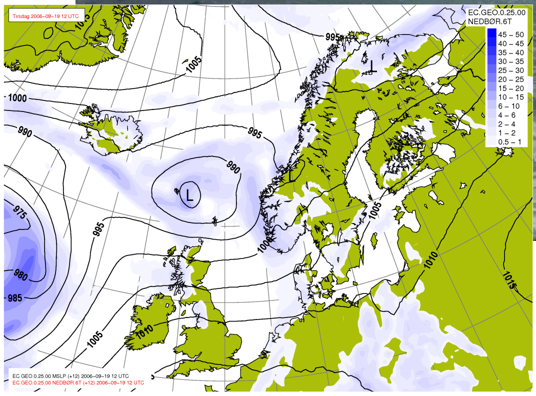
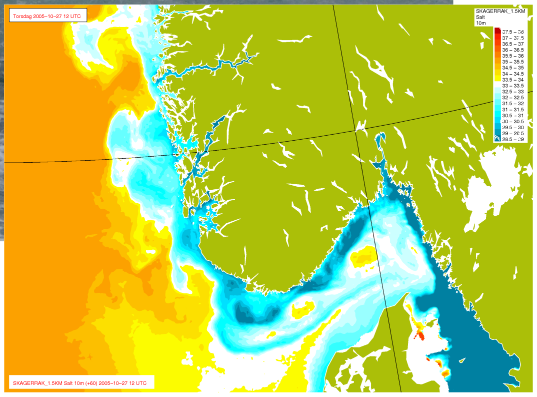
Atmospheres and Oceans on Computers: Fundamentals

Lars Petter Røed[†,⋆,‡]





[†]Research and Development Department, Norwegian Meteorological Institute, Postboks 43 Blindern, 0313 Oslo, Norge.
[⋆]Department of Geosciences, Section Meteorology and Oceanography, University of Oslo, P. O. Box 1022 Blindern, 0315 Oslo, Norway
[‡]E-mail: *larspr@met.no*

Date printed: October 25, 2016

PREFACE

This text is about how to put the atmosphere and the ocean on computers. To this end we use numerical methods, which is a fascinating contemporary tool. In fact it is the only tool that allow us to solve the fully nonlinear, partial differential equations (PDEs) that govern the motion of the atmosphere and the ocean. It is therefore hardly surprising that atmospheric scientists and weather forecasters alike embraced this tool early on.

It all started in 1946 when the mathematician John von Neumann, a well known Professor at the Princeton University, approached the meteorologist Carl-Gustav Rossby[1] to organize a "Conference on Meteorology". The idea was to acquaint the meteorological community with the new electronic computers ENIAC[2] and IAS[3] Machine, and to solicit their advice and support in designing research strategies. The outcome of the conference was the Princeton Meteorological Project (1947-53) which was managed by Dr. Jule G. Charney[4]. Among the participants were two young Norwegians, namely Ragnar Fjørtoft[5] and Arnt Eliassen[6]. The project successfully ended with producing daily numerical weather predictions in less than two hours. The very first attempt of producing a numerical weather forecast was published in 1950 by *Charney et al.* (1950).

The Meteorology Project marked the start of the science field referred to as Numerical Weather Prediction (NWP). An important basis for the rapid development of NWP in the 1950s and 1960s was the deterministic paradigm stated by Vilhelm Bjerknes[7] at the turn of the century (*Bjerknes*, 1904). In his famous 1904 paper Bjerknes stated that "*If it is true, what most scientific persons think, that the atmospheric state at any time can be developed from its earlier state using physical laws, then it follows that the necessary and sufficient condition for a rational solution to the problem of weather forecasting is a sufficiently accurate knowledge of the present atmospheric state, and a sufficiently accurate knowledge of the equations that govern the development*

[1]Carl-Gustaf Arvid Rossby (December 28, 1898 - August 19, 1957) was a Swedish-born American meteorologist who was the first to explain the large-scale motions of the atmosphere in terms of fluid mechanics. He identified and characterized both the jet stream and the long waves in the westerlies that were later named Rossby waves.

[2]Electronic Numerical Integrator And Computer

[3]Institute of Advanced Study

[4]Jule Gregory Charney (January 1, 1917 - June 16, 1981) was an American meteorologist. As part of his PhD work he (in 1947) developed a set of equations for calculating the large-scale motions of planetary-scale waves (The Quasi-Geostrophic Vorticity Equation). He gave the first convincing physical explanation for the development of mid-latitude cyclones known as the Baroclinic Instability theory.

[5]Ragnar Fjørtoft (August 1, 1913 - May 28, 1998) was an internationally recognized Norwegian meteorologist. He was part of a Princeton, New Jersey team that in 1950 performed the first successful numerical weather prediction using the ENIAC electronic computer. He was also a professor of meteorology at the University of Copenhagen and director of the Norwegian Meteorological Institute.

[6]Arnt Eliassen (September 9, 1915 - April 22, 2000) was a Norwegian meteorologist who was a pioneer in the use of numerical analysis and computers for weather forecasting. His early pioneer work was done at the Institute for Advanced Study in Princeton, New Jersey, together with John von Neuman. He received the Carl-Gustaf Rossby Research Medal in 1964 and the very prestigious Balzan Prize in 1996 "For his fundamental contributions to dynamic meteorology that have influenced and stimulated progress in this science during the past fifty years".

[7]Vilhelm Friman Koren Bjerknes (March 14, 1862 - April 9, 1951) was a Norwegian physicist and meteorologist who did much to found the modern practice of weather forecasting.

of the atmosphere from one state to the next.". Another important basis was the later attempt by Lewis Fry Richardson[8] to compute a 6 hour weather forecast by hand. He did this by first casting the governing equations into finite difference form using numerical methods (*Richardson*, 1922). Afterwards, while serving with the Quaker ambulance unit in northern France during World War I, he solved the finite difference equations using only pen and pencil.

Although Bjerknes did not mention it his statement above is also true regarding forecasting the oceanic "weather", that is, the growth and fate of meanders, jets and eddies, the latter being the ocean's high and low pressure systems. The oceanic lows and highs are however much smaller than their atmospheric counterparts. Hence the power and capacity of the early computers where too low to make oceanic forecasts in the 1950s and early 1960s. However, as the computer power and capacity grew[9] it became common to use computers and numerical methods to solve the eqautions governing the oceanic motion as well. Thereby a science field called Numerical Ocean Weather Prediction (NOWP) opened up in the late 1960s. In fact, the capacity and power of today's computers are amenable to forecast the oceanic weather, although not globally, at least for limited areas. In passing it is interesting to note that NWP and NOWP are among the major science fields pushing the computer technology to its very limits.

Inherently the atmosphere and the ocean forms a coupled system exchanging momentum, heat and moisture. This was early on recognized in climate modeling, and hence the very first climate models were coupled atmosphere-ocean models (*Edwards*, 2011). With the ever growing capacity of computers coupled atmosphere-ocean models are today also developed with NWP/NOWP in mind (e.g., *Warner et al.*, 2010). In this endevaour also sea ice and wave models are taken into account. Thus in the not too distant future fully coupled models will probably be the common prediction tool to make weather forecasts of the two spheres in one sweep. In light of this anyone aspiring to become a meteorologist, an oceanographer or a climatologist must have a solid knowledge and insight into the fundamental methods used to develop sound numerical methods to solve oceanographic and meteorological problems.

As alluded to, most processes in the ocean and atmosphere are highly nonlinear. Hence more and more research within NWP ad NOWP relies on sometimes large and monstrous computer codes. It is a growing concern that as a rule many of these codes are written and amended by scientists who are not necessarily skilled programmers. This concern is corroborated by the statistical survey by *Hannay et al.* (2009) who concludes that "*the knowledge required to develop and use scientific software is primarily acquired from peers and through self-study, rather than from formal education and training*". The codes may therefore, even though the numerical methods employed are sound, be rather poorly written from a skilled programmer's point of view. Only rarely do these codes undergo rigorous testing. Hence the model codes may inadvertently contain errors that may potentially be damaging to the results.

Another serious concern is that computers always produce results in terms of numbers. These numbers may even look reasonable, but in reality be totally false due to use of unsound numerical methods. Such results may even lead to wrong conclusions. In fact there exist examples in the

[8]Lewis Fry Richardson (October 11, 1881 - September 30, 1953) was an English mathematician, physicist, meteorologist, psychologist and pacifist who pioneered modern mathematical techniques of weather forecasting.

[9]The growth in computer power and capacity is almost exponential since the 1940s.

literature were the numerical solution is interpreted as a new physical phenomenon that later is shown to be a pure artifact produced by employing an incorrect numerical method. It is therefore important to understand why some methods are sound and some unsound for a specific problem. Likewise it is important to acquire knowledge of the quality of the computations. In light of these concerns an appendix is added that gives insight into procedures whereby the quality of a specific model may be assured (cf. Appendix B).

Although many of the numerical methods we apply were historically first developed and applied to solve atmospheric problems, these methods also works to solve oceanographic problems. The reason is that the *dynamics* of the two spheres are very similar. Within a numerical context it is therefore no need to treat meteorology and oceanography separately. In particular this is true regarding the more fundamental methods. A further rationale is the fact, as already mentioned, that the atmosphere and the ocean is inherently a coupled system.

The objective of these Lecture Notes is to give insight into the *fundamental* numerical methods to solve oceanographic and atmospheric problems. It should be kept in mind though that there are numerical methods and techniques unique to each sphere. In particular this concern numerical methods relevant to solve the *thermodynamic* part of the two spheres. The treatment of these particulars is beyond the scope of these Notes.

Finally, we emphasize that the application of numerical methods to solve atmospheric and oceanographic problems to a large degree is a “hands-on-experience”. It is of no use to learn the how to turn the governing equations into sound finite difference equatons, sometimes referred to as numeric algorithms, without learning how to develop a set of computer instructions in accord with our algorithms, run them on the computer, and visualize the results. The first part concerns what is called *programming*. It involves writing the instructions, or “the model code”, in some “language” that the computer understands. The common programming language used today in most atmospheric and ocean modeling is FORTRAN. To give insight into the fundamentals of the language a brief introduction to FORTRAN programming is attached in Appendix A. In addition the first lectures are devoted to learn basic FORTRAN programming. To familiarize oneself with programming in FORTRAN the reader is encouraged to solve as many as possible of the computer problems contained in the accompanying “Computer problems”. For the same reason the reader is also encouraged to solve the exercises given at the end of each chapter.

These Lecture Notes are based on the 2015 version of the Lecture Notes for the course GEF4510 at the Meteorology and Oceanography Section, Department of Geosciences, University of Oslo, Norway. The Notes will be revised, amended and upgraded as we go along, so the final version will not be ready before the end of fall 2016. A list of revisions appears on page ix.

Blindern August 20, 2016
Lars Petter Røed (sign.)

Acknowledgements

I would like to extend my greatest appreciation and sincere thanks to Dr. Martin Mork, former professor of oceanography at the University of Bergen for introducing me to oceanography in the early 1970s and to Dr. James J. O'Brien, former director of the Center for Atmosphere-Ocean Predictions (COAPS), and now distinguished professor emeritus at the Florida State University, USA for introducing me to numerical methods to solve oceanographic problems. Some of the material covered is actually based on notes taken when I followed his lectures at Florida State University more than 35 years ago (1980/81).

I would also like to thank Dr. Thor Erik Nordeng, former senior scientist at the Norwegian Meteorological Institute and former professor II at the Department of Geosciences, University of Oslo for helping me compiling material on atmospheres on computers. In this respect I also extends my gratitude to Dr. Arne Bratseth former professor at the University of Oslo now deceased. Some of the material covered is to a large degree influenced by his lecture notes on "Numerical Atmosphere Models". I am also indebted to Gunnar Wollan, former scientific programmer at Department of Geosciences, University of Oslo for letting me include a modified version of his notes entitled "A Fortran 2003 introduction by examples" as an Appendix.

Last, but not least, my appreciations also goes to the many students who has pointed out misprints and errors in earlier versions over the years.

Revision history (most recent on top):

All revisions made by the author.

- ☞ August 8, 2016: Upgraded Preface and Acknowledgement sections

Contents

List of Figures

Chapter 1

INTRODUCTION

Numerical methods is one of the most fascinating contemporary tools to solve meteorological and oceanographic problems. The reason is that most, if not all, of the processes in the atmosphere and ocean are governed by a set of highly non-linear partial differential equations (PDEs). Equally important is that computers allow us to solve this set of PDEs, once they are cast into a suitable algorithm using numerical methods, within a reasonable time frame. We emphasize that solving the PDEs numerically on computers is the only method whereby the full non-linear equations governing the motion of the atmosphere and ocean can be solved. Only in special cases, mostly cases in which the governing equations are reduced to being linear, is it possible to solve the them by analytic means. Hence to enable us to get an insight into the non-linear processes in the atmosphere and ocean, and to forecast their states, the only viable method is to solve the governing equations using numerical methods, that is, to put the atmosphere and ocean on computers.

We concern ourselves with the fundamental tools needed to understand how we put oceans and atmospheres on computers. Specifically we limit ourselves to study methods whereby some important balance equations in oceanography and meteorology, namely the advection-diffusion equation and a simplified form of the shallow water equations on a rotating earth, can be solved by numerical means. To this end we make use of *finite difference methods*. Assuming that the reader has little or no prior knowledge of or experience in solving differential equations numerically, we therefore explain the finite difference methods in detail.

The advection-diffusion equation and the shallow water equations belongs to a class of equations known as PDEs. Consequently we include in the preliminary chapter (Chapter 2) a rather detailed account on how various types of partial differential equations relates to the advection-diffusion equations and the shallow water equations.

Moreover, to motivate the reader, and for later reference purposes, we first show how the advection-diffusion equation and the shallow water equations relates to the full equations governing the motion of the atmosphere and ocean. This necessitates a recapitulation of the governing equations, the boundary conditions and the basic approximations commonly made in meteorology and oceanography. We therefore continue this introductory chapter by deriving the shallow water equations from the full governing equations, highlighting the necessary assumptions and approximations needed to derive them. This also conveniently introduces the notation used throughout the text.

1.1 The governing equations

In the atmosphere and ocean the most prominent dependent variables are the three components u, v, and w of the three-dimensional velocity $\mathbf{v}$, pressure p, density ρ, and (potential) temperature θ[1,2]. For the atmosphere also humidity q and cloud liquid water content q_L must be included, while the salinity s must be included among the prominent variables in the ocean. To determine these unknowns we need an equal number of equations. These equations are normally referred to as the governing equations since they govern the motion of the two spheres atmosphere and ocean.

Of the variables above only the velocity is a vector. The remaining variables, commonly referred to as the state variables, are all scalars. The state variables, except density and pressure, are all examples of what is referred to as tracers. Other examples of tracers are any dissolved chemical component or substance. Since the salinity, temperature and humidity influence the motion via the pressure forcing they are commonly referred to as *active* tracers. Tracers that do not influence the motion, like for instance dissolved chemical components, are referred to as *passive* tracers.

As is common when making a mathematical formulation of a physical problem, the governing equations are developed based on conservation principles, in our case the conservation of mass, momentum, internal energy and tracer content. For the atmosphere and ocean the governing equations in their non-Boussinesq form are[3]

$$\begin{aligned} \partial_t \rho + \nabla \cdot (\rho \mathbf{v}) &= 0, && (1.1)\\ \partial_t(\rho \mathbf{v}) + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) &= -2\rho \boldsymbol{\Omega} \times \mathbf{v} - \nabla p + \rho \mathbf{g} - \nabla \cdot (\rho \boldsymbol{\mathcal{F}}_M), && (1.2)\\ \partial_t(\rho C_i) + \nabla \cdot (\rho C_i \mathbf{v}) &= -\nabla \cdot (\rho \mathbf{F}_i) + \rho S_i \quad i = 1, 2, \cdots, && (1.3)\\ \rho &= \rho(p, C_1, C_2, \cdots). && (1.4) \end{aligned}$$

Here we use ∂_t, ∂_x, ∂_y, and ∂_z to denote partial differential with respect to the respective subscript. Thus $\partial_t \rho$ is the time derivative (or time rate of change) of the density. C_i represents the concentration of any tracer including potential temperature and humidity (atmosphere only) and salinity (ocean only), the tensor $\boldsymbol{\mathcal{F}}_M$ and vector $\mathbf{F}_i$ represents fluxes due to turbulent mixing of momentum and tracers, respectively, $\boldsymbol{\Omega}$ is the Earth's rotation rate, g is the gravitational acceleration and S_i is the tracer source, if any. Finally, we use ∇ to denote the three-dimensional del-operator defined by

$$\nabla = \mathbf{i}\partial_x + \mathbf{j}\partial_y + \mathbf{k}\partial_z, \qquad (1.5)$$

Of the above equations (1.1) is the conservation of mass, while (1.2) constitutes the conservation of momentum. Furthermore (1.3) is the tracer conservation equation, while (1.4) is the equation

[1]Velocity is normally referred to as wind in the atmosphere and current in the ocean.

[2]In the following bold upright fonts, e.g., $\mathbf{u}, \mathbf{v}$, are used to denote a vector, while bold special italic fonts, e.g., $\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}}$, are used to denote tensors

[3]See for example *Gill* (1982) or *Griffies* (2004)

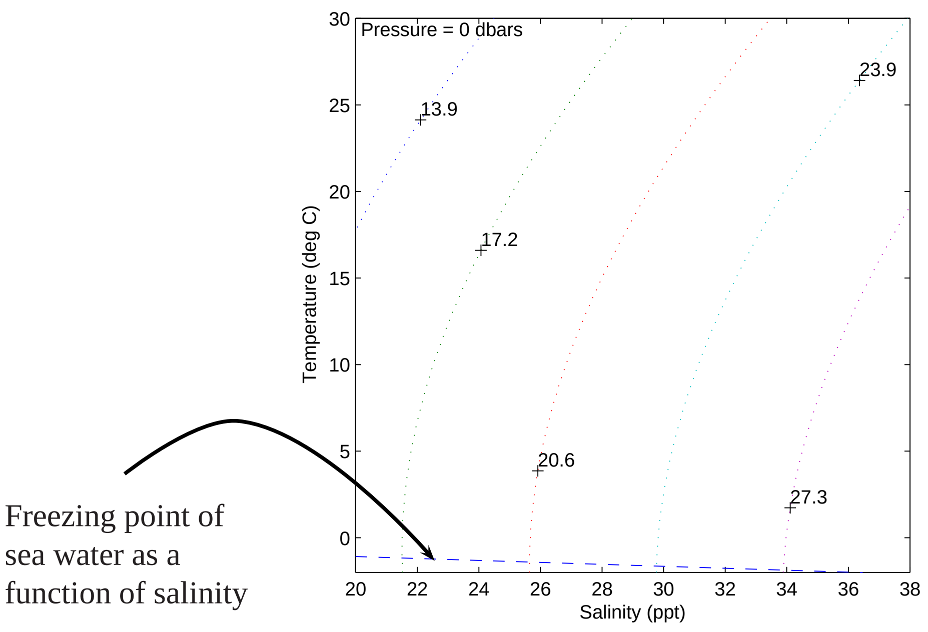


Figure 1.1: The equation of state for the ocean. Dotted curves show isolines of density ρ as a function of salinity (horizontal axis) and potential temperature (vertical axis) for a fixed pressure (here $p = 0$ Pa). Numbers on curves indicate denisty in σ_t units where $\sigma_t = \rho - 1000$ kg/m^3. Dashed line delineate the freezing point of sea water. Note that for temperatures close to the freezing point the density is almost a function of salinity alone, while the importance of temperature increases with increasing temperature. Due to the nonlinear nature of the equation of state of sea water two parcels of equal density may have different temperatures and salinities, for instance the two parcels marked A and B along $\sigma_t = 20.6$ kg/m^3.

of state. Thus the tracers figuring on the right-hand side of (1.4) are limited to active tracers only. Hence (1.4) relates density and pressure to the active tracers.

It should be noted that in the atmosphere the equation of state is linear and follows the ideal gas law, that is,

$$p = \rho R\theta \tag{1.6}$$

where R is the gas constant[4]. In contrast the equation of state for the ocean is highly non-linear. and cannot be expressed in a formal, closed form. We may visualize the equation of state for the ocean in a so called T-S (temperature-salinity) diagram where the salinity s is drawn along the horizontal axis and the (potential) temperature θ is drawn along the vertical axis. Since also pressure enter the equation of state a T-S diagram can only be constructed using a reference density. A typical example for $p = 0$ is displayed in Figure 1.1.

[4]R = 287.04 Jkg^{-1}K^{-1}

1.2 Boundary and initial conditions

We observe that to solve (1.1) - (1.4) we need to specify conditions at the spatial boundaries of the domain. Such conditions are referred to as *boundary conditions*. Furthermore we also need to specify the state of the ocean and/or atmosphere at some given time (cf. the statement quoted on page iii of *Bjerknes*, 1904). The latter is commonly referred to as the *initial conditions*.

The boundary conditions are of two major types, namely the *dynamic boundary conditions* and the *kinematic boundary conditions*. Normally the bounding surface of the volume containing the ocean or the atmosphere is a material surface. We recall that a material surface is a surface that consists of the same particles at all times. Thus the dynamic boundary conditions associated with a material surface requires that there is no acceleration at the surface, that is, that the the pressure and the fluxes must be continuous there. The kinematic boundary conditions at a material surface simply says that a particle once at the surface stays there forever.

As an example let us consider a system consisting of the atmosphere on top of the ocean. Let $\eta = \eta(x, y, t)$ denote the deviation of the atmosphere/ocean interface away from its equilibrium level at $z = 0$, and let $H = H(x, y)$ be the equilibrium depth of the ocean. Then the *kinematic* boundary condition at the interface is

$$w = \partial_t \eta + \mathbf{u} \cdot \nabla_H \eta \quad \text{at} \quad z = \eta \tag{1.7}$$

where $\mathbf{u}, w$ are, respectively, the horizontal and vertical component of the three-dimensional velocity $\mathbf{v}$, and where $\nabla_H = \mathbf{i}\partial_x + \mathbf{j}\partial_y$ is the horizontal component of the three-dimensional del-operator (1.5). The *dynamic* condition at the interface is

$$p_A = p_O, \quad \text{at} \quad z = \eta \tag{1.8}$$

where p_A denotes the atmospheric pressure, and p_O the oceanic pressure. The kinematic boundary condition at the bottom of the ocean is similar to (1.7), that is,

$$w = -\mathbf{u} \cdot \nabla_H H \quad \text{at} \quad z = -H, \tag{1.9}$$

where we assume that the bottom is stationary, that is, does not change in time. We have also assumed that the "bottom" $z = -H$ is a material surface. This surface is described by $S_H = z + H(x, y) = 0$, and hence (1.9) dictates that the bottom is impermeable or that there is no trough-flow across the bottom, that is, $\mathbf{n} \cdot \mathbf{v} = 0$ at $z = -H$, where $\mathbf{n} = \nabla S_H / |\nabla S_H|$ is the unit vector perpendicular to the bottom.

1.3 The hydrostatic approximation

In the atmosphere and ocean the horizontal scales of the dominant motions are large compared to the vertical scale. As a consequence the vertical acceleration, Dw/dt, is small in comparison to, e.g., the gravitational acceleration ρg. Consequently we replace the vertical momentum equation by the hydrostatic equation in which the gravitational acceleration is balanced by the vertical pressure gradient. When one solves this reduced system the model is said to be *hydrostatic*, and the motion satisfies *the hydrostatic approximation*.

To illustrate the hydrostatic approximation we first write the vertical component of the momentum equation using (1.2) in full, that is,

$$\partial_t(\rho w) + \nabla \cdot (\rho \mathbf{v} w) = -\partial_z p - \rho g - \nabla \cdot (\rho \mathbf{F}_M^V), \tag{1.10}$$

where $\mathbf{F}_M^V$ is the vertical vector component of the mixing tensor $\boldsymbol{\mathcal{F}}_M$ [5]. The hydrostatic assumption implies that the terms on the left-hand side of (1.10) are small compared to the gravitational acceleration and hence can safely be neglected[6]. Furthermore, since the vertical motion is small compared to the horizontal motion also the friction term may be neglected. Under these circumstances the vertical momentum equation reduces to

$$\partial_z p = -\rho g, \tag{1.11}$$

which is the *hydrostatic equation*[7]. When the hydrostatic approximation is valid we normally split the momentum equations into its vertical and horizontal components. The vertical component is then the hydrostatic equation (1.11). The two horizontal components are (in vector form)

$$\partial_t(\rho \mathbf{u}) + \nabla_H \cdot (\rho \mathbf{u}\mathbf{u}) + \partial_z(\rho w \mathbf{u}) + \rho f \mathbf{k} \times \mathbf{u} = -\nabla_H p + \partial_z \boldsymbol{\tau} - \nabla_H \cdot (\rho \boldsymbol{\mathcal{F}}_M^H), \tag{1.12}$$

where $f = 2\Omega \sin\phi$ is the Coriolis parameter where ϕ is the latitude and Ω is the Earth's rotation rate. $\boldsymbol{\mathcal{F}}_M^H$ and $\boldsymbol{\tau}$ are, respectively, the horizontal and vertical component of the three-dimensional flux tensor $\boldsymbol{\mathcal{F}}_M$ due to turbulent mixing. $\boldsymbol{\tau}$ is also commonly referred to as the vertical shear stress. Note that we in (1.12) have singled out the horizontal convective acceleration due to the vertical velocity and the vertical flux term due to turbulent mixing.

The tracer equation is left unchanged, but as in the momentum equation we may separate the turbulent mixing into one term associated with vertical mixing and one term associated horizontal mixing. Hence it may be written

$$\partial_t(\rho C_i) + \nabla_H \cdot (\rho C_i \mathbf{u}) + \partial_z(\rho C_i w) = -\partial_z(\rho F^V) - \nabla_H \cdot (\rho \mathbf{F}_i^H) + \rho S_i \quad i = 1, 2, \cdots, \tag{1.13}$$

where F^V and $\mathbf{F}_i^H$ are respectively the vertical and horizontal components of the turbulent mixing.

[5]We note that in a Cartesian coordinate system fixed to the Earth's surface the vertical component of the Coriolis force is small compared to the gravitational pull. The former is therefore dropped in 1.10.

[6]As noted this approximation relies on the fact that in most cases the dominant part of the motion, that is, the energetic part, lies in the long wavenumber band, and hence the horizontal scale is significantly longer than the vertical scale (consists of long waves in shallow water). Consequently, both the vertical velocity and its acceleration is small compared to the gravitational acceleration. The exceptions are cases that include steep topography and/or strong convection, in which cases one has to revert to non-hydrostatic equations.

[7]The name is used since a fluid at rest in the gravitational field satisfies exactly this equation. This is often referred to as a static fluid, hence the name hydrostatic.

1.4 The Boussinesq approximation

Another common approximation employed, particularly in *ocean models*, is the *Boussinesq approximation*. The fundamental basis for this approximation is that in many cases the dynamics of the atmosphere and in particular the oceans is independent of the fact that the atmospheres and oceans are compressible. Under these circumstance we can treat the motion as if the sphere is incompressible. This implies that any parcel of fluid conserves its volume, and that this is true even if the parcel is heated. Thus the Boussinesq approximation is only true as long as the change in density for any parcel of fluid is small with respect to the density itself, that is,

$$\frac{1}{\rho}\frac{D\rho}{dt} = \frac{D\ln\rho}{dt} \approx 0, \tag{1.14}$$

where the operator $\frac{D}{dt}$ is the material derivative[8], defined by

$$\frac{D}{dt} = \partial_t + \mathbf{v}\cdot\nabla. \tag{1.15}$$

Under the Boussinesq approximation the approximation (1.14) is taken as an equality. The mass conservation (1.1) then reduces to

$$\nabla\cdot\mathbf{v} = 0. \tag{1.16}$$

Use of an ocean model employing the Boussinesq approximation, a *Boussinesq ocean*, has one major disadvantage. One particularly pertinent example is the expected change in sea level, or ocean volume, under global warming. When uniformly heating the ocean the equation of state implies that the density decreases. For a non-Boussinesq ocean, which is mass conserving, the response to the decrease in density is to expand its volume. Hence the sea level rises. In contrast a Boussinesq ocean, which conserves volume, responds to heating by decreasing the density, that is, by loosing mass. Obviously the latter is highly unrealistic.

The reason why the Boussinesq approximation is still widely used in the ocean modeling community, despite the Boussinesq fluid's inability to expand due to heating, is the fact that it effectively filters out the acoustic waves while allowing us to retain pressure changes in response to density changes. To filter out the acoustic waves is advantageous in numerical perspective since. A will be shown below (Chapter 5) the time step is then not restricted by these very fast waves (see Section 4.3), dramatically decreasing the wall clock time (or CPU time) spent to perform even relatively short time integrations.

In summary the density under the Boussinesq approximation is treated as a constant except when it appears together with the gravitational acceleration. Under these circumstances the horizontal component of the momentum equation (1.12) becomes

$$\partial_t\mathbf{u} + \nabla\cdot(\mathbf{v}\mathbf{u}) + f\mathbf{k}\times\mathbf{u} = -\rho_0^{-1}\nabla_H p + \rho_0^{-1}\partial_z\boldsymbol{\tau} - \nabla_H\cdot(\boldsymbol{\mathcal{F}}_M^H), \tag{1.17}$$

where ρ_0 is a reference density. Similarly the tracer conservation equation (1.3) reduces to

$$\partial_t C_i + \nabla\cdot(C_i\mathbf{v}) = -\nabla\cdot\mathbf{F}_i + S_i \tag{1.18}$$

[8]Also referred to as the individual derivative

for $i = 1, 2, \cdots$.

We notice that it is quite common to combine the Boussinesq and the hydrostatic equations in meteorology and oceanography. Under these circumstances the governing equations reduce to

$$\nabla_H \cdot \mathbf{u} + \partial_z w = 0, \tag{1.19}$$

$$\partial_t \mathbf{u} + \nabla_H \cdot (\mathbf{uu}) + \partial_z (w\mathbf{u}) + f\mathbf{k} \times \mathbf{u} = -\rho_0^{-1} \nabla_H p + \rho_0^{-1} \partial_z \boldsymbol{\tau} - \nabla_H \cdot (\boldsymbol{\mathcal{F}}_M^H), \tag{1.20}$$

$$\partial_z p = -\rho g, \tag{1.21}$$

$$\partial_t C_i + \nabla_H \cdot (C_i \mathbf{u}) + \partial_z (C_i w) = -\partial_z F^V - \nabla_H \cdot \mathbf{F}_i^H + S_i \quad ; \quad i = 1, 2, \cdots, \tag{1.22}$$

together with the equation of state (1.4). We note that when applying the hydrostatic and Boussinesq approximation the vertical velocity component and the density are reduced to *diagnostic variables* just as pressure. This is in contrast to the horizontal velocity components $\mathbf{u}$ and the tracers C_i, e.g, potential temperature, which are *prognostic variables* in the sense that they are governed by *prognostic equations*, that is, equations containing a time rate of change term of the variable in question.

Finally we note that the introduction of more and more simplifications sometimes complicates the numerical problem. For instance the fairly popular rigid lid approximation implies that the equations must be solved globally rather than locally since the solution at one point not only depends on its nearest neighboring points, but in fact depends on all the points within the computational domain. This requires us to solve an *elliptic problem* for each time step, although the problem in itself, as a time marching problem is *hyperbolic*[9]. We will return to elliptic solvers in Section 4.11 on page 55 when solving an elliptic problem by a direct method.

1.5 The shallow water equations

A very common reduced set of equations in meteorology and oceanography is the so called shallow water equations. We may derive these equations from the full governing equation (1.1) - (1.4). We first assume that the hydrostatic and Boussinesq approximations are valid. Hence the starting point is mass conservation in the form (1.19), the momentum equations in the form (1.21) and (1.20), the tracer equation in the form (1.22) together with the equation of state (1.4). The additional assumption made is that the density is assumed to be uniform in time and space, i.e., $\rho = \rho_0$ where ρ_0 is a constant. We note that this makes the tracer equations (1.22) for the active tracers as well as the equation of state (1.4) obsolete. The resulting governing equations then reduces to

$$\begin{aligned}
\nabla_H \cdot \mathbf{u} + \partial_z w &= 0 && (1.23)\\
\partial_z p &= -\rho_0 g, && (1.24)\\
\partial_t \mathbf{u} + \nabla_H \cdot (\mathbf{uu}) + \partial_z (w\mathbf{u}) &= -f\mathbf{k} \times \mathbf{u} - \rho_0^{-1} \nabla_H p + \rho_0^{-1} \partial_z \boldsymbol{\tau} - \nabla_H \cdot \boldsymbol{\mathcal{F}}_M^H, && (1.25)
\end{aligned}$$

[9]For definitions of elliptic and hyperbolic problems see Sections 2.2 and 2.4 of Chapter 2

We note the assumption of a uniform density allows us to integrate (1.24) from any arbitrary height/depth z to a reference surface $z = \eta(x, y, t)$, viz.,

$$p = p_s + g\rho_0(\eta - z) \tag{1.26}$$

where p_s is the pressure at the reference surface. In the ocean the reference surface is commonly the surface of the ocean in which case η is the deviation of the sea surface from its equilibrium level $z = 0$. In the atmosphere it is common to let the reference surface be the surface of the Earth, e.g., $\eta = -H$, where H is measured as the distance from some fixed level (commonly set to $z = 0$).

Integrating (1.23) and (1.25) from the bottom $z = -H(x, y)$ to the top $z = \eta(x, y, t)$, and using the kinematic boundary conditions (1.7) and (1.9) and the dynamic boundary condition $p = 0$ at $z = \eta$ we get,

$$\begin{aligned}
\partial_t \mathbf{U} + \nabla_H \cdot (\frac{\mathbf{UU}}{h}) + f\mathbf{k} \times \mathbf{U} &= -gh\nabla_H(h - H) + \rho_0^{-1}(\boldsymbol{\tau}_s - \boldsymbol{\tau}_b) + \mathbf{X}, && (1.27)\\
\partial_t h + \nabla_H \cdot \mathbf{U} &= 0, && (1.28)
\end{aligned}$$

where $\mathbf{U} = \int_{-H}^{\eta} \mathbf{u}dz$ is the volume flux of fluid through the fluid column of height/depth $h = \eta + H$, $\boldsymbol{\tau}_s$ and $\boldsymbol{\tau}_b$ are, respectively, the turbulent vertical momentum fluxes at the top and bottom of the fluid column, and $\mathbf{X}$ is what is left of the horizontal momentum fluxes when integrated vertically from bottom to top. To arrive at (1.27) we have also integrated (1.24) from some arbitrary height/depth z to the top $z = \eta$. Furthermore we have used the fact that H is independent of time to replace, e.g., $\partial_t\eta$ by $\partial_t h$. Finally we have absorbed the term arising from the approximation

$$\int_{-H}^{\eta} \nabla_H \cdot (\mathbf{uu})dz \approx \nabla_H \cdot (\frac{\mathbf{UU}}{h}) \tag{1.29}$$

into the last term $\mathbf{X}$ on the right-hand side of (1.27). We commonly refer to (1.27) and (1.28) as the *shallow water equations*. Written in this form the shallow water equations are said to be written in flux form. We note that $\mathbf{U}$ is the total volume flux of fluid through the fluid column of height/depth h. Thus the mean depth average velocity is $\hat{\mathbf{u}} = \mathbf{U}/h$. Replacing $\mathbf{U}$ by $\hat{\mathbf{u}}$ the shallow water equations become

$$\begin{aligned}
\partial_t(h\hat{\mathbf{u}}) + \nabla_H \cdot (h\hat{\mathbf{u}}\hat{\mathbf{u}}) + f\mathbf{k} \times h\hat{\mathbf{u}} &= -gh\nabla_H(h - H) + \rho_0^{-1}(\boldsymbol{\tau}_s - \boldsymbol{\tau}_b) + \mathbf{X}, && (1.30)\\
\partial_t h + \nabla_H \cdot (h\hat{\mathbf{u}}) &= 0, && (1.31)
\end{aligned}$$

For later reference purposes we note that the acceleration terms $\partial_t(h\hat{\mathbf{u}}) + \nabla_H \cdot (h\hat{\mathbf{u}}\hat{\mathbf{u}})$ in (1.30) can be written

$$\begin{aligned}
\partial_t(h\hat{\mathbf{u}}) + \nabla_H \cdot (h\hat{\mathbf{u}}\hat{\mathbf{u}}) &= h\left(\partial_t\hat{\mathbf{u}} + \hat{\mathbf{u}} \cdot \nabla_H\hat{\mathbf{u}}\right) + \hat{\mathbf{u}}\left[\partial_t h + \nabla_H \cdot (h\hat{\mathbf{u}})\right]\\
&= h\left(\partial_t\hat{\mathbf{u}} + \hat{\mathbf{u}} \cdot \nabla_H\hat{\mathbf{u}}\right), && (1.32)
\end{aligned}$$

where we have used (1.31) to arrive at the last equal sign. Thus (1.30) and (1.31) is written

$$\begin{aligned}
\partial_t\hat{\mathbf{u}} + \hat{\mathbf{u}} \cdot \nabla_H\hat{\mathbf{u}} + f\mathbf{k} \times \hat{\mathbf{u}} &= -g\nabla_H\eta + \frac{\boldsymbol{\tau}_s - \boldsymbol{\tau}_b}{\rho_0(H + \eta)} + \frac{\mathbf{X}}{(H + \eta)}, && (1.33)\\
\partial_t\eta + \nabla_H \cdot [(H + \eta)\hat{\mathbf{u}}] &= 0, && (1.34)
\end{aligned}$$

We note that when the shallow water equations are written in their non-flux form, as displayed in (1.33) and (1.34), the mass conservation equation (1.34) becomes non-linear as well. This is in contrast to the mass conservation equation in flux form, that is, (1.28), which is linear.

1.6 The quasi-geostrophic equations

Another common set of reduced equations are based on quasi-geostrophic theory as for instance detailed in *Pedlosky* (1979) or *Stern* (1975). Here we essentially follow *Stern* (1975).

We first note that the starting point for the quasi-geostrophic equations are the governing equations employing the hydrostatic and Boussinesq approximations. Without loss of generality we may therefore start with the shallow water equations (1.33) and (1.34). If we neglect the forcing terms on the right-hand side of (1.33) we get

$$\frac{D_H \mathbf{u}}{dt} + f\mathbf{k} \times \mathbf{u} = -g\nabla_H h, \tag{1.35}$$

$$\frac{1}{h}\frac{D_H h}{dt} + \nabla_H \cdot \mathbf{u} = 0, \tag{1.36}$$

where we have dropped the circumflex for clarity. The notation D_H/dt is used to denote the two-dimensional version of the operator (1.15), that is,

$$\frac{D_H}{dt} = \partial_t + \mathbf{u} \cdot \nabla_H, \tag{1.37}$$

If we furthermore assume that the acceleration $D_H\mathbf{u}/dt$ is small compared to the Coriolis acceleration, that is, assume that the *Rossby number*

$$R \equiv \frac{|D_H \mathbf{u}/dt|}{|f\mathbf{k} \times \mathbf{u}|} \ll 1, \tag{1.38}$$

the momentum equation (1.35) reduces to

$$f\mathbf{k} \times \mathbf{u} = -g\nabla_H h. \tag{1.39}$$

We note that (1.39) is linear and describes a balance between the Coriolis term and the pressure term. Hence (1.39) is referred to as the geostrophic or *thermal wind equation* and the balance is called the *geostrophic balance*. Solving for the relative velocity we get

$$\mathbf{u} = \frac{g}{f}\mathbf{k} \times \nabla_H h. \tag{1.40}$$

The introduction of the Rossby number tells us that we may view the thermal wind equation as a first approximation to an expansion in terms of the Rossby number in which terms of $\mathcal{O}(R)$ or higher are neglected, that is,

$$\mathbf{u} = \frac{g}{f}\mathbf{k} \times \nabla_H h + \mathcal{O}(R). \tag{1.41}$$

We note that (1.39) obviously provides no information about the space-time variations in either the velocity field or the pressure field. To obtain information on those dynamics to order $\mathcal{O}(R)$ we have to look elsewhere. For instance we may obtain it from the relevant asymptotic form of the vorticity equation.

To derive the vorticity equation we start by defining the *relative vorticity*

$$\zeta = \mathbf{k} \cdot \nabla_H \times \mathbf{u}. \tag{1.42}$$

Then operating $\mathbf{k} \cdot \nabla_H \times$ on (1.35) and then substituting for $\nabla_H \cdot \mathbf{u}$ from (1.36) we get

$$\frac{D_H}{dt}\left(\frac{\zeta + f}{h}\right) = 0. \tag{1.43}$$

Here $\zeta + f$ is the absolute vorticity, while $(\zeta + f)/h$ is the *potential vorticity* for a barotropic fluid[10]. If we let L be a typical lateral (horizontal) scale of $\mathbf{u}$, so that $|\mathbf{u}\cdot\nabla_H\mathbf{u}| \sim |\mathbf{u}^2|/L$, then the necessary condition for $R \ll 1$, which is commonly referred to as the quasi geostrophy condition (cf. eq. 1.38), to be satisfied is

$$\frac{|\mathbf{u}|}{fL} \ll 1. \tag{1.44}$$

The condition is however not sufficient since the remaining acceleration term in D_H/dt is the local time rate of change $\partial_t\mathbf{u}$ which might be comparable to the Coriolis term $f\mathbf{k} \times \mathbf{u}$. Consequently we must additionally require that the initial condition is in geostrophic balance, that is, satisfies (1.39). The smallness of $\partial_t\mathbf{u}$ compared to the Coriolis acceleration then depends on the smallness of $\mathbf{u} \cdot \nabla_H\mathbf{u}$. Under these circumstances we may safely regard (1.44) as being the same as requiring $R = |\mathbf{u}|/fL \ll 1$. We may then compute the temporal evolution of the geostrophic field from the asymptotic vorticity equation.

To derive the asymptotic vorticity equation we first observe, by use of (1.42), that

$$\frac{|\zeta|}{f} = \frac{|\mathbf{u}|}{fL}. \tag{1.45}$$

Hence (1.44) requires that the relative vorticity is small compared to f by a factor of R. We also note that the variation in layer thickness h, obtained from (1.39) is

$$h - H_m \sim \frac{fL}{g}|\mathbf{u}| \quad \text{or} \quad \frac{h - H_m}{H_m} \sim RF^2, \tag{1.46}$$

where H_m is the mean layer thickness and

$$F \equiv \left(\frac{L^2}{L_R}\right)^{\frac{1}{2}}, \tag{1.47}$$

where $L_R = gH_m/f^2$ is the Rossby radius of deformation. If we now assume $F \sim \mathcal{O}(1)$ or less, which is tantamount to assuming that L is not large compared to Rossby's deformation radius,

[10]Recall that we have assumed that the density is constant. The fluid is therefore barotropic. The potential vorticity may also be derived for a baroclinic fluid in a similar fashion, but has then a different mathematical expression.

then the layer thickness variation in (1.46) is small to the same order as the ratio of the relative vorticity ζ to the planetary vorticity f, that is, ζ/f.

Under these circumstances we first rewrite the potential vorticity equation (1.43) to get

$$\frac{D}{dt}\left(\frac{\zeta+f}{h}\right)=\frac{f}{H_m}\left(\partial_t+\mathbf{u}\cdot\nabla_H\right)\left(\frac{1+\frac{\zeta}{f}}{1+\frac{h-H_m}{H_m}}\right)=0. \tag{1.48}$$

We are now in a position to expand this expression in terms of R, and thus we get

$$\left(\partial_t+\mathbf{u}\cdot\nabla_H\right)\left[1+\frac{\zeta}{f}-\frac{h-H_m}{H_m}+\mathcal{O}(R^2)\right]=\left(\partial_t+\mathbf{u}\cdot\nabla_H\right)\left(\frac{\zeta}{f}-\frac{h}{H_m}\right)+\mathcal{O}(R^3)=0. \tag{1.49}$$

The leading terms in (1.49) are $\mathcal{O}(R^2)$ since the (non-dimensional) magnitude of the acceleration terms $\partial_t\mathbf{u}$ and $\mathbf{u}\cdot\nabla_H\mathbf{u}$ are $\mathcal{O}(R)$. Thus the fractional error in the asymptotic vorticity equation

$$\left(\partial_t+\mathbf{u}\cdot\nabla_H\right)\left(\frac{\zeta}{f}-\frac{h}{H_m}\right)=0 \tag{1.50}$$

and in the asymptotic momentum equation (1.39) are both of $\mathcal{O}(R)$. Hence substitution of $\mathbf{u}$ from (1.40) wherever the latter appears in (1.50) the resulting differential equation for the layer thickness (or pressure) h is also asymptotic when $R \ll 1$. It is thus permissible to evaluate the velocity and the relative vorticity in (1.50) using the geostrophic equation (1.39) or (1.40). In fact if we substitute the expression (1.40) for $\mathbf{u}$ into (1.42) and then into (1.50) we first get

$$\zeta=\frac{g}{f}\nabla_H^2 h, \tag{1.51}$$

and then

$$\left[\partial_t+\frac{g}{f}(\mathbf{k}\times\nabla_H h)\cdot\nabla_H\right]\left(\nabla_H^2 h-L_R^{-1}h\right)=0. \tag{1.52}$$

Equation (1.51) and (1.52) together with the geostrophic equation (1.39) are commonly referred to as the *quasi-geostrophic equations* (QG equations). Thus we may use the quasi-geostrophic vorticity equation (1.52) to compute the pressure or layer thickness h at an arbitrary time $t>0$ from any initial distribution at time $t=0$. The resulting solution is then almost geostrophic, but not quite. Hence we use the name quasi-geostrophic. We emphasize that it is only under very stringent conditions, as explained, that these equations are valid.

We finally remark that, although each step in the hierarchy of the approximations, that is the Boussinesq approximation, the hydrostatic approximation, the shallow water equations, and finally the quasi-geostrophic approximation, removes or filters out a certain class of phenomena, the advantage of such procedures is that they allow us to isolate effects having different space-time scales. In a numerical contexts they are also very useful in establishing solutions against which our numerical solutions may be tested or verified.

Chapter 2

PRELIMINARIES

The equations that governs the motion of the atmosphere and the ocean, as well as the hierarchy of equations that follows employing the various approximations as outlined in the introductory chapter (Chapter 1), belongs to a class of equations called *partial differential equations* (henceforth PDEs). They differ from ordinary differential equations in that there is more than one independent variable, and sometimes several dependent variables.

In this chapter we learn more about PDEs and reveal that they have different characters depending on the physics they describe. We also introduce some basic mathematics underlying two of the most important numerical methods used to solve atmospheric and oceanic problems, namely *finite difference methods* and *spectral methods*. These mathematics include knowledge about Taylor series expansions, orthogonal functions, Fourier series and Fourier transforms. Finally, we include some notations that conveniently helps us to solve PDEs using numerical methods.

2.1 General PDEs

In general a PDE is written

$$\hat{a}\partial^2_{x'}\theta + 2\hat{b}\partial_{x'}\partial_{y'}\theta + \hat{c}\partial^2_{y'}\theta + 2\hat{d}\partial_{x'}\theta + 2\hat{e}\partial_{y'}\theta + \hat{f}\theta = \hat{g}. \tag{2.1}$$

Here $\partial_{x'}, \partial_{y'}$ denotes differentiation with respect to the independent variables x', y', while $\theta = \theta(x', y')$ denotes the dependent variable. The coefficients $\hat{a}, \hat{b}, .., \hat{g}$ are in general functions of the independent variables, that is, $\hat{a} = \hat{a}(x', y')$, etc. Note that x' and y' represents any independent variable, for instance time or one of the spatial variables, while θ represents any dependent variable, e.g., velocity, pressure, density, salinity, or humidity. We illustrate this by using the shallow water equations (1.33) and (1.34) as an example. To simplify the illustration we first neglect all forcing terms. We then linearize them by assuming that the deviation of the height h of a fluid column is small compared to its equilibrium depth H_m, that is, $(h - H_m)/H_m \ll 1$.

We then get

$$\begin{aligned}\partial_t u - fv &= -g\partial_x h && (2.2)\\ \partial_t v + fu &= -g\partial_y h && (2.3)\\ \partial_t h + H_m\partial_x u &= -H_m\partial_y v. && (2.4)\end{aligned}$$

Here u, v are the components of the horizontal velocity $\mathbf{u}$ along the axes x, y respectively. We may further simplify these equations by assuming that the motion is one-dimensional in space by letting $\partial_y = 0$. Mathematical speaking this is just three equation containing the three dependent variables u, v and h, and the two independent variables t, and x. We note that by some manipulation similar to that detailed in Section 2.4 on page 16 below these three equations may be condensed into one equation, that is,

$$\partial_t^2 h - gH_m\partial_x^2 h + f^2 h = 0, \tag{2.5}$$

containing one dependent variable h only. Furthermore, (2.5) conforms to (2.1) by letting $\theta = h$, $x' = t$, $y' = x$, $\hat{a} = 1$, $\hat{c} = -gH_m$, $\hat{f} = f^2$ and $\hat{b} = \hat{d} = \hat{e} = \hat{g} = 0$.

2.2 Elliptic equations

If $\hat{b}^2 - \hat{a}\hat{c} < 0$ then the roots of (2.1) are imaginary, distinct, and complex conjugated. The corresponding PDE is then *elliptic*. The classic example is *Poisson's equation*,

$$\nabla_H^2\phi = \partial_x^2\phi + \partial_y^2\phi = g(x, y), \tag{2.6}$$

where again ∇_H is the two-dimensional part of the three-dimensional del operator. We arrive at this equation by letting $\theta = \phi$, $x' = x$, $y' = y$, $\hat{a} = \hat{c} = 1$, $\hat{g} = g$ and $\hat{b} = \hat{d} = \hat{e} = \hat{f} = 0$ in (2.1). Other examples are the *Helmholtz equation*

$$\nabla_H^2\phi + f(x, y)\phi = g(x, y), \tag{2.7}$$

and the *Laplace equation*

$$\nabla_H^2\phi = 0. \tag{2.8}$$

2.3 Parabolic equations

If $\hat{b}^2 - \hat{a}\hat{c} = 0$ then the corresponding PDE is *parabolic*. The classic example is the *diffusion equation* or the heat conduction equation,

$$\partial_t\theta = \kappa\partial_x^2\theta, \tag{2.9}$$

where κ is the diffusion coefficient (heat capacity). To arrive at (2.9) from (2.1) we let $\theta = \theta$, $x' = x, y' = t, \hat{a} = 1, \hat{b} = \hat{c} = \hat{d} = \hat{f} = \hat{g} = 0$, and $\hat{e} = 1/2$. We observe that (2.9) is a simplified, one-dimensional version of the full three-dimensional tracer equation (1.3), where the advection term as well as the source and sink terms are neglected. In fact under the latter circumstances the three-dimensional tracer equation (1.18) for a Boussinesq fluid may be written

$$\partial_t C_i = \nabla \cdot (\boldsymbol{\mathcal{K}} \cdot \nabla C_i)\,, \tag{2.10}$$

where the diffusive tracer flux $\mathbf{F}_i$ is parameterized as $\mathbf{F}_i = -\boldsymbol{\mathcal{K}} \cdot \nabla C_i$ where in turn $\boldsymbol{\mathcal{K}}$ is a matrix (dyade) describing the conductive efficiency of the medium with regard to the tracer C_i (cf. Section 3.2). Thus $\boldsymbol{\mathcal{K}} = \kappa_{mn}\mathbf{i}_m\mathbf{j}_n$, $m, n = 1, 2, 3$. To retrieve (2.9) we simply let $\kappa_{11} = \kappa$ and $\kappa_{mn} = 0$ for $m \neq 1$ and $n \neq 1$ and assume that κ is constant.

Let us for a moment assume that the atmosphere/ocean is at rest ($\mathbf{v} = 0$) and that there are no sources or sinks for the tracer C_i ($S_i = 0$). Then (1.3) reduces to (2.10) implying that the diffusion balance is of fundamental importance when solving atmospheric and oceanographic problems .

2.4 Hyperbolic equations

If $\hat{b}^2 - \hat{a}\hat{c} > 0$ then the roots of (2.1) are real and distinct. The corresponding PDE is then *hyperbolic*. The classic example is the wave equation,

$$\partial_t^2\phi - c_0^2\partial_x^2\phi = 0. \tag{2.11}$$

To derive (2.11) from (2.1) we let $\theta = \phi$, $x' = t$, $y' = x$, $\hat{a} = 1$, $\hat{b} = 0$, $\hat{c} = -c_0^2$, and $\hat{d} = \hat{e} = \hat{f} = \hat{g} = 0$. Then $\hat{b}^2 - \hat{a}\hat{c} = -(-c_0^2) = c_0^2$ which is indeed positive.

We note that by defining

$$\Phi = \partial_t\phi - c_0\partial_x\phi \tag{2.12}$$

we may rewrite the wave equation (2.11) to get

$$\partial_t\Phi + c_0\partial_x\Phi = 0. \tag{2.13}$$

Since c_0 is a constant (2.13) may be written

$$\partial_t\Phi + \partial_x(c_0\Phi) = 0. \tag{2.14}$$

We observe that (2.14), commonly referred to as the *advection equation*, is a one-dimensional version of (1.1) with ρ replaced by Φ and $\mathbf{v}$ replaced by $c_0\mathbf{i}$. It is also a one-dimensional version of (1.3) with suitable replacements. Thus the advection equation is of fundamental importance in the modeling of atmospheric and oceanographic motions. It also indicates that the equations governing atmospheric and oceanographic motions viewed as time marching problems are inherently hyperbolic.

We also notice that the two-dimensional version of the shallow water equations (1.27) and (1.28) is inherently a hyperbolic problem. To illustrate this we start with the two-dimensional, linearized version of the shallow water equation, that is, (2.2), (2.3) and (2.4). We start by manipulating (2.2) and (2.3) to find u, v as functions of h. To this end we first differentiate (2.2) with respect to time, and then add (2.3) multiplied by f. This results in an equation containing h and u only, that is,

$$(\partial_t^2 + f^2)u = -gf\partial_y h - g\partial_t\partial_x h. \tag{2.15}$$

Similarly by first differentiate (2.3) with respect to time and then adding (2.2) multiplied by $-f$ gives an expression relating h and v, that is,

$$(\partial_t^2 + f^2)v = gf\partial_x h - g\partial_t\partial_y h. \tag{2.16}$$

Next we substitute the results into (2.4) to get

$$(\partial_t^2 + f^2 - gH_m\nabla_H^2)\partial_t h = 0. \tag{2.17}$$

Let $h = H_m + h'$ and let $h' = 0$ at time $t = 0$. Integration in time t then yields

$$(\partial_t^2 + f^2 - gH_m\nabla_H^2)h' = 0. \tag{2.18}$$

If we in addition assume that the motion is independent of one of the dependent variables, say y, we get

$$(\partial_t^2 + f^2 - gH_m\partial_x^2)h' = 0. \tag{2.19}$$

We note that (2.19) is hyperbolic in t and x. Similarly we observe that (2.18) is elliptic in x and y. Thus, we note that although the steady state solution to (2.19) is elliptic, the time marching problem is inherently hyperbolic.

The governing equations describing the time evolution of atmospheric and oceanographic motions are fundamentally hyperbolic. It is important to keep this in mind when developing numerical methods to solve atmosphere-ocean problems.

We will return to the shallow water equations in Section 6.1 on page 96. There we use them to show how we should treat the Coriolis term, that is, the term that makes geophysical fluid dynamics, like oceanographic and atmospheric problems, stand out from ordinary fluid dynamics. We also conveniently us it as an example problem to show how multiple variable problems are solved using numerical methods.

2.5 Boundary conditions

To solve for the dependent variables we have to integrate the governing PDE in time and space. Thus the solution inherently contains integration constants. The number of integration constants is determined by the order of the PDE. For instance upon integration the linearized shallow water equations (2.2) - (2.4) in time t gives three integration constants, while integration in space (x, y) gives another four integration constants (two in x and two in y), a total of seven. Thus we need seven conditions to determine these constants. These conditions are commonly referred to as *boundary conditions*.

We emphasize that the number of boundary conditions needed must be exactly the same as the number of integration constants, no more, no less. If we specify too many boundary conditions the system is overspecified, and if we specify too few we end up with an underspecified system. It is therefore imperative that we adhere to this fundamental principle when we make use of numerical methods to solve our governing equations. We emphasize that the computer *always* produce numbers. If we over- or underspecify our system, the computer will still produce numbers. These numbers may even look realistic or correct, but are nevertheless incorrect. The reason is that the only way to ensure that our solution exists and is unique is to have an equal number of boundary conditions and integration constants. Furthermore, as a corollary, the solution to our problem is equally dependent on the boundary conditions as on any other forcing.

To determine for instance the solution to the elliptic Poisson equation (2.6) we need four boundary conditions, two in x and two in y. To determine the solution to the diffusion equation (2.9) we need three boundary conditions, two in x and one in time t. Finally, to determine the solutions to the wave equation we need a total of four conditions to determine the four integration constants, namely two in t and two in x. As we increase the dimensions of the equation we note that the number of integration constants increases and thus also the number of boundary conditions needed.

There are essentially two types of boundary conditions belonging to the class of *natural boundary conditions*[1], namely

- Dirichlet conditions,

in which case the variable is known at the boundary, and

- Neuman conditions,

in which case the derivative normal to the boundary is specified. Most other boundary conditions are just combinations of these.

In Section 1.2 we mentioned that there cannot be any flow through an impermeable wall, that is, no throughflow across a solid wall, and formulated this condition as

$$\mathbf{n} \cdot \mathbf{v} = 0 \tag{2.20}$$

at the wall surface. Here $\mathbf{n}$ denotes the unit vector perpendicular to the wall. In fact this is a classic example of the Dirichlet type boundary condition in that specifying the condition $\mathbf{n} \cdot \mathbf{v} = 0$ at the surface constituting the wall is tantamount to specifying the variable itself at the surface. Thus the condition is of the Dirichlet type.

Next we may derive a classic example of a Neuman type condition by the condition prevailing at an insulated wall. The natural condition dictated by the physics is that for the wall to be insulated there cannot be any heat exchange across the boundary. Thus the diffusive flux of heat through the boundary must be zero. In mathematical terms this is tantamount to

$$\mathbf{n} \cdot \mathbf{F}_\theta = 0, \tag{2.21}$$

[1]A natural boundary condition is one in which the condition is dictated by the physics. This is in contrast to open boundaries treated in Chapter 7.

where $\mathbf{F}_\theta = -\kappa\nabla\theta$ is the diffusive heat flux vector. Thus (2.21) is the same as specifying the *gradient* (in this case a zero gradient) at the boundary. Thus the (2.21) is of the Neuman type.

As alluded to the two conditions may be combined to give other natural boundary conditions. One is the so called *Cauchy condition* or "slip" condition. For instance consider a flat bottom or surface at $z = -H$ (or $z = 0$) at which we give the following condition

$$\nu\partial_z\mathbf{u} = C_D\mathbf{u} \quad ; \quad z = -H, \tag{2.22}$$

where ν is the vertical eddy viscosity, $\mathbf{u}$ is the horizontal component of the current (or wind), and C_D is a drag coefficient (more often than not the latter is a constant). We note that since (2.22) does not specify either the gradient nor the variable itself. Thus if $\mathbf{u}$ is the horizontal velocity component then (2.22) requires that $\mathbf{u}$ is nonzero if the gradient is nonzero (and vice versa). Hence the name slip condition.

Other common boundary conditions are *cyclic* or *periodic boundary conditions*. A periodic boundary condition is one in which the solution is specified to be periodic in space, that is, that the solution repeats itself beyond a certain distance. Thus a periodic boundary condition in x for a given tracer concentration $C(x)$ would be

$$C(x,t) = C(x+L,t), \tag{2.23}$$

where L is the distance over which the solution repeats itself, for instance the wavelength in a monochromatic wave. Such conditions are commonly in use when solving problems where the atmosphere or ocean is considered to be contained in a zonal channel bounded to the south and north by a zonal wall. In the longitudinal direction the solution is then dictated by physics to naturally repeat itself every 360 degrees.

2.6 Taylor series and expansions

The basis for all numerical finite difference methods is that all "good" functions can be expanded in terms of a Taylor series. A good function is simply one for which the function itself and all its first and higher order derivatives exist and are continuous[2]. As referred to one characteristic of a good function is that it can always be expanded in a Taylor series. Another is that it can be represented by an infinite sum of orthogonal functions such as for instance trigonometric function (Sections 2.10 and 2.11).

The Taylor series of any good function, say $\phi(x)$, is defined as[3]

$$\phi|_{x+\Delta x} = \phi|_x + \sum_{n=1}^{\infty}\frac{1}{n!}(\partial_x^n\phi)|_x\Delta x^n, \tag{2.24}$$

[2]Note that this definition is slightly different from the one offered in the little known but enlightening book by M. J. Lighthill entitled "Good functions" (*Lighthill*, 1970)

[3]The notation $\phi|_x$ is used to denote evaluation of the function ϕ at the point x.

where $\Delta x > 0$ denotes a positive increment in space. We note that such an expansion is independent of whether the function ϕ depends on more than one variable. Hence[4]

$$\phi|^t_{x+\Delta x,y,z} = \phi|^t_{x,y,z} + \sum_{n=1}^{\infty} \frac{1}{n!}(\partial^n_x \phi)|^t_{x,y,z}\Delta x^n, \tag{2.25}$$

Similarly we also note that we may expand the function ϕ in any of the other independent variables, e.g.,

$$\phi|^{t+\Delta t}_{x,y,z} = \phi|^t_{x,y,z} + \sum_{n=1}^{\infty} \frac{1}{n!}(\partial^n_t \phi)|^t_{x,y,z}\Delta t^n, \tag{2.26}$$

For instance consider the function $\theta(x, t)$ to be a good function in space x and time t. Then we know $\theta(x, t)$ and all its first and higher order derivatives with respect to x at a particular point in space, say $x = x$. We may then use the Taylor series expansion (2.25) to find the values of θ at the neighboring point $x + \Delta x$. Thus,

$$\theta|^t_{x+\Delta x} = \theta|^t_x + \partial_x\theta|^t_x\Delta x + \frac{1}{2}\partial^2_x\theta|^t_x\Delta x^2 + \frac{1}{6}\partial^3_x\theta|^t_x\Delta x^3 + \mathcal{O}(\Delta x^4), \tag{2.27}$$

where, following (2.25), θ and all its first and higher order derivatives with respect to x on the right-hand side of (2.27) are evaluated at the point (x, t) in space and time, and the notation $\mathcal{O}(\Delta x^4)$ - order of Δx to the fourth - is used to emphasize that there are more terms and that the first term we have neglected is of fourth order in Δx. If we solve (2.27) with respect to the first order derivative we get

$$\partial_x\theta|^t_x = \frac{\theta|^t_{x+\Delta x} - \theta|^t_x}{\Delta x} + \mathcal{O}(\Delta x). \tag{2.28}$$

We may repeat this procedure by using a Taylor series to find the value of the function $\theta(x, t)$ at the point $x - \Delta x$. This is achieved by replacing Δx by $-\Delta x$ in (2.27) to get

$$\theta|^t_{x-\Delta x} = \theta|^t_x - \partial_x\theta|^t_x\Delta x + \frac{1}{2}\partial^2_x\theta|^t_x\Delta x^2 - \frac{1}{6}\partial^3_x\theta|^t_x\Delta x^3 + \mathcal{O}(\Delta x^4). \tag{2.29}$$

We observe that the only difference between (2.29) and (2.27) is the alternating sign in front of every second term on the right-hand side of (2.29). Solving (2.29) with respect to the first order derivative we get

$$\partial_x\theta|^t_x = \frac{\theta|^t_x - \theta|^t_{x-\Delta x}}{\Delta x} + \mathcal{O}(\Delta x). \tag{2.30}$$

Moreover, by subtracting (2.29) from (2.27), and solving for $\partial_x\theta|^t_x$ we get

$$\partial_x\theta|^t_x = \frac{\theta|^t_{x+\Delta x} - \theta|^t_{x-\Delta x}}{2\Delta x} + \mathcal{O}(\Delta x^2). \tag{2.31}$$

[4]The notation $\phi|^t_{x,y,z}$ is henceforth used to denote evaluation of the variable in question at the point x, y, z, t in four-dimensional space.

For later convenience we emphasize at this point that the choice Δx is arbitrary, we may equally well choose $2\Delta x$. Thus replacing Δx by $2\Delta x$ in the Taylor series (2.27) and (2.29) we get,

$$\theta|_{x+2\Delta x}^{t} = \theta|_{x}^{t} + 2\partial_x\theta|_{x}^{t}\Delta x + 2\partial_x^2\theta|_{x}^{t}\Delta x^2 + \frac{4}{3}\partial_x^3\theta|_{x}^{t}\Delta x^3 + \mathcal{O}(\Delta x^4), \tag{2.32}$$

and

$$\theta|_{x-2\Delta x}^{t} = \theta|_{x}^{t} - 2\partial_x\theta|_{x}^{t}\Delta x + 2\partial_x^2\theta|_{x}^{t}\Delta x^2 - \frac{4}{3}\partial_x^3\theta|_{x}^{t}\Delta x^3 + \mathcal{O}(\Delta x^4), \tag{2.33}$$

respectively. Again by subtracting the two and solving for $\partial_x\theta|_{x}^{t}$ we get

$$\partial_x\theta|_{x}^{t} = \frac{\theta|_{x+2\Delta x}^{t} - \theta|_{x-2\Delta x}^{t}}{4\Delta x} + \mathcal{O}(\Delta x^2). \tag{2.34}$$

2.7 Finite difference approximations

To derive possible *finite difference approximations* (FDAs) to the various derivatives of our PDEs we actually utilize the Taylor series above. For instance to derive feasible FDAs to the first order derivative in space $\partial_x\theta$ we utilize the expressions (2.27), (2.29) and (2.31) above in which Δx is a finite distance (nonzero). For instance using (2.31) we simply neglect the higher order terms, that is, terms of $\mathcal{O}(\Delta x^2)$, and get[5]

$$[\partial_x\theta]_{x}^{t} = \frac{\theta|_{x+\Delta x}^{t} - \theta|_{x-\Delta x}^{t}}{2\Delta x}. \tag{2.35}$$

We emphasize that this approximation is valid to $\mathcal{O}(\Delta x^2)$ since the first term we have neglected in the Taylor series in this case is $\mathcal{O}(\Delta x^2)$. Similarly we derive other possible FDAs of $\partial_x\theta|_{x}^{t}$ by neglecting terms of $\mathcal{O}(\Delta x)$ in (2.27) and (2.29), respectively, that is,

$$[\partial_x\theta]_{x}^{t} = \frac{\theta|_{x+\Delta x}^{t} - \theta|_{x}^{t}}{\Delta x}, \tag{2.36}$$

$$[\partial_x\theta]_{x}^{t} = \frac{\theta|_{x}^{t} - \theta|_{x-\Delta x}^{t}}{\Delta x}. \tag{2.37}$$

We note that while (2.35) is centered on the spatial point x, (2.36) and (2.37) are one-sided. The approximation (2.35) is therefore denoted a *centered* FDA, while (2.36) and (2.37) are denoted *one-sided* FDAs. We also note that while (2.36) is stepping forward, (2.37) use a backward step. Consequently (2.36) is referred to as a *forward* FDA, while (2.37) is referred to as a *backward* FDA. There is a major difference between the centered and the forward and/or backward FDAs. While the centered differences are valid to second order, that is, $\mathcal{O}(\Delta x^2)$, the two one-sided FDAs are only valid to first order, that is, $\mathcal{O}(\Delta x)$. Consequently the centered FDA (2.35) is also sometimes referred to as a second order FDA, while the one-sided FDAs (2.36) and (2.37) are referred to as first order FDAs.

[5]Henceforth an FDA is denoted by brackets. Thus an FDA approximation to $\partial_x\theta|_{x}^{t}$ is denoted $[\partial_x\theta]_{x}^{t}$.

We may perform exactly the same calculations based on Taylor series expansion, e.g., (2.26) to derive a FDA to the differentials in time t. For instance by expanding θ into a Taylor series in time we get

$$\theta|_x^{t+\Delta t} = \theta|_x^t + \partial_t\theta|_x^t\Delta t + \frac{1}{2}\partial_t^2\theta|_x^t\Delta t^2 + \frac{1}{6}\partial_t^3\theta|_x^t\Delta t^3 + \mathcal{O}(\Delta t^4), \tag{2.38}$$

$$\theta|_x^{t-\Delta t} = \theta|_x^t - \partial_t\theta|_x^t\Delta t + \frac{1}{2}\partial_t^2\theta|_x^t\Delta t^2 - \frac{1}{6}\partial_t^3\theta|_x^t\Delta t^3 + \mathcal{O}(\Delta t^4). \tag{2.39}$$

To construct a centered FDA to the time rate of change of θ we simply subtract (2.39) from (2.38) and solve with respect to $\partial_t\theta|_x^t$ to obtain

$$\partial_t\theta|_x^t = \frac{\theta|_x^{t+\Delta t} - \theta|_x^{t-\Delta t}}{2\Delta t} + \mathcal{O}(\Delta t^2). \tag{2.40}$$

Dropping terms of $\mathcal{O}(\Delta t^2)$ we get

$$[\partial_t\theta]_x^t = \frac{\theta|_x^{t+\Delta t} - \theta|_x^{t-\Delta t}}{2\Delta t}. \tag{2.41}$$

Thus we observe that the centered in time FDA (2.41) is valid to second order.

Similarly we may construct FDAs to higher order derivatives. For instance to find a centered FDA to $\partial_x^2\theta|_x^t$, we first simply add the two Taylor expansion (2.27) and (2.29) and solve with respect to $\partial_x^2\theta|_x^t$. We then get

$$\partial_x^2\theta|_x^t = \frac{\theta|_{x+\Delta x}^t - 2\theta|_x^t + \theta|_{x-\Delta x}^t}{\Delta x^2} + \mathcal{O}(\Delta x^2). \tag{2.42}$$

Then by neglecting terms of $\mathcal{O}(\Delta x^2)$ in (2.42) an FDA to the second order derivative is

$$[\partial_x^2\theta]_x^t = \frac{\theta|_{x+\Delta x}^t - 2\theta|_x^t + \theta|_{x-\Delta x}^t}{\Delta x^2}. \tag{2.43}$$

Since this expression gives equal weight to the points $x+\Delta x$ and $x-\Delta x$, that is, to the points on either side of x, the approximation is denoted centered. Like in (2.28) we note that the neglected terms are $\mathcal{O}(\Delta x^2)$. This is in contrast to the forward and backward approximations in which the neglected terms are of $\mathcal{O}(\Delta x)$. In fact all centered approximations share the fact that the neglected terms are of higher order than the one-sided approximations.

As exemplified in (2.43) we may formulate FDAs to any higher order derivative with respect to t, x and other spatial independent variables. For instance to derive a centered in space FDA for $\partial_x^3\theta|_x^t$ we combine (2.27) and (2.29) with (2.32) and (2.33) to obtain

$$\partial_x^3\theta|_x^t = \frac{\theta|_{x+2\Delta x}^t - 2\theta|_{x+\Delta x}^t + 2\theta|_{x-\Delta x}^t - \theta|_{x-2\Delta x}^t}{2\Delta x^3} + \mathcal{O}(\Delta x^2), \tag{2.44}$$

and hence, neglecting higher order terms, here terms $\mathcal{O}(\Delta x^2$, we get

$$[\partial_x^3\theta]_x^t = \frac{\theta|_{x+2\Delta x}^t - 2\theta|_{x+\Delta x}^t + 2\theta|_{x-\Delta x}^t - \theta|_{x-2\Delta x}^t}{2\Delta x^3}. \tag{2.45}$$

Hence (2.45) represents a second order FDA to $\partial_x^3\theta$. Since the FDA (2.45) is centered it comes as no surprise that (2.45) is valid to second order.

2.8 Truncation errors

As alluded to the main difference between the one-sided and centered FDAs is the order of the terms neglected when making the approximation from the Taylor series expansion. While we neglected terms of $\mathcal{O}(\Delta x^2)$ when using the centered FDA, the terms we neglected when using the one-sided approximation was $\mathcal{O}(\Delta x)$. Thus the centered FDA is more accurate than the one-sided FDA. While the centered FDA has an error of second order, the one-sided FDA has an error of first order. Since the error is a direct consequence of truncating the Taylor series expansion, we often refer to this error as the *the truncation error*. The order of the truncation error is therefore a measure of the accuracy of the scheme we have constructed.

As shown in Section 10.1 we may also use the Taylor series expansion to construct FDAs that are truncated to even higher orders, e.g., to $\mathcal{O}(\Delta x^n)$ where $n \geq 3$. Such FDAs are thus even more accurate and are therefore referred to as *higher order schemes* or higher order FDAs. We note that when constructing such approximations we have to include points that are distances $2\Delta x$ or more away from the point x as we did when deriving a centered FDA for $\partial_x^3 \theta|_x^t$ in (2.45). Although we desire our approximations to be as accurate as possible we emphasize that higher order schemes have other potential complications (cf. end of Section 10.1 on page 159).

Finally, we underscore that it is good practice to ensure that all the FDAs we make use of to approximate the various terms in our governing equations have the same truncation error in space and/or time, but not necessarily to the same order in both time and space. For instance consider a one dimensional wave propagating in a direction forming an angle to the x and y directions. The only way to ensure that the numerical solution then has the same accuracy regardless of the propagation direction of the wave is to use FDAs that have the same accuracy along all spatial directions or axes.

2.9 Notations

When solving a PDE using numerical methods, and in particular finite difference methods, it is common to define a grid or mesh which covers the domain over which the solution is to be found. As an example let us consider a two-dimensional spatial problem for which we seek a solution to the Laplace equation (2.8) within a quadratic domain where x, y both starts at 0 and ends at L[6]. We start by covering the domain by a quadratic mesh as displayed in Figure 2.1. We keep track of the grid points in the mesh by counting along the x-axis and the y-axis, respectively. Let us furthermore assume that there are $J+1$ points along the x-axis and $K+1$ points along the y-axis. To count the points we use dummy indices $j = 1, 2, 3, \ldots, J+1$ along the x-axis and $k = 1, 2, 3, \ldots, K$ along the y-axis. The point $x = 0$ along the x-axis is then associated with $j = 1$, while the point $x = L$ along the x-axis is associated with $j = J+1$. Similarly we associate the point $y = 0$ with $k = 1$ and the point $y = L$ with $k = K+1$. The jth point along the x-axis is then $x = x_j$ where the subscript refers to the value for x at the jth point along the x-axis. Similarly we let $y = y_k$ be associated with the kth point along the y-axis. The coordinates of the grid junctions are then given by x_j, y_k.

[6]Mathematically this can be expressed by $x \in < 0, L >$ and $y \in < 0, L >$

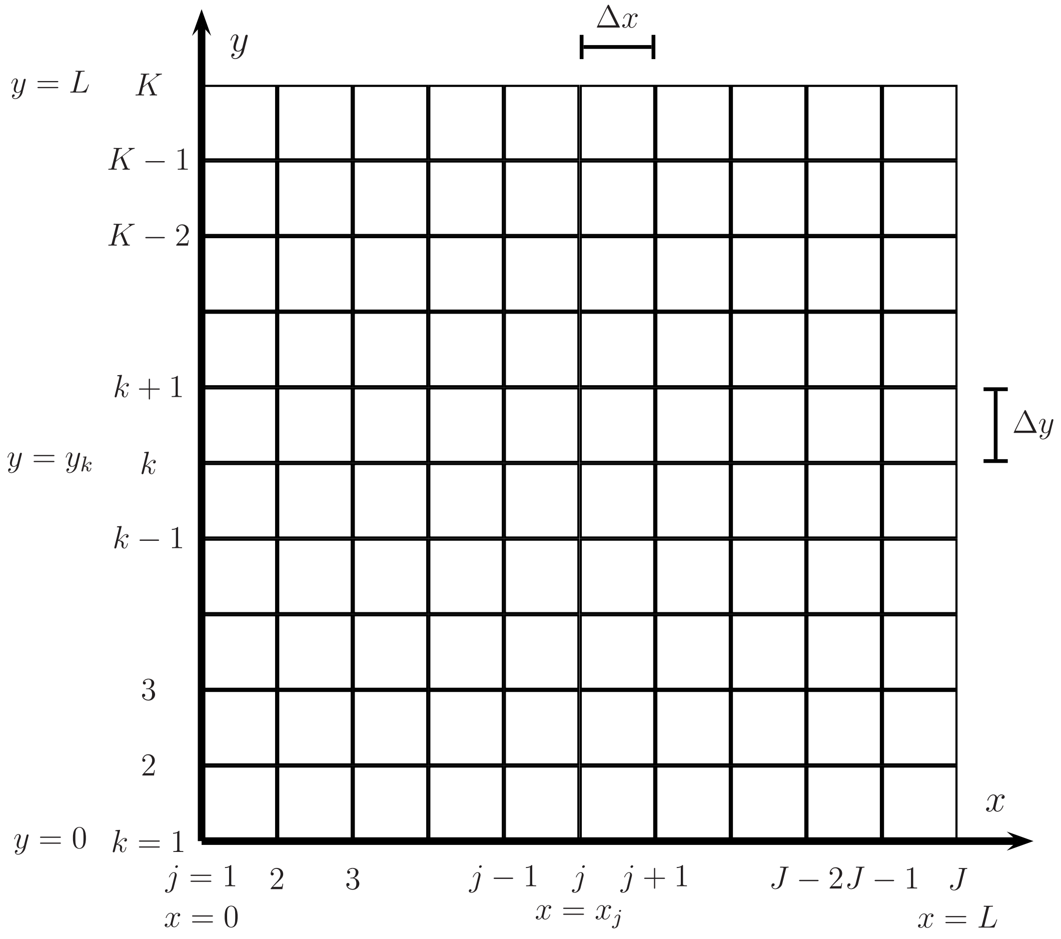


Figure 2.1: Displayed is a commonly used grid when employing numerical methods to solve PDEs. The grid points in the x, y directions are incremented by $\Delta x, \Delta y$, respectively, so that there are a total of J grid points along the x-axis and K grid points along the y-axis. The grid points are counted by using the dummy counters j, k, and the number of grid points are $J \times K$.

Let us denote the distance between two adjacent points along the x-axis by Δx and the distance between two adjacent points along the y-axis by Δy. Then the jth point along the x-axis is denoted

$$x_j = (j-1)\Delta x, \tag{2.46}$$

while the kth point along the y-axis is denoted

$$y_k = (k-1)\Delta y. \tag{2.47}$$

We note in particular that $x_1 = y_1 = 0$ and that $x_{J+1} = y_{K+1} = L$[7]. We also notice for later convenience that the latter gives

$$\Delta x = L/J, \quad \Delta y = L/K, \tag{2.48}$$

respectively[8]. It is also common to use the notation θ_{jk} to denote the value of the variable $\theta(x, y)$ at the grid point x_j, y_k. Thus

$$\theta_{jk} = \theta(x_j, y_k) = \theta[(j-1)\Delta x, (k-1)\Delta y]. \tag{2.49}$$

Furthermore follows that

$$\theta_{jk} = \theta(x_j, y_k) = \theta[(j-1)\Delta x, (k-1)\Delta y], \tag{2.50}$$

$$\theta_{j-1k} = \theta(x_{j-1}, y_k) = \theta[(j-2)\Delta x, (k-1)\Delta y], \tag{2.51}$$

and

$$\theta_{jk+1} = \theta(x_j, y_{k+1}) = \theta[(j-1)\Delta x, k\Delta y]. \tag{2.52}$$

To discriminate between spatial and temporal variables we hereafter use a superscript for the time counter, which is common practice. Let n be the time counter and Δt the time step or time increment. Then the time at the n'th time level is defined by[9].

$$t^n = n\Delta t, \quad n = 0, 1, 2, \cdots \tag{2.53}$$

from which follows that

$$\theta_j^n = \theta(x_j, t^n) = \theta[(j-1)\Delta x, n\Delta t]. \tag{2.54}$$

Thus the variable $\theta(x, t)$ at the point x_j, t^n in space and time is written

$$\theta_j^n = \theta|_j^n = \theta|_{x_j}^{t^n} = \theta(x_j, t^n) = \theta[(j-1)\Delta x, n\Delta t] \tag{2.55}$$

We note that if the variable in question is four-dimensional the notation we use is

$$\theta_{jkl}^n = \theta(x_j, y_k, z_l, t^n), \tag{2.56}$$

where $z_l = (l-1)\Delta z$.

[7]If the starting point in space is at, say $x = x_0$ along the x-axis and $y = y_0$ along the y-axis then $x_j = x_0 + (j-1)\Delta x$ and $y_k = y_0 + (k-1)\Delta y$. Hence $x_1 = x_0$, $y_1 = y_0$, $x_{J+1} = x_0 + L$ and $y_{K+1} = y_0 + L$.

[8]FORTRAN 90/95 allows us to use $j = 0$ and $k = 0$ as dummy counters. Under these circumstances $x_j = j\Delta x$ and $y_k = k\Delta y$. Thus $x_0 = y_0 = 0$ while $x_J = y_K = L$ as before. Under this circumstances $\Delta x = L/J$, and $\Delta y = L/K$

[9]The apparent inconsistency in starting the time counter at $n = 0$ and the space counter at $j = 1$ is historical. To save space on the computer we never store all time levels. Hence we never make use of do-loops when stepping forward in time. How many time steps we need to store depend on the time stepping scheme we use. If for instance the scheme is a two time level scheme we store only two time levels (and sometimes even only one level).

Let us consider the Taylor series expansions, e.g., (2.27) and (2.29). Using the preceding notation we get

$$\theta^n_{j\pm1} = \theta^n_j \pm \partial_x\theta|^n_j \Delta x + \frac{1}{2}\partial^2_x\theta|^n_j \Delta x^2 \pm \frac{1}{6}\partial^3_x\theta|^n_j \Delta x^3 + \mathcal{O}(\Delta x^4), \tag{2.57}$$

and

$$\theta^{n\pm1}_j = \theta^n_j \pm \partial_t\theta|^n_j \Delta x + \frac{1}{2}\partial^2_t\theta|^n_j \Delta x^2 \pm \frac{1}{6}\partial^3_t\theta|^n_j \Delta x^3 + \mathcal{O}(\Delta t^4). \tag{2.58}$$

Hence (2.28) and (2.40) are written

$$\partial_x\theta|^n_j = \frac{\theta^n_{j+1} - \theta^n_j}{\Delta t} + \mathcal{O}(\Delta x), \tag{2.59}$$

$$\partial_x\theta|^n_j = \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta t} + \mathcal{O}(\Delta x^2), \tag{2.60}$$

while the forward in time FDA displayed by (2.41) is

$$[\partial_t\theta]^n_j = \frac{\theta^{n+1}_j - \theta^n_j}{\Delta t}. \tag{2.61}$$

Similarly, the centered FDAs of the first order derivative in time and space are written

$$[\partial_t\theta]^n_j = \frac{\theta^{n+1}_j - \theta^{n-1}_j}{2\Delta t}, \tag{2.62}$$

$$[\partial_x\theta]^n_j = \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x}, \tag{2.63}$$

respectively, while the second order FDAs to the second order derivative in time and space are written

$$\left[\partial^2_t\theta\right]^n_j = \frac{\theta^{n+1}_j - 2\theta^n_j + \theta^{n-1}_j}{\Delta x^2}, \tag{2.64}$$

$$\left[\partial^2_x\theta\right]^n_j = \frac{\theta^n_{j+1} - 2\theta^n_j + \theta^n_{j-1}}{\Delta x^2}. \tag{2.65}$$

Finally we remark that the increments $\Delta x, \Delta y, \Delta z$ and Δt do not have to be constant, but may be allowed to vary in space and even in time. If the increments vary in space only we refer to the grid as an *unstructured mesh*. If the increments vary in both time and space we refer to the grid as an *adaptive unstructured mesh*.

2.10 Orthogonal functions

Note that when using finite difference techniques for time dependent or evolutionary problems, we only consider grid-point values of the dependent variables; no assumption is made about how the variables behave between grid points. An alternative approach is to expand the dependent variables in terms of a finite series of smooth orthogonal functions. The problem is then reduced to solving a set of ordinary differential equations which determine the behavior in time of the expansion coefficients.

As an example consider the general linear, one-dimensional time dependent problem

$$\partial_t \phi = \mathcal{H}[\phi] \quad \text{for} \quad x \in \langle -L, L \rangle \quad \text{and} \quad t > 0 \tag{2.66}$$

where $\phi = \phi(x,t)$ is a good function as defined in Section 2.6, $\mathcal{H}$ is a linear differential operator in x, and the computational domain is of length $2L$ in space. Note that to solve (2.66) we have to specify suitable boundary conditions at $x = \pm L$ and an initial condition at $t = 0$. Here we will simply assume that the condition at $x = \pm L$ is that ϕ is cyclic and that the initial value is known. Since ϕ is a good function it may be expanded in terms of an infinite set of orthogonal functions $e_n(x)$[10], where $n = 1, 2, 3, \ldots$. Thus

$$\phi = \sum_{n=-\infty}^{\infty} \varphi_n(t) e_n(x), \tag{2.67}$$

where $\varphi_n(t)$ are the time dependent expansion coefficients[11]. Without loss of generality we may assume that the expansion functions $e_n(x)$ are orthonormal so that

$$\int_{-L}^{L} e_n(x) e_m^*(x) dx = \begin{cases} 1 & ; \quad n = m \\ 0 & ; \quad n \neq m \end{cases}, \tag{2.68}$$

where $e_n^*(x)$ is the complex conjugate of $e_n(x)$. Consider that we know the expansion functions $e_n(x)$. It is then the expansion coefficients $\varphi_n(t)$ whose behavior we want to determine. To this end we first multiply (2.66) by e_m^*, and then integrate over all possible x-values, to give

$$\int_{-L}^{L} \partial_t \phi(x,t) e_m^*(x) dx = \int_{-L}^{L} \mathcal{H}[\phi] e_m^*(x) dx. \tag{2.69}$$

The left-hand side is further developed by use of (2.67) and (2.68) to give

$$\int_{-L}^{L} \partial_t \phi e_m^* dx = \int_{-L}^{L} \left(\sum_n \partial_t \varphi_n e_n \right) e_m^* dx = \sum_n \partial_t \varphi_n \int_{-L}^{L} e_n e_m^* dx = \sum_n \partial_t \varphi_n. \tag{2.70}$$

Since the operator $\mathcal{H}$ only operates on x follows in addition that

$$\mathcal{H}[\phi] = \sum_m \varphi_m \mathcal{H}[e_m]. \tag{2.71}$$

[10]Note that the expansion functions $e_n(x)$ are in general complex functions, e.g., $e_n(x) = e^{i\alpha_n x}$ where α_n is the wavenumber associated with the n^{th} eigenvalue.

[11]In fact this is a general method commonly used to separate variables when analytically solving differential equations involving more than one independent variable.

Using these results we get

$$\partial_t \varphi_n = \sum_m \varphi_m \int_{-L}^{L} \mathcal{H}[e_m] e_n^* dx \quad ; \quad \forall m. \tag{2.72}$$

We thus have a set of coupled, ordinary differential equations for the time rate of change for the expansion coefficients φ_n.

It is now interesting to consider how our choice of expansion functions can greatly simplify the problem

1. If the expansion functions are eigenfunctions of $\mathcal{H}$, we have $\mathcal{H}[e_m] = \lambda_m e_m$ where λ_m are the eigenvalues. Equation (2.72) then becomes

$$\partial_t \varphi_m = \lambda_m \varphi_m \quad ; \quad \forall m \tag{2.73}$$

   and becomes decoupled.

2. If the original equation is

$$\mathcal{G}\left[\partial_t \phi\right] = \mathcal{H}[\phi] \tag{2.74}$$

   where $\mathcal{G}$ is a linear operator, then our problem is simplified by using expansion functions that are eigenfunctions of $\mathcal{G}$ with eigenvalues λ_n. We then have,

$$\lambda_n \partial_t \varphi_n = \sum_n \varphi_m \int_{-L}^{L} \mathcal{H}[e_m] e_n^* dx. \tag{2.75}$$

2.11 Fourier series

A much used orthogonal set of expansion functions are the trigonometric functions $e^{i\alpha_n x}$ where α_n are an infinite number of discrete wavenumbers[12]. Thus any good function $\phi(x,t)$ may be written

$$\phi(x,t) = \sum_{n=-\infty}^{\infty} \varphi_n(t;\alpha_n) e^{i\alpha_n x}. \tag{2.76}$$

The series (2.76) is called a *Fourier series* and the expression

$$\varphi_n(t) e^{i\alpha_n x} \tag{2.77}$$

is called a *Fourier component*. We note that the complex conjugate to the expansion functions are $e^{-i\alpha_n x}$, and hence the Fourier series may be written

$$\phi(x,t) = \phi_0 + \sum_{n=1}^{\infty} \varphi_n(t;\alpha_n) e^{i\alpha_n x}. \tag{2.78}$$

It is important to realize that the subscript n attached to the expansion coefficients implies that they are different for each wavenumber, and hence depends on the wavenumber α_n as well as time.

[12]In the above problem with cyclic boundary conditions at $x = \pm L$ the wavenumbers are $\alpha_n = n\pi/L$.

2.12 Fourier transforms

Finally, let us assume that the function ϕ depends on x only, and is a good function. Under these circumstances we may define a function $\widetilde{\phi}$ such that

$$\widetilde{\phi}(\alpha) = \int_{-\infty}^{\infty} \phi(x) e^{-i\alpha x} dx. \tag{2.79}$$

We observe that $\widetilde{\phi}$ is a complex function consisting of a real as well as an imaginary part. As is common we refer to $\widetilde{\phi}$ as the *Fourier transform* of the real function ϕ. Furthermore we notice that the $\widetilde{\phi}$ is a continuous functions of the wavenumber $\alpha \in [-\infty, +\infty]$. Hence if we know the Fourier transform the original real function ϕ is retrieved from the real part of the *inverse Fourier transform* defined by

$$\phi(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \widetilde{\phi}(\alpha) e^{i\alpha x} d\alpha. \tag{2.80}$$

We observe that the "expansion coefficient" $\widetilde{\phi}$ now is a continuous function of the wavenumber $\alpha \in [-\infty, +\infty]$, and that the summation in (2.76) is replaced by an integral. We may also plot the Fourier transform $\widetilde{\phi}$ as a function of α. In that case the space spanned by $\widetilde{\phi}$ and α is called the *Fourier space* and its distribution the *Fourier spectrum*.

As revealed by (2.80) the Fourier transform (2.79) is simply the amplitude associated with the wave of wavenumber (or wavelength) α. The amplitude in a sense reveals how much "energy" is associated with each wavelength. Thus if we plot the Fourier transform in Fourier space the distribution reveals how much energy is contained in the various wavelengths. The waves with wavelengths having the highest amplitudes are also the wavelengths that contain the highest energy content. Knowing the Fourier transform thus reveals information about the wavelengths that dominates the motion.

We use the information in the Fourier spectrum to construct the grid, particularly to objectively the size of the spatial increments to choose (cf. Figure 2.1). If we intend to resolve the dominant portion of the motion we must choose the increments so that we have enough points per wavelength to resolve it. Ideally we should have 10 points per wavelength. As a minimum we must require that the size of the increments are such that we have 4 points per wavelength. Finally we emphasize that our solutions are real functions. Hence, if we know the Fourier transform we find the solution to our problem by first finding the inverse Fourier transform (2.80) and then extracting its real part.

Exercises

1. Show that both the Helmholtz and the Laplace equations are elliptic in x and y.
2. Show that the diffusion equation is parabolic in t, x and t, y, but elliptic in x, y.
3. Show by use of Taylor series expansions that a possible centered FDA of $\partial_x^4 \theta(x)$ is

$$[\partial_x^4 \theta]_j = \frac{\theta_{j+2} - 4\theta_{j+1} + 6\theta_j - 4\theta_{j-1} + \theta_{j-2}}{\Delta x^4}, \tag{2.81}$$

and that the truncation error is $\mathcal{O}(\Delta x^2)$. Note that we have to use points that are distances $2\Delta x$ away from the point x_j itself. This is common when deriving centered FDAs to higher order derivatives (cf. 2.45).

4. Assume that $\theta(x,t)$ and all of its derivatives tend to zero as $x \to \pm\infty$. Show that under these conditions the Fourier transform of $\partial_x\theta(x,t)$ and $\partial_x^2\theta(x,t)$ are

$$\widetilde{\partial_x\theta} = i\alpha\widetilde{\theta} \quad \text{and} \quad \widetilde{\partial_x^2\theta} = -\alpha^2\widetilde{\theta}, \tag{2.82}$$

respectively, where the notation $\widetilde{\psi}$ denotes the Fourier transform of ψ.

5. Show by making use of the results in Exercise 4 that a formal analytic solution to the diffusion equation

$$\partial_t\theta = \kappa\partial_x^2\theta, \tag{2.83}$$

where $\theta = \theta(x,t)$, κ is a constant, and the boundary conditions are

$$\theta = \begin{cases} 0 & ; \quad x \to +\infty, -\infty \\ \theta_0 e^{-(\frac{x}{a})^2} & ; \quad t = 0 \end{cases} . \tag{2.84}$$

is

$$\theta = \frac{a\theta_0}{2\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\alpha^2(\frac{1}{4}a^2+\kappa t)} e^{i\alpha x} d\alpha \tag{2.85}$$

Chapter 3

TIME MARCHING PROBLEMS

As alluded to in Chapter 1.1 most of the problems in the atmospheric and oceanographic sciences involve solving a time marching problem. Typically, we know the state of the atmosphere or the ocean at one specific time and want to know what the state is at a later time. Our task in numerical weather prediction (NWP) and numerical ocean weather prediction (NOWP) is then to use the governing equations of Section 1.1 on page 2 to find the state of the atmosphere or the ocean at some later time, just as postulated by *Bjerknes* (1904) (cf. quote on page iii). Such problems are known as *initial value problems* in mathematics.

The purpose of this chapter is to therefore to introduce the reader to some particular characteristics of some of the basic processes in the atmosphere and ocean, namely diffusion and advection and to recapitulate energy conservation by studying the variance.

3.1 Examples of time marching problems

A particular example of a time marching problem, inherent in our governing equations, is the tracer equation (1.3). It balances the time rate of change of a variable in response to advective and diffusive fluxes and source and sink terms, and is therefore sometimes referres to as the *advection-diffusion equation*. As the name indicates it is a combination of two different physical processes. The first is associated with the advection process. As outlined in (Section 2.4) the PDE is then hyperbolic. The second is associated with the turbulent mixing or diffusion process. The PDE is then parabolic (cf. Section 2.3).

Another important example is in the momentum equation (1.2) on page 2. A particular balance included in this equation is the possibility of balancing the pressure force against the Coriolis acceleration, the so called *geostrophic balance* as displayed in (1.39) on page 9. This possibility is what makes geophysical fluid dynamics stand out compared to ordinary (non-rotating) fluid dynamics. The importance of this balance is perhaps best illustrated through the shallow water equations (1.33) and (1.34). Any deviation from this balance manifests itself through non-zero acceleration terms, so called ageostrophic terms. Examples of such terms are the local time rate of change of the velocity, non-linear terms, etc.

The advection and diffusion problems (and their combination) and the shallow water equations are of fundamental importance in meteorology and oceanography. In fact it is at the very core of its dynamics (its “heart and soul”). Knowledge on how to solve these simple equations by numerical means is in fact a “must” for everyone who aspires to become a meteorologist and/or oceanographer. In the next three Chapters (Chapters 4, 5 and 6), we give a detailed account of how to solve respectively the diffusion eqaution, the advection eqaution and the shallow water equations by use of numerical methods. In Chapter 10 (Section 10.2 on page 164) we also give insight into how to solve the combined advection-diffusion problem.

We maintain that it is of fundamental importance to obtain knowledge on how to treat the various terms in these three fundamental problems numerically correct. At the same time these relatively simple problems conveniently serves the purpose of introducing some of the basic concepts needed to solve atmospheric and oceanographic problems employing numerical methods. Moreover, and equally important, they serve the purpose of illustrating some of the pitfalls.

Before venturing into details we highlight in this chapter some physical properties peculiar to each of the three fundamental problems. The motivation is that these important properties must be retained in any numerical solutions, or else the solution must be discarded as being false or incorrect. To check the behavior of the solutions against these fundamental properties is part of what is often referred to as *model verification* which is the first step in a chain of activities commonly referred to as model quality assurance or model evaluation procedures (*GESAMP*, 1991; *Lynch and Davies*, 1995; *Hackett et al.*, 1995). When coding errors are thus found we refer to the process as *debugging* which simply means to weed out all errors in the program code.

3.2 The advection-diffusion equation

We first focus on the tracer conservation equation (1.18) for a Boussinesq fluid. Neglecting possible tracer sources and sinks ($S_i = 0$) we get

$$\partial_t \theta + \nabla \cdot \mathbf{F} = 0, \tag{3.1}$$

where θ is any dependent variable (or tracer), for instance potential temperature, and $\mathbf{F}$ is a *flux vector* that includes both the advective flux and fluxes due to turbulent mixing. If θ is the potential temperature then (3.1) is the conservation equation for internal energy or heat content neglecting any source terms[1].

Note that the flux vector, as the name indicates, represents physical processes that transfer properties from one location to the next. In the atmosphere and oceans this is basically caused by two distinct and different physical processes. One is advective processes transporting or propagating properties from one place to the next via the motion (or waves). The second is turbulent mixing processes associated with small scale, inherently chaotic processes, that cause properties to be exchanged between two locations without invoking any mean motion. It is

[1]The total energy of a system consists of the internal energy and the mechanical energy. The internal energy is concerned with the heat content and is and important part of the thermodynamics. In contrast the mechanical energy concerns the motion of the fluid and is thus part of the fluid dynamics.

therefore useful to separate the flux vector $\mathbf{F}$ into two parts, as we did in (1.18), that is,

$$\mathbf{F} = \mathbf{F}_A + \mathbf{F}_D \tag{3.2}$$

where $\mathbf{F}_A$ represents the flux due to advective processes, hence referred to as the *advective flux vector*, and $\mathbf{F}_D$ represents the fluxes due to turbulent mixing, hence referred to as the *diffusive flux vector*.

Since the advective and diffusive flux vectors represent two very contrasting physical processes, they naturally have very different mathematical formulations or parameterizations. The advective flux vector $\mathbf{F}_A$ depends on the motion only. The advective flux of the property θ therefore follows the path of the individual fluid parcels. Thus its parameterization, or mathematical formulation, becomes

$$\mathbf{F}_A = \mathbf{v}\theta. \tag{3.3}$$

In contrast the mathematical formulation, or parameterization, of the diffusive flux vector is somewhat more complex. The reason is that the turbulent mixing in itself is a complex process, and its impact on the larger scale motion is in fact partly unknown. We do know however that the turbulent mixing in many respects acts to even out disturbances in the atmospheric and oceanic tracer fields, and hence its impact on the larger scale have many characteristics similar to processes like diffusion and/or conduction. In fact this is why we refer to this flux as the diffusive flux vector. Accordingly the most common parameterization of the turbulent mixing of tracers, or turbulence closure, is in terms of a diffusive process. Its mathematical formulation is thus,

$$\mathbf{F}_D = -\boldsymbol{\mathcal{K}} \cdot \nabla\theta, \tag{3.4}$$

where $\boldsymbol{\mathcal{K}}$ is the diffusion coefficient (or conductive) capacity[2]. Equation (3.4) expresses that the larger the gradient (or difference) the larger the diffusive flux and hence the more effective diffusion is to decrease any differences in the tracer θ over small distances. Note that since the diffusion coefficient depends on the strength of the turbulence it is not a constants and may change in time and space according to the local turbulence characteristics.

3.3 Diffusion

If we for a moment neglect the advective part of the flux vector the time rate of change of the tracer concentration is balanced by the diffusive flux only, that is.

$$\partial_t\theta = -\nabla \cdot \mathbf{F}_D = \nabla \cdot (\boldsymbol{\mathcal{K}} \cdot \nabla\theta). \tag{3.5}$$

where the last equal sign follows by use of (3.4) for the diffusive flux. Assuming that $\boldsymbol{\mathcal{K}} = \kappa\mathbf{ii}$ we get

$$\mathbf{F}_D = -\kappa\partial_x\theta\mathbf{i}. \tag{3.6}$$

[2]Its original formulation is due to a Dr. Adolf Eugen Fick who in 1855 formulated the parameterization $\mathbf{F}_D = -\kappa\nabla\theta$ with κ being a constant and a property of the medium.

Substituting (3.6) into (3.5) we get

$$\partial_t \theta = \kappa \partial_x^2 \theta. \tag{3.7}$$

where we have assumed that κ is constant. As outlined in Chapter 2 (3.7) is a parabolic problem (cf. eq. 2.9 on page 14). The resulting equation is called the diffusion equation, and solving it is referred to as solving the *diffusion problem*.

Recall that one of the important properties of the turbulent mixing is to even out small scale differences in the tracer fields. We thus have to ensure that our parameterization of the diffusive flux vector indeed have this property[3]. The noisiness of a field is commonly measured by its variance. The variance is defined as the square of the deviation from the mean, where the deviation from the mean is defined by

$$\theta' = \theta - \overline{\theta} \quad \text{or} \quad \theta = \overline{\theta} + \theta' \tag{3.8}$$

where $\overline{\theta}$ is the mean. We may thus investigate whether the noise increases or decreases by analyzing the time rate of change of θ'^2.

To arrive at an equation for the time rate of change of the variance we first multiply (3.5) by the tracer concentration θ itself. We then get

$$\partial_t \theta^2 = -2\theta \nabla \cdot \mathbf{F}_D = -2\nabla \cdot (\mathbf{F}_D \theta) + 2\mathbf{F}_D \cdot \nabla\theta. \tag{3.9}$$

Let us assume that (3.5) and by implication (3.9) are valid within a fixed (in time and space) volume V bounded by the surface Ω. Integrating (3.9) over the total volume V we get

$$\partial_t \left(\int_V \theta^2 dV \right) = -2 \int_\Omega \theta \mathbf{F}_D \cdot \delta\boldsymbol{\sigma} + 2 \int_V \mathbf{F}_D \cdot \nabla\theta dV. \tag{3.10}$$

Here the vector $\delta\boldsymbol{\sigma} = \mathbf{n}\delta\sigma$ where $\mathbf{n}$ is a unit vector directed along the outward normal to the surface Ω and $\delta\sigma$ is an infinitesimal surface element. To derive (3.10) we have also used the Gauss theorem. At the boundary Ω we must specify a boundary condition. We simply assume that the condition is either a Dirichlet or a Neuman condition. In the former case we let $\theta = 0$ at Ω, while in the latter case we let $\mathbf{n} \cdot \mathbf{F}_D = 0$ at the surface Ω. In either case we observe that the first term on the right-hand side of (3.10) is zero. Hence (3.10) reduces to

$$\partial_t \left(\int_V \theta^2 dV \right) = 2 \int_V \mathbf{F}_D \cdot \nabla\theta dV. \tag{3.11}$$

We then define the mean by the total content of the property θ within the fixed volume V, that is,

$$\overline{\theta} = \int_V \theta dV \quad \Rightarrow \quad \int_V \theta' dV = 0 \tag{3.12}$$

Thus

$$\partial_t \left(\int_V \theta^2 dV \right) = \partial_t \left(\int_V \theta'^2 dV \right) \tag{3.13}$$

[3]We note in passing that the parameterization also acts to even out any noise created by our choice of numerical methods, if any, when solving the equation numerically.

and hence (3.11) reduces to

$$\partial_t \left(\overline{\theta'^2}\right) = 2 \int_V \mathbf{F}_D \cdot \nabla\theta dV. \tag{3.14}$$

Thus if

$$\mathbf{F}_D \cdot \nabla\theta \leq 0 \tag{3.15}$$

the right-hand side of (3.14) is negative, and we get

$$\partial_t \left(\overline{\theta'^2}\right) \leq 0. \tag{3.16}$$

Equation (3.16) shows that as long as (3.15) is satisfied the diffusion term indeed acts to even out any noise in the θ field. We notice that (3.15) is always satisfied as long as the diffusive flux vector $\mathbf{F}_D$ is directed opposite to $\nabla\theta$. Under these circumstances we refer to the parameterization of the diffusive flux vector as being *down the gradient*. Assuming that $\mathbf{F}_D = -\kappa\nabla\theta$, known as *Fickian diffusion*, we get

$$\mathbf{F}_D \cdot \nabla\theta = -\kappa(\nabla\theta)^2 \leq 0, \tag{3.17}$$

which reveals that Fickian diffusion is indeed down the gradient and hence always tends to even out any noise in our solution. Thus we conclude that the diffusive flux vector, when properly parameterized, always acts to even out the variance in any tracer field.

Recall that most problems in oceanography and meteorology are non-linear. While there is no transfer of energy from one wavelength to the next in a linear system, this is not true for a non-linear systems. In such systems energy input on long wavelengths (small wave numbers) is always in the end transferred to progressively shorter wavelengths (high wave numbers). This fact was described elegantly in the following rhyme credited to G. I. Taylor[4]:

"Big whirls have smaller whirls that feed on their velocity, and little whirls have lesser whirls, and so on to viscosity .... in the molecular sense."

However, when making the finite difference approximations to our PDEs the wavelengths that we resolve is limited by the specified spatial increments, say $2\Delta x$, often referred to as the Nyquist wavelength (or frequency in the time domain). Thus as the energy is cascading downwards toward shorter wavelengths we must, in our numerical solutions, mimic this process across the Nyquist wavelength to wavelengths which are not resolved by our grid. Since diffusion has the property of damping differences it is one tool at hand that may prove useful to handle what is known as non-linear instability (cf. Section 10.3 on page 167).

[4]Geoffrey Ingram Taylor (1886 - 1975) made fundamental contributions to turbulence, championing the need for developing a statistical theory, and performing the first measurements of the effective diffusivity and viscosity of the atmosphere. He is commonly remembered as the namesake for several basic fluid flow instabilities (Taylor - Couette, Rayleigh - Taylor, and Saffman - Taylor).

3.4 Advection

If we next for a moment neglect the diffusive part of the flux vector $\mathbf{F}$, the time rate of change of the tracer concentration (or the heat content if θ is the potential temperature) is balanced by the advective flux only. Hence from (3.1) we get

$$\partial_t \theta = -\nabla \cdot \mathbf{F}_A = -\nabla \cdot (\mathbf{v}\theta). \tag{3.18}$$

Equation (3.18) is called the advection equation, and solving it is consequently referred to as solving the *advection problem*.

As for the diffusion problem we are looking for solutions within a limited fixed volume V in space bounded by the surface Ω, and for all times $t \in [0, \infty]$[5]. On the surface Ω the equations are replaced by the boundary conditions, while the initial condition replaces the equations at time $t = 0$. Let the advective flux be parameterized by the common parameterization $\mathbf{F}_A = \mathbf{v}\theta$, and let the boundary condition at the surface Ω be such that $\mathbf{F}_A \cdot \delta\boldsymbol{\sigma} = 0$[6]. Then by performing the same operation on (3.18) as we did in Section 3.3 we find that the total variance becomes

$$\partial_t \left(\int_V \theta^2 dV \right) = 2 \int_V \mathbf{F}_A \cdot \nabla\theta dV = \int_V \mathbf{v} \cdot \nabla\theta^2 dV = -\int_V \theta^2 \nabla \cdot \mathbf{v} dV. \tag{3.19}$$

Thus the total variance may increase or decrease depending on the sign of the velocity divergence. If the sum of the divergence is positive then the variance will decrease, while if it is negative then the variance will increase. The case $\nabla \cdot \mathbf{v} = 0$ is special. In this case the right hand-side of (3.19) is zero and hence any disturbances creating a variance in θ will just prevail, that is, the total variance is conserved.

As mentioned in Section 1.4 the Boussinesq ocean is to a good approximation divergence free due to its incompressibility (see also *Gill*, 1982, side 85). Thus in the ocean the advection process does not lead to any decrease or increase in the property being advected. Hence any disturbance generated in a limited domain may be advected to other locations undisturbed. This is not true for the atmosphere since the atmosphere is highly compressible. Thus in limited areas where the divergence is positive ($\nabla \cdot \mathbf{v} > 0$), that is, the individual fluid parcels are drawn apart, any disturbances in the total tracer variance are smoothed. In contrast the disturbances tend to increase in areas where $\nabla \cdot \mathbf{v} < 0$.

Finally we emphasize that the properties outlined above regarding the advection are important to retain when solving the advective problem by numerical means. In particular we stress that when the fluid is divergence free, like the ocean, then the total variance should be conserved. We also note that this is in stark contrast to the diffusion problem where all down the gradient diffusive fluxes give a decrease in the total tracer variance.

3.5 Shallow water equations

As alluded to the third and final fundamental balance equation important in atmosphere and ocean dynamics are the shallow water equations as displayed in (1.23) through (1.25) on page 7.

[5]In practice we have to limit the computation to a finite time span

[6]This is achieved by assuming $\mathbf{v} = 0$ or $\mathbf{v} \cdot \delta\boldsymbol{\sigma} = 0$, that is, no flow across the boundary.

Neglecting the forcing terms on the right-hand side of (1.25) we get

$$\begin{aligned} \nabla_H \cdot \mathbf{u} + \partial_z w &= 0, & (3.20)\\ \partial_t \mathbf{u} + \nabla_H \cdot (\mathbf{uu}) + \partial_z(w\mathbf{u}) + f\mathbf{k}\times\mathbf{u} &= -g\nabla_H \eta, & (3.21) \end{aligned}$$

where we have used (1.26) to substitute for the pressure[7].

Again we will, as we did for the advection and diffusion equations, investigate the properties of the time rate of change of the variance of the motion integrated over a fixed volume V, that is,

$$\partial_t \left(\int_V \mathbf{u}^2 dV \right). \tag{3.22}$$

We note that $e_K = \frac{1}{2}\mathbf{u}^2$ is the kinetic energy per unit mass. Thus

$$\int_V \mathbf{u}^2 dV = \int_V 2e_K dV = 2E_K, \tag{3.23}$$

where E_K is the *total kinetic energy*[8]. The total variance of the motion is therefore twice the total kinetic energy. We note that the kinetic energy is a positive definite quantity, that is, $E_K \geq 0$.

To find an equation for the time rate of change of the kinetic energy we first multiply (3.21) by $\mathbf{u}$. We then get

$$\partial_t e_K + \nabla_H \cdot (e_K \mathbf{u}) + \partial_z (e_K w) = -g\mathbf{u}\cdot\nabla_H \eta \tag{3.24}$$

where we have made use of the fact that

$$\mathbf{u}\cdot[\nabla_H \cdot (\mathbf{uu}) + \partial_z(w\mathbf{u})] = \nabla_H \cdot (e_K\mathbf{u}) + \partial_z(e_K w). \tag{3.25}$$

The latter follows by use of the continuity equation (3.20). Finally we note that the contribution from the Coriolis term vanishes since $\mathbf{u}\cdot(\mathbf{k}\times\mathbf{u}) = \mathbf{k}\cdot(\mathbf{u}\times\mathbf{u}) = 0$. Next we integrate (3.24) over the fixed volume V to get

$$\partial_t E_K = C, \tag{3.26}$$

where

$$C = -\int_V g\mathbf{u}\cdot\nabla_H \eta dz. \tag{3.27}$$

We observe that under the assumptions that there is no forcing terms[9], the time rate of change of the variance of the motion, or the kinetic energy per unit mass is proportional to C, a quantity yet to be interpreted.

To interpret C we first define the so called *available potential energy* per unit density (*Lorenz*, 1955; *Røed*, 1997, 1999) by

$$E_\Phi = \int_V gzdV - \int_{V_0} gzdV = \int_A \left(\int_{-H}^{\eta} gzdz - \int_{-H}^{0} gzdz \right) dA = \int_A \frac{1}{2} g\eta^2 dA, \tag{3.28}$$

[7]Note that we use $\mathbf{u}$ to denote the horizontal component of the velocity, that is, $\mathbf{v} = \mathbf{u} + w\mathbf{k}$

[8]To be precise E_K is the total kinetic energy per unit density, but since we have assumed a uniform density, that is, $\rho_0 = constant$, it is common to refer to E_K as the total kinetic energy.

[9]The forcing terms leads only to external source or sink terms, that is, irreversible energy conversion terms irrelevant for the present presentation

where A is the projected area of the volume V onto a horizontal surface. E_Φ is thus the potential energy in an arbitrary state from which is subtracted the potential energy of the intial state[10]. The rationale is that the enormous amount of potential energy of the initial state is not available for release into kinetic energy and is therefore of no interest.

We next show that

$$\partial_t E_\Phi = -C, \tag{3.29}$$

To this end we first note that (3.27) may be written

$$-C = g \int_A \left(\int_{-H}^{\eta} \mathbf{u} \cdot \nabla_H \eta dz \right) dA, \tag{3.30}$$

Since η and $\nabla_H \eta$ are both independent of depth/height the inner integral may be expanded as follows

$$\int_{-H}^{\eta} \mathbf{u} \cdot \nabla_H \eta dz = -\mathbf{U} \cdot \nabla_H \eta = -\eta \nabla_H \cdot \mathbf{U} + \nabla_H \cdot (\eta \mathbf{U}) \, . \tag{3.31}$$

where

$$\mathbf{U} = \int_{-H}^{\eta} \mathbf{u} dz \tag{3.32}$$

is the volume transport. Furthermore by integrating the continuity equation (3.20) from bottom to top and using the kinematic boundary condition (1.7) and (1.9) (cf. Section 1.2 on page 4) we get

$$\nabla_H \cdot \mathbf{U} = -\partial_t \eta, \tag{3.33}$$

and hence (3.31) becomes

$$\int_{-H}^{\eta} \mathbf{u} \cdot \nabla_H \eta dz = \frac{1}{2} \partial_t \eta^2 + \nabla_H \cdot (\eta \mathbf{U}) \, . \tag{3.34}$$

Substituting (3.34) into (3.30), noting that the second term on the right-hand side of (3.34) is a flux term that vanishes upon integration over the area A, we retrieve (3.29).

Adding (3.26) and (3.29) we get

$$\partial_t E = 0. \tag{3.35}$$

where $E = E_K + E_\Phi$ is the total mechanical energy. Thus C is a reversible energy conversion term converting potential energy into kinetic energy or vice versa. Equation (3.35) shows that the total mechanical energy is conserved. Thus if the kinetic energy E_K experience an increase (decrease) there is a similar decrease (increase) in the available potential energy E_Φ. We therefore conclude that under the assumptions of no external exchange of energy (a consequence of neglecting the forcing terms in eq. 3.21), the time rate of change of the variance of the motion is proportional to the conversion of kinetic energy to potential energy. For any numerical scheme to be trustworthy it should reflect that the total mechanical energy is conserved when constructing our numerical schemes (cf. *Arakawa and Lamb*, 1977).

[10]The initial state is defined as one at rest, and in static equilibrium, that is, $\mathbf{u}(x, y, z, 0) = 0$ and $\eta(x, y, 0) = 0$.

Chapter 4

THE DIFFUSION PROBLEM

The purpose of this chapter is to introduce the reader to the basic concepts on how to cast a continuum model equation into finite difference form. The reader of this chapter will learn how to discretize a given equation, and learn why some discretizations work and some not. This conveniently introduces concepts such as *numerical stability*, *convergence* and *consistency*. Furthermore the reader will learn how to check whether a dicretization is stable and consistent, learn about *explicit* and *implicit* schemes, the rudiments of *elliptic solvers* and finally the concept of *numerical dissipation* or artificial damping inherent in our discretizations.

4.1 The one-dimensional, diffusion equation

The first equation we will discretize is the one-dimensional, diffusion equation (3.7), that is,

$$\partial_t \theta = \kappa \partial_x^2 \theta, \tag{4.1}$$

where θ can be any dependent variable (e.g., potential temperature, humidity, speed, salinity, etc.), and κ is the diffusion coefficient. Thus we have assumed that the diffusive flux is parameterized as a down-the-gradient diffusion and that the diffusion (mixing) coefficient is uniform in time and space.

As alluded to in Section 2.3 on page 14, we note that (4.1) is parabolic in nature. The physical characteristic of the problem is therefore to transfer properties from one location to adjacent locations by conduction. Hence the diffusion process acts simply to even out differences without dissipation. If we for instance start with a very narrow tracer distribution (cf. Figure 5.6 on page 85) diffusion acts to transfer these high values to adjacent locations at the expense of the peak value as time is marching on. Thus as time passes the peak is diminished while the values at adjacent locations increases. If we allow the diffusion process to go on forever within a finite domain the tracer values becomes uniform[1]. In summary, the diffusion process acts to diminish differences so that the end result is a much smoother field.

An obvious example of a diffusion process in the atmosphere and the oceans is the turbulent mixing of heat. Then θ appearing in (4.1) is the potential temperature. Another classic

[1]In an infinite domain the tracer values will be infinitely small, but will cover the infinite domain.

atmosphere-ocean example is the so called Ekman problem in which θ represent the velocity. In the atmosphere it explains how the velocity is reduced in the planetary boundary layer due to friction at the surface. In the ocean the Ekman problem explains how the momentum due to surface traction is transferred downwards in the water column.

4.2 Finite difference equation

Consider for instance that θ describes the deviation (or anomaly) of the potential temperature away from a given mean temperature profile at zero degree Celsius. Then κ appearing in (4.1) is the strength of the turbulent mixing (considered uniform). Let us furthermore assume that we know the anomalous distribution at time $t = 0$, and that the temperature at the two end points $x = 0, L$ are fixed for all times and equals the initial temperature there, that is,

$$\theta(x,0) = f(x) \quad \forall x, \quad \text{and} \quad \theta(0,t) = \theta(L,t) = 0^{\text{o}}C \quad \forall t, \tag{4.2}$$

where $f(x)$ is a known function for $x \in < 0, L >$ and which is zero for $x = 0, L$. Our task is to find, by numerically solving (4.1), how the anomaly evolves in time between the two positions $x = 0, L$. We note that by considering that $\theta = 0^{\text{o}}C$ at $x = 0$ and $x = L$ for all times we imply that the boundary condition is a Dirichlet condition. We also assume that the initial anomaly is different from the trivial solution $\theta(x,0) = 0^{\text{o}}C; \forall x$, that is, that there exists at least one position in space where $f(x) \neq 0^{\text{o}}C$.

To find a numerical solution to (4.1) we follow the notation in Section 2.9. Thus we first divide the intervals $x \in \langle 0, L\rangle$ and $t \in \langle 0, T\rangle$, where T is some finite time, into respectively J and N sections of width Δx and Δt, respectively. They then form a grid whose grid points are located at (x_j, t^n) where $x_j = (j-1)\Delta x$ and $t^n = n\Delta t$. Here j and n are counters, counting the number of steps needed to reach the grid point (x_j, t^n). Thus $j \in [1, J+1]$ and $n \in [0, N]$ where $x_{J+1} = L$ and $t^N = T$ (cf. Figure 4.1 on page 41).

Next we must define a finite difference approximation to the derivatives $\partial_t \theta$ and $\partial_x^2 \theta$ at the grid points. Using a forward in time approximation to express $\partial_t \theta|_j^n$ and a centered in space approximation to express $\partial_x^2 \theta|_j^n$ it follows from Section 2.6 that

$$[\partial_t \theta]_j^n = \frac{\theta_j^{n+1} - \theta_j^n}{\Delta t}, \quad [\partial_x^2 \theta]_j^n = \frac{\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n}{\Delta x^2}. \tag{4.3}$$

By substituting the expressions (4.3) into (4.1) we get

$$\frac{\theta_j^{n+1} - \theta_j^n}{\Delta t} = \kappa \frac{\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n}{\Delta x^2}; \quad j = 2(1)J, \quad n = 0(1)N \tag{4.4}$$

Solving with respect to θ_j^{n+1} we get

$$\theta_j^{n+1} = \theta_j^n + K\left(\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n\right); \quad j = 2(1)J, \quad n = 0(1)N \tag{4.5}$$

where

$$K = \frac{\kappa \Delta t}{\Delta x^2}. \tag{4.6}$$

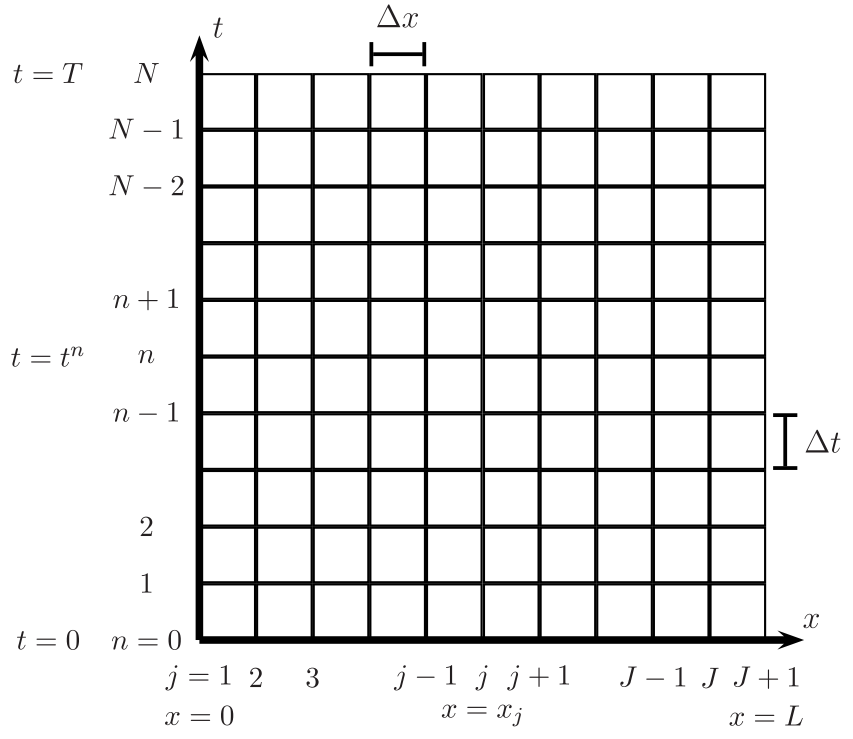


Figure 4.1: Displayed is the employed grid we use to solve (4.1) by numerical means. The grid points in the x, t directions are incremented by $\Delta x, \Delta t$, respectively. There is a total of $J + 1$ points along the x-axis and $N + 1$ points along the t-axis, counted by using the dummy indices j, n. The coordinates of the grid points are $x_j = (j - 1)\Delta x$ and $t^n = n\Delta t$, respectively.

Note that (4.4) and (4.5) are valid for $j = 2(1)J$ and for $n = 0(1)N$ only. At the boundaries $j = 1$ ($x = x_1 = 0$) and $j = J + 1$ ($x = x_{J+1} = L$) and for $n = 0$ ($t = t^0 = 0$) the boundary and initial conditions prevail as given in (4.2). In numerical language they are

$$\theta_j^0 = f_j \quad ; \quad j = 1(1)J+1 \quad \text{and} \quad \theta_1^n = \theta_{J+1}^n = 0 \quad ; \quad n = 0(1)N, \tag{4.7}$$

To find θ at the first time level $n = 1$ we substitute $n = 0$ into (4.5). We then get

$$\theta_j^1 = \theta_j^0 + K\left(\theta_{j+1}^0 - 2\theta_j^0 + \theta_{j-1}^0\right) \quad ; \quad j = 2(1)J. \tag{4.8}$$

Thus for the first "wet" point $j = 2$

$$\theta_2^1 = \theta_2^0 + K\left(\theta_3^0 - 2\theta_2^0 + \theta_1^0\right). \tag{4.9}$$

We note that θ_1^0, θ_2^0 and θ_3^0 on the right-hand side of (4.9) are known from the boundary and/or initial conditions (4.7). We may then proceed to evaluate $\theta_3^1, \theta_4^1, \cdots$ up to and including θ_J^1. For the last wet point $j = J$ we get in particular

$$\theta_J^1 = \theta_J^0 + K\left(\theta_{J+1}^0 - 2\theta_J^0 + \theta_{J-1}^0\right), \tag{4.10}$$

where again θ^0_{J-1}, θ^0_J and θ^0_{J+1} on the right-hand side are known from the initial and/or boundary conditions (4.7). This procedure thus provides values for the potential temperature anomaly at all the interior grid points for time level $n = 1$ (or at time $t = \Delta t$), that is, θ^1_j. In addition θ^1_1 and θ^1_{J+1} are known from the boundary condition at time level $n = 1$. Thus θ is known at all grid points at time level $n = 1$, and we may proceed to find θ at time level $n = 2$. We do this by substitution of $n = 1$ into (4.5). We then get

$$\theta^2_j = \theta^1_j + K\left(\theta^1_{j+1} - 2\theta^1_j + \theta^1_{j-1}\right); \quad j = 2(1)J, \tag{4.11}$$

Again we note that θ^2_1 and θ^2_{J+1} are known from the boundary condition. Having thus found θ for all grid points at time level $n = 2$ we may proceed to time level $n = 3$ and so on for all time levels n up to and including $n = N$.

We emphasize that at the boundaries $j = 1$ and $J + 1$ the variable θ is known from the boundary condition (4.7). This reflects the well known property of differential equations, whether they are PDEs or ordinary differential equations (ODEs), namely that they are valid only in the interior of a domain. At the boundaries (whether in time or space) the equations are replaced by the boundary condition. Thus (4.5) together with the boundary conditions gives us θ for all grid points x_j where $j = 1(1)J$ and all time levels t^n where $n = 0(1)N$.

We underscore that since $x_{J+1} = L = J\Delta x$ we cannot choose J, L and Δx independently. Once two of them are chosen the third is given by the formula

$$J = \frac{L}{\Delta x}. \tag{4.12}$$

Likewise follows that

$$N = \frac{T}{\Delta t} \tag{4.13}$$

showing that N, Δt and T also depend on each other.

We emphasize that we are not allowed to specify more than one boundary condition in time. The application of a forward, one-sided finite difference approximation in time, as for instance employed in (4.5), is therefore the obvious choice in order to bring us from the initial time level $n = 0$ to next time level. The accuracy of this scheme though is $\mathcal{O}(\Delta t)$. Since we applied a centered finite difference approximation in space the spatial accuracy is higher, namely $\mathcal{O}(\Delta x^2)$. We may increase the accuracy in time to the same level by employing a centered in time scheme for the time rate of change as well, that is, let

$$[\partial_t \theta]^n_j = \frac{\theta^{n+1}_j - \theta^{n-1}_j}{2\Delta t}. \tag{4.14}$$

Substitution of (4.14) into (4.1) then gives

$$\frac{\theta^{n+1}_j - \theta^{n-1}_j}{2\Delta t} = \kappa \frac{\theta^n_{j+1} - 2\theta^n_j + \theta^n_{j-1}}{\Delta x^2}; \quad j = 2(1)J, \quad n = 0(1)N, \tag{4.15}$$

or

$$\theta^{n+1}_j = \theta^{n-1}_j + 2K\left(\theta^n_{j+1} - 2\theta^n_j + \theta^n_{j-1}\right); \quad j = 2(1)J \quad n = 0(1)N. \tag{4.16}$$

To obtain the solution at the first time level $n = 1$, that is, to obtain θ_j^1 we substitute $n = 0$ into (4.16). We then get

$$\theta_j^1 = \theta_j^{-1} + 2K\left(\theta_{j+1}^0 - 2\theta_j^0 + \theta_{j-1}^0\right) \quad ; \quad j = 2(1)J, \tag{4.17}$$

which requires knowledge of θ_j^{-1}. This corresponds to knowing the potential temperature anomaly at a time $t < 0$, in this case at one time level *prior* to the initial time level. By using the one-sided forward scheme we avoid this problem, but sacrifices accuracy. As shown in Sections 4.3 - 4.8 there are, however, more pressing needs that makes us shy away from using a centered in time, centered in space finite difference approximation to solve the diffusion equation numerically.

4.3 Numerical stability

In fact the scheme (4.16) is what we refer to as being *numerically unstable*. This entails that the numerical solution, instead of following the continuous solution, steadily deviates from it. Commonly this happens exponentially just like an analytic instability (think of baroclinic and barotropic instabilities in the atmosphere and ocean). We therefore call this behavior numerical instability to distinguish it from the physical barotropic and baroclinic instabilities that we would actually like to simulate using our numerical models. Thus for our numerical solutions to have any legitimacy we must require that they are numerically stable. This is an absolute requirement and is formulated as follows:

A numerical scheme is stable if and only if the numerical solution is limited within any given finite time span

As a prelude to how we analyze the numerical scheme with respect to its numerical stability, let us first consider the analytic solution to (4.1). To this end we note that any good function θ may be written as a sum of cosines and sines, or even more compact as a sum of exponentials (e.g., Section 2.11 on page 27 or *Lighthill*, 1970, page 3)

$$\theta(x,t) = \sum_{m=-\infty}^{\infty} \Theta_m(\alpha_m, t)e^{i\alpha_m x} = \sum_{m=-\infty}^{\infty} \theta_m \tag{4.18}$$

where α_m is the wavenumber of the m't Fourier component. Each component in (4.18), that is,

$$\theta_m = \Theta_m e^{i\alpha_m x}, \tag{4.19}$$

is called a Fourier component. Here $\Theta_m(t)$ is the time dependent amplitude of the m't component. Substituting (4.18) into (4.1) we get

$$\partial_t \Theta_m = -\kappa\alpha_m^2 \Theta_m. \tag{4.20}$$

Note that we have dropped the summation, that is, we analyze each Fourier component separately. We observe that (4.20) is an ordinary differential equation (ODE). Solving it with respect to the amplitude Θ_m we get

$$\Theta_m = \Theta_m^0 e^{-\kappa\alpha_m^2 t}. \tag{4.21}$$

Here Θ_m^0 is the initial amplitude of mode m, that is, the value of Θ_m at time $t = 0$. We find these initial amplitudes by expanding the initial distribution of θ into a Fourier series, that is,

$$\theta(x,0) = \sum_{m=-\infty}^{\infty} \Theta_m^0 e^{i\alpha_m x} \tag{4.22}$$

Thus substituting (4.21) into (4.18) we get

$$\theta(x,t) = \sum_{m=-\infty}^{\infty} \Theta_m^0 e^{-\kappa\alpha_m^2 t} e^{i\alpha_m x}, \tag{4.23}$$

which is the analytic solution to (4.1). We note by looking at (4.21) that the amplitude Θ_m of each individual Fourier component decreases monotonically and exponentially as time increases. Moreover, we observe that the short waves (high wave numbers) decrease faster than the long waves (small wave numbers). This is in accord with Section 3.3 where we concluded, based on (3.16), that diffusion acts to smooth out disturbances. Moreover we learn from (4.23) that this smoothing is not the same for all wavelengths. In fact it is selective in the sense that small scale disturbances are smoothed fast while the longer waves are less prone to damping in the same time period. Thus diffusion acts like a filter efficiently smoothing the small scale noise, if any, without significantly damping the larger scale motion.

As is obvious we would like the numerical solution to behave accordingly. In particular we expect the numerical solution to the diffusion equation to decrease monotonically in time. Thus if the numerical solution increases in time it is obviously wrong and possibly unstable. Note that this instability has nothing to do with the accuracy of the chosen scheme. Yet it is the initial truncation error inherent in our scheme that is allowed to grow uncontrolled when the solution is unstable. We will return to this in Section 4.4 below.

To be able to analyze whether our chosen scheme is stable or not we need a proper mathematical definition. The requirement of numerical stability is commonly formulated by stating that for any finite time T, that is, for $0 < T < \infty$, there must exist a finite number, say B, such that

$$\left| \frac{\Theta_n}{\Theta_0} \right| \le B \tag{4.24}$$

where Θ_0 is the initial amplitude of the variable θ. For *linear systems*, and to certain degree also non-linear systems, it is possible to analyze the stability of the chosen scheme analytically. Note that we always perform such an analysis *before* implementing the chosen scheme on the computer.

4.4 von Neumann's stability analysis

One such method is the so called *von Neumann's method*. To analyze the stability von Neumann suggested to use a method somewhat similar to solving the equations analytically. The first step is to define a discrete Fourier component similar to the analytic one given in (4.19), that is,

$$\theta_j^n = \Theta_n e^{i\alpha j\Delta x}, \tag{4.25}$$

where Θ_n is the discrete amplitude at time level n and α is the wavenumber of that particular discrete mode[2]. We now define a *growth factor*

$$G \equiv \frac{\Theta_{n+1}}{\Theta_n} \quad \Rightarrow \quad \Theta_{n+1} = G\Theta_n \quad \text{and} \quad \Theta_{n-1} = G^{-1}\Theta_n. \tag{4.26}$$

Thus G is the amplification of the amplitude Θ as we proceed from one time level to the next. We observe that (4.26) is formally similar to (4.24), except that the growth factor G is defined as the ratio between the next and the former time level, that is, between time level $n+1$ and time level n, while (4.24) is the ratio between the value at a random time level and the initial value. Letting $n = 0$ in (4.26) then gives

$$\Theta_1 = G\Theta_0, \tag{4.27}$$

where Θ_0 is the initial amplitude. By letting $n = 1$ in (4.26) and making use of (4.27) we obtain

$$\Theta_2 = G\Theta_1 = G^2\Theta_0. \tag{4.28}$$

Continuing by letting $n = 3, 4, \ldots$ up to a random number $n = l$ we get

$$\Theta_l = G^l\Theta_0. \tag{4.29}$$

Thus G^l is the ratio between the amplitude at the random time level $n = l$ or random time $t = l\Delta t$ and the initial amplitude. Thus (4.24) is satisfied if

$$|G| \leq 1, \tag{4.30}$$

since then G^l decreases as the time level or time increases[3]. The criterion (4.30) is called *von Neumann's condition for stability*. Note that it is a sufficient condition, not a necessary condition. We return to this in Section 4.7 below.

4.5 Stability of the discrete diffusion equation

As our first example we analyze the forward in time, centered in space (FTCS) scheme given by (4.5). Substituting (4.25) into (4.5) we get

$$\Theta_{n+1} = \Theta_n + K\left(e^{i\alpha\Delta x} - 2 + e^{-i\alpha\Delta x}\right)\Theta_n \tag{4.31}$$

where K is as given in (4.6) and where we have divided through by the common factor $e^{i\alpha j\Delta x}$. Noting that $e^{i\alpha\Delta x} + e^{-i\alpha\Delta x} = 2\cos\alpha\Delta x$ we get

$$\Theta_{n+1} = \left[1 - 2K(1 - \cos\alpha\Delta x)\right]\Theta_n. \tag{4.32}$$

We thus find the growth factor by simply dividing (4.32) by Θ_n,

$$G = 1 - 2K(1 - \cos\alpha\Delta x), \tag{4.33}$$

[2]Note that we have dropped the subscript m on Θ and α for clarity.

[3]Confer Computer Problem No. 1

To satisfy (4.30) we observe that

$$-1 \leq 1 - 2K(1 - \cos\alpha\Delta x) \leq 1. \tag{4.34}$$

Since $0 \leq (1 - \cos\alpha\Delta x) \leq 2$ the right-hand side inequality is satisfied for all wavenumbers α, all time steps Δt and all space increments Δx. The inequality on the left-hand side, however, is satisfied if and only if

$$K(1 - \cos\alpha\Delta x) \leq 1. \tag{4.35}$$

Recall that $0 \leq (1 - \cos\alpha\Delta x) \leq 2$, and hence (4.35) is satisfied for all wavenumbers α if

$$K \leq \frac{1}{2} \quad \Rightarrow \quad \Delta t \leq \frac{\Delta x^2}{2\kappa}. \tag{4.36}$$

This condition ensures that (4.34) is satisfied. Hence von Neumann's condition (4.30) is satisfied as well. Furthermore (4.36) tells us that we cannot choose Δx and Δt independently. Once Δx is chosen the time step Δt must be chosen in accord with (4.36). We therefore say that the forward in time, centered in space scheme (4.5) is *conditionally stable* under the condition (4.36).

We also observe from (4.35) that the waves that first violate the inequality are waves with wavenumbers given by

$$\cos\alpha\Delta x = -1, \tag{4.37}$$

which correspond to those waves that maximizes $1 - \cos\alpha\Delta x$. The wavenumbers satisfying (4.37) are

$$\alpha_m\Delta x = (2m - 1)\pi; \quad m = 1, 2, \ldots. \tag{4.38}$$

with corresponding wavelengths

$$\lambda_m = \frac{2\pi}{\alpha_m} = \frac{2\Delta x}{2m - 1}. \tag{4.39}$$

The most dominant of these waves is the wave corresponding to $m = 1$. Thus the most unstable wave has wavelength

$$\lambda_1 = 2\Delta x. \tag{4.40}$$

This implies that the numerical instability will appear as "$2\Delta x$" noise, that is, noise of wavelength $2\Delta x$. Commonly it appears as a saw tooth pattern such as the one displayed in Figure 4.2.

In summary we found that the forward in time, centered in space scheme (4.5) applied to the diffusion equation, using von Neumann's method to analyze its stability, is a conditionally stable scheme. If the method had returned $|G| > 1$, in which case von Neumann's condition (4.30) is not met, the scheme would have been called an *unconditionally unstable* scheme. If the analysis had returned $|G| \leq 1$ regardless of our choice of Δx and Δt and wavenumbers α, the scheme is *unconditionally stable*. If the special case $|G| = 1$ is true then we in addition say that the scheme is *neutrally stable*.

It is worthwhile mentioning that when $|G| < 1$ it follows from (4.26) that $|\Theta_{n+1}| < |\Theta_n|$. Thus, inherent for all schemes for which $|G| < 1$ is that they include artificial numerical energy

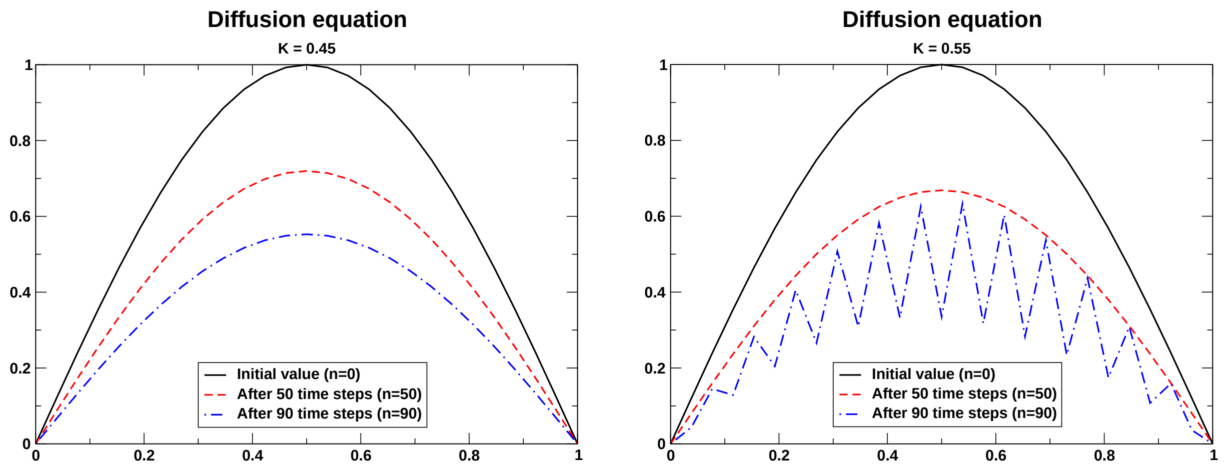


Figure 4.2: Displayed are solutions of the diffusion equation using the scheme (4.5) for respectively $K = \kappa\Delta t/\Delta x^2 = 0.45$ (left panel) and $K = 0.55$ (right panel) for $x \in (0, 1)$. The dependent variable θ is held fixed at the two boundaries $x = 0, 1$ and the initial condition is $\theta = \sin \pi x)$. The solutions are shown for the time levels $n = 0$, $n = 50$ and $n = 90$. Note the saw tooth like pattern in the right panel for $n = 90$ not present in the left panel. This indicates that the stability condition (4.36) is violated for $K = 0.55$, but not for $K = 0.45$.

dissipation[4]. We emphasize that even if the physical problem does not exhibit energy dissipation the numerical solution may exhibit dissipation. We therefore refer to this artificial energy dissipation as *numerical dissipation*. We note that this dissipation depends on the absolute value of the growth factor, and hence by implication, on our choice of scheme and spatial and temporal increments. It is therefore of importance to ensure that the numerical dissipation is as small as possible by making choices so that the absolute value of the growth factor is as close to one as possible to minimize the inherent artifical dissipation.

Any given problem in oceanography and meteorology may include natural energy dissipation. Thus if our scheme exhibits numerical dissipation it is important to ensure that it is small compared to the natural, physical dissipation. We therefore always favor neutral schemes ($|G| = 1$), since such schemes are energy conserving, a highly desirably property. If this is not possible we recommend to choose the time step and the space increments so as to minimize the numerical energy dissipation. This is the same as requiring $|G|$ to be as close to one as possible. Regarding the forward in time, centered in space scheme this implies that we have to choose a time step Δt that is small enough to satisfy (4.36), but at the same time is large enough to make $\Delta t \sim \Delta x^2/2\kappa$.

[4]In this context energy dissipation means that the square of the amplitude of the solution decreases in time.

4.6 The centered in time, centered in space scheme

Let us next analyze the stability of the similar centered in time, centered in space (CTCS) explicit scheme (4.16) using von Neumann's method. Substituting θ_j^n in (4.16) by its discrete Fourier component as defined in (4.25) and removing the common factor $e^{i\alpha j\Delta x}$. Then by use of the definition of the growth factor (4.26) we get

$$G = G^{-1} - 4K(1 - \cos\alpha\Delta x). \tag{4.41}$$

Multiplying by G and rearranging terms we get the second order equation

$$G^2 + 2\lambda G - 1 = 0, \tag{4.42}$$

to determine the growth factor. Here

$$\lambda = 2K(1 - \cos\alpha\Delta x) \geq 0. \tag{4.43}$$

Solving (4.42) with respect to G we get the two solutions

$$G_{1,2} = -\lambda \pm \sqrt{1 + \lambda^2}. \tag{4.44}$$

We recall that in order to be numerically stable both solutions must satisfy von Neumann's condition. We observe that

$$|G_2| = \lambda + \sqrt{1 + \lambda^2} \geq 1, \tag{4.45}$$

and hence that the centered in time, centered in space explicit scheme for the diffusion equation is unconditionally unstable. Thus:

Never use a centered in time, centered in space scheme for the diffusion problem. It is always unconditionally unstable.

4.7 The necessary stability condition

We mentioned above that von Neumann's condition is a sufficient condition. This implies that if (4.30) is satisfied then the scheme is definitively stable. The question is if its too strict, that is, if it is also the *necessary condition*?

To this end we return to the original stability requirement as formulated in (4.24). Substituting for Θ_n by use of (4.29) we get

$$|G|^n \leq B. \tag{4.46}$$

Taking the natural logarithmic on both sides then gives

$$n \ln |G| \leq \ln B \equiv B'. \tag{4.47}$$

Even if von Neumann's condition is too strict $|G|$ cannot be very much larger than one. Thus we may write $|G| = 1 + \epsilon$ where ϵ is a small ($\epsilon \ll 1$) positive number. Hence it follows that

$\ln|G| = \ln(1+\epsilon) \approx \epsilon$. Furthermore we note that $t^n = n\Delta t$ or that at $n = t^n/\Delta t$. When we substitute these expressions into (4.47) we obtain

$$\epsilon \leq \frac{B'\Delta t}{t^n} = \mathcal{O}(\Delta t). \tag{4.48}$$

Thus the necessary condition that satisfies the numerical stability requirement is

$$|G| \leq 1 + \mathcal{O}(\Delta t). \tag{4.49}$$

This shows that von Neumann's condition (4.30) is indeed too strict. However, most physical problems, even those containing instabilities, always involves some physical energy dissipation. Thus for all practical purposes we may apply the sufficient condition $|G| \leq 1$ when analyzing the numerical stability of our schemes, in particular if $|G| \lesssim 1$.

Finally, we remark that the growth factor G associated with the one-dimensional diffusion equation, as displayed by (4.33), is a scalar. For multi-variable and multi-dimensional problems the growth factor will commonly be a tensor or matrix, say $\boldsymbol{\mathcal{G}}$. The sufficient condition is then that its spectral radius is less than or equal to one. This is tantamount to requiring that the largest eigenvalue of $\boldsymbol{\mathcal{G}}$ is less than or equal to one.

4.8 Explicit and implicit schemes

The spatial operator on the right hand side of the schemes (4.5) and (4.16) are all evaluated at the time level n or earlier ($n-1, n-2, \ldots$). We refer to such schemes as being *explicit*. In contrast, if the spatial operator on the right-hand side includes variables evaluated at the new time level $n+1$ we refer to the scheme as being *implicit*. If all of them are evaluated at time level $n+1$ the scheme is a truly implicit scheme. If only one or a few are evaluated at time level $n+1$ we commonly refer to the scheme as being *semi-implicit*. Likewise, if we treat a multi-variable problem, e.g., the shallow water equations, where some of the terms are treated as being explicit and some implicit (Section 6.6 on page 120) we also refer to the scheme as being semi-implicit.

Explicit schemes are, as exemplified by the conditionally stable FTCS scheme (4.5), always simple to solve. Once the unknowns are known for one time level at all grid points, the computation of the next time level is straightforward, we just proceed from one grid point to the next as outlined in Section 4.2 (page 41). This is however not true for implicit schemes. To illustrate this let us turn the explicit FTCS scheme (4.5) into an implicit FTCS scheme. We do this by replacing the FDA or discretization of the spatial operator in (4.5) by one which is at time level $n+1$, that is, replacing $[\partial_x^2\theta]_j^n$ by $[\partial_x^2\theta]_j^{n+1}$ where

$$\left[\partial_x^2\theta\right]_j^{n+1} = \frac{\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1}}{\Delta x^2}. \tag{4.50}$$

Substituting this in the FTCS discretization of the diffusion equation as it appears in (4.5) we get

$$\theta_j^{n+1} = \theta_j^n + K\left(\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1}\right); \quad j = 2(1)J, \quad n = 0(1)\ldots, \tag{4.51}$$

which is an implicit FTCS scheme. Rewriting by moving all terms containing evaluation at time level $n+1$ to the left-hand side we get

$$-K\theta_{j-1}^{n+1} + (1+2K)\theta_j^{n+1} - K\theta_{j+1}^{n+1} = \theta_j^n; \quad j = 2(1)J, \quad n = 0(1)\dots. \tag{4.52}$$

We note that the implicit discretization require us to solve for θ at time level $n+1$ at the three grid points $j-1$, j, and $j+1$ simultaneously. This illustrates that although the diffusion equation is parabolic the implicit formulation turns the problem into an elliptic type problem. We will return to how to solve elliptic problems using direct elliptic solvers in Section 4.11.

We expect that turning an explicit scheme into an implicit scheme may impact its numerical stability and/or condition for stability. To illustrate this we analyse the implicit FTCS scheme (4.51). Using von Neumann's method step by step we get

$$G = \frac{1}{1+\lambda} \tag{4.53}$$

where λ is as given by (4.43). Thus since $\lambda \geq 0$ follows that $|G| \leq 1$ for all Δt, Δx and wavenumbers α. Hence, in contrast to the explicit FTCS scheme there is no restriction on the time step Δt once the space increment Δx is specified. Hence the implicit FTCS scheme is *unconditionally stable*.

As another example let us consider an implicit version of the unconditionally unstable, explicit CTCS scheme (4.16). As above we make it implicit by evaluating the FDA of $\partial_x^2\theta$ at time level $n+1$ rather than at time level n. Consequently we use (4.50) as our FDA for the spatial operator. We then get

$$\theta_j^{n+1} = \theta_j^{n-1} + 2K\left(\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1}\right) \quad ; \quad j = 2(1)J, \quad n = 0(1)\dots, \tag{4.54}$$

which is the implicit CTCS scheme. We note that by treating the CTCS scheme implicitly we again have to use elliptic solvers if we were to solve it numerically. To analyze its stability we employ von Neumann's method. By following the method step by step, and finally solving for the growth factor, we get two solutions given by $G_{1,2} = \pm|G|$ where

$$|G| = \frac{1}{\sqrt{1+2\lambda}}, \tag{4.55}$$

where again λ is as given by (4.43). We observe that as for the implicit FTCS scheme $|G| \leq 1$ for all Δt, Δx and wavenumbers α. Hence the implicit CTCS scheme is *unconditionally stable* as well, which is in stark contrast to the explicit CTCS scheme, the latter being unconditionally unstable.

In fact all truly implicit schemes are unconditionally stable. This is also true for most semi-implicit schemes, but not all as for instance exemplified by the Crank-Nicholson scheme treated in Section 4.10. Since all implicit schemes are unconditionally stable there is no constraint on the time step Δt. Hence we may choose Δt to be as long as we wish when considering the numerical stability alone. It is therefo,re tempting to use implicit schemes when solving our governing equations numerically since we only need a few time steps to reach the solution. Nevertheless it is commonly wise to stay away from implicit schemes, in particular under circumstances when diffusion dominates the physics of our problem. The rationale is, as exemplified by (4.53) and (4.55), that the growth factor $|G| < 1$[5], and hence always contain what is called *numerical dissipation*. Moreover, as shown by (4.53) and (4.55), $|G|$ gets smaller and smaller with increasing time step entailing that the numerical dissipation increases with increasing time step. Consequently, if we employ an implicit scheme we have to ensure that the numerical dissipation is small compared to the physical dissipation. Hence to minimize the artificial dissipation we have to choose a time step Δt that is so small that the growth factor is close to one. In summary, although implicit schemes are unconditionally stable regardless of how long the time step Δt is, we are constrained by the need to keep the numerical dissipation as small as possible. We therefore in gnereal *strongly advice* against the use of implicit scheme.

4.9 Convergence and consistency: DuFort-Frankel

As is obvious we would like our numerical solutions to mimic the true or analytical solution to our governing equations. We analyze this by letting the space and time increments $\Delta x, \Delta y, \Delta z$, and Δt approach zero independent of each other. If we find that the numerical solution converges toward the analytic solution when the increments tend to zero independently we call the scheme *convergent*. However, to perform such a study and proof is tedious especially for complex cases. If on the other hand, as shown by *Lax and Richtmyer* (1956), the numerical scheme is stable and *consistent* then the scheme is convergent[6]. A scheme is *consistent* if it, in the limit when the increments tend zero independently[7], approaches the continuous governing equations. If not we say that the scheme is *inconsistent*. Hence, if we in addition to stability require that the employed scheme is consistent we are ensured that the numerical solution is convergent. Consistency and numerical stability therefore forms the two fundamental properties that our schemes should obey.

We note that all schemes where the FDAs are based on Taylor series expansions (Section 2.6) satisfies the consistency requirement. Since both of the schemes (4.5) and (4.16) are based on a Taylor series expansion, they are both prime examples of consistent schemes. We may, however, quite easily construct numerical schemes without using Taylor series expansions. In these cases a consistency analysis is required, which is a by far an easier task than analyzing the convergence.

One example of such a scheme is the *Dufort-Frankel* scheme. To construct the Dufort-Frankel scheme we start with the consistent, unconditionally unstable and explicit CTCS scheme (4.16). The trick is to replace the term θ_j^n by an interpolated value using the value of θ at the two adjacent grid points (x_j, t^{n+1}) and (x_j, t^{n-1}) in time. Thus, using a two point, linear interpolation we

[5]Only for those wavenumbers for which $\lambda = 0$ gives $|G| = 1$. In general our solution contains all wavenumbers so that in general we get a growth factor that is truly less than one.

[6]This is known as the Lax equivalence theorem, which reads: "Given a properly posed initial value problem, boundary value problem, and a finite difference approximation that is consistent with the PDE, then stability is a necessary and sufficient condition for convergence."

[7]Note that this requirement is independent of how the increments go to zero and independent of how fast each of them go to zero.

replace θ_j^n by $\frac{1}{2}\left(\theta_j^{n+1}+\theta_j^{n-1}\right)$ on the right-hand side of (4.16). Thus we get

$$\theta_j^{n+1}=\theta_j^{n-1}+2K\left(\theta_{j+1}^n-\theta_j^{n-1}-\theta_j^{n+1}+\theta_{j-1}^n\right); \quad j=2(1)J, \quad n=0(1)\dots \tag{4.56}$$

We first note that the introduction of the term θ_j^{n+1} on the right-hand side makes the new scheme semi-implicit. Since all implicit schemes are stable we therefore expect the DuFort-Frankel scheme to be stable as well. We note that in contrast to the implicit scheme (4.54), the implicity is now limited to the single term θ_j^{n+1} involving only the space grid point x_j. We may move this term from the right-hand side of (4.56) to its left-hand side. We then get

$$\theta_j^{n+1}=\left[\theta_j^{n-1}+2K(\theta_{j+1}^n-\theta_j^{n-1}+\theta_{j-1}^n)\right](1+2K)^{-1}. \tag{4.57}$$

Thus the DuFort-Frankel scheme (4.56) is an explicit scheme despite the fact that is is semi-implicit. This is one reason why it has become so popular, in particular in oceanography, (e.g., *Adamec and O'Brien*, 1978). A second reason is that it is *unconditionally stable*. To show the latter we apply von Neumann's stability analysis and get

$$G_{1,2}=\frac{2K\cos\alpha\Delta x\pm\sqrt{1-4K^2\sin^2\alpha\Delta x}}{1+2K}, \tag{4.58}$$

which shows that $|G_{1,2}|<1$ and hence that the implicit scheme (4.56) is indeed unconditionally stable (c.f. Exercise 4 on page 59).

Since we constructed the Dufort-Frankel scheme simply by replacing one variable by an interpolated one we have to analyze the consistency of the scheme as well. To this end we employ the Taylor series expansions of Section 2.6. By substituting (2.57) and (2.58) into (4.56) we get

$$\partial_t\theta|_j^n-\kappa\partial_x^2\theta|_j^n=-\frac{\Delta t^2}{\Delta x^2}\left[1+\mathcal{O}(\Delta t^2)\right]+\mathcal{O}(\Delta t^2)+\mathcal{O}(\Delta x^2). \tag{4.59}$$

If and only if all the terms on the right-hand side of (4.59) goes to zero in the limit $\Delta x\to 0$ and $\Delta t\to 0$ independently, then the scheme is consistent. Evidently this is not the case for the first term on the right-hand side which tends to infinity if Δx tends to zero faster than Δt. We therefore note that (4.59) only converges to the continuous equation if $(\Delta t^2/\Delta x^2)\to 0$ when $\Delta x\to 0$ and $\Delta t\to 0$. This implies that the scheme is consistent iff[8] Δt tends to zero faster than Δx. Thus there is a condition associated with the consistency of the scheme, and hence, in line with the formulation used for the stability condition, we refer to the Dufort-Frankel scheme as a *conditionally consistent* scheme under the condition that $\Delta x\to 0$ slower than $\Delta t\to 0$.

As already mentioned at the end of Section 3.3 the diffusion term $\kappa\partial_x^2\theta$ is often added to the governing equation as a numerical artifact or "trick" to dissipate energy contained on the smaller scales. Commonly this small scale "noise" is created due to the presence of non-linear terms in the equations leading to interactions among the various wavelengths, which in turn is responsible for cascade of energy towards progressively smaller and smaller scales *Phillips* (1966). If we neglect to somehow parameterize this energy cascade at or near our grid resolution

[8]The formulation iff is short for "if and only if.

wavelengths, that is, energy contained in the $2\Delta x - 4\Delta x$ tail of the energy spectrum, energy tend to accumulate in this wavelength band. At some time or another into the integration this accumulation og energy leads to a violation of the linear, numerical stability criterion and the numerical solution goes unstable (or “blows up”).

When the diffusion term is used for as an artifact to parameterize the energy cascade it does not represent any of the physical processes we want to resolve. Rather it is introduced to avoid our model to blow up due to an accumulation of energy near the unresolved scales, so as to parameterize a smooth transfer of energy towards the unresolved scales of our grid[9]. Since this parameterization and/or the parameters it contains may change in accord with the models resolution we refer to it as *subgrid scale* (SGS) parameterization.

4.10 The Crank-Nicholson scheme

We finally consider a popular scheme called the *Crank-Nicholson* scheme. Like the Dufort-Frankel scheme it is semi-implicit. Its popularity is due to two facts. First, it is unconditionally stable, and second it is second order accurate in both time and space.

We start by recalling the two schemes (4.5) and (4.51), that is,

$$\theta_j^{n+1} = \theta_j^n + K(\theta_{j-1}^n - 2\theta_j^n + \theta_{j+1}^n) \tag{4.60}$$

and

$$\theta_j^{n+1} = \theta_j^n + K(\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1}). \tag{4.61}$$

We note that both are centered in space schemes. However, (4.60) is a forward in time (FTCS) and explicit scheme, while (4.61) is a backward in time (BTCS) and a truly implicit scheme. Hence (4.60) is conditionally stable under the condition $2K \leq 1$ while (4.60) is unconditionally stable. Finally, we note that both are consistent since they are based on Taylor series expansions.

We now combine the two schemes (4.60) and (4.61) to get

$$\theta_j^{n+1} = \theta_j^n + K\left[\gamma(\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1}) + (1-\gamma)(\theta_{j-1}^n - 2\theta_j^n + \theta_{j+1}^n)\right], \tag{4.62}$$

where γ is a number so that $0 \leq \gamma \leq 1$. If $\gamma = 0$ then (4.62) reduces to the explicit scheme (4.60). If $\gamma = 1$ then (4.62) reduces to the implicit scheme (4.61). If γ is between 0 and 1 the scheme contains both implicit and explicit terms. Hence the scheme (4.62) is semi-implicit and stability is not ensured.

To analyze the stability of the scheme we use von Neumann’s method. The growth factor G is (cf. Exercise 5 on page 59)

$$G = \frac{1 - 2K(1-\gamma)(1-\cos\alpha\Delta x)}{1 + 2K\gamma(1-\cos\alpha\Delta x)}. \tag{4.63}$$

[9]Note that for a given grid size $2\Delta x$ equals the Nüquist wavelength. Thus wavelengths smaller than $2\Delta x$ are unresolved by our grid.

We recall that the condition $|G| \leq 1$ or $-1 \leq G \leq 1$ is a sufficient condition for numerical stability. From (4.63) follows that $G \leq 1$ is always satisfied, while $G \geq -1$ is satisfied for all wavelengths iff

$$2K(1 - 2\gamma) \leq 1. \tag{4.64}$$

We note that for $\frac{1}{2} \leq \gamma \leq 1$ the left hand side of (4.64) is always negative or zero, implying that (4.64) is automatically satisfied. Under these circumstances the scheme is stable regardless of the value chosen for the increments Δx, Δt and the wavenumber α. Thus the scheme (4.62) is *unconditionally stable* provided $\frac{1}{2} \leq \gamma \leq 1$. This does not come as surprise, since under these circumstances the weight is on the implicit part[10]. If however $0 \leq \gamma < \frac{1}{2}$ the weight is on the explicit part. Under these circumstances the scheme is *conditionally stable* under the condition (4.64). We note that for $\gamma = 0$, in which case (4.62) equals the forward in time, centered in space finite difference approximation for the diffusion equation as displayed in (4.5), we indeed retrieve the condition (4.36) of Section 4.4, that is, $2K \leq 1$.

The value $\gamma = \frac{1}{2}$ is special. It constitutes the critical value at which the scheme (4.62) changes from being unconditionally stable ($\gamma > \frac{1}{2}$) to become conditionally stable ($\gamma < \frac{1}{2}$) under the condition (4.64). Substituing this particular value of γ into (4.62) we get

$$\theta_j^{n+1} = \theta_j^n + \frac{1}{2}K\left(\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1} + \theta_{j-1}^n - 2\theta_j^n + \theta_{j+1}^n\right), \tag{4.65}$$

which is referred to as the *Crank-Nicholson scheme*.

We note that the Crank-Nicholson scheme is unconditionally stable. The scheme is special also in another respect. Despite the fact that we employ a one-sided in time, finite difference approximation for the time rate of change for $\partial_t\theta$, it is actually second order accurate in time as well as in space. To prove it we start by utilizing the Taylor series expansions (2.24) and (2.25) as outlined in Section 2.6 on page 18. By substituting these series into the centered differences on the right-hand side of (4.65) we first get

$$\frac{\theta_j^{n+1} - \theta_j^n}{\Delta t} = \frac{1}{2}\kappa\left(\partial_x^2\theta|_j^{n+1} + \partial_x^2\theta|_j^n\right) + \mathcal{O}(\Delta x^2). \tag{4.66}$$

Expanding θ_j^{n+1} and $\partial_x^2\theta|_j^{n+1}$ using Taylor series we get

$$\frac{\theta_j^{n+1} - \theta_j^n}{\Delta t} = \partial_t\theta|_j^n + \frac{1}{2}\partial_t^2\theta|_j^n\Delta t + \mathcal{O}(\Delta t^2), \tag{4.67}$$

and

$$\partial_x^2\theta|_j^{n+1} = \partial_x^2\theta|_j^n + \partial_t(\partial_x^2\theta)|_j^n\Delta t + \mathcal{O}(\Delta t^2). \tag{4.68}$$

Substituting these series in (4.66) and rearranging terms we get

$$\partial_t\theta_j^n = \kappa\partial_x^2\theta|_j^n - \frac{1}{2}\left[\partial_t^2\theta|_j^n - \kappa\partial_t(\partial_x^2\theta)|_j^n\right]\Delta t + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2), \tag{4.69}$$

[10] As a corollary we note that this proves that the truly implicit scheme (4.61), which follows from (4.62) by letting $\gamma = 1$, is indeed unconditionally stable.

Furthermore, by appling the continuous diffusion equation (4.1) we get

$$\partial_t^2 \theta|_j^n = \kappa \partial_t (\partial_x^2 \theta)|_j^n. \tag{4.70}$$

The second term on the right-hand side of (4.69) therefore vanishes and hence we finally get

$$\partial_t \theta|_j^n = \kappa \partial_x^2 \theta|_j^n + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2). \tag{4.71}$$

Thus besides being unconditionally stable, the Crank-Nicholson scheme is of second order accuracy in both time and space. These two facts is why the Crank-Nicholson scheme is popular when solving true diffusive problems.

Nevertheless it has one disadvantage compared to the more standard schemes. We illustrate this by first rearranging the terms in (4.65) to obtain

$$K\theta_{j-1}^{n+1} - 2(1+K)\theta_j^{n+1} + K\theta_{j+1}^{n+1} = -2\theta_j^n - K\left(\theta_{j-1}^n - 2\theta_j^n + \theta_{j+1}^n\right). \tag{4.72}$$

Thus we cannot solve for θ_j^{n+1} without knowing θ_{j-1}^{n+1} and/or θ_{j+1}^{n+1}. Let us consider that we solve (4.72) for increasing values of j. Then for any arbitrary j we have already solved for θ_{j-1}^{n+1}, and it thus known. However, we have not yet solved for $j+1$, and thus θ_{j+1}^{n+1} is unknown. Thus the otherwise parabolic equation is by use of the Crank-Nicholson scheme turned into one which numerically look like an elliptic problem, just as we observed for the implicit CTCS scheme (4.54). Hence they both require us to employ what is called an elliptic solver for every time step. One such method called a direct elliptic solver is outlined in the next section.

4.11 A direct elliptic solver

Many of the model codes employed in numerical weather and numerical ocean weather prediction today employ semi-implicit methods. As we observed for the implicit CTCS and Crank-Nicholson scheme above, the consequence of introducing terms that are treated implicitly is that we have to solve an equation like (4.72) for each time step (e.g., Section 6.6). We are therefore in need of a method whereby problems such as (4.72) can be solved fast and efficient on a computer. Such methods are commonly referred to as *elliptic solvers*.

The most efficient elliptic solvers are those referred to as direct elliptic solvers. In the infancy of NWP most elliptic solvers were iterative or indirect elliptic solvers. Even though they may be accelerated, as for instance when applying the iterative elliptic solver called "Successive over-relaxation", they are much slower than the direct methods. To get insight into how the direct elliptic solvers work we will show the so called *Gauss elimination* method as an example.

Gauss elimination

The method consists of two steps. The first is called a *forward sweep*. Next we find the final solution by performing a *backward substitution*. To get started, we first rewrite (4.72) into a more general form,

$$a_j^n \theta_{j-1}^{n+1} + b_j^n \theta_j^{n+1} + c_j^n \theta_{j+1}^{n+1} = h_j^n; \quad j = 2(1)J, \quad n = 0(1)N \tag{4.73}$$

where a_j^n, b_j^n, and c_j^n represents the coefficients in (4.72). The use of the subscript j and superscript n attached to these coefficients acknowledges that they in general may be functions of space and time. Likewise h_j on the right-hand side represents all "forcing" terms, that is, our knowledge of the solution at the previous time step(s). We also note that we are required to solve (4.73) within a finite domain. Thus θ_1^n and θ_{J+1}^n are determined by the boundary conditions. We therefore assume for simplicity that these are known whatever boundary condition is applied for all time levels n.

Since we are required to solve (4.73) for every time step we consider one time level only and hence drop the superscripts n and $n+1$ for convenience. Thus we illustrate the Gauss elimination method by solving,

$$a_j\theta_{j-1} + b_j\theta_j + c_j\theta_{j+1} = h_j, \quad j = 2(1)J, \tag{4.74}$$

under the conditions

$$\theta_1 = \hat{\theta}_0, \quad \text{and} \quad \theta_{J+1} = \hat{\theta}_L, \tag{4.75}$$

where $\hat{\theta}_0$ and $\hat{\theta}_L$ are considered known functions. We observe that (4.74) may be more compactly written as

$$\boldsymbol{\mathcal{A}} \cdot \boldsymbol{\theta} = \mathbf{h}', \tag{4.76}$$

where the tensor $\boldsymbol{\mathcal{A}}$ is the *tridiagonal matrix*

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} b_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ a_3 & b_3 & c_3 & \dots & 0 & 0 & 0 \\ 0 & a_4 & b_4 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & a_{J-1} & b_{J-1} & c_{J-1} \\ 0 & 0 & 0 & \dots & 0 & a_J & b_J \end{bmatrix}. \tag{4.77}$$

and the vectors $\boldsymbol{\theta}$ and $\mathbf{h}'$ are, respectively,

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_2 \\ \theta_3 \\ \vdots \\ \theta_{J-1} \\ \theta_J \end{bmatrix}, \tag{4.78}$$

and

$$\mathbf{h}' = \begin{bmatrix} h_2 - a_2\hat{\theta}_0 \\ h_3 \\ \vdots \\ h_J - c_J\hat{\theta}_L \end{bmatrix}. \tag{4.79}$$

Note that the boundary conditions now are consumed into the the vector $\mathbf{h}'$, and therefore is part of the "forcing". This is in line with the mantra that the boundary condtions are as important as the governing equations in order to determine the solution.

Forward sweep

We are now ready to perform the forward sweep. The idea is to replace all elements of the matrix $\boldsymbol{\mathcal{A}}$ positioned in the lower left half with zeros. At the same time it is convenient to normalize the diagonal elements, that is, turn everyone of them into the value 1. We start with the equation for $j = 2$. From (4.76) follows

$$b_2\theta_2 + c_2\theta_3 = h_2'. \tag{4.80}$$

We then normalize by dividing by b_2

$$\theta_2 + d_2\theta_3 = w_2, \tag{4.81}$$

where

$$d_2 = \frac{c_2}{b_2} \quad \text{and} \quad w_2 = \frac{h_2'}{b_2}. \tag{4.82}$$

For $j = 3$ we get from (4.76)

$$a_3\theta_2 + b_3\theta_3 + c_3\theta_4 = h_3' \tag{4.83}$$

Substituting for θ_2 from (4.81) and normalizing gives

$$\theta_3 + d_3\theta_4 = w_3 \tag{4.84}$$

where

$$d_3 = \frac{c_3}{b_3 - d_2a_3} \quad \text{and} \quad w_3 = \frac{h_3' - a_3w_2}{b_3 - d_2a_3}. \tag{4.85}$$

Repeating this for $j = 4$ we get

$$\theta_4 + d_4\theta_5 = w_4 \tag{4.86}$$

where

$$d_4 = \frac{c_4}{b_4 - d_3a_4} \quad \text{and} \quad w_4 = \frac{h_4' - a_4w_3}{b_4 - d_3a_4}. \tag{4.87}$$

Thus repeating this procedure up to and including $j = J - 1$ we get

$$\theta_j + d_j\theta_{j+1} = w_j, \quad j = 2(1)J - 1. \tag{4.88}$$

where the coefficients d_j and w_j are defined by the recursion formula

$$d_j = \begin{cases} \frac{c_2}{b_2} & ; \quad j = 2 \\ \frac{c_j}{b_j - d_{j-1}a_j} & ; \quad j = 3(1)J-1 \\ 0 & ; \quad j = J \end{cases}, \quad w_j = \begin{cases} \frac{h_2'}{b_2} & ; \quad j = 2 \\ \frac{h_j' - a_jw_{j-1}}{b_j - d_{j-1}a_j} & ; \quad j = 3(1)J \end{cases}, \tag{4.89}$$

respectively. We observe that d_J is set to zero. This is due to the fact that for $j = J$ (4.76) becomes

$$a_J\theta_{J-1} + b_J\theta_J = h_J', \tag{4.90}$$

and hence by substituting for θ_{J-1} from (4.88) we simply get

$$\theta_J = w_J. \tag{4.91}$$

Note that all the coefficients d_j and w_j can be calculated once and for all.

We also notice that in matrix form (4.77) now reads

$$\boldsymbol{\mathcal{A}}' \cdot \boldsymbol{\theta} = \mathbf{w}, \tag{4.92}$$

where the matrix $\boldsymbol{\mathcal{A}}'$ is

$$\boldsymbol{\mathcal{A}}' = \begin{bmatrix} 1 & d_2 & 0 & \dots & 0 & 0 \\ 0 & 1 & d_3 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & d_{J-1} \\ 0 & 0 & 0 & \dots & 0 & 1 \end{bmatrix}, \tag{4.93}$$

and the vector $\mathbf{w}$ is

$$\mathbf{w} = \begin{bmatrix} w_2 \\ w_3 \\ \vdots \\ w_{J-1} \\ w_J \end{bmatrix}. \tag{4.94}$$

Thus we have completed a *forward sweep* in which the equation matrix is normalized and is upper triangular only.

Backward substitution

We are now ready to perform the *backward substitution*. First we note that all the w_j's and the d_j's are known using the recursion formula (4.89). Second we note from (4.91) that θ_J is simply given by w_J and that the latter is known from (4.89). Thus we are in a position where θ_J is known. Hence applying (4.88) for $j = J - 1$ and solving with respect to θ_{J-1} we get

$$\theta_{J-1} = w_{J-1} - d_{J-1}\theta_J. \tag{4.95}$$

We may continue this and solve for θ_{J-2}, θ_{J-3}, $\dots$ and so on. Hence we can solve for all the remaining θ_j's by backward substitution into (4.88), that is,

$$\theta_j = w_j - d_j\theta_{j+1} \quad \text{for} \quad j = J - 1(-1)2. \tag{4.96}$$

The Gauss elimination method is very simple to program, and it is also very efficient and fast on the computer. An example on the usefulness of this method, in which you are also required to program the method, is given in Computer Problem 5 named "Yoshida's equatorial jet current" in the accompanied, but separate Computer Problem notes. We urge the reader to do this problem, at least to solve the resulting ODE by employing the Gauss elimination method.

Exercises

1. Show that the scheme (4.16) is unconditionally unstable. Hint: Show that $|G| > 1$ regardless of the choice made for Δt and Δx.

2. Show that if $|G| = 1$ then the chosen scheme has no numerical dissipation.

3. Show that the growth factor associates with the scheme (4.54) is

$$G = \left[1 + \frac{4\kappa\Delta t}{\Delta x^2}(1 - \cos\alpha\Delta x)\right]^{-\frac{1}{2}} \tag{4.97}$$

   and hence that the scheme is unconditionally stable. Also show that $|G| < 1$ for all wavelengths. Note that $|G|$ decreases as Δt increases.

4. Show that the growth factor for the DuFort-Frankel scheme (4.56) is indeed (4.58)

5. Show that the expression (4.63) is indeed the expression for the growth factor of the scheme (4.62) when using von Neumann's analysis method.

Chapter 5

THE ADVECTION PROBLEM

In this chapter we will investigate possible discretizations of the linear advection equation. As for the diffusion equation we will learn why some discretizations work and some not. In particular we learn that the discretization that made the finite difference approximation to the diffusion equation conditionally stable makes the finite difference approximation to the advection equation unconditionally unstable. In addition we will learn about various stable and consistent schemes such as the *leapfrog*, the *upstream* (or upwind) the *Lax-Wendroff* and the *Semi-Lagragian* schemes. Moreover we will learn about how to avoid the initial problem in centered in time schemes, and concepts such as *numerical dispersion*, *numerical diffusion* and *unphysical modes* and how to minimize their effect. In particular we will learn how to correct for numerical diffusion by using a *flux corrective method*.

5.1 The one-dimensional advection equation

Recall that the advective flux vector $\mathbf{F}_A = \mathbf{v}\theta$ where $\mathbf{v}$ is the wind in the atmosphere or current in the ocean. Let us assume that the Boussinesq approximation is valid. Then $\mathbf{v}$ is non-divergent, that is, $\nabla \cdot \mathbf{v} = 0$. Under these circumstances the advection equation (3.18) reduces to

$$\partial_t\theta + \mathbf{v} \cdot \nabla\theta = 0. \tag{5.1}$$

As we did for the diffusion problem we will reduce it to its simplest form[1], and therefore consider a one-dimensional advection process throughout this chapter. Consequently we let $\mathbf{v} = u\mathbf{i}$, where u is the advection speed along the x-axis. Note that in general $\mathbf{v}$ is determined from the momentume equation and so varies in both time and space. However, if we require $\mathbf{v} = u\mathbf{i}$ and the Boussinesq approximation to be valid we get $\partial_x u = 0$, in which case the space dependence vanish. To keep things as simple as possible we therefore assume that $u = u_0$ where u_0 is uniform in both time and space. Hence from (5.1) we get

$$\partial_t\theta + u_0\partial_x\theta = 0. \tag{5.2}$$

[1]"Make things as simple as possible, but no simpler" Albert Einstein (1879-1955)

The general analytic solution to (5.2) is[2]

$$\theta = \theta(x - u_0 t). \tag{5.3}$$

To illustrate this let us assume that θ is a good function (cf. Chapter 2, Section 2.11 on page 27). Then we may represent it by a Fourier series, that is,

$$\theta(x,t) = \sum_{m=0}^{\infty} \Theta_m(t) e^{i\alpha_m x}, \tag{5.4}$$

where α_m is the wavenumber of mode number m, and Θ_m is the associated amplitude of that mode[3]. By differentiating (5.4) with respect to x and then with respect to time t, and substituting the results into (5.2), we get

$$\sum_{m=0}^{\infty} (\frac{d\Theta_m}{dt} + i\alpha_m u_0 \Theta_m) e^{i\alpha_m x} = 0. \tag{5.5}$$

This is true only as long as

$$\frac{d\Theta_m}{dt} + i\alpha_m u_0 \Theta_m = 0 \quad ; \quad \forall m. \tag{5.6}$$

Hence we get

$$\Theta_m = \Theta_m^0 e^{-i\alpha_m t}, \tag{5.7}$$

where Θ_m^0 is the initial value of the amplitude of wavenumber mode m at time $t = 0$. Substituting (5.7) into (5.4) we get

$$\theta(x,t) = \sum_{n=0}^{\infty} \Theta_m^0 e^{i\alpha_m (x - u_0 t)}, \tag{5.8}$$

which indeed shows that the solution is a function of $x - u_0 t$ only. Moreover (5.8) shows that all the waves propagates with same speed u_0. An observer travelling with that speed will therefore experience no change in the property θ as time progresses. This is line with the results of Section 3.4 on page 36, that is, if the wind (or current) is non-divergent then there is no change in the variance.

Finally, let us assume that the initial condition is specified by a single harmonic (monochromatic) wave of wavelength λ and amplitude A. Then initially

$$\theta(x,0) = Ae^{i\frac{2\pi}{\lambda}x}, \tag{5.9}$$

and the only possible solution is a monocromatic wave of wavelength $\alpha_m = \alpha = 2\pi/\lambda$ and $\Theta_m^0 = A$, and the solution (5.8) becomes

$$\theta(x,t) = Ae^{i\frac{2\pi}{\lambda}(x - u_0 t)}. \tag{5.10}$$

[2]Exercise 1 on page 90 at end of this Chapter.

[3]We note that solving (5.2) for a limited domain, say $x \in [0, L]$, there is an upper bound to the wavelength, that is, there is a lower bound on α_m.

This particular solution is a wave of wavelength λ propagating with a phase speed u_0 in the positive x direction (assuming $u_0 > 0$). As alluded to in Chapter 1, this solution is typical of hyperbolic systems. Indeed, the solutions (5.3), (5.8) and (5.10) are all such that if we travel along with the advection speed we will experience no change in the property θ. If we, however, observe the wave from a fixed position in space, the property θ will change in accord with (5.3) as the "wave" passes by. In fact this may be inferred from Section 3.4. In our case the velocity is non-divergent. From (3.19) then follows that the variance of θ is conserved implying that any initial distribution of the property θ is conserved as time progresses.

The formal solution (5.8) underscores that the general analytic solution (5.3) to the advection equation consists of waves of various wavelengths (wavenumbers) and amplitudes all propagating at the same speed u_0. Furthermore, (5.8) underscores that the wave that dominates the solution is the wave belonging to the mode that contains most energy, that is, the wave that initially has the largest amplitude Θ_m^0.

5.2 Finite difference forms

Our concern is to develop a method to solve (5.2) numerically. To this end we discretize (5.2) following the procedure in the previous chapter. Thus we start by replacing the terms in the advection equation by appropriate FDAs. Since we require our numerical solution to converge to the analytic solution, which is equivalent to require that our numerical schemes are numerically stable and consistent, this might not be quite straightforward. If the scheme turns out to be unstable or inconsistent it is of no use to us and has to be discarded. Hence once a finite difference form is chosen we have to analyze it with respect to its stability and consistency. The former is performed making use of von Neumann's method (cf. Section 4.4), while the latter is analyzed by use of Taylor series (cf. Section 2.6) as outlined in Section 4.9 on page 52.

Let us start by applying the forward in time and centered in space (FTCS) scheme that worked well for the diffusion equation. In accord with (2.61) a forward in time FDA for the time rate of change based on Taylor series is

$$[\partial_t \theta]_j^n = \frac{\theta_j^{n+1} - \theta_j^n}{\Delta t}. \tag{5.11}$$

In a similar fashion, a centered in space FDA for the first order space derivative in (5.2) is by use of (2.63)

$$[\partial_x \theta]_j^n = \frac{\theta_{j+1}^n - \theta_{j-1}^n}{2\Delta x}. \tag{5.12}$$

By replacing the two terms in (5.2) by the FDAs (5.11) and (5.12), and solving with respect to θ_j^{n+1} we get

$$\theta_j^{n+1} = \theta_j^n - u_0 \frac{\Delta t}{2\Delta x} \left(\theta_{j+1}^n - \theta_{j-1}^n\right) \tag{5.13}$$

which is then our first constructed numerical scheme to solve the advection equation. As is common we will refer to (5.13) as the *Euler scheme*.

Since the Euler scheme (5.13) is based on Taylor series expansions, we know a priori that the consistency requirement is satisfied. It therefore remains to analyze its stability to satisfy ourselves that the scheme is stable and hence convergent. To this end we make use of von Neumann's method. Hence we start by substituting the discrete Fourier component (4.25) into (5.13). After some manipulations we get

$$\Theta_{n+1} = \Theta_n - \frac{u_0 \Delta t}{2\Delta x}\left(e^{i\alpha\Delta x} - e^{-i\alpha\Delta x}\right)\Theta_n, \tag{5.14}$$

where we have divided through by the common factor $e^{i\alpha j\Delta x}$. Recalling the definition of the growth factor (4.26), and noting that $e^{i\alpha\Delta x} - e^{-i\alpha\Delta x} = 2i\sin\alpha\Delta x$, we get

$$G = 1 - i\lambda, \tag{5.15}$$

where

$$\lambda = \frac{u_0\Delta t}{\Delta x}\sin\alpha\Delta x. \tag{5.16}$$

We observe that the growth factor is a complex number with a real part given by 1 and an imaginary part given by λ. According to von Neumann's method we are required to evaluate the absolute value of the growth factor. To this end we use the well known property of complex numbers, namely that its absolute value equals the square root of the sum of the squares of the real and imaginary parts[4]. Thus

$$|G| = \sqrt{1+\lambda^2}. \tag{5.17}$$

Since λ^2 in general is a positive definite the radical is always larger than one, and hence $|G| \geq 1$. Only for the special wavenumbers that make $\sin\alpha\Delta x = 0$ we get $|G| = 1$. The scheme is therefore in general *unconditionally unstable*. We then have the somewhat curious result that although the forward in time, centered in space scheme worked successfuly for the diffusion problem, it is totally unacceptable with regard to the advection problem.

Never use a forward in time, centered in space scheme for the advection problem. It always leads to an unconditionally unstable scheme.

This does not come as a total surprise. As alluded to in Chapter 2 the advection equation and the diffusion equation represent quite different physics and have quite different characteristics. While the diffusion equation is parabolic the advection equation is hyperbolic. We should therefore expect that a discretization that works well for the diffusion problem does not necessarily work well for the advection problem.

Thus we have to replace the Euler scheme by a discretization that leaves us with a stable numerical scheme. In fact there are many stable and consistent schemes suggested over the past to solve the advection equation (e.g., *O'Brien*, 1986, page 165 and onwards). The reason is that advection is one of the most prominent processes regarding the motion of the atmosphere and the ocean. As in real life there is no such thing as a perfect scheme. The various schemes all have advantages and disdvantages. Often a new scheme is suggested to minimize unwanted properties of

[4]Let $A = a + ib$ be an imaginary number with real part a and imaginary part b. Then $|A| = \sqrt{AA^*} = \sqrt{a^2+b^2}$ where $A^* = a - ib$ is the complex conjugate of A.

other schemes, while other schemes are suggested focusing on their efficiency on the computer. Because of the dramatic increase in the power and speed of computers over the years the earlier requirements of computer efficiency is simply lessened today. The focus is therefore shifting towards deriving schemes that provide better conservation properties and higher order accuracy, say schemes of $\mathcal{O}(\Delta t^4)$ and $\mathcal{O}(\Delta x^4)$ or higher (cf. Section 10.1 on page 159). Among the former are so called flux corrective schemes (cf. Section 5.16 on page 87) and semi-Lagrangian schemes (cf. Section 5.12 on page 80).

It is nevertheless constructive to analyze some of the earlier schemes. In particular we will study schemes that forms the basis for many of the more recently suggested schemes. Thus we start by analyzing four earlier schemes, namely the *leapfrog scheme*, the *Upwind* or *Upstream scheme*, the *Diffusive scheme*, and the *Lax-Wendroff scheme*.

5.3 The leapfrog scheme

We showed in the previous section that the forward in time Euler scheme yielded an unconditionally unstable scheme. It is therefore natural to test out whether a centered in time scheme would work better despite the fact that a centered in time scheme yielded an unconditionally unstable scheme applied to the diffusion equation. In fact the centered in time and centered in space scheme (CTCS) is stable and was early on employed to solve the advection equation in atmospheric and oceanic problems. It is still fairly popular and widely used at least as one option in many atmospheric and oceanic numerical models even today, and is commonly referred to as the *leapfrog scheme*.

To construct the leapfrog scheme we use Taylor series expansions to ensure that the scheme is consistent. Thus we simply replace the forward in time FDA for the time rate of change used in the Euler scheme by one which is centered in time. We thus replace (5.11) by (2.62), while keeping the centered in space FDA (5.12) to the first order space derivative. By replacing the two terms in (5.2) by their respective FDAs (2.62) and (5.12), and solving with respect to θ_j^{n+1}, we get

$$\theta_j^{n+1} = \theta_j^{n-1} - u_0 \frac{\Delta t}{\Delta x} \left(\theta_{j+1}^n - \theta_{j-1}^n \right) . \tag{5.18}$$

Since we used centered FDAs in both time and space to derive the finite difference equation (5.18) the truncation error is of $\mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2)$. The scheme is therefore often referred to as a second order scheme[5]. Inspection of (5.18) shows that we only use information about the variable θ from all the grid points surrounding it in x, t space, but do not incorporate information of the variable from the grid point x_j, t^n itself. We are in a sense leapfrogging the grid point x_j, t^n, and the name of the scheme is derived from this fact.

We underscore that since the leapfrog scheme is derived exclusively using Taylor series, it is a consistent scheme. Since it is also stable it satisfies Lax theorem. Thus its numerical solution approaches the continuous solution as Δx and Δt tend to zero. The associated stability condition

[5]In describing their model people often writes "..., while a second order scheme is employed for advective terms".

is detailed in the next scetion, where the famous Courant-Friedrich-Levy or CFL condition is presented.

As disclosed above the scheme is traditionally fairly popular and still widely used for three main reasons. For one the scheme is, as shown in Section 5.4, neutrally stable. Hence there is no numerical or artificial dissipation (damping) associated with the scheme, a highly desirable property. Secondly the scheme is of second order accuracy. Moreover, the scheme is explicit and thus easy to implement, and last but not least it works fast and efficiently on most computers.

Finally we refer to the fact that the leapfrog scheme also has some disadvantageous properties. First the scheme contains what is described as *numerical dispersion* that sometimes leads to negative tracer concentrations (cf. Section 5.5 on page 67). Secondly the scheme (cf. Section 5.7 on page 70) contains what is labeled *unphysical modes* which has to be dealt with. Finally, since the scheme is centered in time, we also have to deal with the initial value problem disclosed in Section 4.6 on page 48.

5.4 Stability of the leapfrog scheme: The CFL condition

To analyze its stability we make use of von Neumann's method as outlined in Section 4.4. Thus we first replace the dependent variable θ in (5.18) by its discrete Fourier component (4.25) to give

$$\Theta_{n+1} = \Theta_{n-1} - 2iu_0 \frac{\Delta t}{\Delta x} \sin \alpha \Delta x \Theta_n \tag{5.19}$$

To find the growth factor G we first make use of (4.26) and then multiply by the growth factor to obtain

$$G^2 + 2i\lambda G - 1 = 0, \tag{5.20}$$

where as in (5.16)

$$\lambda = u_0 \frac{\Delta t}{\Delta x} \sin \alpha \Delta x. \tag{5.21}$$

The two solutions for the growth factor are

$$G_{1,2} = -i\lambda \pm \sqrt{1-\lambda^2}, \tag{5.22}$$

and hence they are complex functions. This was to be expected since the factor in front of the first order term in (5.20) is imaginary. We note that if the radical in (5.22) is negative, that is, $\lambda \geq 1$, then the two solutions becomes purely imaginary functions. We also observe that under these circumstances $|G_1| \geq 1$. Hence for a stable scheme we have to require that the radical in (5.22) is positive. Then reusing the theorem that the absolute value of a complex number is the square root of the complex number itself multiplied by its complex conjugate we get (cf. eq. 5.17 of Section 5.2)

$$|G_{1,2}| = \sqrt{G_{1,2}G^*_{1,2}} = \sqrt{1-\lambda^2+\lambda^2} = 1. \tag{5.23}$$

Since by definition $\Theta_{n+1} \equiv |G|\Theta_n$ it follows that there is no artificial or numerical dissipation (damping) involved. The scheme (5.18) is therefore *neutrally stable*. We note that since we have assumed the advection velocity to be constant (i.e., $\mathbf{v} = u_0\mathbf{i}$) its gradient is zero. Hence it is highly desirably that the property of the scheme is in line with the property of advection processes as outlined in Section 3.4, namely that when the Boussinesq approximation is observed the total variance should be conserved as well.

We recall that for (5.23) to be true the radical in (5.22) must be positive. Hence

$$1 - \lambda^2 \geq 0 \quad \text{or} \quad |\lambda| \leq 1. \tag{5.24}$$

Since $-1 < \sin \alpha \Delta x < 1$ we note that if

$$|u_0| \frac{\Delta t}{\Delta x} \leq 1 \tag{5.25}$$

is true then (5.24) is satisfied. Thus (5.25) is a sufficient condition for stability for the leapfrog scheme (5.18). The condition (5.25) is referred to as the *Courant-Friedrich-Levy condition* or simply the CFL condition. The ratio or non-dimensional number

$$C = |u_0| \frac{\Delta t}{\Delta x} \tag{5.26}$$

is usually cited as the *Courant number*. The CFL condition is therefore commonly written simply as $C \leq 1$. Since Δx more often than not is given by the need to resolve the spatial structure or typical wavelengths of the physical problem, the CFL condition becomes a rigorous upper bound on the time step Δt, that is, $\Delta t \leq \Delta x/|u_0|$. Therefore the larger the advection speed the smaller the time step, and the smaller the grid size the smaller the time step.

In the atmosphere and ocean the dominant wavelength (or length-scale) we have to resolve is associated with the first baroclinic Rossby radius, say L_R. At Norwegian latitudes $L_R \approx$ 1000 km in the atmosphere, while in the ocean $L_R \approx$ 10 km. Thus in the atmosphere the Rossby radius is well resolved by using a space increment of $\Delta x \approx 100$ km, while in the ocean the similar space increment is two orders of magnitude smaller, that is, $\Delta x \approx$ 1 km. On the other hand a typical wind speed is about 10 ms^{-1}, while a typical current in the ocean is 0.1 ms^{-1}. Hence the speed of the ocean currents is two order of magnitudes smaller than the wind speed. Thus the time step Δt we have to use to satisfy the CFL condition is about 1 day both in the atmosphere and in the ocean. However, we will show later (Chapter 6) that the condition for stability is not determined by the advective part of the solution. In fact the speed that enters the CFL condition is the propagation speed of baroclinic waves, which have serious impact on the time step we have to apply in ocean and atmospheric models.

5.5 Numerical dispersion

The concept of dispersiveness is well known from other branches of physics and geophysics. In particular it is a common phenomenon regarding wave dynamics. By throwing stones in a still water most of us have indeed experienced it in practice. After the initial splash we observe that circular waves propagates away from the original splash point in such a way that that longer waves leads progressively shorter waves. The reason for this is that the phase speed, say c, depends on the wavenumber (or wavelength), that is, waves of different wavelengths propagate at different speeds. Regarding gravity waves in deep waters the longer the wavelength the faster the phase speed. Thus the longer waves will lead the progressively shorter waves. The same is also true for other types of waves for instance planetary Rossby waves.

Mathematically this is expressed through the dispersion relation $\omega = \omega(\alpha)$ where ω is the frequency and α is the wavenumber. Recall that the phase speed is

$$c = \frac{\omega}{\alpha}. \tag{5.27}$$

If the frequency is a linear function of α, then the phase speed becomes a constant and all waves propagate at the same speed, that is, $\partial_\alpha c = 0$. The solution is then said to be *non-dispersive*. In the general case, however, ω is a non-linear function of the wavenumber α. Then $\partial_\alpha c \neq 0$ and hence the phase speed depends on the wavelength. The solution is then said to be *dispersive*. Recall that the energy contained in the wave propagates with the group velocity defined by

$$c_g = \partial_\alpha \omega = \alpha \partial_\alpha c + c. \tag{5.28}$$

Hence if the wave is non-dispersive $c_g = c$ and the energy contained in the wave propagates at the same speed as the wave itself. On the other hand if the wave is dispersive then both the phase speed and the group velocity depend on the wavelength. Moreover if $\partial_\alpha c < 0$, which is the case for gravity waves, then the waves travel at speeds faster than their group velocity. Thus gravity waves tend to travel faster than their energy is propagated[6].

If we apply a wave solution to the advection equation (5.2), that is,

$$\theta = \Theta_0 e^{i\alpha(x-ct)}, \tag{5.29}$$

we find that the phase speed is $c = u_0$. Under these circumstances all the waves propagates with the same constant advection velocity, namely u_0, and hence the true solution to the advection equation is non-dispersive. The question then arises. Is this true for our employed numerical scheme? To investigate this we simply apply a similar analysis based on the FDA to the advection equation using a discrete version of (5.29), that is,

$$\theta_j^n = \Theta_0 e^{i\alpha(j\Delta x - c^* n\Delta t)}, \tag{5.30}$$

where c^* is the wave speed of our numerical scheme.

As an example let us consider the leapfrog scheme (5.18). We know that this scheme is neutrally stable. Furthermore, as long as all the gradients are well resolved by our grid it is a superb scheme in the sense that it is a stable and consistent scheme with no numerical dissipation. To get started we first substitute the discrete Fourier component (5.30) into (5.18). We then get

$$-2i\sin(\alpha c^* \Delta t) = -2iu_0 \frac{\Delta t}{\Delta x} \sin(\alpha \Delta x) \tag{5.31}$$

[6]For capillary waves $\partial_\alpha c > 0$ and hence a capillary wave travels at a speed slower than its energy.

or

$$c^* = \frac{1}{\alpha \Delta t} \arcsin \left(u_0 \frac{\Delta t}{\Delta x} \sin \alpha \Delta x \right). \tag{5.32}$$

Obviously $\partial_\alpha c^* \neq 0$. The leapfrog scheme is therefore dispersive. Since this dispersiveness is due to the FDA, that is, associated with our numerical solver and hence is artificial, we commonly refer to this as *numerical dispersion*. We note that as Δx and Δt goes to zero while $|u_0|\Delta t \leq \Delta x$, that is, the stability condition is observed, then $c \to u_0$ and the wave becomes non-dispersive.

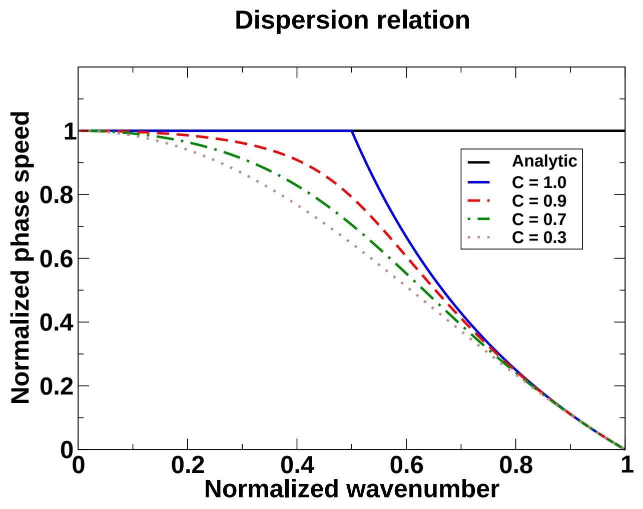


Figure 5.1: Numerical dispersion for the leapfrog scheme. The curves depicts the numerical phase speed c^* as a function of the wavenumber α based on (5.32) for various values of the Courant number $C = |u_0|\Delta t/\Delta x$. The vertical axis indicates the phase speed normalized by the advection speed u_0. The horizontal axis indicates the wavenumber normalized by inverse space increment Δx or $\pi/\Delta x$. The analytic dispersion curve is just a straigt line corresponding to the phase speed $c^* = u_0$, that is $c^*/u_0 = 1$. As the wavenumber increases (that is the wavelength decreases) the numerical phase speed deviates more and more from the correct analytic phase speed for all values of the Courant number. For wavenumbers which gives $\alpha\Delta x/\pi > 0.5$, that is for waves of wavelengths $\lambda < 4\Delta x$ the slope of the curves indicates that the group velocity is negative. Thus for waves of wavelengths shorter than $4\Delta x$ the energy is propagating in the opposite direction of the waves.

This is visualized in Figure 5.1 showing the normalized numerical phase speed as a function of the normalized wave number for various Courant numbers. The figure clearly exhibit the dispersive nature of the leapfrog scheme. By inspection of Figure 5.1 we also notice, as was first discerned by *Grotjhan and O'Brien* (1976), that the dispersiveness gets worse the less the Courant number. In fact from (5.28) follows that c_g is zero when

$$\alpha \partial_\alpha c = -c. \tag{5.33}$$

By use of (5.32) it follows that this is true regarding c^* for wavenumbers satisfying $\cos(\alpha\Delta x) = 0$, that is, for wavenumbers

$$\alpha_m = \frac{1}{2}(2m-1)\pi, \quad m = 1, 2, \ldots \tag{5.34}$$

We therefore conclude that the longest wave for which $c_g = 0$ is for $m = 1$, that is $\lambda = 2\pi/\alpha_1 = 4\Delta x$. As displayed in Figure 5.1 this corresponds to the normalized wavenumber being equal to 0.5. For higher wavenumbers, that is, waves whose wavelength are shorter than $4\Delta x$ the group velocity actually becomes negative. Thus if the wave is poorly resolved the leapfrog scheme will actually propagate energy opposite to the wave itself. This is clearly unphysical and must be avoided.

It is therefore extremely important that the scales that dominates the property that is advected is well resolved. Let the dominant wavenumber be α. Then by looking at Figure 5.1 Δx must be chosen so that $\alpha\Delta x < 0.3\pi$ for Courant numbers close to one and even less if the Courant number is smaller.

5.6 The initial problem in CTCS schemes

Another problem is that the leapfrog scheme requires more than on boundary condition in time. This is referred to as the initial value problem. We have already touched upon this problem in Section 4.2 regarding the diffusion problem. Thus the question arises how to start the time marching procedure when employing the leapfrog scheme. To our rescue comes the Euler scheme[7]. Although, as shown in Section 5.2, it is unconditionally unstable, we may nevertheless make use of the Euler scheme when applying it to a single time step only.

Thus we start by using the scheme

$$\theta_j^1 = \theta_j^0 - u_0 \frac{\Delta t}{2\Delta x}\left(\theta_{j+1}^0 - \theta_{j-1}^0\right) . \tag{5.35}$$

For the time level 2 and onwards we then use the leapfrog scheme (5.18). The step (5.35) and more generally the forward in time, centered in space scheme is usually referred to as the Euler scheme. We emphasize that although the Euler scheme is unconditionally unstable it does not ruin the solution when applied for one time step only. It may even be used from time to time to avoid the unphysical mode inherent in the leapfrog scheme (cf. Section 5.7).

5.7 Computational modes and unphysical solutions

In Section 5.5 above we showed that the leapfrog scheme contains numerical dispersion. Although this property, as it name suggests, have a physical interpretation it is nonetheless results of the employed scheme and hence unphysical or a numerical artifact. Later (Section 10.6 on page 177) we will show that there exists a method called the spectral method which avoids the dispersion inherent in the leapfrog scheme. However, this method is applicable only for global models. For limited area models there is no such remedy available.

We will now show that the leapfrog scheme contains yet an additional unphysical property, namely false *computational modes* or numerical modes that leads to unphysical solutions. Let

[7]Any forward in time scheme may be used for this purpose

us start by assuming that the initial condition is similar to that given in (5.9), that is, a single harmonic wave with wave number α_0 and amplitude Θ_0. The true solution is then given by

$$\theta = \Theta_0 e^{i\alpha_0(x-u_0t)}, \tag{5.36}$$

that is, a single monochromatic wave with wavenumber α_0 propagating with the phase speed u_0 in the *positive* x direction. Recall that any good function can be written in terms of an infinite sum of waves (cf. Section 4.3 on page 43). Thus if we are able to find the solution for one monochromatic wave we find the solution to any arbitrary initial condition by summing up (in wavenumber space) over all possible wavenumbers.

To reveal that the leapfrog scheme actually contains two modes we will solve it analytically using the monochromatic wave of wavenumber α_0 and amplitude A as our initial condition, that is, initially

$$\theta_j^0 = Ae^{ij\alpha_0\Delta x}; \quad \forall j \tag{5.37}$$

We start by recalling that in terms of the growth factor $G = \Theta_{n+1}/\Theta_n$, the analytic solution to any numerical scheme is

$$\theta_j^n = \Theta_n e^{ij\alpha\Delta x} = G\Theta_{n-1}e^{ij\alpha\Delta x} = G^n\Theta_0 e^{ij\alpha\Delta x} \tag{5.38}$$

where the last equality follows by induction. Furthermore we recall from the stability analysis of the leapfrog scheme (cf. Section 4.4 on page 44) that the growth factor for this scheme has two solutions. Thus the full solution to the leapfrog scheme is

$$\theta_j^n = (G_1^n\Theta_1 + G_2^n\Theta_2)e^{ij\alpha\Delta x}, \tag{5.39}$$

where Θ_1 and Θ_2 are as yet two unknown constants. The two possible growth factors are as given by (5.22) on page 66. For our purpose we rewrite these two solutions as

$$G_1 = P^*, \quad \text{and} \quad G_2 = -P, \tag{5.40}$$

where

$$P = \sqrt{1-\lambda^2} + i\lambda \quad \text{and} \quad P^* = \sqrt{1-\lambda^2} - i\lambda \tag{5.41}$$

are complex conjugate numbers and λ is given by (5.21). Note that we have assumed that the CFL condition for stability is satisfied so that $\sqrt{1-\lambda^2}$ is a real number. Recall that any complex number $B = a + ib$ may be written $B = |B|e^{i\phi}$ where $|B| = \sqrt{a^2+b^2}$ and $\phi = \arcsin(b/|B|)$. In our case $|P| = |P^*| = 1$ and hence we get

$$P^* = e^{\phi_1} \quad \text{where} \quad \phi_1 = \arcsin(-\lambda) = -\arcsin\lambda, \tag{5.42}$$

$$P = e^{\phi_2} \quad \text{where} \quad \phi_2 = \arcsin\lambda. \tag{5.43}$$

From (5.32) follows that

$$\arcsin\lambda = \arcsin(u_0\frac{\Delta t}{\Delta x}\sin\alpha\Delta x) = \alpha c^*\Delta t. \tag{5.44}$$

Thus we get

$$G_1 = e^{-i\alpha c^* \Delta t} \quad \text{and} \quad G_2 = -e^{i\alpha c^* \Delta t} \tag{5.45}$$

Substituting these solutions into (5.38) yields

$$\theta_j^n = \Theta_1 e^{i\alpha(j\Delta x - c^* n\Delta t)} + (-1)^n \Theta_2 e^{i\alpha(j\Delta x + c^* n\Delta t)}, \tag{5.46}$$

Invoking the one and only initial condition (5.37) we finally obtain

$$\theta_j^n = (A - \Theta_2) e^{i\alpha_0(j\Delta x - c^* n\Delta t)} + (-1)^n \Theta_2 e^{i\alpha_0(j\Delta x + c^* n\Delta t)}. \tag{5.47}$$

We note that the first term on the right-hand side of (5.47) is a wave propagating in the positive x direction with phase speed c^* and amplitude $A - \Theta_2$. In contrast the second term on the right-hand side is a wave propagating in the *negative* x direction, but with the same phase speed. Furthermore, the latter has an amplitude alternating between $\pm\Theta_2$. We conclude that by making use of the leapfrog scheme, the finite difference solution contains two solutions in the form of two waves propagating in opposite directions with the dispersive phase speed given by (5.32). In contrast the true solution (5.10) to the advection equation contains only one wave that propagates in the *positive* x direction with phase speed u_0. The two waves that occur in the finite difference solution is due to the fact that the leapfrog scheme is of second order. As such it requires us to give two boundary conditions in time. We are therefore in need of an additional condition to determine the remaining constant, which is not at our disposal.

The latter problem is associated with the initial boundary problem discussed in Section 5.6 above, but manifests itself in the unknown constant Θ_2 in (5.47). One remedy suggested in Section 5.6 is to apply an Euler step as the first step. Note that this was suggested to start the time marching problem, otherwise we had to assign a value to θ at the time step prior to the initial time. We recall that the Euler step (5.35) is

$$\theta_j^1 = \theta_j^0 - u_0 \frac{\Delta t}{2\Delta x}\left(\theta_{j+1}^0 - \theta_{j-1}^0\right). \tag{5.48}$$

Substituting the initial condition (5.9) into (5.48) and (5.47) (letting $n = 1$) the additional condition becomes[8]

$$A[1 - i\sin(\alpha_0 c^* \Delta t)] = (A - \Theta_2)e^{-i\alpha_0 c^* \Delta t} - \Theta_2 e^{i\alpha_0 c^* \Delta t} \tag{5.49}$$

and hence that

$$\Theta_2 = -A\frac{1 - \cos(\alpha_0 c^* \Delta t)}{2\cos(\alpha_0 c^* \Delta t)}. \tag{5.50}$$

The complete finite difference solution is then

$$\theta_j^n = A\frac{1 + \cos(\alpha_0 c^* \Delta t)}{2\cos(\alpha_0 c^* \Delta t)} e^{i\alpha_0(j\Delta x - c^* n\Delta t)} + (-1)^{n+1} A\frac{1 - \cos(\alpha_0 c^* \Delta t)}{2\cos(\alpha_0 c\Delta t)} e^{i\alpha_0(j\Delta x + c^* n\Delta t)}. \tag{5.51}$$

We observe, as mentioned in the previous Section 5.5, that as long as the stability criteria (5.25) is satisfied then $c^* \to u_0$ when Δt and Δx goes to zero independently. Under these circumstances

$$\frac{1 + \cos(\alpha_0 c^* \Delta t)}{2\cos(\alpha_0 c^* \Delta t)} \to 1 \quad \text{and} \quad \frac{1 - \cos(\alpha_0 c^* \Delta t)}{2\cos(\alpha_0 c\Delta t)} \to 0. \tag{5.52}$$

[8]Note that from (5.31) follows $u_0 \frac{\Delta t}{\Delta x}\sin(\alpha_0 \Delta x) = \sin(\alpha_0 c^* \Delta t)$

Thus as we allow $\Delta t \to 0$ and $\Delta x \to 0$ the first term on the right-hand side of (5.51) approaches the true solution while the second term vanishes. We therefore conclude that the second term is an unphysical or computational mode while the first term is the physical mode. The occurrence of the two modes in the leapfrog scheme are sometimes referred to as *time splitting* in the literature.

It is of utmost importance that we do control the computational mode so as not to create unphysical solutions. Unless we do it may create noise in our calculations, particularly true since it alternates between attaining positive and negative values. Moreover, in extreme cases it may even generate numerical instabilities.

5.8 How to get rid of the computational mode: The Asselin filter

The simplest way to get rid of the numerical or unphysical mode in the leapfrog scheme is from time to time to replace it with a forward in time scheme, e.g., the Euler scheme (cf. Equation 5.13 on page 63). Although the Euler scheme is numerically unstable, we may still apply it for single time steps without destroying the stability of the leapfrog scheme or any other stable schemes.

Another method, originally suggested by *Robert* (1966) and further developed by *Asselin* (1972), is to apply a time filtering technique. Since the computational mode alternates between positive and negative values from time step to time step it is obvious that a time filter will work the trick. We start by remarking that in general a time filter, invoking only the neighboring time levels, may be written

$$\overline{\theta}(x,t) = \gamma\theta(x,t+\Delta t) + (1-2\gamma)\theta(x,t) + \gamma\theta(x,t-\Delta t), \tag{5.53}$$

where $\overline{\theta}(x,t)$ is the filtered function and γ is a weighting parameter. By use of the notation introduced in Section 2.9 we get

$$\overline{\theta}_j^n = \theta_j^n + \gamma[\theta_j^{n+1} - 2\theta_j^n + \theta_j^{n-1}]. \tag{5.54}$$

Note that if $\gamma = 0$ we retrieve the original function, while for $\gamma = \frac{1}{4}$ the filter is the standard 1-2-1 filter,

$$\overline{\theta}_j^n = \frac{1}{4}[\theta_j^{n+1} + 2\theta_j^n + \theta_j^{n-1}], \tag{5.55}$$

which gives twice the weight to the mid time level n. We also emphasize that the last term on the righ-hand side of (5.54) is a diffusion term with a diffusion coefficient $\gamma\Delta x^2$. Thus to apply a filter is akin to add a diffusion term.

To investigate the properties of the filter it is common to use the so called *response function* defined by

$$R(\gamma) = \frac{\overline{\theta}_j^n}{\theta_j^n} \tag{5.56}$$

As before we study one single period or frequency ω only. To this end we represent the function θ by its Fourier component in time, or

$$\theta_j^n = \hat{\theta}_j e^{i\omega n\Delta t}. \tag{5.57}$$

Substituting this into (5.54) we get

$$R(\gamma) = 1 - 2\gamma + 2\gamma \cos \omega \Delta t. \tag{5.58}$$

For the 1-2-1 filter, that is, for $\gamma = \frac{1}{4}$, the response function becomes

$$R(\frac{1}{4}) = \frac{1}{2}(1 + \cos \omega \Delta t) \tag{5.59}$$

Since the period is $T = 2\pi/\omega$ we notice that for the standard 1-2-1 filter $R = 0$ for waves of period $T = 2\Delta t$, while $R = \frac{1}{2}$ for waves of period $T = 4\Delta t$. Therefore waves or noise on the Nüquist frequency, that is, waves of periods $2\Delta t$, vanish. Since the unphysical mode inherent in the leapfrog scheme alternates between attaining negative and positive values from time step to time its dominate waveperiod is exactly $2\Delta t$. The 1-2-1 filter is therefore a perfect filter in order to get rid of the computational mode inherent in the leapfrog scheme. Moreover, the noise of scales close to slightly longer periods, say $4\Delta t$, is damped to half of their original energy, while the effect on the longer periods are minimal. It was these advantageous properties of the the 1-2-1 filter that lead *Robert* (1966) and *Asselin* (1972) to suggest to use this method to damp the computational mode inherent in the leapfrog scheme.

In practice we do this as follows. Let us assume that the filtered solution have been determined for time level $n - 1$, that is, assume that $\overline{\theta}_j^{n-1}$ is known and has been stored. Let us furthermore assume that the unfiltered value at time level n, that is, θ_j^n, has been stored as well. Applying the leapfrog scheme (5.18) to compute the unfiltered value at the new time time level $n + 1$, that is, θ_j^{n+1}, using the unfiltered values θ_j^n at time level n and the filtered values $\overline{\theta}_j^{n-1}$ at time level $n - 1$ we get

$$\theta_j^{n+1} = \overline{\theta}_j^{n-1} - u_0 \frac{\Delta t}{\Delta x}(\theta_{j+1}^n - \theta_{j-1}^n). \tag{5.60}$$

Next we compute the filtered values at time level n using the filter (5.54), that is,

$$\overline{\theta}_j^n = \theta_j^n + \gamma \left[\theta_j^{n+1} - 2\theta_j^n + \overline{\theta}_j^{n-1} \right]. \tag{5.61}$$

We may then advance to the next time level reusing (5.60).

We empahsize though, as referred to, that applying a time filter like the Asselin filter produces numerical diffusion and hence impacts the numerical stability[9]. It is possible to show that while the numerical diffusion increases with increasing values of the weighing parameter γ the critical value for stability decreases. The latter implies that the stability criterion becomes more strict and that we have to diminish the time step Δt. This fact entails that although we wish to employ the simple 1-2-1 filter since it exactly kills the computational mode, it becomes unstable unless we decrease the time step. Decreasing the time step in turn means that our computations becomes less efficient. It is therefore common to apply a lower value for the weighing function, say $\gamma = 0.08$. Note that even a weak Asselin filter eventually modifies the longer wave periods by diffusion. Hence we must apply the Asselin filter with some care and not necessarily for every time step.

[9]Since we in practise use (5.60) the diffusion term is not as straightforward as alluded to after (5.54)

5.9 The upstream scheme

One of the unwanted properties of the leapfrog scheme is that it is inherently dispersive (cf. Section 5.5) causing it to generate negative tracer concentrations, even though concentrations are positive definite quantities by definition. A scheme that conserves the positive definite nature of concentrations is the so called *upwind* or *upstream* scheme. Due to this fact it was quite popular early on and is quite common even today. As the name indicates the scheme make use of information exclusively from points upstream to calculate the value at the new time level. Thus if $u_0 \geq 0$ then it uses information from the point x_{j-1}, and if $u_0 < 0$ is uses information from points x_{j+1}. The upwind scheme is a first order, two time level scheme, that is, it is forward in time and one-sided in space,. Hence the truncation error is $\mathcal{O}(\Delta t) + \mathcal{O}(\Delta x)$. Thus, again using Taylor series expansions, we get

$$\theta_j^{n+1} = \theta_j^n - |u_0| \frac{\Delta t}{\Delta x} \left\{ \begin{array}{lcl} \theta_j^n - \theta_{j-1}^n & ; & u_0 \geq 0 \\ \theta_j^n - \theta_{j+1}^n & ; & u_0 < 0 \end{array} \right. . \tag{5.62}$$

The scheme is conditionally stable under the CFL condition, that is, $|u_0|\Delta t \leq \Delta x$ or $\Delta t \leq \Delta x/|u_0|$ (cf. Exercise 2 on page 90). We note for later convenience that (5.62) may be written as

$$\theta_j^{n+1} = (1-C)\theta_j^n + C \left\{ \begin{array}{lcl} \theta_{j-1}^n & ; & u_0 \geq 0 \\ \theta_{j+1}^n & ; & u_0 < 0 \end{array} \right. . \tag{5.63}$$

where C is the Courant number as defined in (5.26).

One of the major advantages of the upwind scheme is that it conserves the fact that tracer concentration is a positive definite quantity. Furthermore we observe that it is a consistent scheme since it is derived using Taylor series. Being a two level scheme it also works fast end efficient on the computer. Despite of these circumstances the upwind scheme has one major drawback. It contains, as detailed in Section 5.15 on page 85, what we refer to as *numerical diffusion*. Depending on the choices we make regarding the time step and the space increment the numerical diffusion may be large and sometimes larger than the actual physical diffusion of the original problem. It therefore tends to even out the solution artificially as time progresses. In particular areas where large gradients appear, e.g., frontal areas, are prone to artificial diffusion. Thus fronts are diffused which in turn inhibits realistic baroclinic instability processes.

In addition we observe that the upwind scheme has truncation errors that are first order in time and space, that is, its accuracy is $\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x)$, which is one order of magnitude less than the leapfrog scheme. Because of these rather disadvantageous properties this author does not recommend the use of the upstream scheme alone. Its usefulness is that it forms the basis for more complicated schemes, e.g., the Lax-Wendroff scheme and the flux correction schemes.

5.10 The diffusive scheme

As shown in Section 5.2 the FTCS scheme applied to the advection equation gives an unconditionally unstable scheme. In an attempt to avoid this numerical instability, but retain a forward in time scheme it was early on suggested to replace θ_j^n in (5.13) by $\frac{1}{2}(\theta_{j+1}^n + \theta_{j-1}^n)$. Thus we get

$$\theta_j^{n+1} = \frac{1}{2}(\theta_{j+1}^n + \theta_{j-1}^n) - u_0 \frac{\Delta t}{2\Delta x}\left(\theta_{j+1}^n - \theta_{j-1}^n\right). \tag{5.64}$$

To be useful to us we must satisfy ourselves that the scheme is stable and consistent. The latter is not obvious in this case since the scheme is no longer based on Taylor series.

To analyze its consistency we first subtract θ_j^n on both sides of (5.64) to get

$$\theta_j^{n+1} - \theta_j^n = \frac{1}{2}(\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n) - u_0 \frac{\Delta t}{2\Delta x}\left(\theta_{j+1}^n - \theta_{j-1}^n\right). \tag{5.65}$$

Then using Taylor series we observe that the terms on the left-hand side becomes

$$\theta_j^{n+1} - \theta_j^n = \partial_t \theta|_j^n \Delta t + \frac{1}{2}\partial_t^2 \theta|_j^n \Delta t^2 + \mathcal{O}(\Delta t^3) \tag{5.66}$$

while the first term on the right-hand side become

$$\frac{1}{2}(\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n) = \frac{1}{2}\partial_x^2 \theta|_j^n \Delta x^2 + \mathcal{O}(\Delta x^4). \tag{5.67}$$

The last term on the right-hand side of (5.65) becomes

$$u_0 \frac{\Delta t}{2\Delta x}\left(\theta_{j+1}^n - \theta_{j-1}^n\right) = u_0 \Delta t \left(\partial_x \theta|_j^n + \frac{1}{6}\partial_x^3 \theta|_j^n \Delta x^2 + \mathcal{O}(\Delta x^4)\right), \tag{5.68}$$

Substituting (5.66) - (5.68) into (5.65) and rearranging terms we therefore get

$$\partial_t \theta|_j^n + u_0 \partial_x \theta|_j^n = \frac{1}{2}\frac{\Delta x^2}{\Delta t}(1 - C^2)\partial_x^2 \theta|_j^n + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t). \tag{5.69}$$

where C is the Courant number as defined in (5.26). Hence we find that the scheme is only conditionally consistent in that Δx^2 has to go to zero faster than Δt. We also observe that the source of the inconsistency is the first term on the right-hand side of (5.69). We observe that this term acts as a diffusion term with a diffusion coefficient of $\kappa = \frac{1}{2}\frac{\Delta x^2}{\Delta t}(1 - C^2)$. The finite difference equation (5.64), in which we employ finite space and time increments, therefore contains a diffusive term not present in the continuous advection equation. This is why the numerical scheme (5.64) is commonly referred to as the *diffusive scheme*.

We now turn to analyze the stability of the diffusive scheme. Again using von Neumann's method we find that the growth factor is given by

$$G = \sqrt{1 - (1 - C^2)\sin^2 \alpha \Delta x}, \tag{5.70}$$

where C is the Courant number. Since $\sin^2 \alpha \Delta x$ is a positive definite it follows that $|G| \leq 1$ iff $C \leq 1$. Hence the scheme is conditionally stable under the condition that the Courant number is less than or equal to one. We notice that this condition is exactly the same as the one we derived for the leapfrog and the upwind schemes.

5.11 The Lax-Wendroff scheme

To avoid or lessen the impact of the first order numerical diffusion inherent in the diffusive scheme, to make it consistent, and to increase its accuracy, *Richtmyer and Morton* (1967) advocated the use of a scheme based on the work of *Lax and Wendroff* (1960) now commonly referred to as the Lax-Wendroff scheme. It is a two step scheme which combines the diffusive scheme and the leapfrog scheme by first performing a diffusive step (also known as the predictor step) and then add a leapfrog step (also known as the corrective step). In summary we first derive the solution at the mid time level $t^{n+\frac{1}{2}}$ and at the mid increments in space $x_{j+\frac{1}{2}}$ (cf. the crosses marked in the dashed grid of Figure 5.2) employing the diffusive scheme. Then we use the results as the basis for the corrective step. In the latter we employ the leapfrog scheme to calculate the value of the variable at the new time level (t^{n+1}) and at the regular grid points in space, that is, x_j (cf. the circled points in the solid grid in Figure 5.2).

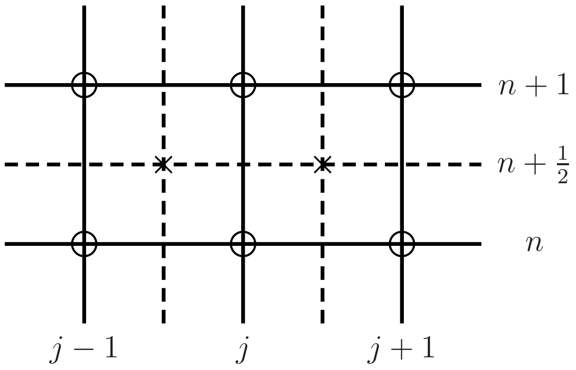


Figure 5.2: Displayed is the grid layout for the Lax-Wendroff scheme. The solid lines denote the grid through the circled x_j, t^n points. The dashed lines denote the grid through the $x_{j+\frac{1}{2}}, t^{n+\frac{1}{2}}$-points which are marked with a cross.

Thus in the predictor step we construct a forward in time, centered in space finite difference equation employing the diffusive scheme. Note that in this first step we only proceed to time level $n+\frac{1}{2}$, that is, we compute $\theta^{n+\frac{1}{2}}_{j+\frac{1}{2}}$ using (5.64). Hence we get

$$\frac{\theta^{n+\frac{1}{2}}_{j+\frac{1}{2}} - \frac{1}{2}\left(\theta^n_{j+1} + \theta^n_j\right)}{\frac{1}{2}\Delta t} + u_0 \frac{\theta^n_{j+1} - \theta^n_j}{\Delta x} = 0, \tag{5.71}$$

or

$$\theta^{n+\frac{1}{2}}_{j+\frac{1}{2}} = \frac{1}{2}\left(\theta^n_{j+1} + \theta^n_j\right) - u_0 \frac{\Delta t}{2\Delta x}\left(\theta^n_{j+1} - \theta^n_j\right). \tag{5.72}$$

We note that even though the forward in time, centered in space scheme is unstable the trick of replacing θ^n_j by half the sum of its nearest space neighbors makes the scheme stable under the condition that the Courant number is less than or equal to one as shown in Section 5.10.

The second step is to employ the leapfrog scheme (5.18) to find θ_j^{n+1}, that is, the solution at time level $n+1$ at the point j based on the values found at the mid time level. Thus we get[10]

$$\theta_j^{n+1} = \theta_j^n - \operatorname{sgn}(u_0) C \left(\theta_{j+\frac{1}{2}}^{n+\frac{1}{2}} - \theta_{j-\frac{1}{2}}^{n+\frac{1}{2}} \right), \tag{5.73}$$

where C is the Courant number. We now eliminate the dependence on $n \pm \frac{1}{2}$ and $j \pm \frac{1}{2}$ by substitution of (5.72) into (5.73). Thus we get

$$\theta_j^{n+1} = \theta_j^n - \frac{1}{2} \operatorname{sgn}(u_0) C \left(\theta_{j+1}^n - \theta_{j-1}^n \right) + \frac{1}{2} C^2 \left(\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n \right), \tag{5.74}$$

which is the Lax-Wendroff scheme

As shown by (5.74) the Lax-Wendroff scheme is an explicit scheme. Hence we expect it to be conditionally stable. To analyze its stability we use von Neumann's method. We therefore start by substituting the variables by their respective discrete Fourier components, and find next the equation for the growth factor G. After some straightforward manipulations we get

$$G = (1 - C^2 + C^2 \cos \alpha\Delta x) - i \operatorname{sgn}(u_0) C \sin \alpha\Delta x \tag{5.75}$$

which shows that the growth factor is a complex function. The magnitude of the growth factor is therefore

$$|G| = \sqrt{(1 - C^2 + C^2 \cos \alpha\Delta x)^2 + C^2 \sin^2 \alpha\Delta x}. \tag{5.76}$$

By performing some straightforward manipulations we get

$$|G| = \sqrt{1 - (1 - C)(1 + C)C^2(1 - \cos \alpha\Delta x)^2}. \tag{5.77}$$

The sign of the last term in the radical is determined by the factor $1 - C$, the remaining factors being positive definite quantities. Consequently, as long as $1 - C$ is positive $|G| \leq 1$. The Lax-Wendroff scheme is therefore, like the leapfrog, the upwind and the diffusive schemes, conditionally stable under the condition that the Courant number is less than one. Other nice features are the absence of any temporal unphysical mode as for instance present in the leapfrog scheme (cf. Section 5.7 on page 70), the lack of numerical diffusion inherent in the upwind and diffusive schemes (cf. Section 5.15 on page 85), and finally the want of any intial value problem present in the CTCS schemes (e.g., the leapfrog scheme). However, as shown in Figure 5.3, the scheme is numerical dispersive and contains some numerical dissipation. The latter is minimized by choosing $\Delta x \approx u_0 \Delta t$ which is tantamount to $C \approx 1$. It may also be minimized by choosing C close to zero, that is, $\Delta t \ll \Delta x / u_0$, but this is not to be recommnded.

We observe that the last term on the right-hand side of (5.74) looks like an FDA of the second order derivative of θ with respect to x, that is, a diffusive term, and it may therefore appear that the Lax-Wendroff scheme has some inherent numerical diffusion. Moreover if we consider the first term on the right-hand side it looks like a discrete version or FDA of the first order derivative

[10]The function sgn returns the sign of its argument. Thus $\operatorname{sgn}(u_0)$ returns the sign of the velocity u_0, that is, $u_0 = \operatorname{sgn}(u_0)|u_0|$.

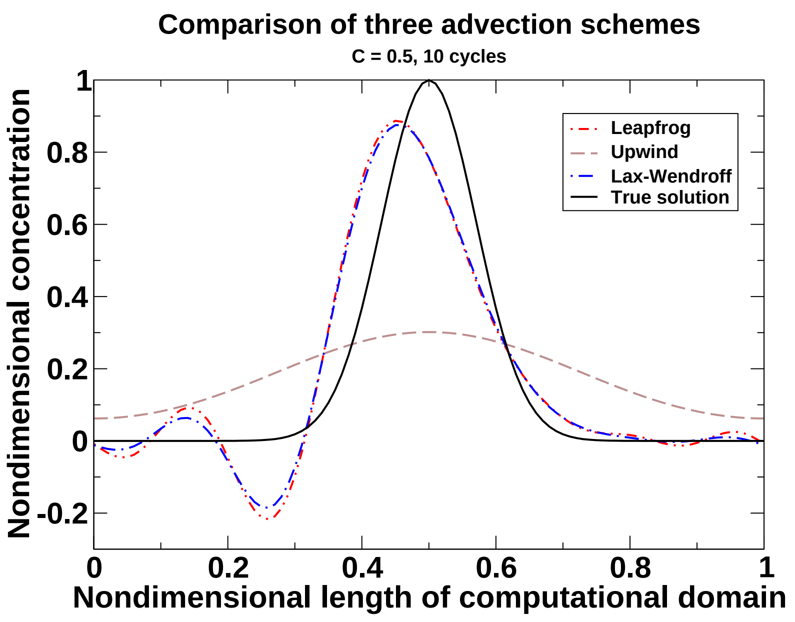


Figure 5.3: Comparison of the numerical solution to the advection equation (5.2) using the leapfrog, the upwind and the Lax-Wendroff schemes with a Courant number $C = 0.5$. The solution, using a periodic or cyclic boundary condtion, is shown after 10 cycles. The true solution is the initial bell function shown by the black solid curve. We note that both the leapfrog and the Lax-Wendroff schemes give rise to numerical dispersion, that the upwind scheme gives rise to numerical diffusion. Also the Lax-Wendroff scheme has some inherent numerical diffusion, but as depicted it is smallcompare to the diffusion inherent in the upstream scheme.

of θ. Hence it looks like an Euler scheme with a diffusion term added, or that we are solving an advection-diffison equation, that is,

$$\partial_t\theta + u_0\partial_x\theta = \kappa\partial_x^2\theta. \tag{5.78}$$

This is however not the case as shown in the next paragraph.

Another question is whether the scheme is consistent? In contrast to the leapfrog scheme and the upwind scheme, who are both derived using Taylor series, the latter is not obvious. In fact we showed in the preceding section that the diffusive scheme, which is the first step we used above is indeed inconsistent. To analyze the consistency of the Lax-Wendroff scheme we start by substituting the respective Taylor series into (5.74). The result is

$$\partial_t\theta|_j^n + u_0\partial_x\theta|_j^n = -\frac{1}{2}\left(\partial_t^2\theta|_j^n - u_0\partial_x^2\theta|_j^n\right)\Delta t + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2). \tag{5.79}$$

Since (5.2) implies that $\partial_t^2\theta = -u_0\partial_x(\partial_t\theta) = u_0^2\partial_x^2\theta$, we may neglect the first term on the right-hand side, and hence we get

$$\partial_t\theta|_j^n + u_0\partial_x\theta|_j^n = \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2). \tag{5.80}$$

We first observe that (5.2) is recovered when $\Delta x \to 0$ and $\Delta t \to 0$ independently. Hence the scheme is indeed consistent. Second we observe that the terms neglected are $\mathcal{O}(\Delta x^2)$ and $\mathcal{O}(\Delta t^2)$, and hence that the scheme is second order accurate. Moreover, and equally important we got rid of the apparent diffusive term mentioned in the preceding paragraph.

5.12 The semi-Lagrangian scheme

Some years ago, several attempts were made to construct stable time integration schemes permitting larger time steps than those limited by the CFL condition, e.g., the leapfrog, upstream and Lax-Wendroff schemes. In the early 1980s *Robert* (1981) proposed what he referred to as the *semi-Lagrangian* technique for the treatment of the advective part of the equations governing the evolution of the atmosphere and ocean. As an introduction it is useful to apply it to the one-dimensional advection equation (5.2). Later in Chapter 6 (e.g., Section 6.2 on page 103) we show how the technique is applied to solve the shallow water equations. We also use this method below (Section 5.14) to understand why the upwind and leapfrog schemes are unstable when the CFL condition is violated.

The scheme evolves from analytic methods developed to solve the breaking of the dam problem (*Stoker*, 1957, page 513), a highly non-linear problem. At that time it was referred to as the method of characteristics. In the 1960s the method was developed into a numerical scheme, (e.g., *Lister*, 1966), but since the works of, e.g., *Robert* (1981) it is commonly referred to as the semi-Lagrangian technique in NWP and NOWP models.

Let the slopes

$$\frac{D^*x}{dt} = u(x,t) \tag{5.81}$$

define special curves in the t, x space (cf. Fig. 5.4), and let us simultaneously define the special differential operator

$$\frac{D^*}{dt} \equiv \partial_t + \frac{D^*x}{dt}\partial_x. \tag{5.82}$$

Then the advection equation (5.2) may be rewritten to yield

$$\frac{D^*\theta}{dt} = 0 \tag{5.83}$$

along the slopes

$$\frac{D^*x}{dt} = u_0. \tag{5.84}$$

We commonly refer to the curves defined by (5.84) as the *characteristics* and (5.84) as the *characteristic equation*. Since the solutions to (5.2) are solutions to (5.83) as well, we often refer (5.83) as the *compatibility equation* in the sense that (5.2) and (5.83) are compatible equations. Thus either one can be used to arrive at the solution.

We observe that (5.83) tells us that θ is conserved along the characteristics (5.84). Thus if we know the solution at time $t = 0$, that is, $\theta(x, 0)$ for $0 \leq x \leq L$, the solution at any later time $t > 0$, and at any particular point x in space, is found by simply following the characteristic

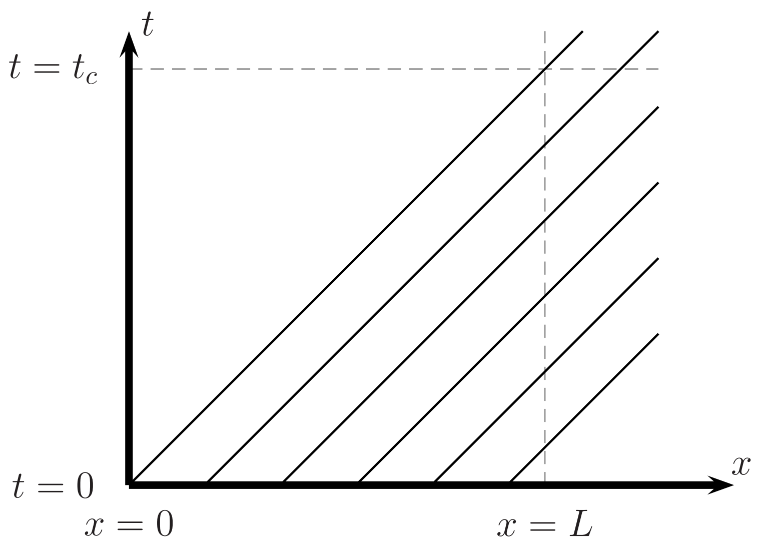


Figure 5.4: Sketch of the characteristics in the x, t plane. For $u = u_0 = \text{constant} > 0$ the characteristics are the straight lines sloping to the the right in x, t space as given by (5.81). If $x = L$ marks the end of the computational domain, then all information about the initial condition is lost for times $t > t_c$.

back from the point x, t toward its origin at the initial time $t = 0$ as illustrated in Figure 5.4. In our case with $u_0 = \text{constant}$ the characteristics are straight lines with positive slopes $1/u_0$ when $u_0 > 0$. From Figure 5.4 we may conclude that after a critical time $t = t_c = L/u_0$ all information about the initial distribution of θ is lost. Indeed for $t > t_c$ it follows that the solution in the computational domain $0 < x < L$ is determined wholly by the boundary condition at $x = 0$. Since (5.2) only contains the first derivative with respect to x, only one condition in x is allowed. The boundary at $x = L$ is therefore open in the sense that there is no boundary condition that replaces the differential equation there. Indeed the differential equation is valid also at the artificial boundary $x = L$. The physical space therefore, in principle, continues to infinity. Thus the boundary $x = L$ is a numerical boundary necessitated by the fact that any computer, however large, are limited in its capacity. This problem is especially compound for oceanographic models, since the oceanic spatial scales are small compared to the similar scales in the atmosphere. We return to this problem in Chapter 7 where we investigate details concerning conditions constraining the solutions at open boundaries.

We emphasize, as alluded to above, that since (5.2) and (5.83) are compatible, a solution to (5.83) is also automatically a solution to (5.2). We may therefore solve (5.83) employing numerical methods in which case the numerical scheme is referred to as the *semi-Lagrangian scheme*. We emphasize that the semi-Lagrangian scheme is applicable to much more complex problems and systems than the simple advection equation (e.g., *Lister*, 1966; *Røed and O'Brien*, 1983), and we return to its application to more complex systems in Chapter 6.

Our problem is thus to solve (5.83) using finite difference approximations. As is common

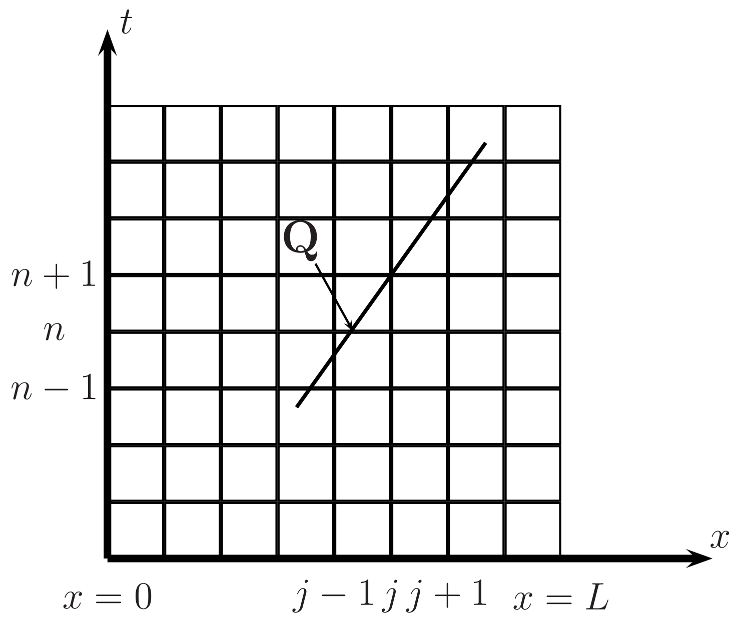


Figure 5.5: Sketch of the method of characteristics. The distance between the grid points are Δt in the vertical and Δx in the horizontal direction. The sloping solid line is the characteristic through the point $j, n+1$. It is derived from (5.81) and the slope is given by $1/u$ ($u > 0$). The point labeled Q is therefore a distance $u\Delta t$ to the left of x_j. As long as $u\Delta t < \Delta x$ then Q is located between x_{j-1} and x_j. If however $u\Delta t > \Delta x$ then the point Q is located to the left of x_{j-1}.

we divide the computational domain in the x, t space into a grid as displayed in Figure 5.5. Recalling that in our simple case (5.83) tells us that θ is conserved *along* the characteristics, a straightforward finite difference approximation to (5.83) is

$$\theta_j^{n+1} = \theta_Q^n, \tag{5.85}$$

where θ_Q^n is the potential temperature at the point Q in space where the characteristic through the grid point (x_j, t^{n+1}) crosses the time level n (Figure 5.5). We refer to this point as x_Q, and we find its position along the x-axis by making a finite difference approximation of (5.84), that is,

$$\frac{x_j - x_Q}{\Delta t} = u_0 \quad \text{or} \quad x_Q = x_j - u_0\Delta t. \tag{5.86}$$

Since we know both u_j^n and x_j, (5.86) is really an equation which determines the location of x_Q. We note that as long as $u_j^n\Delta t < \Delta x$ then x_Q is located between the grid points x_j and x_{j-1}. Since θ_j^n is known for all grid points, we may interpolate linearly between the adjacent grid points to find θ_Q^n, or the value of θ at the location x_Q at time level $t = t^n$. To this end we may use a two point linear interpolation. Thus as long as $x_{j-1} \le x_Q \le x_j$ we get

$$\theta_Q^n = \theta_{j-1}^n + \frac{\theta_j^n - \theta_{j-1}^n}{\Delta x}(x_Q - x_{j-1}). \tag{5.87}$$

Substituting x_Q from (5.86) into (5.87) we get

$$\theta_Q^n = (1-C)\theta_j^n + C\theta_{j-1}^n, \tag{5.88}$$

where

$$C = |u_0|\frac{\Delta t}{\Delta x} \tag{5.89}$$

is the Courant number. Since θ in accord with (5.83) is conserved along the characteristic, that is, (5.85) is satisfied, we finally get

$$\theta_j^{n+1} = (1-C)\theta_j^n + C\theta_{j-1}^n. \tag{5.90}$$

The advantage of this method is that if the position of x_Q does not fall between x_j and x_{j-1}, but for example between x_{j-1} and x_{j-2}, which happens if Δt is such that $\Delta x < u_0\Delta t < 2\Delta x$, then we simply approximate θ_Q^n by linearly interpolation between θ_{j-1}^n and θ_{j-2}^n, that is, we get

$$\theta_j^{n+1} = \theta_{j-2}^n + \frac{\theta_{j-1}^n - \theta_{j-2}^n}{\Delta x}(x_Q - x_{j-2}) = (2-C)\theta_{j-1}^n + (C-1)\theta_{j-2}^n. \tag{5.91}$$

Thus as long as we keep track of the position of x_Q there is no restriction on the time step Δt we may use, that is, the scheme is *unconditionally stable*. This is in contrast to the other schemes where we had to impose the CFL condition for stability.

If we replace the constant speed u_0 by a speed that is varying in time and space, say $u(x,t)$, then the characteristics are no longer straight lines. Under these circumstances we find the position x_Q by for instance a higher order finite difference approximation, say,

$$\frac{x_j - x_Q}{\Delta t} = \frac{1}{2}\left(u_j^n + u_j^{n+1}\right) \quad \text{or} \quad x_Q = x_j - \frac{1}{2}\left(u_j^n + u_j^{n+1}\right)\Delta t. \tag{5.92}$$

The position of x_Q thus found we find a θ_j^{n+1} by performing a two point interpolation, that is, as long as $x_{j-1} \le x_Q \le x_j$ we get

$$\theta_j^{n+1} = (1-C_j^n)\theta_j^n + C_j^n\theta_{j-1}^n, \tag{5.93}$$

where

$$C_j^n = \frac{1}{2}\left(u_j^n + u_j^{n+1}\right)\frac{\Delta t}{\Delta x}. \tag{5.94}$$

5.13 The implicit scheme

As for the diffusion equation we may also construct an implicit scheme for the advection equation. This is easily constructed by using a backward in time, centered in space scheme. Thus

$$\theta_j^{n+1} = \theta_j^n - u_0\frac{\Delta t}{2\Delta x}\left(\theta_{j+1}^{n+1} - \theta_{j-1}^{n+1}\right), \quad \begin{cases} j = 2(1)J-1 \\ n = 0(1)\ldots \end{cases}. \tag{5.95}$$

We observe that this scheme is $\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x^2)$, that is, first order in time and second order in space accurate. We also note that since it is implicit it requires the use of an elliptic solver for each time step. In this regard the direct elliptic solver outlined in Section 4.11 is a good choice.

To investigate the stability of (5.95) we make use of von Neumann's method. Thus we get

$$\Theta_{n+1} = \Theta_n - iC \sin \alpha \Delta x \Theta_{n+1}, \tag{5.96}$$

which results in a complex growth factor given by

$$G = \frac{1}{1 + iC \sin \alpha \Delta x}. \tag{5.97}$$

Thus the magnitude of the growth function is

$$|G| = \frac{1}{\sqrt{1 + C^2 \sin^2 \alpha \Delta x}} \tag{5.98}$$

which satisfies $|G| \leq 1$ for all finite time steps. Thus the implicit scheme, as expected, is unconditionally stable, and hence avoids the restrictive CFL condition.

Finally we note that the time step, just like the grid size, must be sufficient to resolve the typical periods of the physical problem. Commonly in atmospheric and oceanic applications the typical period is much longer than the Nüquist frequency $2\Delta t$, and hence the CFL condition in most cases puts a much more stringent requirement on Δt than the requirement of resolving the typical periods of the physical problem. Thus for most meteorological and oceanographic problems the resolution requirement is on the grid size. We notice, however, that if Δt becomes too large the growth factor will be small implying that the scheme will contain a large numerical dissipation.

5.14 Physical interpretation of the CFL condition

Figure 5.5 is drawn for $u_0 > 0$, implying that $u_0 = |u_0|$, and may be used to visualize the CFL criterion for the upwind scheme. First we note that since $u_0 > 0$ (5.86) says that $x_Q < x_j$. Moreover (5.86) also implies that the distance between x_Q and x_j is $u_0 \Delta t$. Thus if we additionally desire that $x_{j-1} \leq x_Q$ then $u_0 \Delta t \leq \Delta x$. If we compare this result with the upwind scheme as given in (5.62) we observe that for $u_0 > 0$ the information used to compute θ_j^{n+1} does originate by weighting the two points θ_j^n and θ_{j-1}^n. In fact we may rewrite (5.62) for $u_0 > 0$ to give

$$\theta_j^{n+1} = (1 - C)\theta_j^n + C\theta_{j-1}^n, \quad C = |u_0| \frac{\Delta t}{\Delta x} \tag{5.99}$$

which matches (5.90) exactly. Thus from (5.99) follows that the upwind scheme may be interpreted as the value of θ at the time level $n + 1$, that is, θ_j^{n+1}, is found by a simple weighting of the values θ_j^n and θ_{j-1}^n using the Courant number as weight. What the method of characteristics (5.90) reveals is that the latter interpretation is only valid as long as $|u_0|\Delta t \leq \Delta x$. This is exactly what the CFL criterion demands in order to make the numerical upwind scheme stable, that is, the Courant number must be less than one or that (5.25) must be satisfied.

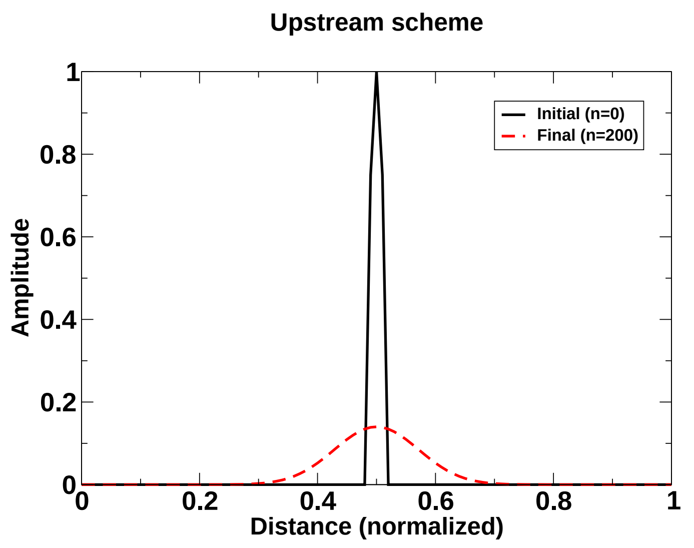


Figure 5.6: Displayed is an example of the diffusion inherent in the upwind scheme. The solid curve shows the initial distribution at time level $n = 0$, while the dashed curve (red) shows the distribution at time level $n = 200$. The Courant number is $C = 0.5$. Cyclic boundary conditions are used at the boundaries of the computational domain.

Moreover, if $|u_0|\Delta t > \Delta x$ then the characteristic through x_j at time level $n+1$ (cf. Figure 5.5) will cross the time level n to the left of x_{j-1}, that is, $x_{j-2} < x_Q < x_{j-1}$. Under these circumstances the upwind scheme will still use (5.99) to calculate θ at the new time level, that is, continue to use the weighted information using values at x_j and x_{j-1} at the previous time level. This is obviously wrong and use of (5.99) will lead to a major error. If this is allowed to continue for time step after time step the error accumulates and will finally give rise to a numerical instability.

The speed defined by Δx and Δt, that is, $\Delta x/\Delta t$, is often referred to as the signal speed of the grid or simply the grid speed. The CFL criterion (5.25) may therefore be interpreted as a condition which constrains the grid speed to be larger than the advection speed u_0. In other words, the advection speed must be small enough to let the area of dependence be between within x_{j-1} and x_{j+1} at time level n.

5.15 Numerical diffusion

Although the leapfrog scheme is neutrally stable we have just shown that it has at least one major disadvantage; it is dispersive. In particular, as displayed in Figure 5.1, this is true when the resolution is poor, that is, in areas where Δx is inadequate to resolve the dominant wavelength, that is, $\alpha\Delta x$ is not necessarily small. Also the impact of the dispersion increases with decreasing Courant number. In addition as shown in Section 5.7 the leapfrog scheme contains an unwanted computational mode giving rise to unphysical solution.

As a result the upstream scheme was for a long time the favored advection scheme. Unfortunately also the upstream scheme is far from perfect. It contains what is referred to as *numerical diffusion*. The name derives from the fact that the error made by truncating the Taylor series to first order to arrive at the finite difference equation (5.65) acts similar to physical diffusion, that is, the inherent truncation error tend to diminish differences in the tracer distribution. This is exemplified in Figure 5.6 where an initial narrow, peak like tracer distribution spreads out while being advected. In contrast the analytic solution that the numerical solution tries to mimic is one in which the initial tracer distribution is advected without change. We underscore that this does not imply that any tracer content is lost. The numerical diffusion process, just like its physical counterpart, conserves the total tracer content. What happens is that the numerical diffusion smooth out any differences in the initial tracer concentration. Thus it redistributes the initial tracer distribution while conserving the initial total tracer content. This is evident in Figure 5.6. Comparing the area under the dashed curve and the area under the solid curve they are actually the same. Note again that this redistribution is artificial and arises due to the application of the upstream scheme to solve the advection equation (5.2).

To analyze the origin of the numerical diffusion in the upstream scheme let us reconsider (5.62). We first rewrite it in terms of an advective flux defined by

$$F_j^n = \frac{1}{2}\left[(u_0 + |u_0|)\theta_j^n + (u_0 - |u_0|)\theta_{j+1}^n\right]\frac{\Delta t}{\Delta x}. \tag{5.100}$$

We note that since $u_0 = \mathrm{sgn}(u_0)|u_0|$ the last term on the right-hand side of (5.100) is zero when $u_0 \geq 0$ and the first term is zero when $u_0 < 0$. Under these circumstances $F_j^n = C\theta_j^n$ if $u_0 \geq 0$ and $F_j^n = -C\theta_{j+1}^n$ if $u_0 < 0$. Thus (5.62) is written

$$\theta_j^{n+1} = \theta_j^n - (F_j^n - F_{j-1}^n), \tag{5.101}$$

which is valid regardless of the sign of u_0. If we substitute each of the terms in (5.101) by its associated Taylor series expansion, that is,

$$\begin{array}{ll} \theta_j^{n+1} & = \theta_j^n + \partial_t\theta|_j^n\Delta t + \frac{1}{2}\partial_t^2\theta|_j^n\Delta t^2 + \mathcal{O}(\Delta t^3), \\ \theta_{j\pm 1}^n & = \theta_j^n \pm \partial_x\theta|_j^n\Delta x + \frac{1}{2}\partial_x^2\theta|_j^n\Delta x^2 + \mathcal{O}(\Delta x^3), \end{array} \tag{5.102}$$

we get

$$\partial_t\theta|_j^n + \frac{1}{2}\partial_t^2\theta|_j^n\Delta t + \mathcal{O}(\Delta t^2) = -u_0\partial_x\theta|_j^n + \frac{1}{2}|u_0|\partial_x^2\theta|_j^n\Delta x + \mathcal{O}(\Delta x^2). \tag{5.103}$$

We note that by differentiating (5.2) we get $\partial_t^2\theta|_j^n = u_0^2\partial_x^2\theta|_j^n$, and hence by rearranging terms that

$$\partial_t\theta|_j^n + u_0\partial_x\theta|_j^n = \frac{1}{2}|u_0|(\Delta x - |u_0|\Delta t)\partial_x^2\theta|_j^n + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2). \tag{5.104}$$

Defining

$$\kappa^* = \frac{1}{2}|u_0|(\Delta x - |u_0|\Delta t) \tag{5.105}$$

(5.104) may be written

$$\partial_t \theta|_j^n = -u_0 \partial_x \theta|_j^n + \kappa^* \partial_x^2 \theta|_j^n + \mathcal{O}(\Delta x^2) + \mathcal{O}(\Delta t^2). \tag{5.106}$$

Thus to second order in time and space we solve the equation

$$\partial_t \theta + u_0 \partial_x \theta = \kappa^* \partial_x^2 \theta. \tag{5.107}$$

We recognize (5.107) as an advection-diffusion equation (Chapter 3) with a diffusion coefficient κ^* given by (5.105). The terms of order $\mathcal{O}(\Delta x)$ and $\mathcal{O}(\Delta t)$ which we neglected when employing the upstream scheme therefore give rise to a diffusion. This diffusion is unphysical and an artifact that appears due to the numerical method used. It is therefore referred to as *numerical diffusion*. The strength of the numerical diffusion is determined by the diffusion coefficient defined in (5.105). We note that the diffusion is insignificant if the Courant number is close to or equals one. This corresponds to the upper limit of the CFL criterion (5.25) for stability, and is associated with a near neutrally stable scheme. The diffusion term also goes to zero when Δx and Δt goes to zero, hence showing that the upstream scheme is consistent.

5.16 Flux corrective schemes

In contrast to the second order leapfrog scheme the first order upwind (or upstream) scheme has the advantage that it is a positive definite scheme. Thus if the distribution of say $\theta(x,t)$ at some arbitrary time t is such that $\theta \geq 0$ for all x then also $\theta \geq 0$ for all later times $t = t + n\Delta t, \quad n = 1, 2, \ldots$. Another important property, as exemplified in Figure 5.6 on page 85, is that the position of the peak values are correctly propagated at any time without any dispersion. These are valuable properties well worth retaining in any scheme. The question arises if it is possible to construct a scheme that retains these properties while at the same time avoids, or at least minimizes, the numerical diffusion inherent in the scheme?

There are several schemes that offers a solution. Here we will present one of them called MPDATA[11], a scheme first suggested by *Smolarkiewicz* (1983) (see also *Smolarkiewicz and Margolin*, 1997). It's key element is to correct the diffusive flux inherent in the upstream scheme. MPDATA therefore belongs to a class of schemes known as *flux corrective schemes*. To illustrate the method we first note that the one-dimensional advection equation (5.2) may be written

$$\partial_t \theta + \partial_x F_A = 0, \tag{5.108}$$

where $F_A = u\theta$, and where the velocity u is considered a function of time and space. As shown in Section 5.15 solving (5.108) using the upstream scheme results in a solution that to second order in time and space solves (5.107), that is, an advection diffusion equation. Thus, rather than solving (5.108) we, to second order, appear to solve

$$\partial_t \theta + \partial_x (F_A + F_D^*) = 0, \tag{5.109}$$

[11]MPDATA is an abbreviation of "Multiple Positive Definite Advection-Transport Algorithm"

where $F_D^* = -\kappa^* \partial_x \theta$ is a diffusive flux with a diffusion coefficient (cf. eq. 5.105)

$$\kappa^* = \frac{1}{2}|u|(\Delta x - |u|\Delta t). \tag{5.110}$$

Thus the upwind scheme introduces an artificial or numerical diffusion represented by the flux F_D^*. To avoid this unwanted diffusion *Smolarkiewicz* (1983) suggested to solve

$$\partial_t \theta + \partial_x (F_A + F_A^*) = 0 \tag{5.111}$$

rather than (5.108). Here $F_A^* = u^* \theta$ is an artificially introduced advective flux where u^* is called the *anti-diffusion velocity* and the flux itself is called the corrective or *anti-diffusive flux*. The idea is to let F_A^* exactly oppose F_D^* introduced by the upstream scheme. We achieve this by letting $F_A^* = -F_D^*$, that is, by letting $u^*\theta = \kappa^* \partial_x \theta$. Thus we get

$$u^* \equiv \frac{\kappa^* \partial_x \theta}{\theta}. \tag{5.112}$$

We note that according to (5.112) $u^* = 0$ where $\partial_x \theta = 0$. Thus the propagation of the peak values where $\partial_x \theta = 0$ are not affected by adding the anti-diffusive flux. The positions of the extrema after time t are therefore correctly advected even though we add the corrective flux. Moreover, we observe that u^* is proportional to $\partial_x \theta$. Hence its magnitude is proportional to $|\partial_x \theta|$ while its sign follows the sign of the slope. Thus, solving the advection equation without any corrective term and where the initial distribution is a narrow bell function, we get for instance the solution illustrated by the red dashed curve in Figure 5.6 on page 85 (assuming $u = u_0 =$ constant). If we at this stage add the corrective flux F_A^* the effect is nil where the distribution has its maximum value. Thus the propagation of the maximum in the initial distribution is unaffected and its position is correctly advected. To the left of the "top" $\partial_x \theta > 0$. Hence, adding the artificial flux offsets the numerical diffusive flux there and brings the distribution closer to its initial distribution or the correct solution. To the right of the top $\partial_x \theta$ changes sign, and hence the anti-diffusive flux changes sign as well and helps to bring the solution towards the correct solution also in this area. Thus, as expected, the anti-diffusive flux helps to revoke the diffused gradients regardless of the sign of the slope. Moreover, it affects the solution most where $\partial_x \theta$ is steepest, that is, where $\partial_x^2 \theta$ changes sign. Moreover the anti-diffusive flux is just right to neutralize the artificial or numerical diffusive flux introduced when using the upwind scheme.

The numerical implementation suggested by *Smolarkiewicz* (1983) is equally simple. He suggested to perform the correction using the so called *predictor-corrector method*. It consists of performing the numerical calculation in two steps. In the first step, the *predictor step*, we compute a prediction θ^* based on the true advection equation (5.108) using a low order advection algorithm, that is, an algorithm with a truncation error of $\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x)$. Using for instance the upstream scheme for this purpose we get

$$\theta_j^* = \theta_j^n - \left(F_A|_j^n - F_A|_{j-1}^n \right), \tag{5.113}$$

where[12]

$$F_A|_j^n = \frac{1}{2}\left[(u_j^n + |u_j^n|)\theta_j^n + (u_{j+1}^n - |u_{j+1}^n|)\theta_{j+1}^n \right] \frac{\Delta t}{\Delta x}. \tag{5.114}$$

[12]The FDA for the advective flux above is valid for a non-staggered grid only. In his original paper (*Smolarkiewicz*, 1983) Smolarkiewicz used a staggered grid (cf. Section 6.3), which results in a slightly different FDA.

We note that this step retains all the advantageous properties of the upwind scheme. We know, however, that the predictor solution θ_j^*, to second order accuracy, is "infected" by a numerical diffusion which in its continuous form is represented by a diffusive flux that reads $F_D^* = -\kappa^* \partial_x \theta$, where κ^* is given (5.110). This causes the prediction θ_j^* in general, and in particular for Courant numbers less than one, to appear smoother than its analytic or continuous counterpart. This is particularly evident in areas where the initial distribution features steep gradients as for instance visualized in Figure 5.7.

In the second step, the *corrector step*, we solve the advection equation (5.111) *without* the original advection term, that is,

$$\partial_t \theta + \partial_x F_A^* = 0. \tag{5.115}$$

Also for this corrector step we apply the low order upwind scheme. Hence we get a corrected solution θ_j^{n+1} by solving

$$\theta_j^{n+1} = \theta_j^* - \left(F_A^*|_j - F_A^*|_{j-1}\right), \tag{5.116}$$

where

$$F_A^*|_j = \frac{1}{2}\left[(u_j^* + |u_j^*|)\theta_j^* + (u_{j+1}^* - |u_{j+1}^*|)\theta_{j+1}^*\right]\frac{\Delta t}{\Delta x}. \tag{5.117}$$

To ensure that the anti-diffusive velocity $u_j^* = 0$ at the peak values we may for instance use a centered scheme when computing the gradient $\partial_x \theta^*$. From (5.112) we then get

$$u_j^* = \frac{1}{4\Delta x}|u_j^n|\left(\Delta x - |u_j^n|\Delta t\right)\left(\frac{\theta_{j+1}^* - \theta_{j-1}^*}{\theta_j^* + \epsilon}\right). \tag{5.118}$$

Note that we have added, as suggested by *Smolarkiewicz* (1983), a small number ϵ in the denominator to ensure that u_j^* goes to zero when θ_j^* is zero at the same time. If we make use of (5.118) to compute the anti-diffusive velocity the gradients are re-steepened. Moreover, it does not effect the predicted solution where the predictor slopes are zero. Thus the position of the maximum is unchanged during the corrector step. As an example look at the red dashed curve curve in Figure 5.6. If this was the predictive step the largest correction will be affected along the two flanks and thus steepen the diffused gradients. Note also that since the area under the curve is conserved when employing the upwind scheme, the maximum value increases during the corrector step. The solution therefore retains all the advantageous properties of the upwind scheme, and appears to avoid the artificial smoothing of the steep gradients when applying the upwind scheme only. Moreover we observe that the corrector step makes the solution correct to $\mathcal{O}(\Delta x^2)$ and $\mathcal{O}(\Delta t^2)$. Hence MPDATA is a second order scheme that in theory compensates exactly for the artificial diffusive flux inherent in the lower order upwind scheme.

We underscore that since we employed an upstream scheme to correct the fluxes, the MPDATA method also contains some artificial diffusion to higher order. This artificial diffusion may in turn be further corrected by running a second corrector step using the corrected solution as input, which in turn contains even higher order diffusion which may be corrected by a third corrector step forming an iterative procedure where the number of iterative steps are determined by the user required accuracy only. A simpler and cheaper method (in terms of consumed computer time) is to slightly overestimate the anti-diffusive velocity by multiplying (5.118) by

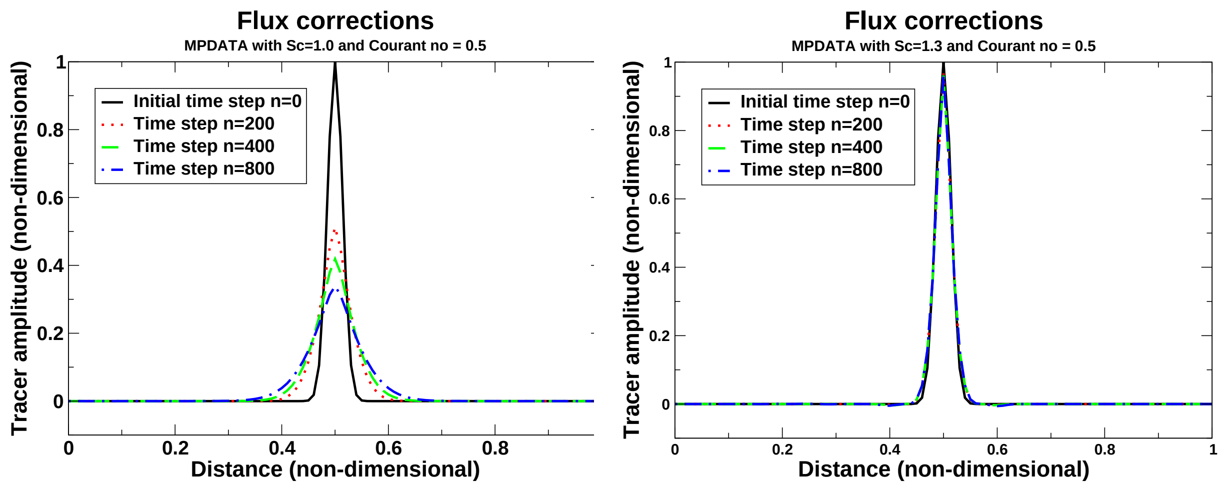


Figure 5.7: Solutions to the advection equation using the MPDATA scheme suggested by *Smolarkiewicz* (1983). Left panel corresponds to a scaling factor of 1.0 (no scaling), while the right-hand panel employs a scaling factor of 1.3. The Courant number is 0.5 in both cases. Solid, black lines show the initial value (time step $n = 0$), while the red dotted lines show the solution after 200 time steps (one cycle). The green dashed lines are after 400 time steps (two cycles) while the blue, dash-dot lines are after 800 time steps (four cycles). Periodic boundary conditions are employed. (cf. Computer problem No. 6 in the Computer Problem notes).

a scaling factor larger than one, a method already suggested by *Smolarkiewicz* (1983). Thus we redefine the anti-diffusive velocity to read

$$u_j^* = \frac{1}{4} S_c (1 - C_j^n)|u_j^n| \left(\frac{\theta_{j+1}^* - \theta_{j-1}^*}{\theta_j^* + \epsilon} \right), \tag{5.119}$$

where $S_c \geq 1$ is the scaling factor. As an example Figure 5.7 shows the solution to (5.108) employing MPDATA featuring an initial narrow Gaussian distribution. In the left-hand panel the scaling factor is set to one (no scaling), while in the right-hand panel a scaling factor of $S_c = 1.3$ is used.

Exercises

1. Show that the true solution to (5.2) is indeed (5.3). Hint: Make use of Fourier series.

2. Show that the CFL criterion for the leapfrog scheme, the diffusive scheme and the upwind scheme all are given by (5.25).
   Hint: Express the growth function in terms of $G = \sqrt{1 - (1 - C)f}$ where $C = |u_0|\Delta t/\Delta x$ is the Courant number and $f = f(C, \alpha, \Delta x)$.

3. Let us assume that the Courant number equals one. Show that under these circumstances the upstream scheme has no truncation errors. Use the method of characteristics to illustrate why this is the case.

4. Show that (5.46) is a solution to (5.18). Moreover, show that (5.51) follows from (5.46) when the initial distribution is given by (5.48), and where (5.13) is made use of to find θ_j^{-1}. Hint: Show first that $G_{1,2}$ from (5.22) may be written

$$G_1 = e^{-i\chi}, G_2 = e^{i(\chi+\pi)} \tag{5.120}$$

where χ is given by (5.47).

Chapter 6

THE SHALLOW WATER PROBLEM

Since advection and mixing (diffusion) are the foremost processes by which tracers are transported and spread in the atmosphere and ocean, we maintain that they are two of the most fundamental and important balances to treat correctly when solving the governing equations numerically. Most commonly the two processes are combined in the so called advection-diffusion equation, an equation treated in Chapter 3. Inherent in this equation is the velocity by which the tracer is advected. In the preceding chapter (Chapter 5) we assumed this velocity to be a known function and mostly treated it as a constant. In reality the velocity is part of the dynamics of the atmosphere-ocean system, and hence a function of time and space. The prognostic equation from which it is determined is the momentum equation (1.1) presented on page 2 in Chapter 1. Accordingly we need to get insight into how to solve the momentum equations numerically in addition to the advection-diffusion equation.

As alluded to in Chapter 3 an essential element of the atmosphere-ocean dynamics, that makes it stand out from ordinary fluid dynamics, is the effect of the Earth's rotation giving rise to the so called Coriolis force[1]. The appearance of this force makes it possible to obtain time-independent solutions to the momentum equation that differs from the trivial state of rest. This stationary solution is made possible by balancing the Coriolis and the pressure force as displayed in (1.39) on page 9 and we commonly refer to it as the *geostrophic balance*.

Consequently we devote this chapter to methods whereby we may determine the advection velocity by solving the momentum equation numerically. As in the preceding chapters we continue to follow Albert Einstein's mantra of making things as simple as possible, but no simpler. Thus we turn our attention to the shallow water equations derived in Section 1.5 on page 7. The equations themselves, as presented in (1.23) - (1.24) or their depth integrated versions (1.30) and (1.31) on page 8, are indeed simple, yet they include the essence of atmosphere-ocean dynamics. Most importantly they retain the possibility of a geostrophic balance. We note that the momentum equation (1.30) involves two unknowns, namely the the horizontal velocity $\mathbf{u}$ and the height

[1]Gaspard-Gustave de Coriolis or Gustave Coriolis (21 May 1792 - 19 September 1843) was a French mathematician, mechanical engineer and scientist. He is best known for his work on the supplementary forces that are detected in a rotating frame of reference, and one of those forces nowadays bears his name. (Source: Wikipedia). Note that this force is virtual in the sense that its presence is caused by our choice of coordinate system, namely one fixed to the rotating Earth

of a fluid column (or pressure). Thus we need the continuity equation (1.31) to close the system.

Even though the shallow water equations are simple, and contains a much reduced momentum equation, they are complex enough to let us appreciate the methods whereby the governing equations, including the momentum equation, are solved numerically. One of the reasons for this is that the fully three-dimensional, barotropic/baroclinic equations of motion may be described in terms of so called vertical normal modes, where each mode is governed by a set of shallow water equations[2]. Another way of illustrating this is to discretize a model into say N moving vertical layers[3] where the density is constant within each layer. For each layer we then get a set of shallow water equations to solve that are coupled to the other layers through the pressure forcing and through the vertical mixing term. In either case we end up with a set of N shallow water equations to solve. Each of them has a so called “equivalent depth” (or equivalent geopotential height) corresponding roughly to the height of the coordinate surface above ground/bottom. Note that the shallow water equations as presented in (1.30) and (1.31) correspond to a one layer model in which the surface is the Lagrangian (movable) vertical coordinate. They also represent the first and foremost vertical normal mode in the normal mode approach. To obtain the effect of baroclinicity we may introduce a second layer or a second normal mode. The interface between the two layers is then the second movable Lagrangian vertical coordinate[4].

Below we therefore study the shallow water equations as given in (1.33) and (1.34). For a one-layer model of uniform density ρ_0 they read,

$$\partial_t \mathbf{u} + \mathbf{u}\cdot\nabla_H \mathbf{u} + f\mathbf{k}\times\mathbf{u} = -\nabla_H \phi + \frac{\boldsymbol{\tau}_s - \boldsymbol{\tau}_b}{\rho_0 h} + \frac{\mathbf{X}}{h}, \tag{6.1}$$

$$\partial_t \phi + \mathbf{u}\cdot\nabla_H \phi = -\phi\nabla_H\cdot\mathbf{u}, \tag{6.2}$$

where $\phi = gh = g(H+\eta)$ is the geopotential and h is the geopotential height. In a one layer model like the present one the geopotential height is simply the depth (or height) of fluid column where $H = H(x,y)$ is the equilibrium depth and $\eta = \eta(x,y,t)$ the deviation of the top surface away from a reference geopotential level as illustrated in Figure 6.1. Furthermore $\boldsymbol{\tau}_s$ is the stress at the top surface (in the ocean referred to as the wind stress, that is, the traction the atmosphere exercises on the ocean surface), while $\boldsymbol{\tau}_b$ is the stress at the bottom, e.g., friction. We note that the wind stress then acts as an energy source, while the bottom friction acts as an energy sink, that is, dissipates energy. Finally, $\mathbf{X}$ contains lateral (horizontal) viscosity arising from lateral viscous processes, e.g., momentum diffusion (commonly referred to as eddy viscosity). Thus the eddy viscosity acts to diffuse or even out small scale motion in the atmosphere and ocean. Thus it is no different from the diffusion of tracers as treated in Chapter 4.

The set (6.1) and (6.2), consisting of the two components of the horizontal momentum equation and the continuity equation (conservation of mass), highlights the importance of geostrophy, and is a particularly useful and simple yet complicated enough set of equations to simulate many dynamical processes active in the atmosphere and ocean. Equally important, they allow us to

[2]An elegant derivation is given in the Appendix of *Lighthill* (1969)

[3]The layer interfaces are then Lagrangian surfaces and thus move up and down in the vertical in response to the dynamics. We will return to this in Chapter 8 on page 143 where we treat various vertical coordinate systems.

[4]This was first introduced by Jule Charney in his fundamental paper published in 1955 (*Charney*, 1955)

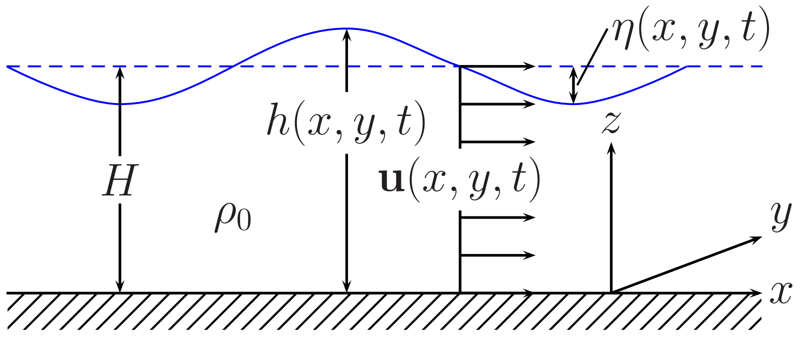


Figure 6.1: Sketched is a one layer-model of the atmosphere or ocean. Here $h = H + \eta$ is the total depth of a fluid column, where $H = H(x, y)$ is the equilibrium depth while $\eta = \eta(x, y, t)$ is the deviation of the top reference surface away from its (level) equilibrium position. The mean velocity of the fluid column is $\mathbf{u}$, while the uniform density of the layer is denoted ρ_0.

introduce methods whereby the momentum and continuity equations may be solved by numerical means, without having to concern ourselves with the thermodynamic processes. It should be stressed though that thermodynamic processes are indeed important processes in the atmosphere as well as the ocean. Their influence on the dynamics is through the active tracers like potential temperature, humidity and salinity. These tracers are in turn determined by the tracer equations, that is, by advection-diffusion equations treated in the previous chapters (Chapters 4 and 5).

In addition we note that the set (6.1) and (6.2) contains multiple dependent variables, that is, ϕ, u, and v, and hence conveniently introduces methods whereby partial differential equations containing more than one dependent variable may be solved numerically. Moreover, as is evident from (6.1) and (6.2), the shallow water equations is a set of non-linear equations. Since there is a fundamental difference between linear and non-linear systems, we split the presentation below into linear and non-linear versions of the shallow water equations. Furthermore, to highlight some salient fact of importance when solving the shallow water equations by numerical means, we also separate between non-rotating and rotating systems.

Since the mixing and friction terms are energy source and/or sink terms we may, without loss of generality, neglect these terms, as we for instance did in Section 3.5. Under these circumstances the set (6.1) and (6.2) becomes

$$\begin{aligned} \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla_H \mathbf{u} + f\mathbf{k} \times \mathbf{u} &= -\nabla_H \phi, && (6.3) \\ \partial_t \phi + \mathbf{u} \cdot \nabla_H \phi &= -\phi \nabla_H \cdot \mathbf{u}. && (6.4) \end{aligned}$$

We observe that (6.3) and (6.4) are non-linear, and that they contain three dependent variables, namely the two horizontal, depth integrated, velocity components u, v and the geopotential ϕ. We also notice that the set (6.3) and (6.4) is complete in the sense that we have three equations to solve for the three unknowns u, v, ϕ. By the same token we observe that to find a solution for the unknowns we need three initial conditions and six boundary conditions to determine the integration constants. Finally, we note that the set (6.3) and (6.4) contains three independent variables, namely x, y and t.

To repeat, the shallow water equations are not only a set of equations that describe important and fundamental aspects of the atmosphere-ocean dynamics. It is also a convenient set of equations whereby the numerical treatment of equations containing more than one dependent variable as well as several independent variables may be introduced within a geophysical fluid context.

6.1 Linearization of the shallow water equations

To linearize the system (6.3) - (6.4) we start by assuming that the dependent variables can be written in terms of a basic state plus a perturbation, that is,

$$\mathbf{u} = \bar{\mathbf{u}} + \mathbf{u}' \quad \text{and} \quad \phi = \bar{\phi} + \phi' \tag{6.5}$$

where the perturbed velocity (or basic state) is $\bar{\mathbf{u}} = \bar{u}\mathbf{i} + \bar{v}\mathbf{j}$ and the perturbed geopotential is $\bar{\phi}$. To be consistent we require that the basic state is in dynamical balance, implying that the basic state is a solution to the governing equations when the perturbations are zero. Thus from (6.3) and (6.4) follows that the basic state is a solution to the set

$$\begin{aligned} \partial_t\bar{\mathbf{u}} + \bar{\mathbf{u}}\cdot\nabla_H\bar{\mathbf{u}} + f\mathbf{k}\times\bar{\mathbf{u}} &= -\nabla_H\bar{\phi}, &(6.6)\\ \partial_t\bar{\phi} + \bar{\mathbf{u}}\cdot\nabla_H\bar{\phi} + \bar{\phi}\nabla_H\cdot\bar{\mathbf{u}} &= 0 &(6.7)\end{aligned}$$

Substituting (6.5) into (6.3) - (6.4) invoking (6.6) and (6.7) we get

$$\begin{aligned} \partial_t\mathbf{u}' + \bar{\mathbf{u}}\cdot\nabla_H\mathbf{u}' + \mathbf{u}'\cdot(\nabla_H\bar{\mathbf{u}} + \nabla_H\mathbf{u}') + f\mathbf{k}\times\mathbf{u}' &= -\nabla_H\phi', &(6.8)\\ \partial_t\phi' + \bar{\mathbf{u}}\cdot\nabla_H\phi' + \mathbf{u}'\cdot(\nabla_H\bar{\phi} + \nabla_H\phi') + \bar{\phi}\nabla_H\cdot\mathbf{u}' &= -\phi'(\nabla_H\cdot\bar{\mathbf{u}} + \nabla_H\cdot\mathbf{u}') &(6.9)\end{aligned}$$

Note that the basic state variables such as $\bar{\mathbf{u}}$ and $\bar{\phi}$ are functions of the independent variables x, y and t, but are slowly varying in both space and time. Thus

$$\begin{aligned} |\partial_t\bar{\mathbf{u}}| &\ll |\partial_t\mathbf{u}'|, \quad |\nabla_H\bar{\mathbf{u}}| \ll |\nabla_H\mathbf{u}'|, \quad |\nabla_H\cdot\bar{\mathbf{u}}| \ll |\nabla_H\cdot\mathbf{u}'|,\\ |\partial_t\phi| &\ll |\partial_t\phi'|, \quad |\nabla_H\bar{\phi}| \ll |\nabla_H\phi'|. \end{aligned} \tag{6.10}$$

Thus (6.8) and (6.9) are further reduced to

$$\begin{aligned} \partial_t\mathbf{u}' + \bar{\mathbf{u}}\cdot\nabla_H\mathbf{u}' + \mathbf{u}'\cdot\nabla_H\mathbf{u}' + f\mathbf{k}\times\mathbf{u}' &= -\nabla_H\phi', &(6.11)\\ \partial_t\phi' + \bar{\mathbf{u}}\cdot\nabla_H\phi' + \mathbf{u}'\cdot\nabla_H\phi' + \bar{\phi}\nabla_H\cdot\mathbf{u}' &= -\phi'\nabla_H\cdot\mathbf{u}'. &(6.12)\end{aligned}$$

By nature a perturbation is defined as being "small". Here we define small to imply that products of the perturbations can be dropped in comparison with the perturbation itself (in a non-dimensional sense), that is,

$$\left|\frac{\mathbf{u}'^2}{\bar{\mathbf{u}}^2}\right| \ll \left|\frac{\mathbf{u}'}{\bar{\mathbf{u}}}\right|, \quad \left|\frac{\phi'^2}{\bar{\phi}^2}\right| \ll \left|\frac{\phi'}{\bar{\phi}}\right|. \tag{6.13}$$

Thus (6.11) and (6.12) become

$$\begin{aligned} \partial_t \mathbf{u} + \bar{\mathbf{u}} \cdot \nabla_H \mathbf{u} + f\mathbf{k} \times \mathbf{u} &= -\nabla_H \phi, & (6.14)\\ \partial_t \phi + \bar{\mathbf{u}} \cdot \nabla_H \phi + \bar{\phi} \nabla_H \cdot \mathbf{u} &= 0. & (6.15) \end{aligned}$$

where we have dropped the primes on the perturbations for clarity. These equations then describe the time evolution of the perturbations.

If we assume that the acceleration of the basic state, that is, $\partial_t \bar{\mathbf{u}} + \bar{\mathbf{u}} \cdot \nabla_H \bar{\mathbf{u}}$, is small compared to the Coriolis acceleration (small Rossby number, cf.Section 1.6 on page 9), it follows that the basic state must be in geostrophic balance, that is,

$$\bar{\mathbf{u}} = \frac{1}{f} \mathbf{k} \times \nabla_H \bar{\phi}. \tag{6.16}$$

Thus if the basic state is one at rest ($\bar{\mathbf{u}} = 0$) then (6.16) requires $\bar{\phi}$ to be constant, say $\bar{\phi} = \bar{\phi}_0 = gH_0$, where H_0 is a constant geopotential height. Substituting this simple basic state into (6.14) and (6.15) we arrive at a particular simple linear subset, namely,

$$\begin{aligned} \partial_t \mathbf{u} + f\mathbf{k} \times \mathbf{u} &= -\nabla_H \phi, & (6.17)\\ \partial_t \phi + c_0^2 \nabla_H \cdot \mathbf{u} &= 0, & (6.18) \end{aligned}$$

where $c_0 = \sqrt{gH_0}$ is known as the phase speed (propagation speed) of inertia-gravity waves.

Again using Einstein's mantra to keep things as simple as possible, but no simpler, we will in what follows assume that the problem is one-dimensional. To achieve this we let $\partial_y = 0$ which forces us to assume that $\partial_y f = 0$ as well. Assuming that f does not change with latitude is normally referred to as the f-plane approximation, an approximation that excludes Rossby waves (cf. page 98). The set (6.14) and (6.15) then becomes

$$\begin{aligned} \partial_t u + \bar{u} \partial_x u - fv + \partial_x \phi &= 0, & (6.19)\\ \partial_t v + \bar{u} \partial_x v + fu &= 0, & (6.20)\\ \partial_t \phi + \bar{u} \partial_x \phi + \bar{\phi} \partial_x u &= 0. & (6.21) \end{aligned}$$

while the subset (6.17) and (6.18) becomes

$$\begin{aligned} \partial_t u - fv + \partial_x \phi &= 0, & (6.22)\\ \partial_t v + fu &= 0, & (6.23)\\ \partial_t \phi + c_0^2 \partial_x u &= 0. & (6.24) \end{aligned}$$

For later reference we note that the geostrophic velocity components of the basic state are given by

$$\bar{u} = -\frac{1}{f} \partial_y \bar{\phi} \quad \text{and} \quad \bar{v} = \frac{1}{f} \partial_x \bar{\phi}, \tag{6.25}$$

respectively. Note that to ensure that the perturbation are function of x only we must for consistency require that $\bar{u} = \bar{u}(x, t)$, that is, $\partial_y \bar{u} = 0$. Thus (6.25) requires that the basic state dependence on y is such that $\bar{\phi}$ is a linear function of y.

Rossby waves

Let us for a moment assume that the time rate of change of the geopotential in (6.18) is so small that the motion is effectively divergence free, that is, $\nabla_H \cdot \mathbf{u} = 0$. We may then introduce a streamfunction ψ by defining

$$\mathbf{u} = \mathbf{k} \times \nabla_H \psi, \quad \text{or} \quad u = -\partial_y \psi \quad \text{and} \quad v = \partial_x \psi. \tag{6.26}$$

Operating on (6.17) using the operator $\mathbf{k} \cdot \nabla_H \times$, and remembering that the Coriolis parameter is a function of the latitude y, we get

$$\partial_t \zeta + \beta \partial_x \psi = 0, \tag{6.27}$$

where $\zeta = \mathbf{k} \cdot \nabla_H \times \mathbf{u} = \nabla_H^2 \psi$ is the vorticity, $\beta = \partial_y f$ is the rate of change of the Coriolis parameter with latitude (sometimes referred to as the Rossby parameter). To arrive at (6.27) we have made use of (6.26) and that $\nabla_H \cdot \mathbf{u} = 0$. We note that (6.27) is the barotropic vorticity equation whose solutions contain the *Rossby waves*. In fact if substitute for a Fourier component, that is, let

$$\psi = \psi_0 e^{i(\alpha x + ly - \omega t)}, \tag{6.28}$$

where α is the zonal wave number and l is the meridional wave number, we arrive at the well known dispersion relation for Rossby waves, that is,

$$c = \frac{\omega}{\alpha} = -\frac{\beta}{(\alpha^2 + l^2)}. \tag{6.29}$$

where c is the zonal phase speed of the wave. The zonal group velocity, as defined by (5.28), is then

$$c_g = \alpha \partial_\alpha c + c = \frac{\beta(\alpha^2 - l^2)}{(\alpha^2 + l^2)^2} \tag{6.30}$$

Thus when the zonal wave number is large compared to the meridional wave number for which the group velicity is $c_g \approx \frac{\beta}{\alpha}$ the group velocity is positive implying that while the Rossby wave itself is propagating westward the energy is propagating eastward.

6.2 Linear, non-rotating shallow water equations

We now proceed to solve the shallow water equations by use of numerical methods. To illustrate the schemes we will use and how to work out the numerical stability of the schemes we first investigate possible numerical methods to solve the non-rotating case. Thus we let $f = 0$ which exclude the possibility of having a basic state in geostrophic balance. Hence the basic state is one at rest ($\bar{u} = \bar{v} = 0$) with a constant geopotential height. Under these circumstance the set (6.22) - (6.24) reduces to

$$\begin{aligned} \partial_t u &= -\partial_x \phi, & (6.31)\\ \partial_t \phi &= -c_0^2 \partial_x u. & (6.32) \end{aligned}$$

Note that when $f = 0$ (6.23) is obsolete (v = constant = 0) and we are left with two equations for the two unknowns u and ϕ. Furthermore, substituting (6.31) into (6.32) we get

$$\partial_t^2 \phi - c_0^2 \partial_x^2 \phi = 0, \tag{6.33}$$

which is the linear wave equation. Hence the system is hyperbolic

We first explore possible analytic solutions to (6.31) and (6.32). Recall that the full solution may be represented as an infinite sum over all wavelengths, in which each Fourier component has its own amplitude. Thus each wave must be a solution of the governing equation. Hence we seek solutions of the form

$$\phi = \phi_0 e^{i\alpha(x-ct)} \quad \text{and} \quad u = u_0 e^{i\alpha(x-ct)} \tag{6.34}$$

where u_0 and ϕ_0 are arbitrary wave amplitudes different from zero and $c = \omega/\alpha$ is the wave speed. Substituting (6.34) into (6.31) and (6.32) and dividing through by the common non-zero factors we get

$$\begin{aligned} -cu_0 + \phi_0 &= 0, & (6.35)\\ c_0^2 u_0 - c\phi_0 &= 0. & (6.36) \end{aligned}$$

Solving with respect to u_0 or ϕ_0 we get the dispersion relation

$$c^2 - c_0^2 = 0. \tag{6.37}$$

Thus we get two solutions for the wave speed c, namely

$$c_{1,2} = \pm c_0, \tag{6.38}$$

implying that the solution for the geopotential[5] is

$$\phi = \phi_1 e^{i\alpha(x-c_0t)} + \phi_2 e^{i\alpha(x+c_0t)}. \tag{6.39}$$

Hence the solution is two ordinary gravity waves propagating with a constant phase speed c_0 in opposite directions.

Thus when constructing an FDA scheme to solve (6.31) and (6.32) numerically we expect the numerical solution to replicate (6.39). As in Chapter 5 there are several schemes that may be used. Here we construct and investigate three schemes, namely the CTCS (leapfrog), the forward-backward and the semi-Lagrangian scheme.

The CTCS scheme

Since (6.31) and (6.32) is a hyperbolic system (cf. Section 2.4) we expect the leapfrog scheme, that is, the centered in time, centered in space (CTCS) scheme to work well. Thus we replace the derivatives using centered FDAs for the differential terms. Hence we get

$$\begin{aligned} u_j^{n+1} - u_j^{n-1} &= -\frac{\Delta t}{\Delta x}(\phi_{j+1}^n - \phi_{j-1}^n) & (6.40)\\ \phi_j^{n+1} - \phi_j^{n-1} &= -c_0^2 \frac{\Delta t}{\Delta x}(u_{j+1}^n - u_{j-1}^n). & (6.41) \end{aligned}$$

[5]Note that u has a similar solution

Since we made use of Taylor series to derive the approximations, we know a priori that the scheme is consistent. To satisfy ourselves that the scheme is numerically stable we use von Neumann's method. Thus we start by defining the discrete Fourier components, that is,

$$u_j^n = U_n e^{i\alpha j\Delta x} \quad \text{and} \quad \phi_j^n = \Phi_n e^{i\alpha j\Delta x}. \tag{6.42}$$

Substituting (6.42) into (6.40) and (6.41) we get

$$U_{n+1} - U_{n-1} = -2i\gamma\Phi_n, \tag{6.43}$$
$$\Phi_{n+1} - \Phi_{n-1} = -2ic_0^2\gamma U_n, \tag{6.44}$$

where

$$\gamma = \frac{\Delta t}{\Delta x}\sin\alpha\Delta x. \tag{6.45}$$

Since n is only a time step counter we may rewrite (6.43) to give

$$U_{n+2} - U_n = -2i\gamma\Phi_{n+1}, \tag{6.46}$$

and similarly

$$U_n - U_{n-2} = -2i\gamma\Phi_{n-1}. \tag{6.47}$$

Subtracting (6.47) from (6.46) we get

$$U_{n+2} - 2U_n + U_{n-2} = -2i\gamma\left(\Phi_{n+1} - \Phi_{n-1}\right). \tag{6.48}$$

Finally substituting for $(\Phi_{n+1} - \Phi_{n-1})$ using (6.44) we get

$$U_{n+2} - 2\lambda U_n + U_{n-2} = 0, \quad \lambda = 1 - 2c_0^2\gamma^2. \tag{6.49}$$

Next we define a growth factor by $G \equiv U_{n+2}/U_n$. Substitution into (6.49) then results in a second order equation to solve for the growth factor. Solving with respect to G we get the two solutions

$$G_{1,2} = -\lambda \pm \sqrt{\lambda^2 - 1} = -\lambda \pm i\sqrt{1-\lambda^2}. \tag{6.50}$$

Thus requiring $\lambda^2 \leq 1$ the growth factor is complex and has an imaginary part. Under these circumstances

$$|G_{1,2}| = \sqrt{\lambda^2 + 1 - \lambda^2} = 1, \tag{6.51}$$

and the scheme is *neutrally stable*. This is as expected since we employ a CTCS scheme to solve for a hyperbolic system. Recall that this result depends on the fact that the radical in (6.50) is positive definite. Thus neutral stability is ensured if and only if $-1 \leq \lambda \leq 1$. While the right-hand inequality is trivially satisfied, the left-hand inequality requires

$$c_0^2\left(\frac{\Delta t}{\Delta x}\right)^2 \leq 1 \quad \Rightarrow \quad C = c_0\frac{\Delta t}{\Delta x} \leq 1, \quad \text{or} \quad \Delta t \leq \frac{\Delta x}{c_0}, \tag{6.52}$$

where C is the Courant number[6]. Since the scheme is neutrally stable it contains no numerical dissipation and hence no energy loss, provided the Courant number is less than or equal to one. We observe that the velocity entering the the Courant number in (6.52) is the phase speed c_0 rather than the advection velocity u. The phase speed is normally much larger than the advection speed. In fact a typical advection velocity or current speed in the ocean is 0.1 m/s, while the internal wave phase speed is typically of order 1 m/s, that is, one order of magnitude larger. Similar, in the atmosphere a typical wind speed is 10 m/s, while the phase speed is typically of order 100 m/s. Hence the constraint on the time step as displayed in (6.52) gives a much more stringent constraint on the time step for the momentum equation than for the tracer (advection) equation.

We note that if we define the growth factor as $G' \equiv U_{n+1}/U_n$, as we normally do, then $G' = \pm\sqrt{G}$. We then get the four solutions

$$G'_{m,n} = (-1)^m \sqrt{G_n} = (-1)^n \sqrt{\lambda - (-1)^n i\sqrt{1-\lambda^2}}, \quad m, n = 1, 2. \tag{6.53}$$

Recalling the formula for a square root of a complex number[7] we get

$$|G'_{m,n}| = \sqrt{G_n} = \sqrt{\frac{1+\lambda}{2} + \frac{1-\lambda}{2}} = 1, \tag{6.54}$$

as expected. Hence whenever we get a fourth order equation to solve for the growth factor we may reduce it to a second order equation by defining the growth factor as $G \equiv U_{n+2}/U_n$ instead of $G \equiv U_{n+1}/U_n$. The result in terms of the stability condition is always the same.

We recall from Sections 5.5 and 5.7 that the leapfrog scheme applied to the advection equation contained numerical dispersion and unphysical modes. The question therefore arises if this carries over when applying the leapfrog scheme to the shallow water equations. To investigate the numerical dispersion we let

$$u_j^n = U_0 e^{i\alpha(j\Delta x - cn\Delta t)} \quad \text{and} \quad \phi_j^n = \Phi_0 e^{i\alpha(j\Delta x - cn\Delta t)}. \tag{6.55}$$

Substituting (6.55) into (6.40) and (6.41) we get

$$\sin(\alpha c\Delta t)U_0 - \gamma\Phi_0 = 0, \tag{6.56}$$
$$-c_0^2\gamma U_0 + \sin(\alpha c\Delta t)\Phi_0 = 0, \tag{6.57}$$

and hence that

$$c_{1,2} = \pm\frac{1}{\alpha\Delta t}\arcsin(\gamma c_0) = \pm\frac{1}{\alpha\Delta t}\arcsin(C\sin\alpha\Delta x), \tag{6.58}$$

where $C = c_0\Delta t/\Delta x$ is the Courant number. Thus the leapfrog scheme applied to the shallow water equations contains a numerical dispersion similar to the one we deduced for the advection

[6]Recall that c_0 is defined as a positive definite quantity. Thus we do not use the absolute value in the definition of the Courant number as we did for the advection problem.

[7]$\sqrt{a+ib} = \pm(a' + ib')$ where $a' = \sqrt{\frac{a+\sqrt{a^2+b^2}}{2}}$ and $b' = \operatorname{sgn}(b)\sqrt{\frac{-a+\sqrt{a^2+b^2}}{2}}$

equation. The main difference is that the advection velocity is replaced by the phase speed. We also note that we retrieve the results given in the analytic dispersion relation, namely that we have two waves propagating in opposite directions. For a more detailed investigation the reader is referred to *Grotjhan and O'Brien* (1976).

Regarding the unphysical mode this may be investigated much the same way as we did in Section 5.7, with the same result. Thus we note that the leapfrog scheme applied to the shallow water equations contains unphysical or artificial modes. The appearances of these modes may be avoided by for instance using the Asselin filter as detailed in Section 5.8.

The forward-backward scheme

In contrast to the advection equation the shallow water equations consist of two equations with two unknowns. Thus we are in a position to use different time schemes for the two equations. For instance, as first suggested by *Sielecki* (1968), we may use a forward in time, centered in space scheme for (6.31) while using a backward in time, centered in space scheme for (6.32), that is,

$$\begin{aligned} u_j^{n+1} - u_j^n &= -\frac{\Delta t}{2\Delta x}(\phi_{j+1}^n - \phi_{j-1}^n), \qquad (6.59)\\ \phi_j^{n+1} - \phi_j^n &= -c_0^2\frac{\Delta t}{2\Delta x}(u_{j+1}^{n+1} - u_{j-1}^{n+1}). \qquad (6.60)\end{aligned}$$

We note that this scheme is forward in time and hence of first order accuracy in time. We also note that it is a two level scheme in the sense that we only have to carry two time levels in memory at any time. This is in contrast to the CTCS scheme which is a three level scheme. Finally we note the factor 2 in the denominators on the right-hand sides of (6.59) and (6.60). It crops up since we switched to a forward in time scheme.

Again we have used Taylor series expansions to construct the scheme, and hence it is consistent. To investigate the stability we use von Neumann's method. Thus we first substitute the variables using the discrete Fourier components (6.42). We then get

$$\begin{aligned} U_{n+1} - U_n &= -i\gamma\Phi_n, \qquad (6.61)\\ \Phi_{n+1} - \Phi_n &= -ic_0^2\gamma U_{n+1}, \qquad (6.62)\end{aligned}$$

where γ is as given by (6.45). Using the same "trick" as we did to arrive at (6.49) and defining the growth factor by $G = U_{n+1}/U_n$ we get

$$G^2 - 2\lambda_{FB}G + 1 = 0, \quad \lambda_{FB} = 1 - \frac{1}{2}c_0^2\gamma^2, \qquad (6.63)$$

and hence that

$$G_{1,2} = \lambda_{FB} \pm i\sqrt{1 - \lambda_{FB}^2}. \qquad (6.64)$$

We note that iff $\lambda_{FB}^2 \leq 1$ then $|G_{1,2}| = 1$ and hence the scheme, like the CTCS scheme, has no dissipation. The energy is therefore conserved despite the fact that the scheme looks as if it is a semi-implicit scheme. We underscore that the scheme is stable if and only if $\lambda_{FB}^2 \leq 1$ is true. We therefore must require that $C = c_0 \frac{\Delta t}{\Delta x} \leq 2$, or that the Courant number is less than or equal to two. The condition for stability of the forward-backward scheme is therefore less stringent on the time step than the leapfrog (CTCS) scheme, in fact,

$$\Delta t \leq \frac{2\Delta x}{c_0}. \tag{6.65}$$

The semi-Lagrangian scheme

As we did for the advection equation (Section 5.14) we may also utilize the Semi-Lagrangian technique to solve the shallow water equations numerically. The task becomes slightly more complex since we need to solve (6.31) and (6.32) simultaneously, that is, two equations with two unknowns. Thus we get two compatibility equations along with two characteristic equations. We also note that when solving the advection equation using the Semi-Lagrangian technique there was one single invariant, namely the variable itself. Regarding the shallow water equation it turns out that we get two invariants that are combinations of the two variables entering (6.31) and (6.32).

To find the compatibility equations and the characteristic equations we first multiply (6.32) by an as yet unknown function λ and add the result to (6.31). We then get

$$(\partial_t + \lambda c_0^2 \partial_x)u + \lambda \left(\partial_t + \frac{1}{\lambda}\partial_x\right)\phi = 0. \tag{6.66}$$

Next we define an operator $\frac{D^*}{dt}$ such that

$$\frac{D^*}{dt} = \partial_t + \frac{D^* x}{dt}\partial_x, \tag{6.67}$$

where we require

$$\frac{D^* x}{dt} = \lambda c_0^2 = \frac{1}{\lambda}. \tag{6.68}$$

Hence we have two solutions for the unknown function λ

$$\lambda_{1,2} = \pm\frac{1}{c_0}, \tag{6.69}$$

which gives us the two characteristic equations

$$\frac{D_{1,2}^* x}{dt} = \pm c_0, \tag{6.70}$$

and the two operators $\frac{D_{1,2}^*}{dt} = \partial_t \pm c_0 \partial_x$. We therefore get two compatibility equations

$$\frac{D_1^* \mathcal{R}^+}{dt} = 0 \quad \text{along} \quad \frac{D_1^* x}{dt} = +c_0, \tag{6.71}$$

$$\frac{D_2^* \mathcal{R}^-}{dt} = 0 \quad \text{along} \quad \frac{D_2^* x}{dt} = -c_0, \tag{6.72}$$

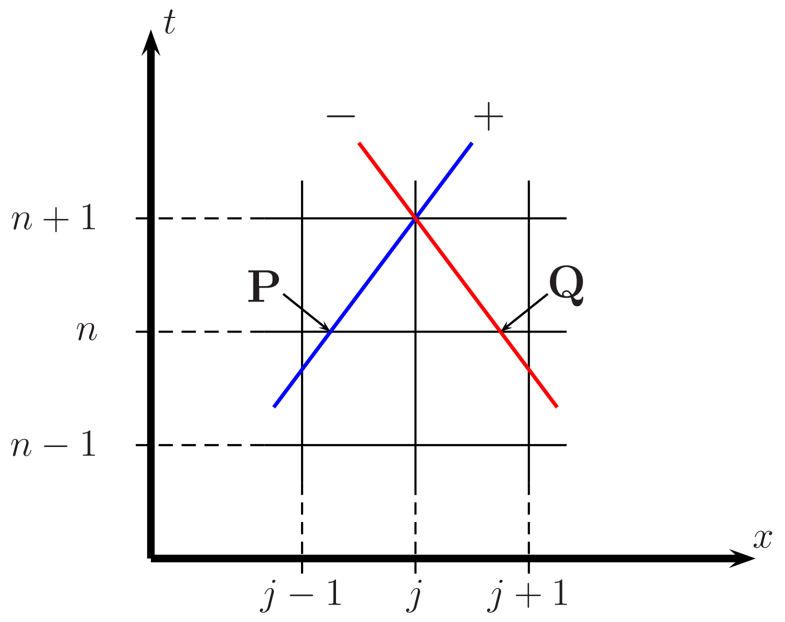


Figure 6.2: Sketch of the Semi-Lagrangian technique for the shallow water equations. The distance between the grid points are Δt in the vertical and Δx in the horizontal direction. The sloping solid, blue line (marked $+$) is the positive characteristic through the grid point $j, n+1$, while the solid, red line (marked $-$) is the negative characteristic through the same grid point. They are derived from (6.71) and (6.72) and the slopes are respectively given by $\pm 1/c_0$. The point labeled P is therefore a distance $c_0\Delta t$ to the left of x_j, while the point Q is a distance $c_0\Delta t$ to the right of x_j. As long as $c_0\Delta t \leq \Delta x$ then P is located between x_{j-1} and x_j and Q between x_{j+1} and x_j. If however $c_0\Delta t > \Delta x$ then the points Q, P are located to the left and right of respectively x_{j-1} and x_{j+1}.

where

$$\mathcal{R}^{\pm} = u \pm \frac{1}{c_0}\phi \tag{6.73}$$

are commonly referred to as the two Riemann invariants[8]. We underscore that solutions to (6.71) and (6.72) are also solutions to (6.31) and (6.32). This is indeed the very reason why we refer to (6.71) and (6.72) as the compatibility equations. We may therefore solve (6.71) and (6.72) numerically, much the same way as we outlined in Section 5.12, that is, using the Semi-Lagrangian technique.

We start by noting that the slope of the two characteristics are given by (6.70) and thus are $\pm 1/c_0$. As illustrated in Figure 6.2 the characteristics through the grid point (x_j, t^{n+1}) crosses

[8]The wording invariants is used since (6.71) and (6.72) states that the Riemann invariants are conserved along their respective characteristic. They are named after Georg Friedrich Bernhard Riemann (1826 - 1866), an influential German mathematician who made lasting contributions to analysis, number theory, and differential geometry, some of them enabling the later development of general relativity. He first obtained the invariants in his work on plane waves in gas dynamics.

the time level n at the two points marked P and Q, respectively. We find the location x_P and x_Q of these two points along the x-axis by making a straightforward first order, finite difference approximation to the characteristic equations (6.70). The result is

$$x_P = x_j - c_0\Delta t \quad \text{and} \quad x_Q = x_j + c_0\Delta t. \tag{6.74}$$

Moreover, making a similar straightforward finite difference approximation of the compatibility equations (6.71) and (6.72), noting that the two Riemann invariants are conserved along their respective characteristic, we get $[\mathcal{R}^+]_j^{n+1} = [\mathcal{R}^+]_P^n$ and $[\mathcal{R}^-]_j^{n+1} = [\mathcal{R}^-]_Q^n$ or

$$u_j^{n+1} + \frac{1}{c_0}\phi_j^{n+1} = [\mathcal{R}^+]_P^n, \tag{6.75}$$

$$u_j^{n+1} - \frac{1}{c_0}\phi_j^{n+1} = [\mathcal{R}^-]_Q^n, \tag{6.76}$$

where $[\mathcal{R}^+]_P^n$ and $[\mathcal{R}^-]_Q^n$ are the values of $\mathcal{R}^\pm$ at the points P and Q respectively at the previous time level n. Solving with respect to the two unknowns u_j^{n+1} and ϕ_j^{n+1} we get

$$u_j^{n+1} = \frac{1}{2}\left([\mathcal{R}^+]_P^n + [\mathcal{R}^-]_Q^n\right), \tag{6.77}$$

and

$$\phi_j^{n+1} = \frac{1}{2}c_0\left([\mathcal{R}^+]_P^n - [\mathcal{R}^-]_Q^n\right). \tag{6.78}$$

It remains to determine $[\mathcal{R}^+]_P^n$ and $[\mathcal{R}^-]_Q^n$. Since we now the exact position of the two points x_P and x_Q along the x-axis we are in a position to make a first estimate by interpolation using their values at the neighboring grid points. As an example let us assume that $c_0\Delta t \leq \Delta x$, or that the Courant number is less than one. Then the points x_P and x_Q are located between x_{j-1} and x_j and x_j and x_{j+1}, respectively. Hence using Newton's Interpolation Formulae to the lowest order, also referred to as a two point linear interpolation, we get

$$[\mathcal{R}^+]_P^n = (1-C)[\mathcal{R}^+]_j^n + C[\mathcal{R}^+]_{j-1}^n, \quad [\mathcal{R}^-]_Q^n = (1-C)[\mathcal{R}^-]_j^n + C[\mathcal{R}^-]_{j+1}^n, \tag{6.79}$$

where $C = c_0\Delta t/\Delta x$ is the Courant number. Substituting (6.79) into (6.77) and (6.78) we finally get

$$u_j^{n+1} = (1-C)u_j^n + \frac{1}{2}C\left([\mathcal{R}^-]_{j+1}^n + [\mathcal{R}^+]_{j-1}^n\right), \tag{6.80}$$

and

$$\phi_j^{n+1} = (1-C)\phi_j^n - \frac{1}{2}c_0 C\left([\mathcal{R}^-]_{j+1}^n - [\mathcal{R}^+]_{j-1}^n\right). \tag{6.81}$$

We note that if $c_0\Delta t > \Delta x$ the points x_P and x_Q are located to the left of, respectively, to the right of, x_{j-1} and x_{j+1}. Under these circumstances the Courant number is actually larger than one, which for other schemes, e.g., the leapfrog scheme, commonly entails that it would be numerically unstable. However, since we know the exact position of the points x_P and x_Q we continue to interpolate using the adjacent points using Newton's Interpolation Formulae to lowest order. Note that we may also use higher order interpolations. In either case we are not limited by the time step constraint posed by the CFL condition. We do however add some overhead calculation to keep track of the location of the points x_P and x_Q, and performing the interpolation, in particular if we employ higher order terms in Newton's Interpolation Formulae. Thus although we may increase the time step, for instance to six times the value constrained by the CFL condition, some of the gain will be lost to this overhead.

6.3 Staggered grids

We note that the system (6.31) and (6.32) contains four integration constants, namely, two in time and two in space. Focusing on space we note we are allowed to specify two boundary conditions in x, no more no fewer. If we were to solve (6.31) and (6.32) for $x \in < 0, L >$ we then have three options. The first option is to specify one condition at $x = 0$ and one condition at $x = L$. The second is to specify two conditions at $x = 0$, while the third option is to specify two condition at $x = L$. Let us assume that there are impermeable walls at $x = 0$ and $x = L$. Then the natural condition is no flow through them, that is,

$$u|_{x=0} = 0 \quad \text{and} \quad u|_{x=L} = 0 \quad \forall t. \tag{6.82}$$

Letting $x_j = (j-1)\Delta x$ in which $x_1 = 0$ and $x_J = L$ (6.82) numerically translates to

$$u_1^n = 0 \quad \text{and} \quad u_J^n = 0 \quad \forall n. \tag{6.83}$$

Note that in this case no boundary condition is specified for ϕ, that is, ϕ_1^n and ϕ_J^n are unknown. To show that this causes a problem we first rewrite (6.40) and (6.41), that is, the CTCS scheme, to give

$$u_j^{n+1} = u_j^{n-1} - \frac{\Delta t}{\Delta x}\left(\phi_{j+1}^n - \phi_{j-1}^n\right), \tag{6.84}$$

$$\phi_j^{n+1} = \phi_j^{n-1} - c_o^2\frac{\Delta t}{\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right). \tag{6.85}$$

Note that these equations are to be solved for all "wet" points $j = 2(1)J-1$ and $n = 1(1)\infty$[9]. To find the value at an arbitrary time level $n+1$ at the first "wet" point $j = 2$ we get

$$u_2^{n+1} = u_2^{n-1} - \frac{\Delta t}{\Delta x}\left(\phi_3^n - \phi_1^n\right), \tag{6.86}$$

$$\phi_2^{n+1} = \phi_2^{n-1} - c_o^2\frac{\Delta t}{\Delta x}\left(u_3^n - u_1^n\right). \tag{6.87}$$

Here the term ϕ_1^n exhibited by (6.86) poses a problem simply because it is unknown, and we have no boundary conditions that allow us to specify it. We also have a similar problem at the other

[9]As alluded to in Section 5.6 we have to use an Euler scheme to get the solution at n=1 by letting $n = 0$ in (6.84). However, this does not avoid the problem to be described.

boundary $x = L$. In fact evaluating (6.84) and (6.85) for the last "wet" point $j = J - 1$ we get

$$u_{J-1}^{n+1} = u_{J-1}^{n-1} - \frac{\Delta t}{\Delta x}\left(\phi_J^n - \phi_{J-2}^n\right), \tag{6.88}$$

$$\phi_{J-1}^{n+1} = \phi_{J-1}^{n-1} - c_o^2\frac{\Delta t}{\Delta x}\left(u_J^n - u_{J-2}^n\right). \tag{6.89}$$

Again the term ϕ_J^n poses a problem since it is unknown.

If we try to remedy this problem by using option (ii) above, that is, specify ϕ at $x = 0$ as well, the problem aggravates at $x = L$. However tempting we are not allowed to specify more than two conditions in space total. If we continue and specifies ϕ at the boundaries in addition to u, we run into the problem of over-specifying the system, a dangerous path. We do get numbers out of the computer, they may even look reasonable, but they are wrong. This author strongly advocates against exploring such an avenue.

To avoid the problem *Mesinger and Arakawa* (1976) suggested to use what is referred to as *staggered grids*. Instead of calculating the two variables u and ϕ at the same points in space, they suggested to stagger one of them with respect to the other, say one half grid length along the x-axis (Figure 6.3). With this arrangement, or grid structure, we calculate u at $x_{j+\frac{1}{2}}$ points and ϕ at x_j points. As an example let us consider the CTCS scheme. Using the notation depicted in Figure 6.3b to avoid the cumbersome use of $j + \frac{1}{2}$ notations the finite difference approximations to (6.31) and (6.32) become

$$u_j^{n+1} = u_j^{n-1} - \frac{2\Delta t}{\Delta x}\left(\phi_{j+1}^n - \phi_j^n\right), \tag{6.90}$$

$$\phi_j^{n+1} = \phi_j^{n-1} - c_o^2\frac{2\Delta t}{\Delta x}\left(u_j^n - u_{j-1}^n\right). \tag{6.91}$$

Note the appearance of the factor 2 in the second term on the right-hand sides of (6.90) and (6.91). It appears because the distance between two adjacent points in the finite difference approximation of ∂_x in the staggered formulation is Δx rather than $2\Delta x$, while the centered in time scheme still carries $2\Delta t$. We emphasize that (6.90) and (6.91) are correct if and only if the two grids are staggered exactly one half grid length. In principle any staggering will avoid the problem of overspecifying the number of boundary conditions.

It remains to check what impact the staggering has on the numerical stability. To investigate this we first note that care has to be exercised when constructing the discrete Fourier components. Since we have staggered u one half grid length with respect to ϕ, as illustrated Figure 6.3b, we get

$$\phi_j^n = \Phi_n e^{i\alpha j\Delta x} \quad \text{and} \quad u_j^n = U_n e^{i\alpha(j+\frac{1}{2})\Delta x} \tag{6.92}$$

respectively. Substituting these expressions into (6.90) and (6.91) we get

$$U_{n+1} - U_{n-1} = -4i\gamma'\Phi_n, \tag{6.93}$$

$$\Phi_{n+1} - \Phi_{n-1} = -4i\gamma' c_o^2 U_n. \tag{6.94}$$

where

$$\gamma' = \frac{\Delta t}{\Delta x}\sin\left(\frac{\alpha\Delta x}{2}\right) \tag{6.95}$$

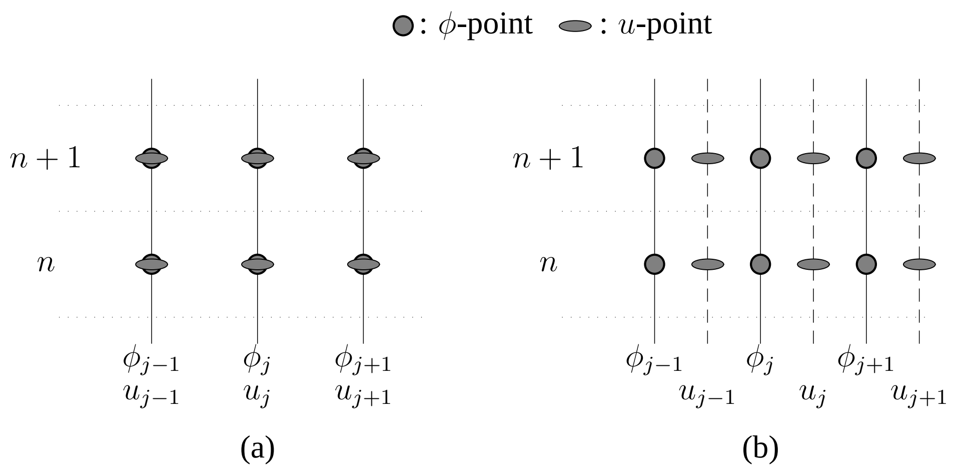


Figure 6.3: Comparison of the structure of (a) an unstaggered and (b) a staggered grid in one spatial dimension. The circels are associated with ϕ-points, while the ellipses are associated with u-points. The illustrated staggering in panel (b) is such that the ϕ-points and u-points are one half grid distance apart, but at the same time level.

Eliminating U_n we get

$$\Phi_{n+2} - 2\Phi_n + \Phi_{n-2} = -16\gamma'^2 c_0^2 \Phi_n. \tag{6.96}$$

Moreover, defining a growth factor by $G \equiv \frac{\Phi_{n+2}}{\Phi_n}$ we get

$$G^2 - 2\lambda' G + 1 = 0, \tag{6.97}$$

where

$$\lambda' = 1 - 8c_0^2 \left(\frac{\Delta t}{\Delta x}\right)^2 \sin^2 \frac{\alpha \Delta x}{2}. \tag{6.98}$$

The growth factor therefore has two complex conjugate solutions given by

$$G_{1,2} = \lambda' \pm i\sqrt{1 - \lambda'^2}. \tag{6.99}$$

Thus as long as $\lambda'^2 \leq 1$ we get $|G_{1,2}| = 1$. The staggered scheme is therefore neutrally stable as expected. However, this is only true as long as $\lambda'^2 \leq 1$, which requires that

$$C \leq \frac{1}{2}, \quad \text{or} \quad \Delta t \leq \frac{\Delta x}{2c_0}. \tag{6.100}$$

We observe that (6.100) is a more stringent CFL condition compared to the CFL condition (6.52) associated with the non-staggered grid. This is no surprise. When we stagger the grids we effectively decreases the distance between two adjacent points to one-half the original grid length. Thus the distance between two adjacent points in the staggered grid, say Δx_{stagg}, is simply $\Delta x_{stagg} = \Delta x/2$. Using Δx_{stagg} instead of Δx the CFL condition becomes the expected

$$\Delta t \leq \frac{\Delta x_{stagg}}{c_0}. \tag{6.101}$$

We finally note that the this stronger constraint on the time step due to the staggering is also valid for other schemes, e.g., the forward-backward scheme.

6.4 Linear, rotating shallow water equations

We are now ready to analyze the effect of the Earth's rotation by retaining the Coriolis terms. We first study their linear, one-dimensional version, that is, (6.22) - (6.24) on page 97.

As we did for the non-rotating case (cf. Section 6.2) we start by analyzing the various wave motions supported by (6.22) - (6.24)[10]. Thus we assume a wave solution,

$$\mathbf{X} = \mathbf{X}_0 e^{i\alpha(x-ct)}, \tag{6.102}$$

where α is the wavenumber in the x direction and $\mathbf{X}$ denotes a vector consisting of the three dependent variables, that is,

$$\mathbf{X} = \begin{bmatrix} u \\ v \\ \phi \end{bmatrix} \quad \text{and} \quad \mathbf{X}_0 = \begin{bmatrix} u_0 \\ v_0 \\ \phi_0 \end{bmatrix}, \tag{6.103}$$

where $\mathbf{X}_0$ is the amplitude.

Inserting (9.25) into (6.22) - (6.24) we get

$$\begin{aligned} -i\alpha c u_0 - f v_0 + i\alpha\phi_0 &= 0, & (6.104)\\ -i\alpha c v_0 + f u_0 &= 0, & (6.105)\\ c\phi_0 - c_o^2 u_0 &= 0, & (6.106) \end{aligned}$$

To find the dispersion relation we first multiply (6.104) by $-i\alpha c$ and (6.105) by f and add the results together. By substituting the result into (6.106) we get

$$c\left\{c^2 - c_0^2\left[1 + \left(\frac{1}{\alpha L_R}\right)^2\right]\right\} = 0, \tag{6.107}$$

where

$$L_R = c_0/f \tag{6.108}$$

is Rossby's deformation radius. Thus one solution is $c_1 = 0$, a stationary wave. The two remaining solutions are

$$c_{2,3} = \pm c_0\sqrt{1 + \left(\frac{1}{\alpha L_R}\right)^2}. \tag{6.109}$$

[10]Recall that any solution of (6.22) - (6.24), under the assumption that the solution is a good function, may be represented by an infinite number of waves of different wavelengths and amplitudes in accord with Section 2.11 on page 27.

These are gravity waves modified by the Earth's rotation and are commonly referred to as inertia-gravity waves. The gravity waves are associated with wave speeds $c = \pm c_0$, and are thus associated with the first term in the radical. The inertia part is associated with oscillating frequencies $\omega = f$ or phase speed $c = f/\alpha$, that is, frequencies proportional to the inertia frequency or inertial oscillation, and is contained in the second term in the radical.

We may also apply the same procedure if the basic state is in geostrophic balance, that is, using the linearized equations (6.19) - (6.21) on page 97. Substituting (6.103) into (6.19) - (6.21) we get

$$\begin{aligned} i\alpha\left(\bar{u}-c\right)u_0 - fv_0 + i\alpha\phi_0 &= 0, & (6.110)\\ fu_0 + i\alpha\left(\bar{u}-c\right)v_0 &= 0, & (6.111)\\ \bar{\phi}u_0 + \left(\bar{u}-c\right)\phi_0 &= 0. & (6.112) \end{aligned}$$

Following the same procedure the dispersion relation becomes

$$\left(\bar{u}-c\right)\left[-\alpha^2\left(\bar{u}-c\right)^2 + f^2 + \alpha^2\bar{\phi}\right] = 0. \tag{6.113}$$

Thus again there are three solutions for the phase speed c, namely

$$\begin{aligned} c_1 &= \bar{u}, & (6.114)\\ c_{2,3} &= \bar{u} \pm c_0\sqrt{1 + \left(\frac{1}{\alpha L_R}\right)^2}, & (6.115) \end{aligned}$$

where

$$c_0 = \sqrt{\bar{\phi}} = \sqrt{gH}. \tag{6.116}$$

The first solution is an infinitely long wave in which the motion is in geostrophic balance commonly referred to as the Rossby mode. We easily derive this interpretation by substituting $c_1 = \bar{u}$ from (6.114) into (6.110). The latter equation then becomes

$$-fv + i\alpha\phi = 0 \quad \text{or} \quad v = \frac{1}{f}i\alpha\phi. \tag{6.117}$$

Using the Fourier solution backwards we thus recover the geostrophic balance (1.40), that is,

$$v = \frac{1}{f}\partial_x\phi. \tag{6.118}$$

The two other solutions are the inertia-gravity wave modes referred to in the previous paragraph, where the inertia part is associated with oscillating frequencies $\omega = f$ and the two gravity waves with wave speeds $c = \pm c_0$. The only difference is that the waves now ride on the basic current $\bar{u}$. We note that commonly the inertia-gravity mode has a much higher wave speed than the Rossby mode, that is $|c_0| \gg |\bar{u}|$. Thus letting $\bar{u} \approx 0$ in (6.19) - (6.21) brings us back to solving (6.22) - (6.24), that is, the problem with a basic state at rest.

Finite difference forms

To solve (6.22) - (6.24) numerically using a finite difference method we replace the derivatives by finite difference approximations. Below follows details on the leapfrog (CTCS), the forward-backward and the semi-Lagrange scheme.

The CTCS scheme

Since the problem is hyperbolic we apply the centered in time, centered in space (CTCS) leapfrog scheme on a non-staggered grid. We then get

$$\begin{aligned} u_j^{n+1} - u_j^{n-1} &= 2f\Delta t v_j^n - \frac{\Delta t}{\Delta x}\left(\phi_{j+1}^n - \phi_{j-1}^n\right), && (6.119)\\ v_j^{n+1} - v_j^{n-1} &= -2f\Delta t u_j^n, && (6.120)\\ \phi_j^{n+1} - \phi_j^{n-1} &= -c_0^2\frac{\Delta t}{\Delta x}\left(u_{j+1}^n - u_{j-1}^n\right). && (6.121)\end{aligned}$$

We note that these equations reduces to (6.40) and (6.41) when $f = 0$. It is therefore of interest to investigate the impact of rotation on the numerical stability. To this end we replace the variables by their discrete Fourier components

$$u_j^n = U_n e^{i\alpha j\Delta x}, \quad v_j^n = V_n e^{i\alpha j\Delta x}, \quad \phi_j^n = \Phi_n e^{i\alpha j\Delta x}. \qquad (6.122)$$

Substituting (6.122) into (6.119) - (6.121) we get

$$\begin{aligned} U_{n+1} - U_{n-1} &= 2f\Delta t V_n - 2i\gamma\Phi_n, && (6.123)\\ V_{n+1} - V_{n-1} &= -2f\Delta t U_n, && (6.124)\\ \Phi_{n+1} - \Phi_{n-1} &= -2i\gamma c_0^2 U_n, && (6.125)\end{aligned}$$

where as before

$$\gamma = \frac{\Delta t}{\Delta x}\sin\alpha\Delta x. \qquad (6.126)$$

Eliminating V_n and Φ_n we get[11]

$$U_{n+2} - 2\lambda U_n + U_{n-2} = 0, \qquad (6.127)$$

where

$$\lambda = 1 - 2\gamma^2 c_0^2 - 2f^2\Delta t^2. \qquad (6.128)$$

Defining the growth factor as $G \equiv U_{n+2}/U_n$ we get two complex conjugate solution $G_{1,2} = \lambda \pm i\sqrt{1-\lambda^2}$ provided the radical is a positive definite quantity. As expected $|G_{1,2}| = 1$ and hence the CTCS scheme is neutrally stable. The impact of throwing in the Coriolis effect is

[11]This most efficiently done by first raising n by one in (6.123) giving an expression involving U_{n+2}, U_n, V_{n+1} and Φ_{n+1}. Next decreasing n by one gives an expression containing U_n, U_{n-2}, V_{n-1} and Φ_{n-1}. Subtracting and using (6.124) and (6.125) then results in (6.127).

inherent in the expression for λ. To ensure that the growth factor is complex we must require $\lambda^2 \leq 1$. This is satisfied if and only if

$$\gamma^2 c_0^2 + f^2 \Delta t^2 \leq 1 \quad \text{or} \quad c_0^2 \left(\frac{\Delta t}{\Delta x}\right)^2 \sin^2 \alpha \Delta x + f^2 \Delta t^2 \leq 1. \tag{6.129}$$

Thus the CFL criterion for stability becomes,

$$\Delta t \leq \frac{\Delta x}{c_0} \left[1 + \left(\frac{\Delta x}{L_R}\right)^2\right]^{-\frac{1}{2}}, \tag{6.130}$$

where L_R is Rossby's deformation radius as defined in (6.108). If we choose the mesh size Δx such that it resolves Rossby's deformation radius, say $\Delta x = \frac{1}{10} L_R$, in which case $\Delta x / L_R << 1$, then the first term in the radical dominates. Under these circumstances it follows that the practical condition is

$$\Delta t < \frac{\Delta x}{c_0} \quad \text{or} \quad C < 1. \tag{6.131}$$

In most models of the atmosphere today's computers are powerful enough so that the mesh size satisfies the condition $\Delta x / L_R << 1$. Thus the condition (6.131) is sufficient. This is not necessarily the case for models of the ocean. In particular this is true for ocean models employed in global climate modeling. Thus for many applications $\Delta x / L_R$ is of $\mathcal{O}(1)$, and hence we must take into account also the second term in the radical. We underscore that for such problems the CFL condition becomes more stringent.

In an atmospheric model the largest equivalent depth is approximately 10 km giving a speed of the inertia-gravity waves of the order of 300 m/s. This is considerably more than the wind speed and sets strong limitations to how long time steps we can take. In an ocean model the situation is the same. Although the oceanic equilibrium depth is one order of magnitude less, about 1 km, the wave speed is still about 100 m/s considerably larger than a typical ocean current speed of 0.1 m/s. Thus both in the ocean and the atmosphere the CFL condition limits the time steps to minutes and sometimes seconds[12]. We emphasize that inertia-gravity waves are important properties in the ocean and contain for instance the barotropic tidal motion as well as storm surges. Thus, in contrast to atmospheric models, oceanic models are restricted to such limitations on the time step since we have to simulate these important oceanic features explicitly. In the atmosphere the inertia-gravity waves carries little energy compared to for instance Rossby waves. Hence these motions are not part of the signal and is commonly neglected by treating the terms semi-implicitly (cf. Section 6.6 on page 120).

[12]For and ocean of equilibrium depth $H = 4 \cdot 10^3$ m, $g = 10$ ms^{-2} and non-eddy resolving grid size $\Delta x = 20$ km follows from (6.131) that $\Delta t < 141$ s or slightly more than two minutes.

The forward-backward scheme

As long as the system is still linear we may also employ the forward-backward scheme (e.g., *Sielecki*, 1968; *Martinsen et al.*, 1979). Thus we construct the scheme

$$\begin{aligned} u_j^{n+1} &= u_j^n + f\Delta t v_j^n - \frac{\Delta t}{2\Delta x}(\phi_{j+1}^n - \phi_{j-1}^n), && (6.132)\\ v_j^{n+1} &= v_j^n - f\Delta t u_j^{n+1}, && (6.133)\\ \phi_j^{n+1} &= \phi_j^n - c_0^2\frac{\Delta t}{2\Delta x}(u_{j+1}^{n+1} - u_{j-1}^{n+1}). && (6.134)\end{aligned}$$

We note again that the scheme reduces to (6.59) and (6.60) when $f = 0$. We emphasize that the order in which (6.132) - (6.134) are solved is random. The idea is that as soon as a variable is updated it is used in the next equation. Thus we may equally well make use of the schemes

$$\begin{aligned} \phi_j^{n+1} &= \phi_j^n - c_0^2\frac{\Delta t}{2\Delta x}(u_{j+1}^{n} - u_{j-1}^{n}), && (6.135)\\ u_j^{n+1} &= u_j^n + f\Delta t v_j^n - \frac{\Delta t}{2\Delta x}(\phi_{j+1}^{n+1} - \phi_{j-1}^{n+1}), && (6.136)\\ v_j^{n+1} &= v_j^n - f\Delta t u_j^{n+1}, && (6.137)\end{aligned}$$

or

$$\begin{aligned} v_j^{n+1} &= v_j^n - f\Delta t u_j^{n}, && (6.138)\\ \phi_j^{n+1} &= \phi_j^n - c_0^2\frac{\Delta t}{2\Delta x}(u_{j+1}^{n} - u_{j-1}^{n}), && (6.139)\\ u_j^{n+1} &= u_j^n + f\Delta t v_j^{n+1} - \frac{\Delta t}{2\Delta x}(\phi_{j+1}^{n+1} - \phi_{j-1}^{n+1}). && (6.140)\end{aligned}$$

Again it is of interest to investigate the impact of rotation on the stability. As so many times before we employ von Neumann's method for this purpose. Thus we first replace the variables in the scheme by their discrete Fourier components as defined in (6.122), then we eliminate two of the amplitudes, say V_n and Φ_n. Finally we define the growth factor by $G = U_{n+2}/U_n$ and get

$$G^2 - 2\lambda' G + 1 = 0, \tag{6.141}$$

where

$$\lambda' = 1 - \gamma^2 c_0^2 - f^2\Delta t^2. \tag{6.142}$$

Thus the scheme is stable provided the condition

$$\Delta t \leq \frac{2\Delta x}{c_0}\left[1 + \left(\frac{\Delta x}{L_R}\right)^2\right]^{-\frac{1}{2}} \tag{6.143}$$

is satisfied. We note that compared to the problem without rotation the condition is less stringent by a factor of 2.

The semi-Lagrange scheme

Finally it is of interest to investigate the impact of rotation on the semi-Lagrange scheme. To find the characteristics we do exactly as we did for the non-rotating case. Thus we multiply (6.24) by an unknown function λ and add it to (6.22). We then get

$$\frac{D^*_{1,2}\mathcal{R}^{\pm}}{dt} = fv \quad \text{along} \quad \frac{D^*_{1,2}x}{dt} = \pm c_0, \tag{6.144}$$

where $\mathcal{R}^{\pm}$ are the Riemann invariants displayed in (6.73) on page 104. These are exactly the same two characteristic as in (6.71) and (6.72), that is, (6.70), while the compatibility equations deviates from (6.71) and (6.72) by the fv term on the right-hand side. When we introduced rotation a third unknown v was introduced and a third equation (6.23) was added to the set of governing equations. Hence a third compatibility equation must be added too. It actually follows directly from (6.23), that is,

$$\frac{D^*_3 v}{dt} = -fu \quad \text{along} \quad \frac{D^*_3 x}{dt} = 0, \tag{6.145}$$

where the third operator is defined by

$$\frac{D^*_3}{dt} = \partial_t + \frac{D^*_3 x}{dt}\partial_x. \tag{6.146}$$

We note that the third characteristic equation is $\frac{D^*_3 x}{dt} = 0$ and hence that it has an infinite slope.

To construct a finite difference scheme we follow the procedure leading to (6.74) - (6.76). Hence we get

$$\begin{aligned} u_j^{n+1} + \frac{\phi_j^{n+1}}{c_0} &= u_P^n + \frac{\phi_P^n}{c_0} + \frac{1}{2}f\Delta t(v_j^{n+1} + v_P^n), & (6.147)\\ u_j^{n+1} - \frac{\phi_j^{n+1}}{c_0} &= u_Q^n - \frac{\phi_Q^n}{c_0} + \frac{1}{2}f\Delta t(v_j^{n+1} + v_Q^n), & (6.148)\\ v_j^{n+1} &= v_j^n - \frac{1}{2}f\Delta t(u_j^{n+1} + u_j^n), & (6.149) \end{aligned}$$

that is, three equation containing the three unknowns u_j^{n+1}, v_j^{n+1} and ϕ_j^{n+1}. Since the integration is along the characteristics, we emphasize the use of $\frac{1}{2}f\Delta t(v_j^{n+1} + v_P)$ in (6.147) and the similar terms in (6.148) and (6.149). Solving with respect to each of the variables at the point x_j, t^{n+1} we get

$$\begin{aligned} u_j^{n+1} &= \left(1 + \frac{1}{2}f^2\Delta t^2\right)^{-1} F_u, & (6.150)\\ v_j^{n+1} &= v_j^n - \frac{1}{2}f\Delta t\left[\left(1 + \frac{1}{2}f^2\Delta t^2\right)^{-1} F_u + u_j^n\right], & (6.151)\\ \phi_j^{n+1} &= F_\phi & (6.152) \end{aligned}$$

where

$$\begin{aligned} F_u &= \frac{1}{2}\left(u_P^n + u_Q^n\right) + \frac{1}{2c_0}\left(\phi_P^n - \phi_Q^n\right) + \frac{1}{4} f\Delta t\left(2v_j^n - f\Delta t u_j^n + v_P^n + v_Q^n\right), \quad (6.153)\\ F_\phi &= \frac{1}{2}c_0\left(u_P^n - u_Q^n\right) + \frac{1}{2}\left(\phi_P^n + \phi_Q^n\right) + \frac{1}{2}c_0 f\Delta t\left(v_P^n - v_Q^n\right), \quad (6.154)\end{aligned}$$

The needed values of the variables at the points P and Q is found exactly as in the former case, that is, by interpolation. We emphasize that in addition to $u_{P,Q}^n$ and $\phi_{P,Q}^n$ we also need to determine $v_{P,Q}^n$ by interpolation. Finally we note that if we let $f = 0$ the above equations reduce to (6.77) and (6.78).

6.5 Non-linear, rotating shallow water equations

Let us further expand the shallow water equations to a fully non-linear and rotating case, that is, (6.3) - (6.4). Again assuming a one-dimensional system, that is, neglecting all terms differentiated with respect to y, we get

$$\begin{aligned} \partial_t u + u\partial_x u - fv &= -\partial_x \phi, \quad (6.155)\\ \partial_t v + u\partial_x v + fu &= 0, \quad (6.156)\\ \partial_t \phi + u\partial_x \phi &= -\phi\partial_x u. \quad (6.157)\end{aligned}$$

We note that we retain three equations with three unknowns u, v and ϕ as dependent variables. The independent variables are time t and the horizontal direction x.

Rewriting (6.155) - (6.157) in terms of the volume flux $U = hu$, $V = hv$, and noting that $\phi = gh$, we get

$$\begin{aligned} \partial_t U + \partial_x\left(\frac{U^2}{h}\right) - fV &= -\frac{1}{2}g\partial_x h^2, \quad (6.158)\\ \partial_t V + \partial_x\left(\frac{UV}{h}\right) + fU &= 0, \quad (6.159)\\ \partial_t h + \partial_x U &= 0. \quad (6.160)\end{aligned}$$

The advantage of using (6.158) - (6.160) instead of (6.155) - (6.157) is that the continuity equation (6.160) becomes linear, and that the non-linear terms in (6.158) and (6.159) are all written in flux form. The latter gives better conservation properties of the associated numerical scheme.

Finite difference forms

The CTCS scheme

The system is still hyperbolic so it is natural to employ a CTCS (leapfrog) scheme. Hence if we employ an unstaggered grid and construct a CTCS scheme based on (6.158) - (6.160) we get

$$U_j^{n+1} = U_j^{n-1} + 2f\Delta t V_j^n + A_j^n + P_j^n \tag{6.161}$$

$$V_j^{n+1} = V_j^{n-1} - 2f\Delta t U_j^n + B_j^n \tag{6.162}$$

$$h_j^{n+1} = h_j^{n-1} - \frac{\Delta t}{\Delta x}\left(U_{j+1}^n - U_{j-1}^n\right), \tag{6.163}$$

where

$$A_j^n = -\frac{\Delta t}{\Delta x}\left(\left[\frac{U^2}{h}\right]_{j+1}^n - \left[\frac{U^2}{h}\right]_{j-1}^n\right), \tag{6.164}$$

$$B_j^n = -\frac{\Delta t}{\Delta x}\left(\left[\frac{UV}{h}\right]_{j+1}^n - \left[\frac{UV}{h}\right]_{j-1}^n\right), \tag{6.165}$$

$$P_j^n = -\frac{g\Delta t}{2\Delta x}\left(\left[h^2\right]_{j+1}^n - \left[h^2\right]_{j-1}^n\right). \tag{6.166}$$

To avoid the use of more boundary conditions in space than allowed, we have to stagger the grids for h and U, V as explained in Section 6.3. Using the cell structure of Figure 6.3 and using j as the cell counter we then get

$$U_j^{n+1} = U_j^{n-1} + 2f\Delta t V_j^n + A_j^n + P_j^n, \tag{6.167}$$

$$V_j^{n+1} = V_j^{n-1} - 2f\Delta t U_j^n + B_j^n, \tag{6.168}$$

$$h_j^{n+1} = h_j^{n-1} - \frac{2\Delta t}{\Delta x}\left(U_j^n - U_{j-1}^n\right), \tag{6.169}$$

where A_j^n and B_j^n are as before, while P_j^n, because of the staggering, becomes

$$P_j^n = -\frac{g\Delta t}{\Delta x}\left(\left[h^2\right]_{j+1}^n - \left[h^2\right]_j^n\right). \tag{6.170}$$

Note that when staggered the evaluation of A_j^n and B_j^n is associated with a u and v-points. Hence we must interpolate to find h at these points. Thus regarding the first term on the right-hand side of (6.164) we get

$$\left[\frac{U^2}{h}\right]_{j+1}^n = \frac{(U_j^n)^2}{\frac{1}{2}(h_{j+2}^n + h_{j+1}^n)}. \tag{6.171}$$

The semi-Lagrangian scheme

We emphasize that in the non-linear case we cannot employ the forward-backward scheme. However, even though we invoke the non-linear terms we may still construct a semi-Lagrangian scheme. We do this, as we have done many times already, by multiplying (6.157) by a yet unknown function λ and add it to the the resulting equation (6.155). Note that it is convenient to first replace ϕ by the propagation speed $c = \sqrt{\phi}$ in (6.155) - (6.157) before the manipulation. In any case we arrive at

$$\left[\partial_t + (u + \frac{1}{2}\lambda c)\partial_x\right] u + \lambda \left[\partial_t + (u + \frac{2c}{\lambda})\partial_x\right] c = fv. \tag{6.172}$$

As before we require that the operators $\partial_t + (u + \frac{1}{2}\lambda c)\partial_x$ and $\partial_t + (u + \frac{2c}{\lambda})\partial_x$ are the same, which is achieved by letting $u + \frac{1}{2}\lambda c = u + \frac{2c}{\lambda}$, which gives $\lambda_{1,2} = \pm 2$. Hence the characteristic equations are

$$\frac{D^*_{1,2}}{dt} = u \pm c, \tag{6.173}$$

while the two Riemann invariants become

$$\mathcal{R}^{\pm} = u \pm 2c. \tag{6.174}$$

Finally, the compatibility equations are

$$\frac{D^*_{1,2}\mathcal{R}^{\pm}}{dt} = fv \quad \text{along} \quad \frac{D^*_{1,2}x}{dt} = u \pm c. \tag{6.175}$$

We also have a third equation, namely (6.157), which must be trnsformed into its compatibility equation. We find this directly from (6.156) by observing that it may be written

$$\frac{D^*_3 v}{dt} = -fu \quad \text{along} \quad \frac{D^*_3 x}{dt} = u. \tag{6.176}$$

Thus we obtain three characteristics with slopes $u + c$, $u - c$ and u as depicted in Figure 6.4.

Applying a simple forward in time finite difference approximations to (6.175) and (6.176) we get

$$\begin{aligned} u_j^{n+1} + 2c_j^{n+1} &= u_P^n + 2c_P^n + \frac{1}{2}f\Delta t\left(v_j^{n+1} + v_P^n\right), && (6.177)\\ u_j^{n+1} - 2c_j^{n+1} &= u_Q^n - 2c_Q^n + \frac{1}{2}f\Delta t\left(v_j^{n+1} + v_Q^n\right), && (6.178)\\ v_j^{n+1} &= v_R^{n+1} - \frac{1}{2}f\Delta t\left(u_j^{n+1} + u_R^n\right). && (6.179)\\ & && (6.180) \end{aligned}$$

where the subscripts P, Q, and R refer to the evaluation of the variable in questions at the points P, Q and R, at time level n, respectively. The locations of these points in space are denoted x_P, x_Q and x_R, respectively, as displayed in Figure 6.4. To find their position we just integrate the characteristic equations using a simple forward in time finite difference approximation. Hence

$$x_P = x_j - (u_j^n + c_j^n)\Delta t,\ x_Q = x_j - (u_j^n - c_j^n)\Delta t,\ \text{and}\ x_R = x_j - u_j^n\Delta t. \tag{6.181}$$

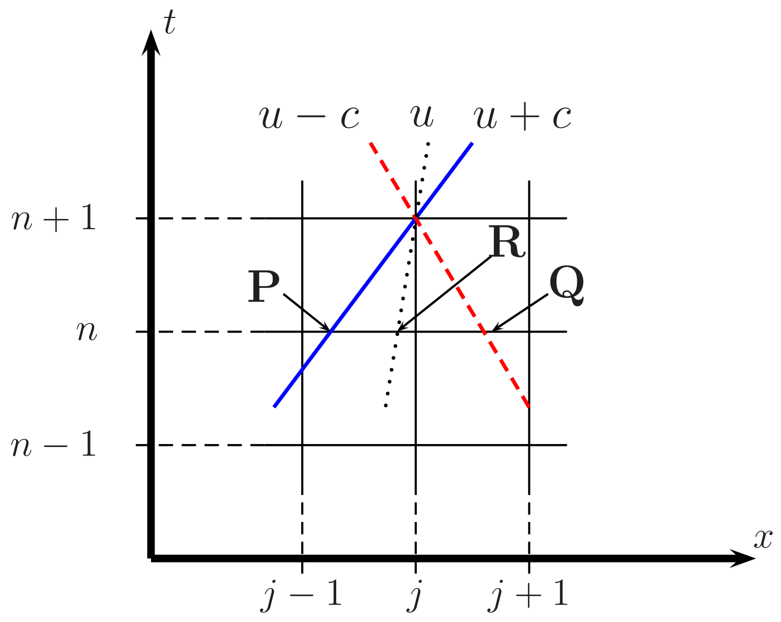


Figure 6.4: Sketch of the semi-Lagrangian technique for a non-linear and rotating case. The distance between the grid points are Δt in the vertical and Δx in the horizontal direction. There are three characteristics through the point $j, n+1$. The blue solid line is the positive characteristic with slope $u+c$, while the dashed red line is the negative characteristic with slope $u-c$. These are derived from (6.173). The last characteristic with slope u is the dotted black line derived from (6.176). Provided $u \geq 0$ the point labeled P is a distance $(u+c)\Delta t$ to the left of x_j, while the point Q is a distance $(u-c)\Delta t$ to the right of x_j. Hence the assymetry. Finally the point labeled R is located a distance $u\Delta t$ to the left of x_j. As long as $(u+c)\Delta t \leq \Delta x$ and $u > 0$ then P, R is located between x_{j-1} and x_j and Q between x_{j+1} and x_j. If however $(u+c)\Delta t > \Delta x$ then the points Q, P are located to the left and right of respectively x_{j-1} and x_{j+1}.

To find evaluate the nine quantities $u^n_{P,Q,R}$, $v^n_{P,Q,R}$ and $c^n_{P,Q,R}$, we resort to interpolation. For instance by find u^n_P using the lowest order of Newton's interpolation formulas (two point, linear interpolation) we find that u^n_P may be approximated to

$$u^n_P = \left(1 - C^n_{Pj}\right) u^n_j + C^n_{Pj} u^n_{j-1}, \tag{6.182}$$

where $C^n_{Pj} = \frac{\Delta t}{\Delta x}(u^n_j + c^n_j)$. The remaining eight values are found by a similar interpolation. We note that in order to use (6.182) we require that $x_{j-1} \leq x_P \leq x_j$ or $C^n_{Pj} \leq 1$. If this is not the case we have to find out where the nearest grid point is and then apply the lowest order interpolation. A similar condition applies to the remaining eight as well. Another option is to resort to higher order interpolation. Either way we end up with three equations to solve for the three unknowns u^{n+1}_j, v^{n+1}_j and c^{n+1}_j.

We note that in the non-linear case the characteristics are no longer straight lines in the t, x space. Thus we may view the locations of x_P, x_Q and x_R we get from (6.181) as a first guess and denoting them $x_P^{(0)}$, $x_Q^{(0)}$ and $x_R^{(0)}$. Similarly we let $u_{P,Q,R}^{(0)}$, $v_{P,Q,R}^{(0)}$ and $c_{P,Q,R}^{(0)}$ denote the first guess we get by interpolation and finally $u_j^{(0)}$, $v_j^{(0)}$ and $c_j^{(0)}$ as the first guess of the variables at the point x_j, t^{n+1}. We are then in a position to compute improved approximations to the positions of the points P, Q and R,

$$\begin{aligned} x_P^{(1)} &= x_j - \frac{1}{2}(u_j^{(0)} + u_P^{(0)} + c_j^{(0)} + c_P^{(0)})\Delta t, && (6.183) \\ x_Q^{(1)} &= x_j - \frac{1}{2}(u_j^{(0)} + u_Q^{(0)} - c_j^{(0)} - c_Q^{(0)})\Delta t, && (6.184) \\ x_R^{(1)} &= x_j - \frac{1}{2}(u_j^{(0)} + u_R^{(0)})\Delta t && (6.185) \end{aligned}$$

We then find updated values of $u_{P,Q,R}^{(1)}$, $v_{P,Q,R}^{(1)}$ and $c_{P,Q,R}^{(1)}$ by interpolation using the improved positions. In turn this enables us to update the values of the variables at the point x_j, t^{n+1} using the formulas,

$$\begin{aligned} u_j^{(\nu)} + 2c_j^{(\nu)} &= u_P^{(\nu-1)} + 2c_P^{(\nu-1)} + \frac{1}{2}f\Delta t\left(v_j^{(\nu)} + v_P^{(\nu-1)}\right), && (6.186) \\ u_j^{(\nu)} - 2c_j^{(\nu)} &= u_Q^{(\nu-1)} - 2c_Q^{(\nu-1)} + \frac{1}{2}f\Delta t\left(v_j^{(\nu)} + v_Q^{(\nu-1)}\right), && (6.187) \\ v_j^{(\nu)} &= v_R^{(\nu)} - \frac{1}{2}f\Delta t\left(u_j^{(\nu)} + u_R^{(\nu-1)}\right), && (6.188) \\ & && (6.189) \end{aligned}$$

where $\nu = 1$. This is the start of an iteration procedure. Once we have calculated $u_j^{(\nu)}$, $v_j^{(\nu)}$ and $c_j^{(\nu)}$ for any $\nu = 1, 2, \ldots$ using (6.186) - (6.188), we may update the positions using the formulas

$$\begin{aligned} x_P^{(\nu+1)} &= x_j - \frac{1}{2}(u_j^{(\nu)} + u_P^{(\nu)} + c_j^{(\nu)} + c_P^{(\nu)})\Delta t, && (6.190) \\ x_Q^{(\nu+1)} &= x_j - \frac{1}{2}(u_j^{(\nu)} + u_Q^{(\nu)} - c_j^{(\nu)} - c_Q^{(\nu)})\Delta t, && (6.191) \\ x_R^{(\nu+1)} &= x_j - \frac{1}{2}(u_j^{(\nu)} + u_R^{(\nu)})\Delta t && (6.192) \end{aligned}$$

to find the new locations followed by an interpolation to find $u_{P,Q,R}^{(\nu+1)}$, $v_{P,Q,R}^{(\nu+1)}$ and $c_{P,Q,R}^{(\nu+1)}$. We may then proceed to find $u_j^{(\nu+1))}$, $v_j^{(\nu+1))}$ and $c_j^{(\nu+1))}$ from (6.186) - (6.188), which allows us to further update the locations and so on. We may repeat this iteration as long as we wish, or until we have reached a satisfactory accuracy.

Numerical stability

The semi-Lagrangian scheme is always stable, whether we study the linear or non-linear rotational shallow water equations case. Regarding the CTCS scheme we know for certain that the scheme is stable if and only if the CFL condition is satisfied. The question therefore arises whether the non-linear CTCS scheme is stable, and if so under what condition. Since the equations are non-linear the analysis is not as straightforward as for a linear system. In fact throwing in the non-linear terms adds to the complexity of the possibilities for an unstable solution. The reason is, as alluded to earlier on in Section 3.3 on page 35 and Section 4.9 on page 52, that in contrast to linear dynamics non-linear dynamics allow energy to be exchanged between waves. Thus the non-linear terms are able to redistribute energy among the different wavelengths present in the problem, something which is impossible in a linear system. In fact as time evolves the non-linear terms acts to cascade the energy progressively towards smaller and smaller wavelengths in accord with the Taylor rhyme quoted on page 35.

When we solve non-linear problems numerically using finite difference approximations to replace the differential operators inherent in our equations, we simply do not resolve waves with wavelengths shorter than $2\Delta x$. In the real world, however, the energy cascade continues beyond this wavelength and progressively towards shorter and shorter waves. In our numerical solution the grid resolution inhibits this cascading across the resolution limit, that is, across the $2\Delta x$ wavelength. The energy thus accumulates in the wavelength bands closest to our grid resolution, that is, in the $2\Delta x < \lambda < 4\Delta x$ wavelength band. Eventually this causes any scheme that works well for a linear system to blow up. This is called non-linear instability and is treated in more detail in Section 10.3 on page 167.

The processes responsible for continue the energy cascade across and beyond the grid resolution wavelength acts on scales that are shorter than our $2\Delta x$ grid resolution. Such processes are commonly referred to as *sub-grid-scale (SGS)* processes. To numerically allow for a continuation of the energy cascade across the the $2\Delta x$ limit we have to parameterize (or mimic) the SGS processes. Commonly we parameterize the SGS processes in terms of some kind of diffusion. As we were showing in Section 4.3, diffusion acts to smooth out noise, that is, diffusion is selective and helps to smooth out the shortest waves. We may therefore add diffusive terms to our governing, non-linear equations to avoid the continues accumulation of energy in the short wavelength band. In our relatively simple one-dimensional cases this entails adding terms of the form $\kappa\partial_x^2 u$, where κ is a diffusion coefficient. As alluded to in Chapter 4 the diffusion inherent in such a term is scale selective, that is, damps out the shortest way most efficiently since all the Fourier components will be damped by a factor $e^{-\kappa\alpha^2 t}$. The results are however sensitive to the choice of diffusion coefficient. Too high and the energy is damped too fast. This will give a result where too much energy is absorbed by diffusion and the solution will too smooth. Too low and the energy contained in low wave number band increases with time. The results is that the non-linear instability sooner or later kicks in and the solution becomes unstable.

6.6 Semi-implicit and time-splitting methods

From the analysis above we notice that by introducing a pressure force (in addition to advection) the CFL criterion becomes much more stringent (shorter time-step). It is therefore tempting to treat terms responsible for this behavior implicitly while we treat other terms explicitly. Such a method is commonly referred to as a *semi-implicit method*.

For clarity we start with a one dimensional shallow water problem, that is,

$$\partial_t u = A_u - \partial_x \phi, \tag{6.193}$$

$$\partial_t v = A_v, \tag{6.194}$$

$$\partial_t \phi = A_\phi - \Phi \partial_x u, \tag{6.195}$$

where A_u, A_v and A_Φ include the non-linear as well as the Coriolis terms. We learned previously that the terms responsible for this behavior was the pressure terms. From Section 4.8 we learned that treating any term implicitly avoid this restriction on the time step. It is therefore tempting to treat the pressure terms, that is, the $\partial_x \phi$ and $\Phi \partial_x u$ terms implicitly while integrating the remaining terms explicitly. The finite difference approximation form of the equations above on an unstaggered grid then becomes,

$$\frac{u^{n+1} - u^{n-1}}{2\Delta t} = [A_u]^n - [\partial_x \phi]^{n+1}, \tag{6.196}$$

$$\frac{v^{n+1} - v^{n-1}}{2\Delta t} = [A_v]^n, \tag{6.197}$$

$$\frac{\phi^{n+1} - \phi^{n-1}}{2\Delta t} = [A_\phi]^n - \Phi \left[\partial_x u\right]^{n+1}, \tag{6.198}$$

To proceed, the first two equations are solved with respect to u_j^{n+1} and v_j^{n+1} respectively giving,

$$u^{n+1} = u^{n-1} + 2\Delta t [A_u]^n - 2\Delta t \left[\partial_x \phi\right]^{n+1} \tag{6.199}$$

$$v^{n+1} = v^{n-1} + 2\Delta t [A_v]^n \tag{6.200}$$

derivation with respect to x and insertion into the equation for ϕ gives a Helmholtz equation,

$$\Phi \Delta t^2 [\partial_x^2 \phi]^{n+1} - \phi^{n+1} = B \tag{6.201}$$

where B contains known quantities at time levels $n, n-1, \ldots$

With proper boundary conditions (ϕ or its normal derivative at lateral boundaries), these equations may easily be solved by standard numerical methods called elliptic solvers. One such elliptic solver, is the direct elliptic solver treated in Section 4.11 (Gauss elimination), but there are also a host of iterative (non-direct) solvers to choose. The most commonly in use is called the successive over-relaxation (SOR) method. Having obtained ϕ^{n+1}, we easily find u^{n+1} and v^{n+1}. This method is widely used in atmospheric models as we do not have to take the gravity mode speed $c_0 = \sqrt{\Phi} = \sqrt{gH}$ into account when estimating an upper bound for the time step. Thus we avoid the restrictive CFL condition and thus makes it is possible to use longer time-steps.

We emphasize that we cannot employ this method in the ocean. Treating the fast barotropic waves implicitly would then ruin the inertia-gravity waves which carries information about such important signals as tides and storm surges. To speed up the computations of ocean models it is common to resort to so called time-splitting. The idea is to treat the barotropic and baroclinic parts of the motion separately. The procedure is to first integrate the slow baroclinic modes forward from time t using a time step Δt_{bc}. Then the barotropic mode, which is influenced by the baroclinic modes, is integrated forward for the time span $t + \Delta t_{bc}$ using a much smaller time step, say Δt_{bt}. Commonly $\Delta t_{bc} = N \Delta t_{bt}$ where N is of order 50.

Chapter 7

OPEN BOUNDARY CONDITIONS AND NESTING TECHNIQUES

As is well known, computers, however large, can only hold a finite amount of numbers in their so called *random access memory* (RAM). Thus even the biggest computers has a limited capacity, which put a constraint on how large a Numerical Weather Prediction (NWP) and/or a Numerical Ocean Weather Prediction (NOWP) model can be. For a given geographical area (or computational domain) the size of a numerical model is mostly determined by the model's horizontal grid size and number of vertical levels. Since there is a limitation to how many grid points we can store in the computer's RAM, there is a lower limit on the grid size we can use for a given computational domain. Implied is that if we would like to decrease the grid size for a given computer, the computational domain covered has to be shrinked. Likewise, if we would like to increase the computational domain the grid size has to be increased as well. Recall that the grid size puts a lower limit on the wavelengths a numerical model is able to represent. Hence the limited computer capacity puts a constraint on the scales a model possibly can resolve for a given geographical domain. In this respect the steady growth in computer capacity experienced since the birth of computers back in the 1940s has resulted in an ever increasing resolution to an extent that today's global atmospheric models more than adequately resolves lows and high. In fact the global atmospheric model run at the ECMWF (European Center for Medium range Weather Forecast), which produces a global weather forecast twice a day with a lead time of 15 days, features a grid size of about 16 km in the horizontal, a resolution one could only dream of in the 1970s.

The question therefore arises if there is a lower limit to the grid size needed? To guide us is the fact that the dynamical scales of the weather systems in the oceans and the atmosphere are mostly given by what is known as Rossby's deformation radius (cf. Section 6.4). Recall that at sub-polar latitudes the deformation radius in the atmosphere is about 500 - 1000 km and the typical time scale a few days. In contrast the deformation radius in the ocean at the same latitude is about 10 - 50 km, while the time scales are a few weeks to months. This is perhaps the main reason why NOWP models are less mature than NWP models. To illustrate this point let us consider a global model with a grid size of about 2 degrees (Figure 7.1 upper panel). A mesh size of 2 degrees, or about 200 km, entails that the grid size is about one fifth of the the atmospheric

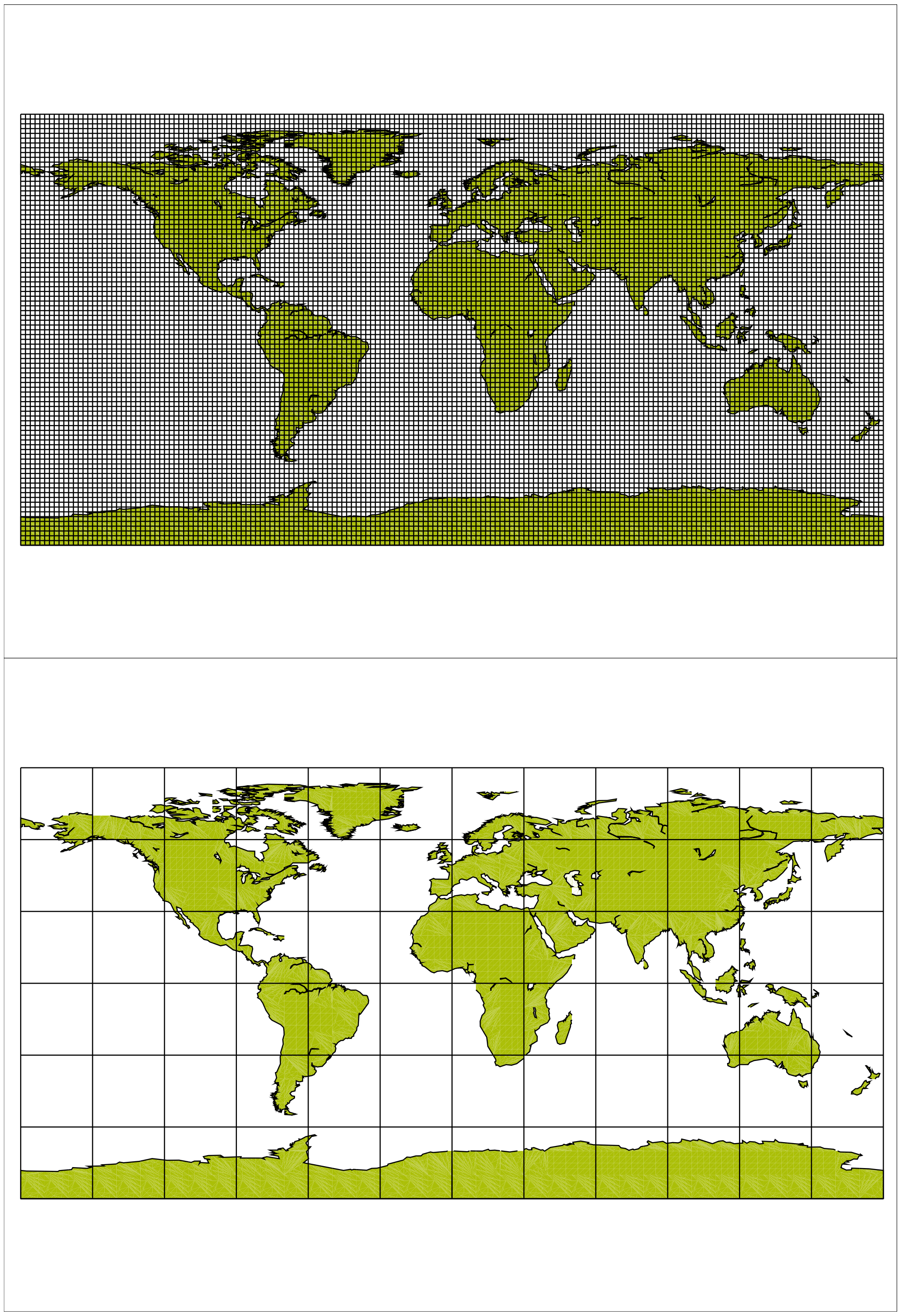


Figure 7.1: Upper panel shows the Earth's surface covered by a 2 degree mesh. Lower panel shows a similar mesh of 30 degrees mesh size. The figure conveniently illustrates how a 2 degree mesh in the ocean would look like in the atmosphere scaled by the Rossby radius of deformation.

Rossby radius. This is a tolerable grid size for a numerical atmosphere model. If we scale this to the Rossby radius of deformation in the ocean, the grid for the atmosphere model would look like the one displayed in the lower panel of Figure 7.1. As is evident the grid size of the latter is about 3-4 times the Rossby radius of deformation. No meteorologist in his right mind would consider it to be an adequate grid for a NWP model. To obtain a similar tolerable resolution in the ocean we have to employ grids of mesh sizes $\sim$2-4 km, or $\sim$1/200th of a degree. Hence for a given geographical region the amount of RAM required by an ocean model is much higher than an atmosphere model. In addition comes the fact that in order to satisfy the CFL criterion the time step is much smaller for an ocean model since the grid size is so much smaller. In practice it is a much greater computational effort to provide say a 24 hour "weather" forecast for the ocean for a given area on a given computer than to provide a similar weather prediction. To enable computers to provide numerical ocean weather forecasts as fast as today's NWP models for the same area we simply need faster computers. To make things even worse recall that the time scale in the ocean is much longer than in the atmosphere. A weather prediction of say ten days corresponds to an ocean forecasts of at least one month.

7.1 Open boundaries

In the infancy of NWP in the 1950s and 1960s the capacity of the computers were too limited to allow global atmospheric forecasts that resolved the Rossby radius to be run, that is to resolve the large scale weather systems (synoptic scales). Hence most national meteorological institutes at that time ran limited area NWP models. The boundaries af these models were therefore "open" in the sense that there was no natural boundary like for instance an impermable wall to help you specify a boundary condition. Fluid were therefore free to pass trough these boundaries, and hence they were denoted *open boundaries*. Nevertheless, in a mathematical sense, it still constituted a boundary at which a boundary condition had to be specified. Such boundary condition are denoted *open boundary conditions* or OBCs for short.

As alluded to in the in the introduction to this chapter, today's situation is quite different regarding atmospheric models. Global NWP models with more than adequate resolution to resolve the atmospheric weather systems or Rossby's deformation radius, also referred to as the synoptic scale, are in fact common today. These models therefore resolve Rossby's deformation radius everywhere including the poles. Nevertheless there are still processes on a scale much smaller than the synoptic scale that have a decisive impact on the local weather, notably processes associated with cloud formation (showers) and irregular topography, that is not yet resolved properly by the global models. Processes accociated with these scales not only impact the local weather, but equally important they also impact the large scale weather. Their effect is therefore parameterized in the global models.

Regarding ocean models the situation is somewhat similar. In fact running global NOWP models resolving the oceanic deformation radius at high latitudes are not yet feasible even on today's supercomputers. Thus oceanographers even today have to revert to limited area models to resolve the scale associated with the oceanic weather systems. In this light it appears that there is no lower limit in the quest for higher and higher resolution.

As a consequence both ocean and atmosphere models early on had to handle open boundaries. In particular they had to develop proper or correct OBCs, that is, OBCs that ensures, in a mathematical sense, that a solution of the child model's governing equations exists and is unique. Recall that at the open boundaries the governing equations are still valid. Nevertheless, since the computational domain of the child model ends at the open boundary the governing equations must be replaced by some kind of OBC there. From a physical point of view we would like the solution to be as close to the "correct" solution as possible. The correct solution refers to the one we would have obtained if the model was global with natural boundary conditions applied along its boundaries. It should be emphasized that this is a dilemma since the solution to the governing equations is determined not only by the equations themselves, but also by the boundary conditions as mentioned in Section 2.5. Thus when applying OBCs to determine the solution we are not ensured that the solution we obtain is the correct one. In fact it is impossible in general to prove that a solution even exists and is unique. The latter is only possible in special cases, for instance for very simplified linear systems.

To help in the development of a proper OBC the following definition of an open boundary, as formulated by *Røed and Cooper* (1986), is useful

An open boundary is a computational boundary at which disturbances originating in the interior of the computational domain are allowed to leave it without disturbing or deteriorating the interior solution.

7.2 Nesting techniques

Although the global models do not resolve the weather systems they do provide information on the scales larger than the weather systems. It was therefore early on recognized that the information inherent in these coarser mesh models should be exploited by the higher resolution, finer mesh limited area models. The answer is *dynamic downscaling* in which the results from a coarser mesh model is used to provide OBCs for the fine mesh models. These techniques are referred to as *nesting techniques*. Common today is to refer to the coarse mesh model as the "parent" model and to refer to the fine mesh model as the "child" model. If the parent is a "stand-alone" model, that is, there is no feedback from the child model, we refer to the nesting as a *one-way nesting* technique. This is in contrast to *two-way nesting techniques* in which the child is allowed to impact the parent. The latter is treated in Section 10.5 (page 172), while we in this section concentrate on the one-way nesting techniques.

The parent-child situation is vizualized in Figure 7.2 in which the parent covers the domain denoted Ω and the child covers the domain denoted ω. The open boundary or interface between them is denoted Γ. The task is then to provide an OBC on Γ so that results from the parent model somehow is transferred to the child model. This then becomes the condition on the boundary Γ that must be obeyed by child model. This is denoted dynamical downscaling since the child model provides a solution that is a consistent downscaling of the parent model dynamics down to scales that take into account the finer scales present in the child model for instance through a refinement of the topography. We emphasize that early on there was no parent models, but due

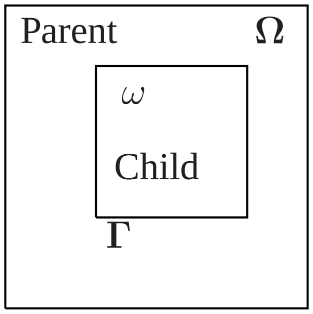


Figure 7.2: Sketch showing the confiduration of a Child model covering a domain ω embedded in a Parent model covering the domain Ω. Commonly the Child has a higher resolution than the Parent with a refinement factor of 3 to 5. The interface, denoted Γ, is then an open boundary at which an open boundary condition must be imposed.

to the limited computer capacity the boundary Γ was still an open boundary at which an OBC had to be specified.

We demand that the conditions we impose at open boundaries satisfy certain requirements. One obvious requirement, which follows directly from the definition above, is that disturbances originating in the interior of our domain propagating toward the open boundary should be allowed to pass through to the exterior without distorting or disturbing the interior solution. Equally obvious is that disturbances originating in the exterior domain is free to enter our domain without distortions. The latter is sometimes hard to achieve since we do not always have sufficient knowledge about the exterior solution.

To illustrate this let us consider a wave created in the child domain which propagates towards the open boundary. The condition we impose at the open boundary should then be able to let that wave pass through and not be reflected, that is, none of the energy contained in the wave should be allowed to be radiated back into the child domain. Likewise if a parent model exists a wave created in the parent domain should be free to enter the child without being distorted or damped. Next we require that the chosen OBC leads to a stable solution (numerically). Finally, from a mathematical point of view we require that the OBC together with the governing equations leads to a mathematical problem that is well posed or at least well-posed enough so that a solution exists and is unique.

7.3 Some historical notes

The very first attempt of making a numerical weather forecast was made by the Meteorological Research Group at the Institute for Advanced Study lead by John von Neumann at the Princeton University in the late 1940s. Their attempt is published in the now famous paper by *Charney et al.* (1950). Recall that just after the second world war the digital computers were in their infancy. They therefore employed a very simple atmospheric model compared to today's standards, and the domain was very limited. In fact they solved a finite difference approximation of the barotropic, quasi-geostrophic vorticity equation on a rectangular domain basically covering the North American continent. Thus the model featured four open boundaries to the north, south east and west where they had to apply OBCs. They opted for a set of OBCs which was to specify the potential vorticity at the inflowing boundaries and to apply a radiation condition (see Section 7.4 below) at outflowing boundaries. Later *Platzman* (1954) showed that their solution was unstable when applying the OBC they had chosen. Thus the problem of specifying OBCs that renders the solution unique without detoriating it is not new.

Since this first application, where the OBC was shown to play a crucial part, there has been a lot of research on OBCs, e.g., (e.g., *Davies*, 1976, 1985; *Orlanski*, 1976; *Sundström and Elvius*, 1979; *Hedstrøm*, 1979; *Røed and Smedstad*, 1984). Regarding ocean models reviews are offered by *Chapman* (1985), *Røed and Cooper* (1986, 1987), *Palma and Matano* (2000) and *Blayo and Debreu* (2005, 2006). For the one-way nesting techniques applied to ocean modeling the recent paper by *Mason et al.* (2010) is enlightning, while regarding two-way nesting techniques *Debreu and Blayo* (2008) and most recently *Debreu et al.* (2012) is worth reading. As a consequence most national meteorological institutes today runs limited area weather forecast models nested into a global model[1]. The global model is then the parent model and the limited area model the child model, the latter covering the area of interest to that particular nation with a much higher resolution (smaller grid size). The nesting technique used is mostly one-way, simply because most nations use forecasts from a global model which is not run inhouse, for instance by the ECMWF. This is also the case regarding ocean forecasting. For instance, recently the European Community established the Copernicus Marine Core Service in which several European institutions collaborate to provide regional ocean weather forecasts for European Seas[2] which include a global ocean model into which the regional models are nested.

In the following we will give some details regarding some of the common OBCs developed over the years. Towards the end we give details about two of the most promising OBCs that are useful as nesting techniques as well as OBCs (Sections 7.7 and 7.8). These are the Flow Relaxation Scheme (e.g., *Davies*, 1976, 1985) and the weakly reflective approach (e.g., *Navon et al.*, 2004; *Blayo and Debreu*, 2006). While the former OBC was developed in the meteorological community (*Davies*, 1976, 1985), the latter comes from the electromagnetic field (*Berenger*, 1994). In Chapter 10 Section 10.5 we in addition give a glimpse of the two-way nesting technique (cf. *Debreu et al.*, 2012) that we mentioned in Section 7.2 above.

7.4 Radiation conditions

Many of the processes in the ocean and atmosphere are processes involving wave propagation in one way or another. The early attempts at developing OBCs were therefore based their OBC

[1]At the Norwegian Meteorological Institute the limited area model at the time of writing is AROME-MetCoOp, a 2.5 km mesh size, non-hydrostatic (convective-scale) weather prediction model nested into the ECMWF model. Forecast provided by this model is what you get when looking at yr (http://www.yr.no/) for the Norwegian forecast area.

[2]Daily updated forecasts are available at http://marine.copernicus.eu/

formulations on simple wave equation. In its simplest form the wave equation reads,

$$\partial_t \phi + c_\phi \partial_n \phi = 0 \tag{7.1}$$

Here ϕ represents the dependent variable, c_ϕ is the component of the phase velocity normal the boundary associated with the variable ϕ, while ∂_n denotes the derivative normal to the open boundary. Imposing (7.1) as an OBC it becomes what is known as the *radiation condition*. When use is made of (7.1) as an OBC we fundamentally assume that the disturbances passing through the open boundary consists of waves. Note that the disturbances passing through the boundary may consist of several waves of different wavelengths, and hence that (7.1), strictly speaking, is only valid for one Fourier component only. It is thus only suitable for linear problems in which there is no energy exchange between wave numbers.

One of the first obstacles in employing (7.1) as our OBC is that we do not know the phase velocity c_ϕ. From Section 5.14 we recall that c_ϕ is the slope of the characteristics. Thus if the choice of c_ϕ perfectly matches the slope of the characteristics then (7.1) is a perfect open boundary condition. However, it is only for very simple, physical problems, e.g., for a monochromatic wave problem, that we are able to determine the characteristics *a priori*, and hence c_ϕ is generally unknown.

We immediately recognize that (7.1) contains two special cases. The first case is $c_\phi = 0$, while the second is the opposite, namely when the phase velocity $c_\phi \to \infty$. In the former case we notice that the characteristics are straight vertical lines in x, t space, and that we may integrate (7.1) in time to give,

$$\phi = \text{const.} \tag{7.2}$$

Thus under these circumstances the dependent variable is known for all times at the OBC, and we recall from Section 2.5 that the OBC is a Dirichlet condition. Commonly this is referred to as a *clamped condition* since the dependent variable ϕ does not change in time at the OB.

In the latter case when $c_\phi \to \infty$ we notice that characteristics are horizontal straight lines in the x, t space. We notice from (7.1) that if $\partial_t \phi$ should remain finite we must require that the gradient $\partial_n \phi$ must be zero, and hence that

$$\partial_n \phi = 0. \tag{7.3}$$

Usually the condition (7.3) is referred to simply as a *gradient condition*. We recall from Section 2.5 that such a condition was referred to as a Neumann condition.

If the phase velocity is finite and differs from zero, then we have a true radiation condition. The problem is then reduced to determine the phase velocity c_ϕ. If the solution is in the form of known waves, say a barotropic Kelvin wave[3]. Under these circumstances the phase velocity is known, and in the case of a Kelvin wave it is,

$$c_\phi = c_0 = \sqrt{gH} \tag{7.4}$$

where g is the gravitational acceleration, and H is the equilibrium depth of a fluid column.

[3]A barotropic Kelvin wave is common phenomena in oceanography. It belongs to the class of planetary gravity waves. Kelvin waves are commonly filtered out in meteorology models.

Let us consider a problem of a fluid contained in a channel of equilibrium depth H. Furthermore, let us consider a frictionless motion and let h denote the total depth or layer thickness of a fluid column and u the speed of the fluid column[4]. Moreover, let us consider that the motion is on a non-rotational Earth, and that the fluid has constant and uniform density. Then the governing equations may be written (cf. Section 6.2)

$$\partial_t u = -g\partial_x h \tag{7.5}$$
$$\partial_t h = -H\partial_x u \tag{7.6}$$

The classic method to solve the above set is to first differentiate (7.5) with respect to x and (7.6) with respect to t and then add the results. The result is

$$\partial_t^2 h - c_0^2\partial_x^2 h = 0, \tag{7.7}$$

that is, a wave equation with a phase speed equal to c_0 as given in (7.4). The set (7.5) and (7.6) thus requires two boundary conditions in space. Let us assume that the channel has two open boundaries at $x = 0$ and $x = L$. The natural boundary condition at these two boundaries is then the radiation condition (7.1) with a phase speed of $\pm c_0$, respectively.

Recalling that the phase velocity is determined by the slope of the characteristics, we may also use the semi-Lagrange technique or method of characteristics to find a useful boundary condition. In fact, as we show, we end up by imposing the radiation condition at the two open boundaries. We start by recalling that the compatibility equations and the characteristic equations for the simple system (7.5) and (7.6) are given by (6.71) and (6.72). Revalling that $\phi = gh$ we get

$$\partial_t\left(u + c_0\frac{h}{H}\right) + \frac{D_1^* x}{dt}\partial_x\left(u + c_0\frac{h}{H}\right) = 0 \quad \text{along } \frac{D_1^* x}{dt} = c_0, \tag{7.8}$$
$$\partial_t\left(u - c_0\frac{h}{H}\right) + \frac{D_2^* x}{dt}\partial_x\left(u - c_0\frac{h}{H}\right) = 0 \quad \text{along } \frac{D_2^* x}{dt} = -c_0. \tag{7.9}$$

While (7.8) describes a wave propagating in the positive x-direction with phase velocity c_0, we observe that (7.9) describes a wave propagating in the opposite direction, but with the same phase velocity. In particular we notice that (7.8) and (7.9) express that the specific combinations of the dependent variables u and h, namely the Riemann invariants $u \pm c_0\frac{h}{H}$, are conserved along their respective characteristics.

Let us assume that our problem is to solve (7.5) for $0 < x < L$ and that the two boundaries $x = 0$ and $x = L$ are open. Let us in addition assume that a motion is generated in the interior of the domain, e.g., in the form of an initial deviation of the layer thickness h locally. The question then arises: what is the correct boundary condition to impose on the two open boundaries? We know from the two compatibility equations (7.8) and (7.9) that the information about the deviation will propagate along the two characteristics. Towards the right-hand boundary at $x = L$ the information will propagate along $\frac{D_1^* x}{dt} = c_0$ characteristic, and towards the left-hand boundary

[4]Since we consider a frictionless motion we may safely assume that u is independent of depth.

$x = 0$ along $\frac{D_2^* x}{dt} = -c_0$ characteristic. To avoid reflection we must impose a condition that ensures that information cannot propagate back into our interior domain from the boundary point. Since information propagates along the characteristics, we must ensure that no characteristics at $x = 0$ or $x = L$ slopes towards the interior. Consequently we require that

$$\frac{D_2^* x}{dt} = 0 \quad \text{at} \quad x = L, \tag{7.10}$$

and

$$\frac{D_1^* x}{dt} = 0 \quad \text{at} \quad x = 0. \tag{7.11}$$

Substituting this into the left-hand sides of (7.8) and (7.9), respectively, we get

$$\partial_t \left(u - c_0 \frac{h}{H} \right) = 0 \quad \text{at} \quad x = L \tag{7.12}$$

and

$$\partial_t \left(u + c_0 \frac{h}{H} \right) = 0 \quad \text{at} \quad x = 0. \tag{7.13}$$

We now integrate (7.12) and (7.13) in time and get

$$u = c_0 \frac{h}{H} + \text{const.}, \quad \text{at} \quad x = L \tag{7.14}$$

and

$$u = -c_0 \frac{h}{H} + \text{const.} \quad \text{at} \quad x = 0. \tag{7.15}$$

This is in fact the radiation condition. Indeed if we substitute the expression (7.6) for $\partial_t h$ into (7.12) we get (7.1) with $\phi = u$ and $c_\phi = c_0$.

The advantage of using the method of characteristics to derive the non-reflective boundary condition is that it gives us insight into how to construct open boundary conditions in general. This was for instance exploited by *Røed and Cooper* (1987) to construct a weakly reflective boundary condition for a more general problem including the effect of Earth's rotation based on earlier work by *Hedstrøm* (1979) (cf. Section 7.8).

7.5 Implementation of the radiation condition

We now consider the numerical implementation of the one-dimensional version of the radiation condition (7.1), and that the space variable is x. In this we essentially follow the implementation given in *Røed and Cooper* (1987). Recall that the radiation condition then reads

$$\partial_t \phi + c_\phi \partial_x \phi = 0. \tag{7.16}$$

To get started let us assume that the computational domain is $x \in < 0, L >$ and $t \in < 0, T >$. The boundaries are then at $x = 0, L$, where we assume that $x = L$ is an open boundary, while

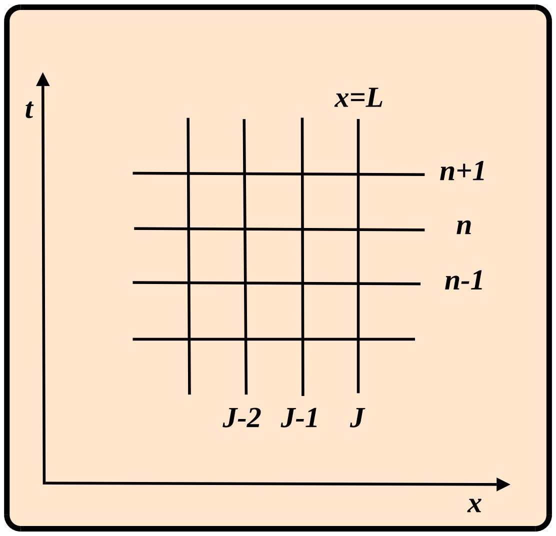


Figure 7.3: Sketch of the mesh in the t, x plane close to the right-hand open boundary. The computational domain is then to the left of $x = L$. The letters J, $J-1$, and $J-2$ denote grid points respectively at the open boundary, the first and second points inside the computational domain, while n, $n-1$, and $n+1$ denote the time levels.

$x = 0$ is a natural boundary. Furthermore we construct a grid in the x, t coordinates where $x_j = (j-1)\Delta x$ and $t^n = n\Delta t$ (cf. Fig. 7.3).

Since (7.16) is an advection equation it is natural that we use one of the stable schemes developed in Section 5.2. We emphasize that it is important that the interior scheme and the scheme we use to solve the radiation equation has the same accuracy. Thus if the interior scheme is of second order accuracy in time and space then it is natural that we choose the leapfrog scheme. If the interior scheme is first order in time and space then it is natural that we choose a similar scheme for the radiation condition (7.16), say the upwind scheme. In the following we assume that the latter is true.

We then proceed using the upwind scheme for the radiation condition at $x = L$, that is for $j = J$. Assuming that $c_\phi \geq 0$, and following the notation of Figure 7.3, we get

$$\frac{\phi_J^{n+1} - \phi_J^n}{\Delta t} + c_\phi \frac{\phi_J^n - \phi_{J-1}^n}{\Delta x} = 0 \tag{7.17}$$

or

$$\phi_J^{n+1} = (1 - r_\phi)\phi_J^n + r_\phi \phi_{J-1}^n \tag{7.18}$$

where

$$r_\phi = c_\phi \frac{\Delta t}{\Delta x}. \tag{7.19}$$

Equation (7.18) says that the radiation condition in essence is an interpolation of values from the interior and at previous times. The problem is that we don't know the weighting function, that is, the phase velocity c_ϕ? As suggested by *Orlanski* (1976) we might solve (7.12) with respect to the phase velocity (or r_ϕ) and get

$$r_\phi = -\frac{\phi_J^{n+1} - \phi_J^n}{\phi_J^n - \phi_{J-1}^n}. \tag{7.20}$$

However, since we do not know the solution at the boundary at time level $n+1$ this expression is useless. Our only way of determining c_ϕ (or r_ϕ) is to use our knowledge about the solution at previous times. We then have several options. One is to use interior points at the same time level, in which case

$$r_\phi = -\frac{\phi_{J-1}^{n+1} - \phi_{J-1}^n}{\phi_{J-1}^n - \phi_{J-2}^n}. \tag{7.21}$$

A second is to use information at previous times at same points in space,

$$r_\phi = -\frac{\phi_J^n - \phi_J^{n-1}}{\phi_J^{n-1} - \phi_{J-1}^{n-1}}. \tag{7.22}$$

Both of these expressions provides an expression for the phase velocity. But which is the correct one? If we interpret (7.21) and (7.22) in terms of characteristics as in the previous section (see also Section 5.12), we notice that (7.21) assumes that the slope of the characteristic through the (x_J, t^{n+1}) point to a first approximation equals the slope through the (x_{J-1}, t^{n+1}) point, while (7.22) assumes that it to a first approximation equals the slope through the point (x_J, t^n). Following this argument a third option is to assume that the characteristic through (x_J, t^{n+1}) continues backward in times and crosses the time level $n-1$ between x_{J-1} and x_{J-2}. This is tantamount to assume that to a first approximation the slope through (x_J, t^{n+1}) equals the slope through (x_{J-1}, t^n). We then get the following expression for the phase velocity

$$r'_\phi = -\frac{\phi_{J-1}^n - \phi_{J-1}^{n-1}}{\phi_{J-1}^{n-1} - \phi_{J-2}^{n-1}} \tag{7.23}$$

On purpose we have used a prime for this expression, since it is a predictor for the phase velocity. We must require that the number returned is not negative. Hence we correct the result by defining r_ϕ (no prime attached) as

$$r_\phi = \begin{cases} r'_\phi & ; \quad 0 \le r'_\phi \\ 0 & ; \quad r'_\phi < 0 \end{cases}. \tag{7.24}$$

As argued by *Røed and Cooper* (1987) we think (7.24) is a better approximation. Consequently, we use the expression (7.24) to substitute for r_ϕ in (7.18) when determining the new boundary value ϕ_J^{n+1} at time level $n+1$. If our open boundary was at $x = 0$ the inequality sign in (7.24) must be reversed to ensure that the phase velocity then is negative.

As alluded to it is only pure wave problems where processes like non-linear interactions, friction, wind forcing and the Coriolis acceleration are neglected that satisfies the radiation condition (7.1). All realistic models employed today within oceanography or meteorology are much more complex, and includes at least the processes just mentioned, and in most cases many more. The radiation condition is therefore far from being a perfect open boundary condition[5].

[5]In fact it may be shown that there is no such thing as a perfect boundary condition mathematically speaking, since the problem in a geophysical context is ill posed.

Since the radiation condition in most cases is far from being perfect the meteorological and oceanographic communities has developed several other optional OBCs (e.g., *Chapman*, 1985; *Røed and Cooper*, 1986, 1987; *Palma and Matano*, 2000; *Blayo and Debreu*, 2006). In the following sections we study some of the more popular ones.

7.6 The sponge

One of the most popular ones is the so called *sponge condition*. In essence the method is to extend the computational domain outside of the area of interest (interior domain) to include an area where the energy leaving the interior domain is gradually decreased so as to avoid reflection. This is what happens to, e.g., waves impinging on a sandy beach. In practice we achieve this by gradually increasing the relative importance of those terms associated with energy extraction, e.g., frictional processes, as the a disturbance is advected or propagated into the exterior or extended domain (sometimes referred to as the sponge layer).

As an example let us study the non-rotating shallow water equations, that is,

$$\partial_t u = -g\partial_x h \tag{7.25}$$

$$\partial_t h = -H\partial_x u. \tag{7.26}$$

Let our domain of interest be $x \in \langle -L_i, L_i \rangle$. There are however no physical boundaries at the two boundaries $x = -L_i, L_i$. Hence these boundaries are open implying that the governing equations (7.25) and (7.26) are valid outside of our domain of interest as well. We may therefore extend the *computational* domain to $x \in \langle -L_e, L_e \rangle$ in which $L_e > L_i$ so that our domain of interest becomes a subdomain. Furthermore we may add friction to the problem *outside* of our domain of interest, for instance in terms of Rayleigh friction[6]. To avoid loosing mass, and since (7.26) is the mass conserving equation, we only add Rayleigh friction to the momentum equation, that is, we replace (7.25) and (7.26) by the equations

$$\partial_t u = -g\partial_x h - \gamma u, \tag{7.27}$$

$$\partial_t h = -H\partial_x u, \tag{7.28}$$

to be solved within the extended domain $x \in \langle -L_e, L_e \rangle$. Furthermore by letting the frictional parameter γ be zero within $x \in \langle -L_i, L_i \rangle$ (7.27) and (7.28) reduces to (7.25) and (7.26) within our domain of interest. However, outside of our domain of interest, that is, in the exterior domains $x \in \langle -L_e, -L_i \rangle$ and $x \in \langle L_i, L_e \rangle$ we let γ gradually and monotonically increase as we get farther and farther away from the open boundaries. For instance we achieve this by using

$$\gamma = \gamma_0 \begin{cases} 1 - e^{-\lambda(x+L_i)} & ; \quad -L_e \le x < -L_i \\ 0 & ; \quad -L_i \le x \le L_i \\ 1 - e^{\lambda(x-L_i)} & ; \quad L_i < x \le L_e \end{cases}, \tag{7.29}$$

[6]Rayleigh friction means a term proportional to the variable in question. For instance if the variable is the velocity u the Rayleigh friction term would be $-\gamma u$ as, e.g., given in (7.27). We also note that the solution to the equation $\partial_t u = -\gamma u$ is $u = u_0 e^{-\gamma t}$ where u_0 is the intial velocity. Thus Rayleigh friction decreases the kinetic energy of all wavelengths equally.

The frictional parameter thus increases exponentially from zero at the two open boundaries at $x = -L_i$ and $x = L_i$ to $\gamma_0(1 - e^{\lambda(L_e - L_i)})$ at the two boundaries of the computational or extended domain. Any wave or disturbance created inside the interior domain will therefore be damped by friction as it propagates into the sponge layers, and increasingly so as it progresses farther away from the two open boundaries. We note that, together with the width of the sponge layers, the parameters λ and γ determine how fast or quickly the frictional effect increases within the two sponge layers $x \in \langle -L_e, -L_i \rangle$ and $x \in \langle L_i, L_e \rangle$. To avoid reflection of disturbances that eventually may deteriorate the interior solution it is extremely important that these parameters are set so that the frictional effect only slowly takes effect as the disturbances progresses into the sponge.

We can derive an analytic solution to (7.27) and (7.28). We start by differentiating (7.28) with respect to time, and then substitute for $\partial_t u$ from (7.27). We then get

$$\partial_t^2 h + \gamma \partial_t h = gH \partial_x^2 h. \tag{7.30}$$

Searching for wave like solution we let

$$h = h_0 e^{\omega t} e^{i\alpha x}, \tag{7.31}$$

where α is the wavenumber and ω is a complex frequency. Substituting this expression into (7.30) we get the dispersion relation

$$\omega^2 + \gamma\omega + gH\alpha^2 = 0, \tag{7.32}$$

which gives the two solutions

$$\omega_{1,2} = -\frac{1}{2}\gamma \pm i\alpha c \tag{7.33}$$

where

$$c = \sqrt{gH - \left(\frac{\gamma}{2\alpha}\right)^2}. \tag{7.34}$$

Thus the solution in terms of the height of a fluid column is hence

$$h = h_0 e^{-\frac{1}{2}\gamma t} e^{i\alpha(x \pm ct)}, \tag{7.35}$$

where h_0 is a constant. We observe from (7.35) that the solution consists of two waves travelling in opposite direction. We also observe the amplitude of the waves within the sponges where $\gamma \neq 0$ domain decreases exponentially in time. Furthermore, we notice that phase velocity (7.34) decreases with increasing γ. Thus, as the wave propagates deeper into the sponge areas, it slows down as well as being damped in amplitude.

Since application of the sponge condition as an open boundary condition requires that the sponge zone is of a certain extension it adds computer time to solve our problem. Hence adding sponges slows down the wall clock time. Another problem with the sponge condition is that if the solution consists of forced waves (cf. *Røed and Cooper*, 1986), for instance is governed by equations like

$$\partial_t u = -g\partial_x h - \gamma u + \tau \tag{7.36}$$

$$\partial_t h = -H\partial_x u \tag{7.37}$$

where τ represents the forcing, then the solution in the sponge layer of the former wave solution (7.35) and a solution dominated by a balance between the forcing term and the frictional term, that is, $u = \tau/\gamma$, which implies that as γ increases u decreases so that mass (volume) accumulates within the sponge zone. For longer term integrations this accumulation of volume changes the pressure forcing and sooner or later this will have an impact on the interior solution as well. It is important to bear this fact in mind when we in the next section discuss the Flow Relaxation Scheme (FRS).

7.7 The Flow Relaxation Scheme

We now construct an OBC that was first suggested by *Davies* (1976). As shown below it is somewhat similar to the sponge OBC in two respects. First it requires us to extend the computational domain to include an exterior domain or buffer zone. Second it is in essence a sponge in which the solution is suppressed as it progresses into the buffer zone. The method is called the *Flow Relaxation Scheme* commonly abbreviated to FRS (*Davies*, 1976; *Martinsen and Engedahl*, 1987; *Cooper and Thompson*, 1989; *Engedahl*, 1995a; *Shi et al.*, 1999, 2001). In particular the two latter references are useful in that they give a detailed description of the FRS and in addition gives a nice example of its use as an OBC.

One of the advantages of the FRS compared to for instance the sponge is that it allows us to specify an exterior solution. The FRS can therefore be used as a one way nesting condition, in which an exterior solution is specified by, e.g., a courser grid model covering a much larger area. The FRS as a nesting technique is for instance used as the main method whereby information from global and semi-global models is transferred to regional models at Norwegian Meteorological Institute. This is true for both their numerical weather prediction (NWP) models as well as their numerical ocean weather prediction (NOWP) models[7].

In essence the method, just as the sponge method, only modifies the *numerical* solution in a buffer or relaxation zone. In this zone, commonly referred to as the FRS zone, the solution is, for each time step, relaxed toward a specified exterior solution. The FRS zone is commonly not too wide, but should at least contain 7 grid points. We emphasize that the FRS zone is an extension of the interior domain and thus extends the computational domain (cf. Figure 7.4). Within the FRS zone the solution is relaxed towards an a priori specified exterior solution[8]. The relaxation is performed by specifying a weighting function that for each grid point in the FRS zone computes a weighted mean between the specified outer solution and the interior solution computed from the governing equations.

Let $\phi(x,t)$ be the dependent variable in our problem and let the interior domain or our domain of interest be $x \in < -L_i, L_i >$, where $x = -L_i, L_i$ are open boundaries. As displayed in Figure 7.4 the FRS zones extend the interior domain so that the computational domain is increased to the left and right by adding FRS zones. The FRS zone to the left starts at $x = -L_e$ and ends at $x = -L_i$, while the the FRS zone to the right starts at $x = L_i$ and ends at $x = L_e$. We note that this is quite similar to the addition of sponge layers as we did in Section 7.6. As usual we define

[7] cf. http://met.no/

[8] Often also referred to as the outer solution

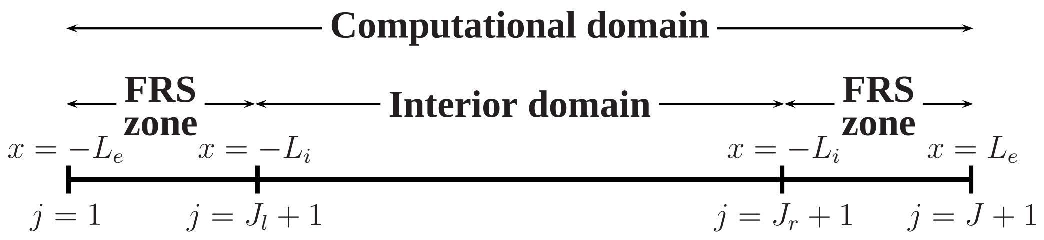


Figure 7.4: Sketch of the FRS zone, the interior domain and the computational domain. Also shown are the appropriate indices.

the grid points by $x_j = (j-1)\Delta x$, where the index j counts all grid points of the computational domain starting with $j = 1$ at $x = -L_e$ at the leftmost boundary of the computational domain and ending with $j = J + 1$ at the rightmost boundary. Furthermore we let $j = J_l + 1$ be associated with $x = -L_i$, the left-hand open boundary, and $j = J_r + 1$ be associated with $x = L_i$, the right-hand open boundary.

As alluded to the FRS allows us to specify an outer solution, which can be the result of another numerical model covering a larger domain than our interior domain. We denote this exterior solution by ${\phi_e}_j^n$ which emphasize that the outer solution is a function of space and time. Let us now assume that we have computed all the our dependent variable ϕ_j^n at all points, including the FRS zones at time level n. Furthermore using the governing equation of our model we can derive a solution at the next time level including the FRS zone except of course the end points at $j = 1$ and $j = J + 1$. We denote this predictor by ϕ_j^*. We underscore that ϕ_j^* is computed at all points $j = 2(1)J$, that is, all points except the end points of the FRS zones. The next step is to correct the solution by computing our dependent variable as a weighted mean between our predicted solution ϕ_j^* and the specified outer solution ${\phi_e}_j^n$ to derive the final or corrected solution at time level $n + 1$. We do this by employing the formula

$$\phi_j^{n+1} = (1 - \alpha_j)\phi_j^* + \alpha_j {\phi_e}_j^{n+1} \quad ; \quad j = 1(1)J, \tag{7.38}$$

where $0 \leq \alpha_j \leq 1$ is a relaxation parameter so that $\alpha_1 = \alpha_{J+1} = 1$ and so that $\alpha_j = 0$ for all grid points within the interior domain including the open boundaries, that is, for $J_l + 1 \leq j \leq J_r + 1$. We also require that the relaxation parameter increases monotonically in the FRS zones. Since the relaxation parameter $\alpha_1 = \alpha_{J+1} = 1$, we note from (7.38) that the solution at time level $n + 1$ equals the specified outer solution at the end points of the FRS zones, e.g., $\phi_1^{n+1} = {\phi_e}_1^{n+1}$. Similarly we notice that at in the interior and including the open boundary points the solution equals the interior solution, or $\phi_j^{n+1} = \phi_j^*$ for $J_l + 1 \leq j \leq J_r + 1$.

Experiments employing the FRS, e.g., *Martinsen and Engedahl* (1987), *Engedahl* (1995a), show that the solution is sensitive to the the distribution of the specified weighting function α throughout the FRS zone. They found that distributing α applying a hyperbolic tangent function, that is,

$$\alpha_j = 1 - \tanh \frac{j-1}{2} \quad ; \quad j = 1(1)JM, \tag{7.39}$$

is a good choice. Furthermore, in similarity with the sponge method, they also found that the solution is sensitive to the width of the FRS zone. They concluded that for oceanic application the width of the FRS zone should be at least seven grid points, that is, $J_l \geq 7$.

In similarity with the sponge condition one of the disadvantages of employing FRS as an OBC (or nesting technique) is that the computational domain is increased, and hence that the computational burden is increased. This disadvantage is, however, somewhat suppressed by the fact that the FRS allows us to specify an outer solution. As shown below this can be used to effectively minimize possible errors due to reflection of disturbances. Another disadvantage is that the solution, in similarity with the sponge method, does not conserve fundamental properties such as volume (or mass).

As an example let us study the numerical solution of the continuous problem

$$\partial_t \phi = \mathcal{L}[\phi] \quad ; \quad x \in < L_i, L_i >, \tag{7.40}$$

where $\mathcal{L}$ is a spatial differential operator. Furthermore, let us assume that the open boundary is at $x = -Li$ and that $x = L_i$ is a natural boundary, that is, that we are left with only one open boundary at $j = J_l + 1$, while at the right-hand boundary at $x = L_i$ (or $j = J_r + 1$) a natural boundary condition applies. As above we let ${\phi_e}_j^n; \quad j = 1(1)J_l$ denote the specified exterior solution. If we solve (7.40) applying a forward in time finite difference scheme we get

$$\phi_j^* = \phi_j^n + \Delta t \mathcal{L}_j^n \quad ; \quad j = 2(1)J_r, \tag{7.41}$$

where ϕ_j^* is the predictor. We then correct the predictor by applying the relaxation formula (7.38). We then get

$$\phi_j^{n+1} = (1 - \alpha_j)\phi_j^* + \alpha_j(\phi_e)_j^{n+1} \quad ; \quad j = 1(1)J_r. \tag{7.42}$$

To ensure that we do no corrections to the predictor within the interior domain we let the relaxation parameter be given by

$$\alpha_j = \begin{cases} 1 - \tanh\left(\frac{j-1}{2}\right) & ; \quad j = 1(1)J_l \\ 0 & ; \quad j = J_l(1)J_r \end{cases} \tag{7.43}$$

We may now use the expression on the right-hand side of (7.41) to substitute for ϕ_j^* in (7.42). If we in addition add the zero $(\alpha_j \phi_j^{n+1} - \alpha_j \phi_j^{n+1})$ to the left-hand side of (7.42) we get

$$\frac{\phi_j^{n+1} - \phi_j^n}{\Delta t} = \mathcal{L}_j^n + \gamma_j \left({\phi_e}_j^{n+1} - \phi_j^{n+1}\right); \quad j = 1(1)J_l \tag{7.44}$$

where the coefficient γ_j is defined by,

$$\gamma_j = \frac{\alpha_j}{1 - \alpha_j}. \tag{7.45}$$

If we now let Δt and Δx tend to zero, we notice that (7.44) is a forward in time, finite difference approximation to the continuous equation

$$\partial_t \phi = \mathcal{L}[\phi] + \gamma\,(\phi_e - \phi) \quad ; \quad x \in < -Le, L_i > . \tag{7.46}$$

We observe that except for the additional "frictional" term $\gamma\,(\phi_e - \phi)$ equation (7.46) equals (7.40). We also notice that the additional term is proportional to the difference between the interior solution and the exterior solution, and that the proportionality factor (7.45) varies from zero at the open boundary ($x = -L_i$) to infinity at the edge of the FRS zone. Thus the relative importance of the frictional term increases as we progress into the FRS zone.

If we now specify an exterior solution as constant and zero, then (7.46) is turned into

$$\partial_t \phi = \mathcal{L}[\phi] - \gamma\phi. \tag{7.47}$$

Under these circumstances the FRS acts like a sponge with a frictional parameter γ which gradually and monotonically increases towards infinity as we progress into the FRS zone, not unlike the exponential function specified in Section 7.6.

To illustrate the non-conservative properties of the FRS we again use the example problem governed by (7.5) and (7.6), that is, the non-rotating shallow water equations,

$$\partial_t u = -g\partial_x h \tag{7.48}$$
$$\partial_t h = -H\partial_x u. \tag{7.49}$$

To get started we notice first that as Δx and Δt tend to zero (7.42) takes on the form

$$\phi = (1-\alpha)\phi^* + \alpha\phi_e. \tag{7.50}$$

We now have two dependent variables, namely h and u, hence

$$h = (1-\alpha)h^* + \alpha h_e, \tag{7.51}$$

$$u = (1-\alpha)u^* + \alpha u_e. \tag{7.52}$$

Substituting for u from (7.52) into (7.49) we get

$$\partial_t h = -H\partial_x u = -H\partial_x u^* + (u_e - u^*)\partial_x \alpha \tag{7.53}$$

The first term on the right hand-side of (7.53) is the term we would have obtained if we make no relaxation ($\alpha = 0$). However, since the relaxation parameter is a function of x also a term appears that contains the divergence of the relaxation parameter α. Thus, unless $u^* = u_e$, the volume (or mass) conservation, as expressed by (7.49), is violated. Thus the varying relaxation paramater builds up volume, which in turn build up a artificial pressure for in the FRS zone that may eventually lead to currents deteriorating the interior solution. To avoid this violation of the mass conservation to impact our interior solution we must ensure that the relaxation parameter α is a very slowly varying function close to the open boundary. In turn this implies that the width of the FRS zone must be long enough for this to be realized. Again this is in similarity to the sponge.

Finally we notice that if the exterior solution is equal to or close to the true solution, then the friction term in (7.47) disappear as do the false divergence in (7.53). Under these circumstances the FRS is close to being a perfect open boundary or nesting condition. Thus the usefulness of the FRS depends to a certain extent on how good we are to "guess" the exterior solution. This is the reason why the FRS is mostly used as a one way nesting condition. When specifying the exterior solution to be the solution of the same governing equation, albeit for a coarser mesh, we ensure that the exterior solution is indeed close to the interior solution of the fine mesh model embedded in the coarser grid.

7.8 A weakly reflective OBC

As alluded to in Section 7.4 we may use the method of characteristics to construct a weakly reflective OBC also for problems including non-linearities, Coriolis effects and forcing (cf. *Røed and Cooper*, 1987).

As an example let us study the full shallow water equation. Thus we start with the equations

$$\begin{aligned} \partial_t u + u\partial_x u - fv &= -g\partial_x h + F^x, & (7.54)\\ \partial_t v + u\partial_x v + fu &= F^y, & (7.55)\\ \partial_t h + \partial_x(hu) &= 0, & (7.56)\end{aligned}$$

where F^x, F^y are the forcing terms, and f is the Coriolis parameter. We immediately recognize the system (7.54) - (7.56) as the non-linear, rotating shallow water equations for a barotropic fluid as given in Section 6.4 page 115 with the addition of the forcing terms. Hence the compatibility and characteristic equations are

$$\frac{D^*_{1,2}}{dt}(u \pm 2c) = fv + F^x \quad \text{along} \quad \frac{D^*_{1,2}x}{dt} = u \pm c. \tag{7.57}$$

To avoid reflections at, say $x = L$, we require

$$\frac{D^*_2 x}{dt} = 0 \quad \text{at } x = L. \tag{7.58}$$

By substitution of this expression in (7.57) the weakly reflective open boundary condition becomes,

$$\begin{aligned} \frac{D^*_1}{dt}(u + 2c) &= fv + F^x, \quad \text{along} \quad \frac{D^*_1 x}{dt} = u + c, & (7.59)\\ \partial_t(u - 2c) &= fv + F^x, \quad \text{at} \quad x = L. & (7.60)\end{aligned}$$

To find u and h at the boundary $x = L$ numerically, that is, for $j = J + 1$, we make finite difference approximations of (7.59) and (7.60). Thus

$$u^{n+1}_{J+1} + 2c^{n+1}_{J+1} = u^n_Q + 2c^n_Q + (fv + F^x)^n_Q \,\Delta t, \tag{7.61}$$

and

$$u_{J+1}^{n+1} - 2c_{J+1}^{n+1} = u_{J+1}^{n} - 2c_{J+1}^{n} + (fv + F^x)_{J+1}^{n}\Delta t. \tag{7.62}$$

As before we find the position of the point x_Q ($j = j_Q$) by utilizing the characteristic equation (7.59), that is,

$$x_Q = x_j - (u_{J+1}^{n} + c_{J+1}^{n})\Delta t, \tag{7.63}$$

Solving (7.61) and (7.62) with respect to u_{J+1}^{n+1} and c_{J+1}^{n+1} we get

$$u_{J+1}^{n+1} = \frac{1}{2}\left\{u_Q^n + u_{J+1}^n + 2\left(c_Q^n - c_{J+1}^n\right) + \left[(fv + F^x)_Q^n + (fv + F^x)_{J+1}^n\right]\Delta t\right\} \tag{7.64}$$

$$c_{J+1}^{n+1} = \frac{1}{4}\left\{u_Q^n - u_{J+1}^n + 2\left(c_Q^n + c_{J+1}^n\right) + \left[(fv + F^x)_Q^n - (fv + F^x)_{J+1}^n\right]\Delta t\right\} \tag{7.65}$$

It remains to find u_Q^n and c_Q^n. As explained in Section 5.12 we find these by a interpolation using the adjacent grid points, for instance by using Newton's two point interpolation formulae or higher order interpolation. It should be noted that since we make use of finite difference approximations, we do not solve (7.59) and (7.60) to perfection, and thus some weak reflections is unavoidable. These may, however, be somewhat supressed by making an iteration along the lines described in Section 6.5.

Exercises

1. Show that a one-sided, finite difference scheme in time and space of the radiation condition (7.1) can be written

$$\phi_B^{n+1} = \begin{cases} \phi_B^n & ; \quad c_\phi > 0 \\ \left(1 + c_\phi \frac{\Delta t}{\Delta x}\right)\phi_B^n - c_\phi \frac{\Delta t}{\Delta x}\phi_{B+1}^n & ; \quad c_\phi \leq 0 \end{cases} \tag{7.66}$$

   The open boundary is to the left so that subscript B denotes the values of the variables on the open boundary while subscript $B+1$ indicates the values to the right of the open boundary.

2. Show by use of (7.66) that the radiation condition is "simply" an interpolation of values on the inside of the computational domain.

Chapter 8

GENERAL VERTICAL COORDINATES

Most modern models employed in the meteorological and oceanographic community replace the natural geopotential vertical coordinate (z-coordinate) with a new vertical coordinate. The reason for this is that the geopotential coordinate is quite cumbersome to work with in the presence of steep topography such as mountains in the atmosphere and shelf breaks and sea mountains in the ocean.

As early as in the late 1940s *Sutcliffe* (1947) and *Eliassen* (1949), at the dawn of numerical weather prediction, suggested to use pressure surfaces to replace surfaces of geopotential height as the vertical coordinate in atmospheric models, a method successfully tested by *Charney and Phillips* (1953) using a quasi-geostrophic model (cf. Section 1.6 on page 9). The pressure coordinate has several advantages over ordinary geopotential height models. For instance it reduces the mass conservation equation to a diagnostic equation, which in turn eases the analysis of the large scale (hydrostatic) motions.

The pressure coordinate, however, has certain computational disadvantages, in particular in the vicinity of mountains since the ground is not a pressure surface. To remedy this *Phillips* (1957) suggested to use terrain-following surfaces as the vertical coordinate. Such a coordinate system is now commonly referred to as the σ-coordinates, a coordinate system that has become quite popular in ocean models (*Blumberg and Mellor*, 1987; *Haidvogel et al.*, 2008).

Also other vertical coordinate systems are suggested. For instance it was early on suggested to use surfaces of potential temperature as the vertical coordinate. This was successfully tested by *Eliassen and Raustein* (1968, 1970) employing a simplified primitive equation models and by *Bleck* (1973) using the potential vorticiy equation, and was extended with success to full three-dimensional, dynamic-thermodynamic atmospheric models by *Shapiro* (1974). In the ocean so called isopycnic models in which surfaces of potential density are used as vertical coordinates was explored in primitive equation models by *Bleck and Smith* (1990). In the recent decade it has also become quite common to explore the use of so called hybrid coordinate models in which the vertical coordinate changes from one to another throughout the height both in the atmosphere and in the ocean (e.g., *Bleck*, 2002). Finally it should be emphasized that the various vertical coordinate systems all have their advantages and disadvantages (cf. *Griffies*, 2004, Chapter 6).

In the following we will first show how we in general transform equations formulated in geopotential coordinates to a new general vertical coordinate, say $s = s(x, y, z, t)$. To this end we follow the derivation made by *Kasahara* (1974). We then show how the governing equations of a hydrostatic, non-Boussinesq fluid (cf. Section 1.3 on page 4) is affected by such a transformation. We end this chapter by showing an explicit example using the σ-coordinates as our example.

8.1 Transformation to a general vertical coordinate

In general we transform from one coordinate system of independent variables, say (x, y, z, t), to another system, say (x', y', s, t'), by specifying how the independent variables in the transformed system depend on the independent variables of the original system. Here we will only replace the vertical height coordinate z. Accordingly we define the transformation simply by

$$x' = x, \quad y' = y, \quad z' = s = s(x, y, z, t), \quad \text{and} \quad t' = t. \tag{8.1}$$

Note that we have only replaced the normal geopotential height coordinate z with a general vertical coordinate s, while the horizontal coordinates are left unchanged. To ensure that the transformation is unique we must require that s is a monotone function of height z. Mathematically this means that the gradient of s with respect to z does not change sign within a fluid column, or

$$\partial_z s \gtrless 0, \quad \text{and } \partial_z s \neq 0. \tag{8.2}$$

This is also a necessary condition to ensure that the inverse transformation $z = z(x', y', s, t')$ exists as well.

From (8.1) we immediately get

$$\partial_z x' = \partial_z y' = \partial_z t' = 0, \quad \partial_t x' = \partial_t y' = 0, \quad \partial_y x' = \partial_x y' = 0, \quad \text{and} \quad \partial_x t' = \partial_y t' = 0, \tag{8.3}$$

while

$$\partial_x x' = \partial_y y' = \partial_t t' = 1. \tag{8.4}$$

Similarly follows

$$\partial_s x = \partial_s y = \partial_s t = 0, \quad \partial_{t'} x = \partial_{t'} y = 0, \quad \partial_{y'} x = \partial_{x'} y = 0, \quad \text{and} \quad \partial_{x'} t = \partial_{y'} t = 0, \tag{8.5}$$

while

$$\partial_{x'} x = \partial_{y'} y = \partial_{t'} t = 1. \tag{8.6}$$

We emphasize that s is monotonic with respect to z, which implies that $\partial_z s \neq 0$ and $\partial_s z \neq 0$. We also observe that if we transform z to z, that is, let $s = z$ then $\partial_z s = \partial_s z = 1$.

Let $\psi = \psi(x, y, z, t) = \psi(x', y', s, t')$ denote any scalar. Then the first property of the transformation is

$$\partial_z \psi = \partial_z s \partial_s \psi. \tag{8.7}$$

If we take the derivative of ψ with respect to one of the independent variables in the coordinate system we transform to, say t', then we get

$$\partial_{t'} \psi = \partial_t \psi \partial_{t'} t + \partial_x \psi \partial_{t'} x + \partial_y \psi \partial_{t'} y + \partial_z \psi \partial_{t'} z = \partial_t \psi + \partial_z s \partial_s \psi \partial_{t'} z, \tag{8.8}$$

where the last equal sign follows by utilizing (8.3) - (8.7). If we solve (8.8) with respect to $\partial_t \psi$ we further get

$$\partial_t \psi = \partial_{t'} \psi - \partial_z s \partial_s \psi \partial_{t'} z. \tag{8.9}$$

Similarly follows that

$$\partial_x \psi = \partial_{x'} \psi - \partial_z s \partial_s \psi \partial_{x'} z, \quad \text{and} \quad \partial_y \psi = \partial_{y'} \psi - \partial_z s \partial_s \psi \partial_{y'} z. \tag{8.10}$$

Let us define the horizontal gradient of ψ in the new coordinate system by

$$\nabla_s \psi = \mathbf{i} \partial_{x'} \psi + \mathbf{j} \partial_{y'} \psi. \tag{8.11}$$

Then making use of (8.9) and (8.10) we obtain

$$\nabla_H \psi = \nabla_s \psi - \partial_z s \partial_s \psi \nabla_s z. \tag{8.12}$$

Furthermore we find that the horizontal divergence of any vector, say $\mathbf{A}$, transforms as

$$\nabla_H \cdot \mathbf{A} = \nabla_s \cdot \mathbf{A} - \partial_z s \partial_s \mathbf{A} \cdot \nabla_s z. \tag{8.13}$$

We note that all vectors project onto the horizontal geopotential surface. This is also true for the gradient (8.12). Thus the metric term associated with the vertical gradient of the surface s in the geopotential coordinate system is eliminated.

We note that since the individual derivative[1] is independent of coordinate transformation we get

$$\frac{D\psi}{dt} = \partial_t \psi + \mathbf{u} \cdot \nabla_H \psi + w \partial_z \psi = \partial_{t'} \psi + \mathbf{u} \cdot \nabla_s \psi + \dot{s} \partial_s \psi, \tag{8.14}$$

where

$$\dot{s} = \frac{Ds}{dt} = \partial_t s + \mathbf{u} \cdot \nabla_H s + w \partial_z s \tag{8.15}$$

is the speed of the surface s in the direction of the three-dimensional velocity. Note that the first equality in (8.14) is the common expression of the individual deriviative in the geopotential coordinate system, while the second equality expresses the individual derivative in the transformed system or the new general vertical coordinate system. We now make use of (8.9) - (8.12) to replace the appropriate terms in the first equality in (8.14). Then we get

$$\frac{D\psi}{dt} = \partial_{t'} \psi + \mathbf{u} \cdot \nabla_s \psi + (w - \partial_{t'} z - \mathbf{u} \cdot \nabla_s z) \partial_z s \partial_s \psi. \tag{8.16}$$

Equating this by the individual derivative expressed in the new general vertical coordinate system as visulized in the second equality in (8.14) we get

$$\dot{s} = (w - \partial_{t'} z - \mathbf{u} \cdot \nabla_s z) \partial_z s \equiv \omega \partial_z s, \tag{8.17}$$

where the identity in (8.17) defines the velocity ω by

$$\omega = w - (\partial_{t'} z + \mathbf{u} \cdot \nabla_s z). \tag{8.18}$$

[1] Also by many authors referred to as the material derivative

As revealed by (8.18) the difference between ω and w is associated with the speed of the surface z in the transformed coordinate system. We observe that if $s = \rho$, that is, if s is a material surface then the kinematic boundary condition requires $w = \partial_{t'} z + \mathbf{u} \cdot \nabla_s z$, and hence that $\omega = 0$. This is to be expected since a material surface is a surface that consists of the same fluid particles for all times, that is, no particles are transported through the surface. Thus ω is interpreted as that part of the vertical movement of particles observed when moving with the surface s, or if we prefer the speed of the fluid particles through the surface s. This is corroborated by the fact that if we let $s = z$ then (8.18) gives $\omega = w$, in which case it equals the vertical velocity in the fixed geopotential coordinate system.

8.2 Transformation of the governing equations

To give insight into how the transformation is applied, we apply it to a non-Boussinesq, hydrostatic fluid.

The hydrostatic equation

We start by transforming the hydrostatic equation

$$\partial_z p + \rho g = 0. \tag{8.19}$$

Using the transformation formulas of the previous section we get

$$\partial_s p + \rho g \partial_s z = 0. \tag{8.20}$$

We may use this equation to determine the metric factors $\partial_s z$ and $\partial_z s$ as follows

$$\partial_s z = -\frac{\partial_s p}{\rho g}, \text{ and } \partial_z s = -\frac{\rho g}{\partial_s p}. \tag{8.21}$$

We note in passing that if $s = p$ then (8.20) reduces to

$$1 + \rho g \partial_p z = 0, \text{ or } \partial_p z = -\frac{1}{\rho g}. \tag{8.22}$$

Mass conservation

Next, we transform the continuity equation

$$\partial_t \rho + \nabla \cdot (\mathbf{v}\rho) = 0. \tag{8.23}$$

We first rewrite this equation to yield

$$\frac{1}{\rho}\frac{D\rho}{dt} + \nabla_H \cdot \mathbf{u} + \partial_z w = 0. \tag{8.24}$$

We then make use of the transformation formulas to obtain

$$\begin{aligned}\frac{1}{\rho}\frac{D\rho}{dt} + \nabla_H \cdot \mathbf{u} + \partial_z w = & - \rho\left(\partial_{t'}\alpha + \mathbf{u}\cdot\nabla_s\alpha + \dot{s}\partial_s\alpha\right) \\ & + \partial_z s[\partial_{t'}(\partial_s z) + \nabla_s\cdot(\mathbf{u}\partial_s z) + \partial_s(\dot{s}\partial_s z)], \end{aligned} \tag{8.25}$$

where $\alpha = 1/\rho$. To arrive at this result we have also solved (8.18) with respect to w to replace $\partial_s w$. We may further develop (8.25) by making use of (8.21) to replace the metric term $\partial_s z$. Thus we get

$$\frac{1}{\rho}\frac{D\rho}{dt} + \nabla_H \cdot \mathbf{u} + \partial_z w = (\partial_s p)^{-1}\left[\partial_{t'}(\partial_s p) + \nabla_s\cdot(\mathbf{u}\partial_s p) + \partial_s(\dot{s}\partial_s p)\right], \tag{8.26}$$

and hence the transformed continuity equation reads

$$\partial_{t'}(\partial_s p) + \nabla_s\cdot(\mathbf{u}\partial_s p) + \partial_s(\dot{s}\partial_s p) = 0. \tag{8.27}$$

We note that if $s = p$ then (8.27) reduced to

$$\nabla_p \cdot \mathbf{u} = \partial_p(\rho g \omega), \tag{8.28}$$

that is, a diagnostic equation.

Energy equation

If we apply a similar procedure to the tracer equation (1.13) we get

$$\partial_{t'}C + \mathbf{u}\cdot\nabla_s C + \dot{s}\partial_s C = \boldsymbol{\mathcal{F}}_C + S_C \tag{8.29}$$

where the right-hand side represents the transformed fluxes and source terms.

The momentum equation

We finally transform the horizontal component of the momentum equation for a non-Boussinesq, hydrostatic fluid (1.12) by first rewriting it to read

$$\frac{D\mathbf{u}}{dt} + f\mathbf{k}\times\mathbf{u} = -\alpha\nabla_H p + \alpha\partial_z\boldsymbol{\tau} + \nabla_H\cdot\boldsymbol{\mathcal{F}}_M^H \tag{8.30}$$

where $\boldsymbol{\tau}$ is the vertical mixing or flux vector, sometimes referred to as the the vertical shear stress. To transform this equation is a bit more complicated so we treat it term by term.

We first consider the pressure term, which is special. For a non-Boussinesq fluid we get

$$\alpha\nabla_H p = \alpha\nabla_s p + g\nabla_s z = \nabla_s M - p\nabla_s\alpha \tag{8.31}$$

where

$$M = \alpha p + gz \tag{8.32}$$

is the *Montgomery potential* (or stream function). In the case $s = \rho$ the last term in (8.32) vanishes since then $\nabla_s \alpha = 0$. Under these circumstances the Montgomery potential becomes a true potential and is a streamfunction for the geostrophic velocity. For any other choice of s, however, the last term in (8.32) must be retained. We finally note that the Montgomery potential appears because all vectors are projected onto the horizontal surface (with respect to gravity), even though all gradients are evaluated in the transformed x', y', s, t' system.

Next we consider the vertical shear stress term. In this we apply (8.7) and (8.21) to get

$$\alpha \partial_z \boldsymbol{\tau} = \alpha \partial_z s \partial_s \boldsymbol{\tau} = -g \frac{\partial_s \boldsymbol{\tau}}{\partial_s p} = \partial_p \boldsymbol{\tau}. \tag{8.33}$$

Recalling that

$$\frac{D\mathbf{u}}{dt} = \partial_{t'} \mathbf{u} + \mathbf{u} \cdot \nabla_s \mathbf{u} + \dot{s} \partial_s \mathbf{u} \tag{8.34}$$

and that

$$\mathbf{u} \cdot \nabla_s \mathbf{u} = \nabla_s \left(\frac{1}{2} \mathbf{u}^2 \right) + \zeta \mathbf{k} \times \mathbf{u} \tag{8.35}$$

where $\zeta = \mathbf{k} \cdot \nabla_s \times \mathbf{u}$ is the relative vorticity relative to the new coordinate system the momentum equation finally becomes

$$\partial_{t'} \mathbf{u} + \nabla_s \left(\frac{1}{2} \mathbf{u}^2 \right) + (\zeta + f) \mathbf{k} \times \mathbf{u} + \dot{s} \partial_s \mathbf{u} = -\nabla_s M + p \nabla_s \alpha - g \partial_p \boldsymbol{\tau} + \nabla_s \cdot \boldsymbol{\mathcal{F}}_M^H. \tag{8.36}$$

We may also write this equation in flux form. We then first recombine the second and third term on the left-hand side of (8.36) using (8.35). Next we multiply (8.36) by $\partial_s p$ and then finally make use of the continuity equation in the form (8.27). We then get

$$\begin{aligned} \partial_{t'} (\mathbf{u} \partial_s p) + \nabla_s \cdot (\mathbf{u}\mathbf{u} \partial_s p) \quad &+ \quad f \mathbf{k} \times \mathbf{u} \partial_s p + \partial_s (\dot{s} \mathbf{u} \partial_s p) \\ &= \quad -\partial_s p \left(\nabla_s M + p \nabla_s \alpha \right) - g \partial_s \boldsymbol{\tau} + \partial_s p \nabla_s \cdot \boldsymbol{\mathcal{F}}_M^H. \end{aligned} \tag{8.37}$$

If $s = p$ then (8.37) becomes

$$\begin{aligned} \partial_{t'} \mathbf{u} + \nabla_s \cdot (\mathbf{u}\mathbf{u}) \quad &+ \quad f \mathbf{k} \times \mathbf{u} + \partial_s (\dot{s} \mathbf{u}) \\ &= \quad -\nabla_s M + p \nabla_s \alpha - g \partial_s \boldsymbol{\tau} + \nabla_s \cdot \boldsymbol{\mathcal{F}}_M^H. \end{aligned} \tag{8.38}$$

As alluded to earlier when treating the diffusion problem, we emphasize at this point that the mixing term or "diffusion" term is mostly added to prevent our numerical model from blowing up. Hence its exact transformation is of secondary importance.

8.3 Terrain following coordinates

As an example we apply these transformation to transform the mass conservation equation to the so called σ-coordinate models. This particular coordinate system is defined by

$$s = \sigma = \frac{z - \eta}{D} \quad \text{or} \quad z = \sigma D + \eta, \tag{8.39}$$

where $D = H+\eta$ is the total depth, η being the deviation of the upper surface from its equilibrium position and H is the equilibrium depth of the fluid columns. The terrain following coordinate models are very popular in the oceanographic community, e.g., ROMS (*Haidvogel et al.*, 2008), and various versions of POM (*Blumberg and Mellor*, 1987; *Engedahl*, 1995a). It is also to some extent applied in numerical weather predictions models (*Phillips*, 1957; *Kasahara*, 1974).

First we note that the metric factor $\partial_z s$ and $\partial_s z$ using (8.21) becomes

$$\partial_z s = \partial_z \sigma = \frac{1}{D} \quad \text{and} \quad \partial_s z = \partial_\sigma z = D, \tag{8.40}$$

which allows us to rewrite the hydrostatic equation to

$$\partial_\sigma p = -\rho g D. \tag{8.41}$$

Furthermore we need to know the speed ω trough the σ surfaces. Applying (8.39) we get

$$\omega = w - \sigma \partial_{t'} D - \partial_{t'} \eta - \sigma \mathbf{u} \cdot \nabla_s D - \mathbf{u} \cdot \nabla_s \eta. \tag{8.42}$$

Thus the mass conservation equation in the form (8.27) becomes

$$\partial_{t'}(\rho D) + \nabla_\sigma \cdot (\rho D \mathbf{u}) + \partial_\sigma(\dot{\sigma} \rho D) = 0, \tag{8.43}$$

Separating the effect of the density we get

$$\frac{D\rho}{dt} + \frac{\rho}{D}\left[\partial_{t'} D + \nabla_\sigma \cdot (D\mathbf{u}) + \partial_\sigma(\dot{\sigma} D)\right] = 0, \tag{8.44}$$

We observe using (8.17) that $\dot{\sigma} = \omega \partial_z \sigma = \omega D^{-1}$. Furthermore we note that $\partial_{t'} D = \partial_{t'} \eta$. Substitution of these expressions into (8.44), and invoking the Boussinesq approximation (1.16) or $\frac{D\rho}{dt} = 0$, the continuity equation for a Boussinesq fluid in terrain-following coordinates is

$$\partial_{t'} \eta + \nabla_\sigma \cdot (D\mathbf{u}) + \partial_\sigma \omega = 0. \tag{8.45}$$

We note that the remaining equations may be derived from their general expressions in a similar fashion. For instance using (8.41) the momentum equation in the flux form (8.37) becomes

$$\begin{aligned} &\partial_{t'}(D\mathbf{u}) + \nabla_s \cdot (D\mathbf{u}\mathbf{u}) + f\mathbf{k} \times D\mathbf{u} + \partial_\sigma(\omega \mathbf{u}) \\ &\qquad = -D\left(\nabla_s M + p \nabla_s \alpha\right) - \frac{1}{\rho_0}\partial_\sigma \boldsymbol{\tau} + D\nabla_s \cdot \boldsymbol{\mathcal{F}}^H_M. \end{aligned} \tag{8.46}$$

Chapter 9

TWO-DIMENSIONAL PROBLEMS

Below we investigate the effect of including more than one-dimension in space. In particular we study its impact on the numerical stability criterion. Further expansion into three dimensions is then straightforward.

9.1 Diffusion equation

We start by expanding the diffusion equation to two dimensions in space. Thus we consider the continuous equation

$$\partial_t \theta = \kappa(\partial_x^2 \theta + \partial_y^2 \theta). \tag{9.1}$$

Note that since we expand to two dimensions we use the notation $\theta(x_j, y_k, t^n) = \theta_{jk}^n$ as outlined in Section 2.9 page 24. As in the one-dimensional case (cf. Chapter 4) we employ the forward in time, centered in space (FTCS) scheme. Thus we replace the two second order derivatives with

$$\left[\partial_x^2 \theta\right]_{jk}^n = \frac{\theta_{j+1k}^n - 2\theta_{jk}^n + \theta_{j-1k}^n}{\Delta x^2}, \quad \text{and} \quad \left[\partial_y^2 \theta\right]_{jk}^n = \frac{\theta_{jk+1}^n - 2\theta_{jk}^n + \theta_{jk-1}^n}{\Delta y^2}, \tag{9.2}$$

while we replace the first order derivative with respect to to time with

$$\left[\partial_t \theta\right]_{jk}^n = \frac{\theta_{jk}^{n+1} - \theta_{jk}^n}{\Delta t}, \tag{9.3}$$

where $\Delta x, \Delta y$ are the space increments along the x, y axis, respectively, and Δt is the time step. Thus we arrive at the scheme

$$\theta_{jk}^{n+1} = \theta_{jk}^n + \kappa \frac{\Delta t}{\Delta x^2}\left(\theta_{j-1k}^n - 2\theta_{jk}^n + \theta_{j+1k}^n\right) + \kappa \frac{\Delta t}{\Delta y^2}\left(\theta_{jk-1}^n - 2\theta_{jk}^n + \theta_{jk+1}^n\right), \tag{9.4}$$

To investigate the numerical stability of the scheme (9.4) we use as before von Neumann's method. As in the one-dimensional case we first substitute θ_{jk}^n by its individual Fourier components. Since we now must allow for waves propagating in a random direction in the x, y-space, the Fourier component must include waves propagating in the x direction as well as in the y direction, that is,

$$\theta_{jk}^{n} = \Theta_n e^{\alpha j \Delta x} e^{\beta k \Delta y}, \tag{9.5}$$

where α and β are wavenumbers in the x and y directions, respectively. Next we insert the discrete Fourier component into (9.4) and solve for the growth factor. We then get

$$G = 1 - 2\kappa \frac{\Delta t}{\Delta x^2}\left(1 - \cos \alpha \Delta x\right) - 2\kappa \frac{\Delta t}{\Delta y^2}\left(1 - \cos \beta \Delta y\right). \tag{9.6}$$

We observe that this expression is comparable to the one derived for the one-dimensional case, that is (4.33) on page 45, except that we have an additional term due to the two-dimensionality of (9.1). Applying von Neumann's criterion for numerical stability (4.30) we require $|G| \leq 1$. Hence

$$-1 \leq 1 - 2\kappa \frac{\Delta t}{\Delta x^2}\left(1 - \cos \alpha \Delta x\right) - 2\kappa \frac{\Delta t}{\Delta y^2}\left(1 - \cos \beta \Delta y\right) \leq 1. \tag{9.7}$$

As in the one-dimensional case we note that the right-hand inequality is trivially satisfied. To satisfy the left-hand inequality we require

$$\kappa \frac{\Delta t}{\Delta x^2}\left(1 - \cos \alpha \Delta x\right) + \kappa \frac{\Delta t}{\Delta y^2}\left(1 - \cos \beta \Delta y\right) \leq 1. \tag{9.8}$$

Since the left-hand side of (9.8) is maximum when $\cos \alpha \Delta x = \cos \beta \Delta y = -1$ we note that if

$$\kappa \frac{\Delta t}{\Delta x^2} + \kappa \frac{\Delta t}{\Delta y^2} \leq \frac{1}{2} \tag{9.9}$$

then the stability criterion is satisfied for all possible values of the wavenumbers α and β. Thus the time step limit is

$$\Delta t \leq \frac{1}{2\kappa} \frac{\Delta x^2 \Delta y^2}{\Delta x^2 + \Delta y^2}. \tag{9.10}$$

In the special case when the grid is regular (a square grid), that is, $\Delta x = \Delta y = \Delta s$, we get

$$\Delta t \leq \frac{\Delta s^2}{4\kappa}. \tag{9.11}$$

If we compare (9.11) with the one dimensional case (eq. 4.36 on page 46) we notice that the allowed time step is reduced by a factor of two. Thus the inclusion of more than one dimension implies that the criterion for numerical stability becomes more stringent. In fact, as we will acknowledge as we proceed, this is a general result which applies to all problems.

9.2 Advection equation

The two-dimensional version of the advection equation is

$$\partial_t \theta + u_0 \partial_x \theta + v_0 \partial_y \theta = 0, \tag{9.12}$$

where u_0, v_0 are the velocity components in the x, y direction respectively (here treated as constants, hence the subscript 0). To solve (9.12) numerically let us employ the well known second order accurate CTCS (leapfrog) scheme that worked well for the one-dimensional case. Thus we replace the forst order derivatives with their respective finite difference approximations as outlined in Section 2.7. Thus we get

$$\frac{\theta_{jk}^{n+1} - \theta_{jk}^{n-1}}{2\Delta t} + u_0 \frac{\theta_{j+1k}^{n} - \theta_{j-1k}^{n}}{2\Delta x} + v_0 \frac{\theta_{jk+1}^{n} - \theta_{jk-1}^{n}}{2\Delta y} = 0. \tag{9.13}$$

or

$$\theta_{jk}^{n+1} = \theta_{jk}^{n-1} + u_0 \frac{\Delta t}{\Delta x}\left(\theta_{j+1k}^{n} - \theta_{j-1k}^{n}\right) + v_0 \frac{\Delta t}{\Delta y}\left(\theta_{jk+1}^{n} - \theta_{jk-1}^{n}\right). \tag{9.14}$$

To investigate the numerical stability we again employ von Neumann's method. Hence we insert the discrete Fourier component (9.5) into 9.14. We then get

$$\Theta_{n+1} = \Theta_{n-1} - 2i\Theta_n \left(u_0 \frac{\Delta t}{\Delta x} \sin \alpha\Delta x + v_0 \frac{\Delta t}{\Delta y} \sin \beta\Delta y\right) \tag{9.15}$$

Thus the equation for the growth factor is as in the one dimensional case, that is,

$$G^2 + 2i\lambda G - 1 = 0, \tag{9.16}$$

where

$$\lambda = u_0 \frac{\Delta t}{\Delta x} \sin \alpha\Delta x + v_0 \frac{\Delta t}{\Delta y} \sin \beta\Delta y. \tag{9.17}$$

Thus, as before, we get the two solutions

$$G_{1,2} = i\lambda \pm \sqrt{1 - \lambda^2}, \tag{9.18}$$

and hence that $|G_{1,2}| = 1$ resulting in a neutral stable scheme (no energy dissipation). The only difference from the one-dimensional problem is λ. As we have done earlier we require that the radical is a positive definite quantity and hence that

$$|u_0 \frac{\Delta t}{\Delta x} \sin \alpha\Delta x + v_0 \frac{\Delta t}{\Delta y} \sin \beta\Delta y| \leq 1. \tag{9.19}$$

For this to be valid for all possible choice of wavenumbers α and β we must require

$$|u_0| \frac{\Delta t}{\Delta x} + |v_0| \frac{\Delta t}{\Delta y} \leq 1 \quad \text{or} \quad \Delta t \leq \frac{\Delta x \Delta y}{|u_0|\Delta y + |v_0|\Delta x}. \tag{9.20}$$

If we let the grid be regular, that is, let $\Delta x = \Delta y = \Delta s$ and $u_0 = v_0 = c_0$ we get

$$\Delta t \leq \frac{\Delta s}{2|c_0|}. \tag{9.21}$$

Thus we observe, as in the diffusion problem, that increasing the dimension from one to two leads to a more stringent stability condition. This is to be expected since the physical interpretation of the CFL condition says that the characteristic must be within the cone of influence from time level n.

9.3 Shallow water equations

Analytic solutions

As we did for the one-dimensional case it is worthwhile to first analyze the various wave motions supported by the two-dimensional problem. The two dimensional, linear rotating shallow water equation are given by (6.14) and (6.15). For convenience we repeat them her in scalar form. Thus in two dimensions we get

$$\begin{aligned} \partial_t u + \bar{u}\partial_x u + \bar{v}\partial_y u - fv + \partial_x \phi &= 0, && (9.22)\\ \partial_t v + \bar{u}\partial_x v + \bar{v}\partial_y v + fu + \partial_y \phi &= 0, && (9.23)\\ \partial_t \phi + \bar{\phi}(\partial_x u + \partial_y v) &= 0. && (9.24) \end{aligned}$$

To analyze the possible wave motions we assume that all the variables are two dimensional waves of frequency ω. Thus we assume that the solution is

$$\mathbf{x} = \mathbf{x}_0 e^{-i\omega t} e^{i(\alpha x + \beta y)}, \tag{9.25}$$

where α and β are wave numbers in the x- and y-direction, respectively. We note that the frequency an the two wave numbers are all assumed to be real quantities. The dependent variable are contained in the vector $\mathbf{x}$, that is,

$$\mathbf{x} = \begin{bmatrix} u \\ v \\ \phi \end{bmatrix} \quad \text{and} \quad \mathbf{x}_0 = \begin{bmatrix} u_0 \\ v_0 \\ \phi_0 \end{bmatrix}, \tag{9.26}$$

where $\mathbf{x}_0$ is the amplitude.

Inserting (9.25) into the linearized equations (9.22) - (9.24) we get

$$\begin{aligned} (\alpha\bar{u} + \beta\bar{v} - \omega)u_0 - ifv_0 + \alpha\phi_0 &= 0, && (9.27)\\ -ifu_0 + i(\alpha\bar{u} + \beta\bar{v} - \omega)v_0 - \beta\phi_0 &= 0, && (9.28)\\ \bar{\phi}\alpha u_0 + \bar{\phi}\beta v_0 + (\alpha\bar{u} + \beta\bar{v} - \omega)\phi_0 &= 0, && (9.29) \end{aligned}$$

which in turn may be formulated as the homogeneous linear equation,

$$\boldsymbol{\mathcal{A}} \cdot \mathbf{x} = 0, \tag{9.30}$$

where the tensor $\boldsymbol{\mathcal{A}}$ is

$$\boldsymbol{\mathcal{A}} = \begin{bmatrix} (\alpha\bar{u} + \beta\bar{v} - \omega) & -f & i\alpha \\ f & i(\alpha\bar{u} + \beta\bar{v} - \omega) & i\beta \\ i\alpha\bar{\phi} & i\beta\bar{\phi} & i(\alpha\bar{u} + \beta\bar{v} - \omega) \end{bmatrix}. \tag{9.31}$$

For non-trivial solutions to exists, the determinant of the tensor $\boldsymbol{\mathcal{A}}$ must be zero, which gives

$$(\alpha\bar{u} + \beta\bar{v} - \omega)\left[(\alpha\bar{u} + \beta\bar{v} - \omega)^2 - \bar{\phi}(\alpha^2 + \beta^2) - f^2\right] = 0. \tag{9.32}$$

As in the one-dimensional case we find that we get three solutions for the frequency ω, namely

$$\begin{aligned}
\omega_1 &= \bar{u}\alpha + \bar{v}\beta, && (9.33)\\
\omega_2 &= \bar{u}\alpha + \bar{v}\beta + \sqrt{c_0^2(\alpha^2+\beta^2)+f^2}, && (9.34)\\
\omega_3 &= \bar{u}\alpha + \bar{v}\beta - \sqrt{c_0^2(\alpha^2+\beta^2)+f^2}, && (9.35)
\end{aligned}$$

where

$$c_0 = \sqrt{\bar{\phi}} = \sqrt{gH} \tag{9.36}$$

is the wave speed of gravity waves. The first solution is simply the geostrophic balance as displayed in (1.40) on page 9 with $\phi = gh$, that is,

$$\mathbf{u} = \frac{1}{f}\mathbf{k} \times \nabla_H \phi. \tag{9.37}$$

We easily derive this interpretation by substituting ω_1 from (9.33) into (9.27) and (9.28), respectively, that is,

$$\begin{aligned}
0 - fv + i\alpha\phi &= 0, && (9.38)\\
fu + 0 + i\alpha\phi &= 0, && (9.39)
\end{aligned}$$

which gives

$$u = -\frac{1}{f}i\beta\phi \quad \text{and} \quad v = \frac{1}{f}i\alpha\phi \quad \Rightarrow \quad u = -\frac{1}{f}\partial_y\phi \quad \text{and} \quad v = \frac{1}{f}\partial_x\phi. \tag{9.40}$$

The last implication follows by using the Fourier solution backwards and shows that the geostrophic balance (9.37) is recovered.

The two other solutions represented by $\pm\sqrt{\bar{\phi}(\alpha^2+\beta^2)+f^2}$ are combined inertia and gravity waves. The inertia part is associated with frequencies ω proportional to f, so called inertial oscillation, while gravity waves are associated with frequencies $\pm c_0\sqrt{(\alpha^2+\beta^2)}$.

Note that to construct the analytic solution to (9.22) - (9.24) for any given initial and boundary conditions we just expand the solution into a two-dimensional Fourier series, that is,

$$\mathbf{x} = \sum_{\alpha=-\infty}^{\infty}\sum_{\beta=-\infty}^{\infty} \mathbf{x}_0(\alpha,\beta)e^{i(\alpha x+\beta y-\omega t)}, \tag{9.41}$$

where we observe that $\mathbf{x}_0(\alpha,\beta)$ contains the amplitudes of each Fourier component at the initial time.

Finite difference equation

To solve (9.22) - (9.24) by numerical means we employ the CTCS (leapfrog) scheme. Hence

$$\begin{aligned}
\frac{u_{jk}^{n+1}-u_{jk}^{n-1}}{2\Delta t}+\bar{u}\frac{u_{j+1k}^{n}-u_{j-1k}^{n}}{2\Delta x}+\bar{v}\frac{u_{jk-1}^{n}-u_{jk-1}^{n}}{2\Delta y}-fv_{jk}^{n} &= -\frac{\phi_{j+1k}^{n}-\phi_{j-1k}^{n}}{2\Delta x}, && (9.42)\\
\frac{v_{jk}^{n+1}-v_{jk}^{n-1}}{2\Delta t}+\bar{u}\frac{v_{j+1k}^{n}-v_{j-1k}^{n}}{2\Delta x}+\bar{v}\frac{v_{jk-1}^{n}-v_{jk-1}^{n}}{2\Delta y}+fu_{jk}^{n} &= -\frac{\phi_{jk+1}^{n}-\phi_{jk-1}^{n}}{2\Delta y}, && (9.43)\\
\frac{\phi_{jk}^{n+1}-\phi_{jk}^{n-1}}{2\Delta t}+c_0^2\frac{u_{j+1k}^{n}-u_{j-1k}^{n}}{2\Delta x}+c_0^2\frac{v_{jk+1}^{n}-v_{jk-1}^{n}}{2\Delta y} &= 0. && (9.44)
\end{aligned}$$

To investigate the numerical stability we are only interested in that part of the solution that contain the waves. Thus we neglect all other momentum and volume sources, as well as any steady state solution upon which the waves may ride. Accordingly we let $\bar{u}=\bar{v}=0$ in which case (9.42) - (9.44) reduces to

$$\begin{aligned}
u_{jk}^{n+1}-u_{jk}^{n-1} &= 2\Delta tfv_{jk}^{n}-\frac{\Delta t}{\Delta x}\left(\phi_{j+1k}^{n}-\phi_{j-1k}^{n}\right), && (9.45)\\
v_{jk}^{n+1}-v_{jk}^{n-1} &= -2\Delta tfu_{jk}^{n}-\frac{\Delta t}{\Delta y}\left(\phi_{jk+1}^{n}-\phi_{jk-1}^{n}\right), && (9.46)\\
\phi_{jk}^{n+1}-\phi_{jk}^{n-1} &= -c_0^2\frac{\Delta t}{\Delta x}\left(u_{j+1k}^{n}-u_{j-1k}^{n}\right)-c_0^2\frac{\Delta t}{\Delta y}\left(v_{jk+1}^{n}-v_{jk-1}^{n}\right). && (9.47)
\end{aligned}$$

The question then arise whether the stability condition changes compared to the one-dimensional case, and if so whether it is more stringent or relaxed. To investigate this we use von Neumann's method. Thus we start by inserting a discrete Fourier component into (9.45) - (9.47). In this case the discrete Fourier component reads

$$\mathbf{x}_{jk}^{n}=\mathbf{X}_n e^{i(\alpha j\Delta x+\beta k\Delta y)} \qquad (9.48)$$

where the transpose of the vector $\mathbf{x}_{jk}^{n}$ is ${\mathbf{x}_{jk}^{n}}^{T}=\left[u_{jk}^{n},v_{jk}^{n},\phi_{jk}^{n}\right]$ and the transpose of the vector $\mathbf{X}_n$ is ${\mathbf{X}_n}^{T}=[U_n,V_n,\Phi_n]$. Insertion into (9.45) - (9.47) then gives

$$\begin{aligned}
U_{n+1}-U_{n-1} &= 2f\Delta tV_n-2i\Phi_n\frac{\Delta t}{\Delta x}\sin\alpha\Delta x, && (9.49)\\
V_{n+1}-V_{n-1} &= -2f\Delta tU_n-2i\Phi_n\frac{\Delta t}{\Delta y}\sin\beta\Delta y, && (9.50)\\
\Phi_{n+1}-\Phi_{n-1} &= -2ic_0^2U_n\frac{\Delta t}{\Delta x}\sin\alpha\Delta x-2ic_0^2V_n\frac{\Delta t}{\Delta y}\sin\beta\Delta y. && (9.51)
\end{aligned}$$

To find an equation for the growth factor we eliminate V_n and U_n. We do this by first replacing n by $n+1$ in (9.49) and (9.50) followed by a replacement of n by $n-1$. By subtracting the results we get

$$U_{n+2}-2U_n+U_{n+2}=2f\Delta t(V_{n+1}-V_{n-1})-2i(\Phi_{n+1}-\Phi_{n-1})\frac{\Delta t}{\Delta x}\sin\alpha\Delta x, \qquad (9.52)$$

$$V_{n+2} - 2V_n + V_{n+2} = -2f\Delta t(U_{n+1} - U_{n-1}) - 2i(\Phi_{n+1} - \Phi_{n-1})\frac{\Delta t}{\Delta y}\sin\beta\Delta y. \tag{9.53}$$

Substituting for $V_{n+1} - V_{n-1}$ from (9.50) in (9.52), and $U_{n+1} - U_{n-1}$ from (9.49) in (9.53) we get

$$\begin{aligned} U_{n+2} - 2(1 - 2f^2\Delta t^2)U_n + U_{n+2} = -4if\Phi_n\frac{\Delta t^2}{\Delta y}\sin\beta\Delta y \\ - 2i(\Phi_{n+1} - \Phi_{n-1})\frac{\Delta t}{\Delta x}\sin\alpha\Delta x, \end{aligned} \tag{9.54}$$

$$\begin{aligned} V_{n+2} - 2(1 - 2f^2\Delta t^2)V_n + V_{n+2} = 4if\Phi_n\frac{\Delta t^2}{\Delta x}\sin\alpha\Delta x \\ - 2i(\Phi_{n+1} - \Phi_{n-1})\frac{\Delta t}{\Delta y}\sin\beta\Delta y. \end{aligned} \tag{9.55}$$

We are now in a position to eliminate U_n and V_n from (9.51). To this end we first replace n by $n+2$ in (9.51), and then replace n by $n-2$. Adding the results and subtracting (9.51) multiplied by $2(1 - 2f^2\Delta t^2)$ we get

$$\begin{aligned} \Phi_{n+3} - \Phi_{n+1} - 2(1 - 2f^2\Delta t^2)(\Phi_{n+1} - \Phi_{n-1}) + \Phi_{n-1} - \Phi_{n-3} = \\ - 2ic_0^2\left[U_{n+2} - 2(1 - 2f^2\Delta t^2)U_n + U_{n-2}\right]\frac{\Delta t}{\Delta x}\sin\alpha\Delta x \\ - 2ic_0^2\left[V_{n+2} - 2(1 - 2f^2\Delta t^2)V_n + V_{n-2}\right]\frac{\Delta t}{\Delta y}\sin\beta\Delta y. \end{aligned} \tag{9.56}$$

Thus substituting from (9.54) and (9.55) we finally get

$$\Phi_{n+3} - (1 + 2\lambda)\Phi_{n+1} + (1 + 2\lambda)\Phi_{n-1} - \Phi_{n-3} = 0 \tag{9.57}$$

where

$$\lambda = 1 - 2f^2\Delta t^2 - 2\left(c_0\frac{\Delta t}{\Delta x}\right)^2\sin^2\alpha\Delta x - 2\left(c_0\frac{\Delta t}{\Delta y}\right)^2\sin^2\beta\Delta y \tag{9.58}$$

Defining the growth factor by $\Phi_{n+2} = G\Phi_n$ we get a third order equation for the growth factor, that is,

$$G^3 - (1 + 2\lambda)G^2 + (1 + 2\lambda)G - 1 = 0. \tag{9.59}$$

We observe (9.59) may be written

$$(G - 1)(G^2 - 2\lambda G + 1) = 0. \tag{9.60}$$

Hence the three solutions are

$$G_1 = 1 \quad \text{and} \quad G_{2,3} = \lambda \pm i\sqrt{1 - \lambda^2}. \tag{9.61}$$

As so many times before we have to require that the radical is real in which case $|G_{1,2,3}| = 1$ and the scheme is neutrally stable (energy conserving). The condition for this to be true is hence that

$$|\lambda| \leq 1 \tag{9.62}$$

or

$$-1 \leq 1 - 2f^2\Delta t^2 - 2\left(c_0\frac{\Delta t}{\Delta x}\right)^2 \sin^2 \alpha\Delta x - 2\left(c_0\frac{\Delta t}{\Delta y}\right)^2 \sin^2 \beta\Delta y \leq 1. \tag{9.63}$$

The right-hand inequality is no problem, but the left-hand inequality requires

$$\left(c_0\frac{\Delta t}{\Delta x}\right)^2 \sin^2 \alpha\Delta x + \left(c_0\frac{\Delta t}{\Delta y}\right)^2 \sin^2 \beta\Delta y \leq 1 - f^2\Delta t^2 \tag{9.64}$$

and hence that

$$\Delta t \leq \frac{\Delta x}{c_0\sqrt{\sin^2 \alpha\Delta x + \left(\frac{\Delta x}{\Delta y}\right)^2 \sin^2 \beta\Delta y + \left(\frac{\Delta x}{L_R}\right)^2}}, \tag{9.65}$$

where $L_R = c_0/f$ is Rossby's deformation readius. For the inequality to hold for all possible wave numbers α and β we must require

$$\Delta t \leq \frac{\Delta x}{c_0\sqrt{1 + \left(\frac{\Delta x}{\Delta y}\right)^2 + \left(\frac{\Delta x}{L_R}\right)^2}}. \tag{9.66}$$

If we in addition require that the Rossby radius of deformation is well resolved in the grid, as is the case in most applications, then $\left(\frac{\Delta x}{L_R}\right) \ll 1$ and the last term in the radical may be neglected. For all practical purposes the stability condition then is

$$\Delta t < \frac{\Delta x}{c_0\sqrt{1 + \left(\frac{\Delta x}{\Delta y}\right)^2}}. \tag{9.67}$$

If we let $\Delta x = \Delta y = \Delta s$ then we get

$$\Delta t < \frac{\Delta s}{c_0\sqrt{2}}. \tag{9.68}$$

Comparing (9.68) with the similar conditions for the one-dimensional case, as displayed in (6.52), we observe that the condition becomes more stringent so that we have to apply a somewhat smaller time step. Since we arrived at the same conclusion regarding the diffusion and the advection problem this result appears to be general, and indeed it is.

Chapter 10

ADVANCED TOPICS

The purpose is to indicate how to expand our knowledge acquired through the previous chapters. For one how do we construct schemes of higher order accuracy, and does it change the properties of the schemes? Secondly, how do we solve problems when advection and diffusion are equally important? Furthermore, what about non-linearities. How do we treat them numerically, and do they harbour implications regarding the instability? Finally we also provide an introduction to smoothing and filtering and the spectral method.

10.1 Higher order advection schemes

As alluded to in Section 2.6 we may construct schemes with higher order accuracy using Taylor series expansion.

A fourth order in space CTCS scheme

As an example let us consider how to construct a fourth order in space accurate scheme for the advection equation

$$\partial_t\theta + u\partial_x\theta = 0. \tag{10.1}$$

First we recall that we use Taylor expansions to derive finite difference approximations (FDAs) to the derivatives in (10.1). From Section 2.6 and in particular (2.27) and (2.29) (cf. page 19) we note that

$$\theta^n_{j\pm1} = \theta^n_j \pm \partial_x\theta^n_j\Delta x + \frac{1}{2}\partial^2_x\theta^n_j\Delta x^2 \pm \frac{1}{6}\partial^3_x\theta^n_j\Delta x^3 + \mathcal{O}(\Delta x^4). \tag{10.2}$$

Subtracting and solving with respect to $\partial_x\theta^n_j$ we get

$$\partial_x\theta^n_j = \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} - \frac{1}{6}\partial^3_x\theta^n_j\Delta x^2 + \mathcal{O}(\Delta x^4), \tag{10.3}$$

Truncating (10.3) by neglecting terms of order $\mathcal{O}(\Delta x^2)$ and higher we get

$$[\partial_x\theta]^n_j = \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x}, \tag{10.4}$$

which then is a viable FDA of the second term in (10.1). Note that in (10.3) only the points adjacent to the point x_j, that is, the points $\pm\Delta x$ away from x_j are used. Suppose now that we use points located $\pm 2\Delta x$ away instead. Then using Taylor series we get

$$\theta^n_{j\pm2} = \theta^n_j \pm 2\partial_x\theta^n_j\Delta x + 2\partial^2_x\theta^n_j\Delta x^2 \pm \frac{4}{3}\partial^3_x\theta^n_j\Delta x^3 + \mathcal{O}(\Delta x^4), \tag{10.5}$$

which is the same as (10.3) only that Δx is replaced by $2\Delta x$. Hence subtracting and solving with respect to $\partial_x\theta^n_j$ we get

$$\partial_x\theta^n_j = \frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x} - \frac{2}{3}\partial^3_x\theta|\Delta x^2 + \mathcal{O}(\Delta x^4). \tag{10.6}$$

Truncating (10.6) by neglecting all terms on the right-hand side of order $\mathcal{O}(\Delta x^2)$ and higher we get an equally valid expansion from which a centered, second order FDA may be constructed. In the limit $\Delta x \to 0$ they both tend to $\partial_x\theta$, that is, they are both numerically consistent FDAs. To arrive at an higher order FDA we now combine (10.3) and (10.6) linearly, while retaining terms of order $\mathcal{O}(\Delta x^2)$. To this end we first multiply (10.3) by an as yet unknown coefficient a and (10.6) by an unknown coefficient b. Adding the results we get

$$a\frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} + b\frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x} = (a+b)\partial_x\theta^n_j + \frac{1}{6}(a+4b)\partial^3_x\theta^n_j\Delta x^2 + \mathcal{O}(\Delta x^4). \tag{10.7}$$

The unknown coefficients a and b are linear weights yet to be found. Solving (10.7) with respect to $(a+b)\partial_x\theta^n_j$ we get

$$(a+b)\partial_x\theta^n_j = a\frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} + b\frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x} - \frac{1}{6}(a+4b)\partial^3_x\theta^n_j\Delta x^2 + \mathcal{O}(\Delta x^4) \tag{10.8}$$

We observe that by requiring $a + 4b = 0$ the second order term actually vanishes. Furthermore if we in addition require $a + b = 1$, which gives $a = \frac{4}{3}$ and $b = -\frac{1}{3}$, we finally get

$$\partial_x\theta^n_j = \frac{4}{3}\frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} - \frac{1}{3}\frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x} + \mathcal{O}(\Delta x^4). \tag{10.9}$$

Thus (10.9) constitutes a fourth order accurate FDA of the first second term on the left-hand side of (10.1). Consequently a centered fourth order in space and second order in time scheme for the advection equation (10.1) is

$$\frac{\theta^{n+1}_j - \theta^{n-1}_j}{2\Delta t} + u\left\{\frac{4}{3}\frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} - \frac{1}{3}\frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x}\right\} = 0. \tag{10.10}$$

or

$$\theta^{n+1}_j = \theta^{n-1}_j - \frac{4}{3}u\frac{\Delta t}{\Delta x}\left[\theta^n_{j+1} - \theta^n_{j-1} - \frac{1}{8}(\theta^n_{j+2} - \theta^n_{j-2})\right]. \tag{10.11}$$

Since the fourth order scheme (10.11) is based on Taylor series we know that it is consistent. We also suspect it to be conditionally stable, but what about the condition? Will it be more restrictive or more tolerant? To consider the stability of (10.11) we use von Neumann's method. Thus letting $\theta_j^n = \Theta_n e^{\alpha j \Delta x}$ we get

$$\theta_{j+1}^n - \theta_{j-1}^n = 2i\Theta_n e^{\alpha j \Delta x} \sin \alpha \Delta x \tag{10.12}$$

and

$$\theta_{j+2}^n - \theta_{j-2}^n = 2i\Theta_n e^{\alpha j \Delta x} \sin 2\alpha \Delta x = 4i\Theta_n e^{\alpha j \Delta x} \sin \alpha \Delta x \cos \alpha \Delta x. \tag{10.13}$$

Thus defining the growth factor as $G = \Theta_{n+1}/\Theta_n$ we get

$$G^2 + 2i\lambda G - 1 = 0 \quad ; \quad \lambda = \frac{u\Delta t}{3\Delta x} \sin \alpha \Delta x (4 - \cos \alpha \Delta x), \tag{10.14}$$

and hence that

$$G_{1,2} = -i\lambda \pm \sqrt{1 - \lambda^2}. \tag{10.15}$$

As expected the CTCS schemes for the fourth order scheme returns a growth factor whose absolute value equals one. As expected the fourth order scheme is therefore neutrally stable under the condition that the radical in (10.15) is a positive definite quantity, that is,

$$\frac{1}{3} C |\sin \alpha \Delta x| (4 - \cos \alpha \Delta x) \leq 1 \tag{10.16}$$

where $C = |u|\Delta t/\Delta x$ as before is the Courant number. To ensure that the inequality is valid for all possible wave numbers we note that the maximum value of $(4 - \cos \alpha \Delta x)$ is five and that the maximum value of $|\sin \alpha \Delta x|$ is one. Thus we get that if

$$C \leq \frac{3}{5} \quad \text{or} \quad \Delta t \leq \frac{3\Delta x}{5|u|}, \tag{10.17}$$

which is indeed more stringent than the condition $C \leq 1$, or $\Delta t \leq \Delta x/|u|$, that we obtained for the CTCS leapfrog scheme.

Recalling that the CTCS scheme is dispersive we may study whether employing a higher order scheme has an effect on the numerical dispersion. To this end we follow the procedure given in Section 5.5 on page 67. Thus we start by decomposing θ_j^n into its Fourier components in time and space,

$$\theta_j^n = \Theta_0 e^{i\alpha(j\Delta x - cn\Delta t)}. \tag{10.18}$$

We then substitute (10.18) into (10.10) and solve with respect to the phase speed c to get

$$c = \frac{1}{\alpha \Delta t} \arcsin \left\{ u\alpha\Delta t \left[\frac{4}{3} \left(\frac{\sin \alpha \Delta x}{\alpha \Delta x} \right) - \frac{1}{3} \left(\frac{\sin 2\alpha \Delta x}{2\alpha \Delta x} \right) \right] \right\}. \tag{10.19}$$

To leading order in $\alpha \Delta x$ we then obtain[1]

$$c \approx u \left\{ 1 - \frac{4}{5!} (\alpha \Delta x)^4 + \cdots \right\}. \tag{10.20}$$

[1]Note that $\sin z/z = 1 - z^2/6 + \cdots$ while $\arcsin z = 1 + z^2/6 + \cdots$ for $|z| < 1$.

which may be compare with the second order CTCS scheme as displayed in Section 5.5 on page 67). To leading order the latter is

$$c_{2nd} \approx u \left\{ 1 - \frac{1}{3!}(\alpha\Delta x)^2 + \cdots \right\}, \tag{10.21}$$

and thus the fourth order scheme is actually less dispersive as long as $0 \leq \alpha\Delta x \leq \pi$. To conclude the fourth order scheme is superior to the second order scheme both with regard to accuracy and dispersivity.

Finally we recall that the CTCS scheme contained an unphysical mode. To investigate how this is affected using a higher order scheme we note that the growth factor as given by (10.15) has exactly the same two solutions as in (5.22) on page 66, only that the expression for λ has changed to that listed in (10.14). Following the procedure outlined in Section 5.7 we in fact get exactly the same result, that is,

$$G_1 = e^{-i\alpha c\Delta t} \quad \text{and} \quad G_2 = (-1)e^{i\alpha c\Delta t} \tag{10.22}$$

where $c = c(\alpha)$ now is the dispersive phase speed defined in (10.19). Thus even for well resolved wave lengths the application of higher order schemes has no effect on the unphysical mode.

This process of constructing higher order finite difference approximations may be continued. For example we note that the scheme

$$\frac{\theta_j^{n+1} - \theta_j^{n-1}}{2\Delta t} + u \left\{ \frac{3}{2} \frac{\theta_{j+1}^n - \theta_{j-1}^n}{2\Delta x} - \frac{3}{5} \frac{\theta_{j+2}^n - \theta_{j-2}^n}{4\Delta x} + \frac{1}{10} \frac{\theta_{j+3}^n - \theta_{j-3}^n}{4\Delta x} \right\} = 0. \tag{10.23}$$

is good to $\mathcal{O}(\Delta x^6)$. The associated numerical dispersion relation becomes

$$c = \frac{1}{\alpha\Delta t} \arcsin \left\{ u\alpha\Delta t \left[\frac{4}{3} \frac{\sin\alpha\Delta x}{\alpha\Delta x} - \frac{3}{5} \frac{\sin 2\alpha\Delta x}{2\alpha\Delta x} + \frac{1}{10} \frac{\sin 3\alpha\Delta x}{3\alpha\Delta x} \right] \right\}. \tag{10.24}$$

which to leading order gives

$$c \approx u \left\{ 1 - \frac{36}{7!}(\alpha\Delta x)^6 + \cdots \right\}. \tag{10.25}$$

Comparing (10.25) with (10.20) and (10.21) shows that the sixth order scheme is superior to the fourth order scheme and so on.

Higher order upwind schemes

It is also possible to construct upwind schemes that are of higher order. We first recall that the Taylor expansion of θ_{j-1}^n, assuming a positive advection velocity, gives

$$\partial_x \theta_j^n = \frac{\theta_j^n - \theta_{j-1}^n}{\Delta x} - \frac{1}{2}\partial_x^2 \theta_j^n \Delta x + \mathcal{O}(\Delta x^2). \tag{10.26}$$

Similarly expanding θ^n_{j-2} using Taylor series we get

$$\partial_x\theta^n_j = \frac{\theta^n_j - \theta^n_{j-2}}{2\Delta x} - \partial^2_x\theta^n_j\Delta x + \mathcal{O}(\Delta x^2). \tag{10.27}$$

Multiplying (10.26) by a and (10.27) by b and adding we get

$$(a+b)\partial_x\theta^n_j = a\frac{\theta^n_j - \theta^n_{j-1}}{\Delta x} + b\frac{\theta^n_j - \theta^n_{j-2}}{2\Delta x} + \frac{1}{2}(a+2b)\partial^2_x\theta^n_j\Delta x + \mathcal{O}(\Delta x^2). \tag{10.28}$$

Thus by choosing $a+b=1$ and $a+2b=0$, that is, $a=2$ and $b=-1$, we get

$$\partial_x\theta^n_j = \frac{1}{2\Delta x}(3\theta^n_j - 4\theta^n_{j-1} + \theta^n_{j-2}) + \mathcal{O}(\Delta x^2), \tag{10.29}$$

and hence that a second order in space, first order in time upwind scheme reads

$$\theta^{n+1}_j = \theta^n_j + u_0\frac{\Delta t}{2\Delta x}(3\theta^n_j - 4\theta^n_{j-1} + \theta^n_{j-2}). \tag{10.30}$$

We may also construct a 3rd order upwind scheme by noting that Taylor expansions of θ^n_{j+1}, θ^n_{j-1} and θ^n_{j-2} retaining terms of second order gives, respectively,

$$\begin{aligned}
\partial_x\theta^n_j &= \frac{\theta^n_{j+1} - \theta^n_j}{\Delta x} - \frac{1}{2}\partial^2_x\theta^n_j\Delta x - \frac{1}{6}\partial^3_x\theta^n_j\Delta x^2 + \mathcal{O}(\Delta x^3), && (10.31)\\
\partial_x\theta^n_j &= \frac{\theta^n_j - \theta^n_{j-1}}{\Delta x} + \frac{1}{2}\partial^2_x\theta^n_j\Delta x - \frac{1}{6}\partial^3_x\theta^n_j\Delta x^2 + \mathcal{O}(\Delta x^3), && (10.32)\\
\partial_x\theta^n_j &= \frac{\theta^n_j - \theta^n_{j-2}}{2\Delta x} + \frac{2}{2}\partial^2_x\theta^n_j\Delta x - \frac{4}{6}\partial^3_x\theta^n_j\Delta x^2 + \mathcal{O}(\Delta x^3). && (10.33)
\end{aligned}$$

Consequently multiplying (10.31) by a, (10.32) by b and (10.33) by c and adding, in which we let $a=1/3$, $b=1$ and $c=-1/3$, we get

$$\partial_x\theta^n_j = \frac{1}{6\Delta x}(2\theta^n_{j+1} + 3\theta^n_j - 6\theta^n_{j-1} + \theta^n_{j-2}) + \mathcal{O}(\Delta x^3), \tag{10.34}$$

and finally that a 3rd order upwind scheme for the advection equation is

$$\theta^{n+1}_j = \theta^n_j + u\frac{\Delta t}{6\Delta x}(2\theta^n_{j+1} + 3\theta^n_j - 6\theta^n_{j-1} + \theta^n_{j-2}). \tag{10.35}$$

Note that we have assumed $u \geq 0$. Expanding to include negative velocities is trivial, and is hence left to the reader.

Final comments regarding higher order schemes

Potential complications, however, can arise from these higher-order spatial treatments. For one the stability condition becomes more restrictive as shown by (10.17). A second complication is associated with the boundaries. Since we need to invoke points further and further away from the x_j-point when constructing our higher order schemes, additional boundary conditions are required. When using a higher order scheme to solve the simple advection equation (10.1) we observe that we are only allowed to specify one single boundary condition for θ in x, while the higher order schemes require us to specify more. One way to avoid this problem is to use a lower order CTCS scheme close to the boundary, but then we lessen the accuracy there. Finally we note with regard to the upwind scheme that the higher order schemes are less diffusive than the lowest order scheme.

10.2 Combined advection-diffusion

In Chapter 4 we learned that the diffusion equation was unstable when applying a centered in time, centered in space scheme, while we in Chapter 5 learned that a forward in time, centered in space scheme (Euler scheme) was unstable for the advection equation. As underscored in Chapter 3 most of the problems encountered regarding evolution of tracers in the atmosphere and oceans contain both advection and diffusion in one and the same equation. The question is therefore what scheme should we employ when solving equations which is a combination of the two processes, that is, when solving the so called advection-diffusion equation?

We investigate this by seeking finite difference approximations (FDA) to the continuous combined advection-diffusion equation (3.1). We start by using the parameterization given by (3.3) for the advective flux and (3.4) for the diffusive flux (cf. page 33). Moreover, we simplify the problem by assuming that the problem is one-dimensional in space and that the velocity is constant, that is, $\mathbf{v} = u_0\mathbf{i}$ where u_0 is a constant. Thus the advection-diffusion equation becomes

$$\partial_t\theta + u_0\partial_x\theta = \kappa\partial_x^2\theta. \tag{10.36}$$

To obtain a stable scheme we must ensure that the diffusive part is forward in time and the advective part is centered in time. We may for instance make use of the FDA

$$\frac{\theta_j^{n+1} - \theta_j^{n-1}}{2\Delta t} = -u_0\frac{\theta_{j+1}^n - \theta_{j-1}^n}{2\Delta x} + \kappa\frac{\theta_{j+1}^{n-1} - 2\theta_j^{n-1} + \theta_{j-1}^{n-1}}{\Delta x^2}. \tag{10.37}$$

We notice that the diffusive part is taken at time level $n-1$ and thus becomes forward in time with a time step of $2\Delta t$. In contrast the advective part is evaluated at time step n and is thus centered in time with a time step of Δt. Hence each part is stable in itself. If $\kappa = 0$ then the advective part is stable if the Courant number $C \equiv |u_0|\Delta t/\Delta x \leq 1$. If $u_0 = 0$ the diffusive part is stable under the condition $K \equiv 2\kappa\Delta t/\Delta x^2 \leq 1/2$. The factor $1/2$ arises because of the $2\Delta t$ time step used for the diffusive part, and thus replaces the $1/2$ factor in (4.36) on page 46.

In the general case with $u_0 \neq 0$ and $\kappa \neq 0$ we therefore expect a condition which is a combination of the two pure condtions above to prevail. To find this condtion we employ von Neumann's stability analysis method. Hence we substitute a single, discrete Fourier component into (10.37) to get an equation for the growth factor. The algebra is left to the reader (cf. Exercise 1 at the end of this Chapter) and gives

$$G^2 + 2i\lambda G - \lambda_2 = 0 \tag{10.38}$$

where

$$\lambda = \frac{u_0 \Delta t}{\Delta x} \sin \alpha \Delta x \quad \text{and} \quad \lambda_2 = 1 - 2K(1 - \cos \alpha \Delta x) \tag{10.39}$$

are two real numbers and where C is the Courant number as defined in (5.26) on page 67, and K is defined as in (4.6) on page 40. We note that since $1 - \cos \alpha \Delta x \geq 0$ it follows that $\lambda_2 \leq 1$. In accord with (10.38) the growth factor has two solutions given by

$$G_{1,2} = -i\lambda \pm \sqrt{\lambda_2 - \lambda^2}. \tag{10.40}$$

To ensure that the two complex solutions have a real part we require that the radical is positive, that is, $\lambda^2 \leq \lambda_2$. As a corollary we note that this also implies that $\lambda_2 \geq 0$. The two roots are then complex conjugates and hence

$$|G| = \sqrt{GG^*} = \sqrt{\lambda^2 + \lambda_2 - \lambda^2} = \sqrt{\lambda_2}. \tag{10.41}$$

The solution is thus conditionally stable because $0 \leq \lambda_2 \leq 1$. Moreover the condition $\lambda^2 \leq \lambda_2$ gives

$$C^2 \sin^2 \alpha \Delta x \leq 1 - 2K(1 - \cos \alpha \Delta x) \tag{10.42}$$

where $C = |u_0|\Delta t/\Delta x$ is the Courant number and $K = 2\kappa \Delta t/\Delta x^2$. The condition (10.42) may be rewritten to give

$$(C^2 + 2K) \sin^2 \alpha \Delta x = 1 + 2K \cos \alpha \Delta x (1 - \cos \alpha \Delta x) \tag{10.43}$$

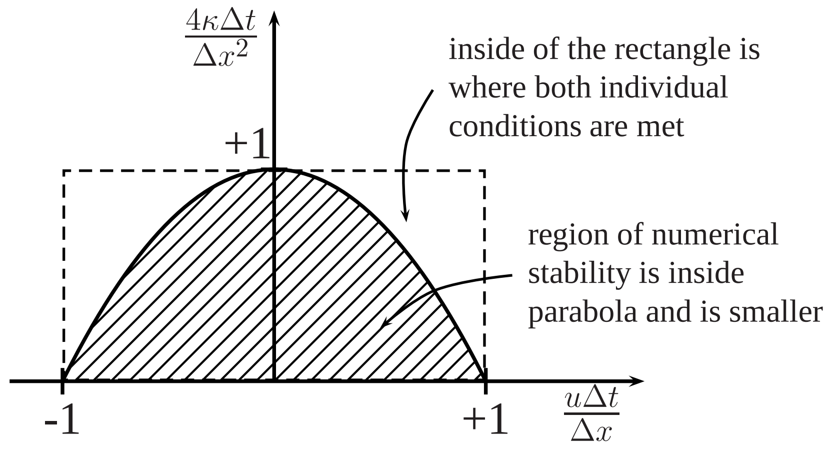
We therefore conclude that the sufficient condition for stability of the combined advection-diffusion scheme (10.37) is

$$C^2 + 2K \leq 1, \quad \text{or} \quad \frac{(u_0 \Delta t)^2 + 2\kappa \Delta t}{\Delta x^2} \leq 1. \tag{10.44}$$

We note that that for either $u_0 = 0$ or $\kappa = 0$, the stability condition for the individual advective and diffusive schemes are recovered. We also note that imposing each condition is not a sufficient condition. We therefore obtain the somewhat surprising results that adding explicit diffusion in the advection equation actually reduces the maximum time step allowed for advection. What (10.44) says is that by adding diffusion we arrive at a more restrictive condition. This is visualized in Fig. (10.1) . For most cases in oceanography and meteorology this is not a serious problem since commonly

$$K \ll C^2. \tag{10.45}$$

We mentioned earlier (cf. Section 4.9) that it is common to add a diffusion term to avoid nonlinear problems to become numerically unstable by so called nonlinear instabilities as discussed in the next section (Section 10.3). The diffusion term is therefore not part of the physics we are solving for, but rather an artificial term added to make the numerical solution stable. Under these circumstances we may use Dufort-Frankel scheme (cf. Section 4.9 on page 51) to approximate the diffusion term, even though it is inconsistent. This is fine as long as the remaining terms



$\frac{4\kappa\Delta t}{\Delta x^2}$
inside of the rectangle is where both individual conditions are met
+1
region of numerical stability is inside parabola and is smaller
-1
+1
$\frac{u\Delta t}{\Delta x}$

Figure 10.1: The diagram illustrates the region of stability for the combined advection-diffusion equation approximated in (10.37). This corresponds to the area inside of the parabola (hatched area). The area inside the rectangular is where both the advection and the diffusion are stable individually. We notice that we a obtain a more stringent stability condition to the advection equation when we are adding diffusion.

in our governing equations are treated by consistent schemes. Thus we proceed by making the following FDA of (10.36),

$$\theta_j^{n+1} = \theta_j^{n-1} - \frac{u_0\Delta t}{\Delta x}(\theta_{j+1}^n - \theta_{j-1}^n) + 2K(\theta_{j+1}^n - \theta_j^{n+1} - \theta_j^{n-1} + \theta_{j-1}^n), \tag{10.46}$$

in which we have combined a consistent conditionally stable scheme for advection with an unconditionally stable, inconsistent scheme for diffusion. The growth factor then follows the equation

$$(1 + 2K)G^2 - 2\lambda G - (1 - 2K) = 0 \tag{10.47}$$

where

$$\lambda = 2K\cos\alpha\Delta x - i\frac{u_0\Delta t}{\Delta x}\sin\alpha\Delta x \tag{10.48}$$

and thus the growth factor has two solutions given by

$$G_{1,2} = \frac{1}{1+2K}\left(\lambda \pm \sqrt{4K^2 + \lambda^2 - 1}\right) \tag{10.49}$$

It can be shown that for the one-dimensional case it is sufficient to satisfy the CFL condition $C \leq 1$. In the more general case for instance for a two-dimensional case a more stringent condition has to be applied (*Cushman-Roisin*, 1984).

Many authors (e.g. *Clancy*, 1981) suggest to use the unstable forward in time, centered in space (FTCS) scheme when combining advection and diffusion. The approximation to (10.36) then becomes

$$\theta_j^{n+1} = \theta_j^n - \frac{u_0 \Delta t}{2\Delta x}(\theta_{j+1}^n - \theta_{j-1}^n) + \frac{\kappa \Delta t}{\Delta x^2}(\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n). \tag{10.50}$$

The amplification or growth factor then follows the equation

$$G = 1 - i\frac{u_0 \Delta t}{2\Delta x}\sin \alpha \Delta x - 2\frac{\kappa \Delta t}{\Delta x^2}(1 - \cos \alpha \Delta x). \tag{10.51}$$

As shown by *Clancy* (1981) the scheme is stable provided the two conditions

$$\frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{2}, \quad \text{and} \quad \frac{|u_0|\Delta t}{\kappa} \leq 1 \tag{10.52}$$

are both satisfied at the same time. Despite the enthusiasm of several authors we do not recommend the use of the FTCS scheme. Rather we advocate to use the more conservative schemes (10.37) and (10.46).

10.3 Non-linear instability

Towards the end of Section 3.3 we mentioned that every non-linear solution of a problem of hyperbolic nature in which friction is neglected will eventually become numerically unstable. This is independent of the time step chosen and is associated with the energy cascade towards smaller and smaller scales that is the nature of non-linear problems. Hence it is not sufficient, for instance in the case of solving the non-linear advection equation (3.18), to satisfy the linear CFL criterion.

To satisfy ourselves that this is indeed true it is enough to solve a simple non-linear advection problem like (3.18). Sooner or later disturbances of wavelengths in the range $2\Delta x$ to $4\Delta x$ crops up. These disturbances are at first small in amplitude but growing. At some stage into the calculation the solution falls short of satisfying the linear CFL condition and the solution blows up, that is, becomes linearly, numerically unstable. The solution is then useless. It is common to credit *Phillips* (1959) to be the first to demonstrate this phenomenon by analytic means. *Richtmyer* (1963) provided another example which is reproduced below. *Robert et al.* (1970) generalized the previous example.

Before entering into details we notice:

1. All good functions may be expanded in terms of a discrete set of waves or exponentials
2. In a linear system waves of different wavelengths exist independent from each other
3. In a non-linear system the latter is no longer true and waves of different wave numbers will interact and sometime generate waves of new periods

4. Given a finite grid of size Δx we have a band-limited wavenumber space, that is, only a finite number of discrete waves can exist

The first point is well known. It simply tells us that all good functions $\Psi(x)$ of period $2L$ may be formulated into a Fourier series, that is, for $x \in [-L, L]$ the function $\Psi(x)$ is written as

$$\Psi(x) = a_0 + \sum_{m=1}^{\infty} a_m \sin(\alpha_m x) + b_m \cos(\alpha_m x) \tag{10.53}$$

where $\alpha_m = m\pi/L$ is the discrete wavenumber, a_m and b_m is the amplitude or energy associated with the wavenumber α_m and a_0 is the mean or average value of Ψ for $x \in [-L, L]$.

The second point tells us that if the system is linear there is no exchange of energy between them. Thus two wave trains of different amplitude, wavelength and direction will pass each other without changing neither of them.

The third point emphasizes the fact that it is the non-linearity that causes exchange to happen. To illustrate this suppose we have a solution, say wind or current $u(x,t)$, given by

$$u(x,t) = \sum_n u_n(t) \sin(\alpha_n x). \tag{10.54}$$

Then nonlinear products will give rise to terms having wavenumbers which are the sum of and difference of the two original wavenumbers, e.g.,

$$\sin(\alpha_1 x)\sin(\alpha_2 x) = \frac{1}{2}\left[\cos(\alpha_1 - \alpha_2)x - \cos(\alpha_1 + \alpha_2)x\right] \tag{10.55}$$

Thus in a non-linear case the two wave trains will be different after the passage, that is, they will experience a change in either wavelength, as illustrated by (10.55), amplitude or direction.

The fourth and last point tells us that when we formulate the function $\Psi(x)$ as a sum of discrete waves on a grid of size Δx we have a band-limited wavenumber space in which the shortest wave that can possibly be resolved is $2\Delta x$. Thus our wavenumber space is limited to wavenumbers $\pi/L \leq \alpha_m \leq \pi/\Delta x$. For a non-linear problem in which the various waves interact to produce waves of wavenumber $\alpha > \pi/\Delta x$, that is waves of wavelengths shorter than $2\Delta x$, they are unresolved by our grid. Unfortunately these unresolved waves are folded into some low wavenumber. In fact as displayed in Figure 10.2 a wave of wavelength $\frac{4}{3}\Delta x$ is indistinguishable from a wave of wavelength $4\Delta x$. Let us arbitrarily call $\alpha < \pi/2\Delta x$ low wavenumbers and $\pi/\Delta x < \alpha < \pi/2\Delta x$ high wavenumbers. The latter are then waves of wavelengths between $2\Delta x$ and $4\Delta x$ and corresponds to the shortest waves that are resolved by our grid of size Δx.

Points three and four above make us expect *a priori* that even though all initial energy is low wavenumber (long waves), non-linear interactions will eventually provide energy (or variance) at high wavenumbers (short waves). This is easily verified by investigating the model problem we use below, which is a simple non-linear advection equation in one dimension, that is,[2]

$$\partial_t u + u\partial_x u = 0. \tag{10.56}$$

[2]Note that (6.155) is the acceleration term in the momentum equation for a one-dimensional problem. Hence non-linearity is ubiquitous in all realistic atmospheric and oceanographic models.

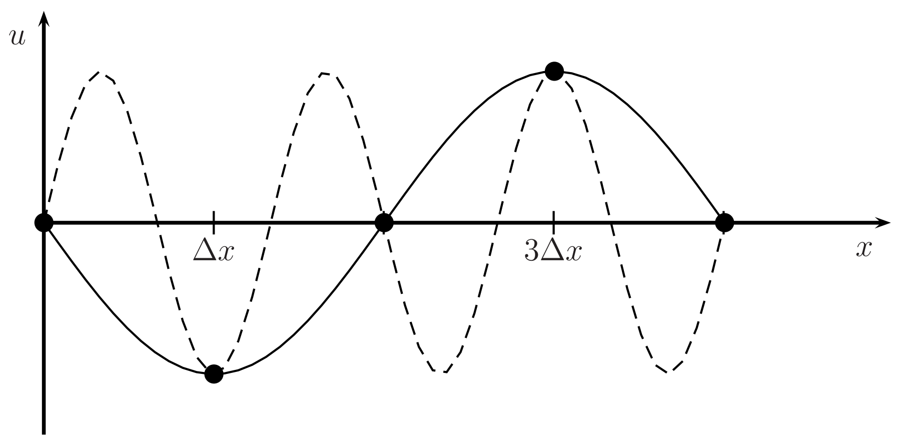


Figure 10.2: Displayed are the two waves of wavelength $4\Delta x$ (solid curve) and $\frac{4}{3}\Delta x$ (dashed curve), in a grid of grid size Δx. Note that our grid cannot distinguish between the unresolved wave of wavelength $\frac{4}{3}\Delta x$ and the resolved wave of wavelength $4\Delta x$. Thus the energy contained in the unresolved wave will be folded into the low wavenumber space represented by the $4\Delta x$ wave.

The difference between (6.155) and the earlier advection equation we studied in Chapter 5, e.g., (5.2) on page 61, is the appearance of the nonlinear term $u\partial_x u$. Suppose we have a solution at a particular time level n that is a monochromatic wave of wavelength $2\pi/\alpha$ where α is the wavenumber, that is, $u^n(x) = u_0 \sin \alpha x$. Then from (6.155), using a scheme that is centered in time, we get

$$u^{n+1} - u^{n-1} = -u_0^2 \sin \alpha x \partial_x \sin \alpha x = -u_0^2 \sin \alpha x \cos \alpha x = -\frac{1}{2} u_0^2 \sin 2\alpha x. \qquad (10.57)$$

Hence the solution at the next time level is a wave of wavelength $2\pi/2\alpha$, that is, a wavelength half of that of the original wavelength at time level n. Thus, we observe, as expected, that all the energy originally contained at low wavenumbers (long waves) end up at high wavenumbers unresolved by our grid. Due to the folding of the energy contained in the unresolved waves, the energy contained in the shortest wave resolved by our grid, that is, waves of wavelength $4\Delta x$, accumulates. Thus after a sufficient time period the numerical model blows up due to ordinary numerical, linear instability, even though the linear problem is numerically stable.

To inspect the non-linear instability in some more detail we use the example of *Richtmyer* (1963). We start with the assumption the problem has variance at wavelengths ∞ (zero wavenumber), $4\Delta x$ and $2\Delta x$, and that the model problem is the simple non-linear advection equation in one dimension (6.155). Let us as *Richtmyer* (1963) approximate (6.155) using the classic leapfrog (CTCS) scheme, that is

$$u_j^{n+1} = u_j^{n-1} - \frac{\lambda}{2}\left[(u_{j+1}^n)^2 - (u_{j-1}^n)^2\right] \qquad (10.58)$$

where $\lambda = \Delta t/\Delta x$. We will not concern ourselves with boundary conditions, but as in the von Neumann analysis of Section 4.4 on page 44 perform a local analysis. Then making use of

(10.53) with $\Psi = u$ and $a_0 = V$ we can show that an exact, formal solution to (10.58) is

$$u_j^n = C_n \cos(\frac{\pi j}{2}) + S_n \sin(\frac{\pi j}{2}) + U_n \cos(\pi j) + V. \tag{10.59}$$

We can identify the amplitudes C_n, S_n as the amplitudes of a wave with length $4\Delta x$, U_n as the amplitude of a wave of length $2\Delta x$, and V as a wave of low wavenumber ($\alpha < \pi/2\Delta x$) with zero wavenumber (infinite wavelength). When we substitute (10.59) into (10.58) and notice that $(u_{j+1}^n)^2 - (u_{j-1}^n)^2 = (u_{j+1}^n - u_{j-1}^n)(u_{j+1}^n + u_{j-1}^n)$ we obtain relationships among the amplitudes,

$$\begin{aligned} C_{n+1} - C_{n-1} &= 2\lambda S_n(U_n - V) \\ S_{n+1} - S_{n-1} &= 2\lambda C_n(U_n + V) \\ U_{n+1} &= U_{n-1}. \end{aligned} \tag{10.60}$$

The last equation in (10.60) says that U_n may take on different values for the odd and even time steps, say A for odd time steps ($n = 1, 3, 5, \ldots$) and B for the even time steps ($n = 2, 4, 6, \ldots$), that is, $U_{2m} = A$ and $U_{2m-1} = B$ for $m = 1, 2, 3, \ldots$. By eliminating S_n from the first equation in (10.60), we obtain

$$C_{n+2} - 2C_n + C_{n-2} = 4\lambda^2(A + V)(B - V)C_n. \tag{10.61}$$

The question then arise. Is this solution stable in the von Neumann sense? As in the simple linear case, using the von Neumann method, we define a growth factor associated with the $4\Delta x$ wave. For the $4\Delta x$ wave to be stable the growth factor has to be less than or equal to one. Thus we first define the growth factor by letting $G \equiv C_{n+2}/C_n$. Substituting this into (10.61) we derive

$$G^2 - 2\gamma G + 1 = 0, \tag{10.62}$$

where γ is a real number given by

$$\gamma = 1 + 2\lambda^2(A + V)(B - V). \tag{10.63}$$

The roots of (10.62) are

$$G_{1,2} = \gamma \pm i\sqrt{1 - \gamma^2}. \tag{10.64}$$

We notice that as long as the radical is real then

$$|G_{1,2}| = \sqrt{\gamma^2 + 1 - \gamma^2} \equiv 1 \tag{10.65}$$

The $4\Delta x$ wave is therefore neutrally stable provided

$$1 - \gamma^2 \geq 0 \quad \text{or} \quad -1 \leq \gamma \leq 1. \tag{10.66}$$

As is obvious it is only possible to satisfy (10.63) if the amplitude of the $2\Delta x$ wave is such that $|A| < V$ and/or $|B| < V$. This is violated when the amplitude of the $2\Delta x$ wave is large in comparison with the energy contained in the longer waves (low wavenumbers). In this case the $4\Delta x$ will grow exponentially and the scheme is unstable.

10.4 Smoothing and filtering

In numerical models of the atmosphere or ocean, we have learned that it is important to damp out the smallest space scales to control for instance non-linear instability. We also notice that this may be done by adding explicit eddy viscosity of momentum diffusion as described in Section 3.3 and Section 4.9. Here we focus on another method, namely employing filtering techniques to control spurious growth of short waves due to numerical errors and computational instabilities that would otherwise obscure a good forecast. In fact even if a catastrophic instability does not occur we still may want to remove the noise in the shortest wavenumber band for aesthetic reasons. Sometimes we apply such smoothing to the final product only.

The simplest form of smoothing is to apply a so called one-dimensional three-point operator or filter often referred to as the Shapiro filter (*Shapiro*, 1970, 1975). The filter is defined by

$$\bar{u}_j^n = (1-\mu)u_j^n + \frac{1}{2}\mu\left(u_{j+1}^n + u_{j-1}^n\right), \tag{10.67}$$

where μ is a constant. If the solution is a monochromatic wave, say $u_j = U_n e^{i\alpha j\Delta x}$, then the filtered solution is

$$\bar{u}_j^n = Ru_j^n \tag{10.68}$$

where

$$R = 1 - \mu(1 - cos\alpha\Delta x) = 1 - 2\mu\sin^2\left(\frac{\alpha\Delta x}{2}\right) \tag{10.69}$$

is the *response function* associated with the filter. Thus the filter does not affect the wave length nor the phase speed (provided $R \geq 0$). Furthermore if $R < 1$ then the wave is damped. Moreover, for the particular wave number $\alpha = 2\pi/2\Delta x$, that is, for the shortest wave resolved in our grid, we get

$$R = 1 - 2\mu. \tag{10.70}$$

For the particular choice $\mu = 1/2$ we then get $R = 0$, and hence the waves of wavelength $2\Delta x$, the two gridlength waves, are completely removed by the filter.

We furthermore observe that the filter may rewritten to yield

$$\bar{u}_j^n = u_j^n + \frac{1}{2}\mu\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right). \tag{10.71}$$

Making use of Taylor series we recognize the last term on the right-hand side of (10.71) as the finite difference approximation of the second order derivative in space with a truncation error of $\mathcal{O}(\Delta x^2)$. Thus we get

$$\bar{u}_j^n = u_j^n + \frac{1}{2}\mu\Delta x^2\left[\partial_x^2 u\right]_j^n, \tag{10.72}$$

which shows that the filter acts similar to diffusion with a mixing coefficient given by $\kappa = \frac{1}{2}\mu\Delta x^2$. For more in depth details the reader is referred to (*Haltiner and Williams*, 1980, Chapter 11-8, page 392 and onward).

10.5 Two-way nesting

Nesting techniques of meshes generally consists of a local high resolution grid (child grid) embedded in a coarse resolution grid (parent grid) that provides the boundary conditions for the child grid (cf. Section 7.2 and Figure 7.2 on page 127). If this is the only transfer of information between the two grids, the method is said to be a *one-way nesting* technique, e.g., Sections 7.7 and 7.8. If there is also a transfer of information from the child back to the parent grid, the method is said to be a *two-way nesting* technique. A general review of two-way nesting algorithms may be found in *Debreu and Blayo* (2008) and *Debreu et al.* (2012), along with applications focusing on upscaling impact (*Biastoch et al.*, 2008), fine-scale dynamics (*Marchesiello et al.*, 2011), and topographic refinement (*Sannino et al.*, 2009).

Let us assume that the governing equations of our model, to be solved for both the parent and child grid domains, are

$$\partial_t q = \boldsymbol{\mathcal{L}}[q] \tag{10.73}$$

in which q represent an array containing the variables and $\boldsymbol{\mathcal{L}}$ is a spatial operator. The model we consider may for instance be one of the three equations treated in Chapters 4 - 6. Thus $\boldsymbol{\mathcal{L}} = \kappa\partial_x^2$ and $\boldsymbol{\mathcal{L}} = -u_0\partial_x$ if the model is the diffusion equation (4.1) respectively the advection equation (5.2). Finally, if our model is the non-linear, rotating shallow water equations (6.22) - (6.24) then

$$q = \begin{bmatrix} u \\ v \\ h \end{bmatrix} \quad \text{and} \quad \boldsymbol{\mathcal{L}} = \begin{bmatrix} -u\partial_x & f & -g\partial_x \\ -f & -u\partial_x & 0 \\ -h\partial_x & 0 & -u\partial_x \end{bmatrix}. \tag{10.74}$$

Equation (10.73) is discretized on the parent (subscript p) and child (subscript c) domains by

$$[\partial_t q]_p = \boldsymbol{\mathcal{L}}_p[q_p] \quad \text{and} \quad [\partial_t q]_c = \boldsymbol{\mathcal{L}}_c[q_c], \tag{10.75}$$

where $\boldsymbol{\mathcal{L}}_p$ and $\boldsymbol{\mathcal{L}}_c$ are the same discretizations of the same continuous operator $\boldsymbol{\mathcal{L}}$, but at different resolutions[3]. The problem is to solve (10.75) within the parent and child domains (Figure 7.2) so that the parent domain solution impacts the child domain solution while at the same time the child domain solution is allowed to impact the parent domain solution.

Let us assume that the discretization we use to solve (10.75) results in an explicit scheme. Moreover, let the time step and space increments we use be denoted $\Delta t_{p,c}$ and $\Delta x_{p,c}$, respectively. Due to the refinement of the grid employed for the child domain both the time step and space increments are smaller. Let i_r denote the spatial refinement factor. Then we get $\Delta x_p = i_r \Delta x_c$. Conveniently we also let the time refinement factor be i_r so that $\Delta t_p = i_r \Delta t_c$. For interface continuity reasons the refinement factor cannot be too large. A number between 1 and 5 is common. We note that if (10.73) are the advection equation or the shallow water equations then the parent and child grids have the the same Courant number C, and thus obey the same stability criterion $C \leq 1$.

With this in mind the steps whereby the two-way nesting is performed are

[3]In principle, a different choice of numerical schemes and parameterizations may be adopted in the refined grid. However, this would complicate the issue of interface continuity already posed by the grid refinement itself.

1. Forward the parent from time level n to time level $n+1$ for all interior parent grid points,

$$q_p^{n+1} = q_p^n + \Delta t_p \boldsymbol{\mathcal{L}}_p^n[q]; \quad \mathbf{r}_p \in \Omega, \tag{10.76}$$

   and update the solution at its outer boundary by taking into account the boundary conditions of the parent domain[4].

2. Update the child at the interface Γ for $m = 1, 2, \ldots, i_r$ by interpolating parent values using a time interpolator $\boldsymbol{\mathcal{P}}$

$$q_c^{n+\frac{m}{i_r}}|_\Gamma = \boldsymbol{\mathcal{P}}[q_p^n, q_p^{n+1}] \tag{10.77}$$

3. Forward the child for $m = 1, 2, \ldots, i_r$ for all interior child grid points by

$$q_c^{n+\frac{m}{i_r}} = q_c^{n+\frac{m-1}{i_r}} + \Delta t_c \boldsymbol{\mathcal{L}}_c^{n+\frac{m-1}{i_r}}[q]; \quad \mathbf{r}_c \in \omega \tag{10.78}$$

4. Finally, update the parent within the child domain by filtering the child at their common grid points by

$$q_p^{n+1} = \boldsymbol{\mathcal{R}}[q_c^{n+1}]_p, \quad \mathbf{r}_c \in \omega \tag{10.79}$$

   where $\boldsymbol{\mathcal{R}}$ is the filter (sometimes referred to as the restriction operator).

Note that Steps 1 through 3 are the same as those we perform when applying the one-way nested technique, except that in Step 2 the OBC (cf. Chapter 7) is replaced by a simple interpolation scheme. The latter forces q_c^{n+1} to equal q_p^{n+1} on Γ. The innovation is the fourth step in which we let the child solution update the parent solution. This is done by replacing the parent domain solution obtained through solving (10.76) by a filtered child domain solution at the grid points the two domains have in common, hence the subscript p on the right-hand side of (10.79). The so corrected parent domain solutions are then used when we forward the parent to the next time level. We emphasize that this is done only within the domain ω, that is, we do not update the parent at the interface Γ of the two domains or outside of the child domain. Nevertheless, since the parent solution is changed within the child, the child solution may impact the parent even outside of its domain.

The rationale behind applying the filter $\boldsymbol{\mathcal{R}}$ in Step 4 is to filter out those smaller scale variations in the child that is more or less unresolved by the parent, but at the same time include as much as possible of those scales that are resolved by the parent. The filter may be a simple avarage, that is,

$$[q_p]_{j_p}^{n+1} = \frac{1}{3}\left([q_c]_{j_c+1}^{n+1} + [q_c]_{j_c}^{n+1} + [q_c]_{j_c-1}^{n+1}\right) \tag{10.80}$$

a Shapiro filter (cf. Section 10.4 above),

$$[q_p]_{j_p}^{n+1} = [q_c]_j^{n+1} + \frac{1}{2}\mu([q_c]_{j_c+1}^{n+1} - 2[q_c]_{j_c}^{n+1} + [q_c]_{j_c-1}^{n+1}), \tag{10.81}$$

[4]Note that any time discretization may be used to solve (10.76) as long as an appropriate spatial dicretization is used that renders the scheme stable and consistent.

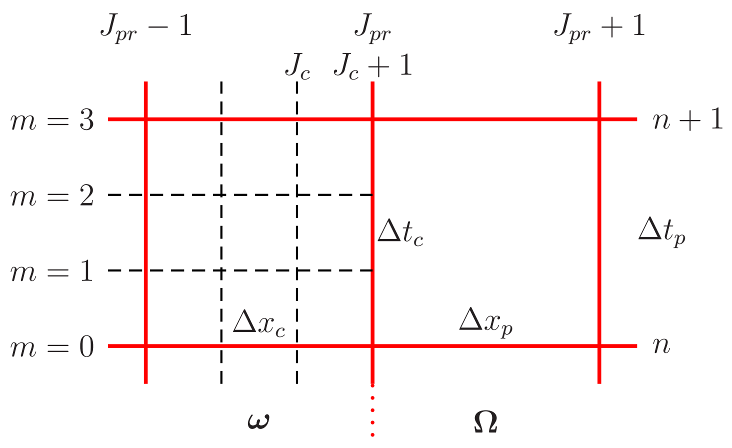


Figure 10.3: Close up of the parent (solid lines) and child (dashed) grid points locations in time and space in the vicinity of the right-hand interface. The refinement factor is $i_r = 3$.

which for $\mu = \frac{1}{2}$ becomes a simple 1-2-1 filter, or a more sophisticated filter like a full weighted filter

$$[q_p]^{n+1}_{j_p} = \frac{1}{9}([q_c]^{n+1}_{j_c+2} + 2[q_c]^{n+1}_{j_c+1} + 3[q_c]^{n+1}_{j_c} + 2[q_c]^{n+1}_{j_c-1} + [q_c]^{n+1}_{j_c-2}). \tag{10.82}$$

Here j_p, j_c are the grid point counters for the parent and child, respectively. Note that it is assumed that in these filter formulae the mid point j_c is a grid point which the child and the parent has in common. Like the Shapiro filter the filters have different response function. Nevertheless they all damp the amplitudes of small scale variations in the child solution before applying them to the parent.

10.5.1 A simple example: Advection of a bell function

As an example let us consider the one-dimensional, advection equation (5.2). Then let $q = \psi$, where ψ is the concentration of a solluble in percent, $\mathcal{L}[q] = u_0\partial_x\psi$, where u_0 is the advection speed (assumed positive), and for which the initial concentration is a bell shaped distribution given by

$$\psi(x, 0) = \psi_0 e^{-(x/\sigma)^2}, \tag{10.83}$$

where σ is a measure of the width of the bell, and ψ_0 the initial maximum concentration.

Let the parent domain spanning the domain $-L < x_p < L$, where L is the halfwidth of the domain, and let $j_p = 1(1)J_p + 1$ and $j_c = 1(1)J_c + 1$ be the counters for the grid points of the parent and child domains, respectively. Furthermore let the location of the common grid points where the left-hand ($j_c = 1$) and right-hand ($j_c = J_c + 1$) interface between the parent and grid domains be $j_p = J_{pl}$ and $j_p = J_{pr}$, respectively, as illustrated by Figure 10.3. Furthermore, let the child grid occupy a portion of the parent domain, that is, $-aL < x_c < bL$ where a and b are positive definite constants less than or equal to one. We then get

$$J_{pl} = 1 + \frac{(1-a)L}{\Delta x_p} \quad \text{and} \quad J_{pr} = 1 + \frac{(1+b)L}{\Delta x_p}. \tag{10.84}$$

Thus for $a = b = 1$ we get $J_{pl} = 1$ and $J_{pr} = J_p + 1$ in which case the child grid covers the entire parent domain[5].

Let $\hat{\psi}_{j_p}$, $\tilde{\psi}_{j_c}$ denote ψ at parent and child grid points, respectively. Employing the convergent, stable and consistent upstream scheme to discretize (5.2) Step 1 above becomes

$$\hat{\psi}_{j_p}^{n+1} = \hat{\psi}_{j_p}^{n} - C(\hat{\psi}_{j_p}^{n} - \hat{\psi}_{j_p-1}^{n}); \quad j_p = 2(1)J_p + 1, \tag{10.85}$$

where $\hat{\psi}_{j_p}$ is used to denote ψ at the parent grid points and $C = u_0\frac{\Delta t_p}{\Delta x_p}$ is the Courant number. Imposing cyclic boundary conditions at the left-hand and right-hand boundaries of the domain we get

$$\hat{\psi}_{1}^{n+1} = \hat{\psi}_{J_p+1}^{n+1}. \tag{10.86}$$

Step 2 involves updating the child at the boundaries for $m = 1, 2, \ldots, i_r$. Using a two-point, linear interpolator we get

$$\tilde{\psi}_{1}^{n+\frac{m}{i_r}} = (1 - \frac{m}{i_r})\hat{\psi}_{J_{pl}}^{n} + \frac{m}{i_r}\hat{\psi}_{J_{pl}}^{n+1}, \tag{10.87}$$

$$\tilde{\psi}_{J_c+1}^{n+\frac{m}{i_r}} = (1 - \frac{m}{i_r})\hat{\psi}_{J_{pr}}^{n} + \frac{m}{i_r}\hat{\psi}_{J_{pr}}^{n+1}. \tag{10.88}$$

Having updated the interface values we may proceed to Step 3 above, which is to solve (5.2) numerically on the child grid. Note that the Courant number is the same since $\frac{\Delta t_p}{\Delta x_p} = \frac{\Delta t_c}{\Delta x_c}$. Employing the same scheme as used for the parent grid we therefore get

$$\tilde{\psi}_{j_c}^{n+\frac{m}{i_r}} = \tilde{\psi}_{j_c}^{n+\frac{m-1}{i_r}} - C(\tilde{\psi}_{j_c}^{n+\frac{m-1}{i_r}} - \tilde{\psi}_{j_c-1}^{n+\frac{m-1}{i_r}}); \quad j_c = 2(1)J_c \tag{10.89}$$

for $m = 1, 2, \ldots, i_r$.

The remaning fourth step is to update the concentration using the child solution at the parent grid points in the interior of the child grid, that is, at those locations where the child and the parent grid points coincide. To this end we may for instance use the Shapiro filter (10.81) with $\mu = \frac{1}{2}$ (a 1-2-1 filter). Step 4 then becomes

$$\hat{\psi}_{j_p}^{n+1} = \frac{1}{4}\left(\tilde{\psi}_{j_c^*+1}^{n+1} + 2\tilde{\psi}_{j_c^*}^{n+1} + \tilde{\psi}_{j_c^*-1}^{n+1}\right); \quad j_p = J_{pl} + 1(1)J_{pr} - 1, \tag{10.90}$$

where j_c^* denotes a child grid point that coincides with the parent grid point. The solution is displayed in Figure 10.4 where $L = 50$ km, $\sigma = L/5$ and the refinement factor is set to $i_r = 5$.

Recall that the upstrean scheme is numerically diffusive with a diffusion coefficient given by (5.105), that is,

$$\kappa_p^* = \frac{1}{2}(1 - C)|u_0|\Delta x_p, \quad \text{and} \quad \kappa_c^* = \frac{1}{2}(1 - C)|u_0|\Delta x_c, \tag{10.91}$$

respectively. Since $\kappa_p^* = i_r\kappa_c^*$ we therefore expect the child solution to be less diffusive than the parent solution when $i_r > 1$. This is illustrated by the solutions to (10.85) and (10.89)

[5]Recall that $2L = (J_p + 1)\Delta x_p$ in accord with (2.48) on page 24, and hence $J_{pr} = J_p + 1$ when $b = 1$.

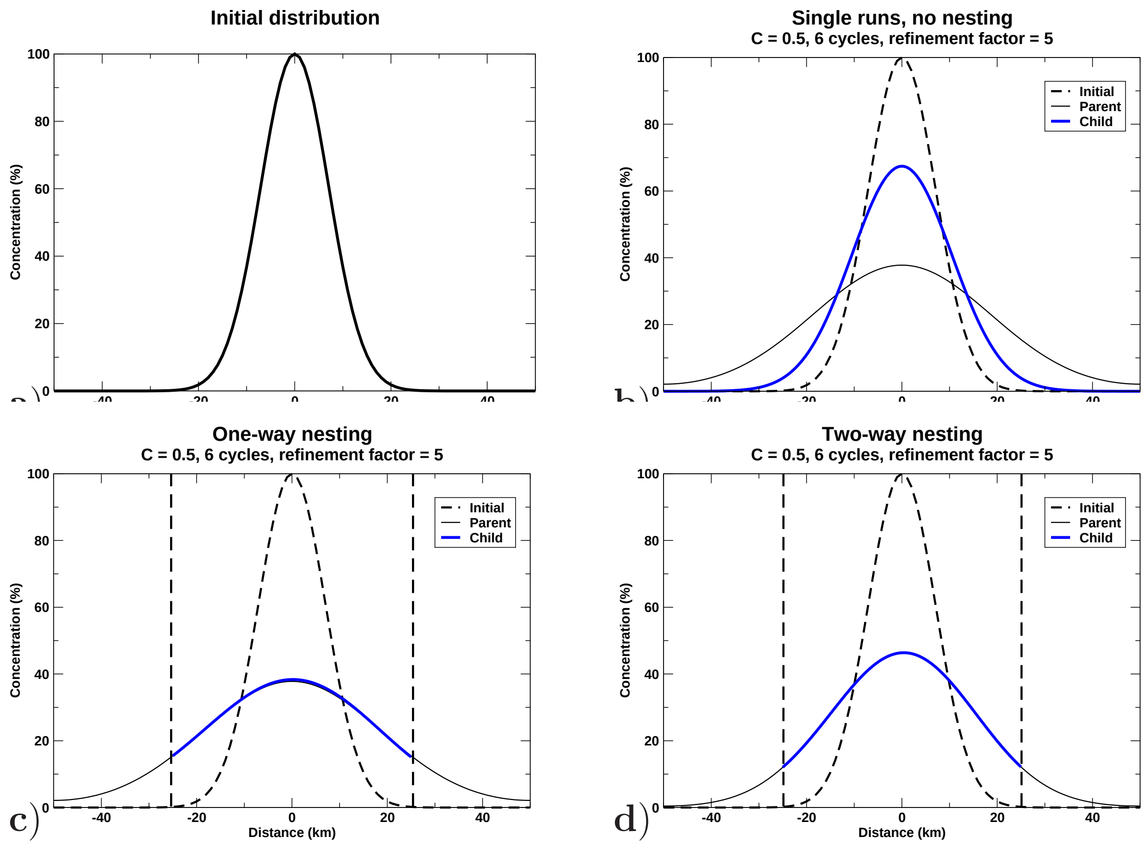


Figure 10.4: a) Initial distribution. b), c) and d) are distributions after 6 cycles with respectively no, one-way, and two-way nesting applied. In b) the child grid covers the entire domain of the parent grid ($a = b = 1$ in eq. 10.84), and applies a cyclic boundary condition on the boundaries. The vertical, dashed lines in c) and d) indicate the interface between the parent and child domains. In b), c) and d) the solid thick, blue line is the child solution while the black, thin line is the parent solution. Dashed line is the analytic solution after 6 cycles and corresponds to the intital distribution shown in a).

displayed by the upper, right-hand panel of Figure 10.4. In this case we apply no nesting and let $a = b = 1$ in (10.84). Under these circumstances the child domain covers the entire parent domain. Consequently there is no interaction between the parent and child solutions and thus we have applied a cyclic boundary condition similar to (10.86) at the common boundaries at $x = \pm L$ when solving (10.85) and (10.89). We note that neither of the two solutions are perfect. In fact the correct, analytic solution is the dashed curve replicating the initial distribution for each cycle. We also note that the parent solution is more diffusive than the child solution as expected since $i_r = 5 > 1$.

Two questions arise. One is the impact on the child solution applying a one-way nesting procedure as described in Chapter 7. A second is the impact on the parent and child solutions applying a two-way nesting procedure as described in this section. The answer to these questions is depicted by the two lower panels of Figure 10.4. In deriving the depicted solutions we have followed the two-way nesting procedure with the child domain embedded within the parent domain by letting $a = b = \frac{1}{2}$ in (10.84). We observe that when applying a one-way nesting (look at the lower, left hand panel of Figure 10.4) nothing happens to the parent solution, while the child solution is severely degraded. This is to be expected since a one-way nesting has no impact on the parent solution outside or inside of the child domain. Applying two-way nesting changes this picture as shown by the lower, right-hand panel of Figure 10.4. Both the parent and the child solutions are improved compared to the one-way nesting. Moreover, the parent solution is improved outside of as well as inside of the child grid domain. Although the child solution is still degraded compared to the no nesting case it is nevertheless improved compared to the one-way nesting case.

A third question that arises is how large the ratio i_r may be. The simple case above is not

10.6 The spectral method

When we applied the various approximation to the advection equation above, we only consider grid-point values of the dependent variables. We did not make any assumption about how the variables behaved between grid points other than assuming that they are good functions.

An alternative approach is to expand the dependent variables in terms of finite series of orthogonal functions (cf. Section 2.10 on page 25). The problem is then reduced to solving a set of ordinary differential equations which determine the behavior in time of the expansion coefficients. Following this approach is known as the *spectral method*.

The spectral method is particularly suitable for global atmospheric models where the dependent variables are zonally cyclic function and hence easy to expand. The method is therefore commonly applied in modern global atmospheric models, for instance in the global model used at the European Centre for Medium range Weather Forecast (ECMWF). We note that the method is a bit more cumbersome to apply in non-global models, and also in global oceanographic models since the latter has to deal with the continental land boundaries.

10.6.1 Application to the one-dimensional linear advection equation

We will demonstrate the spectral method for the one-dimensional advection equation on the globe, i.e., along a latitude. Under these circumstances the natural boundary condition is the periodic or cyclic boundary condition (cf. Section 2.5 on page 16).

We recall from Section 5.2 on page 63 that the one-dimensional advection equation is

$$\partial_t \phi = -u_0 \partial_x \phi, \quad \text{for} \quad x \in [0, L] \quad \text{and} \quad t > 0 \tag{10.92}$$

where L is the length of the circumference at a particular latitude. We recall from Section 2.10 that (10.92) is just a special case of the general equation (2.66) on page 26. Hence the linear operator of Section 2.10 is $\mathcal{H} = -u_0 \partial_x$. Since we will solve (10.92) along a latitude we first conveniently transform to a coordinate system in which

$$2\pi x = \xi L \quad \text{or} \quad \xi = \frac{2\pi x}{L} \tag{10.93}$$

where $\xi \in < 0, 2\pi >$ is the new (dimensionless) zonal coordinate. Since

$$\partial_x \phi = \partial_\xi \phi \partial_x \xi \tag{10.94}$$

(10.92) then transforms to

$$\partial_t \phi = -\gamma \partial_\xi \phi \tag{10.95}$$

where

$$\gamma = \frac{2\pi u_0}{L} \tag{10.96}$$

is the angular velocity (in units one per second). The cyclic boundary condition is written

$$\phi(\xi, t) = \phi(\xi + 2\pi m) \quad m = 1, 2, 3, \ldots \tag{10.97}$$

where m describes the number of times you have traveled around the world at that latitude. We further let the initial condition be described by the good function $f(\xi)$, and hence

$$\phi(\xi, 0) = f(\xi). \tag{10.98}$$

As outlined at the beginning of Section 5.2 on page 62 the true solution to (10.95) is then

$$\phi = f(\xi - \gamma t). \tag{10.99}$$

Solving (10.95) using expansions in terms of orthogonal functions requires us to choose a suitable set of expansion functions. The obvious choice in our case following (10.53) on page 168 above is to choose complex exponentials (sine and cosine functions), since these are eigenfunctions of the differential operator $\mathcal{H} = -u_0 \partial_x$. For a continuous function we get

$$\phi = \sum_\alpha \phi_\alpha(t) e^{i\alpha\xi}, \tag{10.100}$$

where α is the wave number and the summation is for all possible wavenumbers from $-\infty$ to $+\infty$. Solving (10.95) using numerical methods implies that we are band-limited in wavenumber space and hence we must use a truncated version of (10.100), that is,

$$\phi(\xi, t) = \sum_{l=-l_{max}}^{l=l_{max}} \phi_l(t) e^{i\alpha_l \xi}. \tag{10.101}$$

where $l = l_{max}$ gives the maximum wave number $\alpha_{l_{max}}$ resolved on the grid, that is the shortest wavelength resolved by our choice of grid size (here $2\Delta\xi$). Since $\phi_{-l} = \phi_l^*$, we need only be concerned with $0 \leq l \leq l_{max}$, rather than the full set of expansion coefficients.

We now substitute (10.101) into (10.95) and equate coefficients of the expansion functions. Thus

$$\partial_t \phi_l = -i\alpha_l \gamma \phi_l \quad \text{for} \quad l = 0(1)l_{max} \tag{10.102}$$

giving $2l_{max} + 1$ equations for the expansion coefficients ϕ_l's. For this particular case (10.102) can be integrated exactly for each wavenumber α_l separately to give

$$\phi_l(t, \alpha_l) = \phi_l(0, \alpha_l) e^{-i\alpha_l \gamma t} \tag{10.103}$$

where $\phi_l(0, \alpha_l)$ is the initial condition associated with the wavenumber α_l. If we expand the good function $f(\xi)$ in terms of a truncated Fourier series, that is,

$$f(\xi) = \sum_l a_l e^{i\alpha_l \xi} = \sum_l \phi_l(0, \alpha_l) e^{i\alpha_l \xi}, \tag{10.104}$$

we get that $\phi_l(0, \alpha_l) = a_l$, and hence that the complete solution to (10.95) is

$$\phi(\xi, t) = \sum_{l=0}^{l_{max}} a_l e^{i\alpha_l(\xi - \gamma t)}, \tag{10.105}$$

which is the same as the true solution. Hence there is no dispersion due to the space discretization, unlike in the finite difference approximation above. This fact is due to the space derivatives being computed analytically while they were approximated in the finite difference method. We recall due to the orthogonality property of the expansion functions, in our case $e^{i\alpha_l \xi}$, that

$$\phi_l = \int_0^{2\pi} \sum_m \phi_m e^{i\alpha_m \xi} e^{-i\alpha_l \xi} \mathrm{d}\xi. \tag{10.106}$$

Thus by multiplying (10.101) by the complex conjugate of the expansion functions and integrating in space we get

$$\phi_l(t; \alpha_l) = A_l \int_0^{2\pi} \phi(\xi, t) e^{-i\alpha_l \xi} \mathrm{d}\xi, \tag{10.107}$$

where A_l are the normalization factors. Note that (10.107) is the so called direct Fourier transform. The normalization coefficients are determined from the initial condition, or by use of (10.107), that

$$A_l = \frac{a_l}{\int_0^{2\pi} \phi(\xi, 0) e^{-i\alpha_l \xi} \mathrm{d}\xi}. \tag{10.108}$$

In practice we have at our disposal the grid points values of ξ rather than a continuous function in space. Thus we know ξ at $J+1$ points $\Delta\xi$ apart such that $\xi_j = j\Delta\xi$ where $j = 0, 1, 2, \cdots, J-1, J$ and where $\xi_J = 2\pi$. In this case we think of the truncated Fourier series of ϕ as given in (10.101) as representing an interpolating function which exactly fits the values of ϕ at the $J+1$ grid points. We then write (10.107) as a discrete direct Fourier transform,

$$\phi_l(t, \alpha_l) = A'_l \sum_{j=1}^{J} \phi(\xi_j, t) e^{-i\alpha_l \xi_j}, \tag{10.109}$$

where the normalization coefficients are found by discretization (10.108),

$$A'_l = \frac{a_l}{\sum_{j=1}^{J} \phi(\xi_j, 0) e^{-i\alpha_l \xi_j}}. \tag{10.110}$$

The corresponding discrete inverse Fourier transform is then

$$\phi(\xi_j, t) = \sum_{l=-l_{max}}^{l_{max}} \phi_l(t, \alpha_l) e^{i\alpha_l \xi_j}. \tag{10.111}$$

Both (10.109) and (10.111) can be computed with the Fast Fourier Transform (FFT) algorithm. It can be shown that starting from the set $\phi_l(t, \alpha_l)$ going to the set $\phi(\xi_j, t)$ with $j = 0, 1, 2, \cdots, J-1, J$ and returning to the set $\phi_l(t, \alpha_l)$ we recover exactly the original values provided the number of grid points J are such that $J > 2l_{max} + 1$. Recall that l_{max} is the number of waves used to compute the direct Fourier transform in (10.111). In addition we must require that the points ξ_j are equally spaced or that $\Delta\xi$ is a constant.

It remains to find the expansion coefficients $\phi_l(t, \alpha_l)$ at an arbitrary time given their initial values ϕ_{0l}. We do this by a time stepping procedure, for instance applying a centered in time scheme to (10.102),

$$\phi_l^{n+1} = \phi_l^{n-1} + 2i\alpha_l \gamma \Delta t \phi_l^n \quad l = 1, 2, 3, \cdots, LM \tag{10.112}$$

for each wavenumber α_l. Using von Neumann's method we show that numerical stability is ensured provided

$$|\alpha_l \gamma \Delta t| \leq 1; \quad \forall l. \tag{10.113}$$

Since the maximum wavenumber is α_{LM} we require $|\alpha_{LM} \gamma \Delta t| \leq 1$. Moreover, since the maximum dimensionless wavenumber[6] is $\alpha_{LM} = L/2\Delta x$ it follows that the stability condition in terms of the Courant number $C = u_0 \Delta t/\Delta x$ is

$$C \leq \frac{1}{\pi}, \tag{10.114}$$

which is actually more stringent than the one derived for the finite difference approximation. Although being more restrictive the spectral method and scheme has the great advantage that it is nearly non-dispersive, and that the dispersiveness is very small even for the shortest waves of two grid lengths.

Exercises

1. Use von Neumann's method (Chapter 4.4) to show that the expression (10.38) is indeed the correct expression for the growth factor when using the scheme given in (10.37).

2. Show that the condition (10.44) is a sufficient condition for numerical stability.

[6]The maximum dimensional wavenumber is $2\pi/2\Delta x$.

Appendix A

Introduction to Fortran 2003 via examples

by Gunnar Wollan[1] and Lars Petter Røed

The following gives a quick insight into the Fortran 2003 programming language via specific examples. Through this you learn how to solve a computational problem and how to handle reading and writing of data to files. For more details, in particular regarding Fortran 90/95 we recommend to download Fortran texts from the net. Specifically we recommend the site: `http://www.nsc.liu.se/∼boein/f90/`. Here you find versions both in English and in Swedish.

A.1 Why use Fortran?

But first the obvious question. *Why do I have to learn to program in Fortran*? In the field of Meteorology and Oceanography you will probably come across atmospheric models like WRF and CAM and ocean models like ROMS, NEMO, or other models like them. Depending on your project, you will sometimes have to make changes or additions to an already exsisting model written in Fortran. This task is decidedly much easier if you acquire some knowledge of programming in Fortran. Moreover, valuable time may be lost if you have to aquire that knowledge later on, when you need to make changes to the model.

Next we remark that in the last 15 to 20 years or so Fortran is looked upon as an old-fashioned unstructured programming language by researchers and students alike in the field of Informatics. The reason is that earlier versions of Fortran lacked most of the features found in modern programming languages like C++, Java, etc. Especially the lack of object orientation has been the main drawback of Fortran. This is no longer true. Fortran 2003 and Fortran 2008 has all the modern features including Object Oriented Programming (OOP).

The most important reason why we still favor Fortran as a programming language in solving atmospheric and oceanographic problems on the computer, however, is the execution speed of the compiled program. In number crunching speed Fortran is much faster than C and C++. Tests

[1]Former scientific programmer at Department of Geosciences

show that an optimized Fortran program in some cases may run up to 30 percent faster than the equivalent C or C++ program. Thus for large and complex programs and codes with a runtime of weeks even a small increase in speed will reduce the overall time it takes to solve a problem. This is an important fact in the field of meteorology and oceanography since speed is everything when you are going to produce a forecast.

In addition we remark that laboratory experiments and field work are sometimes costly to perform. Computer simulations are less costly, and are therefore becoming increasingly more important as an addition to laboratory and field work.

A.2 Historical background

Fortran is an old programming language. Already in 1954 John W. Backus[2] and his team at IBM began developing the scientific programming language Fortran. It was first introduced in 1957 for a limited set of computer architectures. In a short time the language spread to other architectures. Since then it has been the most widely used programming language for solving numerical problems within natural sciences in general and in atmosphere and ocean science in particular.

The name Fortran is derived from **For**mula **Tran**slation, and as already alluded to is still the language of choice for fast numerical computations. In 1959 a new version, Fortran II, was introduced. This version was more advanced and among the new features was the ability to use complex numbers and splitting a program into subroutines. In the years to follow Fortran was further developed to become a programming language that was fairly easy to understand and well adapted to solve numerical problems.

In 1962 a new version, Fortran IV, emerged. Among its new features was the ability to read and write direct access files. In addition it introduced a new data type called `LOGICAL`. This was a Boolean data type with two states **true** or **false**. At the end of the seventies Fortran 77 was introduced. This version contained better loop and test structures. In 1992 Fortran 90, and shortly thereafter Fortan 95[3], was formally introduced as an ANSI/ISO standard. These versions turned Fortran into a modern programming language. Fortran 90/95 includes many of the features we expect from a modern programming languages. Finally Fortran 2003 is released that incorporates OOP with type extension and inheritance, polymorphism, dynamic type allocation and type-bound procedures.

[2]John Warner Backus (December 3, 1924 - March 17, 2007) was an American computer scientist. He directed the team that invented the first widely used high-level programming language (FORTRAN) and was the inventor of the Backus-Naur form, a widely used notation to define formal language syntax. He also did research in function-level programming and helped to popularize it. The IEEE awarded Backus the W.W. McDowell Award in 1967 for the development of FORTRAN. He received the National Medal of Science in 1975 (Source: Wikipedia).

[3]Fortran 95 is but a small extension of Fortran 90

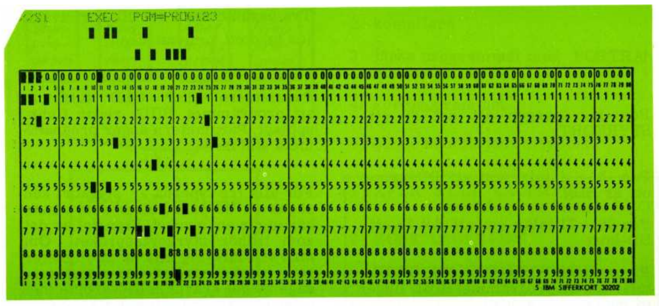


Figure A.1: The punched card

A.3 The Fortran syntax

We start by noticing that as all programming languages Fortran has its own syntax. Hence to start programming in Fortran a knowledge of its syntax is required. Fortran has, as other programming languages, a division of the code into variable declarations and instructions for manipulating the contents of the variables. An important difference between earlier Fortran 77 and Fortran 90/95/2003 is the way the code is written. In Fortran 77 the code is written in fixed form where each line of code is divided into 80 columns and each column has its own meaning.

This division has an historically background. In the 1960s and part of the 1970s the standard media for data input was the punched cards as displayed in Figure A.1. The cards were divided into 80 columns and it was therefore naturally to set the length of each line of code to 80 characters. In Table A.1 we provide an overview of the subdivision of the line of code. We emphasize that Fortran 77 is a subset of Fortran 2003 and all programs written in Fortran 77 can be compiled using a Fortran 2003 compiler.

In addition to the fixed code format from Fortran 77, Fortran 2003 also supports free format coding. This means that the division into columns are no longer necessary and the program code can be written in a more structured way which makes it more readable and easier to maintain. Today *the free format is the default* settings for the Fortran 90/95 and Fortran 2003 compilers.

Column number	Meaning
1	A character here means the line is a comment
2 - 5	Jump address and format number
6	A character here is a continuation from previous line
7 - 72	Program code
73 - 80	Comment

Table A.1: The Fortran 77 (F77) fixed format

A.3.1 Data types in Fortran

In earlier versions of Fortran four basic data types was included. These were `INTEGER` and `REAL` numbers, `LOGICAL` which is a boolean type and `CHARACTER` which represent the alphabet and other special non-numeric types. In Fortran 90/95 the `REAL` data type is split into the `REAL` and `COMPLEX` data types. In addition to this a derived data type can be used in Fortran 2003. A derived data type may contain one or more of the basic data types, other derived data types, and in addition procedures which is a part of the new OOP features in Fortran 2003.

INTEGER

An `INTEGER` datatype is identified with the reserved word `INTEGER`. It has a valid range which varies with the way it is declared and the architecture of the computer it is compiled on. When nothing else is given an `INTEGER` has a length of 32 bits on a typical workstation and can have a value from -2^{31} to 2^{30} and a 64 bit `INTEGER` with a minimum value from -2^{63} to a maximum value of 2^{62}.

REAL and COMPLEX

In the same manner a `REAL` number can be specified with various ranges and accuracies. A real number is identified with the reserved word `REAL` and can be declared with single or double precision. In Table A.2 the number of bits and minimum and maximum values are given.

Precision	Sign	Exponent	Significand	Max. value	Min. value
Single	1	8	23	2^{128}	2^{-126}
Double	1	11	52	2^{1024}	2^{-1022}

Table A.2: `REAL` numbers datatype in Fortran

A double precision real number are declared using the reserved words `DOUBLE PRECISION` or `REAL(KIND=8)`. The latter is the preferred declaration of a double precision real number.

An extension of `REAL` numbers are `COMPLEX` numbers with their real and imaginary parts. A `COMPLEX` number is identified with the reserved word `COMPLEX`. The real part can be extracted by the function `REAL()` and the imaginary part with the function `AIMAG()`. There is no need for writing explicit calculation functions for `COMPLEX` numbers like one has to do in C / C++ which lacks the `COMPLEX` data type.

LOGICAL

The Boolean datatype is identified by the reserved word `LOGICAL` and has only two values true or false. These values are identified with `.TRUE.` or `.FALSE.`. We note that the dot (period mark) at the beginning and end of the declaration is a necessary part of the syntax. To omit one or more dots will give a compilation error.

`CHARACTER`

The `CHARACTER` datatype is identified by the reserved word `CHARACTER` and contains letters and characters in order to represent data in a readable form. Legal characters are among others a to z, A to Z and some special characters +, -, *, / and =.

Derived data types

These are data types which are defined for special purposes. A derived data type is put together of components from one or more of the four basic data types, and also of other derived data types. A derived data type is always identified by the reserved word `TYPE name` as prefix and `END TYPE name` as postfix.

A.4 The structure of Fortran

A.4.1 Declaration of variables

In Fortran there are two ways to declare a variable. The first is called *implicit* declaration, and is inherited from the earliest versions of Fortran. The second is called *explicit* declaration, and is in accordance with other programming languages. Explicit declaration means that all variables has to be declared *before* any instructions occurs.

Implicit declaration on the other hand means that a variable is declared when needed by giving it a value anywhere in the source code, that is, even within the instructions. The data type is determined by the first letter in the variable name. An `INTEGER` is recognized by starting with the letters `I` to `N` and a `REAL` variable by the rest of the alphabet. We emphasize that no special characters are allowed in a variable name only the letters `A - Z`, the numbers `0 - 9` and the underscore character _. A variable cannot start with a number. In addition a `LOGICAL` variable is, in most compilers, identified by the letter `L`.

We underscore that as a general rule an implicit declaration is not a good way to program. For one it renders a code that is not easy to read. Secondly it easily introduces errors in a program due to typing errors. We therefore strongly recommend to *always use explicit declaration* of variables. To ensure that all variables must be declared is to include in the second line of all programs, functions and subroutines the keywords `IMPLICIT NONE`. This tells the compiler to check that all variables are declared. Finally we add that there are some variables which always have to be declared. These are arrays in one or more dimensions and character strings.

`INTEGER` numbers

We start by showing an example of how to declare an `INTEGER` in Fortran 95.

```
INTEGER               :: i ! Declaration of an INTEGER
                           ! length(32 bit)
INTEGER(KIND=2)       :: j ! Declaration of an INTEGER (16bit)
INTEGER(KIND=4)       :: k ! Declaration of an INTEGER (32bit)
INTEGER(KIND=8)       :: m ! Declaration of an INTEGER (64bit)
INTEGER,DIMENSION(100):: n ! Declaration of an INTEGER array
                           ! (100 elements)
```

We note that there are certain differences in the Fortran 77 and the Fortran 95 way of declaring variables. In Fortran 95 there is more to write, but this is offset by greater readability. Note that in Fortran 95 a comment can start anywhere on the code line, but must always be preceded by an exclamation (!) point.

`REAL` numbers

The `REAL` datatype in most compilers now conforms to the IEEE standard for floating point numbers. Declarations of single and double precision is declared as in the next example.

```
REAL                :: x ! Declaration of REAL
                         ! defaultlength (32bit)
REAL(KIND=8)        :: y ! Declaration of REAL
                         ! double precision (64 bit)
REAL,DIMENSION(200) :: z ! Declaration of REAL array
                         ! (200 elements)
```

`COMPLEX` numbers

Fortran has, unlike C/C++, an intrinsic datatype of complex numbers. Declaration of a `COMPLEX` variable in Fortran is as follows.

```
COMPLEX                :: a ! Complex number
COMPLEX,DIMENSION(100) :: b ! Array of complex numbers
                            ! (100 elements)
```

`LOGICAL` variables

Unlike `INTEGER` and `REAL` numbers a `LOGICAL` variable has only two values, `.TRUE.` or `.FALSE.`, and therefore uses a minimum of space. The number of bits a `LOGICAL` variable is using depends on the architecture and the compiler. It is possible to declare a single `LOGICAL` variable or an array of them. The following example shows a Fortran 90/95 declaration. In other programming languages the `LOGICAL` variable is often called a Boolean variable after George Boole the mathematician[4].

[4]George Boole (November 2, 1815 - December 8, 1864) was an English mathematician, philosopher and logician. He worked in the fields of differential equations and algebraic logic, and is now best known as the author of ”The Laws of Thought”, and as the inventor of the prototype of what is now called Boolean logic (source: Wikipedia).

```
LOGICAL                :: l1 ! Single LOGICAL variable
LOGICAL,DIMENSION(100) :: l2 ! Array of LOGICAL variables
                             ! (100 elements)
```

`CHARACTER` variables

Characters can either be declared as a single `CHARACTER` variable, a string of characters or an array of single characters or character strings.

```
CHARACTER                         :: c1 ! Single character
CHARACTER(LEN=80)                 :: c2 ! String of characters
CHARACTER,DIMENSION(10)           :: c3 ! Array of single
                                        ! characters
CHARACTER(LEN=80),DIMENSION(10) :: c4 ! Array of character
                                        ! strings (10 elements)
```

Derived data types

The Fortran 95 syntax for the declaration of a derived datatype can be like the one shown here.

```
TYPE derived
   ! Internal variables
   INTEGER           :: counter
   REAL              :: number
   LOGICAL           :: used
   CHARACTER(LEN=10) :: string
END TYPE derived
! A declaration of a variable of
! the new derived datatype
TYPE (derived)   :: my_type
```

The question arises, why use derived data types? The answer is is that sometimes it is desireable to group variables together to be able to refer to them under a common name. It is usually a good practice to select a name of the abstract data type to indicate the contents and area of use.

A.4.2 Instructions

There are two main types of instructions. One is for program control and the other is for assigning a value to a variable.

Instructions for program control

Instructions for program control can be split into three groups, one for loops, a second for tests (even though a loop usually have an implicit test), and a third for assigning values to variables and perform mathematical operations on the variables.

In Fortran all loops starts with the reserved word DO. The following piece of code shows a short example of a simple loop.

```
DO i = 1, 100
  !// Here instructions are performed 100 times
  !// before the loop is finished
END DO
```

The next example shows a loop with instructions. This loop is a non-terminating loop, where an IF-test inside the loop is used to exit the loop when the result of the test is true.

```
....
do
  a = a * sqrt(b) + c
  if (a > z) then
    !// Jump out of the loop
    exit
  end if
end do
```

This piece of code instructs the computer to give the variable a a value equal the sum of the square root of the variable b multiplied by a's prior value and a third variable c. When the value of a becomes greater then the value of the variable z the program transfers control to the next instruction following the loop. Note that we have assumed that all the variables are declared and initialized somewhere in the program *before* the loop as indicated by the code line .... appearing before the loop. The various Fortran instructions will be described in the example in Section A.5 below.

A.5 Sample programs

A.5.1 A daynumber converter

We start with a very simple program where the task is to calculate the daynumber of a specific date in the year. In this we assume that the year is a non-leapyear. We first write the program skeleton and then we fill in the necessary code to solve the problem.

```
PROGRAM daynumber
  implicit none
  ....
END PROGRAM daynumber
```

All Fortran programs begins with the reserved word PROGRAM and then the program name. In our case the program name is daynumber. The code line implicit none is, as alluded to above, almost mandatory or at least good programming practise. It appears to prevent the use of implicit declarations, which else is the default behavior of the Fortran compiler.

Next we declare some variables and constants which we will use to calculate the daynumber.

```
PROGRAM daynumber
  implicit none
  integer                :: counter
  integer,dimension(12) :: months
  integer                :: day, month
  integer                :: daynr
  ....
END PROGRAM daynumber
```

We have declared four integer variables, namely `counter`, `day`, `month` and `daynr`, and one integer array `months` with 12 elements. The variable `counter` is used to traverse the array to select the number of days in the months before the given month. The variables `day` and `month` hold the day and month. The variable `daynr` contains the result of the calculations.

Then we specify numbers for the constant integers `day` and `month`, and initialize the variable `daynr` and the array `months`,

```
PROGRAM daynumber
  implicit none
  ....
  day = 16
  month = 9
  daynr = 0
  months(:) = 31
  ....
END PROGRAM daynumber
```

Initializing scalar arrays are not difficult, but usually we would have to initialize each element of the array separately. Fortunately Fortran 95 and 2003 has a built in functionality which allow us to initialize a whole array with one value.

However, not all months contain 31 days. Thus the next step is to change the number of days in the months that differ from 31, that is, `months(2)` (February), `months(4)`(April) , `months(6)` (June) , `months(9)` (September), and `months(11)` (November).

```
PROGRAM daynumber
  implicit none
  ....
  months(2) = 28
  months(4) = 30
  months(6) = 30
  months(9) = 30
  months(11) = 30
  ....
END PROGRAM daynumber
```

The next step is to loop through all the elements in the array `months` up to the month minus one summing up the number of days in each month into the `daynr` variable. To arrive at our result we just further add the value from the variable day to the `daynr`. To display the result we use the command `PRINT *, daynr` that writes the result on the terminal.

```
PROGRAM daynumber
  implicit none
  ....
  DO counter = 1, month - 1
    daynr = daynr + months(counter)
  END DO
  daynr = daynr + day
  PRINT*, daynr
END PROGRAM daynumber
```

Compiling a program

In order to have an executable program we have to compile it. This requires that our sample program (or source code) resides in a file on the computer, say `daynr.f90`, where the extension indicates that the file contains a Fortran program written in Fortran 90. The compilation process takes the file with the source code and creates a binary file linked in with the necessary system libraries so we may run the program on the computer. The binary file contains our program in machine specific assembley language, that is, instructions written in machine language. We use an open source[5] compiler called `gfortran`. The command line for compiling our program is simply

```
 gfortran -o daynr daynr.f90
```

where `gfortran` is the name of the compiler[6]. The argument `-o` means that the next argument to the compiler is the name of the executable program. The last argument is the name of the file containing our source code. You may also simply write

```
 gfortran daynr.f90
```

In this case the executable program by default is given the name `a.out`. To run the compiled program we may use the command `./daynr` (or `./a.out`) in the terminal window.

The resulting output from our sample program with the `month` = 9 and the `day` = 16 is **259**. You can use a calculator and perform the calculations by hand to check that the result is correct.

Doing this simple program we have learned to *never use implicit declarations of variables* which is very important. There is a story from the seventies about 10 implicit declarations where a typing error created an uninitialized variable causing a NASA rocket launch to fail and the rocket had to be destroyed before it could cause sever damage.

[5]Open source means that the compiler may be downloaded and used free of charge

[6]Note that there are other compilers.

Exercises

1. Use the code in this section, fill in what is missing and save the source code in a file. Compile the code and run it to check that daynumber in a non-leapyear for September 16 is indeed 259.

2. Given the radius of a circle write a program calculating the length of the circumference of the circle, compile and run the program and check that the result is correct.

3. Given a radius of a circle write a program calculating the area inside of the circle, compile and run the program and check that the result is correct.

4. Given a radius of a sphere write a program calculating the volume of the sphere, compile and run the program and check that the result is correct.

A.5.2 A temperature converter

We now develop a program that converts a temperature given in degreees Fahrenheit to degrees Celsius (or centigrades). To see the results we must print the two temperatures in the terminal window. The formulae for the conversions are

$$C = \frac{5}{9}(F - 32) \quad \text{or vice versa} \quad F = \frac{9}{5}C + 32, \tag{A.1}$$

where F represents the temperature in Fahrenheit and C the temperature in Celsius or centigrades.

To proceed we need two floating point variables, one to hold the temperature in Fahrenheit, say `F`, and a second to hold the temperature in Celsius, say `C`. To declare them as floating point variables we use the `REAL` keyword[7]. Thus we must include the following piece of code

```
   ....
   !// The Fahrenheit variable
   REAL :: F
   !// The Centigrade variable
   REAL :: C
   ....
```

Note that, in contrast to most other languages, the Fortran language is case insensitive. That means that a variable or function name is the same whether it is written with uppercase or lowercase letters. Moreover, beginning with the Fortran 90 version we separate the variable type from the variable name with a double colon. Furthermore, an exclamation sign is used to signal the compiler that the rest of the line is a comment. In order to make the code more readable we recommend to include an additional double slash before writing the comment as shown in the above example. As we shall see later a program that is easy to read is also easy to understand and makes it easier to find and correct errors.

[7]In other languages the keyword `float` is used to declare a floating point variable.

The next step is to initialize the Fahrenheit variable and perform the conversion according to the formula. The whole program may be something like,

```
PROGRAM f2c_simple
  IMPLICIT NONE
  !// Declare variables
  REAL :: F    ! Fahrenheit variable (floating point)
  REAL :: C    ! Centigrade variable C (floating point)
  !// Assigne a value to F as a constant number
  !// Note the decimal point
  F = 75.
  !// Perform the calculations
  C = (F - 32.)*5./9.
  !// Write the result to the terminal window
  PRINT *, C
END PROGRAM f2c_simple
```

Note that we avoid using the Fortran default of implicit declarations of variables by writing `IMPLICIT NONE` in the second code line. To tell the compiler that the constant value 75 is a real number we use a decimal point as part of the number. If we omit the dot the compiler will assume that it is an integer and in some cases the calculations will be wrong. Finally note the use of parenthesis to perform the calculations in the proper sequence. The statement or command `PRINT *, C` writes the temperature in degrees Celsius to the terminal window.

As in the former example we have to compile the program to get a binary executable program file. There are several commercial Fortran compilers, but we will use the open source GNU Fortran compiler called `gfortran` to compile our program. Let us assume that we have written our program into a file named `f2c_simple.f90`. We may then compile it using

```
 gfortran -o f2c f2c_simple.f90
```

where again the `-o` option signals to the compiler that the next argument is the name of the executable program and the last argument is the name of the file with the source code. In this case the binary program file called `f2c` is created, which may be run by typing `./f2c` in the terminal window. This is exactly as we did for the former sample progran (Section A.5.1). All Fortran programs are compiled and run this way.

A.5.3 A more user friendly version of the converter program

The above program is note very user friendly. Everytime we would like to convert a new temperature in degrees Fahrenheit to degress Celsius we have to change the source code, that is, specify a new `F`, recompile and rerun the program. To avoid this we may add to our program a user interface asking us to give a temperature in degrees Fahrenheit.

To accomplish this we must add some code lines for communication in the form of text strings. In the example below we first declare and assign two text strings `prompt1` and `prompt2`. The first text string, declared as `prompt1`, asks us whether our input is in Fahrenheit or Celsius, while the second, `prompt2`, asks us to enter the temperature we like to convert. To declare the text strings we use the data type `CHARACTER`. Thus our program begins the following code lines

```
!////////////////////////////////////////////////////////
!//
!// f2c.f90
!//
!// Program to convert from Fahrenheit to
!// Celsius or vice versa
!////////////////////////////////////////////////////////
PROGRAM f2c
  IMPLICIT NONE
  REAL :: F       !// Temperature in Fahrenheit
  REAL :: C       !// Temperature in degree Celsius
  !// Character strings to hold the prompts for
  !// communicating with the user
  CHARACTER(LEN=80) :: prompt1, prompt2
  !// A single character to hold the answer
  !// which is either F or C
  CHARACTER :: answer
  !// Assign a value to the prompt1
  prompt1 = 'Enter F for Fahrenheit or C for Celsius'
  !// Assign a value to the prompt2
  prompt2 = 'Enter a temperature'
  ....
```

The declaration `CHARACTER(LEN=80)` means that we have allocated a space for a text string up to 80 characters long. The character `answer` is unassigned, but is used to hold a single character variable which is the answer to the question `prompt1`, that is, `answer` is used to hold the characters (`F` or `C`) according to our answer to `prompt1`.

Note also that we have added some comments above the `PROGRAM` line. It gives the name of the file containing the source code, and also a brief description of the programs purpose. This makes the code more readable and easier to understand and is recommendable even for small programs like this one.

Then we continue

```
  ....
  !// Print the contents to the terminal window
  !// without trailing blank characters
  PRINT *, TRIM(prompt1)
  !// Read the input from the keyboard
  READ(*,*) answer
  !// Print the contents to the terminal window
  !// without trailing blank characters
  PRINT *, TRIM(prompt2)
  ....
```

This part prints the assigned value of `prompt1` to the terminal window, then read our input and put it in `answer`. Next it prints the `prompt2` to the terminal window and waits for our input. The use of the keyword TRIM tells the compiler to print out the text without printing the trailing space characters. Note that we use the `READ` command to read in our answer to `prompt1`. The construct `READ(*,*)` means we read the input from the keyboard with default formatting into the receiving variable.

The next program steps then read our input, then branches out according to our input regarding `answer`, performs the conversion and prints the result before it ends the program in a proper way. Thus

```
  ....
  !// Is the temperature given in Fahrenheit?
  IF(answer .EQ. 'F') THEN
    !// Yes, read input into the F variable
    READ(*,*) F
    !// Convert from Fahrenheit to Celsius
    C = (F - 32) * (5. / 9.)
    !// Print the result to the screen
    PRINT *, C
  ELSE
    !// No, read input into the C variable
    READ(*,*) C
    !// convert from Celsius to Fahrenheit
    F = (C * 9. / 5.) + 32
    !// Print the result to the screen
    PRINT *, F
  END IF
END PROGRAM f2c
```

A.5.4 Variable types, arrays, loops and memory allocation

Most often than not atmosphere and ocean variables are stored in large files where the variables are stored in multiple dimensioned arrays. To illustrate we first study how we store data in a one dimensional array[8].

For this purpose we construct a program that calculates the Fibonacci sequence[9]. The sequence consists of a series of integers, the so called Fibonacci numbers, which require us to store them in a one dimensional array or vector of integers. We first recall that the formula for calculating the Fibonacci sequence is

$$F_{j-1} = F_{j-2} + F_{j-3}, j = 3(1)n \tag{A.2}$$

where the two first numbers are $F_0 = 0$ ($j = 1$) and $F_1 = 1$ ($j = 2$).

First we need an array of integers to hold the numbers. We will construct the program so that the user may choose the lenth of the sequence. In that case we have to include a user interface like we used in the temperature conversion program asking for the length of the sequence. Thus the length of this array (or vector) is not known in advance. So we will have to use a so called allocatable array where we allocate the needed space at runtime. In addition we need an index variable and a status variable where the first one is for accessing the various elements of the array and a status variable to check the result of the allocation. Consequently the first part of the code reads

[8] A one dimensional array is often referred to as a vector

[9] Leonardo Pisano Bigollo (ca. 1170 - ca. 1250) - known as Fibonacci, and also Leonardo of Pisa, Leonardo Pisano, Leonardo Bonacci, Leonardo Fibonacci - was an Italian mathematician, considered by some ”the most talented western mathematician of the Middle Ages”. Fibonacci is best known to the modern world for the spreading of the Hindu-Arabic numeral system in Europe, primarily through his composition in 1202 of Liber Abaci (Book of Calculation), and for a number sequence named the Fibonacci sequence (or numbers) after him, which he did not discover but used as an example in the Liber Abaci.

```
!/////////////////////////////////////////////////////////////
!//
!// fibonacci.f90
!//
!// Program to display the Fibonacci
!// sequence from 1 to n
!//
!/////////////////////////////////////////////////////////////
PROGRAM fibonacci
  IMPLICIT NONE
!// Variable declarations
 !// Declaring integer variables
  !// first a counter variable
  INTEGER :: i
  !// then the length of Fibonacci sequence
  INTEGER :: n
  !// and finally a status variable (if errors)
  INTEGER :: res
  !// Declare an array to hold the Fibonacci sequence
  !// with unknown length at compilation time
  INTEGER, ALLOCATABLE, DIMENSION(:) :: sequence
  ....
```

To declare an array with unknown length, here `sequence`, we use the keywords `INTEGER, ALLOCATABLE, DIMENSION(:)` to declare it. Note that we replace the length of the array with a colon `:`. If we know the length of the array in advance we simply use the construct `INTEGER, DIMENSION(100)` for an array containing 100 elements.

This done we continue with prompting for the finite number in the infinite Fibonacci sequence we will calculate, a number we later use to allocate space for the Fibonacci numbers in the sequence. Thus we have to declare a prompting character string of some length, say 80, and then assign values to it. Note that the text string is shorter than 80 characters long so we insert a line of code counting the actual number of characters. Furthermore, we push the prompt to the terminal window. Thus the program continues

```
  ....
  !// Declare the length of the prompt to ask
  !// for length of the Fibonacci sequence
  CHARACTER(LEN=80) :: prompt
  !// Assign value to the prompt
  prompt = 'Enter the length of the sequence: '
  !// Get the number of non blank characters in prompt
  i = LEN(TRIM(prompt))
  !// Display the prompt asking for the length suppressing
  !// the line feed
  WRITE(*,FMT='(A)',ADVANCE='NO') prompt(1:i+1)
  ....
```

Since use of the keyword `PRINT *` always prints the text to the terminal window with a linefeed as the last operation, we have replaced it with the command `WRITE(*,FMT='(A)ADVANCE='NO')`. Use of of the `WRITE` command makes it possible to avoid or supress the linefeed by adding the formatting code in the `WRITE` command as shown above.

Now we are ready to read the input from the terminal window and then perform the calculations. Hence the rest of the program reads

```
  ....
  !// Read the keyboard input
  READ(*,*) n
  !// Allocate space for the sequence
  ALLOCATE(sequence(n), STAT=res)
  !// Test the value of the res variable for errors
  IF(res /= 0) THEN
    !// We have an error. Print a message and stop the program
    PRINT *, 'Error in allocating space, status: ', res
    STOP
  END IF
  !// Initialize the two first elements in the sequence
  sequence(1) = 0
  sequence(2) = 1
  !// Loop and calculate the Fibonacci numbers
  DO i = 3, n
    sequence(i) = sequence(i-1) + sequence(i-2)
  END DO
  !// Print the sequence to the screen
  PRINT *, sequence
END PROGRAM fibonacci
```

Note that once we read in the variable `n` we were able to allocate space for the Fibonacci array `sequence` using the construct: `ALLOCATE(sequence(n), STAT=res)`. Then we make an if test using the integer `res` to check if the allocation is OK. If it is different from zero which means that the allocation failed we stop the program. If not we continue by first specifying the first two numbers in the Fibonacci sequence. Next we start a loop to calculate the next integers in the sequence in accord with the formula given in (A.2). Note that since we have specified the two first numbers in the sequence the loop starts with an index variable equal 3, in accord with (A.2). Finally we added the code line `PRINT *, sequence` at the end to display the contents of the sequence in the terminal window (unformatted).

A.5.5 File input/output or I/O

Commonly the input of data we use in our programs are stored in files. These may be numbers generated through the output of another program or observations produced by instruments sensors in one way or another. In either case they ar usually available to us on a file stored on a computer somewhere or residing on a memory device of some sort.

In the example to follow we learn how to read data from a file into an array, perform some operation on the data set and write the result to a new file. In our case we assume that we know in advance the length of the array, say 7 elements long. So we start the program like this

```
!//////////////////////////////////////////////////////////////
!//
!// f2c_file.f90
!//
!// Program to calculate the degree Celsius from a
!// file containing seven observations of temperatures
!// in Fahrenheit
!//
!//////////////////////////////////////////////////////////////
PROGRAM f2c_file
  IMPLICIT NONE
  !// Declare a static variable to hold the
  !// length of the arrays
  INTEGER, PARAMETER :: n = 7
  ....
```

Here `n` is declared using the keyword `PARAMETER` to specify that it is constant or static, that is, unchanged at runtime. Then we continue by declaring the the arrays that will hold the temperatures in Fahrenheit and Celsius, and a counter variable, that is,

```
  ....
  !// Declare the arrays for the temperatures,
  !// and a counter variable
  REAL,DIMENSION(n)  :: F             !// Fahrenheit
  REAL,DIMENSION(n)  :: C             !// Celsius
  INTEGER            :: j             !// Counter variable
  ....
```

In addition to this we need two character strings to hold the names of the input and output files, that is,

```
  ....
  !// Declare character strings to hold the filenames
  CHARACTER(LEN=80)  :: infile  !// Holds the Fahrenheit
                                !// temperatures
  CHARACTER(LEN=80)  :: outfile !// Holds the results of
                                !// the conversion to
                                !// degree Celsius
  ....
```

Further we need two parameters to hold the unit numbers which we use to reference the input and output files. We also need to specify the names of the input and output files so we can recognize them in our directory once the operation is completed. The reference numbers are integers. In this regard we also need to declare a status variabel as we did in the last example above. Thus the program continues

```
  ....
  !// Declare constant values for the Logical Unit Number for
  !// referencing the files for opening, reading and writing
  INTEGER, PARAMETER :: ilun = 10
  INTEGER, PARAMETER :: olun = 11
  !// A status variable to hold the result of file
  !// operations
  INTEGER :: res
  !// Assign an input filename for temperatures in Fahrenheit
  infile = "fahrenheit.txt"
  !// Assign an output filename for temperatures in Celsius
  outfile = "celsius.txt"
  ....
```

Before we can access the contents of the output file we have to open it using the unit number we declared and its filename, that is, `infile`. We also test whether the opening was successful. To effectuate this we continue with the following statements

```
  ....
  !// Open the input file
  OPEN(UNIT=ilun,FILE=infile,FORM="FORMATTED",IOSTAT=res)
  !// Test if the operation was successful
  IF(res /= 0) THEN
    !// No, an error occurred, print a message to
    !// the screen
    PRINT *, "Error in opening file, status: ", res
    !// Stop the program
    STOP
  END IF
  ....
```

We are now in a position to read the contents of the input file into the array `F`. This is carried out by using a loop that runs from `1` to the length of the array, in this example `n`. To fulfill this we make use of the `READ` statement which is similar to the `OPEN` statement. Thus the program continues

```
  ....
  !// Loop and read each line of the file
  DO j = 1, n
    !// Read the current line
    READ(UNIT=ilun,FMT='(F4.1,X,F4.1)',IOSTAT=res) F(j)
    !// Was the read successful?
    IF(res /= 0) THEN
      !// No, test if we have reached End Of File (EOF)
      IF(res /= -1) THEN
        !// No, an error has occurred, print a message
        PRINT *, "Error in reading file, status: ", res
        !// Close the file
        CLOSE(UNIT=ilun)
        !// Stop the program
        STOP
      END IF
    END IF
  END DO
  ....
```

Note that the `infile` is formatted. Thus we need to know the format of the data contained in the `infile`, that is, the file `fahrenheit.txt`, and make sure that the argument specified in the argument `FMT` is an exact match of the format in `fahrenheit.txt`. We use the keyword/argument pair `FMT='(F4.1,X,F4.1)'` to tell the computer that we have a floating point number with 4 digits including the decimal point and one decimal, a space character and then another floating point number like the first.

We may now proceed to convert from Fahrenheit to Celsius, and to store the result into the second array, that is, `C`. To see the result in the terminal window we also add a `PRINT` statement. This is accomplished by adding the code lines

```
....
!// Loop and convert from Fahrenheit to Celsius
DO j = 1, n
  C(j) = (F(j) - 32) * 5. / 9.
END DO
!// Print the temperatures to the screen
DO j = 1, n
  PRINT *, " Degrees Farenheit ", F(j), &
           " Degrees Celsius ", C(j)
END DO
....
```

Once the conversion is completed we may proceed to write the contents of the array `C` to the output file we named `outfile` and gave the reference number `olun = 11`. To enable writing to the output file we first have to ensure that it is open. We do this exactly as we did for the input file. Thus we proceed

```
....
!// Open the output file
OPEN(UNIT=olun,FILE=outfile,FORM="FORMATTED",IOSTAT=res)
!// Test if the operation was successful
IF(res /= 0) THEN
  !// No, an error occurred, print a message to
  !// the screen
  PRINT *, "Error in opening output file, status: ", res
  !// Stop the program
 STOP
END IF
....
```

Note that we specfically ensured that also the outfile is formatted by including the argument `FORM="FORMATTED"` in the `OPEN` statement.

We may then write the result of the conversion to the output file, by continuing with the following code lines

```
  ....
  DO j = 1, n
    WRITE(UNIT=olun,FMT='(F4.1,A1,F4.1)',IOSTAT=res) C(j)
    !// Test if the operation was successful
    IF(res /= 0) THEN
      !// No, an error occurred, print a message to
      !// the screen
      PRINT *, "Error in writing file, status: ", res
      !// Exit the loop
      EXIT
    END IF
  END DO
  ....
```

Finally we close the input and output files and terminate program, that is,

```
  ....
  !// Close the input file
  CLOSE(UNIT=ilun)
  !// Close the output file
  CLOSE(UNIT=olun)
END PROGRAM f2c_file
```

Note that when using the `OPEN` function we made use of the respective unit numbers `ilun = 10` and `olun = 11`. We also provided the filename as part of the `FILE` argument. Finally note that the result of the call to the `OPEN` is returned in the `IOSTAT=res` keyword/argument pair. We underscore that the same procedure is used in reading from files.

Note also that we in this example made use of formatted files. This entails that the file content may be displayed in the terminal window. This is in contrast to binary files which is not very meaningful to us. To visualize this let us first construct a formatted file containg two pairs of seven temperatures in Fahrenheit in two columns formatted following the keyword/argument `FMT='FMT='(F4.1,X,F4.1)'`, that is,

```
68.2 65.5
69.6 63.7
73.2 66.0
75.0 68.0
77.5 70.2
79.2 71.4
91.2 73.2
```

The binary file in contrast looks like this

```
The binary file looks like this:
.....
^@H<8D><BC>$<D0>^@^@^@<BE>
^@^@^@<B9><B8>_N^@H<C7>D$0P^@^@^@<BA>^C<FF><84>
^C3<C0>H<C7>D$8<80><DC>s^@L<8D>D$0H<C7>D$@
^@^@^@H<C7>D$H<E0>'N^@H<C7>D$P^D^@^@^@<E8>O<C3>
.....
This is the end of the binary file$
```

As is obvious the binary format is unreadable as is unless you use a program that translates the binary information into a textfile written in so called ASCII format. ASCII is short for American Standard Code for Information Interchange, and is the format used by formatted files in Fortran.

A.5.6 Multidimensional arrays

In the field of Meteorology and Oceanography the data sets we operate on is usually in four dimensions, three in space space and one in time. This consequence is that we have to declare matrices in four dimensions to be able to store the data. The next example shows how we may use a two dimensional matrix to store a set of temperatures in degree Fahrenheit from two measuring stations, read it in from a file, convert the temperatures into Celsius and write them to a file.

The program will be very similar of course to the previous example with the exception that we here operate on a matrix (vector, array) with two dimensions. This is possible by use of nested loops. The complete code is[10]

[10]We have split it into parts here to avoid open spaces

```
!////////////////////////////////////////////////////////////
!//
!// f2c_advanced.f90
!//
!// A program calculating degree Celsius from Fahrenheit
!// from two measuring stations
!//
!////////////////////////////////////////////////////////////
PROGRAM f2c_advanced
  IMPLICIT NONE
  !// Declare a static variable to hold the
  !// dimension of the vectors or arrays
  INTEGER, PARAMETER :: m = 7
  INTEGER, PARAMETER :: n = 2
  !// Declare the arrays for the temperatures in
  !// Fahrenheit and Celsius, and counter variables
  REAL,DIMENSION(m,n) :: F
  REAL,DIMENSION(m,n) :: C
  INTEGER :: j, k
  ....
```

```
....
!// Character strings to hold the filenames
CHARACTER(LEN=80) :: F_file
CHARACTER(LEN=80) :: C_file
!// Constant values for the Logical Unit Number
!// for referencing the files for opening, reading
!// and writing
INTEGER, PARAMETER :: ilun = 10
INTEGER, PARAMETER :: olun = 11
!// A status variable to hold the result of file
!// operations
INTEGER :: res
!// Assign the input filename for Fahrenheit temp.
infile = "fahrenheit.txt"
!// Assign the output filename for Centigrade temp.
outfile = "celsius.txt"
!// Open the Fahrenheit input file
OPEN(UNIT=ilun,FILE=infile,FORM="FORMATTED",IOSTAT=res)
!// Test if the operation was successful
IF(res /= 0) THEN
  !// No, an error occurred, print a message to
  !// the screen
  PRINT *, "Error in opening file, status: ", res
  !// Stop the program
  STOP
END IF
!// Loop and read each line of the file
DO j = 1, m
  !// Read the current line
  READ(UNIT=ilun,FMT='(F4.1,X,F4.1)',IOSTAT=res) F(j,1), F(j,2)
  !// Successfully read?
  IF(res /= 0) THEN
    !// No, test if we have reached End Of File (EOF)
    IF(res /= -1) THEN
      !// No, an error has occurred, print a message
      PRINT *, "Error in reading file, status: ", res
      !// Close the file
      CLOSE(UNIT=ilun)
      !// Stop the program
      STOP
    END IF
  END IF
END DO
....
```

```
  ....
  !// Close the input file
  CLOSE(UNIT=ilun)
  !// Loop and convert from Fahrenheit to Centigrade
  DO k = 1, n
    DO j = 1, m
      C(j,k) = (F(j,k) - 32.) * 5./9.
    END DO
  END DO
  !// Open the Centigrade output file
  OPEN(UNIT=olun,FILE=centigradefile,FORM="FORMATTED",IOSTAT=res)
  !// Test if the operation was successful
  IF(res /= 0) THEN
    !// No, an error occurred, print a message to
    !// the screen
    PRINT *, "Error in opening output file, status: ", res
    !// Stop the program
    STOP
  END IF
  DO j = 1, m
    WRITE(UNIT=olun,FMT='(F4.1,A1,F4.1)',IOSTAT=res) C(i,1), &
    ' ', C(i,2)
    !// Test if the operation was successful
    IF(res /= 0) THEN
      !// No, an error occurred, print a message to
      !// the terminal window
      PRINT *, "Error in writing file, status: ", res
      !// Exit the loop
      EXIT
    END IF
  END DO
  !// Close the output file
  CLOSE(UNIT=olun)
END PROGRAM f2c_advanced
```

Here we have made use of nested loops, that is, a loop within the loop. It is important for the efficiency of the program to know how Fortran accesses a matrix. Fortran accesses a matrix columnwise[11]. Thus all the row elements in a column is accessed before the row elements in the next column. Therefore as a rule of thumb we *always let the first index be the innermost loop* in Fortran.

[11]This is also true for Matlab

A.5.7 Functions and Subroutines

It is not a good programming practice to write one long program that includes all the necessary code in one single source file. First of all it makes the program hard to understand. Secondly it is also makes it difficult to maintain. Consequently it is common to break the program into smaller parts or subprograms that is called into action by the main program. In Fortran we call these subprograms `FUNCTION`s or `SUBROUTINE`s.

In what follows we break the former program of the previous section into a main program and a subprogram. The task of the subprogram is simply to do the conversion from Fahrenheit to Celsius, or vice versa. This may be done either bey use a function or a subroutine.

Functions

We start with the `FUNCTION`. In the former program the conversion was either from Fahrenheit to Celsius or from Celsius to Fahrenheit. We therefore need two functions, one that converts from Fahrenheit to Celsius, which we call `f2c(arg)`, and a second that converts from Celsius to Fahrenheit, which we call `c2f(arg)`. First we program `f2c(arg)`,

```
FUNCTION f2c(F) RESULT(C)
  IMPLICIT NONE
!// The input argument which is read only
  REAL(KIND=8), INTENT(IN) :: F
!// The result from the calculations
  REAL(KIND=8) :: C
!// Perform the calculation
  C = (F -32) * 5./9.
!// Return the result
  RETURN
END FUNCTION f2c
```

and then `c2f(arg)`,

```
FUNCTION c2f(C) RESULT(F)
  IMPLICIT NONE
!// The input argument which is read only
  REAL(KIND=8), INTENT(IN) :: C
!// The result from the calculations
  REAL(KIND=8) :: F
!// Perform the calculation
  F = (C * 9. / 5.) + 32
!// Return the result
  RETURN
END FUNCTION c2f
```

In contrast to the way we write mathematical functions, Fortran 90 - 2003 adds the keyword `RESULT(arg)` where the datatype of the argument states what kind of function it is. In Fortran 77 and older versions the syntax was `REAL FUNCTION f2c(arg)`. Note the use of `INTENT(IN)` for the input argument to the functions. This is to prevent accidental overwriting of the argument since Fortran function and subroutine arguments are always called by reference and not by value. The only thing we need to do in the main program with the exception of the user interface is to replace the formula for the conversion with a call to the respective functions. Note that in order to use other the Fortran intrinsic functions we have to declare them as external functions of the correct type. If we omit the external attribute the compiler will flag en error and the compilation will be aborted. The complete main program is shown below.

```
!////////////////////////////////////////////////////////////
!//
!// array2.f90
!//
!// A program converting from Fahrenheit to Celsius
!// and vice versa
!//
!////////////////////////////////////////////////////////////
PROGRAM array2
  IMPLICIT NONE
  !// Declare everything .....
  !// 1. Arrays for the temperatures
  REAL,DIMENSION(7,2) :: F   ! Fahrenheit
  REAL,DIMENSION(7,2) :: C   ! Celsius
  !// 2. Index variables
  INTEGER :: i, j
  !// 3. Character strings to hold the filenames
  CHARACTER(LEN=80) :: fahrenheitfile
  CHARACTER(LEN=80) :: centigradefile
  !// 3. Constant values for the Logical Unit Number (lun)
  !//    for referencing the files for opening, reading
  !//    and writing
  INTEGER, PARAMETER :: ilun = 10
  INTEGER, PARAMETER :: olun = 11
  !// 4. A status variable to hold the result of file
  !//    operations
  INTEGER :: res
  !// 5. External function(s)
  REAL, EXTERNAL :: f2c
  REAL, EXTERNAL :: c2f
  !// 6. A character string for a prompt
  CHARACTER(LEN=80) :: prompt
  CHARACTER :: answer
  !// ..... End declarations
  !// Assign filenames for temperatures
  fahrenheitfile = "fahrenheit.txt"
  centigradefile = "centigrade.txt"
  !// Ask if we are to convert from Fahrenheit to
  !// Celsius or vice versa
  ....
```

The program continues

```
  ....
  prompt = ’Convert from Fahrenheit to Centigrade (F/C)?’
  PRINT *, TRIM(prompt)
  READ(*,*) answer
  !// Check if the answer is F or f for Fahrenheit
  IF(answer .EQ. ’F’ .OR. answer .EQ. ’f’) THEN
  !// Yes, open the Fahrenheit input file
    OPEN(UNIT=ilun,FILE=fahrenheitfile,FORM="FORMATTED", &
    IOSTAT=res)
    !// Test if the operation was successful
    IF(res /= 0) THEN
    !// No, an error occurred, print a message to
    !// the screen
      PRINT *, "Error in opening file, status: ", res
    !// Stop the program
      STOP
    END IF
    !// Loop and read each line of the file
    DO i = 1, 7
      !// Read the current line
      READ(UNIT=ilun,FMT=’(F4.1,X,F4.1)’,IOSTAT=res) &
      F(i,1), F(i,2)
      !// Was the read successful?
      IF(res /= 0) THEN
        !// No, test if we have reached End Of File (EOF)
        IF(res /= -1) THEN
          !// No, an error has occurred, print a message
          PRINT *, "Error in reading file, status: ", res
          !// Close the file
          CLOSE(UNIT=ilun)
          !// Stop the program
          STOP
        END IF
      END IF
    END DO
  ELSE
    !// No, open the Celsius input file
    OPEN(UNIT=ilun,FILE=centigradefile,FORM="FORMATTED", &
    IOSTAT=res)
    ....
```

The program continues

```
    ....
    !// Test if the operation was successful
    IF(res /= 0) THEN
      !// No, an error occurred, print a message to
      !// the screen
      PRINT *, "Error in opening file, status: ", res
      !// Stop the program
      STOP
    END IF
    !// Loop and read each line of the file
    DO i = 1, 7
      !// Read the current line
      READ(UNIT=ilun,FMT='(F4.1,X,F4.1)',IOSTAT=res) &
      C(i,1), C(i,2)
      !// Was the read successful?
      IF(res /= 0) THEN
        !// No, test if we have reached End Of File (EOF)
        IF(res /= -1) THEN
          !// No, an error has occurred, print a message
          PRINT *, "Error in reading file, status: ", res
          !// Close the file
          CLOSE(UNIT=ilun)
          !// Stop the program
          STOP
        END IF
      END IF
    END DO
  END IF
  !// Close the input file
  CLOSE(UNIT=ilun)
  !// Which way to convert ?
  IF(answer .EQ. 'F' .OR. answer .EQ. 'f') THEN
    !// Loop and convert from Fahrenheit to Celsius
    DO j = 1, 2
      DO i = 1, 7
        C(i,j) = f2c(F(i,j))
      END DO
    END DO
  ELSE
    ....
```

The program continues

```
  ....
!// Loop and convert from Celsisus to Fahrenheit
DO j = 1, 2
  DO i = 1, 7
    F(i,j) = c2f(C(i,j))
  END DO
END DO
END IF
!// Which file to write to ?
IF(answer .EQ. ’F’ .OR. answer .EQ. ’f’) THEN
  !// Open the Centigrade output file
  OPEN(UNIT=olun,FILE=centigradefile,FORM="FORMATTED", &
  IOSTAT=res)
  !// Test if the operation was successful
  IF(res /= 0) THEN
    !// No, an error occurred, print message to the screen
    PRINT *, "Error in opening output file, status: ", res
    !// Stop the program
    STOP
  END IF
  DO i = 1, 7
    WRITE(UNIT=olun,FMT=’(F4.1,A1,F4.1)’,IOSTAT=res) &
    C(i,1), ’ ’, C(i,2)
    !// Test if the operation was successful
    IF(res /= 0) THEN
      !// No, an error occurred, print message to the screen
      PRINT *, "Error in writing file, status: ", res
      !// Exit the loop
      EXIT
    END IF
  END DO
ELSE
  !// Open the Fahrenheit output file
  OPEN(UNIT=olun,FILE=fahrenheitfile,FORM="FORMATTED", &
  IOSTAT=res)
  !// Test if the operation was successful
  ....
```

The program continues

```
    ....
    IF(res /= 0) THEN
      !// No, an error occurred, print message to the screen
      PRINT *, "Error in opening output file, status: ", res
      !// Stop the program
      STOP
    END IF
    DO i = 1, 7
      WRITE(UNIT=olun,FMT='(F4.1,A1,F4.1)',IOSTAT=res) &
      F(i,1), ' ', F(i,2)
      !// Test if the operation was successful
      IF(res /= 0) THEN
        !// No, an error occurred, print message to the screen
        PRINT *, "Error in writing file, status: ", res
        !// Exit the loop
        EXIT
      END IF
    END DO
  END IF
  !// Close the output file
  CLOSE(UNIT=olun)
END PROGRAM array2
```

In contrast to the functions a subroutine does not return a value and is the same as a void function in other languages. We may replace the functions f2c() and c2f() with corresponding subroutines which can look like this:

```
!///////////////////////////////////////////////////////////
!//
!// SUBROUTINE f2c(F,C)
!//
!// Called from program array2.f90
!//
!///////////////////////////////////////////////////////////
SUBROUTINE f2c(F,C)
  IMPLICIT NONE
  !// The input argument which is read only
  REAL(KIND=8), INTENT(IN) :: F
  !// The result of the conversion which is
  !// write only
  REAL(KIND=8), INTENT(OUT) :: C
    ....
```

The program continues

```
    ....
  !// Perform the conversion
  C = (F -32) * 5./9.
  !// Return the result through the second argument
  RETURN
END SUBROUTINE f2c
```

and

```
!////////////////////////////////////////////////////////////
!//
!// SUBROUTINE c2f(C,F)
!//
!// Called from program array2.f90
!//
!////////////////////////////////////////////////////////////
SUBROUTINE c2f(C,F)
  IMPLICIT NONE
  !// The input argument which is read only
  REAL(KIND=8), INTENT(IN) :: C
  !// The result of the conversion which is
  REAL(KIND=8), INTENT(OUT) :: F
  !// Perform the conversion
  F = (C * 9. / 5.) + 32
  !// Return the result through the second argument
  RETURN
END SUBROUTINE c2f
```

The calling from the main program is like this:

```
......
  !// Loop and perform the conversion using
  !// nested loops
  DO j = 1, 2
    DO i = 1, 7
    !// Call the subroutine with the current element
    !// of the farenheit array as the first argument to
    !// the soubroutine and the current element of the
    !// centigrade array as the second argument.
      CALL f2c(farenheit(i,j),centigrade(i,j))
    END DO
  END DO
```

and

```
......
  !// Loop and perform the conversion using
  !// nested loops
  DO j = 1, 2
    DO i = 1, 7
    !// Call the subroutine with the current element
    !// of the farenheit array as the first argument to
    !// the soubroutine and the current element of the
    !// centigrade array as the second argument.
      CALL c2f(centigrade(i,j).farenheit(i,j))
    END DO
  END DO
```

A subroutine shall not be declared as external, only non intrinsic functions. Also note the use of the INTENT(OUT) which means we can only give value to the argument and trying to read the value from the argument would flag a compilation error just like it would if we were trying to give the argument a value when it has the attribute INTENT(IN) which means read only.

A.6 Modules

Now it is time to progress further into the world of Fortran programming. We know now how to use functions and subroutines, but often we need to put global variables together with the corresponding procedures working with these variables. It is here we utilize the MODULE which was introduced in Fortran 90. A module consists of a set of variable declarations and an optional set of functions and subroutines working on the variables. A skeleton module can look like this:

Appendix B

Quality assurance procedures

The aim is to present a *summary* of a set of sound procedures to be followed to establish what is referred to below as a *good* model. The text is based on earlier reports by the author on the subject, in particular *McClimans et al.* (1992) and *Røed* (1993). For more extensive reading on the subject the reader is recommended the in depth analysis documented in the GESAMP report *GESAMP* (1991), or the anthology *Lynch and Davies* (1995).

B.1 Introduction

Although many of today's engineering models are formulated into mathematical equations leading to a mathematical model that can be solved reliably with almost ”canned” routines that require little understanding on the part of the user (referred to as ”expert systems for non-experts”), atmospheric and oceanographic weather prediction models available today are not yet among them. Today's atmospheric and oceanographic prediction models are thus prime examples of complex mathematical models involving coupling of intricate physical, and sometimes chemical and biological model modules. Inherently most complex models requires a minimum of expertise to be transferred with the model, so that only in exceptional circumstances is it possible to turn a complex model developed by one group over to another. The reason is simply that all complex model systems have their inherent limitations that demands an understanding of the underlying processes, and in the case of numerical model also the numerical techniques used to solve them. The simulation is never perfect; different models and methods preserve different features of the original problem implying that one needs to understand what is important for the purpose at hand. Nevertheless, in meteorology and oceanography such almost "canned" systems are publically available for downloading on the on the web. Regarding models of the atmosphere the Weather Research & Forecasting Model (WRF; http://www.wrf-model.org) is a prime example, while the same is true for the Regional Ocean Modeling System (ROMS; http://www.myroms.org/) regarding the ocean.

As such any mathematical model is, at best, an approximate representation of the real world. Hence, its predictions are inherently uncertain. This uncertainty results from both a lack of knowledge of the full set of equations and an inability to solve them; therefore approximations have to be made that involve the use of parameterizations of the processes in space and time. Uncertainty also arises from errors in observational data used to derive input and parameter values, that is, the initial state of the model and the boundary conditions. In addition there may be problems with the accuracy of the computer code and the method and techniques used to solve the discretized numerical analogue of the original continuous mathematical model. All these need attention when determining the accuracy of the model predictions.

As a precursor Section B.2 therefore highlight one of the common problems in solving a mathematical model by numerical techniques, namely the parameterizations of unresolved scales most often referred to as sub-grid scale parameterizations, using the advection eqauiton as an example. This paves the way for Section B.3 which describes in general terms what is meant by a *good* model, and defines such terms as a *tuned* model, a *transportable* model, and a *robust* model. Section B.4 then describes what is sometimes referred to as quality assurance or model validation procedures. This is a three step process in which the first step (Section B.4.1) is to check or verify that the mathematical equations are solved correctly (referred to as model verification). The next step (Section B.4.2) is to perform a sensitivity analysis to uncover the models response to changes in the input data, parameter values and parameterizations. This is suitable to uncover the predictive skills of the model and is referred to as a model sensitivity study. The third and final step (Section B.4.3) is to investigate the agreement between the model predictions and observations, a task commonly referred to as model validation. Model calibration, that is, the tuning of parameter values to make the model output fit a given data set, is included in these discussions. Finally a concluding Section B.5 is offered in which some final remarks are made.

B.2 Sub-grid scale parameterizations and spectral cutoffs

An exact numerical solution of the governing equations for the atmosphere and ocean as they are outlined in Chapter 1 is impossible, mostly due to processes not resolved by our grid. Thus there exists spectral cutoffs regarding processes on scales smaller than the grid resolution (cf. Section), that is, the sub-grid scale processes. The effect of these sub-grid scale processes on the resolved scales must then to be parameterized, that is, be given an approximate mathematical formulation. Commonly this is in the form of simplified formulas involving the specification of one or several parameters that may or may not be functions of the resolved scales.

The need for such parameterizations can best be illustrated by considering the advection-diffusion equation simplified to include one dimension in space only. Let $C = C(x,t)$ denote any property of the fluid, that is, any state scalar such as potential temperature or concentration of a particular contaminant, at location x at time t. Then mass conservation requires

$$\partial_t C + \partial_x(uC) = 0, \tag{B.1}$$

where $u = u(x,t)$ is the speed along the x-axis by which the property C is advected[1] (or propagated). The first term on the left-hand side of (B.1) then represent the time rate of change of the

[1]In a three-dimensional problem the speed becomes a current, that is, a vector $\mathbf{u}$, and (B.1) becomes $\partial_t C + \nabla \cdot (\mathbf{u}C) = 0$ where ∇ is the three-dimensional gradient vector.

potential temperature or the contaminant in question, while the second term is the divergence of the advective flux $F_{adv} = uC$, that is, the divergence due to the transport of property C.

In practice it is impossible to describe such a flow field as u changes rapidly in both space and time. Hence an ensemble average or a space-time averaging process, the latter taken over a certain length scale T in time and/or L in space, must be invoked, which separates the current into an average or mean current, $\overline{u}$, and a random component, u', such that $u = \bar{u} + u'$ where $\overline{u'} = 0$. In this u may be thought of as being the mean flow over a certain time period and u' as the motion deviating from the mean so that $\bar{u} + u'$ makes up the instant flow at any time or location.

If the same separation is used for the concentration it follows from (B.1) that

$$\partial_t(\bar{C} + C') + \partial_x \left[(\bar{u} + u')(\bar{C} + C')\right] = 0. \tag{B.2}$$

Averaging (B.2) over the averaging period (or length), noting that terms like u', C', $u'\bar{C}$, and $\bar{u}C'$ average out, it becomes

$$\partial_t C + \partial_x(\bar{u}\bar{C}) = \partial_x \left(\overline{u'C'}\right). \tag{B.3}$$

Note that the left-hand side of (B.3) is very similar to (B.1), except for the non-zero term on the right-hand. As such it is an advection equation for a concentration C with a speed $\bar{u}$, which in fact is the concentration and motion resolved by our ”model”. The term on the right-hand side of (B.3) is simply a measure of the influence of the fluctuating motion u0 and the fluctuations in the concentration C' on the mean concentration C. To solve (B.3) with respect to u and C, the right-hand side, which contains the unresolved concentrations and motions, must somehow be expressed in terms of the average or resolved quantities, that is, be parameterized. Commonly, with regard to the advection-diffusion equation, this is done by parameterizing the influence as a diffusive process, that is,

$$F_{diff} = -\overline{u'C'} = -K(x,t)\partial_x C, \tag{B.4}$$

where F_{diff} is the diffusive flux[2]. The parameter or coefficient K is called the eddy diffusivity or dispersion coefficient, and is in general a function of time and space. As alluded to below (Section B.4.1) it is important to test the sensitivity of a model to these parameterizations. This can give insight into the fitness of the parameterization and may help to build confidence in the model and its predictive capability. Model prognoses that are highly sensitive to a particular parameterization or to the value given to a particular parameter should be treated with caution. Similar problems occur when parameterizing, e.g., biological and physical processes. The division of species, size distribution, patchiness, algae successions, like details of turbulence, are all poorly known and must be parameterized. Integral quantities like biomass, chlorophyll and Secci depth represent a multitude of biological variables. These are in total affected by the physical/chemical environment, which effects can be calibrated into flux/transformation formulas.

[2]In a three-dimensional problem the diffusive flux becomes a vector $\mathbf{F}_{diff} = -\boldsymbol{\mathcal{K}} \cdot \nabla C$, where $\boldsymbol{\mathcal{K}}$ is a tensor. The right-hand side of (B.3) is then written $\nabla \cdot \mathbf{F}_{diff}$.

B.3 What is a good model

B.3.1 Tuned, transportable, and robust models

In theory an integrated model with refined descriptions of the many processes involved and the interactions between them, and which includes complex and sophisticated parameterizations, should provide more accurate results and be more applicable to different situations and/or geographical areas than a model invoking simpler and coarser descriptions and parameterizations. This philosophy reflects a conviction that more detailed formulations provide a better description of the processes than simpler ones. In practice, this concept breaks down in many cases for the following reasons

- it is sometimes questionable whether a ”true” description exist for all relevant processes (e.g., turbulent mixing),
- too many processes are included that have to be parameterized, or
- our knowledge and understanding of the unresolved processes upon which the parameterizations are based is poor.

Under these circumstances a complex model is of little value, since no more fundamental knowledge is being incorporated into the model, only parameterizations of poorly understood processes. This is visualized in Figure 1.

Before constructing a good model a set of criteria has to be selected in order to make the necessary choices, that is, which processes and which interactions between processes should be included to answer the management question. It is convenient in this respect to introduce the terms tuned model, transportable model, and robust model.

- A *tuned* model is one in which the parameterizations and parameter selection have been adjusted to reproduce, as accurate as possible, a given data set in a specific region for a specific time interval. In general the more adjustable parameters there are in a model, the more difficult it is to tune the model, the more data is required, and the more site-specific the model becomes.
- A model is *transportable* if the parameterizations within the model are sufficiently comprehensive and representative of all relevant processes that, once calibrated and validated in one geographical area, the model can be used in any area containing the same generic processes. This does not imply that the specific parameter values that have been chosen for one area should remain invariant when the model is transported. However, a transportable model should yield similar levels of accuracy in a different geographical area once it has been properly calibrated.
- A model is *robust* when it can provide similar degrees of accuracy over a wide range of variations in the forcing functions. For instance, storm surge models are normally validated against observations during major storm events. Nevertheless, they are expected to give the same level of accuracy for even more extreme events (e.g., the hundred year storm) for which no direct validation is usually possible.

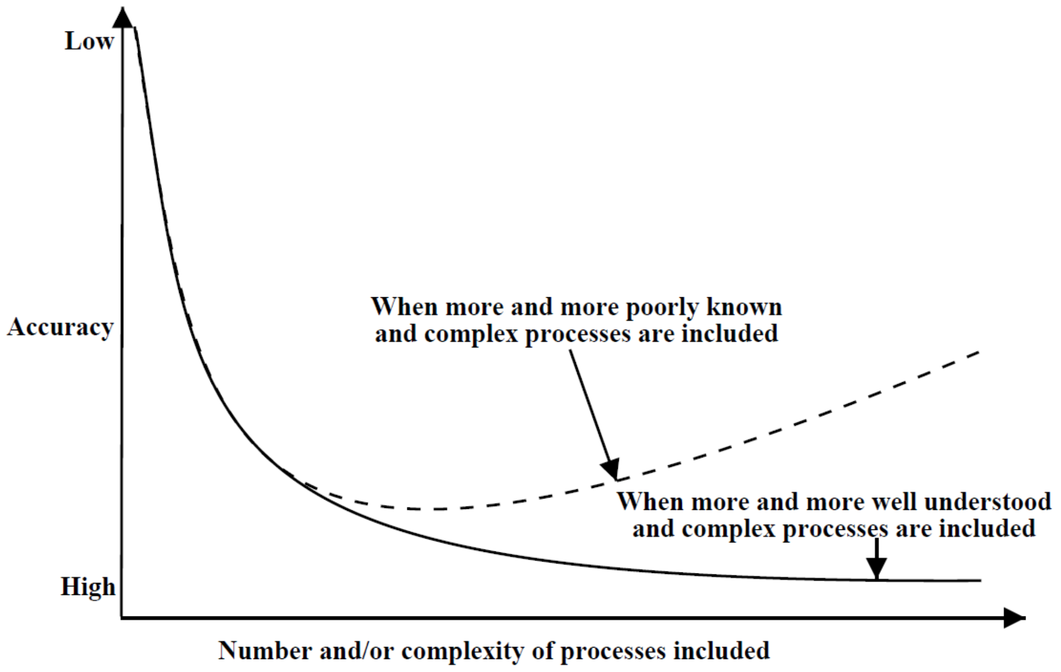


Figure B.1: Sketch showing the effect on the accuracy by including more and more complex and sophisticated parameterizations of processes. Decreasing accuracy is shown along the vertical axis, while increasing number and/or complexity of processes are shown along the horizontal axis. Note that the accuracy first increases but then decreases (dashed curve) when more and more poorly known processes are included. This is contrast to the case when the processes included are well known and understood (solid curve).

B.3.2 The concept of a good model

With the definitions given in Section 4.1 in mind, one may think of a good model as one which

- retains a conceptual representation of the processes known to be important,
- uses parameterizations consistent with our knowledge of those processes,
- does not use parameterizations that are so complex that the model needs to be highly tuned,
- can reproduce and/or predict phenomena over a wide range of geographical location (i.e., transportable), and
- can reproduce and/or predict phenomena over a wide range of differing conditions (i.e., robust).

These criteria differ somewhat from the commonly accepted definitions of a good model, which may be drawn from many examples in the literature. There a good model is simply one which accurately reproduces a given set of field observations. However, if this has been accomplished by tuning the model, so that it essentially fits a narrow set of conditions, it is no longer good since it may not accurately predict different events in the same area, the same kind of events in another geographical area, or provide insight into the nature of the underlying processes.

B.4 Quality assurance procedures

To ensure that a particularly model fulfills the requirements of a good model, it should be possible somehow to assess its ”quality”. Ideally there should exist documentation that reports results from procedures or steps that have been followed to test the model’s behavior with regard to the above definition of robustness, transportability and tuning. The idea is that it should be possible for a third party to assess the procedures taken to ensure the quality of a particular model, and to evaluate the results emanating from these procedures. Collectively such procedures, including their documentation, are commonly referred to as quality assurance procedures. They are also sometimes pragmatically referred to as model validation (*Dee*, 1995) reflecting the fact that the ultimate goal is that the model should be able to reliably simulate what happens in nature.

Below follows a descriptions of three steps that taken together form the steps necessary to construct a useful quality assurance procedures. These are loosely called

1. Model verification,
2. Sensitivity analysis
3. Model validation

Other examples of results where such procedures are attempted and/or reviewed are found in the anthology *Lynch and Davies* (1995). Besides the steps reviewed here also model-model comparison exercises helps to elucidate a models quality. Examples of the latter is found in *Hackett and Røed* (1994), *Røed et al.* (1995), and *Hackett et al.* (1995).

B.4.1 Model verification

Model verification is a particularly important first step in a model development and in the quality assurance procedures. The aim is to ensure that there are no errors in the computer coding or in the numerical solution method. The first part entails that the computer program (or code) should be rigorously checked by comparing the coding with the numerical algorithm chosen to solve the model’s governing equations[3]. Obviously this part should be performed at an early stage in the model development. More often than not this is a very time consuming task, and in particular in the case of a complex numerical model which frequently includes many tens of thousands of lines of coding.

[3]The governing equations are the mathematical formulation of the physical and or biochemical processes that are to be simulated. Commonly this is a set of coupled partial differential equations.

The second part of the model verification is to ensure that there are no errors in the numerical solution method chosen. Ideally, this can be performed by comparing the numerical solution with analytical solutions or at least other accurate numerical solutions. However, for most problems of a certain complexity no analytical solutions exist. This is simply because most processes of interest are non-linear. Nevertheless, it is usually possible to test separate parts of the entire model against either analytical solutions or accurately known numerical solutions. Only by this means can confidence be established in the accuracy of the model. Note that by accuracy in this context it is meant to which degree the numerical solution approximates the true solution of the governing equations. Regrettably, this important stage in model development is frequently omitted in the construction and application of many numerical models.

Additional complications arise because the methods used to solve the governing equations once the mathematical model is formulated are commonly wholly numerical. It is therefore a major task to ensure that the numerical methods employed are sufficiently rigorous that the solution provided is accurate under a range of conditions. This can be particularly difficult since most numerical solutions are prone to errors in regions where or during periods when the predicted contaminant or any other variable in the model exhibits strong gradients (sometimes referred to as fronts across which the variable in question experience a large or abrupt change). In some cases, the occurrence of such a front can be anticipated, e.g., high velocity gradients in shear boundary layers and high concentrations gradients close to the source of a contaminant discharge. However, in many cases the occurrence of fronts or frontal structures cannot be anticipated. Fronts may also evolve in time, e.g., a frontal structure may form within a certain time span (frontogenesis) and may break down due to strong outflow or wind events and/or diffusion, but may later reestablish at low outflow or wind conditions.

It is reasonable therefore to state that an integrated model of a certain complexity that aims at giving the correct solution for all times and at all locations (an ”all singing - all dancing” model) is still in its infancy. It is also fair to state that the development of numerical techniques and advanced supercomputing, which can provide an accurate solution of the coupled partial differential equations representing the processes in an integrated model is also currently in its infancy, but fortunately fast growing.

Although the numerical model, once verified, is deemed a proper representation of the mathematical formulation, the mathematical model in itself may still be a gross approximation of the real system. Thus a model verification is only one step toward the ultimate goal of establishing our confidence in the validity of the results that the integrated model produce when used to answer certain specific management questions.

B.4.2 Sensitivity analysis

The next step in the determination whether the chosen model accurately reproduces conditions in the ”real world”, is a model sensitivity study. The aim of this step is to establish the predictability power of the model. In essence there are two distinct components of the sensitivity analysis; the first deals with sensitivity to input data and conditions, and the second involves sensitivity to the chosen parameter values and to the parameterizations themselves.

In the first the sensitivity of the model output to variations in the input data (usually based on variations in field observations) is tested accepting the model as formulated knowing that certain terms in the governing equations are approximated and other intricate processes reduced to simple parameterizations. The range of variations in the input data can be determined from a knowledge of variations and estimated errors in the observed data. Such an exercise, often referred to as an uncertainty analysis, is particularly revealing both in terms of establishing the sensitivity of the model and in identifying crucial field observations. If the sensitivity study reveals that the model output is crucially dependent on the precision and accuracy of certain measurements, then effort must be made to reduce the error in these measurements. If for example the model shows that representation of processes and boundary conditions at one geographical location has a larger effect upon model output than others, then an observational program can be designed to sample more intensely in that critical area.

The second component of the sensitivity analysis involves the various assumptions which are made in developing the model. The major difficulty here is related to the problem of parameterizations of small scale processes, e.g., mixing processes in hydrodynamical and biogeochemical models, that cannot be resolved explicitly within the numerical model (cf. Section 3). Consider, for instance, the parameterization of mixing processes. Physically these processes are associated with the turbulent motions in the fluid. The mechanisms producing this turbulence and its intensity is a prime example of a poorly understood process; however, they are clearly related to larger scale physical phenomena. In this context bed roughness determines near-bed turbulence, and larger scale obstructions in a river bed causes hydraulic jumps which is associated with vigorous turbulent shocks downstream. The representation of such and similar processes in the physical compartment of the model as well as similar processes in the chemical and biological compartments of the integrated model (e.g., a water quality model), is particularly difficult. They may sometimes be parameterized by a single coefficient, e.g., a diffusion coefficient as exemplified in Section 3, or they can be represented in a hydrodynamical model by a complex system of turbulence energy equations. In any true sensitivity study, a range of formulations parameterizations of these mixing processes must be considered. If such a sensitivity study shows that the contaminant distribution are sensitive to the mixing formulation (which is normally the case), then confidence limits can be placed upon the model based upon the accepted range of parameterizations of the mixing process. In the unlikely event that a sensitivity analysis reveals that the model is insensitive to the formulation of mixing then only the simplest formulation of this process is required. (Naturally, similar conclusions hold in the bio-geochemical parts of the model.) However, in reality, the major problem arising in a sensitivity analysis of an ”all-encompassing model” model is that, in certain circumstances, results may be insensitive to one part of the model. In other circumstances the formulation of this same part of the model may be critical. Such a finding obviously leads to a conclusion that, in practice, a range of models are required.

A conservative approach in developing an integrated model is therefore that each model only needs to embody those processes that are essential for providing accurate answers to the specific management questions raised. In most applications, this approach to modeling is to be preferred. By contrast, an ”all singing - all dancing” model, designed to cover every conceivable situation, is rarely constructed because of the requirements for immense computer power, a large body of supporting field data, and the problems imposed in conducting a comprehensive sensitivity analysis. The latter is particularly relevant, since the task of effectuating a comprehensive sensitivity analysis is nearly open ended requiring large amounts of resources to be available, and hence is rarely undertaken. In fact, it is often a relief to find out that a model does not have to be perfect, all-encompassing or complicated to be useful.

When an integrated model is verified and a proper sensitivity analysis has been executed, some confidence in the model is definitely established. It is then ensured that the model code is accurate, that the numerical methods are sound for the problem at hand, and that the numerical solutions is able to reproduce known solutions, albeit in a reduced, simplified and idealized context. A properly conducted (and documented) sensitivity study further increases our confidence in the predictive power of the model and help to understand which parameters and parameterizations are crucial to know with a proper certainty and which are not. However, there is still no confidence in that the model produces results that are valid in the sense that the model is able to reproduce a given observational data set for the correct reasons. Thus there is still one final step that is needed.

B.4.3 Model validation

The ultimate test of an integrated model's usefulness is its ability to accurately predict the water quality of a river bed, or if the management question is related to a particular contaminant, the contaminants transport and distribution at appropriate interfaces with the effects models. At first sight the ability of a model to reproduce a given data set would appear to be a good guide to its predictive capability. However, some care must be exercised in reaching this conclusion. If a sensitivity analysis has revealed that the model is sensitive to variations of a poorly known parameter, and a good fit between model output and observations is achieved by adjusting this parameter, then the model may legitimately be be regarded as a tuned or at least a highly calibrated model. Ideally, such a model should be able to produce similarly accurate results under similar conditions elsewhere; but in practice, a tuned model is probably neither transportable nor robust in the sense defined in Section 4. A potential user of the model should then be cautioned and the conditions under which the model may be applied with some confidence should be clearly stated, that is, be part of the model's documentation.

The user must be aware of how extensively a model is validated before it can be used. If the model can reproduce various observational data obtained under a large range of physical and bio-geochemical conditions without adjusting parameterizations, the model can be regarded as transportable. It may therefore be applied with confidence over a wider range of situations.

In general, data are required both for model operation and model validation. Data used for model operation include initial conditions, boundary conditions, source terms, and meteorological forcing functions. One of the most difficult aspects of modeling in some fields, e.g., fjord modeling, is how to provide a suitable description of conditions at an open boundary, that is, conditions to apply at the boundary where the estuary meets the ocean. In particular data for defining conditions of the hydrodynamics and contaminant fluxes between the far field and the open ocean beyond are not always available. In the absence of appropriate data, the only recourse is to use simple assumptions, such as diffusion into an infinite field or periodic flow conditions at the boundary, in order to keep the model operational.

Ideally, model validation is achieved when the model output compares favorably with data sets independent of those used during model calibration, that is, those used to tune the parameters of the model. In the case of a complex deterministic model this could be an overwhelming task. In theory, the predictions of the model should be compared at all appropriate levels with different data obtained from real systems. However, this is rarely done in practice. In some cases, all or part of the calibration data is used again in the validation. Such a partial validation, using data from the same site and/or under similar conditions, is called model confirmation.

In the final analysis, it is crucial that independent data sets from many different regimes is used to establish the model's credibility through rigorous statistical tests. For example the standard deviation between modeled and measured values gives a quantitative measure of how well the model works. Very few validations provide this quantitative measure. Obviously, if the observational data are very limited, then all of it is used in the calibration stage and a model validation and even a model confirmation is not possible until further and preferably wider range of data sets become available.

It is also important to realize that a good validation performance does not necessarily guarantee that the model will accurately predict future conditions. Some uncertainties will always remain in the model coefficients, the models variables, and the model structure itself. Therefore, models should be subjected to post-audits in which their predictions are tested with data obtained after an environmental control program is implemented. The purpose of this stage of validation is to check whether the model reproduces the expected changes. Unfortunately, it is only in exceptional cases that a post-audit is feasible, and hence it rarely occurs. Only recently has there been some activity in this phase of validation.

Model users must also be aware of the quality and relevance of observational data. Even the best of models cannot make reasonable and accurate predictions if these predictions are based on imprecise or inaccurate input data. Although the adage ”garbage in - garbage out” has become common modeling jargon, it nonetheless provides an important cautionary note for potential model users. In many cases, the underlying cause of such a situation is that data used for model development were originally collected for a purpose other than modeling. If data collection programs are more closely linked with modeling studies, then the constraints imposed by the lack of suitable data can be substantially reduced. The bottom line is that there is always an acute need for high quality, relevant data sets for model calibrations and validations.

A most difficult problem is proving that the model is robust (cf. Section 4), namely that it can predict extreme conditions with some confidence. Since most validations data are collected under normal conditions, they are of little value in assessing confidence in the model output under extreme circumstances. A sensitivity analysis is then probably the most appropriate manner of determining the value of the model under such circumstances. If corrected parameterizations of the various processes are included in the model, and the confidence in our knowledge of these processes is high, then the model should be reasonably accurate under extreme conditions. Fortunately, extreme conditions are seldom a problem for most quality management issues.

B.5 Summary and final remarks

This note provides some guidelines toward development of criteria whereby the quality of integrated models can be assessed objectively. Discussed are three steps that are deemed necessary to objectively establish confidence in any model aiming to answer specific management questions relating to problems involving hydrodynamic and biochemical processes. These guidelines are referred to as quality assurance procedures, a task that is rarely undertaken to its full extent, mostly due to lack of data. Nevertheless, the guidelines are recommended as the backbone in the development of a set of objective criteria aiming at evaluating the usefulness of integrated models for the WFD.

The first step in the quality assurance procedures is a model verification. It involves checking the numerical code developed and the numerical solution methods used to solve the underlying mathematical formulation of the model. The next step is to perform a model sensitivity analysis. The aim of such an excercise is to establish which parameters and/or parameterizations are critical to know accurately under what conditions. Sometimes these parameters are tuned to make the model match a certain given observational data set. However, if this involves tuning of critical parameters, the model's predictive skill is poor in the sense that the model probably fails when used under different conditions. The final step is to perform a model validation which aims at establishing a measure of how the model output compares with observational data. In this it is important that the data set exploited consist of measured data collected under different conditions than the data used to calibrate the model, that is, the data used to determine the models parameters and parameterizations. If not the exercise is not a true model validation, but is classified as a model confirmation activity.

In the above the concept of a good model is introduced. A good model is one which can be used in any area containing the same generic processes (transportable model) and one which can provide similar degree of accuracy under a wide range of conditions (robust model). It should be emphasized that this does not imply that all models have to be good models to be useful. Also highly tuned models may be useful under certain conditions. However, these underlying conditions should be clearly stated and be transparent to potential users.

Finally, the description above is general in nature because there are so many processes and interaction between processes which must be formulated and parameterized to construct a useful integrated model for even the simplest specific management questions. Thus there is a general feeling that to develop and apply routinely an ”all singing - all dancing” model to give answers to specific problems would be too expensive and too complicated . This is why there exist a plethora of models from the simplest box type models to the most expensive three-dimensional models. Most models are in one way or another tuned to the local river basin, lake or fjord situations and the problem at hand.

Given the above it is likely and probably sound that there exist a wide range and number of integrated models that can potentially be used to answer management questions regarding the WFD. Fortunately, many of the existing models use standard formulations and parameterizations that are well proven and/or widely accepted by the international community. The important message here is that anyone who offers a model as a tool to answer a specific management question or problem for a potential user should also provide documentation of the quality of the offered model, that is, provide documentation of the quality assurance procedures that has been followed and the results thereof, so that the potential user objectively can assess the models quality and suitability. Hence the bottom line is that any integrated model that is offered for use as a management tool within the WFD must be able to document its quality in the sense described above. This is the only means whereby it can be applied with some confidence.

Bibliography

Adamec, D., and J. J. O'Brien (1978), The seasonal upwelling in the Gulf of Guinea due to remote forcing, *J. Phys. Oceanogr.*, *8*, 1050–1060.

Arakawa, A., and V. R. Lamb (1977), Computational design of the basic dynamical process of the ucla general circulation model, *Methods in Computational Physics*, *17*, 173–265.

Asselin, R. A. (1972), Frequency filter for time integrations, *Mon. Weath. Rev.*, *100*, 487–490.

Berenger, J.-P. (1994), A perfectly matched layer for the absorption of electromagnetic waves, *J. Comput. Phys.*, *126*, 185–200.

Biastoch, A., C. BÃ¶ning, and J. Lutjeharms (2008), Agulhas leakage dynamics affects decadal variability in atlantic overturning circulation, *Nature*, *456*, 489–492.

Bjerknes, V. (1904), Das Problem der Vettervorhersage, betrachtet vom Standpunkte der Mechanik und der Physik, *Meteor. Zeitschr.*, *21*, 1–7.

Blayo, E., and L. Debreu (2005), Revisiting open boundary conditions from the point of view of characterstics, *Ocean Modell.*, *9 (3)*, 234–252, doi:10.1016/j.ocemod.2004.07.001.

Blayo, E., and L. Debreu (2006), Nesting Ocean Models, in *Ocean Weather Forecasting: An Integrated View of Oceanography*, edited by E. Chassignet and J. Verron, pp. 127–146, Springer, doi:10.1007/1-4020-4028-8_5.

Bleck, R. (1973), Numerical forecasting experiments using based on conservation of potential vorticity on isentropic surfaces, *J. Appl. Meteor.*, *12*, 737–752.

Bleck, R. (2002), An oceanic general circulation model framed in hybrid isopycnic-cartesian coordinates, *Ocean modelling*, *4*(1), 55–88.

Bleck, R., and L. Smith (1990), A wind-driven isopycnic coordinate model of the north and equatioral Atlantic Ocean. 1. Model development and supporting experiments., *J. Geophys. Res.*, *95C*, 3273–3285.

Blumberg, A., and G. Mellor (1987), A description of a three-dimensional coastal ocean circulation model., in *Three-dimensional Coastal Ocean Models*, *Coastal and Estuarine Sciences*, vol. 4, edited by N. Heaps., pp. 1–16, American Geophys. Union.

Chapman, D. C. (1985), Numerical treatment of cross-shelf open boundaries in a barotropic coastal ocean model, *J. Phys. Oceanogr.*, *15*, 1060–1075.

Charney, J. G. (1955), The generation of ocean currents by wind, *J. Mar. Res.*, *14*, 477–498.

Charney, J. G., and N. A. Phillips (1953), Numerical integration of the quasi-geostrophic equations for barotropic and simple baroclinic flows, *J. Meteor.*, *10*, 71–99.

Charney, J. G., R. Fjørtoft, and J. von Neumann (1950), Numerical integration of the barotropic vorticity equation, *Tellus*, *2*, 237–254.

Clancy, R. M. (1981), On wind-driven quasi-geostrophic water movement at fast ice edges, *Mon. Weath. Rev.*, *109*, 1807–1809.

Cooper, C., and J. D. Thompson (1989), Hurricane generated currents on the outer continental shelf, *J. Geoph. Res.*, *94*, 12,513–12,539.

Cushman-Roisin, B. (1984), Analytic, linear stability criteria for the leap-frog, dufort-frankel method, *J. Comp. Phys.*, *53*, 227–239.

Davies, H. C. (1976), A lateral boundary formulation for multilevel prediction models, *Quart. J. Roy. Meteor. Soc.*, *102*, 405–418.

Davies, H. C. (1985), Limitation of some common lateral boundary schemes used in regional NWP models, *Mon. Weath. Rev.*, *111*, 1002–1012.

Debreu, L., and E. Blayo (2008), Two-way embedding algorithms: a review, *Ocean Dynamics*, *58*, 415–428, doi:10.1007/s10236-008-0150-9.

Debreu, L., P. Marchesiello, P. Penven, and G. Cambon (2012), Two-way mesting in split-explicit ocean models: Algorithms, implementation and validation, *Ocean Modeling*, *49-50*, 1–21.

Dee, D. P. (1995), A pragmatic approach to model validation, in *Quantitative Skill Assessment for Coastal Ocean Models*, *Coastal and Estuarine Studies*, vol. 47, edited by D. Lynch and A. Davies, pp. 1–13, American Geophysical Union.

Edwards, P. N. (2011), History of climate modeling, *Wiley Interdisciplinary Reviews: Climate Change*, *2*(1), 128–139, doi:10.1002/wcc.95.

Eliassen, A. (1949), The quasi-static equations of motion with pressure as independent variable, *Geof. publ.*, *17*(3), 44 pp.

Eliassen, A., and E. Raustein (1968), A numerical integration experiment with a model atmosphere based on isentropic coordinates, *Meteor. Ann.*, *5*, 45–63.

Eliassen, A., and E. Raustein (1970), A numerical integration experiment with a six-level atmospheric model with isentropic information surface, *Meteor. Ann.*, *5*, 429–449.

Engedahl, H. (1995a), Use of the flow relaxation scheme in a three-dimensional baroclinic ocean model with realistic topography, *Tellus*, *47A*, 365–382.

GESAMP (1991), (IMO/FAO/UNESCO/WMO/WHO/IAEA/UN/UNEP Joint Group of Experts on the Scientific Aspects of Marine Pollution), Coastal Modelling, *Tech. rep.*, International Atomic Energy Agency (IAEA), GESAMP Reports and Studies 43, 192 pp.

Gill, A. E. (1982), *Atmosphere-ocean dynamics*, *International Geophysical Ser.*, vol. 30, Academic Press.

Griffies, S. M. (2004), *Fundamentals of ocean climate models*, Princeton University Press, ISBN 0-691-11892-2.

Grotjhan, R., and J. J. O'Brien (1976), Some inaccuracies in finite differencing hyperbolic equations, *Mon. Weath. Rev.*, *104*, 180–194.

Hackett, B., and L. P. Røed (1994), Numerical modeling of the Halten Bank area: a validation study, *Tellus*, *46A*, 113–133.

Hackett, B., L. P. Røed, B. Gjevik, E. A. Martinsen, and L. I. Eide (1995), A review of the metocean modeling project (MOMOP). Part 2: Model validation study, in *Quantitative Skill Assessment for Coastal Ocean Models*, *Coastal and Estuarine Studies*, vol. 47, edited by D. R. Lynch and A. M. Davies, pp. 307–327, American Geophysical Union.

Haidvogel, D. B., H. Arango, P. W. Budgell, B. D. Cornuelle, E. Curchitser, E. D. Lorenzo, K. Fennel, W. R. Geyer, A. J. Hermann, L. Lanerolle, J. Levin, J. C. McWilliams, A. J. Miller, A. M. Moore, T. M. Powell, A. F. Shchepetkin, C. R. Sherwood, R. P. Signell, J. C. Warner, and J. Wilkin (2008), Ocean forecasting in terrain-following coordinates: Formulation and skill assessment of the Regional Ocean Modeling System, *J. Comput. Phys.*, *227*(7), 3595–3624, doi:http://dx.doi.org/10.1016/j.jcp.2007.06.016.

Haltiner, G. J., and R. T. Williams (1980), *Numerical prediction and dynamic meteorology, second edition*, 477 pp., John Wiley & Sons.

Hannay, J. E., C. MacLeod, J. Singer, H. P. Langtangen, D. Pfahl, and G. Wilson (2009), How do scientists develop and use scientific software?, in *SECSE '09: Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering*, pp. 1–8, IEEE Computer Society, Washington, DC, USA, doi:10.1109/SECSE.2009.5069155.

Hedstrøm, G. W. (1979), Nonreflecting boundary conditions for nonlinear hyperbolic systems, *J. Comp. Phys.*, *30*, 222–237.

Kasahara, A. (1974), Various vertical coordinate systems used for numerical weather prediction, *Mon. Wea. Rev.*, *102*, 509–522.

Lax, P., and B. Wendroff (1960), Systems of conservation laws, *Comm. Pure Appl. Math*, *13*, 217–237.

Lax, P. D., and R. D. Richtmyer (1956), Survey of the stability of linear finite difference equations, *Comm. Pure Appl. Math*, *9*, 267–293 MR 79,204, doi:10.1002/cpa.3160090206.

Lighthill, M. J. (1969), Unsteady wind-driven ocean currents, *Quart. J. R. Met. Soc.*, *95*, 675–688.

Lighthill, M. J. (1970), *Fourier analysis and generalised functions*, Cambridge University Press, First printed 1958.

Lister, M. (1966), The numerical solution of hyperbolic partial differential equations by the method of characteristics, in *Mathematical Methods for Digital Computers*, edited by A. Ralston and H. S. Wilf, John Wiley, New York.

Lorenz, E. N. (1955), Available potential energy and the maintenance of the general circulation., *Tellus*, *2*, 157–167.

Lynch, D. R., and A. M. Davies (1995), *Quantitative Skill Assessment for Coastal Ocean Models*, *Coastal and Estuarine Studies*, vol. 47, American Geophysical Union.

Marchesiello, P., X. Capet, C. Menkes, and S. C. Kennan (2011), Submesoscale dynamics in tropical instability waves, *Ocean Modell.*, *39*, 31–46, doi:10.1016/j.ocemod.2011.04.011.

Martinsen, E. A., and H. Engedahl (1987), Implementation and testing of a lateral boundary scheme as an open boundary condition in a barotropic ocean model., *Coast. Eng.*, *11*, 603–627.

Martinsen, E. A., B. Gjevik, and L. P. Røed (1979), A numerical model for long barotropic waves and storm surges along the western coast of norway, *J. Phys. Oceanogr.*, *9*, 1126–1138.

Mason, E., J. Molemaker, A. F. Shchepetkin, F. Colas, and J. McWilliams (2010), Procedures for offline grid nesting in regional ocean models, *Ocean Modell.*, *35*, 1–15, doi: 10.1016/j.ocemod.2010.05.007.

McClimans, T. A., L. P. Røed, and A. Thendrup (1992), Fjord water quality/ecological modelling. State of the art and needs, *Tech. rep.*, Royal Norwegian Council for Scientific and Industrial Research, Programme on Marine Pollution, 54 p. + appendices and attachments, ISBN 82-7224-335-0.

Mesinger, F., and A. Arakawa (1976), Numerical methods used in atmospheric models, GARP Publication Series No. 17, 64 p., World Meteorological Organization, Geneva, Switzerland.

Navon, I. M., B. Neta, and M. Y. Hussaini (2004), A perfectly matched layer approach to the linearized shallow water equations models, *Mon. Weath. Res.*, *132*(6), 1369–1378.

O'Brien, J. J. (Ed.) (1986), *Advanced Physical Oceanographic Numerical Modelling*, *NATO ASI Series, Ser. C: Mathematical and Physical Sciences*, vol. C 186, D. Reidel Publishing Company.

Orlanski, I. (1976), A simple boundary condition for unbounded hyperbolic flows, *J. Comp. Phys.*, *21*, 251–269.

Palma, E. D., and R. P. Matano (2000), On the implementation of open boundary conditions to a general circulation model: The 3-d case., *J. Geophys. Res.*, *105*, 8605–8627.

Pedlosky, J. (1979), *Geophysical Fluid Dynamics*, Springer-Verlag.

Phillips, N. A. (1957), A coordinate system having some special advantages for numerical forecasting, *J. Meteorology*, *14*, 184–185.

Phillips, N. A. (1959), An example of nonlinear computational instability, in *The Atmosphere and the Sea in motion*, edited by B. Bolin, pp. 501–504, Rockefeller Institute Press, New York.

Phillips, O. M. (1966), *The Dynamics of the Upper Ocean*, 269 pp., Cambridge University Press, New York.

Platzman, G. W. (1954), The computational stability of boundary conditions in numerical integration of the vorticity equation, *Arch. Met. Geophys. Biokl.*, *7*, 29–40.

Richardson, L. F. (1922), *Weather Prediction by Numerical Process*, 250 pp., Cambridge University Press.

Richtmyer, R. D. (1963), A survey of difference methods for non-steady fluid dynamics, *Technical Note 63-2*, National Center for Atmospheric Research (NCAR).

Richtmyer, R. D., and K. W. Morton (1967), *Difference methods for initial value problems*, 406 pp., Interscience.

Robert, A. (1981), A stable numerical integration scheme for the primitive meteorological equations, *Atmosphere-Ocean*, *19*, 35–46.

Robert, A. J. (1966), The integration of a low order spectral form of the primitive meteorologocal equations, *J. Meteor. Soc. Japan*, *44*, 237–245.

Robert, A. J., F. G. Shuman, and J. P. Gerrity (1970), On partial difference equations in mathematical physics, *Mon. Weath. Rev.*, *98*, doi:10.1175/1520-0493(1970)098<0001:OPDEIM>2.3.CO;2.

Røed, L. P. (1993), Models as management tools for environmental problems in water, *SFT-Rapport 93:14*, Statens forurensningstilsyn - SFT, Box 8100 Dep, 0032 Oslo, Norway, (in Norwegian), 38 p. + attachments, ISBN 82-7655-130-0.

Røed, L. P. (1997), Energy diagnostics in a $1\frac{1}{2}$-layer, nonisopycnic model, *J. Phys. Oceanogr.*, *27*, 1472–1476.

Røed, L. P. (1999), A pointwise energy diagnostic scheme for multilayer, nonisopycnic, primitive equation ocean models, *Mon. Weath. Rev.*, *127*, 1897–1911.

Røed, L. P., and C. K. Cooper (1986), Open boundary conditions in numerical ocean models, in *Advanced Physical Oceanographic Numerical Modelling, Series C: Mathematical and Physical Sciences*, vol. 186, edited by J. O'Brien, pp. 411–436, D. Reidel Publishing Co.

Røed, L. P., and C. K. Cooper (1987), A study of various open boundary conditions for wind-forced barotropic numerical ocean models, in *Three-dimensional models of marine and estuarine dynamics*, *Elsevier Oceanography Series*, vol. 45, edited by J. C. J. Nihoul and B. M. Jamart, pp. 305–3, Elsevier Science Publishers B.V.

Røed, L. P., and J. J. O'Brien (1983), A coupled ice-ocean model of upwelling in the marginal ice zone, *J. Geophys. Res.*, *29*(C5), 2863–2872.

Røed, L. P., and O. M. Smedstad (1984), Open boundary conditions for forced waves in a rotating fluid, *SIAM J. Sci. Stat. Comput.*, *5*, 414–426.

Røed, L. P., B. Hackett, B. Gjevik, and L. I. Eide (1995), A review of the metocean modeling project (MOMOP). Part 1: Model comparison study., in *Quantitative Skill Assessment for Coastal Ocean Models*, *Coastal and Estuarine Studies*, vol. 47, edited by D. R. Lynch and A. M. Davies, pp. 285–305, American Geophysical Union.

Sannino, G., M. Herrmann, A. Carillo, V. Rupolo, V. Ruggiero, V. Artale, and P. Heimbach (2009), An eddy-permitting model of the Mediterranean sea with a two-way grid refinement at the strait of Gibraltar, *Ocean Modell.*, *30*, 56–72.

Shapiro, M. A. (1974), The use of isentropic coordinates in the formulation of objective analysis and numerical prediction models, *Atmosphere*, *12*, 10–17.

Shapiro, R. (1970), Smoothing, filtering, and boundary effects, *Rev. Geoph. Space Phys.*, *8*, 359–387.

Shapiro, R. (1975), Linear filtering, *Math. Comp.*, *29*, 1094–1097.

Shi, X. B., B. Hackett, and L. P. Røed (1999), Documentation of DNMI's MICOM version, part 2: Implementation of a Flow Relaxation Scheme (FRS), *Research Report 87*, Norwegian Meteorological Institute, [Avaialable from Norwegian Meteorological Institute, Postboks 43 Blindern, N-0313 Oslo, Norway].

Shi, X. B., L. P. Røed, and B. Hackett (2001), Variability of the Denmark Strait overflow: a numerical study, *J. Geophys. Res.*, *106*, 22,277–22,294.

Sielecki, A. (1968), An energy-conserving numerical scheme for the solution of the storm surge equations, *Mon. Weather Rev.*, *96*, 150–156.

Smolarkiewicz, P., and L. G. Margolin (1997), MPDATA: a finite difference solver for geophysical flows, *J. Comp. Phys.*, *140*, 459–480.

Smolarkiewicz, P. K. (1983), A simple positive definite advection scheme with small implicit diffusion, *Mon. Wea. Rev.*, *111*, pp. 479.

Stern, M. (1975), *Ocean Circulation Physics*, *International Geophysics Series*, vol. 19, Academic Press.

Stoker, J. J. (1957), *Water waves: The mathematical theory with applications*, *Pure and Applied Mathematics: A series of texts and monographs*, vol. IV, Interscience publishers, Inc., New York.

Sundström, A., and T. Elvius (1979), Computational problems related to limited-area modeling, *GARP Publication Series*, *17*, 11.

Sutcliffe, R. C. (1947), A contribution to the problem of development, *Quart. J. Roy. Meteor. Soc.*, *73*, 370–383.

Warner, J. C., B. Armstrong, R. He, and J. B. Zambon (2010), Development of a coupled ocean-atmosphere-wave-sediment transport (coawst) modeling system, *Ocean Modelling*, *35*(3), 230–244, doi:http://dx.doi.org/10.1016/j.ocemod.2010.07.010.