. In the domain of fluid dynamics, we frequently come across this typical property; thus the SWEs are widely applicable. The shallow fluid flows are commonly observed in the real world such as the atmosphere on the earth and the lava drifting from the peak of a mountain, flood waves in rivers and surges, tide and are also used in different contexts in meteorology.

1.2 Previous works and results concerning the shallow water equations

This section summarises some of previous works related to the SWEs. Many different methods, different domains of computation, and different test cases of study are presented.

Paper 1 (Schwanenberg *et al.* (2000))**.** This paper presents a numerical solution for the SWEs based on Runge-Kutta discontinuous Galerkin method (RKDGM). Mathematical expressions of the SWEs and adaptation of the RKDGM to the SWEs are presented for one-dimensional transient flow. The HLL numerical flux is chosen to approximate the fluxes along the element boundaries. A comparison between analytical and computational results appears is made and it was satisfactory so that the SWEs can be applied on dam-break type problem. Applications to one and two-dimensional hydraulic test cases are given, which are computations of one-dimensional dam-break, a computational of a circular dam-break, and an experimental and a computational investigation on a two-dimensional break-type flow. The analytical solutions of each test case are used to prove the accuracy of the method. The paper shows that the performance of the scheme is efficient and stable.

Paper 2 (Williamson *et al.* (1992))**.** In this paper, seven test cases are presented to estimate any proposed numerical methods to solve the SWEs in spherical geometry. The choice of the SWEs is due to its challenging characteristic related to the horizontal dynamical aspects of the atmospheric modelling on the spherical earth. The paper aims to evaluate a chosen numerical method for the climate modelling and to identify the potential drawbacks. The test cases are used to measure the performance of a proposed scheme before it is applied to a full baroclinic atmospheric problem. The complexity of

each test cases is different depending on the considered parameters and the characteristics that are taken into account, and also from some additional features and properties influenced by the geographical constraints or any exterior physical forces. Each case is presented with a numerical value of each parameter, analytical expressions of some functions presented in the model, list of all the must verified conditions. But most importantly, most cases are equipped with the definitions of the errors measures, instructions about the final time measurements and the types of plots to verify the compatibility and the efficiency. In the beginning, detailed mathematical expressions of the SWEs are produced in the spherical coordinates. The first case regards to the advection of Cosine Bell over the pole; the second case concerns the global steady state nonlinear zonal geostrophic flow; the third case accounts for the global steady state nonlinear zonal geostrophic flow with compact support; the fourth case regards about the forced nonlinear system with translating flow; the fifth case accounts for the zonal flow over an isolated mountain; the sixth case concerns about the Rossby-Haurwitz wave, and the last case regards to the analysed 500mb height and wind field initial conditions. When using these test cases, one should always mention and report the computational tools used such as the type of machine, compiler and precision. It is also mentioned that any measurement should not be made before a five-day simulation. All these test cases and instructions are made to offer the users a wide overview and to test the performance of their proposed schemes.

Paper 3 (Nair *et al.* (2005*a*))**.** This paper develops the full shallow water model defined on a sphere, and presented in curvilinear coordinates, using DG method. The latitude-longitude grid and the geodesic grid, which are commonly used in spherical geometry are totally forgotten in this work. Instead, a new method called the cubed sphere is adopted. It consists of projecting an inscribed cube onto a sphere. The mapping divides the spherical surface area into six identical sub-domains and the resulting grid does not contain any singularity. This is the main advantage of this spatial discretisation method. In addition, the modal basis set composed with Legendre polynomials is employed. The mapping of each face of the inscribed cube to the sphere is then assured by a central equiangular projection. The discontinuity along element edges and the interaction of adjacent elements is ensured by Lax-Friedrichs numerical flux. Applications of the scheme to test case two, five and six of the standard test cases of Williamson, presented in Paper 2, and to the polar rotating low-high are made to provide numerical results. It has been shown that numerical solutions are very accurate, and fake oscillations are not found in the test case five and in the flow over a mountain test case. The second test case exhibits an exponential convergence. The conservation of global invariant is proved to be better than in the finite volume (FV) method. DG solutions for the shallow water test cases are much better than the solutions of a spectral model for a given spatial resolution.

Paper 4 (Eskilsson *et al.* (2009))**.** This article presents a parallel program for shallow water solver following the cactus framework. The spatial discretisation adopts the spectral/*hp* element library of Nektar++ and an unstructured high-order DG method. The fundamental objective of this work is a scalable, parallel, non-hydrostatic wave solver, based on multi-layered Boussinesq-type equations including time-dependent bathymetry and sediment transport. The spectral/*hp* DG method and its combination to the cactus framework are reviewed. In order to develop an open source software, the implementation is made separately. The Coastal DG wave model is established supported by the spectral/*hp* element library Nektar++, which permits the coastal engineers to focus on developing a coastal code using Nektar++. Meanwhile, to achieve parallelism, an unstructured mesh driver was developed for the computational cactus framework. This allows the computational scientists to focus on parallelism, scalability and performance of the unstructured mesh driver. Applications to a linear standing wave and to a wave interacting with cylinders are exhibited. It has been shown that the method displays an exponential convergence. The weak and the strong scaling are largely independent of the spatial order of the scheme.

Paper 5 (Aizinger and Dawson (2002))**.** In this paper, the DG FEM for approximating the SWEs in hydrodynamics and contaminants transport is presented, followed by the formulation of the weak form of the shallow water problem. This approach generalises and extends the Gudonov method. It permits the variation of the polynomial order approximation by its local property. It allows the incorporation of diffusive terms and the use of a non-conforming grid. The method is locally conservative and integrates upwinded numerical fluxes for modelling problem with high flow gradient. The scheme is also based on the local discontinuous Galerkin (LDG) method. Several test cases are used to obtain numerical results, such as supercritical flow through a constricted channel, tidal flow near the Bahamas islands, contaminant transport in Galveston Bay, and river inflow into the Gulf of Mexico. It has been proved that the scheme supports both lower and higher-order approximations and it locally conserves both mass and momentum for all cases. An approximation of sharp fronts created by high speed flows along with flows produced by tidal boundaries conditions can be accomplished. A piecewise constant approximations can capture the qualitative dynamics of the flow in most of the cases, and a piecewise linear approximations do a better job of resolving a sharp variation in the solution.

Paper 6 (Eskilsson and Sherwin (2000))**.** This paper presents a spectral/*hp* element DG method for simulating the two-dimensional (2D) SWEs on unstructured triangular meshes. The spatial discretisation is assured by the use of modal expansion basis of arbitrary order, and a third Runge-Kutta scheme is applied for time integration. The application of the DG method allows solutions to be discontinuous at elemental boundaries, and the same techniques

as in FV are employed to couple elements together using the HLLC Riemann solver. The scheme is tested on four example cases, such as the simple case of a linear standing wave in a rectangular frictionless basin, the equatorial Kelvin and Rossby waves, the dam-break problem, and the Harbour problem. It has been shown that the exponential convergence is reached. The model is also computationally efficient due to this exponential convergence and the use of orthogonal expansion basis which produced the diagonal mass matrix, and this efficiency increases with the order of the model and the time integration. A formation of Gibbs oscillations is exhibited in the dam-break test case, which is an inevitable fact caused by the use of an high-order model without limitation. But the utilisation of the slope limiter removes the oscillations and deterioration of accuracy is avoided by combining it with h-refinement. The Harbour problem is used to show the geometrical flexibility of the model.

Paper 7 (Eskilsson (2011))**.** This paper presents an hp-adaptive DG method for the 2D smooth SWEs, that is without shock-capturing. The discretisation in space is assured by the use of orthogonal modal basis of arbitrary polynomial order p defined on unstructured triangular non-conforming meshes. A third Runge-Kutta method is applied for time-stepping. The use of the spectral/hp element method allows variation on the mesh and polynomial order during the simulation. This variation yields geometric and functional incompatibilities which are resolved by applying adaptivity. h-, p- and hp-adaptivity are implemented. The approaches are validated by using five computational examples, in real life geometry, which are two linear cases with analytical solutions: the Stommel gyre and the equatorial Kelvin waves; and three nonlinear cases: the nonlinear Stommel gyre and equatorial Rossby modon, and the harbour disturbance. The adaptivity is driven by an error indicator based on the difference between the solutions of approximation order p and $p-1$. The $p-1$ solution is readily available and the computational cost for the indicator is low because of the use of the PKD basis. The indicator worked satisfactorily for a higher order polynomial but was generally found to diminish the error for the linear expansions. All the test cases show their benefit on the choice of higher order schemes for smooth problem, and the p-adaptivity procedure generated the fastest computations and the least N_{dof}. It is simpler to implement the p-adaptivity than the h-adaptivity, and also conservation is guaranteed. For the Stommel gyre problems, the N_{dof} for the higher-order adaptive schemes was of order magnitude less than for h-adaptivity scheme using linear expansions. For the Kelvin waves and Rossby modon, the low-order h-adaptivity method is the least efficient adaptive approach for the test cases investigated.

1.3 Definitions used in this paper

1. A *fluid* is a substance that can flow and has no precise shape but follows the form of its container, such as liquids or gas.

2. *Flux* is the amount of information passing through a given surface.

3. *Numerical flux* is the flux at the boundary following the normal direction.

4. *Navier-Stokes equations* are mathematical equations that describe the motion of fluid. The analytical expression is given in Chapter 2 (Eq. 2.1.2).

1.4 Organisation of the thesis

The remainder of this thesis is organised as follows: Chapter 2 gives a general overview of the SWEs, shows how they were derived and introduces the mathematical expressions of the governing equations that are going to be studied in the whole thesis. The method that is going to be used in the study will be discussed in more details in Chapter 3, followed by its theoretical application to the SWEs. Different types of numerical fluxes are compared in this study, their definitions and derivations, are discussed in Chapter 4. An illustration and application of the method using a two dimensional test case are presented in Chapter 5 followed by all the numerical results. The conclusion and the summary are stated in Chapter 6.

Chapter 2

Shallow Water Equations

This chapter aims to give a general overview of the shallow water equations (SWEs). Section 2.1 exhibits how the SWEs are derived from the mother equation of the fluid, known as the Navier-Stokes equations. Detailed expression of the SWEs is presented in Section 2.2.

2.1 Derivation of the Shallow Water Equations

The SWEs can be derived from the Navier-Stokes equations, which describe the motion of fluid. The Navier-Stokes equations can be derived from combination of Cauchy's equation of motion for a deformable body and the equations for an incompressible Newtonian fluid. This section will show how the SWEs are derived from the conservation laws step by step, following the ideas presented in Dawson and Mirabito (2009); Randall (2006). First, the Navier-Stokes equations will be derived, then the SWEs will be derived by applying all necessary boundary conditions and assumptions.

The law of conservation of mass states that, for any system closed to all transfers of matter and energy, the mass of the system must remain constant over time. Considering a fluid flow in a non-deformable control volume Ω bounded by the control surface $S = \partial\Omega$ (Karniadakis and Sherwin (2005)), we have the following mass integral form for mass conservation equation:

$$\frac{d}{dt}\int_{\Omega} \rho \, \mathrm{d}\Omega = -\int_{\partial\Omega} (\rho \mathbf{v}) \cdot \mathbf{n} \, \mathrm{d}S \,,$$

where ρ is the density of the fluid, $\mathbf{v} = (u, v, w)^T$ is the fluid velocity and $\mathbf{n}$ is the outward normal vector on $\partial\Omega$.

Applying Gauss's theorem (see Appendix B) to the mass conservation equation gives:

$$\frac{d}{dt}\int_{\Omega} \rho \, \mathrm{d}\Omega = -\int_{\Omega} \nabla \cdot (\rho \mathbf{v}) \, \mathrm{d}\Omega \,.$$

Assuming that ρ is smooth, the Leibniz integral rule (see Appendix B), produces:

$$\int_\Omega \left[\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v})\right] \mathrm{d}\Omega = 0\,.$$

Since Ω is arbitrary, this leads to the following continuity equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0\,. \tag{2.1.1}$$

Newton's second law states that the rate change of the momentum of a particle is equal to the force acting on it. Considering the linear momentum balance over the non-deformable control volume Ω, we have:

$$\frac{d}{dt}\int_\Omega \rho \mathbf{v}\,\mathrm{d}\Omega = -\int_{\partial\Omega} (\rho \mathbf{v})\,\mathbf{v}\cdot\mathbf{n}\,\mathrm{d}S + \int_\Omega \rho \mathbf{b}\,\mathrm{d}\Omega + \int_{\partial\Omega} \mathbf{T}\mathbf{n}\,\mathrm{d}S\,,$$

where $\mathbf{b}$ is the body force density per unit mass acting on the fluid and $\mathbf{T}$ the Cauchy stress tensor. Applying Gauss's theorem to this equation gives:

$$\frac{d}{dt}\int_\Omega \rho \mathbf{v}\,\mathrm{d}\Omega + \int_\Omega \nabla\cdot(\rho\mathbf{v}\mathbf{v})\,\mathrm{d}\Omega - \int_\Omega \rho\mathbf{b}\,\mathrm{d}\Omega - \int_\Omega \nabla\cdot\mathbf{T}\,\mathrm{d}\Omega = 0\,.$$

Assuming that $\rho\mathbf{v}$ is smooth, the Leibniz integral rule gives:

$$\int_\Omega \left[\frac{\partial \rho\mathbf{v}}{\partial t} + \nabla\cdot(\rho\mathbf{v}\mathbf{v}) - \rho\mathbf{b} - \nabla\cdot\mathbf{T}\right] \mathrm{d}\Omega = 0\,.$$

Since Ω is arbitrary,

$$\frac{\partial \rho\mathbf{v}}{\partial t} + \nabla\cdot(\rho\mathbf{v}\mathbf{v}) - \rho\mathbf{b} - \nabla\cdot\mathbf{T} = 0\,. \tag{2.1.2}$$

The following assumptions will be made to obtain the final expression of the Navier-Stokes equations, (Dawson and Mirabito (2009)):

- The fluid is supposed to be incompressible, that is, it is unable to sustain volume change. This means that the density ρ of the fluid does not change during its motion, so it is a constant independent on time t. Thus, from equation (2.1.1), we have

  $$\nabla\cdot\mathbf{v} = 0\,.$$

- Suppose that gravity $\mathbf{g}$ and Coriolis force $\mathbf{f}$ are the only body forces acting on the fluid. All other forces that could act on the fluid body are neglected. The Coriolis force per unit mass is $\mathbf{f} = -2\boldsymbol{\Omega}\times\mathbf{v} = -f\hat{\mathbf{z}}\times\mathbf{v}$ where $\boldsymbol{\Omega}$ is the angular velocity vector directed along the axis of rotation of the rotating reference frame and f is called the Coriolis parameter. Thus we can write:

  $$\mathbf{b} = \mathbf{g} + \mathbf{f} = g\hat{\mathbf{z}} - f\hat{\mathbf{z}}\times\mathbf{v}\,,$$

  where g is the acceleration due to gravity.

- Suppose that the fluid is Newtonian, that is, the state of stress at any point is proportional to the time rate of strain at that point; the proportionality factor is the viscosity coefficient. And for Newtonian fluid we have

$$\mathbf{T} = -P\mathbf{I} + \bar{\mathbf{T}},$$

where P is the pressure, $\mathbf{I}$ the identity matrix, and $\bar{\mathbf{T}}$ is the matrix of stress terms, which defines the state of stress at a point inside a material in the deformed placement, and is given by:

$$\bar{\mathbf{T}} = \begin{pmatrix} \tau_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \tau_{yy} & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \tau_{zz} \end{pmatrix}.$$

Applying all these assumptions to equation (2.1.2), we obtain:

$$\begin{aligned} \frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) &= \rho \left(g\hat{\mathbf{z}} - f\hat{\mathbf{z}} \times \mathbf{v} \right) + \nabla \cdot \left(-P\mathbf{I} + \bar{\mathbf{T}} \right), \\ &= \rho g\hat{\mathbf{z}} - \rho f\hat{\mathbf{z}} \times \mathbf{v} - \nabla P + \nabla \cdot \bar{\mathbf{T}}. \end{aligned}$$

Thus, the final form of the Navier-Stokes equations in 3D is:

$$\begin{cases} \nabla \cdot \mathbf{v} = 0, \\ \dfrac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = \rho g\hat{\mathbf{z}} - \rho f\hat{\mathbf{z}} \times \mathbf{v} - \nabla P + \nabla \cdot \bar{\mathbf{T}}. \end{cases} \tag{2.1.3}$$

The first equation is derived from the conservation of mass and the second equation is derived from the conservation of momentum. It can be written in a three-dimensional Cartesian coordinates system as follows:

$$\begin{cases} \dfrac{\partial u}{\partial x} + \dfrac{\partial v}{\partial y} + \dfrac{\partial w}{\partial z} = 0, \\ \dfrac{\partial (\rho u)}{\partial t} + \dfrac{\partial (\rho u^2)}{\partial x} + \dfrac{\partial (\rho uv)}{\partial y} + \dfrac{\partial (\rho uw)}{\partial z} = \dfrac{\partial (\tau_{xx} - P)}{\partial x} + \dfrac{\partial \tau_{xy}}{\partial y} + \dfrac{\partial \tau_{xz}}{\partial z} + \rho f v, \\ \dfrac{\partial (\rho v)}{\partial t} + \dfrac{\partial (\rho uv)}{\partial x} + \dfrac{\partial (\rho v^2)}{\partial y} + \dfrac{\partial (\rho vw)}{\partial z} = \dfrac{\partial \tau_{xy}}{\partial x} + \dfrac{\partial (\tau_{yy} - P)}{\partial y} + \dfrac{\partial \tau_{yz}}{\partial z} - \rho f u, \\ \dfrac{\partial (\rho w)}{\partial t} + \dfrac{\partial (\rho uw)}{\partial x} + \dfrac{\partial (\rho vw)}{\partial y} + \dfrac{\partial (\rho w^2)}{\partial z} = \dfrac{\partial \tau_{xz}}{\partial x} + \dfrac{\partial \tau_{yz}}{\partial y} + \dfrac{\partial (\tau_{zz} - P)}{\partial z} - \rho g. \end{cases} \tag{2.1.4}$$

Since the general form for the Navier-Stokes equations is built, the SWEs now can be derived. The equation for all fluid is based on the same equation source, only the changes made in the initial and boundary conditions define the nature of the equation. In the case of the SWEs, the following assumptions are made (Dawson and Mirabito (2009); Randall (2006)):

- Figure (2.1) depicts the description of the domain for shallow water. Some of the primitive variables are defined as follows: suppose that

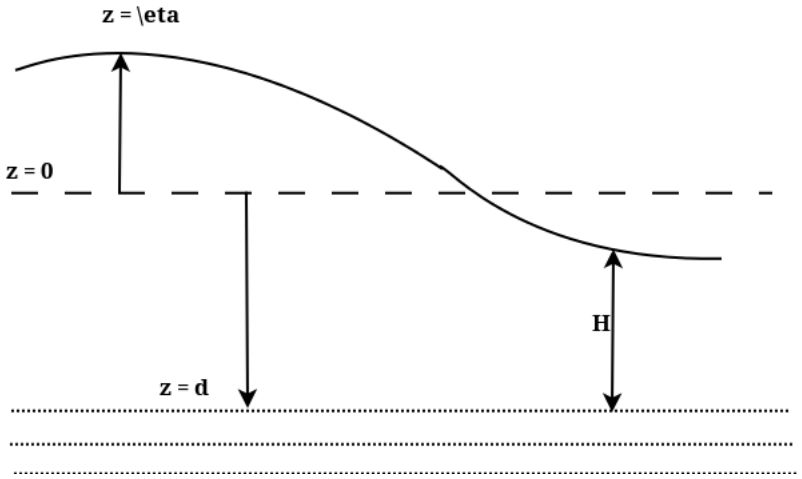

Figure 2.1: Description of the domain for shallow water, Dawson and Mirabito (2009).

$\eta = \eta(x, y, t)$ is the free surface elevation from the surface of the fluid, and $d = d(x, y)$ the still water depth which is constant and independent of time. Let $H = H(x, y, t) = \eta + d$ be the total depth of the fluid.

- The pressure P and density ρ are defined in a such a way that satisfies the hydrostatic equation:
$$\frac{dP}{dz} = -g\rho\,.$$
The hydrostatic equation states that whenever there is no vertical motion, the difference in pressure (dP) between two levels (dz) is caused by the weight of the layer of the air. This implies that, integrating downward from the surface:
$$P = \int_H^z g\rho\,\mathrm{d}z = \rho g(H - z),$$
and
$$\nabla P = \rho g\left(\frac{\partial H}{\partial x}\hat{\mathbf{x}} + \frac{\partial H}{\partial y}\hat{\mathbf{y}} - \hat{\mathbf{z}}\right).$$

- Suppose the following boundary conditions, (Dawson and Mirabito (2009)):
  - At the bottom, we have $z = -d$, $u = v = 0$ and
  $$u\frac{\partial d}{\partial x} + v\frac{\partial d}{\partial y} + w = 0\,.$$
  - At the free surface, we have $z = \eta$, $P = 0$ and
  $$\frac{\partial \eta}{\partial t} + u\frac{\partial \eta}{\partial x} + v\frac{\partial \eta}{\partial y} - w = 0\,.$$

- And last we assume that there is no shear stress contact on the fluid body, that is $\bar{\mathbf{T}} = \mathbf{0}$.

Integrating the Navier-stokes (2.1.3) over the depth on $[-d, \eta]$ leads to the SWEs. From the continuity equation in (2.1.4), we have:

$$
\begin{aligned}
0 =& \int_{-d}^{\eta} \nabla \cdot \mathbf{v} \, \mathrm{d}z \\
=& \int_{-d}^{\eta} \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} \right) \mathrm{d}z, \\
=& \int_{-d}^{\eta} \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) \mathrm{d}z + \left(w|_{z=\eta} - w|_{z=-d} \right), \\
=& \frac{\partial}{\partial x} \int_{-d}^{\eta} u \, \mathrm{d}z - \left(u|_{z=\eta} \frac{\partial \eta}{\partial x} + u|_{z=-d} \frac{\partial d}{\partial x} \right) + \frac{\partial}{\partial y} \int_{-d}^{\eta} v \, \mathrm{d}z - \left(v|_{z=\eta} \frac{\partial \eta}{\partial y} + v|_{z=-d} \frac{\partial d}{\partial y} \right) \\
& + \left(w|_{z=\eta} - w|_{z=-d} \right) \qquad \text{(from Leibniz integral rule)}, \\
=& \frac{\partial}{\partial x} \int_{-d}^{\eta} u \, \mathrm{d}z + \frac{\partial}{\partial y} \int_{-d}^{\eta} v \, \mathrm{d}z - \underbrace{\left(u \frac{\partial d}{\partial x} + v \frac{\partial d}{\partial y} + w \right)_{|_{z=-d}}}_{=0} - \underbrace{\left(u \frac{\partial \eta}{\partial x} + v \frac{\partial \eta}{\partial y} - w \right)_{|_{z=\eta}}}_{=-\frac{\partial \eta}{\partial t}}.
\end{aligned}
$$

Adopting the following notation for the depth-averaged velocities:

$$
\bar{u} = \frac{1}{H} \int_{-d}^{\eta} u \, \mathrm{d}z, \qquad \bar{v} = \frac{1}{H} \int_{-d}^{\eta} v \, \mathrm{d}z,
$$

$$
\bar{u}^2 = \frac{1}{H} \int_{-d}^{\eta} u^2 \, \mathrm{d}z, \qquad \bar{u}\bar{v} = \frac{1}{H} \int_{-d}^{\eta} uv \, \mathrm{d}z,
$$

and applying the boundary conditions, we obtain the depth-averaged continuity equation:

$$
\frac{\partial H}{\partial t} + \frac{\partial (H\bar{u})}{\partial x} + \frac{\partial (H\bar{v})}{\partial y} = 0. \tag{2.1.5}
$$

Integrating other the depth the x-momentum equation in (2.1.4), we get:

$$
\int_{-d}^{\eta} \left[\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} + \frac{\partial uw}{\partial z} \right] \mathrm{d}z = \int_{-d}^{\eta} \left[-g \frac{\partial \eta}{\partial x} + fv \right] \mathrm{d}z \, ,
$$

$$
\Rightarrow \frac{\partial}{\partial t} \int_{-d}^{\eta} u \, \mathrm{d}z + \frac{\partial}{\partial x} \int_{-d}^{\eta} u^2 \, \mathrm{d}z + \frac{\partial}{\partial y} \int_{-d}^{\eta} uv \, \mathrm{d}z = (\eta - (-d)) \left[-g \frac{\partial \eta}{\partial x} + fv \right] \, ,
$$

$$
\Rightarrow \frac{\partial H\bar{u}}{\partial t} + \frac{\partial (H\bar{u}^2)}{\partial x} + \frac{\partial (H\bar{u}\bar{v})}{\partial y} = -gH \frac{\partial \eta}{\partial x} + fHv. \tag{2.1.6}
$$

In the same way, from the y-momentum equation, we get:

$$\frac{\partial H\bar{v}}{\partial t} + \frac{\partial(H\bar{u}\bar{v})}{\partial x} + \frac{\partial(H\bar{v}^2)}{\partial y} = -gH\frac{\partial \eta}{\partial y} - fHu. \tag{2.1.7}$$

At last, the SWEs are successfully derived and the mathematical expressions are given by equations (2.1.5), (2.1.6) and (2.1.7).

2.2 The governing equations

The conservative form of the two-dimensional SWEs, defined in a certain domain Ω, can be written as:

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}) = \mathbf{S}(\mathbf{U}), \tag{2.2.1}$$

where $\mathbf{U}$ is the vector of conserved variable, $\mathbf{S}$ the source vector, and $\mathbf{F}$ the flux vector such that $\mathbf{F}(\mathbf{U}) = [\mathbf{E}(\mathbf{U}) \quad \mathbf{G}(\mathbf{U})]^T$. Each term is expressed as follows, (Eskilsson (2011); Eskilsson and Sherwin (2000)):

$$\mathbf{U} = \begin{bmatrix} H \\ uH \\ vH \end{bmatrix}, \mathbf{E} = \begin{bmatrix} uH \\ u^2H + gH^2/2 \\ uvH \end{bmatrix}, \mathbf{G} = \begin{bmatrix} vH \\ uvH \\ v^2H + gH^2/2 \end{bmatrix}, \mathbf{S} = \begin{bmatrix} 0 \\ Hfv \\ -Hfu \end{bmatrix}. \tag{2.2.2}$$

Here $H(x, y, t) = \eta(x, y, t) + d(x, y)$ is the total water depth where η is the free surface elevation and d the still water depth; $u(x, y, t)$ and $v(x, y, t)$ are the velocities in x- and y-direction respectively, g the acceleration due to gravity and f the Coriolis term.

The linearised version of the equation is valid for a constant depth. In that case, the equations are solved only in $z \in [0, -d]$, even though we do actually solve for the free surface elevation variable. Thus the expression of each terms in the linearised SWEs, in two dimensional plane, is given as follow:

$$\mathbf{U} = \begin{bmatrix} \eta \\ u \\ v \end{bmatrix}, \qquad \mathbf{F}(\mathbf{U}) = \begin{bmatrix} du & dv \\ g\eta & 0 \\ 0 & g\eta \end{bmatrix}, \text{ and } \qquad \mathbf{S}(\mathbf{U}) = \begin{bmatrix} 0 \\ fv \\ -fu \end{bmatrix}. \tag{2.2.3}$$

2.2.1 Vectorial notation of each expressions of the SWEs in two dimensional

With a parametric two dimensional space $\mathbf{x} = (x, y) \in \mathbb{R}^2$, the main equation is stated as:

$$\frac{\partial \mathbf{U}(\mathbf{x})}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{U}(\mathbf{x})) = \mathbf{S}(\mathbf{U}(\mathbf{x})), \quad \mathbf{x}, \mathbf{U} \in \mathbb{R}^2. \tag{2.2.1}$$

For the linear case, we have:

$$\begin{cases} \dfrac{\partial \eta(\mathbf{x})}{\partial t} + \dfrac{\partial}{\partial x}(du(\mathbf{x})) + \dfrac{\partial}{\partial y}(dv(\mathbf{x})) = 0, & (2.2.4) \\ \dfrac{\partial u(\mathbf{x})}{\partial t} + \dfrac{\partial}{\partial x}(g\eta(\mathbf{x})) = fv(\mathbf{x}), & (2.2.5) \\ \dfrac{\partial v(\mathbf{x})}{\partial t} + \dfrac{\partial}{\partial y}(g\eta(\mathbf{x})) = -fu(\mathbf{x}). & (2.2.6) \end{cases}$$

If the vector of primitive variable is denoted as follows:

$$\mathbf{U}(\mathbf{x}) = \begin{bmatrix} \eta(\mathbf{x}) \\ \mathbf{h}(\mathbf{x}) \end{bmatrix}, \qquad \text{where } \mathbf{h}(\mathbf{x}) = \begin{pmatrix} u(\mathbf{x}) \\ v(\mathbf{x}) \end{pmatrix},$$

then the linear expressions of the SWEs can be rewritten again in a vectorial form. Thus, for d constant, from (2.2.4), we have:

$$\frac{\partial \eta(\mathbf{x})}{\partial t} + \nabla \cdot d\mathbf{h}(\mathbf{x}) = 0.$$

And (2.2.5) and (2.2.6) become:

$$\begin{cases} \dfrac{\partial u(\mathbf{x})}{\partial t}\hat{\mathbf{x}} + \dfrac{\partial}{\partial x}(g\eta(\mathbf{x}))\hat{\mathbf{x}} &= fv(\mathbf{x})\hat{\mathbf{x}}, \\ \dfrac{\partial v(\mathbf{x})}{\partial t}\hat{\mathbf{y}} + \dfrac{\partial}{\partial y}(g\eta(\mathbf{x}))\hat{\mathbf{y}} &= -fu(\mathbf{x})\hat{\mathbf{y}}, \end{cases} \tag{2.2.7}$$

where $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are the tangent vector of the Cartesian axis x and y, respectively. Combining system (2.2.7), we get:

$$\frac{\partial \mathbf{h}(\mathbf{x})}{\partial t} + \nabla(g\eta(\mathbf{x})) = f(v(\mathbf{x})\hat{\mathbf{x}} - u(\mathbf{x})\hat{\mathbf{y}}),$$

if we note $\mathbf{h}(\mathbf{x}) \times \mathbf{z} = v(\mathbf{x})\hat{\mathbf{x}} - u(\mathbf{x})\hat{\mathbf{y}}$, then we obtain:

$$\frac{\partial \mathbf{h}(\mathbf{x})}{\partial t} + \nabla(g\eta(\mathbf{x})) = f(\mathbf{h}(\mathbf{x}) \times \mathbf{z}).$$

Thus the vectorial notation of the linear SWEs (2.2.1), in two dimensional plane, is:

$$\begin{cases} \dfrac{\partial \eta(\mathbf{x})}{\partial t} + \nabla \cdot (d\mathbf{h}(\mathbf{x})) = 0, & (2.2.8) \\ \dfrac{\partial \mathbf{h}(\mathbf{x})}{\partial t} + \nabla(g\eta(\mathbf{x})) = f(\mathbf{h}(\mathbf{x}) \times \mathbf{z}). & (2.2.9) \end{cases}$$

Similarly for the nonlinear case, we have:

$$\begin{cases} \dfrac{\partial H(\mathbf{x})}{\partial t} + \dfrac{\partial}{\partial x}(Hu)(\mathbf{x}) + \dfrac{\partial}{\partial y}(Hv)(\mathbf{x}) = 0, & (2.2.10) \\ \dfrac{\partial (Hu)(\mathbf{x})}{\partial t}\hat{\mathbf{x}} + \dfrac{\partial}{\partial x}\left(u^2(\mathbf{x})H(\mathbf{x}) + gH^2(\mathbf{x})/2\right)\hat{\mathbf{x}} \\ \quad + \dfrac{\partial}{\partial y}(Huv)(\mathbf{x})\hat{\mathbf{x}} = H(\mathbf{x})fv(\mathbf{x})\hat{\mathbf{x}}, & (2.2.11) \\ \dfrac{\partial (Hv)(\mathbf{x})}{\partial t}\hat{\mathbf{y}} + \dfrac{\partial}{\partial x}(Huv)(\mathbf{x})\hat{\mathbf{y}} \\ \quad + \dfrac{\partial}{\partial y}\left(v^2(\mathbf{x})H(\mathbf{x}) + gH^2(\mathbf{x})/2\right)\hat{\mathbf{y}} = -H(\mathbf{x})fu(\mathbf{x})\hat{\mathbf{y}}\ . & (2.2.12) \end{cases}$$

Equation (2.2.10) becomes:

$$\frac{\partial H(\mathbf{x})}{\partial t} + \nabla \cdot (H\mathbf{h})(\mathbf{x}) = 0.$$

And equations (2.2.11) and (2.2.12) become:

$$\begin{cases} \dfrac{\partial (Hu)(\mathbf{x})}{\partial t}\hat{\mathbf{x}} + \dfrac{\partial}{\partial x}(u^2H)(\mathbf{x})\hat{\mathbf{x}} + gH(\mathbf{x})\dfrac{\partial H(\mathbf{x})}{\partial x}\hat{\mathbf{x}} + \dfrac{\partial}{\partial y}(Huv)(\mathbf{x})\hat{\mathbf{x}} = H(\mathbf{x})fv(\mathbf{x})\hat{\mathbf{x}}, \\ \dfrac{\partial (Hv)(\mathbf{x})}{\partial t}\hat{\mathbf{y}} + \dfrac{\partial}{\partial y}(v^2H)(\mathbf{x})\hat{\mathbf{y}} + gH(\mathbf{x})\dfrac{\partial H(\mathbf{x})}{\partial y}\hat{\mathbf{y}} + \dfrac{\partial}{\partial x}(Huv)(\mathbf{x})\hat{\mathbf{y}} = -H(\mathbf{x})fu(\mathbf{x})\hat{\mathbf{y}}. \end{cases}$$

Which in vectorial notation can be written as follow:

$$\frac{\partial (H\mathbf{h})(\mathbf{x})}{\partial t} + \nabla \cdot (\mathbf{h}H\mathbf{h})(\mathbf{x}) + gH(\mathbf{x})\nabla H(\mathbf{x}) = f\hat{\mathbf{z}} \times (H\mathbf{h})(\mathbf{x}),$$

where

$$\begin{aligned} \nabla \cdot (\mathbf{h}H\mathbf{h})(\mathbf{x}) &= \begin{pmatrix} \nabla \cdot (Hu\mathbf{h})(\mathbf{x}) \\ \nabla \cdot (Hv\mathbf{h})(\mathbf{x}) \end{pmatrix}, \\ &= \begin{pmatrix} \frac{\partial}{\partial x}(u^2H)(\mathbf{x}) + \frac{\partial}{\partial y}(Huv)(\mathbf{x}) \\ \frac{\partial}{\partial x}(Huv)(\mathbf{x}) + \frac{\partial}{\partial y}(v^2H)(\mathbf{x}) \end{pmatrix}. \end{aligned}$$

Thus the vectorial form of the nonlinear SWEs (2.2.1), in two dimensional plane, is:

$$\begin{cases} \dfrac{\partial H(\mathbf{x})}{\partial t} + \nabla \cdot (H\mathbf{h})(\mathbf{x}) = 0, & (2.2.13) \\ \dfrac{\partial (H\mathbf{h})(\mathbf{x})}{\partial t} + \nabla \cdot (\mathbf{h}H\mathbf{h})(\mathbf{x}) + gH(\mathbf{x})\nabla H(\mathbf{x}) = f\hat{\mathbf{z}} \times (H\mathbf{h})(\mathbf{x}). & (2.2.14) \end{cases}$$

And we found the same expressions as those in the previous section, by deriving the Navier-Stokes equations.

Chapter 3

Spectral/hp element method and Discontinuous Galerkin method

3.1 Spectral/hp element method

This section outlines the spectral/hp element method for one-dimensional linear problems presented by Karniadakis and Sherwin (2005) with additional ideas from Schwab (1998).

The FEM is a discretisation technique to solve PDEs in its equivalent variational form. The classical approach of the FEM is to partition the computational domain into a mesh of many small subdomains and to approximate the unknown solution by piecewise linear interpolation functions, each with local support. The solution within each element is then reconstructed related to the neighbouring elements. The order of the method is equivalent to the order of the expansion basis. There are several number of references which elaborate the FEM, one can refer to Suli (2011) for more detail. The h-version of FEM consists of fixing the degree p of the piecewise polynomial basis functions. Any change of discretisation to increase the accuracy is achieved by means of a mesh refinement i.e. reduction in h, the element size mesh. The p-version of FEM is to fix the partitioning of the domain while the discretisation is changed through a modification in polynomial degree p. The hp-version of FEM is then straightforward; both the idea of mesh refinement and degree enhancement are combined. The spectral method approximate the solution by a truncated series of global basis function. The spectral element method encompasses the high accuracy of the spectral methods with the geometric flexibility of the FEM. Terminologically and methodologically, the spectral/hp element method involves all the methods mentioned above.

3.1.1 Discontinuous Galerkin methods

To illustrate the framework of the weighted residual method, we consider a domain Ω which is a subset of $\mathbb{R}$ defined as follow:

$$\Omega = \{x|0 < x < l\},$$

and assume that f is a given function. Then consider the following differential equation, with suitable initial and boundary conditions:

$$\mathbb{L}(u) = f, \tag{3.1.1}$$

where $\mathbb{L}$ is a continuous partial differential operator.

We assume that the exact solution $u(x,t)$ of the equation (3.1.1) can be approximated by $u^\delta(x,t)$ as:

$$u(x,t) \approx u^\delta(x,t) = u_0(x,t) + \sum_{i=1}^{N_{dof}} \hat{u}_i(t)\Phi_i(x), \tag{3.1.2}$$

where $u_0(x,t)$ satisfies the initial and boundary conditions, $\hat{u}_i(t)$ are N_{dof} (degree of freedom) unknown coefficients and $\Phi_i(x)$, the trial functions, satisfies the homogeneous boundary conditions.
The non-zero residual is defined as $\mathbb{L}(u^\delta) - f = R(u^\delta)$. The aim of the weighted residuals method is to minimise $R(u^\delta)$. Thus we force the residual to be zero by multiplying it with a weight function, $v_j(x)$, and integrate over the domain Ω, that is:

$$\int_\Omega v_j(x)R \ dx = (v_j(x), R) = 0, \quad j = 0, \ldots, N_{dof}. \tag{3.1.3}$$

This is true for some choice of the weight function v_j. For our concern, we choose $v_j = \Phi_j$. This produces the Galerkin method.

3.2 Formulation of the Galerkin problem

We assume that $a(.,.)$ is a bilinear form defined as follow:

$$a(v,u) = \int_0^l \frac{\partial v}{\partial x}\frac{\partial u}{\partial x}dx,$$

and we define the following spaces:

- energy space: $E(\Omega) = \{u|a(u,u) < \infty\} = H^1$ associated to the energy norm $||u||_E = \sqrt{a(u,u)}$,
- trial space: $\chi = \{u|u \in H^1, u(0) = g_D\}$ where g_D is a given constant,

- space of all test functions: $\nu = \{u | u \in H^1, u(0) = 0\} = H_0^1$.

- Multiplying equation (3.1.1) by an arbitrary test function $v(x) \in \nu$ and integrating by parts over the domain Ω leads to the weak formulation of the Galerkin method Suli (2011) which can be expressed as:

$$\text{find } u \in \chi \text{ such that } \int_\Omega \mathbb{L}(u)v(x)dx = \int_\Omega fv(x)dx, \quad \forall v \in \nu.$$

  Setting $a(u,v) = \int_\Omega \mathbb{L}(u)v(x)dx$ and $f(v) = \int_\Omega fv(x)dx$, we can write in a more formal form:

$$\text{find } u \in \chi \text{ such that } a(v,u) = f(v), \quad \forall v \in \nu. \tag{3.2.1}$$

  In approximating the exact solution numerically, we replace an infinite expansion by a finite representation. Because the trial space χ and the test space ν are infinite spaces, we select subspaces $\chi^\delta \subset \chi$ and $\nu^\delta \subset \nu$ containing a finite number of functions, and the weak problem can then be stated as:

$$\text{find } u^\delta \in \chi^\delta \text{ such that } a(v^\delta, u^\delta) = f(v^\delta), \quad \forall v^\delta \in \nu^\delta. \tag{3.2.2}$$

- To impose the Dirichlet boundary condition, we lift the solution by decomposing the function $u^\delta \in \chi^\delta$ into $u^\delta = u^\mathcal{H} + u^\mathcal{D}$ where $u^\mathcal{H} \in \nu^\delta$ which is known and $u^\mathcal{D} \in \chi^\delta$. The Galerkin form of the problem can now be stated as:

$$\text{find } u^\delta = u^\mathcal{H} + u^\mathcal{D} \text{ where } u^\mathcal{H} \in \nu^\delta, u^\delta \in \chi^\delta \text{ such that,}$$

$$a(v^\delta, u^\mathcal{H}) = f(v^\delta) - a(v^\delta, u^\mathcal{D}), \quad \forall v^\delta \in \nu^\delta. \tag{3.2.3}$$

  We do not need to impose any condition for Neumann boundary conditions because they will be introduced naturally to the problem.

3.2.1 Expansion bases

In the h-type method, a fixed order polynomial is used in every element and convergence is achieved by reducing the size of the elements. In the p-type method, a fixed mesh is used and convergence is achieved by increasing the order of the polynomial in every element. The spectral/hp element method combines attributes from both the h-type and p-type extensions permitting a combination of both approaches.

3.2.2 The h-type extension

This method decomposes the expansion into elemental contribution.

We start by partitioning the solution domain Ω as:

$$\Omega = \{x|0 < x < l\} = \bigcup_{e=1}^{N_{el}} \Omega^e \quad \text{where} \quad \bigcap_{e=1}^{N_{el}} \Omega^e = \emptyset,$$

and $\Omega^e = \{x|x_{e-1} < x < x_e\}$ with $0 = x_0 < x_1 < \cdots < x_{N_{el}-1} < x_{N_{el}} = l$, N_{el} indicates the number of elements.

Let us introduce the one dimensional standard region $\Omega_{st} = \{\xi| -1 < \xi < 1\}$. For every elemental region Ω^e, there exists a one-to-one mapping χ^e relating the global coordinate $x \in \Omega^e$ to the local coordinate $\xi \in \Omega_{st}$, that is $\Omega^e = \chi^e(\Omega_{st}), x = \chi^e(\xi)$.

Evidently, a linear mapping χ^e will suffice:

$$x = \chi^e(\xi) = \frac{(1-\xi)}{2} x_{e-1} + \frac{(1+\xi)}{2} x_e, \quad \xi \in \Omega_{st}. \tag{3.2.4}$$

Its analytic inverse is of the form:

$$\xi = (\chi^e)^{-1}(\xi) = \frac{2x - x_e - x_{e-1}}{x_e - x_{e-1}}, \quad x \in \Omega^e. \tag{3.2.5}$$

Consequently, the global modes $\Phi_i(x)$ can now be expressed in terms of the local expansion modes $\phi_p(\xi)$. Therefore, we get:

$$u^\delta = \sum_{i=0}^{N_{dof}-1} \hat{u}_i \Phi_i(x) = \sum_{e=1}^{N_{el}} \sum_{p=0}^{P} \hat{u}_p^e \phi_p^e(\xi)$$

where P is the order of the polynomial expansion and $\phi_p^e(\xi) = \phi_p([\chi^e]^{-1}(x))$.

3.2.3 The p-type extension

The partitioning of the domain is kept fixed and any change of discretisation is introduced through a modification in the polynomial of degree P.

We define $\mathcal{P}_P(\Omega_{st})$, the space of all polynomials of degree P defined on the standard element Ω_{st} and $\chi^\delta = \{u^\delta | u^\delta \in H^1, u^\delta(\chi^e(\xi)) \in \mathcal{P}_{P^e}(\Omega_{st}), e = 1, \ldots, N_{el}\}$, the discrete hp extension space.

- Let us first introduce the concepts of modal and nodal expansions.

  The modal expansion depends on the frequency basis, and there is a notion of hierarchy in the sense that higher order expansion sets are built from lower order expansions sets. Legendre polynomial, $L_p(x)$, is

an example of modal expansion. Recall that Legendre polynomials are the solutions of the Legendre's differential equation, defined as

$$(1-x^2)y'' - 2xy' + n(n+1)y = 0, \quad n \in \mathbb{N}.$$

An important property of the Legendre polynomial is its orthogonality in the Legendre inner product, that is:

$$(L_p(x), L_q(x)) = \int_{-1}^{1} L_p(x)L_q(x)dx = \left(\frac{2}{2n+1}\right)\delta_{pq}.$$

The nodal expansion is based on a series of node points. The expansion coefficients ($\hat{u}_p$) represent the approximate solution at a given set of nodes. An approximate solution using a nodal expansion does not necessarily satisfy the equation exactly at the nodal points. Lagrange polynomial $H_p(x)$, are an example of nodal expansion, defined as:

$$H_p(x) = \prod_{\substack{k=0 \\ k\neq p}}^{P} \frac{x - x_k}{x_p - x_k}, \quad 0 \leq p \leq P,$$

where x_k for $0 \leq k \leq P$, are $(P+1)$ node points. Lagrange polynomial verifies $H_p(x_q) = \delta_{pq}$.

- The choice of an expansion set is influenced by its numerical efficiency, conditioning and linear independence of the basis as well as its approximation properties. To illustrate some of these factors we consider the two expansions $\Phi_p^A(x) = L_p(x)$ and $\Phi_p^B(x) = H_p(x)$, defined above, in a Galerkin projection. The Galerkin or L^2 projection of a smooth function $f(x)$ in the domain Ω_{st}, onto the polynomial expansion $u^\delta(x)$ is the solution to the problem (3.2.2). In the absence of explicit boundary conditions, which need not be prescribed to obtain the solution for this problem, the trial and the test space are both in the space of square integrable functions, that is $\chi^\delta = \nu^\delta \subset L^2$. Letting $u^\delta(x) = \sum_{p=0}^{P} \hat{u}_p\Phi_p(x)$ and $v^\delta(x) = \sum_{p=0}^{P} \hat{v}_p\Phi_p(x)$, problem (3.2.2) becomes:

$$\hat{\boldsymbol{v}}^T\left[\boldsymbol{M}\hat{\boldsymbol{u}} = \boldsymbol{f}\right] \quad \Rightarrow \quad \boldsymbol{M}\hat{\boldsymbol{u}} = \boldsymbol{f} \quad \Rightarrow \quad \hat{\boldsymbol{u}} = \boldsymbol{M}^{-1}\boldsymbol{f},$$

where $\boldsymbol{M}$, known as the mass matrix, is invertible and

$$M_{pq} = (\Phi_p, \Phi_q), \hat{u} = [\hat{u}_0, \cdots, \hat{u}_P]^T, f = (f_0, \cdots, f_P) \text{ where } f_p = (\Phi_p, f).$$

The conditioning of the matrix $\boldsymbol{M}$ is related to the linear independence of the expansion. The conditioning number $\mathrm{k}_2 = ||\boldsymbol{M}||_2||\boldsymbol{M}^{-1}||_2$, where $||\boldsymbol{M}||_2$ denotes the matrix L^2 norm of $\boldsymbol{M}$, is important in the inversion matrix system. The Legendre basis is well conditioned with the exact

conditioning number $\mathrm{k}_2 = 2P + 1$. However, a Lagrange expansion has poor conditioning which reflects the fact that the bases are becoming numerically linearly dependent.

Another requirement is the boundary and interior decomposition. That is, the boundary modes have a magnitude of one at the elemental boundaries and are zero at all other boundaries; and interior modes only have magnitude in the interior of the element and are zero along all boundaries. The equispaced Lagrange expansion satisfies these conditions, where the end-points are included as nodal points. But it is not the case for the Legendre polynomials.

So far, Legendre polynomials seem to be the best choice for an expansion set since they are orthogonal and have well conditioned matrices.

- We have seen that polynomial nodal expansions are based upon the Lagrange polynomials which are associated with a set of nodal points which include the ends of the domain. The choice of these points, however, plays an important role in the stability of the approximation and the conditioning of the system. Note also that if we denote $g(x)$ the polynomial of order $(P+1)$ with zeros at the $(P+1)$ nodal points x_q, then we can write $H_p(x)$ in the more compact form as:

$$H_p(x) = \frac{g(x)}{g'(x)(x - x_p)}. \tag{3.2.6}$$

The spectral elements use Lagrange polynomials through the zeros of the Gauss-Lobatto polynomials (Section 3.3.1). Recall that the Jacobi polynomials $P_n^{\alpha,\beta}$ represent the family of polynomial solutions to a singular Sturm-Liouville problem . An important property of these polynomials is their orthogonal relationship:

$$\int_{-1}^{1} (1-x)^{\alpha}(1+x)^{\beta} P_p^{\alpha,\beta}(x) P_q^{\alpha,\beta}(x) dx = C\delta_{pq}, \tag{3.2.7}$$

where

$$C = \frac{2^{\alpha+\beta+1}}{2p+\alpha+\beta+1} \frac{\Gamma(p+\alpha+1)\Gamma(p+\beta+1)}{p!\Gamma(p+\alpha+\beta+1}},$$

where Γ is the gamma function. Thus the Legendre polynomial is a special case of Jacobi polynomials for $\alpha = \beta = 0$, that is $P_p^{0,0}(x) = L_p(x)$. Considering the nodal points at the roots of the polynomial $g(\xi) = (1-\xi)(1+\xi)P_{P-1}^{1,1}(\xi)$, we obtain the p-type expansion in the standard element Ω_{st}

$$\phi_p(\xi) \mapsto H_p(\xi) = \begin{cases} 1 & \text{for } \xi = \xi_p, \\ \dfrac{(\xi-1)(1+\xi)P_{P-1}^{1,1}(\xi)}{P(P+1)L_P(\xi)(\xi_p - \xi)} & \text{otherwise,} \end{cases} \qquad 0 \le p \le P. \tag{3.2.8}$$

All modes are polynomials of order P. If we use the Gauss-Legendre-Lobatto quadrature rule corresponding to the same choice of nodal points on which the expansion was defined, the mass matrix is diagonal due to the Kronecker delta property:

$$\boldsymbol{M}^e[p][q] = (H_p, H_q) = \sum_{i=0}^{P} w_i H_p(\xi_i) H_q(\xi_i) = \sum_{i=0}^{P} w_i \delta_{pi} \delta_{qi} = w_p \delta_{pq}, \tag{3.2.9}$$

where w_i are the weights for the Gauss-Legendre-Lobatto rule using $(P+1)$ points.

The diagonal components of the elemental mass matrix using the reduced quadrature rule are equal to the row sum of the elemental mass matrix using the exact integration . Considering the row sum

$$\sum_{q=0}^{P} M^e_{pq} = \sum_{q=0}^{P} (H_p(\xi), H_q(\xi)) = (H_p(\xi), \sum_{q=0}^{P} H_q(\xi)) = (H_p(\xi), 1) = w_p,$$

where w_p is the weight corresponding to the p^{th} point in the Gauss-Lobatto-Legendre quadrature rule using $(P+1)$ points. Since w_p is also the diagonal entry of the mass matrix using reduced integration, we see that mass lumping of the spectral element expansion is equivalent to constructing the mass matrix with reduced integration rule .

3.3 Elemental operations

To complete our Galerkin formulation, we need to know how to integrate and differentiate the polynomial bases in the standard region.

3.3.1 Numerical Integration

Galerkin formulation requires a technique to evaluate, within each elemental domain, integrals of the form

$$\int_{-1}^{1} u(\xi) d\xi. \tag{3.3.1}$$

The fundamental concept is the approximation of the integral by a finite summation of the form

$$\int_{-1}^{1} u(\xi) d\xi \approx \sum_{i=0}^{Q-1} w_i u(\xi_i), \tag{3.3.2}$$

where w_i are weights and ξ_i represents Q distinct points in the interval $-1 \leq \xi_i \leq 1$.

Gaussian quadrature is a particularly accurate method for treating integrals where the integrand, $u(\xi)$, is smooth. It defines a series of nodal points upon which we know all values of the integrand.

The integrand $u(\xi)$ is represented by a Lagrange polynomial using the Q points ξ_i, which are to be specified, that is

$$u(\xi) \approx \sum_{i=0}^{Q-1} u(\xi_i) H_i(\xi) + \epsilon(u), \tag{3.3.3}$$

where the $\epsilon(u)$ is the approximation error. Thus, substituting equation (3.3.3) into (3.3.2), we obtain a representation of the integral as a summation:

$$\int_{-1}^{1} u(\xi) d\xi = \sum_{i=0}^{Q-1} w_i u(\xi_i) + r(u), \quad \text{(Legendre integration)}, \tag{3.3.4}$$

where $w_i = \int_{-1}^{1} H_i(\xi) d\xi$ and $r(u) = \int_{-1}^{1} \epsilon(u) d\xi$.

To perform w_i, the integral of the Lagrange polynomial, we need to know the location of the zeros ξ_i. Introducing $\xi_{i,P}^{\alpha,\beta}$ to denote the P zeros of the P^{th} order Jacobi polynomial $P_P^{\alpha,\beta}$, such that

$$P_P^{\alpha,\beta}(\xi_{i,P}^{\alpha,\beta}) = 0, \quad i = 0, \ldots, P-1 \text{ where } \xi_{0,P}^{\alpha,\beta} < \xi_{1,P}^{\alpha,\beta} < \cdots < \xi_{P-1,P}^{\alpha,\beta},$$

Gauss-Lobatto-Legendre defines zeros and weights which approximate the Legendre integral (3.3.4) as:

$$\begin{aligned} \xi_i &= \begin{cases} -1, & i = 0, \\ \xi_{i-1,Q-2}^{1,1}, & i = 1, \ldots, Q-2, \\ 1, & i = Q-1. \end{cases} \\ w_i^{0,0} &= \frac{2}{Q(Q-1)[L_{Q-1}(\xi_i)]^2}, \quad i = 0, \ldots, Q-1. \\ r(u) &= 0, \quad \text{if } u(\xi) \in \mathcal{P}_{2Q-3}([-1,1]). \end{aligned} \tag{3.3.5}$$

3.3.2 Differentiation

If we assume that $u(\xi) \in \mathcal{P}_P([-1,1])$, then it can be exactly expressed in the terms of Lagrange polynomials $H_i(\xi)$ through a set of Q nodal points ξ_i $(0 \leq i \leq Q-1)$, as

$$u(\xi) = \sum_{i=0}^{Q-1} u(\xi_i) H_i(\xi),$$

where $Q \geq P+1$. Therefore the derivative of $u(\xi)$ can be represented as

$$\frac{du(\xi)}{d\xi} = \sum_{i=0}^{Q-1} u(\xi_i) \frac{d}{d\xi} H_i(\xi).$$

Typically, we only require the derivative at the nodal points ξ_i which is given by

$$\left.\frac{du(\xi)}{d\xi}\right|_{\xi=\xi_i} = \sum_{j=0}^{Q-1} d_{ij} u(\xi_j) \quad \text{where} \quad d_{ij} = \left.\frac{dH_j(\xi)}{d\xi}\right|_{\xi=\xi_i}.$$

From (3.2.6), taking $g(\xi) = g_Q(\xi) = \prod_{j=0}^{Q-1}(\xi - \xi_j)$, we obtain

$$\frac{dH_i(\xi)}{d\xi} = \frac{g'_Q(\xi)(\xi - \xi_i) - g_Q(\xi)}{g'_Q(\xi_i)(\xi - \xi_i)^2}.$$

Noting that the numerator and the denominator of this expression are zero as $\xi \to \xi_i$ and $g_Q(\xi_i) = 0$ by definition, we get

$$\lim_{\xi \mapsto \xi_i} \frac{dH_i(\xi)}{d\xi} = \lim_{\xi \mapsto \xi_i} \frac{g''_Q(\xi)}{2g'_Q(\xi)} = \frac{g''_Q(\xi)}{2g'_Q(\xi)}.$$

Therefore

$$d_{ij} = \begin{cases} \dfrac{g'_Q(\xi)}{g'_Q(\xi)} \dfrac{1}{(\xi_i - \xi_j)} & i \neq j, \\ \dfrac{g''_Q(\xi)}{2g'_Q(\xi)} & i = j. \end{cases} \tag{3.3.6}$$

Equation (3.3.6) is the general representation of the derivative of the Lagrange polynomials evaluated at the nodal points ξ_i $(0 \leq i \leq Q-1)$.

Denoting by $\xi_{i,P}^{\alpha,\beta}$ $(0 \leq i \leq P-1)$ the P zeros of the Jacobi polynomial $P_P^{\alpha,\beta}$, the derivative matrix d_{ij} for Gauss-Lobatto-Legendre is defined as:

$$d_{ij} = \begin{cases} \dfrac{-Q(Q-1)}{4}, & i = j = 0, \\ \dfrac{L_{Q-1}(\xi_i)}{L_{Q-1}(\xi_j)} \dfrac{1}{(\xi_i - \xi_j)}, & i \neq j, \quad 0 \leq i, j \leq Q-1, \\ 0, & 1 \leq i = j \leq Q-2, \\ \dfrac{Q(Q-1)}{4}, & i = j = Q-1, \end{cases} \tag{3.3.7}$$

where ξ_i are the same as defined in (3.3.5).

3.4 The spectral/*hp* element discontinuous Galerkin methods for shallow water equations

To solve numerically the SWEs, the spectral/*hp* element discontinuous Galerkin method presented in will be used in this study. The main idea is to bring the

PDEs problem into a linear algebra problem, which is easy to compute numerically, by the weak formulation. As a finite element method, the computational domain Ω must be partitioned into N_{el} conforming triangular elements Ω_e with boundaries $\partial\Omega_e$. The advantage of using DG method is that it works over a trial function space that is piecewise discontinuous. That is what we need here since the domain have been partitioned and the solution is allowed to be discontinuous over the element boundaries. Let V_δ be a discrete space which contains a finite number of functions such that:

$$V_\delta = \{v_\delta \in L^2(\Omega) : v_\delta|_{\Omega_e} \in \mathcal{P}^p(\Omega_e) \ \ \forall\Omega_e\},$$

where $\mathcal{P}^p(\Omega_e)$ is the space of all polynomial of degree at most p defined on the elemental domain Ω_e. Now we can formulate our Galerkin problem. First we approximate the exact solution $\mathbf{U}$ of the equation (2.2.1) with $\mathbf{U}_\delta \in V_\delta$. After, we multiply with an arbitrary smooth test function $\phi_\delta \in V_\delta$ and integrate over the local element Ω_e i.e.

$$\begin{aligned}&\int_{\Omega_e} \phi_\delta \frac{\partial \mathbf{U}_\delta}{\partial t}\ \mathrm{d}\Omega_e + \int_{\Omega_e} \phi_\delta \nabla\cdot\mathbf{F}(\mathbf{U}_\delta)\ \mathrm{d}\Omega_e \\ &= \int_{\Omega_e} \phi_\delta \mathbf{S}(\mathbf{U}_\delta)\ \mathrm{d}\Omega_e\,. \end{aligned} \tag{3.4.1}$$

Applying the integration by part for the flux term, the weak formulation of the DG problem can be stated as follow. Find $\mathbf{U}_\delta \in V_\delta$ such that $\forall\phi_\delta \in V_\delta$

$$\int_{\Omega_e} \phi_\delta \frac{\partial \mathbf{U}_\delta}{\partial t}\ \mathrm{d}\Omega_e - \int_{\Omega_e} \nabla\phi_\delta\cdot\mathbf{F}(\mathbf{U}_\delta)\ \mathrm{d}\Omega_e + \int_{\partial\Omega_e} \phi_\delta \left(\hat{\mathbf{F}}(\mathbf{U}_\delta)\cdot\mathbf{n}\right)\ \mathrm{dS} = \int_{\Omega_e} \phi_\delta \mathbf{S}(\mathbf{U}_\delta)\ \mathrm{d}\Omega_e, \tag{3.4.2}$$

where $\mathbf{n}$ is the outward unit normals to the element Ω_e and $\hat{\mathbf{F}}$ is the numerical flux to be defined later.

We seek an approximation U_δ whose restriction to each element Ω_e is, for each value of the time variable, an element of the local space $\mathcal{P}^p(\Omega_e)$. The integrals over the local elements Ω_e could either be computed exactly or approximately by using a numerical quadrature method as discussed in the previous section. Note that the function $\mathbf{U}_\delta$ is discontinuous along the boundaries $\partial\Omega_e$ of a local element and the boundary integral is not uniquely defined. Consequently the analytic flux $\mathbf{F}(\mathbf{U}_\delta)$ have to be replaced by a numerical flux $\hat{\mathbf{F}}(\mathbf{U}_\delta)$ which will resolve the discontinuity along the element edges and couple all the elements together. The accuracy of the method will depend in a large part on this normal flux definition (Bernard *et al.* (2009)). The numerical flux can be computed by using Riemann solver. We thus apply, in this work, four different types of numerical flux defined in the next chapter (Chapter 4).

Boundary condition

To complete the computation, it remains to choose the boundary conditions. The slip wall boundary conditions is used in this work since it is a

common condition to bound fluid regions. Here normal velocity component is set to be zero, Eskilsson and Sherwin (2000), and

$$\eta_L = \eta_R, \quad u_L = -u_R \quad \text{and} \quad v_L = v_R.$$

Chapter 4

Numerical Flux

As defined previously in Section (1.3), a numerical flux is the amount of information that passes at the boundary, from one surface to another surface, following the normal direction. The choice of the discontinuous Galerkin scheme permits the solutions to be discontinued at the boundaries, thus one needs to apply the numerical flux to find the appropriate solution at the boundary. Computation of the numerical flux is assured by Riemann solvers which are solutions of the Riemann problem. This chapter presents four kinds of numerical flux which will be used for comparison in this work. To achieve this, the chapter starts by a short introduction of the Riemann problem.

4.1 Godunov scheme

A system of equations is said to be in conservative form if it can be written in the following form:

$$\mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = \mathbf{0}\,, \tag{4.1.1}$$

with initial and boundary condition given by

$$(\text{IC}) : \mathbf{U}(x,0) = \mathbf{U}^{(0)}(x) \tag{4.1.2}$$

$$(\text{BC}) : \mathbf{U}(0,t) = \mathbf{U}_L(t) \quad \text{and} \quad \mathbf{U}(l,t) = \mathbf{U}_R(t), \tag{4.1.3}$$

where $\mathbf{U}$ is the vector of conserved variables, while $\mathbf{F}(\mathbf{U})$ is the fluxes vector, $\mathbf{U}(x,0)$ is the initial data at time $t = 0$, $[0,l]$, for $l \in \mathbb{R}$, is the spatial domain and boundary conditions are assumed to be represented by the boundary functions $\mathbf{U}_L(t)$ and $\mathbf{U}_R(t)$.

The Riemann problem is a composition of conservation law and a piecewise constant data having a single discontinuity. For the one-dimensional time-dependent equations, the Riemann problem is the initial value problem (IVP) for the conservation laws

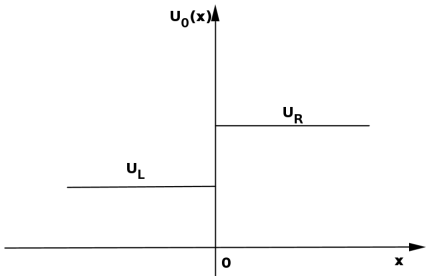

Figure 4.1: Initial condition for the Riemann problem, from Toro (2009).

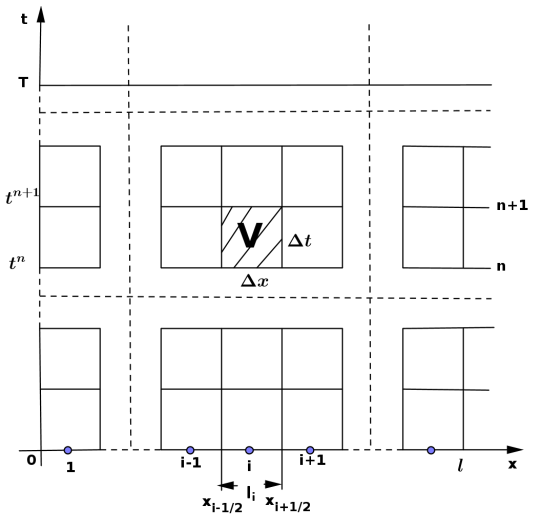

Figure 4.2: Control volume V, from Zanotti and Manca (2010).

$$\begin{cases} \mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \\ \text{IC: } \mathbf{U}(x,0) = \mathbf{U}^{(0)}(x) = \begin{cases} \mathbf{U_L} & \text{if } x > 0, \\ \mathbf{U_R} & \text{if } x < 0, \end{cases} \end{cases} \tag{4.1.4}$$

where $\mathbf{U_L}$ and $\mathbf{U_R}$ are given constant values. The initial condition (IC) says that at the initial state, the function has a constant value $\mathbf{U_L}$ for all negative x, and a constant value $\mathbf{U_R}$ for all positive x, but differs between left (L) and right (R), as shown in Figure (4.1). Owing to the fact that the grids are disconnected, Riemann problems arise implicitly in finite volume methods for the solution of conservation law equations.

Discretising the spatial domain $[0, l]$ into $N(\in \mathbb{N})$ computing cells $I_i = [x_{i-1/2}, x_{i+1/2}]$, $1 \leq i \leq N$, of size $\Delta x = x_{i+1/2} - x_{i-1/2}$, and the temporal domain $[0, T]$, where T is a chosen final time, into $M(\in \mathbb{N})$ computing cells $K^n = [t^n, t^{n+1}]$, $1 \leq n \leq M$ of size $\Delta t = t^{n+1} - t^n$; a control volume $V = I_i \times [t^n, t^{n+1}]$ can be defined as depicted in Figure (4.2). Integrating the conservative equations (4.1.1) over the control volume V gives the integral

form of the conservation laws. First by integrating in space over I_i:

$$\frac{d}{dt}\int_{x_{i-1/2}}^{x_{i+i/2}} \mathbf{U}(x,t)\,dx = \mathbf{F}(\mathbf{U}(x_{i-1/2},t)) - \mathbf{F}(\mathbf{U}(x_{i+1/2},t)),$$

and then in time between t^n and t^{n+1}, with $t^n < t^{n+1}$ to obtain

$$\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t^{n+1})\,dx = \int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t^n)\,dx + \int_{t^n}^{t^{n+1}} \left[\mathbf{F}(\mathbf{U}(x_{i-1/2},t)) - \mathbf{F}(\mathbf{U}(x_{i+1/2},t))\right]\,dt. \quad (4.1.5)$$

Adopting the following notations:

$$\mathbf{U}_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} \mathbf{U}(x,t^n)\,dx, \quad (4.1.6)$$

and

$$\mathbf{F}_{i\pm1/2} = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \mathbf{F}(\mathbf{U}(x_{i\pm1/2},t))\,dt, \quad (4.1.7)$$

we can re-write the integration form of the conservation laws (4.1.5) as

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n + \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i-1/2} - \mathbf{F}_{i+1/2}\right). \quad (4.1.8)$$

No approximations are made so far, thus (4.1.8) is not a numerical scheme yet. It becomes a numerical scheme, and indeed it is called "Godunov scheme" when approximations are introduced for the computations of the numerical fluxes $\mathbf{F}_{i-1/2}$ and an interpretation is given to the average $\mathbf{U}_i$. Depending on the method to compute the fluxes at each interface, $\mathbf{F}_{i-1/2}$ and $\mathbf{F}_{i+1/2}$, diverse numerical algorithms can be conceived from (4.1.8). At the adjacent numerical cells the quantity $\mathbf{U}_i$ manifests a jump, thus generating a sequence of local Riemann problems. Hence, (4.1.8) is said to be a Godunov's first-order upwind method if the fluxes are calculated by solving such sequence of local Riemann problems. The left and right states are the same piecewise constant distribution of data giving by (4.1.6). Solving the Riemann problem provides either the term $\mathbf{U}(x_{i\pm1/2},t)$ to be used in (4.1.7), or the $\mathbf{F}[\mathbf{U}(x_{i\pm1/2,t})]$.

4.2 Approximate Riemann solvers

This is the conservative form of the Godunov scheme. And the intercell numerical flux is given by, Toro (2009)

$$\mathbf{F}_{i+1/2} = \mathbf{F}(\mathbf{U}_{i+1/2}(0)), \quad (4.2.1)$$

where $\mathbf{U}_{i+1/2}(0)$ is the exact similarity solution $\mathbf{U}_{i+1/2}(x/t)$ of the Riemann problem

$$\begin{cases} \mathbf{U}_t + \mathbf{F}(\mathbf{U})_x = 0, \\ \mathbf{U}(x,0) = \begin{cases} \mathbf{U_L} & \text{if } x > 0, \\ \mathbf{U_R} & \text{if } x < 0, \end{cases} \end{cases} \tag{4.2.2}$$

evaluated at $x/t = 0$.

Similarity solutions to PDEs are solutions which depend on certain groupings of the independent variables, rather than on each variable separately.

A similarity solution is defined mathematically as solution where a change of variables allows for a reduction in the number of independent variables. In fluids, a similarity solution can be interpreted physically as a case that, when appropriately non-dimensionalised, causes the data taken at different locations or times to collapse.

The similarity method is one of the standard methods for obtaining exact solutions of PDEs. The number of independent variables in a PDE is reduced by one by making use of appropriate combinations of the original independent variables as new independent variables, called "similarity variables." The similarity variables can themselves be identified by using the invariance properties of PDEs when subjected to finite or infinitesimal transformations.

The numerical flux can be computed in two ways, either by giving an approximation to the state $\mathbf{U}_{i+1/2}(0)$ which is then used in (4.2.1), this is known as approximate Riemann solver, or by giving an approximation to the flux directly, this is known as the direct Riemann solver. A variety of numerical fluxes are available to approximate the solution of the resulting Riemann problem. The way in which we approach the numerical flux in function of the discrete unknowns determines the numerical scheme.

The time integral form of (4.2.2) on the control volume defined in figure 4.3 (left) gives:

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)\,dx = \int_{x_L}^{x_R} \mathbf{U}(x,0)\,dx + \int_0^T \mathbf{F}(\mathbf{U}(x_L,t))\,dt - \int_0^T \mathbf{F}(\mathbf{U}(x_R,t))\,dt,$$

thus

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)\,dx = x_R\mathbf{U_R} - x_L\mathbf{U_L} + T(\mathbf{F_L} - \mathbf{F_R}), \tag{4.2.3}$$

where $\mathbf{F_L} = \mathbf{F}(\mathbf{U_L})$ and $\mathbf{F_R} = \mathbf{F}(\mathbf{U_R})$.

But we can also have:

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)\,dx = \int_{x_L}^{TS_L} \mathbf{U}(x,T)\,dx + \int_{TS_L}^{TS_R} \mathbf{U}(x,T)\,dx + \int_{TS_R}^{x_R} \mathbf{U}(x,T)\,dx,$$

thus

$$\int_{x_L}^{x_R} \mathbf{U}(x,T)\,dx = \int_{TS_L}^{TS_R} \mathbf{U}(x,T)\,dx + (TS_L - x_L)\mathbf{U_L} + (TS_R - x_R)\mathbf{U_R} \tag{4.2.4}$$

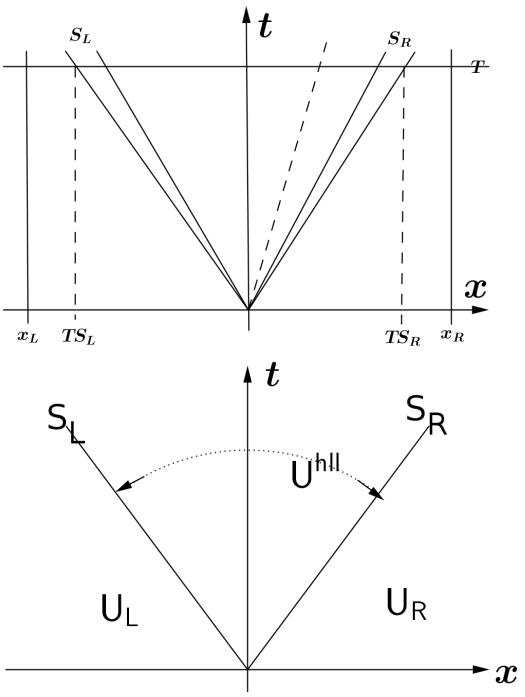

Figure 4.3: Control volume $[x_L, x_R] \times [0, T]$ (left), three wave structure of the HLL approximate Riemann solver (right), from Toro (2009).

Comparing the two equations (4.2.3) and (4.2.4), and dividing by the length $T(S_R - S_L)$, we obtain the integral average of the exact solution of the Riemann problem between the slowest and the fastest signals at time T, Toro (2009):

$$\mathbf{U}^{hll} = \frac{1}{T(S_R - S_L)} \int_{TS_L}^{TS_R} \mathbf{U}(x, T)\, dx = \frac{S_R \mathbf{U_R} - S_L \mathbf{U_L} + \mathbf{F_L} - \mathbf{F_R}}{S_R - S_L}. \quad (4.2.5)$$

For the linear equation, the HLL (Harten, Lax and van Leer) Riemann solver is used. This method assumes two waves model as illustrated in figure 4.3 (right) and computes directly an approximation for the intercell numerical flux. This gives us a left state $\mathbf{U_L}$, a right state $\mathbf{U_R}$ and a middle state $\mathbf{U}^{hll}$. An approximate Riemann solver is defined in Harten *et al.* (1983):

$$\mathbf{U}(x/t; \mathbf{U}_L, \mathbf{U}_R) = \begin{cases} \mathbf{U}_L & x/t \le S_L \\ \mathbf{U}^{hll} & S_L \le x/t \le S_R \\ \mathbf{U}_R & x/t \ge S_R \end{cases} \quad (4.2.6)$$

where $\mathbf{U}^{hll}$ is the constant state vector defined in (4.2.5), and S_L and S_R are the wave speeds. We will define them later.

We now recall the Rankine-Hugoniot condition, Toro (2009). Let us consider the integral form of equation (4.1.1) on the interval $[x_L, x_R]$ at time t:

$$\frac{d}{dt} \int_{x_L}^{x_R} \mathbf{U}(x, t)\, dx = \mathbf{F}(\mathbf{U}(x_L, t)) - \mathbf{F}(\mathbf{U}(x_L, t)), \quad (4.2.7)$$

where x_L and x_R are chosen such that $x_L \leq s(t) \leq x_R$ where $s(t)$ is a line of discontinuity of the solution. Thus we can write (4.2.7) as follow:

$$\frac{d}{dt}\int_{x_L}^{s(t)} \mathbf{U}(x,t)\,dx + \frac{d}{dt}\int_{s(t)}^{x_R} \mathbf{U}(x,t)\,dx = \mathbf{F}(\mathbf{U}(x_L,t)) - \mathbf{F}(\mathbf{U}(x_L,t)).$$

Recalling that:

$$\frac{d}{dt}\int_{x_1(t)}^{x_2(t)} f(x,t)\,dx = \int_{x_1(t)}^{x_2(t)} \frac{\partial f(x,t)}{\partial t}\,dx + f(x_2(t),t)\frac{dx_2(t)}{dt} - f(x_2(t),t)\frac{dx_2(t)}{dt},$$

we obtain :

$$\begin{aligned}\mathbf{F}(\mathbf{U}(x_L,t)) - \mathbf{F}(\mathbf{U}(x_L,t)) = &\left[\mathbf{U}(s_L,t) - \mathbf{U}(s_R,t)\right] S \\ &+ \int_{x_L}^{s(t)} \frac{\partial \mathbf{U}(x,t)}{\partial t}\,dx + \int_{s(t)}^{x_R} \frac{\partial \mathbf{U}(x,t)}{\partial t}\,dx,\end{aligned} \tag{4.2.8}$$

where $S = \dfrac{ds(t)}{dt}$ and $\mathbf{U}(s_L,t)$ and $\mathbf{U}(s_R,t)$ are the left and right limit of $\mathbf{U}(x,t)$ for $x \to s(t)$. And when $x_L \to s(t)$ and $x_R \to s(t)$ the two integrals on the right hand side of equation (4.2.8) tend to 0 and we obtain the Rankine-Hugoniot condition:

$$\mathbf{F}(\mathbf{U}(x_L,t)) - \mathbf{F}(\mathbf{U}(x_L,t)) = \left[\mathbf{U}(s_L,t) - \mathbf{U}(s_R,t)\right] S, \tag{4.2.9}$$

usually expressed as $\Delta\mathbf{F} = S\Delta\mathbf{U}$.

Applying this Rankine-Hugoniot condition across the left and right waves of our model gives:

$$\begin{aligned}\mathbf{F}^{hll} &= \mathbf{F}_L + S_L(\mathbf{U}^{hll} - \mathbf{U}_L),\\ \mathbf{F}^{hll} &= \mathbf{F}_R + S_R(\mathbf{U}^{hll} - \mathbf{U}_R).\end{aligned}$$

Combining these two equations leads to:

$$\mathbf{F}^{hll} = \frac{1}{2}\left[\mathbf{F}_L + \mathbf{F}_R + S_L(\mathbf{U}^{hll} - \mathbf{U}_L) + S_R(\mathbf{U}^{hll} - \mathbf{U}_R)\right],$$

and applying the value of $\mathbf{U}^{hll}$ given in (4.2.5) gives the value of the numerical flux:

$$\mathbf{F}^{hll} = \frac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_LS_R(\mathbf{U}_R - \mathbf{U})_L}{S_R - S_L}. \tag{4.2.10}$$

The corresponding HLL intercell flux for the approximate Godunov method is then given by (Toro (2009)):

$$\mathbf{F}^{hll}_{i+1/2} = \begin{cases}\mathbf{F}_L, & \text{if } 0 \leq S_L,\\ \dfrac{S_R\mathbf{F}_L - S_L\mathbf{F}_R + S_LS_R(\mathbf{U}_R - \mathbf{U}_L)}{S_R - S_L}, & \text{if } S_L \leq 0 \leq S_R,\\ \mathbf{F}_R, & \text{if } 0 \geq S_R.\end{cases} \tag{4.2.11}$$

What remains is the computation of the wave speeds. There are several possible choice for the wave speeds but in this work, we choose the one proposed by Davis (1988) presented in Toro (2009):

$$S_L = u_L - \sqrt{gd} \quad \text{and} \quad S_R = u_R + \sqrt{gd}\,,$$

where $\sqrt{gd}$ is the long wave speed. Thus, we have:

- If $S_L \geq 0$:

$$\hat{\mathbf{F}} = \mathbf{F}(\mathbf{U_L}) = \begin{pmatrix} du_L \\ g\eta_L \\ 0 \end{pmatrix}$$

- If $S_R \leq 0$:

$$\hat{\mathbf{F}} = \mathbf{F}(\mathbf{U_R}) = \begin{pmatrix} du_R \\ g\eta_R \\ 0 \end{pmatrix}$$

- Otherwise:

$$\hat{\mathbf{F}} = \frac{1}{S_R - S_L} \begin{pmatrix} S_R du_L - S_L du_R + S_L S_R(\eta_R - \eta_L) \\ S_R \eta_L g - S_L \eta_R g + S_R S_L(u_R - u_L) \\ S_R S_L(v_R - v_L) \end{pmatrix}$$

At this end, we should take note that the assumption of a two waves configuration is correct only for hyperbolic systems of two equations, such as the 1D SWEs, Toro (2009). Thus it works only if there is no contact and shear waves in the flow. Otherwise, another approximate solver is used like the HLLC, where the letter "C" stands for contact, presented in Toro (2009). This solver is chosen for the nonlinear SWEs. Figure 4.4 illustrates the assumed wave

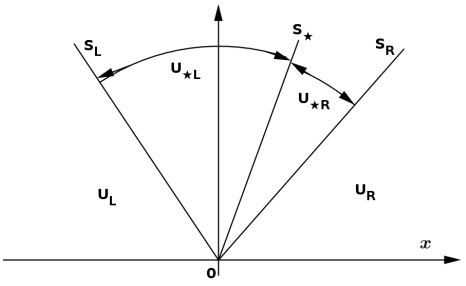

Figure 4.4: HLLC Riemann solver for x-split 2D shallow water equations from Toro.

structure in the HLLC Riemann solver, we added a middle speed S_*. There are now two distinct fluxes for the star region, and we need to estimates S_*

for the speed of the middle wave. The HLLC approximate Riemann solver is given as follow, from Toro (2009):

$$\mathbf{U}(x,t) = \begin{cases} \mathbf{U}_L & \text{if } x/t \leq S_L, \\ \mathbf{U}_{*L} & \text{if } S_L \leq x/t \leq S_*, \\ \mathbf{U}_{*R} & \text{if } S_* \leq x/t \leq S_R, \\ \mathbf{U}_R & \text{if } x/t \geq S_R. \end{cases}$$

The HLLC numerical flux is then, from Toro (2009):

$$\mathbf{F}^{hllc}_{i+\frac{1}{2}} = \begin{cases} \mathbf{F}_L & \text{if } 0 \leq S_L, \\ \mathbf{F}_{*L} & \text{if } S_L \leq 0 \leq S_*, \\ \mathbf{F}_{*R} & \text{if } S_* \leq 0 \leq S_R, \\ \mathbf{F}_L & \text{if } 0 \geq S_R, \end{cases}$$

where

$$\begin{aligned} \mathbf{F}_{*L} &= \mathbf{F}_L + S_L(\mathbf{U}_{*L} - \mathbf{U}_L) \\ \mathbf{F}_{*R} &= \mathbf{F}_R + S_R(\mathbf{U}_{*R} - \mathbf{U}_R) \\ S_* &= \frac{S_L h_R(u_R - S_R) - S_R h_L(u_L - S_L)}{h_R(u_R - S_R) - h_L(u_L - S_L)} \\ \mathbf{U}_{*K} &= H\left(\frac{S_K - u_K}{S_K - S_*}\right)\begin{pmatrix} 1 \\ S_* \\ v_K \end{pmatrix} \\ S_L &= u_L - a_L q_L \\ S_R &= u_R - a_R q_R \\ a_K &= \sqrt{g H_K} \\ q_K &= \begin{cases} \sqrt{\frac{1}{2}\left(\frac{(H_* + H_K)H_*}{H_K^2}\right)} & \text{if } H_* > H_K, \\ 1 & \text{if } H_* \leq H_K \end{cases} \\ H_* &= \frac{1}{2}\left(\frac{1}{2}(a_L + a_R) + \frac{1}{4}(u_L - u_R)\right) \end{aligned}$$

- Define the average flux at $x_{i-1/2}$ based on the data $\mathbf{U}^n_{i-1}$ and $\mathbf{U}^n_{i+1}$ to the left and right of this point.

$$\begin{aligned} \mathbf{F}^c_{i+\frac{1}{2}} &= \frac{1}{2}\left(\mathbf{F}(\mathbf{U}^n_{i-1}) + \mathbf{F}(\mathbf{U}^n_{i+1})\right) \\ &= \frac{1}{2}\left(\mathbf{F}_L + \mathbf{F}_R\right). \end{aligned}$$

- The Rusanov flux is generated from the HLL flux. Suppose a positive wave speed estimate S is available. By substituting the speeds S_L and S_R in HLL into $-S$ and S respectively.

$$\mathbf{F}^{rus}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{F}_L + \mathbf{F}_R\right) + \frac{1}{2}S\left(\mathbf{U}_L - \mathbf{U}_R\right),$$

  where $S = \max\{|u_L| + a_L, |u_R| + a_R\}$ and $a_K = \sqrt{gH_K}$.

- Choosing the largest possible speed namely $S = |\lambda|_{\max}$ results the Lax-Friedrichs flux

$$\mathbf{F}^{lax}_{i+\frac{1}{2}} = \frac{1}{2}\left(\mathbf{F}_L + \mathbf{F}_R\right) + \frac{1}{2}|\lambda|_{\max}\left(\mathbf{U}_L - \mathbf{U}_R\right),$$

  where $\lambda_{\max}$ is the maximum eigenvalue i.e. $\lambda_{\max} = \max\{u - a, u, u + a\}$, $a = \sqrt{gH}$.

At this end, we can see that the Lax-Friedrichs flux and the Rusanov flux as closely related as we will see from the experimental results in the next chapter (Chapter 5).

Chapter 5

Numerical Results

This chapter will show and compare all the numerical and convergence results of the method as shown in (3.4.2) whith to the four presented numerical fluxes. The DG scheme was implemented in the open-source spectral/hp library Nektar++, [Karniadakis and Sherwin (2005); nek]. Nektar++ a C++ object-oriented toolkit which allows developers to implement spectral solvers for a variety of different engineering problems. Nektar++ provides the fundamental tools associated with high-order FEM, such as the calculation of expansion functions, inner products and differentiation. My main work has been focused on implementing the new three numerical fluxes, presented in Chapter 5, which are the average flux, the Rusanov flux and the Lax-Friedrich flux. The HLL and HLLC numerical fluxes are already implemented in the spectral/hp library Nektar++ and have been modified appropriately in accordance with the scheme. The corresponding codes can be found in Appendix A.

5.1 Error norm

In order to evaluate the numerical solution quantitatively, two norms, known as L_2-norm and L_∞-norm, are introduced and defined as follows:

$$L_2 = ||\eta_\delta - \eta||_2 = \left[\int (\eta_\delta - \eta)^2 dx\right]^{1/2},$$
$$L_\infty = \max(|\eta_\delta - \eta|),$$

where η is the analytical solution and η_δ is the numerical solution.

But one should notice that not every equations have an analytical solution, most of the time, they are difficult to solve mathematically. In that case, the use of a reference solution is required. A reference solution is obtained numerically by using a very refined grid with high local order approximation. Remember that the hallmark of spectral/hp method is its accuracy when the element size mesh is decreased as small as possible and when the degree of the

polynomial expansion is increased as large as possible (Chapter 3). The error estimate then made by comparing the numerical solution with this generated reference solution. The accuracy of the method is assured if the problem is well-posed. Indeed,

$$\begin{aligned} L_2 &= ||\eta_\delta - \eta||_2, \\ &= ||\eta_\delta - \eta_{ref} + \eta_{ref} - \eta||_2, \\ &\leq ||\eta_\delta - \eta_{ref}||_2 + \underbrace{||\eta_{ref} - \eta||_2}_{<\mathcal{O}\left(\frac{1}{N_{el}^K}\right)}, \\ &< ||\eta_\delta - \eta_{ref}||_2 + \mathcal{O}\left(\frac{1}{N_{el}^K}\right), \end{aligned}$$

where N_{el} is the number of elements of the discretised domain where the numerical solution η_δ is computed, η_{ref} is the reference solution, and $K > 1$.

5.2 Description of the test problem

The performance of the numerical schemes for the SWEs is evaluated by applying a test case problem. Here the Rossby waves on a 2D space are used to illustrate the work and to test the accuracy of each numerical flux.

Rossby waves are undulating movements of atmospheric or oceanic circulation wavelength whose initiation is due to the variation of the Coriolis force depending on latitude. This is described by the nonlinear SWEs. The computational domain is $\mathbf{x} \in [-24, 24] \times [-8, 8]$. All boundaries are treated as walls. The zeroth-order initial conditions are given by Boyd (1980):

$$\begin{aligned} \eta(\mathbf{x}, 0) = \eta_0(\mathbf{x}) &= \Gamma(x)\left(\frac{3 + 6y^2}{4}\right) e^{-\frac{1}{2}y^2}, \\ u(\mathbf{x}, 0) = u_0(\mathbf{x}) &= \Gamma(x)\left(\frac{-9 + 6y^2}{4}\right) e^{-\frac{1}{2}y^2}, \\ v(\mathbf{x}, 0) = v_0(\mathbf{x}) &= \frac{\partial \Gamma(x)}{\partial x} 2y e^{-\frac{1}{2}y^2}, \end{aligned}$$

where $\Gamma(x) = 0.771a^2 \operatorname{sech}^2(ax)$ and where $a = 0.395$ is the parameter determining the amplitude of the solitary wave. The corresponding initial conditions $\mathbf{U}(\mathbf{x}, 0)$ are depicted in Figure 5.1. The Coriolis parameter is $f = y$ and the water depth is $d = 1$. The analytical solution is not known, thus in order to see the convergence, we define a reference numerical solution, and compare the computed solutions against this reference numerical solution.

5.3 Computational results

The reference numerical solution is obtained by using a very refined grid with high local order approximation. Here we use $(Ne, P) = (2020, 12)$, where Ne

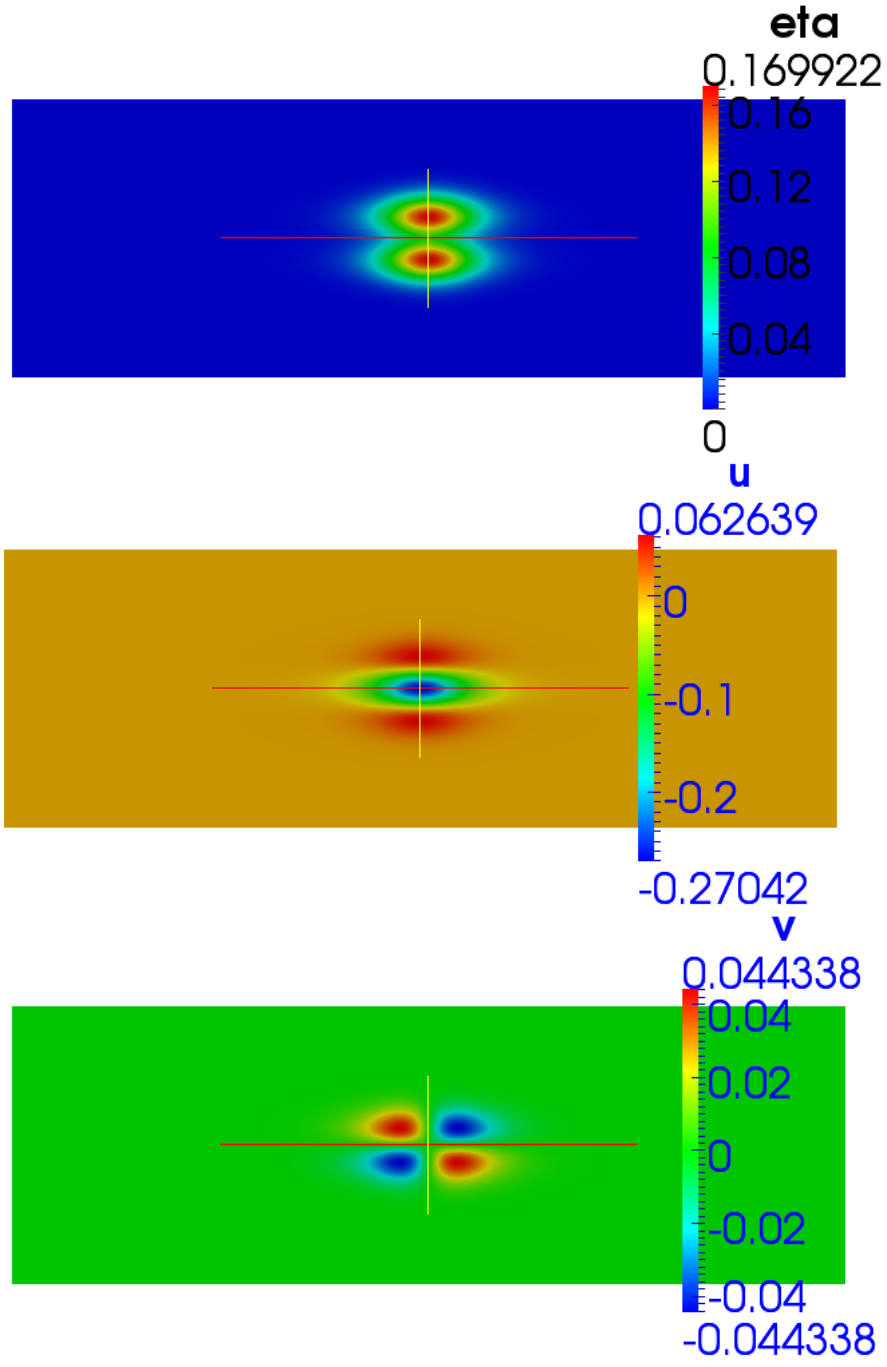

Figure 5.1: Initial conditions for the Rossby wave test of the SWEs; the top, the middle and the bottom panels show the three fields η, u and v, respectively.

is the number of elements and P is the order of approximation.

In order to demonstrate the exponential convergence of the scheme, we solved the equation on grids with different resolutions (here $N_{el} \in \{18, 72, 288, 1152\}$), and different order of the polynomial expansions (here $P \in \{2, 3, 4, 5, 7, 9\}$). Measurements of the L_2 and L_∞ errors for the η field are made at time $t = 1$ second of each experiment. The time step is chosen to be sufficiently small, 0.001, so the temporal error is negligible compared to spatial error from Runge-Kutta 4-th order explicit time scheme.

Tables (5.1) - (5.6) show the error results while fixing the order of the expansion to $2, 3, 4, 5, 7$ and 9 respectively for each different element size. Tables (5.8) and (5.7) show the error results while fixing the size of the element mesh and varying the degree of the polynomial expansion.

The illustration of the h-refinement is depicted in Figure 5.3 and 5.4, and the illustration of the p-refinement in Figure 5.5 and 5.6.

Numerical solutions of the problem at time $t = 1s$ for the field η for each different numerical fluxes are depicted in Figure 5.2.

Table 5.1: h-convergence error, by fixing the order of approximation to $P = 2$.

Norm	N_e	h	HLLC	Rusanov	Lax	Average
L_2	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.00187706	0.00142523	0.00142732	0.00232406
	1152	0.25	0.00017043	0.000177741	0.00017762	0.000214919
L_∞	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.252534	0.160658	0.161778	0.352784
	1152	0.25	0.0496415	0.0516206	0.0516284	0.0581733

Table 5.2: h-convergence error, by fixing the order of approximation to $P = 3$.

Norm	N_e	h	HLLC	Rusanov	Lax	Average
L_2	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.000557674	0.000638331	0.000639032	0.000927097
	1152	0.25	$4.75E-005$	$4.88E-005$	$4.88E-005$	$5.97E-005$
L_∞	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.183135	0.191328	0.191013	0.274734
	1152	0.25	0.0267934	0.0255858	0.0256101	0.0321325

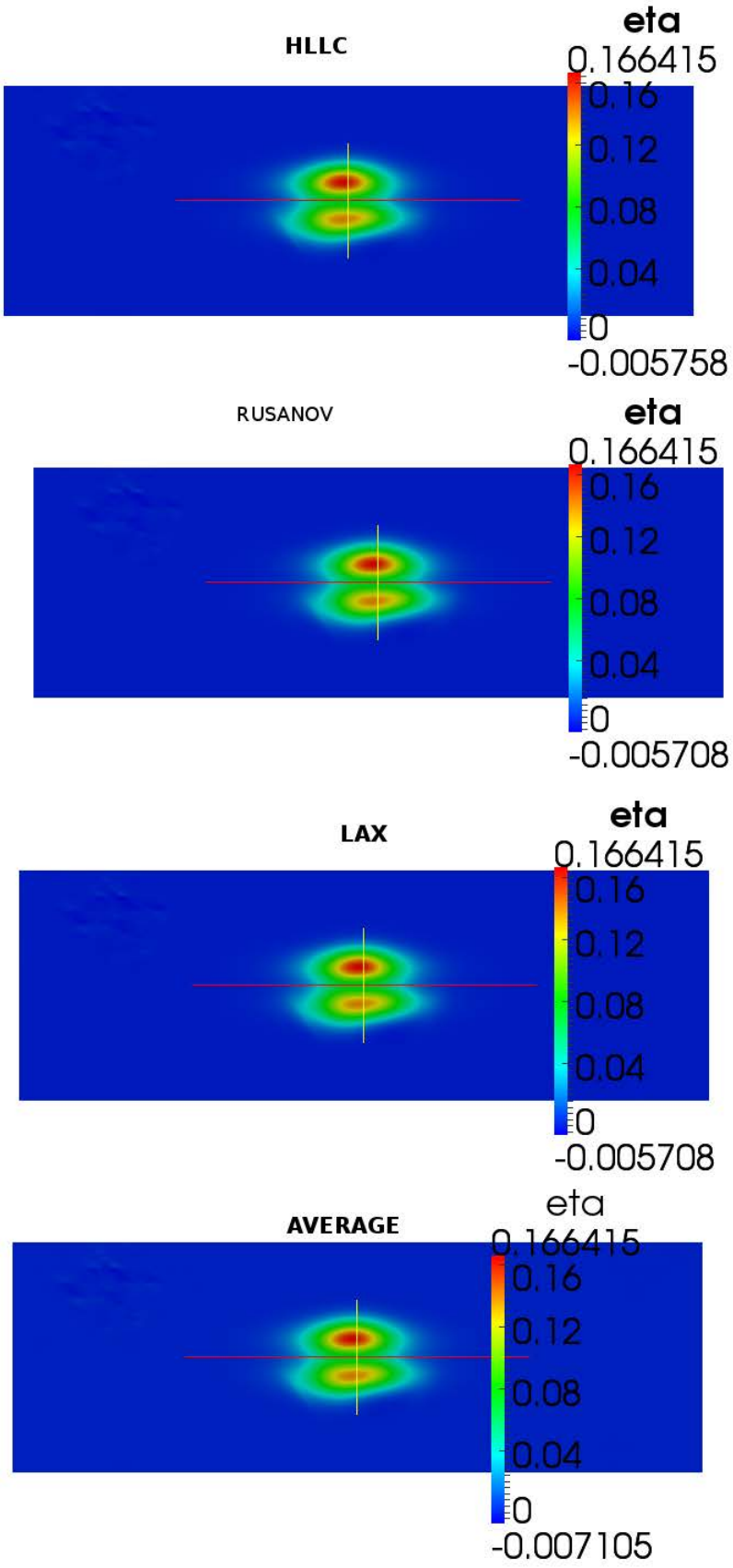

Figure 5.2: The field η at time $t = 1.0s$ for different fluxes (from the top to the bottom): HLLC, Rusanov, Lax and Average fluxes.

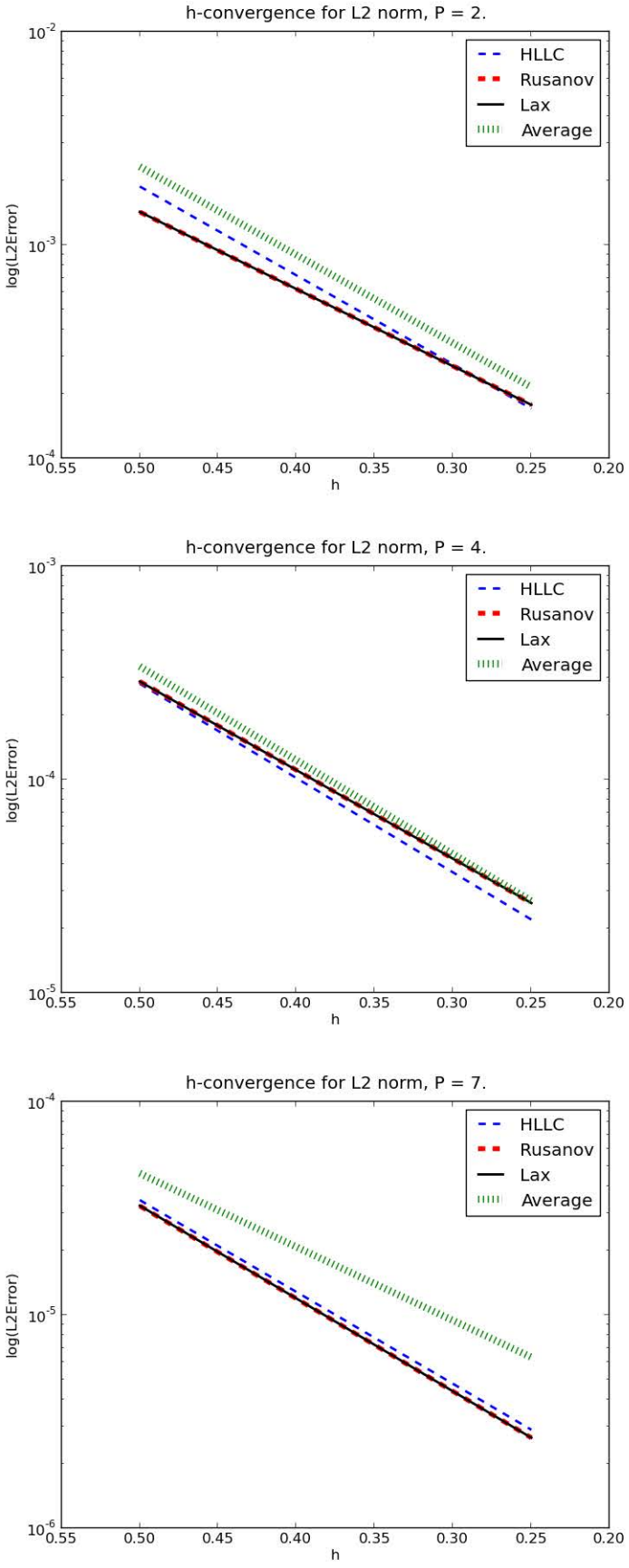

Figure 5.3: The logarithm of L_2 error as a function of the mesh size h. The top panel is for a fix degree of expansion $p = 2$, the middle panel for $p = 4$ and the bottom panel for $p = 7$.

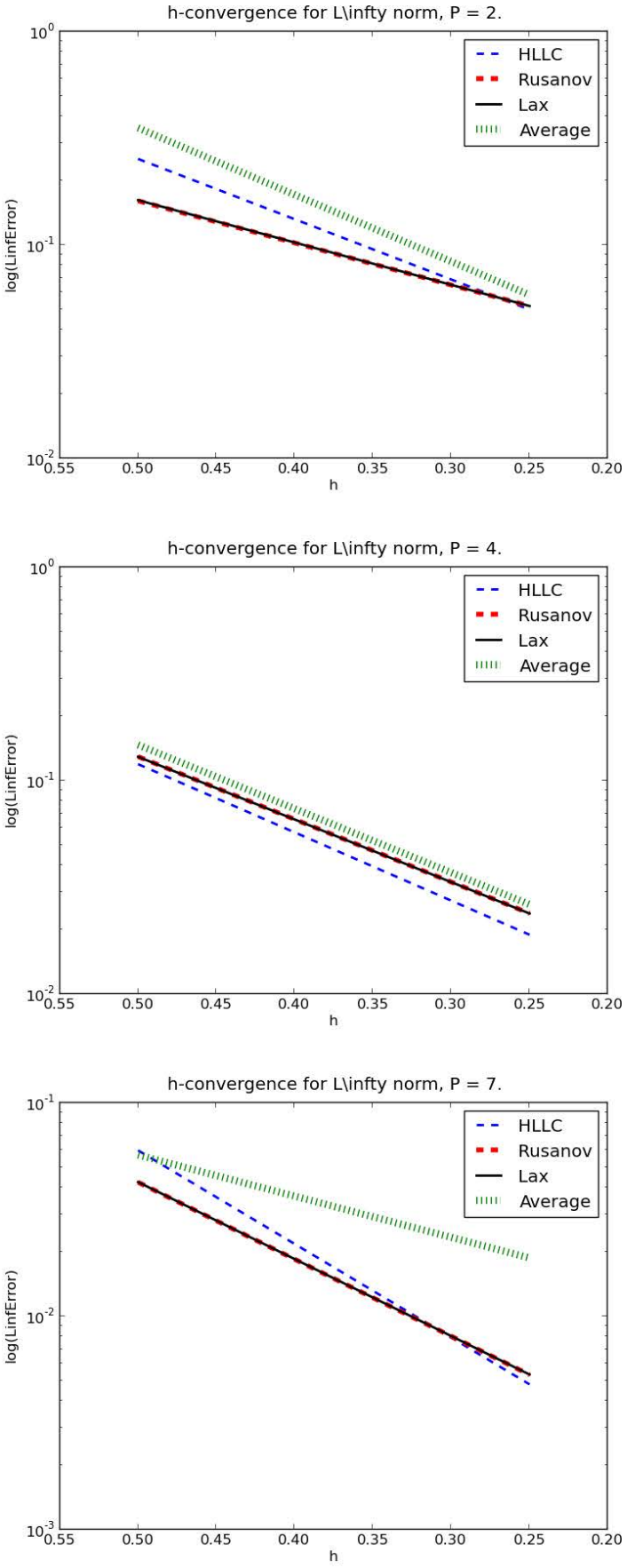

Figure 5.4: The logarithm of L_∞ error as a function of the mesh size h. The top panel is for a fix degree of expansion $p = 2$, the middle panel for $p = 4$ and the bottom panel for $p = 7$.

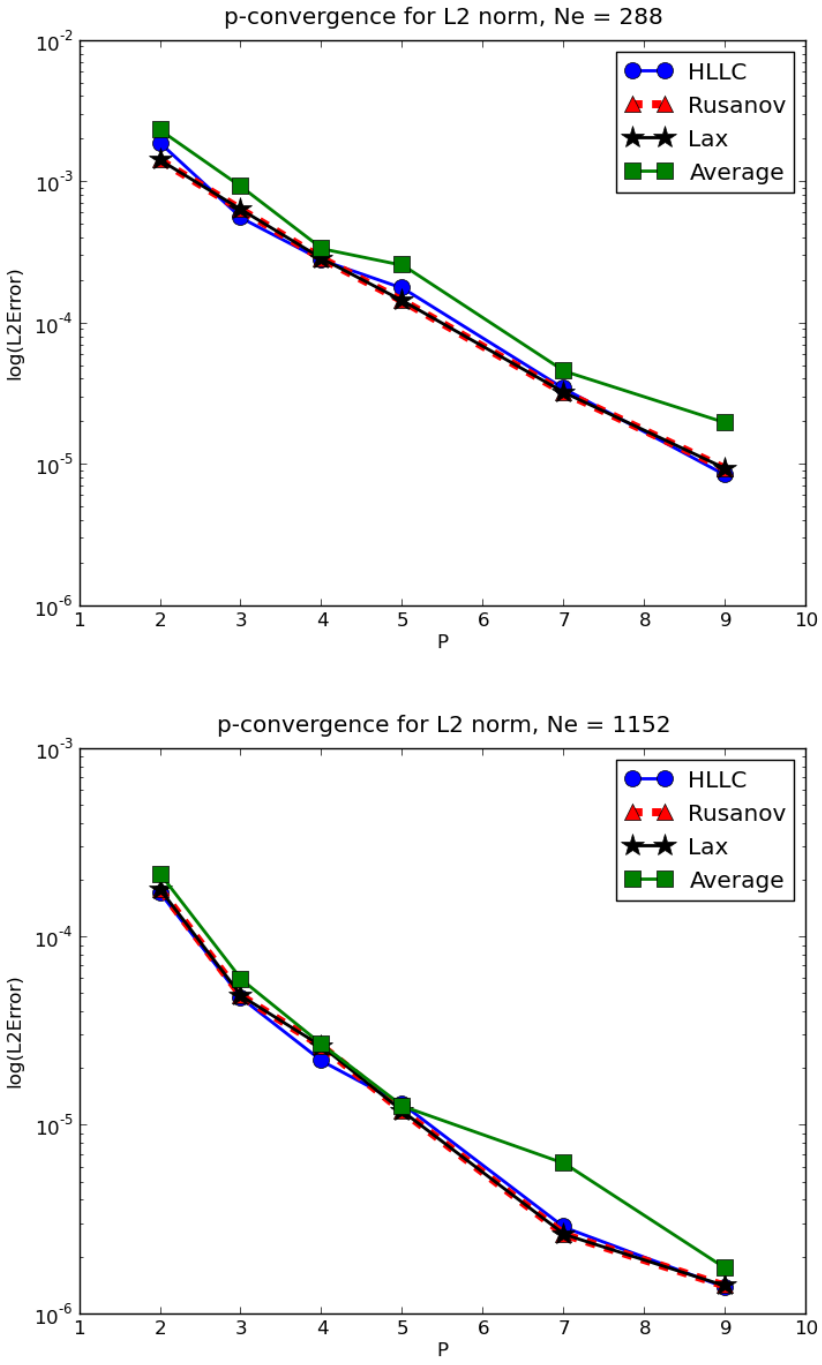

Figure 5.5: The logarithm of L_2 error as a function of the degree of the polynomial expansion p. The top panel is for the grid with $N_e = 288$ elements and the bottom panel is for the grid with $N_e = 1152$ elements.

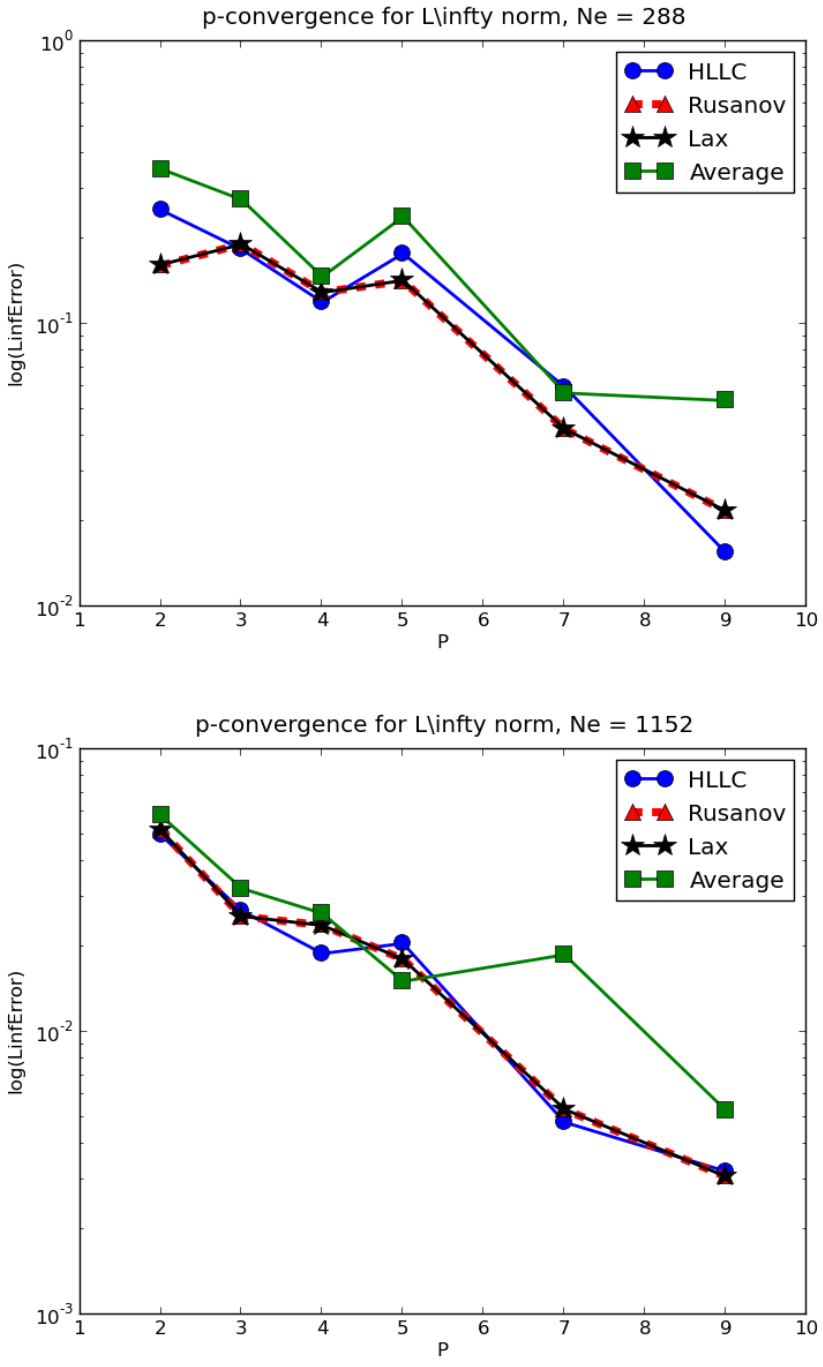

Figure 5.6: The logarithm of L_∞ error as a function of the degree of the polynomial expansion p. The top panel is for the grid with $N_e = 288$ elements and the bottom panel is for the grid with $N_e = 1152$ elements.

Table 5.3: h-convergence error, by fixing the order of approximation to $P = 4$.

Norm	N_e	h	HLLC	Rusanov	Lax	Average
L_2	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.00028082	0.000287734	0.000287622	0.000338441
	1152	0.25	$2.20E-005$	$2.63E-005$	$2.63E-005$	$2.69E-005$
L_∞	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.119201	0.129232	0.129047	0.146647
	1152	0.25	0.0189017	0.0238128	0.0237768	0.0262126

Table 5.4: h-convergence error, by fixing the order of approximation to $P = 5$.

Norm	N_e	h	HLLC	Rusanov	Lax	Average
L_2	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.000177699	0.000144341	0.000144636	0.000258629
	1152	0.25	$1.30E-005$	$1.19E-005$	$1.19E-005$	$1.27E-005$
L_∞	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.177217	0.141855	0.142247	0.239123
	1152	0.25	0.0205232	0.0180333	0.0180489	0.0150706

Table 5.5: h-convergence error, by fixing the order of approximation to $P = 7$.

Norm	N_e	h	HLLC	Rusanov	Lax	Average
L_2	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	$3.45E-005$	$3.24E-005$	$3.25E-005$	$4.60E-005$
	1152	0.25	$2.89E-006$	$2.65E-006$	$2.65E-006$	$6.32E-006$
L_∞	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.0597598	0.0423726	0.0424574	0.0568439
	1152	0.25	0.0047821	0.00530619	0.00531596	0.0187188

Table 5.6: h-convergence error, by fixing the order of approximation to $P = 9$.

Norm	N_e	h	HLLC	Rusanov	Lax	Average
L_2	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	$8.43E-006$	$9.39E-006$	$9.39E-006$	$1.98E-005$
	1152	0.25	$1.39E-006$	$1.41E-006$	$1.41E-006$	$1.75E-006$
L_∞	18	1	–	–	–	–
	72	0.75	–	–	–	–
	288	0.5	0.0154523	0.0217205	0.0217386	0.0534387
	1152	0.25	0.00320415	0.00306958	0.00306958	0.00526209

Convergence rate:

The speed at which a convergent sequence approaches its limit is called the rate of convergence. It is defined as follow:

$$rate = \frac{log[error_{h_{i+1}}/error_{h_i}]}{log(h_i/h_{i+1})}$$

Table 5.7: p-convergence error, by fixing the number of elements to $N_e = 288$.

Norm	P	HLLC	Rusanov	Lax	Average
L_2	2	0.00187706	0.00142523	0.00142732	0.00232406
	3	0.000557674	0.000638331	0.000639032	0.000927097
	4	0.00028082	0.000287734	0.000287622	0.000338441
	5	0.000177699	0.000144341	0.000144636	0.000258629
	7	0.0000344838	0.0000324409	0.0000324759	0.0000460498
	9	0.00000843244	0.00000938695	0.00000939476	0.0000198481
L_∞	2	0.252534	0.160658	0.161778	0.352784
	3	0.183135	0.191328	0.191013	0.274734
	4	0.119201	0.129232	0.129047	0.146647
	5	0.177217	0.141855	0.142247	0.239123
	7	0.0597598	0.0423726	0.0424574	0.0568439
	9	0.0154523	0.0217205	0.0217386	0.0534387

Table 5.8: p-convergence error, by fixing the number of elements to $N_e = 1152$.

Norm	P	HLLC	Rusanov	Lax	Average
L_2	2	0.00017043	0.000177741	0.00017762	0.000214919
	3	0.0000475131	0.0000488052	0.0000488093	0.000059725
	4	0.0000220014	0.0000263491	0.0000263301	0.0000269058
	5	0.0000129814	0.0000118904	0.0000118948	0.0000126872
	7	0.00000288951	0.00000265119	0.00000265301	0.00000631529
	9	0.00000138515	0.00000141025	0.0000014108	0.00000174784
L_∞	2	0.0496415	0.0516206	0.0516284	0.0581733
	3	0.0267934	0.0255858	0.0256101	0.0321325
	4	0.0189017	0.0238128	0.0237768	0.0262126
	5	0.0205232	0.0180333	0.0180489	0.0150706
	7	0.0047821	0.00530619	0.00531596	0.0187188
	9	0.00320415	0.00306958	0.00306958	0.00526209

For the h-convergence, the corresponding convergence rate is given in the following table:

Table 5.9: Convergence rate for the L_2-norm.

p	HLLC	Rusanov	Lax	Average
2	3.4612	3.0033	3.0064	3.4347
3	4.5530	3.7091	3.7106	3.9563
4	4.6739	4.4489	4.4493	4.6529
5	5.7749	5.6016	5.6040	5.3494
7	7.5776	7.6131	7.6136	6.8662
9	10.605	10.734	10.735	9.5053

5.4 Analysis

For fix values of the degree of the polynomial expansions, h-convergence exhibits exponential convergences for all four numerical fluxes both for L_2 and L_∞ norms as shown in Figure 5.3 and 5.4. Six different polynomial orders,

$p = 2, 3, 4, 5, 7, 9$, are used for the p-refinement, and exponential convergences are demonstrated for all the numerical fluxes in the L_2-norm, as illustrated in Figure 5.5. With less grid points of 18 and 72, the solutions blow-up (Tables 5.1-5.6). For the L_∞-norm, we can notice that convergence is better with fine grid (Figure 5.6). The use of averaging is known to produce sub-optimal convergence for odd p, as shown in Figure 5.6. The convergence rate table, Table 5.9, shows that the convergence is optimal, that is $\mathcal{O}(h^{p+1})$. The average flux is generally unstable for hyperbolic problems and cannot be used, even if the time step is small enough that the CFL condition is satisfied, Leveque (2004). The results of the Rusanov flux and the Lax-Friedrich flux seem identical as illustrated in the convergence plots. Physically, eigenvalues represent speeds of propagation of information. In addition, the use of the Lax-Friedrich flux is recommended by Bernard *et al.* (2009). The HLLC flux is also considered as an appropriate flux for the SWEs, and it is suggested in (Eskilsson and Sherwin, 2000).

Chapter 6

Conclusion

Any fluids that have horizontal length scale much greater than the vertical length scale can be considered as shallow water. Thus, the SWEs can be met more often since this characteristics is common in the area of fluid dynamics. Some examples of shallow water are the atmosphere of the Earth, dam break, river flood, tide, wave, tsunami and many others. The 2D SWEs are derived from the Navier-Stokes equations which describe the motion of fluids. The derivation and the mathematical expressions of the SWEs are reviewed in Chapter 2. The spectral/hp DG scheme is outlined in Chapter 3 for solving SWEs. The hallmark of the (DG) method is that solutions are allowed to be discontinuous over elemental boundaries, the elements are coupled by numerical fluxes, as in FVM and it is geometrically flexible, that is can handle complex spatial domain. The spectral/hp element method is the combination of the hp-version the finite element method and spectral element method. It uses the Lagrange polynomials as a basis of the expansion through the zeros of the Gauss-Lobatto-Legendre polynomials. The convergence of the method is then observed when fixing the size of the element mesh h and increasing the degree of the polynomial expansion, this is known as p-refinement as depicted in Table 5.7-5.8 and in Figure 5.3-5.4. Or by fixing the degree of the polynomial expansion p and decreasing the size of the element mesh h, this is known as h-refinement and illustrated in Table 5.1-5.6 and in Figure 5.5-5.6. Due to the discontinuity at the boundary, the use of numerical flux is needed. Numerical flux is the amount of information that passes at the boundary from one state to another state following the normal direction. There are different types of numerical fluxes, and they are computed using Riemann solvers. Four types of of numerical fluxes are presented in Chapter 4, which are HLLC, Lax-Friedrich, Rusanov and Average fluxes. They are all derived from approximate Riemann solvers and the HLL flux. But it turns out that the Lax-Friedrich and Rusanov fluxes are related, as we can see from the definition in Section 4.2 and from the computational results in Section 5.3.

To investigate the performance of the scheme and all the numerical fluxes, the nonlinear Rossby wave problem, is used as test cases. Applications to the

nonlinear Rossby wave problem exhibit the expected exponential convergences for the four numerical fluxes; and as stated previously, the errors (L_2 and L_∞) decrease as the size of the element mesh h decreases or as the degree of the polynomial expansion increases. The expected accuracy is achieved for all value of p, that is, $error \propto \mathcal{O}(h^{p+1})$. The Lax-Friedrich flux and the HLLC flux seems to be the best numerical flux, although HLLC is difficult to implement.

Appendices

Appendix A

Code

```
//=============================================================
// Case where the numerical flux includes the normal in the output
  void NonlinearSWE::NumericalFlux2D(Array<OneD, Array<OneD, NekDouble> > &physfield,
                                     Array<OneD, Array<OneD, NekDouble> > &numflux)
  {
    int i,k;
    int nTraceNumPoints = GetTraceTotPoints();
    int nvariables      = 3; // only the dependent variables

    // get temporary arrays
    Array<OneD, Array<OneD, NekDouble> > Fwd(nvariables);
    Array<OneD, Array<OneD, NekDouble> > Bwd(nvariables);
    Array<OneD, NekDouble>  DepthFwd(nTraceNumPoints);
    Array<OneD, NekDouble>  DepthBwd(nTraceNumPoints);

    for (i = 0; i < nvariables; ++i)
      {
        Fwd[i] = Array<OneD, NekDouble>(nTraceNumPoints);
        Bwd[i] = Array<OneD, NekDouble>(nTraceNumPoints);
      }

    // get the physical values at the trace
    for (i = 0; i < nvariables; ++i)
      {
        m_fields[i]->GetFwdBwdTracePhys(physfield[i],Fwd[i],Bwd[i]);
      }
    m_fields[0]->GetFwdBwdTracePhys(m_depth,DepthFwd,DepthBwd);

    Array<OneD, NekDouble> fluxX(nvariables);
    Array<OneD, NekDouble> fluxY(nvariables);
    Array<OneD, NekDouble> fluxAdd(nvariables);

    for (k = 0; k < nTraceNumPoints; ++k)
      {
        switch(m_upwindType)
          {
          case eAverage:
            {
              AverageFlux(Fwd[0][k]+DepthFwd[k],Fwd[1][k],Fwd[2][k],
                          Bwd[0][k]+DepthFwd[k],Bwd[1][k],Bwd[2][k],
                          fluxX, fluxY, fluxAdd);
            }
            break;
          case eLax:
            {
              LaxFriedrichFlux(Fwd[0][k]+DepthFwd[k],Fwd[1][k],Fwd[2][k],
                               Bwd[0][k]+DepthFwd[k],Bwd[1][k],Bwd[2][k],
                               fluxX, fluxY, fluxAdd);
            }
            break;
          case eRusanov:
            {
              RusanovFlux(Fwd[0][k]+DepthFwd[k],Fwd[1][k],Fwd[2][k],
                          Bwd[0][k]+DepthFwd[k],Bwd[1][k],Bwd[2][k],
                          fluxX, fluxY, fluxAdd);
            }
            break;
          default:
            {
              ASSERTL0(false,"populate switch statement for upwind flux");
            }
            break;
```

```
        }
      //Rotate back some terms to Cartesian
      numflux[0][k] = fluxX[0]*m_traceNormals[0][k] + fluxY[0]*m_traceNormals[1][k] + fluxAdd[0];
      numflux[1][k] = fluxX[1]*m_traceNormals[0][k] + fluxY[1]*m_traceNormals[1][k] + fluxAdd[1];
      numflux[2][k] = fluxX[2]*m_traceNormals[0][k] + fluxY[2]*m_traceNormals[1][k] + fluxAdd[2];
    }
  }

//============================================================================

//============================================================================
  void NonlinearSWE::AverageFlux(NekDouble hL,NekDouble uL,NekDouble vL,NekDouble hR,NekDouble uR,
                                 NekDouble vR, Array<OneD, NekDouble> &fluxX,
                                 Array<OneD, NekDouble> &fluxY,
                                 Array<OneD, NekDouble> &fluxAdd)
  {
    NekDouble g = m_g;

    fluxX[0]   = 0.5*(hL*uL + hR*uR);
    fluxY[0]   = 0.5*(hL*vL + hR*vR);
    fluxAdd[0] =  0.0;

    fluxX[1]   = 0.5*(hL*uL*uL + hR*uR*uR + 0.5*g*(hL*hL + hR*hR) );
    fluxY[1]   = 0.5*(hL*uL*vL + hR*uR*vR);
    fluxAdd[1] =  0.0;

    fluxX[2]   = 0.5*(hL*uL*vL + hR*uR*vR);
    fluxY[2]   = 0.5*(hL*vL*vL + hR*vR*vR + 0.5*g*(hL*hL + hR*hR) );
    fluxAdd[2] =  0.0;
  }

//============================================================================

//============================================================================
  void NonlinearSWE::RusanovFlux(NekDouble hL,NekDouble uL,NekDouble vL,NekDouble hR,NekDouble uR,
                                 NekDouble vR, Array<OneD, NekDouble> &numfluxX,
                                 Array<OneD, NekDouble> &numfluxY,
                                 Array<OneD, NekDouble> &numfluxAdd)
  {
    NekDouble g = m_g;

    NekDouble SL, SR;
    SL = fabs(uL) + sqrt(g*hL);
    SR = fabs(uR) + sqrt(g*hR);

    NekDouble S;
    if(SL > SR)
      S = SL;
    else
      S = SR;

    numfluxX[0]   = 0.5*(hL*uL + hR*uR);
    numfluxY[0]   = 0.5*(hL*vL + hR*vR);
    numfluxAdd[0] =  0.5*S*(hL - hR);

    numfluxX[1]   = 0.5*(hL*uL*uL + hR*uR*uR + 0.5*g*(hL*hL + hR*hR) );
    numfluxY[1]   = 0.5*(hL*uL*vL + hR*uR*vR);
    numfluxAdd[1] =  0.5*S*(uL*hL - uR*hR);

    numfluxX[2]   = 0.5*(hL*uL*vL + hR*uR*vR);
    numfluxY[2]   = 0.5*(hL*vL*vL + hR*vR*vR + 0.5*g*(hL*hL + hR*hR) );
    numfluxAdd[2] =  0.5*S*(vL*hL - vR*hR);
  }

//============================================================================


//============================================================================
  void NonlinearSWE::LaxFriedrichFlux(NekDouble hL,NekDouble uL,NekDouble vL,NekDouble hR,NekDouble uR,
                                      NekDouble vR, Array<OneD, NekDouble> &numfluxX,
                                      Array<OneD, NekDouble> &numfluxY,
                                      Array<OneD, NekDouble> &numfluxAdd)
  {
    int j;
    NekDouble g = m_g;
    NekDouble lambda;

    Array<OneD, NekDouble>Eigenvalue(6);

    //Define the element of the eigenvalue
    Eigenvalue[0] = uL - sqrt(g*hL);
    Eigenvalue[1] = uL;
    Eigenvalue[2] = uL + sqrt(g*hL);
    Eigenvalue[3] = uR - sqrt(g*hR);
    Eigenvalue[4] = uR;
    Eigenvalue[5] = uR + sqrt(g*hR);

    //Find the maximum eigenvalue
    lambda = Eigenvalue[0];
    for (j = 1; j < 6; ++j)
      {
```

```
        if (Eigenvalue[j] > lambda)
          lambda = Eigenvalue[j];
      }
    lambda = fabs(lambda);

    numfluxX[0]   = 0.5*(hL*uL + hR*uR);
    numfluxY[0]   = 0.5*(hL*vL + hR*vR);
    numfluxAdd[0] =  0.5*lambda*(hL - hR);

    numfluxX[1]   = 0.5*(hL*uL*uL + hR*uR*uR + 0.5*g*(hL*hL + hR*hR) );
    numfluxY[1]   = 0.5*(hL*uL*vL + hR*uR*vR);
    numfluxAdd[1] =  0.5*lambda*(uL*hL - uR*hR);

    numfluxX[2]   = 0.5*(hL*uL*vL + hR*uR*vR);
    numfluxY[2]   = 0.5*(hL*vL*vL + hR*vR*vR + 0.5*g*(hL*hL + hR*hR) );
    numfluxAdd[2] =  0.5*lambda*(vL*hL - vR*hR);
  }

//=============================================================

/*HLLC flux*/
  void NonlinearSWE::RiemannSolverHLLC(NekDouble hL,NekDouble uL,NekDouble vL,NekDouble hR,NekDouble uR,
                                       NekDouble vR, NekDouble &hflux, NekDouble &huflux,NekDouble &hvflux )
  {
    NekDouble g = m_g;
    NekDouble hC,huC,hvC,SL,SR,hstar,Sstar;
    NekDouble cL = sqrt(g * hL);
    NekDouble cR = sqrt(g * hR);

    // the two-rarefaction wave assumption
    hstar = 0.5*(cL + cR) + 0.25*(uL - uR);
    hstar *= hstar;
    hstar *= (1.0/g);

    // Compute SL
    if (hstar > hL)
      SL = uL - cL * sqrt(0.5*((hstar*hstar + hstar*hL)/(hL*hL)));
    else
      SL = uL - cL;

    // Compute SR
    if (hstar > hR)
      SR = uR + cR * sqrt(0.5*((hstar*hstar + hstar*hR)/(hR*hR)));
    else
      SR = uR + cR;

//     if (fabs(hR*(uR-SR)-hL*(uL-SL)) <= 1.0e-15)
//       Sstar = 0.0;
//     else
//       Sstar = (SL*hR*(uR-SR)-SR*hL*(uL-SL))/(hR*(uR-SR)-hL*(uL-SL));
                Sstar = 0.5*(uL+uR)+cL-cR;

    if (SL >= 0)
      {
        hflux  = hL * uL;
        huflux  = uL * uL * hL + 0.5 * g * hL * hL;
        hvflux  = hL * uL * vL;
      }
    else if (SR <= 0)
      {
        hflux  = hR * uR;
        huflux  = uR * uR * hR + 0.5 * g * hR * hR;
        hvflux  = hR * uR *vR;
      }
//     else
//       {
        else if ((SL < 0) && (Sstar >= 0))
          {
            hC  = hL * ((SL - uL) / (SL - Sstar));
            huC = hC * Sstar;
            hvC = hC * vL;

            hflux = hL*uL + SL * (hC - hL);
            huflux = (uL*uL*hL+0.5*g*hL*hL) + SL * (huC - hL*uL);
            hvflux = (uL*vL*hL) + SL * (hvC - hL*vL);
          }
        else if ((Sstar <=0) && (SR > 0))
          {
            hC  = hR * ((SR - uR) / (SR - Sstar));
            huC = hC * Sstar;
            hvC = hC * vR;

            hflux = hR*uR + SR * (hC - hR);
            huflux = (uR*uR*hR+0.5*g*hR*hR) + SR * (huC - hR*uR);
            hvflux = (uR*vR*hR) + SR * (hvC - hR*vR);
          }
        else
       ASSERTL0(false,"Error in HLLC solver");
  }
```

Appendix B

Theorems, Definitions and Mathematical rules

Theorem 1 (Gauss's theorem)**.** *Suppose V a subset of $\mathbb{R}^2$ which is compact and has a piecewise smooth boundary S. If $\mathbf{F}$ is a continuously differentiable vector field defined on a neighbourhood of V, then we have:*

$$\int_V (\nabla \cdot \mathbf{F})\, \mathrm{d}V = \oint_S (\mathbf{F} \cdot \mathbf{n})\, \mathrm{d}S\,.$$

Theorem 2 (Leibniz integral rule)**.** *Let $f(x,t)$ be a function such that both f and its partial derivative $\frac{\partial f}{\partial t}$ are continuous in t and x in some region of the (x,t)-plane, including $a(t) \leq x \leq b(t), t_0 \leq t \leq t_1$. Also suppose that the functions $a(t)$ and $b(t)$ are both continuous and both have continuous derivatives for $t_0 \leq t \leq t_1$. Then for $t_0 \leq t \leq t_1$:*

$$\frac{d}{dt}\left(\int_{a(t)}^{b(t)} f(x,t)\, \mathrm{d}x\right) = f(a(t),t)b'(t) - f(b(t),t)a'(t) + \int_{a(t)}^{b(t)} \frac{\partial f}{\partial t}\, \mathrm{d}x\,.$$

Definition 1 (Cauchy stress tensor)**.** *Tensors are geometric objects that describe linear relations between vectors, scalars, and other tensors, used to represent correspondences between sets of geometric vectors. The Cauchy stress tensor is a second order tensor of type $(1,1)$ (that is, a linear map), with nine components σ_{ij} that completely define the state of stress at a point inside a material in the deformed placement or configuration.*

List of References

(). Nektar++, spectral/*hp* element framework.
Available at: `http://www.nektar.info/`

Aizinger, V. and Dawson, C. (2002). A discontinuous galerkin method for two-dimensional flow and transport in shallow water. *Elsevier*, vol. 25, pp. 67–84.

Barth, T. and Deconinck, H. (eds.) (2003). *High-order Methods for Computational Physics*. Springer.

Bernard, P.-E., Remacle, J.-F., Comblen, R., Legat, V. and Hillewaert, K. (2009). High-order discontinuous galerkin schemes on general 2d manifolds applied to the shallow water equations. *J. Comput. Phys.*

Boyd, J. (1980 November). Equatorial solitary waves. part 1: Rossby solitons. *Journal of Physical Oceanography*, vol. 10, pp. 1699–1717.

Dawson, C. and Mirabito, C.M. (2009). The shallow water equations.

Eskilsson, C. (2011). An hp-adaptive discontinuous galerkin method for shallow water flows. *Int. J. for Num. Meth. in Fluids*, vol. 67, pp. 1605–1623.

Eskilsson, C., El-Khamra, Y., Rideout, D., Allen, G., Chen, Q. and Tyagi, M. (2009). A parallel high-order discontinuous galerkin shallow water model. *Springer-Verlag Berlin Heidelberg*, pp. 63–72.

Eskilsson, C. and Sherwin, S. (2000). A triangular spectral/*hp* discontinuous galerkin method for modelling 2d shallow water equations. *Int. J. for Num. Meth. in Fluids*.

Goncalves, E. (2008 Novembre). Resolution numerique des equations d'euler monodimensionnelles.

Harten, A., Lax, P. and Leer, B.V. (1983 Janvier). On upstream differencing and godunov-type schemes for hyperbolic conservation laws. *SIAM*, vol. 25, no. 1, pp. 35–61.

Karniadakis, G.E. and Sherwin, S. (eds.) (2005). *Spectral/hp element Method for Computational Fluid dynamics*. Oxford University Press.

Kong, C. (2011 May). *Comparison of Approximate Riemann Solvers*. Masters thesis, University of Reading.

Leveque, R.J. (ed.) (2004). *Finite-Volume Methods for Hyperbolic Problems*. Cambridge University Press.

Nair, R.D., Thomas, S.J. and Loft, R.D. (2005 April*a*). A discontinuous galerkin global shallow water model. *Scientific Computing Division, Numerical Center for Atmospheric Research, Boulder, Colorado*, vol. 133, pp. 876–888.

Nair, R.D., Thomas, S.J. and Loft, R.D. (2005 April*b*). A discontinuous galerkin transport scheme on the cubed sphere. *Scientific Computing Division, Numerical Center for Atmospheric Research, Boulder, Colorado*, vol. 133, pp. 814–828.

Randall, D.A. (2006). The shallow water equations.

Schwab, C. (ed.) (1998). *p- and hp- Finite Element Methods, theory and Applications in solid and fluid mechanics*. Numerical Mathematics and Scientific computation. Clarendon Press, Oxford University.

Schwanenberg, D., Liem, R. and Kongeter, J. (2000). Discontinuous galerkin method for shallow water equations. *Institute of Hydraulic Engineering and Water Resources Management (IWW), University of Technology, Kreuzherrentraβe, 52056 Aachen, Germany.*

Suli, E. (ed.) (2011). *Finite Element Methods For Partial Differential Equations*. Mathematical Institute University of Oxford.

Toro, E. (). *Shock-Capturing Methods for Free Surface Flows*.

Toro, E. (ed.) (2009). *Riemann Solvers and Numerical Methods for Fluid Dynamics*. Springer Dordrecht Heidelberg London New York.

Williamson, D.L., Drake, J.B., Hack, J.J., Jakob, R. and Swartztrauber, P.N. (1992). A standard test set for numerical approximation to the shallow water equations in spherical geometry. *J. Comput. Phys.*, vol. 102, pp. 211–224.

Zanotti, O. and Manca, G. (2010 February). A very short introduction to gudunov methods.