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 20, 2015
This booklet is about how to put the atmosphere and oceans on computers. To this end we use numerical methods. To me numerical methods is one of the most fascinating contemporary tools to solve meteorological and oceanographic problems. The rationale is that it is the only tool allowing us to solve the fully nonlinear, partial differential equations (PDFs) that governs the motion of the atmosphere and oceans.

It is therefore hardly surprising that atmospheric scientists and weather forecasters alike embraced this tool early on. It all started when the mathematician John von Neumann, a well known Professor at the Princeton University, in 1946 (August) organized a “Conference on Meteorology” in order to acquaint the meteorological community with the electronic computer being built at Princeton and to solicit their advice and support in designing research strategies. The outcome of the conference was the Princeton Meteorological Project (1946-53) which was managed by Dr. Jule G. Charney. Among the participants were two young Norwegians, namely Ragnar Fjørtoft and Arnt Eliassen. The project successfully ended with producing daily numerical weather predictions in less than two hours. It was the very first attempt of producing a numerical weather forecast based on a set of simplified, nonlinear equations was published in 1950 by Jule G. Charney, Ragnar Fjørtoft and John von Neumann (Charney et al., 1950). The project was also the start of a whole new science field now referred to as Numerical Weather Prediction (NWP). Interestingly NWP, and its twin Numerical Ocean Weather Prediction (NOWP), are among the major science fields pushing the computer technology to its very limits.

An important basis for the rapid development of the NWP in the 1950s and 1960s was the deterministic paradigm stated by Vilhelm Bjerknes (Bjerknes, 1904), and the later attempt by Lewis Fry Richardson (Richardson, 1922) to compute a 6 hour weather forecast by hand. In his

---

1 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.

2 Ragnar Fjørtoft (1 August 1913 - 28 May 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.

3 Arnt Eliassen (9 September 1915 - 22 April 2000) was a Norwegian meteorologist who was a pioneer in the use of numerical analysis and computers for weather forecasting. The early pioneer work was done at the Institute for Advanced Study in Princeton, New Jersey, together with John von Neumann. His areas of research included free and thermally driven circulations, frontogenesis, and shear and gravitational-acoustic wave propagation in stratified media. He received the Carl Gustaf Rossby Research Medal in 1964 for his many important contributions to dynamical meteorology. He received 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”.

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

5 Lewis Fry Richardson,(11 October 1881 - 30 September 1953) was an English mathematician, physicist, meteorologist, psychologist and pacifist who pioneered modern mathematical techniques of weather forecasting. He did this by first casting the governing equations into finite difference form using numerical methods. Afterwards, while serving with the Quaker ambulance unit in northern France during World War I, he solved the finite difference
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 of the atmosphere from one state to the next.”

Although Bjerknes did not mention it the same statement is true regarding forecasts of oceanic “weather”, that is, the growth and fate of meanders, jets and eddies. It is therefore not surprising that as the power and capacity of computers grew\(^6\) it became common also to use computers and numerical methods to solve oceanographic problems. Indeed, the capacity and power of today’s computers are now amenable to forecast the oceanic weather, at least for limited areas. Therefore also NOWP emerged in the late 1960s.

Inherently the atmosphere and the oceans form a coupled system exchanging momentum, heat and moisture. This was early on recognized in climate modeling, and hence even the very first climate models were coupled atmosphere-ocean models (Edwards, 2011). With the ever growing capacity of computers also coupled models are developed for use in forecasting atmosphere and oceans today (e.g., Warner et al., 2010). Thus in the not too distant future coupled models will be the forecasting model for both oceans and atmosphere. In light of this I strongly believe that 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 oceanographic and meteorological models.

As a consequence of the fact that most processes in the ocean and atmosphere are highly nonlinear more and more research in the atmosphere, ocean and climate related fields relies on sometimes large and monstrous computer codes. It is a growing concern that as a rule many of these codes, commonly written in the programming language FORTRAN, are written and amended by scientists who are not necessarily skilled programmers. This concern is further corroborated by the statistical survey published 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. The results produced may even look reasonable, but in reality they may be totally false, and possibly lead to wrong conclusions. In fact there exist examples in the 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 work and some do not for a specific problem. Likewise it is important to early on to acquire knowledge about how to ensure the quality of your computations. To give some insight into the the procedures whereby the quality may be as-

---

\(^6\)The growth in computer power and capacity is almost exponential since the 1940s.
sured I have added some notes on what is commonly referred to as quality assurance procedures, sometimes also referred to as validation (Appendix B).

As alluded to above many of the numerical methods we apply to solve atmospheric and oceanic problems were historically first applied and developed to solve atmospheric problems. The reason is that, although the thermodynamics of atmosphere and oceans are different, the dynamics are very similar, and hence most of the numerical methods we use to solve atmospheric problems also work for solving oceanographic problems. 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. To fully appreciate and understand the coupled system, and its modeling, one therefore needs to have knowledge of meteorology as well as oceanography.

Though the objective is to give insight into the fundamental numerical methods to solve oceanographic and atmospheric problems, it should be kept in mind that there are indeed numerical methods and techniques that are unique to each sphere. In particular this concern numerical methods relevant to solve the thermodynamic part of the two spheres. This is beyond the scope of this booklet, and additional literature should be consulted if applications of numerical models including thermodynamic processes are essential in your work.

Finally, it is worth emphasizing 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 theory behind the numerical methods employed without knowing how to implement them and run them on the computer, not to mention how to visualize them. Once we have used theory of numerical methods we need to instruct the computer how to perform the calculations. This is done by programming, that is, to write the instructions in some “language” that the computer understands. These instructions are commonly referred to as “the model code”. The common programming language used today in most atmospheric and ocean modeling is FORTRAN. Thus I have attached a brief introduction to FORTRAN programming as an appendix (Appendix A). To familiarize oneself with programming the reader is therefore encouraged to solve as many as possible of the computer problems contained in the accompanying booklet “Computer problems”. For the same reason I also encourage the reader to solve the exercises given at the end of each chapter.

This booklet is the 8th edition compiled for the 2015 version of the course GEF4510 at the Meteorology and Oceanography Section, Department of Geosciences, University of Oslo. Note that it will be amended and corrected as the lectures proceed. It is therefore not complete before the end of the course. To keep track of the changes a list of revisions is included on page ix before the main body of the text.

Blindern August 20, 2015
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, Norwegian Meteorological Institute 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.

☞ October 20, 2015: The sections up to and including Section 6.2 of Chapter “The shallow water equations” starting on page 93 is updated.

☞ September 16, 2015: Chapter “Diffusion” starting on page 33 is

☞ September 9, 2015: Chapter “Time marching problems” starting on page 33 is updated. Changed formulations and straightened misprints

☞ September 1, 2015: Section “Preface” starting on page iii updated by adding footnotes on Ragnar Fjørtoft and Arnt Eliassen.

☞ June 16, 2015: Section “Preface” starting on page iii is rewritten by adding some historical notes about John von Neumann and Lewis F. Richardson. Also Section "Acknowledgement" on page vii is updated.
# Contents

Preface ........................................ iii
Acknowledgements ........................... vii
Revision history ............................ ix

1 INTRODUCTION ............................ 1
1.1 The governing equations .............. 4
1.2 Boundary and initial conditions ...... 5
1.3 The hydrostatic approximation ...... 7
1.4 The Boussinesq approximation ...... 8
1.5 The shallow water equations ......... 9
1.6 The quasi-geostrophic equations .... 11

2 PRELIMINARIES ......................... 15
2.1 General PDEs .......................... 15
2.2 Elliptic equations .................... 16
2.3 Parabolic equations.................. 16
2.4 Hyperbolic equations ................. 17
2.5 Boundary conditions .................. 18
2.6 Taylor series and expansions ....... 20
2.7 Finite difference approximations ..... 22
2.8 Truncation errors ...................... 24
2.9 Notations ............................... 24
2.10 Orthogonal functions ................. 27
2.11 Fourier series ........................ 29
2.12 Fourier transforms .................. 29

3 TIME MARCHING PROBLEMS .......... 33
3.1 Advection-diffusion .................... 34
3.2 Diffusion ................................ 35
3.3 Advection ............................... 37
3.4 Shallow water equations ............. 38
4 THE DIFFUSION PROBLEM
4.1 Finite difference form .................................................. 42
4.2 Numerical stability ...................................................... 45
4.3 Stability analysis ......................................................... 46
4.4 Necessary stability ...................................................... 49
4.5 Explicit and implicit schemes ......................................... 50
4.6 DuFort-Frankel ............................................................ 52
4.7 Crank-Nicholson .......................................................... 54
4.8 A direct elliptic solver ................................................... 56

5 THE ADVECTION PROBLEM 63
5.1 Finite difference form .................................................. 64
5.2 The leapfrog scheme ..................................................... 67
5.3 The CFL condition ....................................................... 67
5.4 The upstream scheme .................................................... 69
5.5 The diffusive scheme .................................................... 70
5.6 The Lax-Wendroff scheme .............................................. 71
5.7 The semi-Lagrangian scheme .......................................... 73
5.8 The implicit scheme ..................................................... 77
5.9 The initial problem in CTCS schemes ................................ 78
5.10 Physical interpretation .................................................. 79
5.11 Numerical dispersion ................................................... 79
5.12 Unphysical solutions .................................................... 82
5.13 The Asselin filter ......................................................... 84
5.14 Numerical diffusion ..................................................... 86
5.15 Flux correction .......................................................... 88

6 THE SHALLOW WATER PROBLEM 93
6.1 Linearization ............................................................. 96
6.2 Linear, non-rotating ..................................................... 98
6.3 Staggered grids ........................................................... 106
6.4 Linear and rotating ..................................................... 109
6.5 Non-linear and rotating ................................................ 115

7 OPEN BOUNDARY CONDITIONS 123
7.1 Definition ................................................................. 126
7.2 Radiation conditions .................................................... 127
7.3 Implementation .......................................................... 130
7.4 The sponge ............................................................... 132
7.5 Flow relaxation ........................................................ 134
7.6 Weakly reflective ......................................................... 138
8 GENERAL VERTICAL COORDINATES
8.1 Transformation ............................................. 142
8.2 Governing equations ...................................... 144
8.3 Terrain-following .......................................... 146

9 TWO-DIMENSIONAL PROBLEMS ............................... 149
9.1 Diffusion equation ........................................... 149
9.2 Advection equation ......................................... 150
9.3 Shallow water equations ................................. 152

10 ADVANCED TOPICS ........................................... 157
10.1 Higher order schemes ................................... 157
10.2 Combined advection-diffusion ............................ 162
10.3 Non-linear instability .................................... 165
10.4 Smoothing and filtering ................................... 168
10.5 The spectral method ...................................... 169

A Introduction to Fortran 2003 via examples .................. 174
A.1 Why use Fortran? ........................................... 174
A.2 Historical background .................................... 175
A.3 The Fortran syntax ........................................... 176
  A.3.1 Data types in Fortran .................................. 177
A.4 The structure of Fortran .................................... 178
  A.4.1 Declaration of variables ............................... 178
  A.4.2 Instructions .............................................. 180
A.5 Sample programs ........................................... 181
  A.5.1 A daynumber converter ............................... 181
  A.5.2 A temperature converter .............................. 184
  A.5.3 A more user friendly version of the converter program .............................. 185
  A.5.4 Variable types, arrays, loops and memory allocation .............................. 187
  A.5.5 File input/output or I/O ................................ 190
  A.5.6 Multidimensional arrays .............................. 195
  A.5.7 Functions and Subroutines ............................ 199
A.6 Modules .................................................... 207

B Quality assurance procedures ................................ 208
B.1 Introduction .................................................. 208
B.2 Sub-grid scale parameterizations and spectral cutoffs .... 209
B.3 What is a good model ...................................... 211
  B.3.1 Tuned, transportable, and robust models .............. 211
  B.3.2 The concept of a good model ........................... 212
B.4 Quality assurance procedures .............................. 213
  B.4.1 Model verification ...................................... 213
List of Figures

1.1 The equation of state for the ocean. Dotted curves show isolines of density $\rho$ as a function of salinity (horizontal axis) and potential temperature (vertical axis) for a fixed pressure (here $p = 0$ Pa). Numbers on curves indicate density in $\sigma_t$ 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$^3$. 6

2.1 Displayed is a commonly used grid when employing numerical methods to solve PDEs. The points in the $x, y$ directions are incremented by $\Delta x, \Delta y$, respectively, so that there are a total of $J$ points along the $x$-axis and $K$ points along the $y$-axis. The points are counted by using the dummy counters $j, k$. 25

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. 43

4.2 Displayed are solutions of the diffusion equation using the scheme (4.4) 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 $\theta$ 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 (4.34) is violated for $K = 0.55$, but not for $K = 0.45$. 49

5.1 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. 72
5.2 Comparison of the numerical solution to the advection equation (5.1) using the leapfrog, the upwind and the Lax-Wendroff schemes with a Courant number of one half ($C = 0.5$). The solution is shown after 10 cycles (periodic boundary condition). The true solution is a bell function as shown by the black solid curve. We note that both the leapfrog and the Lax-Wendroff schemes give rise to numerical dispersion, while the upwind scheme gives rise to numerical diffusion.

5.3 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 right in $x,t$ space as given by (5.48). If $x = L$ marks the end of the computational domain, then all information about the initial condition is lost for times $t > t_c$.

5.4 Sketch of the method of characteristics. The distance between the grid points are $\Delta t$ in the vertical and $\Delta x$ in the horizontal direction. The sloping solid line is the characteristic through the point $j, n + 1$. It is derived from (5.48) 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 $x_{j-1}$.

5.5 Numerical dispersion for the leapfrog scheme. The curves depicts the numerical phase speed as a function of the wavenumber based on (5.72) for various values of the Courant number $C = u\Delta t/\Delta x$. The vertical axis indicates the phase speed $c$ normalized by the advection speed $u$. The horizontal axis indicates the wavenumber normalized by $\pi/\Delta x$ where $\Delta x$ is the space increment or the grid size. The analytic dispersion curve is just a straight line corresponding to the phase speed $c = u$, that is $c/u = 1$. Note that 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 wave numbers 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 with wavelengths shorter than $4\Delta x$ the energy is propagating in the opposite direction of the waves.

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.

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. The green dashed lines are after 400 time steps while the blue, dash-dot lines are after 800 time steps. Cyclic boundary conditions are employed. (cf. Computer problem No. 6 in the Computer Problem notes).
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 \( \rho_0 \).

6.2 Sketch of the semi-Lagrangian technique for a non-linear and rotating case. The distance between the grid points are \( \Delta t \) in the vertical and \( \Delta x \) in the horizontal direction. The sloping solid, blue line is the positive characteristic through the grid point \( j, n + 1 \) (marked +), while the solid, red line is the negative characteristic through the same grid point (marked −). They are derived from (6.68) and (6.69) and the slopes are respectively given by ±\( c_0 \). 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 \( x_{j+1} \).

6.3 Comparison of the structure of (a) an unstaggered and (b) a staggered grid in one spatial dimension. The circles are associated with \( \phi \)-points, while the ellipses are associated with \( u \)-points. The illustrated staggering in panel (b) is such that the \( \phi \)-points and \( u \)-points are one half grid distance apart, but at the same time level.

6.4 Sketch of the semi-Lagrangian technique for a non-linear and rotating case. The distance between the grid points are \( \Delta t \) in the vertical and \( \Delta x \) 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 (6.171). The last characteristic with slope \( u \geq 0 \) the point labeled \( u \) is the dotted black line derived from (6.174). 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 \( x_{j+1} \).

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.

7.2 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.

7.3 Sketch of the FRS zone, the interior domain and the computational domain. Also shown are the appropriate indices.
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.

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 $\Delta x$. 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.

A.1 The punched card

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).
Chapter 1

INTRODUCTION

Numerical methods is one of the most fascinating contemporary tools to solve meteorological and oceanographic problems. The rationale is that most, if not all, of the processes in the atmosphere and ocean are highly non-linear, and thus governed by a set of highly non-linear partial differential equations (PDFs). Equally important is that computers allows us to solve this set of PDFs using numerical methods within a reasonable time frame. We emphasize that numerical methods is in fact the only method whereby the full non-linear equations governing the motion of the atmosphere and ocean can be solved. Only in special cases, and mostly for problems reduced to linear cases, is it possible to solve the governing equations 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.

It is therefore not surprising that atmospheric scientists and weather forecasters embraced this tool early on. In fact when the first digital computers were developed in the 1940s a whole new science field developed in meteorology commonly referred to as numerical weather prediction (NWP). Among the first to embrace the new tool were Jules Charney, Ragnar Fjørtoft, and John von Neumann in the late 1940’s. In 1950 (Charney et al., 1950) they published results on the very first attempt of producing a weather forecast using a (simplified) numerical atmospheric model. Since then NWP has been one of the major science fields pushing the computer technology to its very limits.

An important basis for the rapid development of the NWP in the 1950s and 1960s was the deterministic paradigm already stated by Vilhelm Bjerknes\(^1\) in his famous 1904 paper entitled “Das Problem der Wettervorhersage, betrachtet vom Standpunkte der Mechanik und der Physik” (Bjerknes, 1904). Here he states:

\(^1\)Vilhelm Friman Koren Bjerknes (14 March 1862 - 9 April 1951) was a Norwegian physicist and meteorologist who did much to found the modern practice of weather forecasting.
“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 of the atmosphere from one state to the next.”

Although Bjerknes didn’t mentioned it the same statement is true regarding forecasts of oceanic “weather”, that is, the growth and fate of meanders, jets and eddies. It is therefore not surprising that as the power and capacity of computers grew$^2$ it became common also to use computers and numerical methods to solve oceanographic problems. Indeed, the capacity and power of today’s computers are now amenable to forecast the oceanic weather, at least for limited areas. Thus, since the 1990s a whole new science field has emerged referred to as numerical ocean weather prediction (NOWP).

I believe that in the not too distant future we will experience the appearance of the first coupled atmosphere-ocean forecasting models. In fact, the coupling of the model modules and the coupling techniques necessary to do the job are already to a certain extent developed within the climate modeling community. However, much work is still required to adapt these tools for coupled NWP/NOWP systems. In light of this I strongly believe that 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 oceanographic and meteorological models.

Moreover, to acquire this knowledge and insight I also strongly believe that it is necessary to get some “hands-on-experience”. With this in mind these Lecture notes are accompanied by a set of Computer problems, or exercises to be solved, while reading these Lecture notes. The objective of the combination is to give the reader a basis for evaluating the soundness of the various numerical methods implemented in a particular numerical atmosphere and/or ocean model, whether it is a simple model constructed to study a particular process or a full blown NWP/NOWP model.

As a consequence of the fact that most processes in the ocean and atmosphere are highly non-linear more and more research in the atmosphere, ocean and climate related fields relies on sometimes large and monstrous computer codes. It is a growing concern that as a rule many of these codes, commonly written in the programming language FORTRAN, are written and amended by scientists who are not necessarily skilled programmers. This concern is further corroborated by the statistical survey published 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 programmers 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. The results produced may even look reasonable, but in reality they may be totally false, and possibly lead to wrong conclusions. In fact there exist examples in the

$^2$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 work and some don’t for a specific problem.

Historically many of the numerical methods we apply to solve atmospheric and oceanic problems were first applied and developed to solve atmospheric problems. The reason is that, although the thermodynamics of atmosphere and oceans are different, the dynamics are very similar, and hence most of the numerical methods we use to solve atmospheric problems also work for solving oceanographic problems. 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. To fully appreciate and understand the coupled system, and its modeling, one therefore needs to have knowledge of meteorology as well as oceanography, a fact perhaps most evident within climate modeling today.

Though the objective is to give insight into the fundamental numerical methods to solve oceanographic and atmospheric problems, it should be kept in mind that there are indeed numerical methods and techniques that are unique to each sphere. In particular this concern numerical methods relevant to solve the thermodynamic part of the two spheres. This is beyond the scope of these Lecture notes, and additional literature should be consulted if applications of numerical models are essential in your work.

It is worth reemphasizing 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 theory behind the numerical methods employed without knowing how to implement them and run them on the computer, not to mention how to visualize them. Thus the reader is not only encouraged to solve the computer problems contained in an accompanying Computer problem notes, but also to solve the exercises given at the end of each chapter. Finally I would like to add that even though the notes are directed towards graduate students in meteorology and oceanography, it is my hope that also mature scientists will find these notes useful in their professional life. We concern ourselves with the fundamental tools needed to understand how we put oceans and atmospheres on computers. Specifically we limit ourselves to develop 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. In this we assume that the reader has little or no prior knowledge of or experience in solving differential equations numerically. We therefore explain the finite difference methods we use in detail.

The advection-diffusion equation and the shallow water equations belongs to a class of equations known as partial differential equations. 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 $\rho$, and (potential) temperature $\theta$. 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

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

Here we use $\partial_t$, $\partial_x$, $\partial_y$, and $\partial_z$ 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 $\mathbf{F}_M$ and vector $\mathbf{F}_i$ represents fluxes due to turbulent mixing of momentum and tracers, respectively, $\Omega$ is the Earth’s rotation rate, $g$ is the gravitational acceleration and $S_i$ is the tracer source, if any. Finally, we use $\nabla$ to denote the three-dimensional del-operator defined by

\[
\nabla = \hat{i} \partial_x + \hat{j} \partial_y + \hat{k} \partial_z, \quad (1.5)
\]
INTRODUCTION

1.2 Boundary and initial conditions

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 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,

$$p = \rho R \theta$$

where $R$ is the gas constant\(^6\). 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 $\theta$ 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.

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 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

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

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

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

\(^6\)R = 287.04 Jkg\(^{-1}\)K\(^{-1}\)
Freezing point of sea water as a function of salinity

Figure 1.1: The equation of state for the ocean. Dotted curves show isolines of density $\rho$ as a function of salinity (horizontal axis) and potential temperature (vertical axis) for a fixed pressure (here $p = 0$ Pa). Numbers on curves indicate density in $\sigma_t$ 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$^3$.

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,

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

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, \( \frac{Dw}{dt} \), is small in comparison to, e.g., the gravitational acceleration \( \rho g \). 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,

\[
\partial_t (\rho w) + \nabla \cdot (\rho v w) = -\partial_z p - \rho g - \nabla \cdot (\rho F_M^V),
\]

(1.10)

where \( F_M^V \) is the vertical vector component of the mixing tensor \( F_M \). 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. 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

\[
\partial_z p = -\rho g,
\]

(1.11)

which is the hydrostatic equation. 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)

\[
\partial_t (\rho u) + \nabla_H \cdot (\rho u u) + \partial_z (\rho w u) + \rho f \mathbf{k} \times \mathbf{u} = -\nabla_H p + \partial_z \tau - \nabla_H \cdot (\rho F_M^H),
\]

(1.12)

where \( f = 2\Omega \sin \phi \) is the Coriolis parameter where \( \phi \) is the latitude and \( \Omega \) is the Earth’s rotation rate. \( F_M^H \) and \( \tau \) are, respectively, the horizontal and vertical component of the three-dimensional flux tensor \( F_M \) due to turbulent mixing. \( \tau \) 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

\[
\partial_t (\rho C_i) + \nabla_H \cdot (\rho C_i u) + \partial_z (\rho C_i w) = -\partial_z (\rho F_V^i) - \nabla_H \cdot (\rho F_M^H) + \rho S_i \quad i = 1, 2, \ldots,
\]

(1.13)

---

7We 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.

8As 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.

9The 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 \(^{10}\), 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 4.2), 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

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

\(^{10}\)Also referred to as the individual derivative
where $\rho_0$ is a reference density. Similarly the tracer conservation equation (1.3) reduces to
\[ \partial_t C_i + \nabla \cdot (C_i \mathbf{v}) = -\nabla \cdot \mathbf{F}_i + S_i, \]
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
\[ \nabla_H \cdot \mathbf{u} + \partial_z w = 0, \]
(1.19)
\[ \partial_t \mathbf{u} + \nabla_H \cdot (\mathbf{uu}) + \partial_z (w\mathbf{u}) + f_k \times \mathbf{u} = -\rho_0^{-1} \nabla_H \rho + \rho_0^{-1} \partial_z \mathbf{\tau} - \nabla_H \cdot (\mathbf{F}_H^M), \]
(1.20)
\[ \partial_z p = -\rho g, \]
(1.21)
\[ \partial_t C_i + \nabla_H \cdot (C_i \mathbf{u}) + \partial_z (C_i w) = -\partial_z \mathbf{F}_V^V - \nabla_H \cdot \mathbf{F}_i^H + S_i ; \quad i = 1, 2, \cdots, \]
(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\(^{11}\). We will return to elliptic solvers in Section 4.8 on page 56 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 $\rho_0$ 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
\[ \nabla_H \cdot \mathbf{u} + \partial_z w = 0 \]
(1.23)
\[ \partial_z p = -\rho_0 g, \]
(1.24)
\[ \partial_t \mathbf{u} + \nabla_H \cdot (\mathbf{uu}) + \partial_z (w\mathbf{u}) = -f_k \times \mathbf{u} - \rho_0^{-1} \nabla_H \rho + \rho_0^{-1} \partial_z \mathbf{\tau} - \nabla_H \cdot \mathbf{F}_M^H, \]
(1.25)
\(^{11}\)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.,

$$ p = p_s + g\rho_0(\eta - z) $$

(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 $\eta$ 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,

$$ \partial_t \mathbf{U} + \nabla_h \cdot (\frac{\mathbf{U}\mathbf{U}}{h}) + f \mathbf{k} \times \mathbf{U} = -gh\nabla_h(h - H) + \rho_{0}^{-1}(\mathbf{\tau}_s - \mathbf{\tau}_b) + \mathbf{X}, $$

(1.27)

$$ \partial_t h + \nabla_h \cdot \mathbf{U} = 0, $$

(1.28)

where $\mathbf{U} = \int_{-H}^{\eta} \mathbf{u} dz$ is the volume flux of fluid through the fluid column of height/depth $h = \eta + H$, $\mathbf{\tau}_s$ and $\mathbf{\tau}_b$ 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

$$ \int_{-H}^{\eta} \nabla_h \cdot (\mathbf{uu}) dz \approx \nabla_h \cdot \left( \frac{\mathbf{UU}}{h} \right) $$

(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 $\bar{\mathbf{u}} = \mathbf{U}/h$. Replacing $\mathbf{U}$ by $\bar{\mathbf{u}}$ the shallow water equations become

$$ \partial_t (h\bar{\mathbf{u}}) + \nabla_h \cdot (h\bar{\mathbf{u}}\bar{\mathbf{u}}) + f \mathbf{k} \times h\bar{\mathbf{u}} = -gh\nabla_h(h - H) + \rho_{0}^{-1}(\mathbf{\tau}_s - \mathbf{\tau}_b) + \mathbf{X}, $$

(1.30)

$$ \partial_t h + \nabla_h \cdot (h\bar{\mathbf{u}}) = 0, $$

(1.31)

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

$$ \partial_t (h\bar{\mathbf{u}}) + \nabla_h \cdot (h\bar{\mathbf{u}}\bar{\mathbf{u}}) = h(\partial_t \bar{\mathbf{u}} + \bar{\mathbf{u}} \cdot \nabla_h \bar{\mathbf{u}}) + \bar{\mathbf{u}}[\partial_t h + \nabla_h \cdot (h\bar{\mathbf{u}})] $$

$$ = h(\partial_t \bar{\mathbf{u}} + \bar{\mathbf{u}} \cdot \nabla_h \bar{\mathbf{u}}), $$

(1.32)

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

$$ \partial_t \bar{\mathbf{u}} + \bar{\mathbf{u}} \cdot \nabla_h \bar{\mathbf{u}} + f \mathbf{k} \times \bar{\mathbf{u}} = -g\nabla_h \eta + \frac{\mathbf{\tau}_s - \mathbf{\tau}_b}{\rho_{0}(H + \eta)} + \frac{\mathbf{X}}{(H + \eta)}, $$

(1.33)

$$ \partial_t \eta + \nabla_h \cdot [(H + \eta)\bar{\mathbf{u}}] = 0, $$

(1.34)
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

\[
\frac{D_H u}{dt} + f \mathbf{k} \times u = -g \nabla_H h, \quad (1.35)
\]

\[
\frac{1}{h} \frac{D_H h}{dt} + \nabla_H \cdot u = 0, \quad (1.36)
\]

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,

\[
\frac{D_H}{dt} = \partial_t + u \cdot \nabla_H, \quad (1.37)
\]

If we furthermore assume that the acceleration \( D_H u / dt \) is small compared to the Coriolis acceleration, that is, assume that the Rossby number

\[
R \equiv \frac{|D_H u / dt|}{|f \mathbf{k} \times u|} \ll 1, \quad (1.38)
\]

the momentum equation (1.35) reduces to

\[
f \mathbf{k} \times u = -g \nabla_H h. \quad (1.39)
\]

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

\[
u = \frac{g}{f} \mathbf{k} \times \nabla_H h. \quad (1.40)
\]

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,

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

$$\zeta = k \cdot \nabla_H \times u.$$  \hspace{1cm} (1.42)

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

$$\frac{D_H}{dt} \left( \frac{\zeta + f}{h} \right) = 0.$$  \hspace{1cm} (1.43)

Here $\zeta + f$ is the absolute vorticity, while $(\zeta + f)/h$ is the potential vorticity for a barotropic fluid\textsuperscript{12}. If we let $L$ be a typical lateral (horizontal) scale of $u$, so that $|u \cdot \nabla_H u| \sim |u|^2/L$, 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

$$\frac{|u|}{fL} \ll 1.$$  \hspace{1cm} (1.44)

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

$$\frac{||\zeta||}{f} = \frac{|u|}{fL}.$$  \hspace{1cm} (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

$$h - H_m \sim \frac{fL}{g} |u| \quad \text{or} \quad \frac{h - H_m}{H_m} \sim RF^2,$$  \hspace{1cm} (1.46)

where $H_m$ is the mean layer thickness and

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

where $L_R = g H_m / f^2$ is the Rossby radius of deformation. If we now assume $F \sim O(1)$ or less, which is tantamount to assuming that $L$ is not large compared to Rossby’s deformation radius,\textsuperscript{12}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.
the quasi-geostrophic equations

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

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

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

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

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

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

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

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

\[
\zeta = \frac{g}{f} \nabla^2_H h, \tag{1.51}
\]

and then

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

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.
1.6 The quasi-geostrophic equations
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

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

(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 \( \hat{a}, \hat{b}, \ldots, \hat{g} \) 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 \( \theta \) 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
2.2 Elliptic equations

PRELIMINARIES

\( H_m \), that is, \((h - H_m)/H_m \ll 1\). We then get

\[
\begin{align*}
\partial_t u - f v &= -g \partial_x h \\
\partial_t v + f u &= -g \partial_y h \\
\partial_t h + H_m \partial_x u &= -H_m \partial_y v.
\end{align*}
\]

(2.2)

(2.3)

(2.4)

Here \( u, v \) are the components of the horizontal velocity \( u \) along the axes \( x, y \) respectively. We may further simplify these equations by assuming that the motion is one-dimensional in space by letting \( \partial_y = 0 \). Mathematically speaking this is just three equation containing the three dependent variables \( u, v \) and \( h \), and the two independent variables \( t, x \). We note that by some manipulation similar to that detailed in Section 2.4 on page 18 below these three equations may be condensed into one equation, that is,

\[
\partial_t^2 h - g H_m \partial_x^2 h + f^2 h = 0
\]

(2.5)

containing one dependent variable \( h \) only. Furthermore, (2.5) conforms to (2.1) by letting \( \theta = h, x' = t, y' = x, \hat{a} = 1, \hat{c} = -g H_m, \hat{f} = f^2 \) 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 \textit{elliptic}. The classic example is \textit{Poisson’s equation},

\[
\nabla^2_H \phi = \partial_x^2 \phi + \partial_y^2 \phi = g(x, y),
\]

(2.6)

where again \( \nabla_H \) 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 \textit{Helmholtz equation}

\[
\nabla^2_H \phi + f(x, y) \phi = g(x, y),
\]

(2.7)

and the \textit{Laplace equation}

\[
\nabla^2_H \phi = 0.
\]

(2.8)

2.3 Parabolic equations

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

\[
\partial_t \theta = \kappa \partial_x^2 \theta,
\]

(2.9)

where \( \kappa \) 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

$$\partial_t C_i = \nabla \cdot (\mathbf{K} \cdot \nabla C_i),$$  

(2.10)

where the diffusive tracer flux $F_i$ is parameterized as $F_i = -\mathbf{K} \cdot \nabla C_i$ where in turn $\mathbf{K}$ is a matrix (dyade) describing the conductive efficiency of the medium with regard to the tracer $C_i$ (cf. Section 3.1). Thus $\mathbf{K} = \kappa_{mn} \hat{i}_m \hat{j}_n$, $m, n = 1, 2, 3$. 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 $\kappa$ is constant.

Let us for a moment assume that the atmosphere/ocean is at rest ($v = 0$) 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,

$$\partial_t^2 \phi - c_0^2 \partial_x^2 \phi = 0.$$  

(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

$$\Phi = \partial_t \phi - c_0 \partial_x \phi$$  

(2.12)

we may rewrite the wave equation (2.11) to get

$$\partial_t \Phi + c_0 \partial_x \Phi = 0.$$  

(2.13)

Since $c_0$ is a constant (2.13) may be written

$$\partial_t \Phi + \partial_x (c_0 \Phi) = 0.$$  

(2.14)

We observe that (2.14), commonly referred to as the *advection equation*, is a one-dimensional version of (1.1) with $\rho$ replaced by $\Phi$ and $v$ replaced by $c_0 i$. 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,

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

Next we substitute the results into (2.4) to get

\[
(\partial_t^2 + f^2 - gH_m \nabla^2_H)\partial_t h = 0. \tag{2.17}
\]

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

\[
(\partial_t^2 + f^2 - gH_m \nabla^2_H)h' = 0. \tag{2.18}
\]

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

\[ \mathbf{n} \cdot \mathbf{v} = 0 \]  \hspace{1cm} (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

\[ \mathbf{n} \cdot \mathbf{F_\theta} = 0, \]  \hspace{1cm} (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

\[
\nu \partial_z u = C_D u ; \quad z = -H,
\]

(2.22)

where \( \nu \) is the vertical eddy viscosity, \( u \) 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 \( u \) is the horizontal velocity component then (2.22) requires that \( u \) 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

\[
C(x, t) = C(x + L, t),
\]

(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 derivatives exist and are continuous\(^2\). One characteristic of a good function is that it can always be expanded in a so called 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\(^3\)

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

(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 \( \phi \) 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 $\phi$ depends on more than one variable. Hence:

$$\phi_{x+\Delta x,y,z}^t = \phi_{x,y,z}^t + \sum_{n=1}^{\infty} \frac{1}{n!} (\partial^n \phi)_{x,y,z}^t \Delta x^n,$$  \hfill (2.25)

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

$$\phi_{y+\Delta y,x,z}^t = \phi_{y,x,z}^t + \sum_{n=1}^{\infty} \frac{1}{n!} (\partial^n \phi)_{y,x,z}^t \Delta y^n,$$  \hfill (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 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 $\theta$ at the neighboring point $x + \Delta x$. Thus,

$$\theta_{x+\Delta x}^t = \theta_{x}^t + \partial_x \theta_{x}^t \Delta x + \frac{1}{2} \partial_x^2 \theta_{x}^t \Delta x^2 + \frac{1}{6} \partial_x^3 \theta_{x}^t \Delta x^3 + O(\Delta x^4),$$  \hfill (2.27)

where, following (2.25), $\theta$ and all its 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 $O(\Delta x^4)$ - order of $\Delta x$ 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 $\Delta x$. If we solve (2.27) with respect to the first derivative we get

$$\partial_x \theta_{x}^t = \frac{\theta_{x+\Delta x}^t - \theta_{x}^t}{\Delta x} + O(\Delta x).$$  \hfill (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 $\Delta x$ by $-\Delta x$ in (2.27) to get

$$\theta_{x-\Delta x}^t = \theta_{x}^t - \partial_x \theta_{x}^t \Delta x + \frac{1}{2} \partial_x^2 \theta_{x}^t \Delta x^2 - \frac{1}{6} \partial_x^3 \theta_{x}^t \Delta x^3 + O(\Delta x^4).$$  \hfill (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 derivative we get

$$\partial_x \theta_{x}^t = \frac{\theta_{x}^t - \theta_{x-\Delta x}^t}{\Delta x} + O(\Delta x).$$  \hfill (2.30)

Moreover, by subtracting (2.29) from (2.27), and solving for $\partial_x \theta_{x}^t$ we get

$$\partial_x \theta_{x}^t = \frac{\theta_{x}^t + \Delta x - \theta_{x}^t - \Delta x}{2\Delta x} + O(\Delta x^2).$$  \hfill (2.31)

\footnote{The notation $\phi_{x,y,z}^t$ 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 $\Delta x$ is arbitrary, we may equally well choose $2\Delta x$. Thus replacing $\Delta x$ by $2\Delta x$ in the Taylor series (2.27) and (2.29) we get,

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

and

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

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

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

### 2.7 Finite difference approximations

To derive at possible finite difference approximations (FDAs) to the various derivatives of our PDEs we actually utilize the Taylor series above. For instance to derive possible FDAs to the first derivative $\partial_x \theta$, that is, the first derivative of $\theta$ with respect to $x$, we utilize the expressions (2.27), (2.29) and (2.31) above in which $\Delta x$ is a finite distance (nonzero). For instance using (2.31) we simply neglect the higher order terms, in this case terms of $O(\Delta x^2)$, and get

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

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

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

and

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

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 approximation, while (2.36) and (2.37) are denoted a one-sided approximations. 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, one-sided approximation while (2.37) is referred to as a backward one-sided approximation. Another difference between the centered and the forward and/or backward FDAs is the order of the neglected terms. While the centered difference is valid to second order, that is, $O(\Delta x^2)$, the two one-sided FDAs are only valid to first order, that is, $O(\Delta x)$. Consequently the centered FDA (2.35) is also sometimes

---

\[5\text{Henceforth an FDA is denoted by brackets. Thus the FDA approximation to } \partial_x \theta|_{x}^{t} \text{ is denoted } [\partial_x \theta|_{x}^{t}].\]
referred to as a second order, centered approximation, while the one-sided FDAs (2.36) and (2.37) are referred to as first order, forward and/or backward approximations, respectively.

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

\[
\theta\big|_{x}^{t+\Delta t} = \theta\big|_{x}^{t} + \partial_{t}\theta\big|_{x}^{t} \Delta t + \frac{1}{2} \partial_{t}^{2}\theta\big|_{x}^{t} \Delta t^{2} + \frac{1}{6} \partial_{t}^{3}\theta\big|_{x}^{t} \Delta t^{3} + \mathcal{O}(\Delta t^{4}),
\]

(2.38)

\[
\theta\big|_{x}^{t-\Delta t} = \theta\big|_{x}^{t} - \partial_{t}\theta\big|_{x}^{t} \Delta t + \frac{1}{2} \partial_{t}^{2}\theta\big|_{x}^{t} \Delta t^{2} - \frac{1}{6} \partial_{t}^{3}\theta\big|_{x}^{t} \Delta t^{3} + \mathcal{O}(\Delta t^{4}).
\]

(2.39)

To construct a centered FDA to the time rate of change of \( \theta \) we simply subtract (2.39) from (2.38) and solve with respect to \( \partial_{t}\theta\big|_{x}^{t} \) to obtain

\[
\partial_{t}\theta\big|_{x}^{t} = \frac{\theta\big|_{x}^{t+\Delta t} - \theta\big|_{x}^{t-\Delta t}}{2\Delta t} + \mathcal{O}(\Delta t^{2}),
\]

(2.40)

which when dropping terms of \( \mathcal{O}(\Delta t^{2}) \) gives the FDA

\[
\left[\partial_{t}\theta\right]_{x}^{t} = \frac{\theta\big|_{x}^{t+\Delta t} - \theta\big|_{x}^{t-\Delta t}}{2\Delta t}.
\]

(2.41)

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

Similarly we may construct approximations to higher order derivatives. For instance to find a centered FDA to \( \partial_{x}^{2}\theta\big|_{x}^{t} \), we first simply add the two Taylor expansion (2.27) and (2.29) and solve with respect to \( \partial_{x}^{2}\theta\big|_{x}^{t} \) to get

\[
\partial_{x}^{2}\theta\big|_{x}^{t} = \frac{\theta\big|_{x}^{t+\Delta x} - 2\theta\big|_{x}^{t} + \theta\big|_{x-\Delta x}^{t}}{\Delta x^{2}} + \mathcal{O}(\Delta x^{2}).
\]

(2.42)

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

\[
\left[\partial_{x}^{2}\theta\right]_{x}^{t} = \frac{\theta\big|_{x}^{t+\Delta x} - 2\theta\big|_{x}^{t} + \theta\big|_{x-\Delta x}^{t}}{\Delta x^{2}}.
\]

(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 were of \( \mathcal{O}(\Delta x) \). Thus the centered approximations appear to 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 \( \partial_{x}^{3}\theta\big|_{x}^{t} \) we combine (2.27) and (2.29) with (2.32) and (2.33) to obtain

\[
\partial_{x}^{3}\theta\big|_{x}^{t} = \frac{\theta\big|_{x}^{t+2\Delta x} - 2\theta\big|_{x}^{t+\Delta x} + 2\theta\big|_{x-\Delta x}^{t} - \theta\big|_{x-2\Delta x}^{t}}{2\Delta x^{3}} + \mathcal{O}(\Delta x^{2}),
\]

(2.44)

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

\[
\left[\partial_{x}^{3}\theta\right]_{x}^{t} = \frac{\theta\big|_{x}^{t+2\Delta x} - 2\theta\big|_{x}^{t+\Delta x} + 2\theta\big|_{x-\Delta x}^{t} - \theta\big|_{x-2\Delta x}^{t}}{2\Delta x^{3}}.
\]

(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 $O(\Delta x^2)$ when using the centered FDA, the terms we neglected when using the one-sided approximation was $O(\Delta x)$. 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 truncation error in that the series is truncated when making the FDA. 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 $O(\Delta x^n)$ 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 $\partial_x^2 \theta|_x$ 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 157).

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 $J$ points along the $x$-axis and $K$ points along the $y$-axis. To count the points we use dummy indices $j = 1, 2, 3, \cdots, J$ along the $x$-axis and $k = 1, 2, 3, \cdots, K$ 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 $j = J$. Similarly we associate the point $y = 0$ with $k = 1$ and the point $y = L$ with $k = K$. The $j$th point along the $x$-axis is then $x = x_j$ where the subscript refers to the value for $x$ at the $j$th point along the $x$-axis.

\[^{6}\text{Mathematically this can be expressed by } x \in <0, L> \text{ and } y \in <0, L>\]
Similarly we let \( y = y_k \) be associated with the \( k \)th point along the \( y \)-axis. The coordinates of the grid junctions are then given by \( x_j, y_k \).

Let us denote the distance between two adjacent points along the \( x \)-axis by \( \Delta x \) and the distance between two adjacent points along the \( y \)-axis by \( \Delta y \). Then the \( j \)th point along the \( x \)-axis is denoted

\[
x_j = (j - 1)\Delta x,
\]

Figure 2.1: Displayed is a commonly used grid when employing numerical methods to solve PDEs. The points in the \( x, y \) directions are incremented by \( \Delta x, \Delta y \), respectively, so that there are a total of \( J \) points along the \( x \)-axis and \( K \) points along the \( y \)-axis. The points are counted by using the dummy counters \( j, k \).
while the \( k \)th 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 \( x_J = y_K = L \). We also notice for later convenience that the latter gives

\[
\Delta x = L/(J - 1), \quad \Delta y = L/(K - 1), \tag{2.48}
\]

respectively\(^8\). It is also common to use the notation \( \theta_{jk} \) 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

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

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 \( \Delta t \) the time step or time increment. Then the time at the \( n \)'th time level is defined by\(^9\).

\[
t^n = n\Delta t, \quad n = 0, 1, 2, \ldots \tag{2.53}
\]

from which follows that

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

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

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

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

\[
\theta^n_{jkl} = \theta(x_j, y_k, z_l, t^n), \tag{2.56}
\]

\(^{\text{7}}\)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 \( x_1 = x_0, y_1 = y_0, x_J = x_0 + L \) and \( y_K = y_0 + L \).

\(^{\text{8}}\)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 \( \Delta y = L/K \).

\(^{\text{9}}\)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 only one level).
where $z_l = (l - 1)\Delta z$.

As an example let us consider the Taylor series expansions (2.27) and (2.29). Using the preceding notation we get

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

(2.57)

and

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

(2.58)

and hence that the forward in space FDA to the first derivative is written

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

(2.59)

while the second order, centered approximation is written

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

(2.60)

Similarly follows that the second order, centered FDA to the second derivative with the above notation is written

$$
[\partial^2_x\theta]_j^n = \frac{\theta^n_{j+1} - 2\theta^n_j + \theta^n_{j-1}}{\Delta x^2}.
$$

(2.61)

Finally we remark that the increments $\Delta x, \Delta y, \Delta z$ and $\Delta t$ 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

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

(2.62)

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 (2.62) 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 $\phi$ is cyclic and that the initial value is known.
Since \( \phi \) is a good function it may be expanded in terms of an infinite set of orthogonal functions \( e_n(x) \), where \( n = 1, 2, 3, \ldots \). Thus

\[
\phi = \sum_{n=-\infty}^{\infty} \varphi_n(t)e_n(x),
\]

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

\[
\int_{-L}^{L} e_n(x)e^*_m(x)dx = \begin{cases} 1 & n = m \\ 0 & n \neq m \end{cases},
\]

where \( e^*_m(x) \) 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 (2.62) by \( e^*_m \), and then integrate over all possible \( x \)-values, to give

\[
\int_{-L}^{L} \partial_t \phi(x,t)e^*_m(x)dx = \int_{-L}^{L} \mathcal{H}[\phi]e^*_m(x)dx.
\]

The left-hand side is further developed by use of (2.63) and (2.64) to give

\[
\int_{-L}^{L} \partial_t \phi(x,t)e^*_m(x)dx = \int_{-L}^{L} \left( \sum_n \partial_t \varphi_n e_n \right)e^*_m dx = \sum_n \partial_t \varphi_n \int_{-L}^{L} e_n e^*_m dx = \sum_n \partial_t \varphi_n.
\]

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

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

Using these results we get

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

We thus have a set of coupled, ordinary differential equations for the time rate of change for the expansion coefficients \( \varphi_n \).

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 \( \lambda_m \) are the eigenvalues. Equation (2.68) then becomes

\[
\partial_t \varphi_m = \lambda_m \varphi_m \quad ; \quad \forall m
\]

and becomes decoupled.

\(^{10}\)Note that the expansion functions \( e_n(x) \) are in general complex functions, e.g., \( e_n(x) = e^{i\alpha_n x} \) where \( \alpha_n \) 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.
2. If the original equation is

\[ G[\partial_t \phi] = H[\phi] \]  

where \( G \) is a linear operator, then our problem is simplified by using expansion functions that are eigenfunctions of \( G \) with eigenvalues \( \lambda_n \). We then have,

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

2.11 Fourier series

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

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

The series (2.72) is called a Fourier series and the expression

\[ \varphi_n(t) e^{i\alpha_n x} \]  

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

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

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 \( \alpha_n \) as well as time.

2.12 Fourier transforms

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

\[ \tilde{\phi}(\alpha) = \int_{-\infty}^{\infty} \phi(x) e^{-i\alpha x} dx. \]  

We observe that \( \tilde{\phi} \) is a complex function consisting of a real as well as an imaginary part. As is common we refer to \( \tilde{\phi} \) as the Fourier transform of the real function \( \phi \). Furthermore we notice that the \( \tilde{\phi} \) is a continuous functions of the wavenumber \( \alpha \in [-\infty, +\infty] \). Hence if we know the

\(^{12}\)In the above problem with cyclic boundary conditions at \( x = \pm L \) the wavenumbers are \( \alpha_n = n\pi/L \).
Fourier transform the original real function $\phi$ is retrieved from the real part of the inverse Fourier transform defined by

$$\phi(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \tilde{\phi}(\alpha)e^{i\alpha x} d\alpha. \quad (2.76)$$

We observe that the “expansion coefficient” $\tilde{\phi}$ now is a continuous function of the wavenumber $\alpha \in [-\infty, +\infty]$, and that the summation in (2.72) is replaced by an integral. We may also plot the Fourier transform $\tilde{\phi}$ as a function of $\alpha$. In that case the space spanned by $\tilde{\phi}$ and $\alpha$ is called the Fourier space and its distribution the Fourier spectrum.

As revealed by (2.76) the Fourier transform (2.75) is simply the amplitude associated with the wave of wavenumber (or wavelength) $\alpha$. 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 (2.76) 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

$$[\partial_x^4 \theta]_j = \frac{\theta_{j+2} - 4\theta_{j+1} + 6\theta_j - 4\theta_{j-1} + \theta_{j-2}}{\Delta x^4}, \quad (2.77)$$

and that the truncation error is $O(\Delta x^2)$. 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

$$\tilde{\partial_x \theta} = i\alpha \tilde{\theta} \quad \text{and} \quad \tilde{\partial_x^2 \theta} = -\alpha^2 \tilde{\theta}, \quad (2.78)$$

respectively, where the notation $\tilde{\psi}$ denotes the Fourier transform of $\psi$. 
5. Show by making use of the results in Exercise 4 that a formal analytic solution to the diffusion equation

\[ \partial_t \theta = \kappa \partial_x^2 \theta, \quad (2.79) \]

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

\[ \theta = \begin{cases} 
0 & ; \quad x \to +\infty, -\infty \\
\theta_0 e^{-\frac{(x)^2}{4}} & ; \quad t = 0
\end{cases} \quad (2.80) \]

is

\[ \theta = \frac{a \theta_0}{2\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-a^2(\frac{1}{4}a^2+\kappa t)} e^{iax} d\alpha \quad (2.81) \]
Chapter 3

TIME MARCHING PROBLEMS

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 4 to find the state of the sphere in question 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.

A particular example, 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. 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 4. 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 11. 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 the respectively diffusion problem, the advection problem and the shallow water equations by use of numerical methods. In Chapter 10 (Section 10.2 on page 162) 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.

3.1 The advection-diffusion equation

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

\[
\partial_t \theta + \nabla \cdot \mathbf{F} = 0,
\]

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

\[
\mathbf{F} = \mathbf{F}_A + \mathbf{F}_D
\]

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

\(^1\)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 an important part of the thermodynamics. In contrast the mechanical energy concerns the motion of the fluid and is thus part of the fluid dynamics.
advective flux vector $F_A$ depends on the motion only. The advective flux of the property $\theta$ therefore follows the path of the individual fluid parcels. Thus its parameterization, or mathematical formulation, becomes

$$F_A = v\theta.$$  \hspace{1cm} (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, commonly referred to as turbulence closure, is diffusion. Its mathematical formulation is,

$$F_D = -\mathcal{K} \cdot \nabla \theta,$$  \hspace{1cm} (3.4)

where $\mathcal{K}$ 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 $\theta$ 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.

### 3.2 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.

$$\partial_t \theta = -\nabla \cdot F_D = \nabla \cdot (\mathcal{K} \cdot \nabla \theta).$$  \hspace{1cm} (3.5)

where the last equal sign follows by use of (3.4) for the diffusive flux. Assuming that $\mathcal{K} = \kappa \mathbb{I}$, where $\kappa$ is constant, we get

$$F_D = -\kappa \partial_x \theta,$$  \hspace{1cm} (3.6)

which substituted into (3.5) becomes

$$\partial_t \theta = \kappa \nabla^2 \theta.$$  \hspace{1cm} (3.7)

which is a parabolic problem (cf. eq. 2.9 on page 16). 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

---

\(^2\)Its original formulation is due to a Dr. Adolf Eugen Fick who in 1855 formulated the parameterization $F_D = -\kappa \nabla \theta$ with $\kappa$ being a constant and a property of the medium.
3.2 Diffusion

Flux vector indeed have this property\(^3\). The noisiness of a field is commonly measured by its variance. A measure of this variance is \(\theta^2\), that is, the square of the tracer concentration. We may thus investigate whether the noise increases or decreases by analyzing the time rate of change of \(\theta^2\).

To arrive at an equation for the time rate of change of the variance we thus first multiply (3.5) by the tracer concentration \(\theta\) itself to get

\[
\partial_t \theta^2 = -2\theta \nabla \cdot \mathbf{F}_D. \quad (3.8)
\]

The left hand side of (3.8) is the time rate of change of the variance. Let us assume that (3.5) and by implication (3.8) are valid within a fixed (in time and space) volume \(V\) bounded by the surface \(\Omega\). We then get the time rate of change of the total variance within the volume \(V\) by integrating (3.8) over the total volume \(V\). Performing the integration we get

\[
\partial_t \left( \int_V \theta^2 dV \right) = -2 \int_\Omega \theta \mathbf{F}_D \cdot \delta \sigma + 2 \int_V \mathbf{F}_D \cdot \nabla \theta dV. \quad (3.9)
\]

Here the vector \(\delta \sigma = n \delta \sigma\) where \(n\) is a unit vector directed along the outward normal to the surface \(\Omega\) and \(\delta \sigma\) is an infinitesimal surface element. To derive (3.9) we have also used the Gauss theorem. At the boundary \(\Omega\) 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 \(\Omega\), while in the latter case we let \(n \cdot \mathbf{F}_D = 0\) at the surface \(\Omega\). In either case we observe that the first term on the right-hand side of (3.9) is zero. Hence (3.9) reduces to

\[
\partial_t \left( \int_V \theta^2 dV \right) = 2 \int_V \mathbf{F}_D \cdot \nabla \theta dV. \quad (3.10)
\]

Thus if

\[
\mathbf{F}_D \cdot \nabla \theta \leq 0 \quad (3.11)
\]

the right-hand side of (3.10) is negative, and we get

\[
\partial_t \left( \int_V \theta^2 dV \right) \leq 0. \quad (3.12)
\]

Equation (3.12) shows that as long as (3.11) is satisfied the diffusion term indeed acts to even out any noise in the \(\theta\) field. We notice that (3.11) 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

\[
\mathbf{F}_D \cdot \nabla \theta = -\kappa (\nabla \theta)^2 \leq 0, \quad (3.13)
\]

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.

\(^3\)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.
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:

“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 165).

### 3.3 Advection

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

$$\partial_t \theta = -\nabla \cdot F_A = -\nabla \cdot (v \theta).$$

Equation (3.14) 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 $\Omega$, and for all times $t \in [0, \infty]$\(^5\). On the surface $\Omega$ 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 $F_A = v \theta$, and let the boundary condition at the surface $\Omega$ be such that $F_A \cdot \delta \sigma = 0$\(^6\). Then by performing the same operation on (3.14) as we did in Section 3.2 we find that the total variance becomes

$$\partial_t \left( \int_V \theta^2 dV \right) = 2 \int_V F_A \cdot \nabla \theta dV = \int_V v \cdot \nabla \theta^2 dV = - \int_V \theta^2 \nabla \cdot v dV. \quad (3.15)$$

\(^4\)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).

\(^5\)In practice we have to limit the computation to a finite time span

\(^6\)This is achieved by assuming $v = 0$ or $v \cdot \delta \sigma = 0$, that is, no flow across the boundary.
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 (3.15) is zero and hence any disturbances creating a variance in $\theta$ 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.

### 3.4 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 9. Neglecting the forcing terms on the right-hand side of (1.25) we get

\[
\nabla_H \cdot \mathbf{u} + \partial_z w = 0, \quad (3.16)
\]

\[
\partial_t \mathbf{u} + \nabla_H \cdot (\mathbf{u} \mathbf{u}) + \partial_z (w \mathbf{u}) + f \mathbf{k} \times \mathbf{u} = -g \nabla_H \eta, \quad (3.17)
\]

where we have used (1.26) to substitute for the pressure

\textsuperscript{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,

\[
\partial_t \left( \int_V u^2 dV \right). \quad (3.18)
\]

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

\[
\int_V u^2 dV = \int_V 2e_K dV = 2E_K, \quad (3.19)
\]

where $E_K$ is the total kinetic energy\textsuperscript{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, $E_K \geq 0$.

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

\textsuperscript{8}To be precise $E_K$ is the total kinetic energy per unit density, but since we have assumed a uniform density, that is, $\rho_0 = \text{constant}$, it is common to refer to $E_K$ as the total kinetic energy.
To find an equation for the time rate of change of the kinetic energy we first multiply (3.17) by \( u \). We then get
\[
\frac{\partial t}{e_K} + \nabla_H \cdot (e_K u) + \partial_z (e_K w) = -g u \cdot \nabla_H \eta \tag{3.20}
\]
where we have made use of the fact that
\[
u \cdot \left[ \nabla_H \cdot (uu) + \partial_z (w u) \right] = \nabla_H \cdot (e_K u) + \partial_z (e_K w). \tag{3.21}
\]
The latter follows by use of the continuity equation (3.16). Finally we note that the contribution from the Coriolis term vanishes since \( \mathbf{u} \cdot (k \times \mathbf{u}) = \mathbf{k} \cdot (\mathbf{u} \times \mathbf{u}) = 0 \). Next we integrate (3.20) over the fixed volume \( V \) to get
\[
\frac{\partial t}{E_K} = C, \tag{3.22}
\]
where
\[
C = -\int_V g u \cdot \nabla_H \eta dz. \tag{3.23}
\]
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 note that the so called available gravitational potential energy (AGPE) per unit density (Lorenz, 1955; Røed, 1997, 1999) is defined by
\[
E_\Phi = \int_V g z dV - \int_{V_0} g z dV = \int_A \left( \int_{-H}^\eta g z dz - \int_{-H}^0 g z dz \right) dA = \int_A \frac{1}{2} g \eta^2 dA, \tag{3.24}
\]
where \( A \) is the projected area of the volume \( V \) onto a horizontal surface. Note that \( E_\Phi \) is defined by subtracting the potential energy of the initial state\(^9\) from the potential energy in an arbitrary later state. The rationale is that the enormous amount of potential energy of the initial state is not available for release into kinetic energy and thus of no interest\(^11\).

We next show that
\[
C = -\partial_t E_\Phi, \tag{3.25}
\]
To this end we first note that (3.23) may be written
\[
C = -g \int_A \left( \int_{-H}^\eta \mathbf{u} \cdot \nabla_H \eta dz \right) dA, \tag{3.26}
\]
Noting that \( \eta \) and \( \nabla_H \eta \) are independent of depth/height we get
\[
\int_{-H}^\eta \mathbf{u} \cdot \nabla_H \eta dz = -\nabla_H \eta \cdot \left( \int_{-H}^\eta \mathbf{u} dz \right) = -\eta \nabla_H \cdot \left( \int_{-H}^\eta \mathbf{u} dz \right) + \nabla_H \cdot \left( \int_{-H}^\eta \eta \mathbf{u} dz \right). \tag{3.27}
\]
\(^9\)The forcing terms leads only to external source or sink terms, that is, irreversible energy conversion terms irrelevant for the present presentation
\(^10\)The initial state is defined as one at rest, and in static equilibrium, that is, \( \mathbf{u}(x, y, z, 0) = 0 \) and \( \eta(x, y, 0) = 0 \).
\(^11\)Although it is common to refer to \( E_\Phi \) as the available potential energy (APE), we refer to it as AGPE since we concerns ourselves with the potential energy due to gravity only.
The first term on the right-hand side of the second equality in (3.27) may be developed further by integrating the continuity equation (3.16) from bottom to top and using the kinematic boundary condition (1.7) and (1.9) (cf. Section 1.2 on page 5). Thus we get

$$\nabla_H \cdot \left( \int_{-H}^{\eta} u \, dz \right) = -\partial_t \eta,$$

(3.28)

and hence (3.27) becomes

$$\int_{-H}^{\eta} u \cdot \nabla_H \eta \, dz = \eta \partial_t \eta + \nabla_H \cdot \left( \int_{-H}^{\eta} \eta u \, dz \right).$$

(3.29)

Substituting (3.29) into (3.26), noting that the second term on the right-hand side of (3.29) is a flux term that vanishes upon integration over the area $A$, we retrieve (3.25).

Thus $C$ is able to convert potential energy into kinetic energy or vice versa. It therefore constitutes a *reversible* transfer of energy between the two mechanical energy forms. This is perhaps best illustrated if we substitute (3.25) into (3.22) to get

$$\partial_t (E_K + E_\Phi) = 0.$$

(3.30)

Since $E_K + E_\Phi$ is the total (mechanical) energy, (3.30) shows that the total mechanical energy is conserved. Thus if $E_K$ experience an increase (decrease) there is a similar decrease (increase) in the AGPE. We therefore conclude that under the assumptions of no external exchange of energy (a consequence of neglecting the forcing terms in eq. 3.17), 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 this latter property should be retained, that is, should reflect that the total kinetic energy is conserved when constructing our numerical schemes (cf. *Arakawa and Lamb*, 1977).
Chapter 4

THE DIFFUSION PROBLEM

In this chapter we present finite difference methods to solve the diffusion equation by numerical means. To this end we consider the diffusion equation in its simplest form. Hence we show how to solve the one-dimensional (in space) diffusion equation. Additionally we assume that the diffusive flux can be parameterized as a down the gradient diffusion and that the diffusion (mixing) coefficient is uniform in time and space. Under these circumstances the diffusion equation (3.5) in its continuous form reads,

$$\partial_t \theta = \kappa \partial_x^2 \theta,$$

(4.1)

where \( \theta \) can be any dependent variable (e.g., potential temperature, humidity, speed, salinity, etc.), and \( \kappa \) is the diffusion coefficient.

As alluded to in Section 2.3 on page 16, 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 86) 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 an infinite domain the tracer values becomes infinitely small, but will cover an infinite domain. In summary, the diffusion process transfers properties from one location to the next by conduction that acts to diminish differences in a distribution so that the end result is a much smoother field.

An obvious example of a diffusion process is the turbulent mixing of heat in both atmospheres and oceans. Then \( \theta \) appearing in (4.1) is the potential temperature. Another classic atmosphere-ocean example is the so called Ekman problem in which \( \theta \) 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.
4.1 Finite difference equation

Let us for instance consider lateral turbulent mixing of heat in the atmosphere and ocean, and let the dependent variable \( \theta \) describe the deviation (or anomaly) of the potential temperature away from a given mean temperature profile, say zero degree Celsius. Then \( \kappa \) appearing in (4.1) is the strength of the turbulent mixing (her considered a constant), while the independent variables \( x, t \) are respectively the horizontal (or lateral) coordinate and time. 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, \( \theta(0, t) = \theta(L, t) = 0^\circ C \forall t \). Thus \( \theta(x, 0) = f(x) \), where \( f(x) \) is a known function for \( x \in [0, L] \).

Our task is to find, by numerically solving (4.1), how the anomaly evolves in time between the two end points \( x = 0, L \). We note that by considering that \( \theta = 0^\circ C \) 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 \( \theta(x, 0) = 0^\circ C; \forall x \), that is, that there exists at least one position in space where \( f(x) \neq 0^\circ C \).

To find a numerical solution to (4.1) we follow the notation in Section 2.9. Thus we first divide the intervals \( x \in [0, L) \) and \( t \in (0, T) \), where \( T \) is some finite time, into respectively \( J \) and \( N \) sections of width \( \Delta x \) and \( \Delta t \), 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 43).

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

\[
[\partial_t \theta]^n_j = \frac{\theta^{n+1}_j - \theta^n_j}{\Delta t} ; \quad [\partial^2_x \theta]^n_j = \frac{\theta^{n+1}_j - 2\theta^n_j + \theta^{n-1}_j}{\Delta x^2}.
\]

By substituting the expressions (4.2) into (4.1) we get

\[
\frac{\theta^{n+1}_j - \theta^n_j}{\Delta t} = \kappa \frac{\theta^{n+1}_{j+1} - 2\theta^n_j + \theta^{n+1}_{j-1}}{\Delta x^2} ; \quad \begin{cases} j = 2(1) J \\ n = 0(1) N \end{cases}
\]

(4.3)

Solving with respect to \( \theta^{n+1}_j \) we finally get

\[
\theta^{n+1}_j = \theta^n_j + \frac{\kappa \Delta t}{\Delta x^2} \left( \theta^{n+1}_{j+1} - 2\theta^n_j + \theta^{n+1}_{j-1} \right) ; \quad \begin{cases} j = 2(1) J \\ n = 0(1) N \end{cases}
\]

(4.4)

Note that (4.3) and (4.4) 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. In numerical language they are

\[
\theta^0_j = f_j ; \quad j = 1(1) J + 1 \quad \text{and} \quad \theta^n_1 = \theta^n_{J+1} = 0 ; \quad n = 0(1) N,
\]

(4.5)

To find \( \theta \) at the first time level \( n = 1 \) we substitute \( n = 0 \) into (4.4). We then get

\[
\theta^1_j = \theta^0_j + \frac{\kappa \Delta t}{\Delta x^2} \left( \theta^0_{j+1} - 2\theta^0_j + \theta^0_{j-1} \right) ; \quad j = 2(1) J.
\]

(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.

Thus for the first “wet” point \( j = 2 \)

\[
\theta_2^1 = \theta_2^0 + \frac{\kappa \Delta t}{\Delta x^2} (\theta_3^0 - 2\theta_2^0 + \theta_1^0). \tag{4.7}
\]

We note that \( \theta_1^0, \theta_2^0 \) and \( \theta_3^0 \) on the right-hand side of (4.7) are known from the boundary and/or initial conditions (4.5). We may then proceed to evaluate \( \theta_3^1, \theta_4^1, \ldots \) up to and including \( \theta_J^1 \). For the last wet point \( j = J \) we get in particular

\[
\theta_J^1 = \theta_J^0 + \frac{\kappa \Delta t}{\Delta x^2} (\theta_{J+1}^0 - 2\theta_J^0 + \theta_{J-1}^0), \tag{4.8}
\]

where again \( \theta_{J-1}^0, \theta_J^0 \) and \( \theta_{J+1}^0 \) on the right-hand side are known from the initial and/or boundary conditions (4.5). 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, \( \theta_j^1 \). Note that at the boundaries \( j = 1 \) and \( J + 1 \) \( \theta \) is known from the boundary condition (4.5). This reflects the well known property of differential equations, whether they are PDEs or ordinary differential
4.1 Finite difference form

THE DIFFUSION PROBLEM

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 (4.6) together with the boundary conditions gives us \( \theta \) for all \( j = 1(1)J \) at time level \( n = 1 \).

We may then proceed to compute \( \theta \) at time level \( n = 2 \). We do this by substitution of \( n = 1 \) into (4.4). We then get

\[
\theta_j^2 = \theta_j^1 + \frac{\kappa \Delta t}{\Delta x^2} \left( \theta_{j+1}^1 - 2\theta_j^1 + \theta_{j-1}^1 \right) ; \quad j = 2(1)J.
\]

(4.9)

Having thus found \( \theta \) 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 note that since \( x_{J+1} = L = J \Delta x \) we cannot choose \( J \), \( L \) and \( \Delta x \) independently. Once two of them are chosen the third is given by the formula

\[
J = \frac{L}{\Delta x}.
\]

(4.10)

Likewise follows that

\[
N = \frac{T}{\Delta t}
\]

(4.11)

showing that \( N \), \( \Delta t \) and \( T \) also depend on each other.

Since we know \( \theta \) at time \( t = 0 \) (or time level \( n = 0 \)), the application of a forward, one-sided finite difference approximation in time, as for instance employed in (4.4), is the obvious choice. The accuracy of this scheme is thus \( O(\Delta t) \), while the application of a centered finite difference approximation in space makes the spatial accuracy higher, namely \( O(\Delta x^2) \). To increase the accuracy to the same level in time we may for instance employ a centered in time scheme for the time rate of change as well, that is, let

\[
[\partial_t \theta]^n_j = \frac{\theta_j^{n+1} - \theta_j^{n-1}}{2\Delta t}.
\]

(4.12)

Substitution of (4.12) into (4.1) then gives is

\[
\frac{\theta_j^{n+1} - \theta_j^{n-1}}{2\Delta t} = \frac{\theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n}{\Delta x^2} ; \quad \begin{cases} j = 2(1)J \\ n = 0(1)N \end{cases} ;
\]

(4.13)

or

\[
\theta_j^{n+1} = \theta_j^{n-1} + \frac{2\Delta t}{\Delta x^2} \left( \theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n \right) ; \quad \begin{cases} j = 2(1)J \\ n = 0(1)N \end{cases}.
\]

(4.14)

To obtain the solution at the first time level \( n = 1 \), that is, to obtain \( \theta_j^1 \) we again substitute \( n = 0 \) into (4.14) which gives

\[
\theta_j^1 = \theta_j^{-1} + \frac{2\Delta t}{\Delta x^2} \left( \theta_{j+1}^0 - 2\theta_j^0 + \theta_{j-1}^0 \right) ; \quad j = 2(1)J,
\]

(4.15)

which requires knowledge of \( \theta_j^{-1} \). 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 the Sections 4.2 - 4.5 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.
4.2 Numerical stability

Another additional problem with the scheme (4.14) is that it is *numerically unstable*. This means that the numerical solution, instead of following the continuous solution, steadily deviates from it. Commonly this happens explosively 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 model. For our numerical solution to have any legitimacy we must require that it is 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 \( \theta \) may be written as a sum of cosines and sines or even more compact as a sum of exponentials (see for instance Section 2.11 on page 29 or Lighthill, 1970, page 3)

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

(4.16)

where \( \alpha_m \) is the wavenumber of the \( m \)'t Fourier component. Each component in (4.16), that is,

\[
\theta_m = \Theta_m e^{i\alpha_m x},
\]  

(4.17)

is called a Fourier component. Here \( \Theta_m(t) \) is the time dependent amplitude of the \( m \)'t component. Substituting (4.16) into (4.1) we obtain

\[
\partial_t \Theta_m = -\kappa \alpha_m^2 \Theta_m.
\]  

(4.18)

Note that we have dropped the summation, that is, we analyze each Fourier component separately. We observe that (4.18) is an ordinary differential equation (ODE). Solving it with respect to \( \Theta \) we get

\[
\Theta_m = \Theta_m^0 e^{-\kappa \alpha_m^2 t}.
\]  

(4.19)

Here \( \Theta_m^0 \) is the initial amplitude of mode \( m \), that is, the value of \( \Theta_m \) at time \( t = 0 \). We find these initial amplitudes by expanding the initial distribution of \( \theta \) into a Fourier series, that is,

\[
\theta(x, 0) = \sum_{m=-\infty}^{\infty} \Theta_m^0 e^{i\alpha_m x}
\]  

(4.20)

Thus substituting (4.19) into (4.16) we get

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

(4.21)
4.3 Stability analysis

which is then the analytic solution to (4.1). We note by looking at (4.19) that the amplitude $\Theta_m$ of each individual Fourier component decreases monotonically and exponentially as time increases. Furthermore, we observe that the shortest waves (highest wave numbers) decrease faster than the amplitude of the longer waves (low wave numbers). This is in accord with Section 3.2 where we concluded, based on (3.12), that diffusion acts to smooth out disturbances. Moreover we learn from (4.21) 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 longer period motion.

As is obvious we would like the numerical solution in general 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 4.3 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

$$
|\Theta_n - \Theta_0| \leq B,
$$

(4.22)

where $\Theta_0$ is the initial amplitude of the $\theta$. 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.

4.3 Stability analysis: von Neumann’s method

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 (4.17), that is,

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

(4.23)

where $\Theta_n$ is the discrete amplitude at time level $n$ and $\alpha$ is the wavenumber of that particular discrete mode\(^1\). We now define a growth factor

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

(4.24)

Thus $G$ is the amplification of the amplitude $\Theta$ as we proceed from one time level to the next. We observe that (4.24) is formally similar to (4.22), 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

\(^1\)Note that we have dropped the subscript $m$ on $\Theta$ and $\alpha$ for clarity.
THE DIFFUSION PROBLEM 4.3 Stability analysis

level \( n \), while (4.22) is the ratio between the value at a random time level and the initial value. Letting \( n = 0 \) in (4.24) then gives

\[
\Theta_1 = G \Theta_0, \quad (4.25)
\]

where \( \Theta_0 \) is the initial amplitude. By letting \( n = 1 \) in (4.24) and making use of (4.25) we obtain

\[
\Theta_2 = G \Theta_1 = G^2 \Theta_0. \quad (4.26)
\]

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

\[
\Theta_l = G^l \Theta_0. \quad (4.27)
\]

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 (4.22) is satisfied if

\[
|G| \leq 1, \quad (4.28)
\]

since then \( G^l \) decreases as the time level or time increases\(^2\). The criterion (4.28) 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 4.4 below.

As our first example we use von Neumann’s method to analyze the forward in time, centered in space scheme for the diffusion as given by (4.4). Substituting (4.23) into (4.4) we get

\[
\Theta_{n+1} = \Theta_n + \frac{\kappa \Delta t}{\Delta x^2} (e^{ia\Delta x} - 2 + e^{-ia\Delta x}) \Theta_n \quad (4.29)
\]

where we have divided by the common factor \( e^{iaj\Delta x} \). Noting that \( e^{ia\Delta x} + e^{-ia\Delta x} = 2 \cos \alpha \Delta x \) we get

\[
\Theta_{n+1} = \left[ 1 - 2 \frac{\kappa \Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) \right] \Theta_n. \quad (4.30)
\]

We thus find the growth factor by simply dividing (4.30) by \( \Theta_n \),

\[
G = 1 - 2 \frac{\kappa \Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x). \quad (4.31)
\]

To satisfy (4.28) we observe that

\[
-1 \leq 1 - 2 \frac{\kappa \Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) \leq 1. \quad (4.32)
\]

Since \( 0 \leq (1 - \cos \alpha \Delta x) \leq 2 \) the right-hand side inequality is satisfied for all \( \Delta t \) and \( \Delta x \). The inequality on the left-hand side, however, is satisfied if and only if

\[
\frac{\kappa \Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) \leq 1. \quad (4.33)
\]

\(^2\)Confer Computer Problem No. 1
Recall that $0 \leq (1 - \cos \alpha \Delta x) \leq 2$, and hence (4.33) is satisfied for all wavenumbers $\alpha$ if
\[
\frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{2} \quad \text{or} \quad \Delta t \leq \frac{\Delta x^2}{2\kappa}.
\] (4.34)

This condition also ensures that (4.32) is satisfied, and hence that von Neumann’s condition (4.28) is satisfied as well. Furthermore (4.34) tells us that we cannot choose $\Delta x$ and $\Delta t$ independently. Once $\Delta x$ is chosen the time step $\Delta t$ must be chosen in accord with (4.34). We therefore say that the forward in time, centered in space scheme (4.4) is conditionally stable under the condition (4.34).

We also observe from (4.33) that the waves that first violate the inequality are waves with wavenumbers given by
\[
\cos \alpha \Delta x = -1,
\] (4.35)
which correspond to those waves that maximizes $1 - \cos \alpha \Delta x$. The wavenumber satisfying (4.35) are
\[
\alpha_m \Delta x = (2m - 1)\pi; \quad m = 1, 2, \ldots
\] (4.36)
with corresponding wavelengths
\[
\lambda_m = \frac{2\pi}{\alpha_m} = \frac{2\Delta x}{2m - 1}.
\] (4.37)

The most dominant of these waves is the wave corresponding to $m = 1$. Thus the most unstable wave has wavelength
\[
\lambda_1 = 2\Delta x.
\] (4.38)
This implies that the numerical instability will appear as “$2\Delta x$” noise, that is, noise of wavelength $2\Delta x$, and commonly as a saw tooth pattern as displayed in Figure 4.2.

We mentioned that the forward in time, centered in space scheme (4.4) for the diffusion equation is a conditionally stable scheme. If no such condition can be found that makes the chosen FD approximation satisfies the von Neumann’s condition (4.28), then we say that the scheme is unconditionally unstable. If von Neumann’s condition is satisfied regardless of our choice of spatial and temporal increments (e.g., $\Delta x$ and $\Delta t$), then we say that 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 (4.24) that $|\Theta_{n+1}| < |\Theta_n|$. Thus, inherent for all schemes for which $|G| < 1$ is that they include artificial numerical energy dissipation\(^3\). We emphasize that even if the physical problem does not exhibit energy dissipation the numerical solution may exhibit such a decrease. 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.

\(^3\)In this context energy dissipation means that the amplitude of the solution decreases in time.
4.4 Necessary stability

Figure 4.2: Displayed are solutions of the diffusion equation using the scheme (4.4) 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 $\theta$ 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 (4.34) is violated for $K = 0.55$, but not for $K = 0.45$.

For problems that include natural energy dissipation it is therefore important to ensure that the numerical energy dissipation is small compared to the physical dissipation. We therefore always favor neutral schemes ($|G| = 1$), since such schemes are energy conserving, a highly desirable 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 $\Delta t$ that is small enough to satisfy (4.34), but at the same time is large enough to make $\Delta t \sim \Delta x^2 / 2 \kappa$.

4.4 The necessary stability condition

We mentioned above that von Neumann’s condition is a sufficient condition. This implies that if (4.28) 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 determine this we return to the original requirement as formulated in (4.22). Substituting the discrete Fourier component (4.23) into (4.22) then gives

$$\left| \frac{\Theta_{n+1}}{\Theta_0} \right| \leq B \quad \Rightarrow \quad |G|^n \leq B.$$ (4.39)
Taking the natural logarithmic on both sides then gives

\[ n \ln |G| \leq \ln B \equiv B'. \quad (4.40) \]

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 \(\epsilon\) 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 (4.40) we obtain

\[ \epsilon \leq \frac{B'\Delta t}{t^n} = \mathcal{O}(\Delta t). \quad (4.41) \]

Thus the necessary condition that satisfies the numerical stability requirement is

\[ |G| \leq 1 + \mathcal{O}(\Delta t). \quad (4.42) \]

This shows that von Neumann’s condition (4.28) 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 (4.31), is a scalar. For multi-variable and multi-dimensional problems the growth factor will commonly be a tensor or matrix, say \(G\). 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 \(G\) is less than or equal to one.

### 4.5 Explicit and implicit schemes

The schemes (4.4) and (4.14) both contains the variable at time level \(n + 1\) on the left-hand side. In addition all the terms on the right-hand side are 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 6.5 on page 120) we also refer to the scheme as being semi-implicit.

Explicit schemes, as exemplified by (4.4) and (4.14), are always relatively 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 4.1 (page 43). This is in most cases not true for implicit and semi-implicit schemes.

Whether a scheme is explicit or implicit also impacts the stability of the scheme. Let us for instance consider the centered in time, centered in space scheme applied to the diffusion equation (4.1). If we use an explicit scheme we then evaluate all the terms on the right hand-side at the
time level \( n \). Accordingly the scheme becomes the one displayed in (4.14). If we choose to make the scheme implicit we would then evaluate all the terms on the right-hand side at the new time level \( n + 1 \). The scheme then becomes

\[
\theta_{j}^{n+1} = \theta_{j}^{n-1} + \kappa \frac{2\Delta t}{\Delta x^2} (\theta_{j-1}^{n+1} - 2\theta_{j}^{n+1} + \theta_{j+1}^{n+1}) ; \quad \left\{ \begin{array}{ll}
  j = 2(1)J \\
  n = 0(1) \ldots \end{array} \right. 
\]  

(4.43)

Let us first analyze the stability of the similar CTCS explicit scheme (4.14) using von Neumann’s method. Substitution of the Fourier component (4.23) into (4.14), removing the common factor \( e^{i\alpha j \Delta x} \) and using (4.24) then gives

\[
G = G^{-1} - \kappa \frac{4\Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x). 
\]  

(4.44)

Multiplying by \( G \) and rearranging terms yields the equation

\[
G^2 + 2\lambda G - 1 = 0, 
\]  

(4.45)

where

\[
\lambda = \kappa \frac{2\Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) \geq 0, 
\]  

(4.46)

to determine the growth factor. Solving (4.45) with respect to \( G \) we get the two solutions

\[
G_{1,2} = -\lambda \pm \sqrt{1 + \lambda^2}. 
\]  

(4.47)

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

\[
|G_2| = \lambda + \sqrt{1 + \lambda^2} \geq 1, 
\]  

(4.48)

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.


Then what about the implicit scheme (4.43)? To analyze the stability of the scheme (4.43) we first rearrange the terms to obtain

\[
\theta_{j-1}^{n+1} - \left( 2 + \frac{\Delta x^2}{2\kappa \Delta t} \right) \theta_{j}^{n+1} + \theta_{j+1}^{n+1} = -\frac{\Delta x^2}{2\kappa \Delta t} \theta_{j}^{n-1} ; \quad \left\{ \begin{array}{ll}
  j = 2(1)J \\
  n = 0(1) \ldots \end{array} \right. 
\]  

(4.49)

We note in passing that the implicit formulation require us to solve for \( \theta \) at time level \( n + 1 \) at the three grid points \( j-1 \), \( j \), and \( j+1 \) simultaneously. We solve these kind of equations efficiently by employing an elliptic solver as for instance shown in Section 4.8. Curiously enough the implicit formulation of the parabolic diffusion equation turns it into an elliptic numerical equation.
To determine the growth factor we employ von Neumann’s method. Thus substituting the Fourier component into (4.49) we get
\[ G \left[ 2 \cos \alpha \Delta x - \left( 2 + \frac{\Delta x^2}{2 \kappa \Delta t} \right) \right] = -\frac{\Delta x^2}{2 \kappa \Delta t} G^{-1}. \] (4.50)

Multiplying by \( \frac{2 \kappa \Delta t}{\Delta x^2} G \) and solving with respect to the growth factor \( G \) then gives
\[ |G_1| = |G_2| = \frac{1}{\sqrt{1 + \frac{4 \kappa \Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x)}}. \] (4.51)

We observe that \( |G_{1,2}| \) is in fact less than or equal to one for all \( \Delta t \) and \( \Delta x \). Hence the implicit formulation of the CTCS scheme is unconditionally stable. This property is shared by all truly implicit schemes. We emphasize that while this is so for all truly implicit schemes it is not necessarily true for semi-implicit schemes, as for instance exemplified by the Crank-Nicholson scheme treated in Section 4.7. In contrast we recall that the explicit formulation (4.14) gave an unconditionally unstable scheme.

We notice that the implicit formulation (4.43) does not contain any constraint on the time step \( \Delta t \), and hence we may choose \( \Delta t \) to be as long as we wish when considering the numerical stability alone. Nevertheless it is commonly wise to stay away from implicit schemes, in particular for the terms forming the dominant balance in our governing equations. The rationale is, as exemplified by (4.51), that the growth factor \( |G| \) is less than one for all wavelengths longer than the shortest resolved waves of wavelength \( 2 \Delta x \). Moreover, \( |G| \) decreases with increasing time step. Thus the implicit formulation always contains numerical dissipation which increases with increasing time step. Again this is a property shared by all implicit schemes. Although all true implicit schemes are unconditionally stable, we strongly advice to choose a time step \( \Delta t \) so as to bring the growth factor as close to one as possible. The rationale is that we must control the numerical dissipation inherent in all implicit schemes. Thus although implicit schemes have no restrictions on the length of the time step its length is nevertheless constrained by the need to keep the numerical damping as small as possible.

### 4.6 Consistency and inconsistency: DuFort-Frankel

One important aspect about the schemes we use to solve our governing equations numerically are that they should mimic their continuous counterpart. We must therefore require that the employed scheme converges toward the continuous equations in the limit when \( \Delta t \to 0 \) and the space increments \( \Delta x, \Delta y, \Delta z \to 0 \) independently. Note that this requirement is independent of how they go to zero and independent of how fast each of them goes to zero. If this requirement is met we say that the scheme is consistent. If not we say that the scheme is inconsistent. Together with the absolute requirement of numerical stability it forms the two fundamental properties that our schemes should obey.

As is obvious all schemes where the finite difference approximations are based on Taylor series expansions as outlined in Section 2.6 satisfies the consistency requirement. Since both
of the schemes (4.4) and (4.14) are based on a Taylor series expansion, they are both prime examples of consistent schemes.

We may, however, quite easily construct numerical schemes without having to resort to Taylor series expansions. It is particular in these cases that we need to analyze its consistency. If the scheme turns out to be inconsistent we must require as a minimum that the dominant physical processes that the governing equations simulate are consistently represented in the finite difference analogue. One example of such a scheme elaborated below is the DuFort-Frankel scheme.

To construct the DuFort-Frankel scheme we start with the consistent explicit, centered in time, centered in space scheme (4.14). We recall (cf. Section 4.5) that this scheme is unconditionally unstable when applied to the diffusion equation. However, as alluded to, the diffusion term is mostly added to prevent small scale noise to grow, that is, to dissipate energy that accumulates on the smaller scales. In these cases the term does not represent the dominant physics and we may relax on our consistency requirement. To construct the scheme we observe that the value of \( \theta^n_j \) at the grid point \( x_j, t^n \) in space and time can be thought of as a linear interpolation in time of the two adjacent grid points \( x_j, t^{n+1} \) and \( x_j, t^{n-1} \), or

\[
\theta^n_j = \frac{1}{2} (\theta^{n+1}_j + \theta^{n-1}_j). \tag{4.52}
\]

Using this as a substitute for \( \theta^n_j \) on the right-hand side of (4.14) we get

\[
\theta^{n+1}_j = \theta^{n-1}_j + 2\frac{\kappa \Delta t}{\Delta x^2} (\theta^n_{j+1} - \theta^n_{j-1} - \theta^{n+1}_j + \theta^{n-1}_j) \quad ; \quad \left\{ \begin{array}{l} j = 2(1)J \\ n = 0(1)N - 1 \end{array} \right. \tag{4.53}
\]

We first note that the introduction of the term \( \theta^{n+1}_j \) 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. In fact, as shown by Exercise 4 on page 61, by simply adding some implicitness to the centered in time, centered in space scheme we have actually turned it into an unconditionally stable scheme. We note that in contrast to the implicit scheme (4.43), the implicitness is now limited to the single term \( \theta^{n+1}_j \) involving only the space grid point \( x_j \). We may move this term from the right-hand side of (4.53) to its left-hand side. After some rewriting we get

\[
\theta^{n+1}_j = [\theta^{n-1}_j + \chi (\theta^n_{j+1} - \theta^n_{j-1} + \theta^{n-1}_j)] (1 + \chi)^{-1}. \tag{4.54}
\]

where

\[
\chi = \frac{2\kappa \Delta t}{\Delta x^2}. \tag{4.55}
\]

Thus the numerical algorithm associated with the implicit DuFort-Frankel scheme (4.53) is solved explicitly. This is one reason why it has become so popular, in particular in oceanography, (e.g., Adamec and O’Brien, 1978). A second reason, as alluded to, is that it is unconditionally stable.

It remains to analyze the consistency of the scheme. To this end we employ the Taylor series expansions of Section 2.6,

\[
\theta^n_{j\pm 1} = \theta^n_j \pm \partial_x \theta^n_j \Delta x + \frac{1}{2} \partial^2_x \theta^n_j \Delta x^2 \pm \frac{1}{6} \partial^3_x \theta^n_j \Delta x^3 + \mathcal{O}(\Delta x^4), \tag{4.56}
\]
4.7 Crank-Nicholson

and

\[
\theta_j^{n+1} = \theta_j^n + \delta_t \theta_j^n \Delta t + \frac{1}{2} \partial^2_t \theta_j^n \Delta t^2 + \frac{1}{6} \partial^3_t \theta_j^n \Delta t^3 + O(\Delta t^4).
\] (4.57)

Substitution of these series in (4.53) then gives

\[
\partial_t \theta_j^n - \kappa \partial^2_x \theta_j^n = -\varsigma \partial^2_t \theta_j^n + O(\Delta t^2) + O(\Delta x^2).
\] (4.58)

where

\[
\varsigma = \kappa \left( \frac{\Delta t}{\Delta x} \right)^2.
\] (4.59)

To be consistent all the terms on the right-hand side of (4.58) must converge to zero in the limit \(\Delta x \to 0\) and \(\Delta t \to 0\) independently. This is, however, not the case for the first term on the right-hand side which tends to infinity if \(\Delta x\) tends to zero faster than \(\Delta t\). We therefore note that (4.58) only converges to the continuous equation if \(\varsigma \to 0\) when \(\Delta x \to 0\) and \(\Delta t \to 0\). This implies that the scheme is consistent iff \(\Delta t\) tends to zero faster than \(\Delta x\). 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 3.2 the diffusion term is often used as a numerical artifact or “trick” to dissipate energy contained on the smaller scales. Commonly this “noise” is created due to the presence of non-linear terms in the governing equations. This leads to non-linear interaction among the various wavelengths which in turn is responsible for a more or less continuous cascade of energy towards progressively smaller and smaller scales. If we neglect to dissipate the energy contained in the tail of the energy spectrum we get an accumulation of energy at the \(2\Delta x - 4\Delta x\) scales. In turn this accumulation, at some time or another into the integration, 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 this purpose it does not represent any of the physical processes that we want to resolve. Rather it is introduced as a numerical method to avoid our model to blow up. Nevertheless it does represent low energy physics acting on scales smaller than those resolved by our grid\(^5\). It is therefore a parameterization of these processes. 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. Note that the SGS processes act on scales smaller than our grid resolution.

4.7 The Crank-Nicholson scheme

We will now consider another popular scheme called the Crank-Nicholson scheme. Like the Dufort-Frankel scheme it is also 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.

\(^4\)The formulation iff is short for “if and only if.

\(^5\)Note that for a given grid size \(\Delta x\) the resolution equals the Nüquist wavelength \(2\Delta x\).
We start by recalling the two schemes
\[ \theta_j^{n+1} = \theta_j^n + \frac{1}{2} \chi (\theta_{j-1}^n - 2\theta_j^n + \theta_{j+1}^n) \] (4.60)
and
\[ \theta_j^{n+1} = \theta_j^n + \frac{1}{2} \chi (\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1}), \] (4.61)
where
\[ \chi \equiv \frac{2\kappa \Delta t}{\Delta x^2}. \] (4.62)
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 truly implicit scheme. Hence (4.60) is conditionally stable under the condition \( \chi \leq 1 \) 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
\[ \theta_j^{n+1} = \theta_j^n + \frac{1}{2} \chi \left[ \gamma (\theta_{j-1}^{n+1} - 2\theta_j^{n+1} + \theta_{j+1}^{n+1}) + (1 - \gamma)(\theta_{j-1}^n - 2\theta_j^n + \theta_{j+1}^n) \right], \] (4.63)
where \( \gamma \) is a number so that \( 0 \leq \gamma \leq 1 \). If \( \gamma = 0 \) then (4.63) reduces to the explicit scheme (4.60). If \( \gamma = 1 \) then (4.63) reduces to the implicit scheme (4.61). If \( \gamma \) is between 0 and 1 the scheme contains both implicit and explicit terms. Hence the scheme (4.63) 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 61)
\[ G = \frac{1 - (1 - \gamma) \chi (1 - \cos \alpha \Delta x)}{1 + \gamma \chi (1 - \cos \alpha \Delta x)}. \] (4.64)
We recall that the condition \( |G| \leq 1 \) or \(-1 \leq G \leq 1\) is a sufficient condition for numerical stability. From (4.64) follows that \( G \leq 1 \) is always satisfied, while \( G \geq -1 \) is satisfied if
\[ \chi (1 - 2\gamma) \leq 1. \] (4.65)
We note that for \( \frac{1}{2} \leq \gamma \leq 1 \) the left hand side of (4.65) is always negative or zero, implying that (4.65) is automatically satisfied. Under these circumstances the scheme is stable regardless of the value chosen for the increments \( \Delta x \) and \( \Delta t \). Thus the scheme (4.63) 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. 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 (4.65). We note that for \( \gamma = 0 \), in which case (4.63) equals the forward in time, centered in space finite difference approximation for the diffusion equation as displayed in (4.4), we indeed retrieve the condition (4.34) of Section 4.3, that is, \( \chi \leq 1 \).

As a corollary we note that this proves that the truly implicit scheme (4.61), which follows from (4.63) by letting \( \gamma = 1 \), is indeed unconditionally stable.

55
The value $\gamma = \frac{1}{2}$ is special. It constitutes the critical value at which the scheme (4.63) is still unconditionally stable. Substituting this particular value of $\gamma$ into (4.63) we get

$$\theta_{j}^{n+1} = \theta_{j}^{n} + \frac{1}{4} \chi \left[ (\theta_{j-1}^{n+1} - 2\theta_{j}^{n+1} + \theta_{j+1}^{n+1}) + (\theta_{j-1}^{n} - 2\theta_{j}^{n} + \theta_{j+1}^{n}) \right],$$

(4.66)

which is referred to as the Crank-Nicholson scheme.

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 20. By substituting these series into the centered differences on the right-hand side of (4.66) we first get

$$\theta_{j}^{n+1} - \theta_{j}^{n} \Delta t = \frac{1}{2} \kappa \left( \partial_{x}^{2} \theta |_{j}^{n+1} + \partial_{x}^{2} \theta |_{j}^{n} \right) + O(\Delta x^2).$$

(4.67)

Expanding $\theta_{j}^{n+1}$ and $\partial_{x}^{2} \theta |_{j}^{n+1}$ using Taylor series we get

$$\theta_{j}^{n+1} - \theta_{j}^{n} = \partial_{t} \theta |_{j}^{n} + \frac{1}{2} \partial_{x}^{2} \theta |_{j}^{n} \Delta t + O(\Delta t^2),$$

(4.68)

$$\partial_{x}^{2} \theta |_{j}^{n+1} = \partial_{x}^{2} \theta |_{j}^{n} + \partial_{t} (\partial_{x}^{2} \theta) |_{j}^{n} \Delta t + O(\Delta t^2).$$

(4.69)

Substituting these series in (4.67) and rearranging terms we get

$$\partial_{t} \theta |_{j}^{n} = \kappa \partial_{x}^{2} \theta |_{j}^{n} - \frac{1}{2} \left[ \partial_{x}^{2} \theta |_{j}^{n} - \kappa \partial_{t} (\partial_{x}^{2} \theta) |_{j}^{n} \right] \Delta t + O(\Delta t^2) + O(\Delta x^2),$$

(4.70)

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

$$\partial_{x}^{2} \theta |_{j}^{n} = \kappa \partial_{t} (\partial_{x}^{2} \theta) |_{j}^{n}.$$ 

(4.71)

Thus the second term on the right-hand side of (4.70) vanishes and and hence we get

$$\partial_{t} \theta |_{j}^{n} = \kappa \partial_{x}^{2} \theta |_{j}^{n} + O(\Delta t^2) + O(\Delta x^2).$$

(4.72)

Thus besides being unconditionally stable, the Crank-Nicholson scheme is of second order accuracy in time even though we employ a one-sided in time finite difference approximation. These two facts is why the Crank-Nicholson scheme is popular when solving true diffusive problems. However, as shown in the next section it has one disadvantage compared to the more standard schemes. It turns the otherwise parabolic equation into one which numerically look like an elliptic problem, and hence requires us to employ an elliptic solver for every time step.

### 4.8 A direct elliptic solver

If we inspect the Crank-Nicholson scheme (4.66) in more detail we find that including the implicit terms on the right-hand side of (4.66) turns the original parabolic equation into an apparent
elliptic equation. By adding the new terms the original local algorithm is turned into a non-local or global algorithm. The implication is that the solution $\theta_{n+1}^j$ at time level $n+1$ in addition to depend on the solution at the previous time level $n$ also depends on $\theta_{n+1}^{j-1}$ and $\theta_{n+1}^{j+1}$, that is, depends on the solution at the adjacent space points $\pm \Delta x$ away at the new time level $n+1$. We may illustrate this by rearranging the terms in (4.66) to obtain

$$-\frac{\kappa \Delta t}{2 \Delta x^2} \theta_{j-1}^{n+1} + \left(1 + 2 \frac{\kappa \Delta t}{2 \Delta x^2}\right) \theta_j^{n+1} - \frac{\kappa \Delta t}{2 \Delta x^2} \theta_{j+1}^{n+1} = \theta_j^n + \frac{\kappa \Delta t}{2 \Delta x^2} \left[\theta_{j-1}^n - 2 \theta_j^n + \theta_{j+1}^n\right].$$

(4.73)

Thus we cannot solve for $\theta_{n+1}^j$ without knowing $\theta_{n+1}^{j-1}$ and/or $\theta_{n+1}^{j+1}$. Let us consider that we solve (4.73) for increasing values of $j$. Then for any arbitrary $j$ we have already solved for $\theta_{n+1}^{j-1}$, and it is thus known. However, we have not yet solved for $j+1$, and thus $\theta_{n+1}^{j+1}$ is unknown.

Many of the model codes employed in numerical weather and numerical ocean weather prediction today employ semi-implicit methods. We are therefore in need of a method whereby such problem can be solved. Such methods are commonly referred to as elliptic solvers. Moreover, since we deal with time marching problems, we have to apply the elliptic solver for each time step. Thus we additionally require that the elliptic solver is efficient in the sense that it is fast on any computer. The most efficient elliptic solvers are those referred to as direct elliptic solvers. One such method is the so called Gauss elimination which we will use as an example. It 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 (4.73) into a more general form,

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

(4.74)

where $a_j$, $b_j$, and $c_j$ represents the coefficients in (4.73). The use of the subscript $j$ attached to these coefficients is to acknowledge that they in general are functions of space. 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 (4.74) within a finite domain. Thus $\theta_1^n$ and $\theta_J^n$ are determined by the boundary conditions. We therefore assume for simplicity that these are known whatever boundary condition is applied for all time levels $n$.

For convenience we will also drop the superscript $n$ and $n+1$. Thus we are required to solve,

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

(4.75)

under the conditions

$$\theta_1 = \hat{\theta}_0,$$

and

$$\theta_J = \hat{\theta}_L,$$

(4.76)

where $\hat{\theta}_0$ and $\hat{\theta}_L$ are known functions. We observe that (4.75) may be more compactly written as

$$A \cdot \theta = h,$$

(4.77)

In the infancy of numerical weather prediction (NWP) most elliptic solvers were the 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 shown here.
where the tensor $\mathbf{A}$ is the \textit{tridiagonal matrix}

$$
\mathbf{A} = \begin{bmatrix}
  b_2 & c_2 & 0 & \ldots & 0 & 0 & 0 \\
  a_3 & b_3 & c_3 & \ldots & 0 & 0 & 0 \\
  0 & a_4 & b_4 & \ldots & 0 & 0 & 0 \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
  0 & 0 & 0 & \ldots & a_{J-1} & b_{J-1} & c_{J-1} \\
  0 & 0 & 0 & \ldots & 0 & a_J & b_J \\
\end{bmatrix}
$$

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

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

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

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 conditions are as important in order to determine the solution as is the governing equations.

\textbf{Forward sweep}

We are now ready to perform the forward sweep. The idea is to replace all elements of the matrix $\mathbf{A}$ 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 (4.77) follows

$$
b_2 \theta_2 + c_2 \theta_3 = h'_2. \quad (4.81)
$$

We then normalize by dividing by $b_2$

$$
\theta_2 + d_2 \theta_3 = w_2, \quad (4.82)
$$

where

$$
d_2 = \frac{c_2}{b_2} \quad \text{and} \quad w_2 = \frac{h'_2}{b_2}. \quad (4.83)
$$

For $j = 3$ we get from (4.77)

$$
a_3 \theta_2 + b_3 \theta_3 + c_3 \theta_4 = h'_3. \quad (4.84)
$$
Substituting for $\theta_2$ from (4.82) and normalizing gives

$$\theta_3 + d_3 \theta_4 = w_3$$  \hspace{1cm} (4.85)

where

$$d_3 = \frac{c_3}{b_3 - d_2 a_3} \quad \text{and} \quad w_3 = \frac{h_3' - a_3 w_2}{b_3 - d_2 a_3}.$$  \hspace{1cm} (4.86)

Repeating this for $j = 4$ we get

$$\theta_4 + d_4 \theta_5 = w_4$$  \hspace{1cm} (4.87)

where

$$d_4 = \frac{c_4}{b_4 - d_3 a_4} \quad \text{and} \quad w_4 = \frac{h_4' - a_4 w_3}{b_4 - d_3 a_4}.$$  \hspace{1cm} (4.88)

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

$$\theta_j + d_j \theta_{j+1} = w_j, \quad j = 2(1)J - 1.$$  \hspace{1cm} (4.89)

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

$$d_j = \begin{cases} \frac{c_j}{b_j - d_{j-1} a_j} & ; \quad j = 2(1)J - 1 \\ 0 & ; \quad j = J \end{cases}, \quad w_j = \begin{cases} \frac{h_j'}{b_j} & ; \quad j = 2(1)J - 1 \\ \frac{h_{j-1} - a_j w_{j-1}}{b_j - d_{j-1} a_j} & ; \quad j = 3(1)J \end{cases},$$  \hspace{1cm} (4.90)

respectively. We observe that $d_J$ is set to zero. This is due to the fact that for $j = J$ (4.77) becomes

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

and hence by substituting for $\theta_{J-1}$ from (4.89) we simply get

$$\theta_J = w_J.$$  \hspace{1cm} (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 (4.78) now reads

$$\mathbf{A}' \cdot \theta = w,$$  \hspace{1cm} (4.93)

where the matrix $\mathbf{A}'$ is

$$\mathbf{A}' = \begin{bmatrix} 1 & d_2 & 0 & \ldots & 0 & 0 \\ 0 & 1 & d_3 & \ldots & 0 & 0 \\ 0 & 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 1 & d_{J-1} \\ 0 & 0 & 0 & \ldots & 0 & 1 \end{bmatrix}.$$  \hspace{1cm} (4.94)
and the vector \( w \) is

\[
\begin{bmatrix}
w_2 \\
w_3 \\
\vdots \\
w_{J-1} \\
w_J
\end{bmatrix}
\]

(4.95)

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 (4.90). Second we note from (4.92) that \( \theta_j \) is simply given by \( w_j \) and that the latter is known from (4.90). Thus we are in a position where \( \theta_j \) is known. Hence applying (4.89) for \( j = J - 1 \) and solving with respect to \( \theta_{J-1} \) we get

\[
\theta_{J-1} = w_{J-1} - d_{J-1} \theta_J.
\]

(4.96)

We may continue this and solve for \( \theta_{J-2}, \theta_{J-3}, \ldots \) and so on. Hence we can solve for all the remaining \( \theta_j \)'s by backward substitution into (4.89), that is,

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

(4.97)

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, and at least solve the resulting ODE by employing the Gauss elimination method.

**Exercises**

1. Show that the scheme (4.14) is unconditionally unstable. Hint: Show that \( |G| > 1 \) regardless of the choice made for \( \Delta t \) and \( \Delta x \).

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

3. Show that the growth factor associates with the scheme (4.43) is

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

(4.98)

and hence that the scheme is unconditionally stable. Also show that \( |G| < 1 \) for all wavelengths. Note that \( |G| \) decreases as \( \Delta t \) increases.
4. Show that the growth factor for the DuFort-Frankel scheme (4.53) is

\[ G_{1,2} = \frac{\chi \cos \alpha \Delta x \pm \sqrt{1 - \chi^2 \sin^2 \alpha \Delta x}}{1 + \chi} \]

where \( \chi = \frac{2\kappa \Delta t}{\Delta x^2} \). \hspace{1cm} (4.99)

5. Show that the expression (4.64) is indeed the expression for the growth factor of the scheme (4.63) when using von Neumann’s analysis method.
Chapter 5

THE ADVECTION PROBLEM

In this chapter we will investigate possible numerical methods whereby the advection equation (3.14) may be solved by numerical means. As we did for the diffusion problem we will reduce it to its simplest form\(^1\), and hence we consider a one-dimensional advection process. Thus we let \( F_A = u \theta i \), and hence the advection equation (3.14) reduces to
\[
\frac{\partial \theta}{\partial t} + \frac{\partial}{\partial x} (u \theta) = 0. \tag{5.1}
\]
Here \( u(x, t) \) is the advection speed along the \( x \)-axis. Note that in general \( u \) varies in time and space in which case the problem is non-linear. Following Einstein’s principle of making things as simple as possible, but no simpler, we will, however, in most instances below assume that \( u \) is uniform in time and space. We therefore let \( u = u_0 = \text{constant} \). Hence (5.1) becomes
\[
\frac{\partial \theta}{\partial t} + u_0 \frac{\partial}{\partial x} \theta = 0. \tag{5.2}
\]

The general true solution to (5.2) is\(^2\)
\[
\theta = \theta(x - u_0 t). \tag{5.3}
\]
To illustrate this let us assume that \( \theta \) is a good function (Chapter 2, Section 2.11 on page 29). Then we may represent it by a Fourier series, that is,
\[
\theta(x, t) = \sum_{n=0}^{\infty} \Theta_n(t) e^{i \alpha_n x}, \tag{5.4}
\]
where \( \alpha_n \) is the wavenumber of mode number \( n \), and \( \Theta_n \) 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
\[
\sum_{n=0}^{\infty} \left( \frac{d\Theta_n}{dt} + i \alpha_n u_0 \Theta \right) e^{i \alpha_n x} = 0. \tag{5.5}
\]

---
\(^1\)“Make things as simple as possible, but no simpler” Albert Einstein (1879-1955)
\(^2\)Exercise 1 on page 91 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 \( \alpha_n \).
This is true only as long as
\[ \frac{d\Theta_n}{dt} + i\alpha_n u_0 \Theta = 0. \] (5.6)

Hence we get
\[ \Theta_n = \Theta_{0n} e^{-i\alpha_n t}, \] (5.7)

where \( \Theta_{0n} \) is the initial value of the amplitude of wavenumber mode \( n \) at time \( t = 0 \). Substituting (5.7) into (5.4) we get
\[ \theta(x, t) = \sum_{n=0}^{\infty} \Theta_{0n} e^{i\alpha_n (x - u_0 t)}, \] (5.8)

which indeed shows that the solution is a function of \( x - u_0 t \) only.

Moreover, let us assume that the initial condition is specified by a single harmonic (monochromatic) wave of wavelength \( \lambda \) and amplitude \( \Theta_0 \) then initially
\[ \theta(x, 0) = \Theta_0 e^{i\frac{2\pi}{\lambda} x}. \] (5.9)

Then \( \alpha_n = 2\pi/\lambda \) and \( \Theta_{0n} = \Theta_0 \), and the solution (5.8) becomes
\[ \theta(x, t) = \Theta_0 e^{i\frac{2\pi}{\lambda} (x - u_0 t)}, \] (5.10)

that is, a monochromatic wave of the same wavelength. This particular solution is a wave propagating in the positive \( x \) direction with a phase speed given by \( u_0 \). As alluded to in Chapter 1, this solution is typical of hyperbolic systems. Indeed, the solutions (5.3), (5.8) and (5.10) to (5.2) are such that if we travel along with the advection speed we will experience no change in the property \( \theta \). If we, however, observe the wave from a fixed position in space, the property \( \theta \) will change in accord with (5.3) as the “wave” passes by. In fact this may be inferred from Section 3.3. In our case \( u = u_0 i \) and hence \( \nabla \cdot \mathbf{u} = 0 \). From (3.15) then follows that the variance of \( \theta \) is conserved implying that any initial distribution of the property \( \theta \) is conserved as time progresses.

The formal solution (5.8) underscores that the general true solution to the advection equation as given in (5.3) consists of waves of various wavelengths (wavenumbers) and amplitudes all of which propagates at the same speed \( u_0 \). Furthermore, (5.8) underscores that the wave that dominate the solution is the wave belonging to the mode that contains most energy, that is, the wave that initially has the largest amplitude \( \Theta_{0n} \).

## 5.1 Finite difference forms

Our concern is to develop an algorithm by which (5.2) can be solved by numerical means. To find a suitable finite difference equation to (5.2) we follow the procedure used to derive the finite difference equation for the diffusion equation. Thus we start by replacing the terms in the advection equation by finite difference approximations. However, our experience when performing this on the diffusion equation taught us that this might not be straightforward in that some numerical schemes must be discarded because they might be numerically unstable or inconsistent.
Our task is therefore to construct a scheme that is numerically stable and consistent. The latter is done by performing a stability analysis as well a consistency analysis. If the scheme turns out to be unstable or inconsistent then 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 4.3), while consistency is analyzed via use of Taylor series (cf. Section 2.6) as outlined in Section 4.6 on page 54.

Let us start by applying the successful forward in time and centered in space (FTCS) scheme that worked well for the diffusion equation. Thus we first construct a forward in time finite difference approximations for the time rate of change. To this end we use Taylor series expansions and get

\[ \frac{\partial t \theta^n_j}{\Delta t} = \theta^{n+1}_j - \theta^n_j. \] (5.11)

In a similar fashion, we next construct a centered in space finite difference approximation for the first order space differential in (5.2), that is,

\[ \frac{\partial x \theta^n_j}{2\Delta x} = \frac{\theta^{n+1}_j - \theta^n_{j-1}}{2\Delta x}. \] (5.12)

By replacing the two terms in (5.2) by the finite difference approximations (5.11) and (5.12) we get the finite difference equation

\[ \frac{\theta^{n+1}_j - \theta^n_j}{\Delta t} + u_0 \frac{\theta^{n+1}_j - \theta^n_{j-1}}{2\Delta x} = 0, \] (5.13)

which we solve with respect to \( \theta^{n+1}_j \) to get

\[ \theta^{n+1}_j = \theta^n_j - u_0 \frac{\Delta t}{2\Delta x} (\theta^{n+1}_{j+1} - \theta^n_{j-1}). \] (5.14)

Since the finite difference approximation (5.14) 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 useful to us. To this end we make use of von Neumann’s method, and start by substituting the discrete Fourier component (4.23) into (5.14). After some manipulations we get

\[ \Theta_{n+1} = \Theta_n - \frac{u_0 \Delta t}{2\Delta x} (e^{i\alpha \Delta x} - e^{-i\alpha \Delta x}) \Theta_n, \] (5.15)

where we have divided through with the common factor \( e^{i\alpha \Delta x} \). Recalling the definition of the growth factor (4.24), and noting that \( e^{i\alpha \Delta x} - e^{-i\alpha \Delta x} = 2i \sin \alpha \Delta x \), we get

\[ G = 1 - i\lambda, \] (5.16)

where

\[ \lambda = \frac{u_0 \Delta t}{\Delta x} \sin \alpha \Delta x. \] (5.17)
5.1 Finite difference form

We observe that the growth factor is a complex number with a real part given by 1 and an imaginary part given by $\lambda$. 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

$$|G| = \sqrt{1 + \lambda^2}.$$  \hspace{1cm} (5.18)

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

\begin{quote}
\textit{Never use a forward in time, centered in space scheme for the advection problem.}
\textit{It always leads to an unconditionally unstable scheme.}
\end{quote}

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 FD scheme that works well for the diffusion problem does not necessarily work well for the advection problem.

As listed in, e.g., \textit{O’Brien} (1986) (page 165 and onwards) there are many stable schemes that are suggested over the past to solve the advection equation. The reason is that advection is one of the most prominent atmospheric and oceanic processes. While many of the schemes are constructed to minimize unwanted properties of other schemes (e.g., numerical diffusion as detailed in Section 5.14 below, or numerical dispersion as detailed in Section 5.11 below), other schemes are constructed 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 schemes, say schemes of $O(\Delta t^4)$ and $O(\Delta x^4)$ or higher (cf. Section 10.1 on page 157). Among the former are so called flux corrective schemes (cf. Section 5.15 on page 88) and Lagrangian schemes (cf. Section 5.7 on page 73).

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 \textit{leapfrog scheme}, the \textit{upwind} or \textit{upstream} scheme, the \textit{diffusive scheme}, and the \textit{Lax-Wendroff scheme}.

\(^4\text{Let } A = a + ib \text{ be an imaginary number with real part } a \text{ and imaginary part } b. \text{ Then } |A| = \sqrt{AA^*} = \sqrt{a^2 + b^2} \text{ where } A^* = a - ib \text{ is the complex conjugate of } A.\)
5.2 The leapfrog scheme

One of the first schemes employed in atmospheric and oceanic problems is the centered in time and centered in space (CTCS) scheme, a scheme we emphasized was totally useless for the diffusion problem. To construct the scheme we use Taylor series expansions. A centered in time finite difference approximation for the time rate of change is

$$\left[\partial_t \theta\right]_j^n = \frac{\theta_j^{n+1} - \theta_j^{n-1}}{2\Delta t}, \tag{5.19}$$

while a centered in space finite difference approximation to the first order space derivative is

$$\left[\partial_x \theta\right]_j^n = \frac{\theta_j^{n+1} - \theta_j^{n-1}}{2\Delta x}. \tag{5.20}$$

By replacing the two terms in (5.2) by their finite difference approximations (5.19) and (5.20), and rearranging terms, we get

$$\theta_j^{n+1} = \theta_j^{n-1} - u_0 \frac{\Delta t}{\Delta x} \left( \theta_{j+1}^n - \theta_{j-1}^n \right). \tag{5.21}$$

Note that since we used centered approximations to derive the finite difference equation (5.21) the truncation error is of $O(\Delta x^2) + O(\Delta t^2)$. The scheme is therefore often referred to as a second order scheme. The scheme is also commonly referred to as the leapfrog scheme. The rationale is that we use information about the variable $\theta$ from all the points surrounding the $x_j, t^n$ point in $x, t$ space, that is, $\theta_{j+1}^n$ and $\theta_{j-1}^n$ to find $\theta_j^{n+1}$, but do not incorporate information of the variable from the point $x_j, t^n$ itself, that is, $\theta_j^n$. In a sense we leapfrog the point $x_j, t^n$.

The scheme is traditionally fairly popular for three main reasons. For one the scheme is, as shown in Section 5.3, neutrally stable. Hence there is no numerical or artificial damping or energy dissipation associated with the scheme when going from one time level to the next, a highly desirable property. Secondly the scheme is of second order accuracy. Finally, the scheme is easy to implement and works fast and efficiently on most computers. It has, however, some disadvantageous properties. First, as detailed in Section 5.11 (page 79), the scheme contains what is referred to as numerical dispersion, which sometimes leads to negative tracer concentrations. Secondly the scheme, as detailed in Section 5.12 below, also contains what is referred to as unphysical modes which has to be dealt with.

5.3 Stability of the leapfrog scheme: The CFL condition

We note that since the leapfrog scheme is derived exclusively using Taylor series, it is a consistent scheme. What about its stability? And if stable, under what conditions is it stable?

5In describing their model people often writes “..., while a second order scheme is employed for advective terms”.
5.3 The CFL condition

To analyze its stability we make use of von Neumann’s method as outlined in Section 4.3. Thus we first replace the dependent variable $\theta$ in (5.21) by its discrete Fourier component (4.23) to give

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

(5.22)

To find the growth factor $G$ we first make use of (4.24) and then multiply by the growth factor to obtain

$$G^2 + 2i\lambda G - 1 = 0,$$

(5.23)

where

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

(5.24)

The two solutions for the growth factor are

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

(5.25)

and hence they are complex functions. This was to be expected since the factor in front of the first order term in (5.23) is imaginary. We note that if the radical in (5.25) 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 (5.25) 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. 5.18 of Section 5.1)

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

(5.26)

Since by definition $\Theta_{n+1} = |G|\Theta_n$ it follows that there is no artificial or numerical damping or dissipation involved when advancing from one time level to the next. The scheme (5.21) is therefore neutrally stable implying that the energy associated with $\theta$ is conserved. We note that since we have assumed the velocity by which the property $\theta$ is advected to be constant (i.e., $u = u_0 i$) its gradient is zero. Hence it is highly desirable that the property of the scheme is in line with the property of advection processes as outlined in Section 3.3, namely that when the gradient of the advection velocity is zero then the variance should be conserved.

We recall that for (5.26) to be true we did require that the radical in (5.25) is positive. Hence

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

(5.27)

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

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

(5.28)

is true then (5.27) is satisfied. Thus (5.28) is a sufficient condition for stability for the leapfrog scheme (5.21). Moreover, since $|G_{1,2}| = 1$ the leapfrog scheme is a neutrally stable scheme. The condition (5.28) is referred to as the Courant-Friedrich-Levy condition or simply the CFL condition for stability. The ratio or non-dimensional number

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

(5.29)
is usually referred to as the Courant number and the CFL condition becomes simply \( C \leq 1 \). Since \( \Delta x \) 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 stringent upper bound on the time step \( \Delta t \), 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 \( \Delta t \) 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.

### 5.4 The upstream scheme

One of the unwanted properties of the leapfrog scheme is that it is inherently dispersive (cf. Section 5.11) causing it to generate negative concentrations, even though concentrations are positive definite quantities by definition. A scheme that does conserve 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 \( O(\Delta t) + O(\Delta x) \). Thus, again using Taylor series expansions, we get

\[
\theta_{j}^{n+1} = \theta_{j}^{n} - |u_0| \frac{\Delta t}{\Delta x} \left\{ \begin{array}{l}
\theta_{j}^{n} - \theta_{j-1}^{n} ; \quad u_0 \geq 0 \\
\theta_{j}^{n} - \theta_{j+1}^{n} ; \quad u_0 < 0
\end{array} \right.
\]  

(5.30)

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 91). We note for later convenience that (5.30) may be written as

\[
\theta_{j}^{n+1} = (1 - C)\theta_{j}^{n} + C \left\{ \begin{array}{l}
\theta_{j-1}^{n} ; \quad u_0 \geq 0 \\
\theta_{j+1}^{n} ; \quad u_0 < 0
\end{array} \right.
\]  

(5.31)

where \( C \) is the Courant number as defined in (5.29).

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
5.5 The diffusive scheme

As shown in Section 5.1 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 $\theta_j^n$ in (5.14) by $\frac{1}{2}(\theta_{j+1}^n + \theta_{j-1}^n)$. Thus we get

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

(5.32)

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 $\theta_j^n$ on both sides of (5.32) to get

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

(5.33)

Then using Taylor series we observe that the terms on the left-hand side becomes

$$\theta_j^{n+1} - \theta_j^n = \partial_t \theta_j^n \Delta t + \frac{1}{2} \partial_t^2 \theta_j^n \Delta t^2 + O(\Delta t^3)$$

(5.34)

while the first term on the right-hand side become

$$\frac{1}{2} \left( \theta_{j+1}^n - 2\theta_j^n + \theta_{j-1}^n \right) = \frac{1}{2} \partial_x^2 \theta_j^n \Delta x^2 + O(\Delta x^4).$$

(5.35)

The last term on the right-hand side of (5.33) becomes

$$u_0 \frac{\Delta t}{2\Delta x} \left( \theta_{j+1}^n - \theta_{j-1}^n \right) = u_0 \Delta t \left( \partial_x \theta_j^n + \frac{1}{6} \partial_x^3 \theta_j^n \Delta x^2 + O(\Delta x^4) \right).$$

(5.36)

Substituting (5.34) - (5.36) into (5.33) and rearranging terms we therefore get

$$\partial_t \theta_j^n + u_0 \partial_x \theta_j^n = \frac{1}{2} \frac{\Delta x^2}{\Delta t} (1 - C^2) \partial_x^2 \theta_j^n + O(\Delta x^2) + O(\Delta t).$$

(5.37)
where $C$ is the Courant number as defined in (5.29). Hence we find that the scheme is only conditionally consistent in that $\Delta x^2$ has to go to zero faster than $\Delta t$. We also observe that the source of the inconsistency is the first term on the right-hand side of (5.37). 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 (5.32), 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 (5.32) 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

$$G = \sqrt{1 - (1 - C^2) \sin^2 \alpha \Delta x}, \quad (5.38)$$

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.

### 5.6 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, Richtmeyer 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 5.1) 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 5.1).

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 $\theta_j^{n+\frac{1}{2}}$ using (5.32). Hence we get

$$\frac{\theta_j^{n+\frac{1}{2}}}{\Delta t} \left( \frac{1}{2} \left( \theta_{j+1}^n + \theta_j^n \right) \right) + u_0 \frac{\theta_{j+1}^n - \theta_j^n}{\Delta x} = 0, \quad (5.39)$$

or

$$\theta_j^{n+\frac{1}{2}} = \frac{1}{2} \left( \theta_{j+1}^n + \theta_j^n \right) - u_0 \frac{\Delta t}{2 \Delta x} \left( \theta_{j+1}^n - \theta_j^n \right). \quad (5.40)$$

We note that even though the forward in time, centered in space scheme is unstable the trick of replacing $\theta_j^n$ 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 5.5.
5.6 The Lax-Wendroff scheme

The second step is to employ the leapfrog scheme (5.21) to find \( \theta_{j}^{n+1} \), 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

\[
\theta_{j}^{n+1} = \theta_{j}^{n} - \frac{1}{2} \text{sgn}(u_0) C \left( \theta_{j+1}^{n+\frac{1}{2}} - \theta_{j-1}^{n+\frac{1}{2}} \right),
\]

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 (5.40) into (5.41). Thus we get

\[
\theta_{j}^{n+1} = \theta_{j}^{n} - \frac{1}{2} \text{sgn}(u_0) C \left( \theta_{j+1}^{n} - \theta_{j-1}^{n} \right) + \frac{1}{2} C^{2} \left( \theta_{j+1}^{n} - 2\theta_{j}^{n} + \theta_{j-1}^{n} \right),
\]

(5.42)

We observe that the last term on the right-hand side of (5.42) looks like an FDA of the second order derivative of \( \theta \) with respect to \( x \), that is, a diffusive term, and it may therefore appear that the Lax-Wendroff scheme has some inherent numerical diffusion. This is however not the case as shown in the next paragraph.

We observe that the Lax-Wendroff scheme is an explicit scheme, and hence if it is numerically stable it is most probably only conditionally stable. 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 first step using a diffusive scheme is indeed inconsistent. To analyze the consistency of the Lax-Wendroff scheme we first substitute the respective Taylor series into (5.42). The result is

\[
\partial_t \theta^n_j + u_0 \partial_x \theta^n_j = \frac{1}{2} \left( \partial_t^2 \theta^n_j - u_0 \partial_x^2 \theta^n_j \right) \Delta t + \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2).
\]

(5.43)

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 second term on the right-hand side, and hence we get

\[
\partial_t \theta^n_j + u_0 \partial_x \theta^n_j = \mathcal{O}(\Delta t^2) + \mathcal{O}(\Delta x^2).
\]

(5.44)

\(^\text{6}\)The function \( \text{sgn} \) returns the sign of its argument. Thus \( \text{sgn}(u_0) \) returns the sign of the velocity \( u_0 \), that is, \( u_0 = \text{sgn}(u_0)|u_0| \).
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 $O(\Delta x^2)$ and $O(\Delta t^2)$, 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.

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

$$G = (1 - C^2 + C^2 \cos \alpha \Delta x) - i \text{sgn}(u_0) C \sin \alpha \Delta x$$

which shows that the growth factor is a complex function. The magnitude of the growth factor is

$$|G| = \sqrt{(1 - C^2 + C^2 \cos \alpha \Delta x)^2 + C^2 \sin^2 \alpha \Delta x}$$

which after some manipulations may be written

$$|G| = \sqrt{1 - (1 - C^2)C^2(1 - \cos \alpha \Delta x)^2}.$$  \hfill (5.47)

We notice that the sign of the last term is determined by the factor $1 - C^2 = (1 - C)(1 + C)$, since the remaining factors are positive definite quantities. Thus as long as $1 - C$ is positive we observe from (5.47) that $|G| \leq 1$, and hence that the Lax-Wendroff scheme is conditionally stable under the condition that the Courant number is less than one. Other nice features about the scheme are the absence of a temporal unphysical mode present in the leapfrog scheme (cf. Section 5.12 on page 82) and that it avoids the diffusion characteristics of the upwind and diffusive schemes (cf. Section 5.14 on page 86). However, as shown in Figure 5.2, the scheme is numerical dispersive.

### 5.7 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 (Section ??) we show how the technique is applied to solve the shallow water equations. We also use this method below (Section 5.10) 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

$$\frac{D^*x}{dt} = u(x, t)$$  \hfill (5.48)
5.7 The semi-Lagrangian scheme

The advection problem

Figure 5.2: Comparison of the numerical solution to the advection equation (5.1) using the leapfrog, the upwind and the Lax-Wendroff schemes with a Courant number of one half ($C = 0.5$). The solution is shown after 10 cycles (periodic boundary condition). The true solution is a bell function as shown by the black solid curve. We note that both the leapfrog and the Lax-Wendroff schemes give rise to numerical dispersion, while the upwind scheme gives rise to numerical diffusion.

Define special curves in the $t, x$ space (cf. Fig. 5.3), and let us simultaneously define the special differential operator

$$\frac{D^*}{dt} \equiv \partial_t + \frac{D^*x}{dt} \partial_x.$$  

(5.49)

Then the advection equation (5.2) may be rewritten to yield

$$\frac{D^* \theta}{dt} = 0$$  

(5.50)

along the slopes

$$\frac{D^*x}{dt} = u_0.$$  

(5.51)

We commonly refer to the curves defined by (5.51) as the characteristics and (5.51) as the characteristic equation. Since the solutions to (5.2) are solutions to (5.50) as well, we often refer (5.50) as the compatibility equation in the sense that (5.2) and (5.50) are compatible equations. Thus either one can be used to arrive at the solution.

We observe that (5.50) tells us that $\theta$ is conserved along the characteristics (5.51). 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
THE ADVECTION PROBLEM

5.7 The semi-Lagrangian scheme

Figure 5.3: 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 right in $x,t$ space as given by (5.48). 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 5.3. 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 5.3 we may conclude that after a critical time $t = t_c = L/u_0$ all information about the initial distribution of $\theta$ 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 (5.50) are compatible, a solution to (5.50) is also automatically a solution to (5.2). We may therefore solve (5.50) 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 (5.50) using finite difference approximations. As is common
5.7 The semi-Lagrangian scheme

THE ADVECTION PROBLEM

The semi-Lagrangian scheme is used to solve the advection problem. The equation for the semi-Lagrangian scheme is given by:

\[ \frac{\partial u}{\partial x} = 0 \]

\[ u(x,t) = \frac{1}{1 - u} \]

where \( u \) is the velocity. The solution to this problem is a straight line through the grid points.

**Figure 5.4:** Sketch of the method of characteristics. The distance between the grid points are \( \Delta t \) in the vertical and \( \Delta x \) in the horizontal direction. The sloping solid line is the characteristic through the point \( j, n + 1 \). It is derived from (5.48) and the slope is given by \( \frac{1}{u} \) (\( u > 0 \)). 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 5.4. Recalling that in our simple case (5.50) tells us that \( \theta \) is conserved along the characteristics, a straightforward finite difference approximation to (5.50) is

\[ \theta_{j+1}^n = \theta_Q^n \]

where \( \theta_Q^n \) 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 5.4). We refer to this point as \( x_Q \), and we find its position along the \( x \)-axis by making a finite difference approximation of (5.51), that is,

\[ \frac{x_j - x_Q}{\Delta t} = u_0 \quad \text{or} \quad x_Q = x_j - u_0 \Delta t. \]

(5.53)

Since we know both \( u_j^n \) and \( x_j \), (5.53) 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 \( \theta_j^n \) is known for all grid points, we may interpolate linearly between the adjacent grid points to find \( \theta_Q^n \), or the value of \( \theta \) 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 \( x_{j-1} \leq x_Q \leq x_j \) we get

\[ \theta_Q^n = \theta_{j-1}^n + \frac{\theta_j^n - \theta_{j-1}^n}{\Delta x} (x_Q - x_{j-1}). \]

(5.54)
Substituting \( x_Q \) from (5.53) into (5.54) we get

\[
\theta^n_Q = (1 - C)\theta^n_j + C\theta^n_{j-1},
\]

(5.55)

where

\[
C = |u_0| \frac{\Delta t}{\Delta x}
\]

(5.56)
is the Courant number. Since \( \theta \) in accord with (5.50) is conserved along the characteristic, that is, (5.52) is satisfied, we finally get

\[
\theta^{n+1}_j = (1 - C)\theta^n_j + C\theta^n_{j-1}.
\]

(5.57)

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 \( \Delta t \) is such that \( \Delta x < u_0 \Delta t < 2 \Delta x \), then we simply approximate \( \theta^n_Q \) by linearly interpolation between \( \theta^n_{j-1} \) and \( \theta^n_{j-2} \), that is, we get

\[
\theta^{n+1}_j = \theta^n_j - 2 + \theta^n_{j-1} - 2 \frac{\Delta x}{x_Q - x_{j-2}} = (2 - C)\theta^n_{j-1} + (C - 1)\theta^n_{j-2}.
\]

(5.58)

Thus as long as we keep track of the position of \( x_Q \) there is no restriction on the time step \( \Delta t \) we may use, that is, the scheme is \textit{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,

\[
\frac{x_j - x_Q}{\Delta t} = \frac{1}{2} (u^n_j + u^{n+1}_j) \text{ or } x_Q = x_j - \frac{1}{2} (u^n_j + u^{n+1}_j) \Delta t.
\]

(5.59)

The position of \( x_Q \) thus found we find a \( \theta^{n+1}_j \) by performing a two point interpolation, that is, as long as \( x_{j-1} \leq x_Q \leq x_j \) we get

\[
\theta^{n+1}_j = (1 - C^n_j)\theta^n_j + C^n_j\theta^n_{j-1},
\]

(5.60)

where

\[
C^n_j = \frac{1}{2} \left( u^n_j + u^{n+1}_j \right) \frac{\Delta t}{\Delta x}.
\]

(5.61)

### 5.8 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

\[
\theta^{n+1}_j = \theta^n_j - u_0 \frac{\Delta t}{2\Delta x} \left( \theta^{n+1}_{j+1} - \theta^{n+1}_{j-1} \right), \quad \begin{cases} j = 2(1)J - 1 \\ n = 0(1) \ldots \end{cases}
\]

(5.62)
We observe that this scheme is $O(\Delta t)$ and $O(\Delta x^2)$, 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 4.8 is a good choice.

To investigate the stability of (5.62) we make use of von Neumann’s method. Thus we get

$$\Theta_{n+1} = \Theta_n - iC \sin \alpha \Delta x \Theta_{n+1},$$

which results in a complex growth factor given by

$$G = \frac{1}{1 + iC \sin \alpha \Delta x}.$$  \hspace{1cm} (5.63)

Thus the magnitude of the growth function is

$$|G| = \frac{1}{\sqrt{1 + C^2 \sin^2 \alpha \Delta x}}$$ \hspace{1cm} (5.64)

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 $\Delta t$ 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 $\Delta t$ becomes too large the growth factor will be small implying that the scheme will contain a large numerical dissipation.

5.9 The initial problem in CTCS schemes

Although the leapfrog scheme is conditionally and neutrally stable it is not without disadvantageous properties. One of the problems with the scheme is associated with the number of initial conditions required, a problem already touched upon in Section 4.1 regarding the diffusion problem. Thus the question is how to start the time marching procedure. The forward in time centered in space scheme is one possibility, but as shown in Section 5.1 this is unconditionally unstable. However, we may nevertheless make use of this scheme when applied to a single time step.

Thus we start by using the scheme

$$\theta^1_j = \theta^0_j - u_0 \frac{\Delta t}{2\Delta x} (\theta^0_{j+1} - \theta^0_{j-1}).$$  \hspace{1cm} (5.66)

For the time level 2 and onwards we then use the leapfrog scheme (5.21). The step (5.66) 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 5.12).
5.10 Physical interpretation of the CFL condition

Figure 5.4 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 $u_0 > 0$ (5.53) says that $x_Q < x_j$. Moreover (5.53) 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 (5.30) we observe that for $u_0 > 0$ the information used to compute $\theta_{j}^{n+1}$ does originate by weighting the two points $\theta_{j}^{n}$ and $\theta_{j-1}^{n}$. In fact we may rewrite (5.30) for $u_0 > 0$ to give

$$\theta_{j}^{n+1} = (1 - C)\theta_{j}^{n} + C\theta_{j-1}^{n}, \quad C = |u_0| \frac{\Delta t}{\Delta x}$$

(5.67)

which matches (5.57) exactly. Thus from (5.67) follows that the upwind scheme may be interpreted as the value of $\theta$ at the time level $n + 1$, that is, $\theta_{j}^{n+1}$, is found by a simple weighting of the values $\theta_{j}^{n}$ and $\theta_{j-1}^{n}$ using the Courant number as weight. What the method of characteristics (5.57) 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 (5.28) must be satisfied.

Moreover, if $|u_0| \Delta t > \Delta x$ then the characteristic through $x_j$ at time level $n + 1$ (cf. Figure 5.4) 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 (5.67) to calculate $\theta$ 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 (5.67) 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 $\Delta x$ and $\Delta t$, 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 (5.28) 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 $n$.

5.11 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 \( \omega \) is the frequency and \( \alpha \) is the wavenumber. Recall that the phase speed is

\[
c = \frac{\omega}{\alpha}
\]  

(5.68)

If the frequency is a linear function of \( \alpha \), 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, \( \omega \) is a non-linear function of the wavenumber \( \alpha \). 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

\[
c_g = \alpha \partial_{\alpha}c + c.
\]  

(5.69)

Hence if the wave is non-dispersive \( c_g = c \), that is, 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 depends 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.

If we apply a wave solution to the advection equation (5.1), that is, \( \theta = \Theta_0 e^{i\alpha(x-ct)} \), we find that the phase speed is \( c = u \). Thus if \( u = u_0 \), that is, is uniform in time and space the phase speed becomes constant. Under these circumstances all the waves propagate with the same velocity, namely \( u_0 \), and hence the true solution to the advection equation is non-dispersive. The question then arises. Is this true for the numerical solution?

To investigate this we apply a similar analysis based on the finite difference approximation to the advection equation (5.1). Let us consider the leapfrog scheme (5.21). 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 or diffusion. But what about the dispersiveness? To get started we first substitute the discrete Fourier component

\[
\theta_j^n = \Theta_0 e^{i\alpha(j\Delta x-cn\Delta t)}
\]  

(5.70)

into (5.21). This gives

\[
-2i\sin(\alpha c\Delta t) = -2iu_0 \frac{\Delta t}{\Delta x} \sin(\alpha \Delta x)
\]  

(5.71)

or

\[
c = \frac{1}{\alpha \Delta t} \arcsin \left( u_0 \frac{\Delta t}{\Delta x} \sin(\alpha \Delta x) \right)
\]  

(5.72)

Obviously \( \partial_{\alpha}c \neq 0 \), and hence the leapfrog scheme is dispersive. Since this dispersiveness is due to the finite difference approximation, that is, associated with our numerical solver, we commonly refer to this as numerical dispersion. We note that as \( \Delta x \) and \( \Delta t \) goes to zero, but such that the stability condition is satisfied, that is,

\[
C \leq 1 \quad \text{or} \quad |u_0|\Delta t \leq \Delta x
\]  

(5.73)
5.11 Numerical dispersion

Figure 5.5: Numerical dispersion for the leapfrog scheme. The curves depicts the numerical phase speed as a function of the wavenumber based on (5.72) for various values of the Courant number \( C = \frac{u\Delta t}{\Delta x} \). The vertical axis indicates the phase speed \( c \) normalized by the advection speed \( u \). The horizontal axis indicates the wavenumber normalized by \( \frac{\pi}{\Delta x} \) where \( \Delta x \) is the space increment or the grid size. The analytic dispersion curve is just a straight line corresponding to the phase speed \( c = u \), that is \( \frac{c}{u} = 1 \). Note that 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 wave numbers 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 with wavelengths shorter than \( 4\Delta x \) the energy is propagating in the opposite direction of the waves.

then \( c \to u_0 \) and the wave becomes non-dispersive.

This is visualized in Figure 5.5 showing the normalized 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. We also observe from Figure 5.5, as was first noted by Grotjhan and O’Brien (1976), that the dispersiveness gets worse the less the Courant number. In fact from (5.69) follows that \( c_g \) becomes zero when

\[
\alpha \partial_\alpha c = -c. \tag{5.74}
\]

By use of (5.72) it follows that this is true for wavenumbers satisfying \( \cos(\alpha \Delta x) = 0 \), that is, for wavenumbers

\[
\alpha_m = \frac{1}{2}(2m - 1)\pi, \quad m = 1, 2, \ldots \tag{5.75}
\]

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 5.5 this corresponds to the normalized wavenumber \( \alpha' = 0.5 \).

\textsuperscript{7}For capillary waves \( \partial_\alpha c > 0 \) and hence a capillary wave travels at a speed slower than its energy.
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 $\alpha$. Then by looking at Figure 5.5 $\Delta x$ 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.

### 5.12 Unphysical solutions and numerical modes

In Section 5.14 and Section 5.11 we showed that two of the most popular numerical schemes for solving the advection equation, namely the upwind scheme and the leapfrog scheme, harbored some unwanted properties. While the upwind or upstream scheme was shown to contain numerical diffusion, the leapfrog scheme turned out to contain numerical dispersion. Although these properties, as their name suggests, have a physical interpretation they are nonetheless results of the employed scheme and hence unphysical or a numerical artifact. In Section 5.15 we showed how the numerical diffusion to some extent could be avoided by use of flux correction. Below (Section 10.5 on page 169) 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 an additional unphysical property, namely false numerical modes. Let us start by assuming that the initial condition is as given in (5.9), that is, a harmonic wave with wave number $\alpha$ and amplitude $\theta_0$. The true solution is then given by (5.10), that is, the solution is a single monochromatic wave with wavenumber $\alpha$ propagating with the phase speed $c = u_0$ in the positive $x$ direction.

To reveal that the leapfrog scheme actually contains two modes we solve the leapfrog scheme analytically. We start by assuming that the initial condition is a monochromatic wave with wavenumber $\alpha$ as given by (5.9). Recalling from Section 4.2 (page 45) that any good function can be written in terms of an infinite sum of waves, we note that if we can 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. Furthermore we recall from the stability analysis of Section 4.3 on page 46 using induction that the solution may be written

$$\theta_j^n = G^n \Theta_0 e^{i\alpha j \Delta x}. \quad (5.76)$$

where $G$ is the growth factor and $\Theta_0$ is the initial amplitude. We underscore that $G$ in general is a complex number. Recall that the growth factor for the leapfrog scheme actually has two solutions for the growth factor as given by (5.25) on page 68. For our purpose we rewrite these two solutions as

$$G_1 = g_r - ig_i, \quad \text{and} \quad G_2 = -(g_r + ig_i), \quad (5.77)$$

where the real and imaginary parts are given by

$$g_r = \sqrt{1 - \lambda^2} \quad \text{and} \quad g_i = \lambda. \quad (5.78)$$
respectively, and where \( \lambda \) is given by (5.24). Note that we have assumed that the CFL condition for stability is satisfied so that \( g_r \) is a real number. Recalling that any complex number \( P = a + ib \) may be written

\[
P = |P|e^{i\phi}
\]

where

\[
|P| = \sqrt{PP^*} = \sqrt{a^2 + b^2} \quad \text{and} \quad \phi = \arcsin\left(\frac{b}{|P|}\right),
\]

and noting that \( |G_{1,2}| = 1 \) we get

\[
G_1 = e^{-i\alpha c \Delta t} \quad \text{and} \quad G_2 = (-1)e^{i\alpha c \Delta t}
\]

where \( c = c(\alpha) \) is the dispersive phase speed defined in (5.72). Substituting these solutions into (5.76) we get

\[
\theta^n_{j} = \Theta_1 e^{i\alpha(j\Delta x - cn\Delta t)} + (-1)^n \Theta_2 e^{i\alpha(j\Delta x + cn\Delta t)},
\]

where \( \Theta_1 \) and \( \Theta_2 \) are as yet two unknown constants. Since we have only one initial condition at our disposal, we can only obtain a relationship between them. Utilizing the initial condition (5.9) we find

\[
\theta^n_{j} = (\Theta_2 - \Theta_0)e^{i\alpha(j\Delta x - cn\Delta t)} + (-1)^n \Theta_2 e^{i\alpha(j\Delta x + cn\Delta t)}.\]

We note that the first term on the right-hand side of (5.83) is a wave propagating in the positive \( x \) direction with phase speed \( c \) and amplitude \( \Theta_2 - \Theta_0 \). 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 (5.72). 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.

The latter problem is associated with the initial boundary problem discussed in Section 5.9 above, but manifests itself in the unknown constant \( \Theta_2 \) in (5.83). One remedy suggested in Section 5.9 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 \( \theta \) at the time step prior to the initial time. We recall that the Euler step (5.66) is

\[
\theta^1_j = \theta^0_j - u_0 \frac{\Delta t}{2\Delta x} \left( \theta^0_{j+1} - \theta^0_{j-1} \right).\]

Substituting the initial condition (5.9) into (5.84) and (5.83) (letting \( n = 1 \)) the additional condition becomes\(^8\)

\[
\Theta_0 [1 - i \sin(\alpha c \Delta t)] e^{i\alpha j \Delta x} = (\Theta_2 - \Theta_0) e^{-i\alpha c \Delta t} - \Theta_2 e^{i\alpha c \Delta t}
\]

\(^8\)Note that from (5.71) follows \( \lambda = u_0 \frac{\Delta t}{\Delta x} \sin(\alpha \Delta x) = \sin(\alpha c \Delta t) \)
and hence that
\[ \Theta_2 = \Theta_0 \frac{1 - \cos(\alpha c \Delta t)}{2 \cos(\alpha c \Delta t)}. \] (5.86)

The complete finite difference solution is then
\[ \theta^n_j = \Theta_0 \left\{ \frac{1 + \cos(\alpha c \Delta t)}{2 \cos(\alpha c \Delta t)} e^{i\alpha(j \Delta x - cn \Delta t)} + (-1)^{n+1} \frac{1 - \cos(\alpha c \Delta t)}{2 \cos(\alpha c \Delta t)} e^{i\alpha(j \Delta x + cn \Delta t)} \right\}. \] (5.87)

We observe, as mentioned in the previous Section 5.11, that as long as the stability criteria (5.28) is satisfied then \( c \to u_0 \) when \( \Delta t \) and \( \Delta x \) goes to zero independently. Under these circumstances we take note that \( \frac{1 + \cos(\alpha c \Delta t)}{2 \cos(\alpha c \Delta t)} \to 1 \) and \( \frac{1 - \cos(\alpha c \Delta t)}{2 \cos(\alpha c \Delta t)} \to 0 \). Thus as we allow \( \Delta t \to 0 \) and \( \Delta x \to 0 \) the first term on the right-hand side of (5.87) approaches the true solution while the second term vanishes. We conclude therefore that the second term is an unphysical 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 unphysical mode. Unless we do it may create a lot of noise in our calculations, or in extreme cases even numerical instabilities. This is particularly true since it alternates between attaining positive and negative values.

### 5.13 How to get rid of the unphysical 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 make use of a one-sided, forward in time, centered in space step from time to time as outlined in (5.14), that is, perform an Euler step. Although, as shown in Section 5.1, the Euler scheme is numerically unstable, we may still apply it for single time steps without destroying the stability of for instance the leapfrog scheme.

Another method, originally suggested by Robert (1966) and further developed by Asselin (1972), is to apply a time filtering technique. That a time filter will work the trick is obvious since the unphysical mode alternates between positive and negative values from time step to time step. We start by remarking that in general a time filter invoking the neighboring time levels only may be written

\[ \bar{\theta}(x, t) = \gamma \theta(x, t + \Delta t) + (1 - 2\gamma) \theta(x, t) + \gamma \theta(x, t - \Delta t), \] (5.88)

where \( \bar{\theta}(x, t) \) is the filtered function and \( \gamma \) is a weighting parameter. By use of the notation introduced in Section 2.9 we may rewrite (5.88) to yield

\[ \bar{\theta}^n_j = \theta^n_j + \gamma[\theta^{n+1}_j - 2\theta^n_j + \theta^{n-1}_j]. \] (5.89)

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,

\[ \bar{\theta}^n_j = \frac{1}{4} \left[ \theta^{n+1}_j + 2\theta^n_j + \theta^{n-1}_j \right], \] (5.90)
which gives twice the weight to the mid time level \( n \). To investigate the properties of the filter let us study one period or frequency \( \omega \) only. We start by representing the function \( \theta \) by it Fourier component

\[
\theta_j^n = \hat{\theta}_j e^{i\omega \Delta t}.
\]

Substituting this into (5.89) gives

\[
\overline{\theta}_j^n = R \theta_j^n,
\]

where the ratio

\[
R(\gamma) = \frac{\overline{\theta}_j^n}{\theta_j^n} = 1 - 2\gamma + 2\gamma \cos \omega \Delta t
\]

is the response function of the filter. For the 1-2-1 filter, that is, for \( \gamma = \frac{1}{4} \), the response function becomes

\[
R(\frac{1}{4}) = \frac{1}{2}(1 + \cos \omega \Delta t)
\]

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 \( T = 4\Delta t \). Thus waves or noise on the Nüquist frequency, that is, waves of periods \( 2\Delta t \), cannot exist and hence vanish. Since the unphysical mode inherent in the leapfrog scheme alternates between attaining negative and positive values from time step to time step, its dominate waveperiod is exactly \( 2\Delta t \). Hence the 1-2-1 filter is perfect in this sense. Noise of scales close to slightly longer periods, say \( 4\Delta t \), is damped to half of their original energy. 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 unphysical 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 \( \overline{\theta}_{j-1}^{n-1} \) is known and has been stored. Let us furthermore assume that the unfiltered value at time level \( n \), that is, \( \theta_j^n \), has been stored as well. We then first apply the leapfrog scheme to compute the function \( \theta \) at the new time level \( n + 1 \), that is, \( \theta_{j+1}^{n+1} \), using the unfiltered values of \( \theta \) at time level \( n \) and the filtered values of \( \theta \) at time level \( n - 1 \). For instance, if we consider the simple advection equation (5.1) and make use of the leapfrog scheme (5.21) to advance to the next time level we obtain

\[
\theta_{j+1}^{n+1} = \overline{\theta}_j^{n-1} - u \frac{\Delta t}{\Delta x} (\theta_{j+1}^n - \theta_{j-1}^n).
\]

Note that we in this first step have used the filtered value at time level \( n - 1 \) to compute \( \theta_{j+1}^{n+1} \).

The next step is to compute the filtered values at time level \( n \) using the filter (5.89), that is,

\[
\overline{\theta}_j^n = \theta_j^n + \gamma \left[ \theta_{j+1}^{n+1} - 2\theta_j^n + \overline{\theta}_{j-1}^{n-1} \right].
\]

In the third and final step we replace the filtered values at time level \( n - 1 \) by the new filtered values at time level \( n \). We can safely do this because after we have computed the new filtered values at time level \( n \) the filtered values at time level \( n - 1 \) are obsolete. We may now proceed to the next time level to compute \( \theta_{j+2}^{n+2} \), and so on.
5.14 Numerical diffusion

We take note, however, that the Asselin filter affects the numerical stability and that it produces numerical diffusion. It is possible to show that while the numerical diffusion increases with increasing values of the weighing parameter $\gamma$ 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 $\Delta t$. This fact entails that although we wish to employ the simple 1-2-1 filter since it exactly kills the unphysical 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 care and not necessarily for every time step.

5.14 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 5.5, this is true when the resolution is poor, that is, in areas where $\Delta x$ 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 5.12 the leapfrog scheme contains an unwanted unphysical mode.

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
**THE ADVECTION PROBLEM**  

5.14 Numerical diffusion

*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 (5.33) 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, does conserve 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 (5.30). We first rewrite it in terms of an advective flux defined by

\[
F_j^n = \frac{1}{2} \left[ (u_0 + |u_0|) \theta_j^n + (u_0 - |u_0|) \theta_{j+1}^n \right] \frac{\Delta t}{\Delta x}. \tag{5.97}
\]

We note that since \( u_0 = \text{sgn}(u_0) \cdot |u_0| \) the last term on the right-hand side of (5.97) 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 (5.30) is written

\[
\theta_j^{n+1} = \theta_j^n - (F_j^n - F_{j-1}^n), \tag{5.98}
\]

which is valid regardless of the sign of \( u_0 \). If we substitute each of the terms in (5.98) by its associated Taylor series expansion, that is,

\[
\begin{align*}
\theta_j^{n+1} &= \theta_j^n + \partial_t \theta_j^n \Delta t + \frac{1}{2} \partial_x^2 \theta_j^n \Delta t^2 + O(\Delta t^3), \\
\theta_{j+1}^n &= \theta_j^n + \partial_x \theta_j^n \Delta x + \frac{1}{2} \partial_x^2 \theta_j^n \Delta x^2 + O(\Delta x^3),
\end{align*}
\]

we get

\[
\partial_t \theta_j^n + \frac{1}{2} \partial_x^2 \theta_j^n \Delta t + O(\Delta t^2) = -u_0 \partial_x \theta_j^n + \frac{1}{2} |u_0| \partial_x^2 \theta_j^n \Delta x + O(\Delta x^2). \tag{5.100}
\]

We note that by differentiating (5.2) we get \( \partial_t \theta_j^n = u_0^2 \partial_x^2 \theta_j^n \), and hence by rearranging terms that

\[
\partial_t \theta_j^n + u_0 \partial_x \theta_j^n = \frac{1}{2} |u_0| (\Delta x - |u_0| \Delta t) \partial_x^2 \theta_j^n + O(\Delta x^2) + O(\Delta t^2). \tag{5.101}
\]

Defining

\[
\kappa^* = \frac{1}{2} (1 - C) |u_0| \Delta x, \tag{5.102}
\]

where \( C \) is the Courant number as defined in (5.28), (5.101) may be written

\[
\partial_t \theta_j^n = -u \partial_x \theta_j^n + \kappa^* \partial_x^2 \theta_j^n + O(\Delta x^2) + O(\Delta t^2). \tag{5.103}
\]
Thus to second order in time and space we solve the equation
\[ \partial_t \theta + u \partial_x \theta = \kappa^* \partial_x^2 \theta. \] (5.104)

We recognize (5.104) as an advection-diffusion equation (Chapter 3) with a diffusion coefficient \( \kappa^* \) given by (5.102). The terms of order \( O(\Delta x) \) and \( O(\Delta t) \) 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 a numerical diffusion inherent in the scheme. The strength of the numerical diffusion is determined by the diffusion coefficient defined in (5.102). 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 (5.28) for stability, and is associated with a near neutrally stable scheme. The diffusion term also goes to zero when \( \Delta x \) and \( \Delta t \) goes to zero, hence showing that the upstream scheme is consistent.

5.15 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 86, 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\(^9\), 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 (5.1) may be written
\[ \partial_t \theta + \partial_x F_A = 0, \] (5.105)
where \( F_A = u\theta \). As shown in Section 5.14 solving (5.105) using the upstream scheme results in a solution that to second order in time and space solves (5.104), that is, an advection diffusion equation. Thus, rather than solving (5.105) we, to second order, appear to solve
\[ \partial_t \theta + \partial_x(F_A + F_D^*) = 0, \] (5.106)
where \( F_D^* = -\kappa^* \partial_x \theta \) is a diffusive flux with a diffusion coefficient \( \kappa^* = \frac{1}{2}(\Delta x - u_0 \Delta t) u_0 \) (cf. eq. 5.102). 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
\[ \partial_t \theta + \partial_x(F_A + F_D^*) = 0 \] (5.107)

\(^9\)MPDATA is an abbreviation of “Multiple Positive Definite Advection-Transport Algorithm”
rather than (5.105). 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
\[
\theta^* = \frac{\kappa^* \partial_x \theta}{\theta}.
\tag{5.108}
\]

We note that according to (5.108) \( u^* = 0 \) 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 \( \partial_x \theta \). Hence its magnitude is proportional \( \partial_x \theta \) and 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 86. 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. 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 \( \theta^* \) based on the true advection equation (5.105) using a low order advection algorithm, that is, an algorithm with a truncation error of \( O(\Delta t) \) and \( O(\Delta x) \). Using for instance the upstream scheme for this purpose we get
\[
\theta^*_j = \theta^n_j - (F_A^n_j - F_A^{n-1}_j) \Delta t \Delta x.
\tag{5.109}
\]

where
\[
F_A^n_j = \frac{1}{2} \left[ (u^n_j + |u^n_j|) \theta^n_j + (u^n_j - |u^n_j|) \theta^{n+1}_j \right] \Delta t \Delta x.
\tag{5.110}
\]

We note that this step retains all the advantageous properties of the upwind scheme. We know, however, that the predictor solution \( \theta^*_j \), 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 \( \kappa^* \) is given (5.102). This causes the prediction \( \theta^*_j \) 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 (5.107) without the original advection term, that is,
\[
\partial_t \theta + \partial_x F_A^* = 0.
\tag{5.111}
\]
5.15 Flux correction

The advection problem

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. The green dashed lines are after 400 time steps while the blue, dash-dot lines are after 800 time steps. Cyclic boundary conditions are employed. (cf. Computer problem No. 6 in the Computer Problem notes).

Also for this corrector step we apply the low order upwind scheme. Hence the corrected solution $\theta_{j}^{n+1}$ reads

$$\theta_{j}^{n+1} = \theta_{j}^{*} - (F_{A}^{*}|_{j} - F_{A}^{*}|_{j-1})$$

(5.112)

where

$$F_{A}^{*}|_{j} = \frac{1}{2} \left[(u_{j}^{*} + |u_{j}^{*}|)\theta_{j}^{*} + (u_{j}^{*} - |u_{j}^{*}|)\theta_{j+1}^{*}\right] \frac{\Delta t}{\Delta x}.$$  

(5.113)

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 $\partial_{x}\theta_{j}^{*}$. From (5.108) we then get

$$u_{j}^{*} = \frac{1}{4} (1 - C_{j}^{n}) |u_{j}^{*}| \left(\frac{\theta_{j+1}^{*} - \theta_{j-1}^{*}}{\theta_{j}^{*} + \epsilon}\right),$$

(5.114)

where $C_{j}^{n} = |u_{j}^{*}| \Delta t / \Delta x$ is the local Courant number. Note that we have added, as suggested by Smolarkiewicz (1983), a small number $\epsilon$ in the denominator to ensure that $u_{j}^{*}$ goes to zero when $\theta_{j}^{*}$ is zero at the same time. If we make use of (5.114) to compute the anti-diffusive velocity the gradients are re-steepened. Moreover, it does not affect 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 $O(\Delta x^2)$ and $O(\Delta t^2)$. 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 (5.114) by a scaling factor larger than one, a method already suggested by Smolarkiewicz (1983). Thus we redefine the anti-diffusive velocity to read

$$u_j^* = \frac{1}{4} S_c (1 - C_j^n) |u_j^n| \left( \frac{\theta_j^{n+1} - \theta_j^n}{\theta_j^n + \epsilon} \right),$$

where $S_c \geq 1$ is the scaling factor. As an example Figure 5.7 shows the solution to (5.105) 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 (5.28).
   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 (5.82) is a solution to (5.21). Moreover, show that (5.87) follows from (5.82) when the initial distribution is given by (5.84), and where (5.14) is made use of to find $\theta_j^{-1}$.
   Hint: Show first that $G_{1,2}$ from (5.25) may be written

$$G_1 = e^{-i\chi}, G_2 = e^{i(\chi + \pi)}$$

where $\chi$ is given by (5.83).
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 4 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. 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 11 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 9. The equations themselves, as presented in (1.23) - (1.24) or their depth integrated versions (1.30) and (1.31) on page 10, 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 horizontal velocity $u$ and the

---

1Gaspard-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.
height 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\(^3\)\(N\) moving vertical layers 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 \(\rho_0\) they read,

\[
\begin{align*}
\partial_t \mathbf{u} + \mathbf{u} \cdot \nabla_H \mathbf{u} + \mathbf{f} \times \mathbf{u} &= -\nabla_H \phi + \frac{\tau_s - \tau_b}{\rho_0 h} + \frac{X}{h}, \\
\partial_t \phi + \mathbf{u} \cdot \nabla_H \phi &= -\phi \nabla_H \cdot \mathbf{u},
\end{align*}
\]

(6.1)

(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 \(\tau_s\) 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 \(\tau_b\) 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, \(X\) 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

\(^2\)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 141 where we treat various vertical coordinate systems.

\(^4\)This was first introduced by Jule Charney in his fundamental paper published in 1955 (Charney, 1955)
THE SHALLOW WATER PROBLEM

dynamical processes active in the atmosphere and ocean. Equally important, they allow us to 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, \( \phi, u, \) 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 3.4. Under these circumstances the set (6.1) and (6.2) becomes

\[
\partial_t u + u \cdot \nabla_H u + f k \times u = -\nabla_H \phi, \tag{6.3}
\]

\[
\partial_t \phi + u \cdot \nabla_H \phi = -\phi \nabla_H \cdot u. \tag{6.4}
\]

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 \( \phi. \) 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 \( u, v, \phi. \) 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,

$$ u = \bar{u} + u' \quad \text{and} \quad \phi = \bar{\phi} + \phi' \quad (6.5) $$

where the perturbed velocity (or basic state) is $\bar{u} = \bar{u}i + \bar{v}j$ 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

$$\begin{align*}
\partial_t \bar{u} + \bar{u} \cdot \nabla H \bar{u} + f k \times \bar{u} &= -\nabla_H \bar{\phi}, \\
\partial_t \bar{\phi} + \bar{u} \cdot \nabla H \bar{\phi} + \bar{\phi} \nabla H \cdot \bar{u} &= 0
\end{align*} \quad (6.6)$$

Substituting (6.5) into (6.3) - (6.4) invoking (6.6) and (6.7) we get

$$\begin{align*}
\partial_t u' + \bar{u} \cdot \nabla H u' + u' \cdot (\nabla_H \bar{u} + \nabla_H u') + f k \times u' &= -\nabla_H \phi', \\
\partial_t \phi' + \bar{u} \cdot \nabla H \phi' + u' \cdot (\nabla_H \bar{\phi} + \nabla_H \phi') + \bar{\phi} \nabla H \cdot u' &= -\phi' (\nabla_H \bar{u} + \nabla_H \cdot u') \quad (6.9)
\end{align*}$$

Note that the basic state variables such as $\bar{u}$ and $\bar{\phi}$ are functions of the independent variables $x$, $y$ and $t$, but are slowly varying in both space and time. Thus

$$\begin{align*}
|\partial_t \bar{u}| &\ll |\partial_t u'|, \quad |\nabla_H \bar{u}| \ll |\nabla_H u'|, \quad |\nabla_H \cdot \bar{u}| \ll |\nabla_H \cdot u'|, \\
|\partial_t \bar{\phi}| &\ll |\partial_t \phi'|, \quad |\nabla_H \bar{\phi}| \ll |\nabla_H \phi'|.
\end{align*} \quad (6.10)$$

Thus (6.8) and (6.9) are further reduced to

$$\begin{align*}
\partial_t u' + \bar{u} \cdot \nabla H u' + u' \cdot \nabla H u' + f k \times u' &= -\nabla_H \phi', \\
\partial_t \phi' + \bar{u} \cdot \nabla H \phi' + u' \cdot \nabla H \phi' + \bar{\phi} \nabla H \cdot u' &= -\phi' \nabla_H \cdot u'.
\end{align*} \quad (6.11)$$

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,

$$\frac{|u'^2|}{\bar{u}^2} \ll \frac{|u'|}{\bar{u}}, \quad \frac{|\phi'^2|}{\bar{\phi}^2} \ll \frac{|\phi'|}{\bar{\phi}}. \quad (6.13)$$
Thus (6.11) and (6.12) become
\[
\begin{align*}
\partial_t u + \bar{u} \cdot \nabla_H u + f k \times u &= -\nabla_H \phi, \\
\partial_t \phi + \bar{u} \cdot \nabla_H \phi + \bar{\phi} \nabla_H \cdot u &= 0.
\end{align*}
\] (6.14) (6.15)
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, \( \partial_t \bar{u} + \bar{u} \cdot \nabla_H \bar{u} \), is small compared to the Coriolis acceleration (small Rossby number, cf. Section 1.6 on page 11), it follows that the basic state must be in geostrophic balance, that is,
\[
\bar{u} = \frac{1}{f} k \times \nabla_H \bar{\phi}. 
\] (6.16)
Thus if the basic state is one at rest (\( \bar{u} = 0 \)) 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,
\[
\begin{align*}
\partial_t u + f k \times u &= -\nabla_H \phi, \\
\partial_t \phi + c_0^2 \nabla_H \cdot u &= 0,
\end{align*}
\] (6.17) (6.18)
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
\[
\begin{align*}
\partial_t u + \bar{u} \partial_x u - f v + \partial_x \phi &= 0, \\
\partial_t v + \bar{u} \partial_x v + f u &= 0, \\
\partial_t \phi + \bar{u} \partial_x \phi + \bar{\phi} \partial_x u &= 0.
\end{align*}
\] (6.19) (6.20) (6.21)
while the subset (6.17) and (6.18) becomes
\[
\begin{align*}
\partial_t u - f v + \partial_x \phi &= 0, \\
\partial_t v + f u &= 0, \\
\partial_t \phi + c_0^2 \partial_x u &= 0.
\end{align*}
\] (6.22) (6.23) (6.24)

For later reference we note that the geostrophic velocity components of the basic state are given by
\[
\bar{u} = -\frac{1}{f} \partial_y \bar{\phi} \quad \text{and} \quad \bar{v} = \frac{1}{f} \partial_x \bar{\phi},
\] (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, $\nabla_H \cdot u = 0$. We may then introduce a streamfunction $\psi$ by defining

$$u = k \times \nabla_H \psi, \quad \text{or} \quad u = -\partial_y \psi \quad \text{and} \quad v = \partial_x \psi. \quad (6.26)$$

Operating on (6.17) using the operator $k \cdot \nabla_H \times$, and remembering that the Coriolis parameter is a function of the latitude $y$, we get

$$\partial_t \zeta + \beta \partial_x \psi = 0, \quad (6.27)$$

where $\zeta = k \cdot \nabla_H \times u = \nabla_H^2 \psi$ is the vorticity, $\beta = \partial_y f$ is the rate of change of the Coriolis parameter with latitude, and where we have made use of (6.26) and that $\nabla_H \cdot u = 0$. We note that (6.27) is the barotropic vorticity equation whose solutions contains the Rossby waves. In fact if substitute for a Fourier component, that is, let

$$\psi = \psi_0 e^{i(\alpha x + ly - \omega t)}, \quad (6.28)$$

we arrive at the well known dispersion relation for Rossby waves, viz.

$$c = \frac{\omega}{\alpha} = -\frac{\beta}{(\alpha^2 + l^2)}. \quad (6.29)$$

The group velocity, as defined by (5.69), is then

$$c_g = c + \frac{\beta \alpha^2}{\alpha^2 + l^2}. \quad (6.30)$$

Thus the group velocity is negative implying that while the Rossby wave itself is propagating eastward the energy is propagating westward.

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

$$\partial_t u = -\partial_x \phi, \quad (6.31)$$

$$\partial_t \phi = -c_g^2 \partial_x u. \quad (6.32)$$

Note that when $f = 0$ (6.23) is obsolete ($\nu = \text{constant} = 0$) and we are left with two equations for the two unknowns $u$ and $\phi$. Furthermore, substituting (6.31) into (6.32) we get

$$\partial_t^2 \phi - c_g^2 \partial_x^2 \phi = 0, \quad (6.33)$$
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

\[ \phi = \phi_0 e^{i\alpha(x-ct)} \quad \text{and} \quad u = u_0 e^{i\alpha(x-ct)} \]  

(6.34)

where \( u_0 \) and \( \phi_0 \) 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

\[ -cu_0 + \phi_0 = 0, \]  

(6.35)

\[ c_0^2 u_0 - c\phi_0 = 0. \]  

(6.36)

Solving with respect to \( u_0 \) or \( \phi_0 \) we get the dispersion relation

\[ c^2 - c_0^2 = 0. \]  

(6.37)

Thus we get two solutions for the wave speed \( c \), namely

\[ c_{1,2} = \pm c_0, \]  

(6.38)

implying that the solution for the geopotential\(^5\) is

\[ \phi = \phi_1 e^{i\alpha(x-c_0t)} + \phi_2 e^{i\alpha(x+c_0t)}. \]  

(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

\[ u_{j+1}^{n+1} - u_j^{n-1} = -\frac{\Delta t}{\Delta x} (\phi_{j+1}^{n} - \phi_{j-1}^{n}) \]  

(6.40)

\[ \phi_{j+1}^{n+1} - \phi_j^{n-1} = -c_0^2 \frac{\Delta t}{\Delta x} (u_{j+1}^{n} - u_{j-1}^{n}). \]  

(6.41)

\(^5\)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,

\[ u^n_j = U_ne^{i\alpha j \Delta x} \quad \text{and} \quad \phi^n_j = \Phi_ne^{i\alpha j \Delta x}. \]  

(6.42)

Substituing (6.42) into (6.40) and (6.41) we get

\[ U_{n+1} - U_{n-1} = -2i\gamma \Phi_n, \]  

(6.43)

\[ \Phi_{n+1} - \Phi_{n-1} = -2i c_0^2 \gamma U_n, \]  

(6.44)

where

\[ \gamma = \frac{\Delta t}{\Delta x} \sin \alpha \Delta x. \]  

(6.45)

Since \( n \) is only a time step counter we may rewrite (6.43) to give

\[ U_{n+2} - U_n = -2i\gamma \Phi_{n+1}, \]  

(6.46)

and similarly

\[ U_n - U_{n-2} = -2i\gamma \Phi_{n-1}. \]  

(6.47)

Subtracting (6.47) from (6.46) we get

\[ U_{n+2} - 2U_n + U_{n-2} = -2i \gamma (\Phi_{n+1} - \Phi_{n-1}). \]  

(6.48)

Finally substituting for \((\Phi_{n+1} - \Phi_{n-1})\) using (6.44) we get

\[ U_{n+2} - 2\lambda U_n + U_{n-2} = 0, \quad \lambda = 1 - 2c_0^2 \gamma^2. \]  

(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

\[ G_{1,2} = -\lambda \pm \sqrt{\lambda^2 - 1} = -\lambda \pm i \sqrt{1 - \lambda^2}. \]  

(6.50)

Thus requiring \( \lambda^2 \leq 1 \) the growth factor is complex and has an imaginary part. Under these circumstances

\[ |G_{1,2}| = \sqrt{\lambda^2 + 1 - \lambda^2} = 1, \]  

(6.51)

and the scheme is \textit{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

\[ c_0^2 \left( \frac{\Delta t}{\Delta x} \right)^2 \leq 1 \quad \Rightarrow \quad C = c_0 \frac{\Delta t}{\Delta x} \leq 1, \quad \text{or} \quad \Delta t \leq \frac{\Delta x}{c_0}, \]  

(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 is much more stringent 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

$$G' = \pm \sqrt{G}.$$

Recalling the formula for a square root of a complex number\(^7\) we get

$$|G'_{m,n}| = \sqrt{G_n} = \sqrt{\frac{1 + \lambda}{2} + \frac{1 - \lambda}{2}} = 1,$$

as expected. Thus whenever we get a fourth order equation to solve for the growth factor that we may reduce 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 is always the same.

We recall from Sections 5.11 and 5.12 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

$$u_j^n = U_0 e^{i \alpha (j \Delta x - c_0 \Delta t)} \quad \text{and} \quad \phi_j^n = \Phi_0 e^{i \alpha (j \Delta x - c_0 \Delta t)}.$$  

(6.55)

Substituting (6.55) into (6.40) and (6.41) we get

$$\sin(\alpha c_0 \Delta t) U_0 - \gamma \Phi_0 = 0,$$

(6.56)

$$-c_0^2 \gamma U_0 + \sin(\alpha c_0 \Delta t) \Phi_0 = 0,$$

(6.57)

and hence that

$$c = \frac{1}{\alpha \Delta t} \arcsin(\gamma c_0) = \frac{1}{\alpha \Delta t} \arcsin(C \sin \alpha \Delta x),$$

(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 exactly the same numerical dispersion as we deduced for the advection equation except that the advection velocity is replaced by the phase speed. For a more detailed investigation the reader is referred to Grotjhan and O’Brien (1976).

\(^6\)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 $b' = \text{sgn}(b) \sqrt{-\frac{a - \sqrt{a^2 + b^2}}{2}}$. 

101
Regarding the unphysical mode this may be investigated much the same way as we did in Section 5.12, 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 5.13.

**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 Siellecki (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,

\[
\begin{align*}
    u^n_{j+1} - u^n_j &= -\frac{\Delta t}{2\Delta x} (\phi^n_{j+1} - \phi^n_{j-1}), \\
    \phi^n_{j+1} - \phi^n_j &= -c_0^2 \frac{\Delta t}{2\Delta x} (u^n_{j+1} - u^n_{j-1}).
\end{align*}
\]

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. 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. Defining the growth factor by \( G = U_{n+1}/U_n \) we find that the latter is determined by the equation

\[
G^2 - 2\lambda' G + 1 = 0,
\]

where \( \lambda' = 1 - \gamma^2 c_0^2 \), and hence that

\[
G_{1,2} = \lambda' \pm i\sqrt{1 - \lambda'^2}.
\]

We note that iff \( \lambda'^2 \leq 1 \) then \( |G_{1,2}| = 1 \) and hence the scheme has no dissipation. The energy is therefore conserved despite the fact that the scheme is semi-implicit. We underscore that the scheme is stable if and only if \( \lambda'^2 \leq 1 \) is true. We therefore require that \( c_0 \frac{\Delta t}{\Delta x} \leq \sqrt{2} \), or that the Courant number is less than or equal to the square root of two. The condition for stability of the forward-backward scheme is therefore less stringent than the leapfrog (CTCS) scheme.

**The semi-Lagrangian scheme**

As when treating the advection equation (Section 5.10) 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 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 \( \lambda \) and add the result to (6.31). We then get

\[
(\partial_t + \lambda c_0^2 \partial_x)u + \lambda \left( \partial_t + \frac{1}{\lambda} \partial_x \right)\phi = 0.
\]  

(6.63)

Next we define an operator \( \frac{D^*}{dt} \) such that

\[
\frac{D^*}{dt} = \partial_t + \frac{D^* x}{dt} \partial_x,
\]

(6.64)

where we require

\[
\frac{D^* x}{dt} = \lambda c_0^2 = \frac{1}{\lambda}.
\]

(6.65)

Hence we have two solutions for the unknown function \( \lambda \)

\[
\lambda_{1,2} = \pm \frac{1}{c_0},
\]

(6.66)

which gives us the two characteristic equations

\[
\frac{D^*_{1,2} x}{dt} = \pm c_0,
\]

(6.67)

and the two operators \( \frac{D^*_{1,2}}{dt} = \partial_t \pm c_0 \partial_x \). We therefore get two compatibility equations

\[
\frac{D^*_{1} \mathcal{R}^+}{dt} = 0 \quad \text{along} \quad \frac{D^*_{1} x}{dt} = +c_0,
\]

(6.68)

\[
\frac{D^*_{2} \mathcal{R}^-}{dt} = 0 \quad \text{along} \quad \frac{D^*_{2} x}{dt} = -c_0,
\]

(6.69)

where

\[
\mathcal{R}^\pm = u \pm \frac{1}{c_0} \phi
\]

(6.70)

are commonly referred to as the two Riemann invariants\(^8\). We underscore that solutions to (6.68) and (6.69) are also solutions to (6.31) and (6.32). This is indeed the very reason why we refer to (6.68) and (6.69) as the compatibility equations. We may therefore solve (6.68) and (6.69) numerically, much the same way as we outlined in Section 5.7, that is, using the semi-Lagrangian technique.

\(^8\)The wording invariants is used since (6.68) and (6.69) 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.
6.2 Linear, non-rotating

THE SHALLOW WATER PROBLEM

Figure 6.2: Sketch of the semi-Lagrangian technique. The distance between the grid points are $\Delta t$ in the vertical and $\Delta x$ in the horizontal direction. The sloping solid, blue line is the positive characteristic through the grid point $j, n + 1$ (marked $+$), while the solid, red line is the negative characteristic through the same grid point (marked $-$. They are derived from (6.68) and (6.69) and the slopes are respectively given by $\pm c_0$. 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 $x_{j+1}$.

We start by noting that the slope of the two characteristics are given by (6.67). As illustrated in Figure 6.2 the characteristics through the grid point $(x_j, t^{n+1})$ crosses 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 (6.67). The result is

$$x_P = x_j - c_0\Delta t \quad \text{and} \quad x_Q = x_j + c_0\Delta t.$$  \hfill (6.71)

Since the compatibility equations (6.68) and (6.69) tells us that the two Riemann invariants are conserved along their respective characteristic we get

$$u_j^{n+1} + \frac{1}{c_0}\phi_j^{n+1} = u_P^n + \frac{1}{c_0}\phi_P^n, \quad (6.72)$$

$$u_j^{n+1} - \frac{1}{c_0}\phi_j^{n+1} = u_Q^n - \frac{1}{c_0}\phi_Q^n, \quad (6.73)$$

where $u_{P,Q}^n$ and $\phi_{P,Q}^n$ are the value of $u$, respectively $\phi$ at the points $P$ and $Q$ at the previous time.
level \( n \). Solving with respect to the two unknowns \( u_{j}^{n+1} \) and \( \phi_{j}^{n+1} \), respectively, we get

\[
    u_{j}^{n+1} = \frac{1}{2} \left( u_{Q}^{n} + u_{P}^{n} \right) - \frac{1}{2c_{0}} \left( \phi_{Q}^{n} - \phi_{P}^{n} \right),
\]

(6.74)

and

\[
    \phi_{j}^{n+1} = \frac{1}{2} \left( \phi_{Q}^{n} + \phi_{P}^{n} \right) - \frac{1}{2c_{0}} \left( u_{Q}^{n} - u_{P}^{n} \right).
\]

(6.75)

It remains to determine \( u_{P,Q}^{n} \) and \( \phi_{P,Q}^{n} \). 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

\[
    u_{P}^{n} = \left( 1 - C \right) u_{j}^{n} + C u_{j-1}^{n}, \quad u_{Q}^{n} = \left( 1 - C \right) u_{j}^{n} + C u_{j+1}^{n},
\]

(6.76)

and

\[
    \phi_{P}^{n} = \left( 1 - C \right) \phi_{j}^{n} + C \phi_{j-1}^{n}, \quad \phi_{Q}^{n} = \left( 1 - C \right) \phi_{j}^{n} + C \phi_{j+1}^{n},
\]

(6.77)

where \( C = c_{0} \Delta t / \Delta x \) is the Courant number. Substituting (6.76) and (6.77) into (6.74) and (6.75) we finally get

\[
    u_{j}^{n+1} = \left( 1 - C \right) u_{j}^{n} + \frac{1}{2} C \left[ \left( u_{j+1}^{n} - \frac{1}{c_{0}} \phi_{j+1}^{n} \right) + \left( u_{j-1}^{n} + \frac{1}{c_{0}} \phi_{j-1}^{n} \right) \right],
\]

(6.78)

and

\[
    \phi_{j}^{n+1} = \left( 1 - C \right) u_{j}^{n} + \frac{1}{2} C \left[ \left( u_{j+1}^{n} - \frac{1}{c_{0}} \phi_{j+1}^{n} \right) + \left( u_{j-1}^{n} + \frac{1}{c_{0}} \phi_{j-1}^{n} \right) \right].
\]

(6.79)

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 actuallay larger than one, which for another scheme, 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. Focussing 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 \( x \in \langle 0, L \rangle \) 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, \( u \big|_{x=0} = 0 \) and \( u \big|_{x=L} = 0 \) \( \forall t \). 

(6.80)

Letting \( x_j = (j - 1)\Delta x \) in which \( x_1 = 0 \) and \( x_J = L \) (6.80) numerically translates to

\[
\begin{align*}
  u^n_1 &= 0 \quad \text{and} \quad u^n_J = 0 \quad \forall n. 
\end{align*}
\]

(6.81)

Note that in this case no boundary condition is specified for \( \phi \), that is, \( \phi^n_1 \) and \( \phi^n_J \) are unknown. To show that this causes a problem when employing the CTCS scheme we first rewrite (6.40) and (6.41) to give

\[
\begin{align*}
  u^{n+1}_j &= u^{n-1}_j - \frac{\Delta t}{\Delta x} (\phi^{n+1}_{j+1} - \phi^{n+1}_{j-1}), \quad (6.82) \\
  \phi^{n+1}_j &= \phi^{n-1}_j - c_o^2 \frac{\Delta t}{\Delta x} (u^{n+1}_{j+1} - u^{n+1}_{j-1}), \quad (6.83)
\end{align*}
\]

to be solved for all the “wet” points \( j = 2(1)J - 1 \) and \( n = 1(1)\infty \). To find the value at an arbitrary time level \( n \) at the first “wet” point \( j = 2 \) we get by use of (6.82) and (6.83)

\[
\begin{align*}
  u^{n+1}_2 &= u^{n-1}_2 - \frac{\Delta t}{\Delta x} (\phi^{n}_3 - \phi^{n}_1), \quad (6.84) \\
  \phi^{n+1}_2 &= \phi^{n-1}_2 - c_o^2 \frac{\Delta t}{\Delta x} (u^{n}_3 - u^{n}_1). \quad (6.85)
\end{align*}
\]

The term in (6.84) that poses a problem is the one containing \( \phi^n_1 \) since the number of allowable boundary conditions do not allow us to specify it. We observe that we have a similar problem at the other boundary \( x = L \). In fact evaluating (6.82) and (6.83) for the last “wet” point \( j = J - 1 \) we get

\[
\begin{align*}
  u^{n+1}_{J-1} &= u^{n-1}_{J-1} - \frac{\Delta t}{\Delta x} (\phi^n_{J} - \phi^n_{J-2}), \quad (6.86) \\
  \phi^{n+1}_{J-1} &= \phi^{n-1}_{J-1} - c_o^2 \frac{\Delta t}{\Delta x} (u^n_{J} - u^n_{J-2}). \quad (6.87)
\end{align*}
\]

Again the term that poses a problem is the term \( \phi^n_J \), that is, the value of \( \phi \) at the boundary.

---

9As alluded to in Section 5.9 we have to use an Euler scheme to get the solution at \( n=1 \) by letting \( n = 0 \) in (6.82). However, this does not avoid the problem to be described.
Figure 6.3: Comparison of the structure of (a) an unstaggered and (b) a staggered grid in one spatial dimension. The circles are associated with \( \phi \)-points, while the ellipses are associated with \( u \)-points. The illustrated staggering in panel (b) is such that the \( \phi \)-points and \( u \)-points are one half grid distance apart, but at the same time level.

If we try to remedy this problem by using option (ii) above, that is, specify \( \phi \) 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 \( \phi \) 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 \( \phi \) at the same points, we simply 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 \( x = x_j + \frac{1}{2} \) points and \( \phi \) at \( x_j \) points. Using the notation depicted in Figure 6.3(b) to avoid the cumbersome use of \( j + \frac{1}{2} \) notations, the CTCS finite difference approximation to (6.31) and (6.32) now becomes

\[
\begin{align*}
    u_j^{n+1} &= u_j^{n-1} - \frac{2\Delta t}{\Delta x} \left( \phi_{j+1}^n - \phi_j^n \right), \\
    \phi_j^{n+1} &= \phi_j^{n-1} - \frac{c^2}{\Delta x} \frac{2\Delta t}{\Delta x} \left( u_j^n - u_{j-1}^n \right),
\end{align*}
\]

Note the appearances of the factor 2 in the second term on the right-hand side of (6.88) and (6.89). It appears because the distance between two adjacent points in the finite difference approximation of \( \partial_x \) in the staggered formulation is \( \Delta x \) rather than \( 2\Delta x \), while the centered in time scheme still carries \( 2\Delta t \). We emphasize that (6.88) and (6.89) are correct if and only if the two grids are staggered exactly one half grid length.

It remains to answer whether the staggering has an impact on the stability. To investigate this we first note that care has to be exercised when constructing the discrete Fourier components.
Since they are associated with the point numbering we have to take into account that \( u \) and \( \phi \) points are one half grid-length apart. Thus the discrete Fourier components with the notation used in Figure 6.3 are

\[
\phi^n_j = \Phi^n e^{i\alpha j \Delta x} \quad \text{and} \quad u^n_j = U^n e^{i\alpha (j + \frac{1}{2}) \Delta x}
\]

respectively. Substituting these expressions into (6.88) and (6.89) we get

\[
U_{n+1} - U_{n-1} = -4i\gamma \Phi_n, \quad (6.91)
\]

\[
\Phi_{n+1} - \Phi_{n-1} = -4i\gamma c_0^2 U_n. \quad (6.92)
\]

where

\[
\gamma' = \frac{\Delta t}{\Delta x} \sin \left( \frac{\alpha \Delta x}{2} \right) \quad (6.93)
\]

Eliminating \( U_n \) we get

\[
\Phi_{n+2} - 2\Phi_n + \Phi_{n-2} = -16\gamma'^2 c_0^2 \Phi_n. \quad (6.94)
\]

Moreover, defining a growth factor by

\[
G \equiv \frac{\Phi_{n+2}}{\Phi_n} \quad (6.95)
\]

and using the formula \( 2 \sin^2 \frac{\psi}{2} = 1 - \cos \psi \) we get

\[
G^2 - 2\lambda' G + 1 = 0, \quad (6.96)
\]

where

\[
\lambda' = 1 - 4c_0^2 \left( \frac{\Delta t}{\Delta x} \right)^2 \left( 1 - \cos \alpha \Delta x \right). \quad (6.97)
\]

The growth factor therefore has two complex conjugate solutions given by

\[
G_{1,2} = \lambda' \pm i\sqrt{1 - \lambda'^2}. \quad (6.98)
\]

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

\[
C \leq \frac{1}{2}, \quad \text{or} \quad \Delta t \leq \frac{\Delta x}{2c_0}. \quad (6.99)
\]

We observe that (6.99) 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 \( \Delta x_{\text{stagg}} \), is simply \( \Delta x_{\text{stagg}} = \Delta x/2 \). Using \( \Delta x_{\text{stagg}} \) instead of \( \Delta x \) the CFL condition becomes the expected

\[
\Delta t \leq \frac{\Delta x_{\text{stagg}}}{c_0}. \quad (6.100)
\]
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). Thus we assume a wave solution,

\[ X = X_0 e^{i \alpha (x-ct)}, \]  

where \( \alpha \) is the wavenumber in the \( x \) direction and \( X \) denotes a vector consisting of the three dependent variables, that is,

\[ X = \begin{bmatrix} u \\ v \\ \phi \end{bmatrix} \quad \text{and} \quad X_0 = \begin{bmatrix} u_0 \\ v_0 \\ \phi_0 \end{bmatrix}, \]  

where \( X_0 \) is the amplitude.

Inserting (9.25) into (6.22) - (6.24) we get

\[ -i \alpha cu_0 - fv_0 + i \alpha \phi_0 = 0, \]  
\[ -i \alpha cv_0 + fu_0 = 0, \]  
\[ c \phi_0 - c_0^2 u_0 = 0, \]  

To find the dispersion relation we first multiply (6.103) by \(-i \alpha c\) and (6.104) by \(f\) and add the results together. By substituting the result into (6.105) we get

\[ c \left\{ c^2 - c_0^2 \left[ 1 + \left( \frac{1}{\alpha L_R} \right)^2 \right] \right\} = 0, \]  

where

\[ L_R = c_0 / f \]  

is Rossby’s deformation radius. Thus one solution is \( c_1 = 0 \), a stationary wave. The two remaining solutions are

\[ c_{2,3} = \pm c_0 \sqrt{1 + \left( \frac{1}{\alpha L_R} \right)^2}. \]  

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 under in the radical.

\[ ^{10} \text{Recall that any solution of (6.22) - (6.24), under the assumption that the solution is a good function, may be represented by an infinite number of waves of different wavelengths and amplitudes in accord with Section 2.11 on page 29.} \]
6.4 Linear and rotating

The same procedure may also be used if the basic state is in geostrophic balance, that is, using the linearized equations (6.19) - (6.21) on page 97. Substituting (6.102) into (6.19) - (6.21) we get

\[
\begin{align*}
    i\alpha (\bar{u} - c) u_0 - f v_0 + i\alpha \phi_0 &= 0, \\
    f u_0 + i\alpha (\bar{u} - c) v_0 &= 0, \\
    \bar{\phi} u_0 + (\bar{u} - c) \phi_0 &= 0.
\end{align*}
\]

(6.109) \hspace{1cm} (6.110) \hspace{1cm} (6.111)

Following the same procedure the dispersion relation becomes

\[
(\bar{u} - c) \left[ -\alpha^2 (\bar{u} - c)^2 + f^2 + \alpha^2 \bar{\phi} \right] = 0.
\]

(6.112)

Thus again there are three solutions for the phase speed \(c\), namely

\[
\begin{align*}
    c_1 &= \bar{u}, \\
    c_{2,3} &= \bar{u} \pm c_0 \sqrt{1 + \left( \frac{1}{\alpha L_R} \right)^2},
\end{align*}
\]

(6.113) \hspace{1cm} (6.114)

where

\[
c_0 = \sqrt{\bar{\phi}} = \sqrt{g\bar{H}}.
\]

(6.115)

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 (6.113) into (6.109). The latter equation then becomes

\[
-f v + i\alpha \phi = 0 \quad \text{or} \quad v = \frac{1}{f} i\alpha \phi.
\]

(6.116)

Using the Fourier solution backwards we thus recover the geostrophic balance (1.40), that is,

\[
v = \frac{1}{f} \partial_x \phi.
\]

(6.117)

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

\[

t_{n+1}^j - t_{n-1}^j = 2f \Delta t v_n^j - \frac{\Delta t}{\Delta x} (\phi_{j+1}^n - \phi_{j-1}^n),
\]

(6.118)

\[

t_{n+1}^j - t_{n-1}^j = -2f \Delta t u_n^j,
\]

(6.119)

\[
\phi_{j+1}^n - \phi_{j-1}^n = -c_0^2 \frac{\Delta t}{\Delta x} (u_{j+1}^n - u_{j-1}^n).
\]

(6.120)

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

\[

t_j^n = U_ne^{i\alpha j \Delta x}, \quad v_j^n = V_ne^{i\alpha j \Delta x}, \quad \phi_j^n = \Phi_ne^{i\alpha j \Delta x}.
\]

(6.121)

Substituting (6.121) into (6.118) - (6.120) we get

\[

U_{n+1} - U_{n-1} = 2f \Delta t V_n - 2i\gamma \Phi_n,
\]

(6.122)

\[

V_{n+1} - V_{n-1} = -2f \Delta t U_n,
\]

(6.123)

\[

\Phi_{n+1} - \Phi_{n-1} = -2i\gamma c_0^2 U_n,
\]

(6.124)

where as before

\[
\gamma = \frac{\Delta t}{\Delta x} \sin \alpha \Delta x.
\]

(6.125)

Eliminating \(V_n\) and \(\Phi_n\) we get\(^{11}\)

\[

U_{n+2} - 2\lambda U_n + U_{n-2} = 0,
\]

(6.126)

where

\[
\lambda = 1 - 2\gamma^2 c_0^2 - 2f^2 \Delta t^2.
\]

(6.127)

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 inherent in the expression for \(\lambda\). To ensure that the growth factor is complex we must require \(\lambda^2 \leq 1\). This is satisfied if and only if

\[
\gamma^2 c_0^2 + f^2 \Delta t^2 \leq 1 \quad \text{or} \quad c_0^2 \left(\frac{\Delta t}{\Delta x}\right)^2 \sin^2 \alpha \Delta x + f^2 \Delta t^2 \leq 1.
\]

(6.128)

\(^{11}\)This most efficiently done by first raising \(n\) by one in (6.122) giving an expression involving \(U_{n+2}, U_n, V_{n+1}\) and \(\Phi_{n+1}\). Next decreasing \(n\) by one gives an expression containing \(U_n, U_{n-2}, V_{n-1}\) and \(\Phi_{n-1}\). Subtracting and using (6.123) and (6.124) then results in (6.126).
Thus the CFL criterion for stability becomes,

\[ \Delta t \leq \frac{\Delta x}{c_0} \left[ 1 + \left( \frac{\Delta x}{L_R} \right)^2 \right]^{-\frac{1}{2}}, \tag{6.129} \]

where \( L_R \) is Rossby’s deformation radius as defined in (6.107). If we choose the mesh size \( \Delta x \) 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

\[ \Delta t < \frac{\Delta x}{c_0} \quad \text{or} \quad C < 1. \tag{6.130} \]

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 (6.130) 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 \( O(1) \), 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 6.5 on page 120).

**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

\[
\begin{align*}
  u_j^{n+1} &= u_j^n + f \Delta t v_j^n - \frac{\Delta t}{2 \Delta x} (\phi_{j+1}^n - \phi_{j-1}^n), \\
  v_j^{n+1} &= v_j^n - f \Delta t u_j^{n+1}, \\
  \phi_j^{n+1} &= \phi_j^n - c_0^2 \frac{\Delta t}{2 \Delta x} (u_{j+1}^{n+1} - u_{j-1}^{n+1}).
\end{align*}
\tag{6.131-6.133}
\]

\(^{12}\)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 (6.130) that \( \Delta t < 141 \) s or slightly more than two minutes.
We note again that the scheme reduces to (6.59) and (6.60) when \( f = 0 \). We emphasize that the order in which (6.131) - (6.133) 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

\[
\begin{align*}
    \phi_j^{n+1} &= \phi_j^n - c_0^2 \frac{\Delta t}{2\Delta x} (u_{j+1}^n - u_{j-1}^n), \\
    u_j^{n+1} &= u_j^n + f \Delta t v_j^n - \frac{\Delta t}{2\Delta x} (\phi_{j+1}^{n+1} - \phi_{j-1}^{n+1}), \\
    v_j^{n+1} &= v_j^n - f \Delta t u_j^{n+1},
\end{align*}
\] (6.134)

or

\[
\begin{align*}
    v_j^{n+1} &= v_j^n - f \Delta t u_j^n, \\
    \phi_j^{n+1} &= \phi_j^n - c_0^2 \frac{\Delta t}{2\Delta x} (u_{j+1}^n - u_{j-1}^n), \\
    u_j^{n+1} &= u_j^n + f \Delta t v_j^{n+1} - \frac{\Delta t}{2\Delta x} (\phi_{j+1}^{n+1} - \phi_{j-1}^{n+1}).
\end{align*}
\] (6.138)

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 (6.121), then we eliminate two of the amplitudes, say \( V_n \) and \( \Phi_n \). Finally we define the growth factor by

\[
G^2 - 2\lambda' G + 1 = 0,
\] (6.140)

where

\[
\lambda' = 1 - \gamma^2 c_0^2 - f^2 \Delta t^2.
\] (6.141)

Thus the scheme is stable provided the condition

\[
\Delta t \leq \frac{2\Delta x}{c_0} \left[ 1 + \left( \frac{\Delta x}{L_R} \right)^2 \right]^{-\frac{1}{2}}
\] (6.142)

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 \( \lambda \) and add it to (6.22). We then get

\[
\frac{D_{1.2}^* \mathcal{R}^\pm}{dt} = f v \quad \text{along} \quad \frac{D_{1.2}^* x}{dt} = \pm c_0,
\] (6.143)

where \( \mathcal{R}^\pm \) are the Riemann invariants displayed in (6.70) on page 103. These are exactly the same two characteristic as in (6.68) and (6.69), that is, (6.67), while the compatibility equations...
deviates from (6.68) and (6.69) 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,

$$D^*_3 v = -fu$$

along

$$D^*_3 x = 0,$$

(6.144)

where the third operator is defined by

$$D^*_3 = \partial_t + D^*_3 x \partial_x.$$  

(6.145)

We note that the third characteristic equation is $D^*_3 x = 0$ and hence that it has an infinite slope.

To construct a finite difference scheme we follow the procedure leading to (6.71) - (6.73). Hence we get

$$u^{n+1}_j + \frac{\phi^{n+1}_j}{c_0} = u^n_P + \frac{\phi^n_P}{c_0} + \frac{1}{2} f \Delta t (v^{n+1}_j + v^n_P),$$

(6.146)

$$u^{n+1}_j - \frac{\phi^{n+1}_j}{c_0} = u^n_Q - \frac{\phi^n_Q}{c_0} + \frac{1}{2} f \Delta t (v^{n+1}_j + v^n_Q),$$

(6.147)

$$v^{n+1}_j = v^n_j - \frac{1}{2} f \Delta t (u^{n+1}_j + u^n_j),$$

(6.148)

that is, three equation containing the three unknowns $u^{n+1}_j$, $v^{n+1}_j$ and $\phi^{n+1}_j$. Since the integration is along the characteristics, we emphasize the use of $\frac{1}{2} f \Delta t (v^{n+1}_j + v^n_P)$ in (6.146) and the similar terms in (6.147) and (6.148). Solving with respect to each of the variables at the point $x_j, t^{n+1}$ we get

$$v^{n+1}_j = \left( 1 + \frac{1}{2} f^2 \Delta t^2 \right)^{-1} F_u,$$

(6.149)

$$v^{n+1}_j = v^n_j - \frac{1}{2} f \Delta t \left[ \left( 1 + \frac{1}{2} f^2 \Delta t^2 \right)^{-1} F_u + u^n_j \right],$$

(6.150)

$$\phi^{n+1}_j = F_\phi,$$

(6.151)

where

$$F_u = \frac{1}{2} \left( u^n_P + u^n_Q \right) + \frac{1}{2 c_0} \left( \phi^n_P - \phi^n_Q \right) + \frac{1}{4} f \Delta t \left( 2v^n_j - f \Delta tu^n_j + v^n_P + v^n_Q \right),$$

(6.152)

$$F_\phi = \frac{1}{2 c_0} \left( u^n_P - u^n_Q \right) + \frac{1}{2} \left( \phi^n_P + \phi^n_Q \right) + \frac{1}{2} c_0 f \Delta t \left( v^n_P - v^n_Q \right),$$

(6.153)

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 $u^n_{P,Q}$ and $\phi^n_{P,Q}$ we also need to determine $v^n_{P,Q}$ by interpolation. Finally we note that if we let $f = 0$ the above equations reduce to (6.74) and (6.75).
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

\[
\begin{align*}
\partial_t u + u \partial_x u - fv &= -\partial_x \phi, \\
\partial_t v + u \partial_x v + fu &= 0, \\
\partial_t \phi + u \partial_x \phi &= -\phi \partial_x u.
\end{align*}
\]  

(6.154)
\[ \quad (6.155) \]
\[ \quad (6.156) \]

We note that we retain three equations with three unknowns \(u, v\) and \(\phi\) as dependent variables. The independent variables are time \(t\) and the horizontal direction \(x\).

Rewriting (6.154) - (6.156) in terms of the volume flux \(U = hu, V = hv\), and noting that \(\phi = gh\), we get

\[
\begin{align*}
\partial_t U + \partial_x \left( \frac{U^2}{h} \right) - fV &= -\frac{1}{2}g \partial_x h^2, \\
\partial_t V + \partial_x \left( \frac{UV}{h} \right) + fU &= 0, \\
\partial_t h + \partial_x U &= 0.
\end{align*}
\]  

(6.157)
\[ \quad (6.158) \]
\[ \quad (6.159) \]

The advantage of using (6.157) - (6.159) instead of (6.154) - (6.156) is that the continuity equation (6.159) becomes linear, and that the non-linear terms in (6.157) and (6.158) 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 (6.157) - (6.159) we get

\[
\begin{align*}
U_{j}^{n+1} &= U_{j}^{n-1} + 2f \Delta t V_{j}^{n} + A_{j}^{n} + P_{j}^{n} \\
V_{j}^{n+1} &= V_{j}^{n-1} + 2f \Delta t U_{j}^{n} + B_{j}^{n} \\
h_{j}^{n+1} &= h_{j}^{n-1} - \frac{\Delta t}{\Delta x} \left( U_{j+1}^{n} - U_{j-1}^{n} \right),
\end{align*}
\]  

(6.160)
\[ \quad (6.161) \]
\[ \quad (6.162) \]

where

\[
\begin{align*}
A_{j}^{n} &= -\frac{\Delta t}{\Delta x} \left( \frac{U^2}{h} \right)_{j+1}^{n} - \left( \frac{U^2}{h} \right)_{j-1}^{n}, \\
B_{j}^{n} &= -\frac{\Delta t}{\Delta x} \left( \frac{UV}{h} \right)_{j+1}^{n} - \left( \frac{UV}{h} \right)_{j-1}^{n}, \\
P_{j}^{n} &= -\frac{g \Delta t}{2 \Delta x} \left( h^2 \right)_{j+1}^{n} - \left( h^2 \right)_{j-1}^{n}.
\end{align*}
\]  

(6.163)
\[ \quad (6.164) \]
\[ \quad (6.165) \]
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

\[
U_{j}^{n+1} = U_{j}^{n-1} + 2f \Delta t V_{j}^{n} + A_{j}^{n} + P_{j}^{n}, \quad (6.166)
\]

\[
V_{j}^{n+1} = V_{j}^{n-1} + 2f \Delta t U_{j}^{n} + B_{j}^{n}, \quad (6.167)
\]

\[
h_{j}^{n+1} = h_{j}^{n-1} - \frac{2\Delta t}{\Delta x} \left( U_{j}^{n} - U_{j-1}^{n} \right), \quad (6.168)
\]

where \( A_{j}^{n} \) and \( B_{j}^{n} \) are as before, while \( P_{j}^{n} \), because of the staggering, becomes

\[
P_{j}^{n} = -\frac{g \Delta t}{\Delta x} \left( [h^{2}]_{j+1}^{n} - [h^{2}]_{j}^{n} \right). \quad (6.169)
\]

**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 by multiplying (6.154) by a yet unknown function \( \lambda \) and adding it to (6.156). Replacing \( \phi \) by the propagation speed \( c = \sqrt{\phi} \), we get

\[
\left[ \partial_t u + (u + \lambda c^2) \partial_x \right] u + 2c \lambda \left[ \partial_t + (u + \lambda c^2) \partial_x \right] c = f v. \quad (6.170)
\]

We require that the operators are the same, which gives \( \lambda_{1,2} = \pm \frac{1}{c} \), and hence that the characteristic equations are

\[
\frac{D_{1,2}^{*} x}{dt} = u \pm c. \quad (6.171)
\]

Furthermore the two Riemann invariants become

\[
\mathcal{R}^{\pm} = u \pm 2c, \quad (6.172)
\]

and hence that the compatibility equations are

\[
\frac{D_{1,2}^{*} \mathcal{R}^{\pm}}{dt} = f v \quad \text{along} \quad \frac{D_{1,2}^{*} x}{dt} = u \pm c. \quad (6.173)
\]

The third compatibility and characteristic equations is found directly from (6.155), that is,

\[
\frac{D_{3}^{*} v}{dt} = -fu \quad \text{along} \quad \frac{D_{3}^{*} x}{dt} = u. \quad (6.174)
\]

Thus we get three characteristics with slopes \( u + c, u - c \) and \( u \), respectively.

Applying a simple forward in time finite difference approximations as before we get

\[
u_{j}^{n+1} + 2c_{j}^{n+1} = u_{P}^{n} + 2c_{P}^{n} + \frac{1}{2} f \Delta t \left( v_{j}^{n+1} + v_{P}^{n} \right), \quad (6.175)
\]

\[
u_{j}^{n+1} - 2c_{j}^{n+1} = u_{Q}^{n} - 2c_{Q}^{n} + \frac{1}{2} f \Delta t \left( v_{j}^{n+1} + v_{Q}^{n} \right), \quad (6.176)
\]

\[
v_{j}^{n+1} = v_{R}^{n+1} - \frac{1}{2} f \Delta t \left( u_{j}^{n+1} + u_{R}^{n} \right). \quad (6.177)
\]
The shallow water problem

With reference to Figure 6.4 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. To find their position we just integrate the characteristic equations using a simple forward in time finite difference approximation. Hence we get

$$x_P = x_j - (u_j^n + c_j^n)\Delta t, \quad x_Q = x_j - (u_j^n - c_j^n)\Delta t, \quad \text{and} \quad x_R = x_j - u_j^n\Delta t. \quad (6.179)$$

To evaluate $u_{P,Q,R}^n, v_{P,Q,R}^n$ and $c_{P,Q,R}^n$, we resort to interpolation. For instance using the lowest order of Newton’s interpolation formulae (two point, linear interpolation), we get

$$u_P^n = u_{j-1}^n + \frac{u_j^n - u_{j-1}^n}{\Delta x} (x_P - x_{j-1}). \quad (6.180)$$

We note that in order to use (6.180) we require that $x_{j-1} \leq x_P \leq x_j$ or $(u + c)\Delta t \leq \Delta x$. If this is not the case we have to find out where the nearest grid point is and then apply the lowest order interpolation. Another option is to resort to higher order interpolation. Either way we end up with three equations to solve for the three unknowns $u_j^{n+1}, v_j^{n+1}$ and $c_j^{n+1}$.

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 (6.179) 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 $x_j, t_n^{n+1}$. We are then in a position to compute improved approximations to the positions of the points $P, Q$ and $R$,

$$x_P^{(1)} = x_j - \frac{1}{2}(u_j^{(0)} + u_P^{(0)} + c_{j}^{(0)} + c_{P}^{(0)})\Delta t, \quad (6.181)$$
$$x_Q^{(1)} = x_j - \frac{1}{2}(u_j^{(0)} + u_Q^{(0)} - c_{j}^{(0)} - c_{Q}^{(0)})\Delta t, \quad (6.182)$$
$$x_R^{(1)} = x_j - \frac{1}{2}(u_j^{(0)} + u_R^{(0)})\Delta t \quad (6.183)$$

We then find updated values of $u_{P,Q,R}^{(1)}, v_{P,Q,R}^{(1)}$, and $c_{P,Q,R}^{(1)}$ by interpolation to the improved positions. In turn this enables us to update the values of the variables at the point $x_j, t_n^{n+1}$ using the formulae,

$$u_j^{(\nu)} + 2c_j^{(\nu)} = u_P^{(\nu-1)} + 2c_P^{(\nu-1)} + \frac{1}{2}f\Delta t \left(v_j^{(\nu)} + v_P^{(\nu-1)} \right), \quad (6.184)$$
$$u_j^{(\nu)} - 2c_j^{(\nu)} = u_Q^{(\nu-1)} - 2c_Q^{(\nu-1)} + \frac{1}{2}f\Delta t \left(v_j^{(\nu)} + v_Q^{(\nu-1)} \right), \quad (6.185)$$
$$v_j^{(\nu)} = v_R^{(\nu)} - \frac{1}{2}f\Delta t \left(u_j^{(\nu)} + u_R^{(\nu-1)} \right), \quad (6.186)$$

(6.187)
6.5 Non-linear and rotating THE SHALLOW WATER PROBLEM

\[
\begin{align*}
\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} &= \frac{\partial}{\partial x} \left( \frac{1}{\rho} \left( \frac{\partial P}{\partial x} - \frac{\partial Q}{\partial t} \right) \right) + \frac{\partial}{\partial t} \left( \frac{1}{\rho} \frac{\partial Q}{\partial x} \right) + f v, \\
\frac{\partial v}{\partial t} + c \frac{\partial v}{\partial x} &= \frac{\partial}{\partial x} \left( \frac{1}{\rho} \frac{\partial R}{\partial x} \right) + \frac{\partial}{\partial t} \left( \frac{1}{\rho} \frac{\partial R}{\partial x} \right),
\end{align*}
\]

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 (6.184) - (6.186), we may update the positions using the formulas

\[
\begin{align*}
\begin{align*}
x_{P}^{(\nu+1)} &= x_j - \frac{1}{2}(u_{j}^{(\nu)} + u_{P}^{(\nu)} + c_{j}^{(\nu)} + c_{P}^{(\nu)}) \Delta t, \quad (6.188) \\
x_{Q}^{(\nu+1)} &= x_j - \frac{1}{2}(u_{j}^{(\nu)} + u_{Q}^{(\nu)} - c_{j}^{(\nu)} - c_{Q}^{(\nu)}) \Delta t, \quad (6.189) \\
x_{R}^{(\nu+1)} &= x_j - \frac{1}{2}(u_{j}^{(\nu)} + u_{R}^{(\nu)}) \Delta t. \quad (6.190)
\end{align*}
\end{align*}
\]

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 (6.184) - (6.186), 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.

Figure 6.4: Sketch of the semi-Lagrangian technique for a non-linear and rotating case. The distance between the grid points are \( \Delta t \) in the vertical and \( \Delta x \) 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 (6.171). The last characteristic with slope \( u \) is the dotted black line derived from (6.174). 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 asymmetry. 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 \( x_{j+1} \).
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 3.2 on page 37 and Section 4.6 on page 54, 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 37.

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 165.

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 4.2, 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 \(\kappa\) 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.
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,

\[
\begin{align*}
\partial_t u &= A_u - \partial_x \phi, \\
\partial_t v &= A_v, \\
\partial_t \phi &= A_\phi - \Phi \partial_x u,
\end{align*}
\] (6.191, 6.192, 6.193)

where \(A_u, A_v\) and \(A_\phi\) 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 4.5 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,

\[
\begin{align*}
\frac{u^{n+1} - u^{n-1}}{2\Delta t} &= [A_u]^n - [\partial_x \phi]^{n+1}, \\
\frac{v^{n+1} - v^{n-1}}{2\Delta t} &= [A_v]^n, \\
\frac{\phi^{n+1} - \phi^{n-1}}{2\Delta t} &= [A_\phi]^n - \Phi [\partial_x u]^{n+1},
\end{align*}
\] (6.194, 6.195, 6.196)

To proceed, the first two equations are solved with respect to \(u_j^{n+1}\) and \(v_j^{n+1}\) respectively giving,

\[
\begin{align*}
u^{n+1} &= u^{n-1} + 2\Delta t [A_u]^n - 2\Delta t [\partial_x \phi]^{n+1} \\
v^{n+1} &= v^{n-1} + 2\Delta t [A_v]^n
\end{align*}
\] (6.197, 6.198)

derivation with respect to \(x\) and insertion into the equation for \(\phi\) gives a Helmholtz equation,

\[
\Phi \Delta t^2 [\partial_x^2 \phi]^{n+1} - \phi^{n+1} = B
\] (6.199)

where \(B\) contains known quantities at time levels \(n, n-1, \ldots\)

With proper boundary conditions (\(\phi\) 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 4.8 (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 \(\phi^{n+1}\), 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 $\Delta t_{bc}$. 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 $\Delta t_{bt}$. 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 number of numbers in their so called random access memory (RAM). Thus even the biggest computers are limited in their capacity. This is perhaps the main reason why numerical oceanography is less mature than numerical meteorology. Recall that at sub-polar latitudes the typical length scale of a synoptic low in the atmosphere is about 500 - 1000 km and the typical time scale a few days. In contrast the scales of a low in the ocean at the same latitude are about 10 - 50 km in length and a few weeks to months in time. These differences are associated with the difference in the Rossby radius of deformation that in the ocean is about two order of magnitudes smaller than in the atmosphere.

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 Rossby radius. This is a tolerable grid size for a numerical atmosphere model. If we, however, 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 which has a grid size of 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 numerical weather prediction (NWP) model. To obtain a similar tolerable resolution in the ocean we have to employ grids of mesh sizes 2-4 km, or 1/200th of a degree. Thus the need for RAM is much higher for an ocean model than for an atmosphere model covering the same region. In addition comes the fact that with decreased mesh size the time step is much smaller as well in order to satisfy the CFL criterion. In practice it therefore takes a much larger 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 therefore need much 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.

Global weather predictions with more than adequate resolution to resolve the atmospheric weather systems are common today. Such forecasts are run by several national institutes as well
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.
OPEN BOUNDARY CONDITIONS

as at the European Centre for Medium-range Weather Forecasts (ECMWF). However, there are still local effects, notably processes associated with irregular topography, that is not yet resolved properly by the global model. Thus most national institutes providing public meteorological services run a limited area model which is “nested” into a global model1. Nesting a finer mesh model (sometimes referred to as the inner model) into a coarser mesh model (sometimes referred to as the outer model) entails that we want to transfer results from the outer model to the inner model and vice versa. This has to be done via their common boundary. Thus one need a condition at these boundaries. Since fluid is allowed to pass freely through these boundaries they are referred to as open boundaries. Nesting of a finer mesh model into a coarser model is also sometimes referred to as dynamical downscaling since the inner model provides a solution that is a dynamically consistent downscaling of the coarser solution to a scale that takes into account the finer scales, for instance due to a more realistic representation of the topography.

The situation for the ocean is even more profound. Global numerical ocean weather predictions (NOWP) are not yet feasible at all on today’s computers. Thus none of the global ocean models that exist today resolve the oceanic weather at higher latitudes. This is particularly true for the oceanic component of the coupled global climate models, or so called AOGCMs (Atmosphere - Ocean Global Climate Models). To provide synoptic NOWPs in limited areas we therefore have to make use of nesting techniques or dynamical downscaling for the time being.

As a consequence both ocean and atmosphere models have to deal with open boundaries. At these boundaries the governing equations are still valid, which makes them very different from natural boundaries where the governing equations are replaced by a natural boundary condition as for instance the condition of no normal flow through a solid, impermeable boundary. Nevertheless, since our computational domain ends at the open boundary we have to provide a boundary condition there. Such conditions are called open boundary conditions or OBCs for short. The mathematical challenge is to construct OBCs which ensure that a solution to the governing equations exists and is unique. From a physical point of view though 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 only. 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 our solution we are not ensured that the solution we obtain is the correct one. In fact it is impossible in general to prove that the solution we obtain exists and is unique. The latter is only possible in special cases, for instance for very simplified linear systems.

One of the first to derive a numerical solution of a NWP problem applying OBCs was Jules Charney, Ragnar Fjørtoft and John von Neumann in the late 1940s. Their attempt was published in the 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. In fact they attempted to solve the barotropic, quasigeostrophic equations formulated in terms of a potential vorticity equation for a computational

1At the Norwegian Meteorological Institute the limited area model at the time of writing is HIRLAM - the High Resolution Limited Area Model

125
domain limited to a square located in the middle of the North Atlantic Ocean. Thus they had to apply OBCs at all four sides of the domain. 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 7.2 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 and correct is not new. Since these early papers on the problems encountered in specifying OBC there has been a variety of papers treating the problem. For a review see Chapman (1985); Røed and Cooper (1986, 1987); Palma and Matano (2000) and most recently Blayo and Debreu (2006).

7.1 What is an open boundary?

To help in choosing an OBC that would render the solution stable, the following definition of an open boundary was offered by Røed and Cooper (1986):

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.

Thus we demand that the conditions we impose at open boundaries satisfy certain requirements. These requirements may later be used to construct criteria by which the success of our employed OBC may be assessed. We emphasize that an open boundary is an artificial boundary between what we call our computational domain (or our area of interest) and a domain exterior to our computational domain.

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 Kelvin wave created in the interior of our domain and propagating towards the open boundary. The condition we impose at the open boundary should then be able to let the wave pass through and not be reflected, that is, none of the energy contained in the Kelvin wave should be allowed to be radiated back into the interior. Likewise a Kelvin wave created in the exterior domain, for instance by a model covering a larger domain that encompasses our computational domain, should be free to enter without being distorted or damped. Next we require that the chosen OBC leads to a stable solution (numerically). Moreover, 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.

In the following we will give some details regarding some of the common OBCs developed over the years. In the two last sections we will give details about two of the most promising
OBCs that are useful as nesting techniques as well as OBCs. These are the Flow Relaxation Scheme (e.g., Engedahl, 1995a) and the Perfectly Matched Layer 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).

### 7.2 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 therefore based their OBC formulations on simple wave equation. In its simplest form the wave equation reads,

$$\partial_t \phi + c_\phi \partial_n \phi = 0$$  \hspace{1cm} (7.1)

Here $\phi$ represents the dependent variable, $c_\phi$ is the component of the phase velocity normal the boundary associated with the variable $\phi$, while $\partial_n$ 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 $c_\phi$. From Section 5.10 we recall that $c_\phi$ is the slope of the characteristics. Thus if the choice of $c_\phi$ 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 $c_\phi$ 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,

$$\phi = \text{const.}$$  \hspace{1cm} (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 $\phi$ 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

$$\partial_n \phi = 0.$$  \hspace{1cm} (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 \( c_\phi \). If the solution is in the form of known waves, say a barotropic Kelvin wave\(^2\). Under these circumstances the phase velocity is known, and in the case of a Kelvin wave it is,

\[
c_\phi = c_0 = \sqrt{gH}
\]  
(7.4)

where \( g \) is the gravitational acceleration, and \( H \) is the equilibrium depth of a fluid column.

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\(^3\). 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)

\[
\begin{align*}
\partial_t u &= -g \partial_x h \\
\partial_t h &= -H \partial_x u
\end{align*}
\]  
(7.5) (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

\[
\partial_t^2 h - c_0^2 \partial_x^2 h = 0,
\]  
(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 (6.68) and (6.69). Revalling that \( \phi = gh \) we get

\[
\begin{align*}
\partial_t \left( u + c_0 \frac{h}{H} \right) + \frac{D}{dt} x \partial_x \left( u + c_0 \frac{h}{H} \right) &= 0 \quad \text{along} \quad \frac{D}{dt} x = c_0, \\
\partial_t \left( u - c_0 \frac{h}{H} \right) + \frac{D}{dt} x \partial_x \left( u - c_0 \frac{h}{H} \right) &= 0 \quad \text{along} \quad \frac{D}{dt} x = -c_0.
\end{align*}
\]  
(7.8) (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

\(^2\)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.

\(^3\)Since we consider a frictionless motion we may safely assume that \( u \) is independent of depth.
OPEN BOUNDARY CONDITIONS

7.2 Radiation conditions

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 \( \frac{D^*_2 x}{dt} = c_0 \) characteristic, and towards the left-hand boundary
\( x = 0 \) along \( \frac{D^*_1 x}{dt} = -c_0 \) 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 \textit{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 \textit{Hedstrøm} (1979) (cf. Section 7.6).
Figure 7.2: 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.

### 7.3 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 \( x \in \langle 0, L \rangle \) and \( t \in \langle 0, T \rangle \). The boundaries are then at \( x = 0, L \), where we assume that \( x = L \) is an open boundary, while \( x = 0 \) 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. 7.2).

Since (7.16) is an advection equation it is natural that we use one of the stable schemes developed in Section 5.1. 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 7.2, we get

\[
\frac{\phi_{j+1}^{n+1} - \phi_j^n}{\Delta t} + c_\phi \frac{\phi_j^n - \phi_{j-1}^n}{\Delta x} = 0 \tag{7.17}
\]
or

\[ \phi_J^{n+1} = (1 - r_\phi) \phi_J^n + r_\phi \phi_{J-1}^n \]  

(7.18)

where

\[ r_\phi = c_\phi \frac{\Delta t}{\Delta x}. \]  

(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 \( c_\phi \)? As suggested by Orlanski (1976) we might solve (7.12) with respect to the phase velocity (or \( r_\phi \)) and get

\[ r_\phi = \frac{\phi_{J-1}^{n+1} - \phi_{J-1}^n}{\phi_{J-1}^n - \phi_{J-1}^{n-1}}. \]  

(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 \( c_\phi \) (or \( r_\phi \)) 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

\[ r_\phi = \frac{\phi_{J-1}^{n+1} - \phi_{J-1}^n}{\phi_{J-1}^n - \phi_{J-1}^{n-1}}. \]  

(7.21)

A second is to use information at previous times at same points in space,

\[ r_\phi = \frac{\phi_J^n - \phi_{J-1}^{n-1}}{\phi_J^{n-1} - \phi_{J-1}^{n-1}}. \]  

(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 5.7), 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

\[ r_\phi' = \frac{\phi_{J-1}^{n+1} - \phi_{J-1}^{n-1}}{\phi_{J-1}^{n-1} - \phi_{J-1}^{n-2}}. \]  

(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 \( r_\phi \) (no prime attached) as

\[ r_\phi = \begin{cases} 
  r_\phi' ; & 0 \leq r_\phi' \\
  0 ; & r_\phi' < 0 
\end{cases}. \]  

(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 \( r_\phi \) in (7.18) when determining the new
7.4 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,

\[
\begin{align*}
\partial_t u &= -g \partial_x h, \\
\partial_t h &= -H \partial_x u.
\end{align*}
\]  

Let our domain of interest be \( x \in (-L_i, L_i) \). 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 \( x \in (-L_e, L_e) \) 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\(^5\). 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

\[
\begin{align*}
\partial_t u &= -g \partial_x h - \gamma u, \\
\partial_t h &= -H \partial_x u,
\end{align*}
\]

\(^4\)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.

\(^5\)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 initial velocity. Thus Rayleigh friction decreases the kinetic energy of all wavelengths equally.
to be solved within the extended domain \( x \in (-L_e, L_e) \). Furthermore by letting the frictional parameter \( \gamma \) be zero within \( x \in (-L_i, L_i) \) (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 \( x \in (-L_e, -L_i) \) and \( x \in (L_i, L_e) \) we let \( \gamma \) gradually and monotonically increase as we get farther and farther away from the open boundaries. For instance we achieve this by using

\[
\gamma = \gamma_0 \begin{cases} 
1 - e^{-\lambda (L + L_i)} & ; -L_e \leq x < -L_i \\
0 & ; -L_i \leq x \leq L_i \\
1 - e^{\lambda (L - L_i)} & ; L_i < x \leq L_e 
\end{cases},
\]

(7.29)

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 \( \lambda \) and \( \gamma \) determine how fast or quickly the frictional effect increases within the two sponge layers \( x \in (-L_e, -L_i) \) and \( x \in (L_i, L_e) \). 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

\[
\partial^2_t h + \gamma \partial_t h = gH \partial^2_x h.
\]

(7.30)

Searching for wave like solution we let

\[
h = h_0 e^{\omega t} e^{i\alpha x},
\]

(7.31)

where \( \alpha \) is the wavenumber and \( \omega \) is a complex frequency. Substituting this expression into (7.30) we get the dispersion relation

\[
\omega^2 + \gamma \omega + gH \alpha^2 = 0,
\]

(7.32)

which gives the two solutions

\[
\omega_{1,2} = -\frac{1}{2} \gamma \pm i \alpha c
\]

(7.33)

where

\[
c = \sqrt{gH - \left(\frac{\gamma}{2\alpha}\right)^2}.
\]

(7.34)

Thus the solution in terms of the height of a fluid column is hence

\[
h = h_0 e^{-\frac{1}{2} t \gamma} e^{i \omega (x \pm ct)},
\]

(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
7.5 Flow relaxation

\( \gamma \neq 0 \) domain decreases exponentially in time. Furthermore, we notice that phase velocity \((7.34)\) decreases with increasing \(\gamma\). 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

\[
\begin{align*}
\partial_t u &= -g \partial_x h - \gamma u + \tau \\
\partial_t h &= -H \partial_x u
\end{align*}
\]

where \(\tau\) 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 \(\gamma\) 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).

7.5 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\(^6\).

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

\(^6\)cf. http://met.no/
of the interior domain and thus extends the computational domain (cf. Figure 7.3). Within the FRS zone the solution is relaxed towards an a priori specified exterior solution\(^7\). 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 7.3 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 7.4. As usual we define 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 \( \phi^e_{j} \) 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 \( \phi^n_{j} \) 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 \( \phi^*_{j} \). We underscore that \( \phi^*_{j} \) 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 \( \phi^*_{j} \) and the specified outer solution \( \phi^e_{j} \) to derive the final or corrected solution at time level \( n + 1 \). We do this by employing the formula

\[
\phi^{n+1}_{j} = (1 - \alpha_{j})\phi^*_{j} + \alpha_{j}\phi^{e}_{j} \quad ; \quad j = 1(1)J,
\]

\(^7\)Often also referred to as the outer solution
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., $\phi_{j}^{n+1} = \phi_e^{n+1}$.

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 $\alpha$ throughout the FRS zone. They found that distributing $\alpha$ applying a hyperbolic tangent function, that is,

$$\alpha_j = 1 - \tanh \frac{j - 1}{2}; \quad j = 1(1)JM,$$

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

$$\partial_t \phi = \mathcal{L}[\phi]; \quad x \in <L_i,L_i>, \quad (7.40)$$

where $\mathcal{L}$ is a spatial differential operator. Furthermore, let us assume that the open boundary is at $x = -L_i$ 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 $\phi^*_j$; $j = 1(1)J_l$ denote the specified exterior solution. If we solve (7.40) applying a forward in time finite difference scheme we get

$$\phi_j^n = \phi_j^n + \Delta t \mathcal{L}_j^n; \quad j = 2(1)J_r,$$

where $\phi_j^*$ is the predictor. We then correct the predictor by applying the relaxation formula (7.38). We then get

$$\phi_j^{n+1} = (1 - \alpha_j)\phi_j^* + \alpha_j(\phi_e)_j^{n+1}; \quad j = 1(1)J_r.$$  \hspace{1cm} (7.42)

To ensure that we do no corrections to the predictor within the interior domain we let the relaxation parameter be given by

$$\alpha_j = \begin{cases} 
1 - \tanh \left(\frac{j+1}{2}\right), & j = 1(1)J_l \\
0, & j = J_l(1)J_r
\end{cases} \hspace{1cm} (7.43)$$
We may now use the expression on the right-hand side of (7.41) to substitute for \( \phi^* \) in (7.42). If we in addition add the zero \( (\alpha_j \phi_j^{n+1} - \alpha_j \phi_j^n) \) to the left-hand side of (7.42) we get

\[
\frac{\phi_j^{n+1} - \phi_j^n}{\Delta t} = L_j^n + \gamma_j (\phi_{e_j}^{n+1} - \phi_j^{n+1}) ; \quad j = 1(1)J_t
\]

(7.44)

where the coefficient \( \gamma_j \) is defined by,

\[
\gamma_j = \frac{\alpha_j}{1 - \alpha_j}.
\]

(7.45)

If we now let \( \Delta t \) and \( \Delta x \) tend to zero, we notice that (7.44) is a forward in time, finite difference approximation to the continuous equation

\[
\partial_t \phi = L[\phi] + \gamma (\phi_e - \phi) \quad ; \quad x \in < -Le, L_i >.
\]

(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

\[
\partial_t \phi = L[\phi] - \gamma \phi.
\]

(7.47)

Under these circumstances the FRS acts like a sponge with a frictional parameter \( \gamma \) which gradually and monotonically increases towards infinity as we progress into the FRS zone, not unlike the exponential function specified in Section 7.4.

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,

\[
\begin{align*}
\partial_t u &= -g \partial_x h \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \qu
7.6 Weakly reflective OPEN BOUNDARY CONDITIONS

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 $\alpha$. Thus, unless $u^* = u_e$, the volume (or mass) conservation, as expressed by (7.49), is violated. Thus the varying relaxation parameter builds up volume, which in turn builds up an 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 $\alpha$ 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.

7.6 A weakly reflective OBC

As alluded to in Section 7.2 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

$$\begin{align*}
\partial_t u + u \partial_x u - fv &= -g \partial_x h + F^x, \\
\partial_t v + u \partial_x v + fu &= F^y, \\
\partial_t h + \partial_x (hu) &= 0,
\end{align*}$$

(7.54) (7.55) (7.56)

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

$$\frac{D^*_1,2}{dt}(u \pm 2c) = f v + F^x \quad \text{along} \quad \frac{D^*_1,2,x}{dt} = u \pm c.$$  \hspace{1cm} (7.57)

To avoid reflections at, say $x = L$, we require

$$\frac{D^*_2 x}{dt} = 0 \quad \text{at} \quad x = L.$$  \hspace{1cm} (7.58)
OPEN BOUNDARY CONDITIONS 7.6 Weakly reflective

By substitution of this expression in (7.57) the weakly reflective open boundary condition becomes,

\[
\left( \frac{D^*}{dt} \right)_1 (u + 2c) = f_v + F^x, \quad \text{along} \quad \left( \frac{D^*}{dt} \right)_1 = u + c
\]

(7.59)

\[
\partial_t (u - 2c) = f_v + F^x,
\]

(7.60)

valid at \( x = L \). To find \( u \) and \( h \) at the boundary \( x = L \) \((j = J + 1)\) numerically we first make a finite difference approximation to (7.59). Thus we get

\[
u_{j+1}^{n+1} + 2c_{j+1}^{n+1} = u_Q^n + 2c_Q^n + (f_v + F^x)_Q^n \Delta t,
\]

(7.61)

where the position of the point \( x_Q \) \((j = j_Q)\) is found by utilizing the characteristic equation in (7.59), that is,

\[
x_Q = x_j - (u_{j+1}^n + c_{j+1}^n)\Delta t.
\]

(7.62)

Next the a finite difference approximation to (7.60) gives

\[
u_{j+1}^{n+1} - 2c_{j+1}^{n+1} = (f_v + F^x)_{j+1}^n \Delta t.
\]

(7.63)

Solving (7.61) and (7.63) with respect to \( u_{j+1}^{n+1} \) and \( c_{j+1}^{n+1} \) we get

\[
u_{j+1}^{n+1} = \frac{1}{2} \left( u_Q^n + 2c_Q^n \right) + (f_v + F^x)_Q^n \Delta t
\]

(7.64)

\[
c_{j+1}^{n+1} = \frac{1}{4} \left( u_Q^n + 2c_Q^n \right) + \frac{1}{4} \left[ (f_v + F^x)_Q^n - (f_v + F^x)_{j+1}^n \right] \Delta t
\]

(7.65)

It remains to find \( u_Q^n \) and \( c_Q^n \). As explained in Section 5.7 we find these by a two point interpolation of the adjacent grid points or higher order interpolation for instance using Newton’s interpolation formulae. 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 descibed in Section 5.7 6.5 on page 116.

**Exercises**

1. Show that a one-sided, finite difference scheme in time and space of the radiation condition (7.1) can be written

\[
\phi_{B+1}^{n+1} = \begin{cases} 
\phi_B^n & ; \ c_\phi > 0 \\
(1 + c_\phi \Delta t) \phi_B^n - c_\phi \frac{\Delta t}{\Delta x} \phi_{B+1}^n & ; \ c_\phi \leq 0 
\end{cases}
\]

(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.
7.6 Weakly reflective

OPEN BOUNDARY CONDITIONS
Chapter 8

GENERAL VERTICAL COORDINATES

Most modern models employed in the meteorological and oceanographic community replace the natural geopotential vertical coordinate \((z\text{-coordinate})\) 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 11). 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 \(\sigma\)-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 suggested by Shapiro and Hastings (1973) to use surfaces of potential temperature as the vertical coordinate. This was tested in a quasi-geostrophic weather prediction model by Bleck (1973) and later in primitive equation models, so-called isentropic models, by Eliassen and Raustein (1968). 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, as summarized in (Griffies, 2004, Chapter 6) that the various vertical coordinate systems all have their advantages and disadvantages.

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
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

\[ x' = x, \quad y' = y, \quad s = s(x, y, z, t), \quad \text{and} \quad t' = t. \]  

(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

\[ \partial_z s \geq 0, \quad \text{and} \quad \partial_z s \neq 0. \]  

(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

\[ \partial_z x' = \partial_z y' = \partial_z t' = 0, \quad \partial_t x' = \partial_t y' = 0, \quad \partial_y x' = \partial_x y' = 0, \quad \text{and} \quad \partial_x t' = \partial_y t' = 0, \]  

(8.3)

while

\[ \partial_x x' = \partial_y y' = \partial_t t' = 1. \]  

(8.4)

Similarly follows

\[ \partial_s x = \partial_s y = \partial_s t = 0, \quad \partial_{t'} x = \partial_{t'} y = 0, \quad \partial_{y'} x = \partial_{x'} y = 0, \quad \text{and} \quad \partial_{x'} t = \partial_{y'} t = 0, \]  

(8.5)

while

\[ \partial_{s'} x = \partial_{y'} y = \partial_{t'} t = 1. \]  

(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 \(\partial_z s = \partial_z z = 1\).

Let \(\psi = \psi(x, y, z, t) = \psi(x', y', s, t')\) denote any scalar. Then the first property of the transformation is

\[ \partial_z \psi = \partial_x s \partial_s \psi. \]  

(8.7)

If we take the derivative of \(\psi\) with respect to one of the independent variables in the coordinate system we transform to, say \(t'\), then we get

\[ \partial_{t'} \psi = \partial_t \psi \partial_{t'} t + \partial_x \psi \partial_{t'} x + \partial_y \psi \partial_{t'} y + \partial_z \psi \partial_{t'} z = \partial_t \psi + \partial_z s \partial_s \psi \partial_{t'} t. \]  

(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 \( \psi \) in the new coordinate system by
\[
\nabla_s \psi = i \partial_{x'} \psi + j \partial_{y'} \psi. \tag{8.11}
\]
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 \( A \), transforms as
\[
\nabla_H \cdot A = \nabla_s \cdot A - \partial_z s \partial_s A \cdot \nabla_s z. \tag{8.13}
\]
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
\[
\frac{D \psi}{dt} = \partial_t \psi + u \cdot \nabla_H \psi + w \partial_z \psi = \partial_{t'} \psi + u \cdot \nabla_s \psi + \dot{s} \partial_s \psi, \tag{8.14}
\]
where
\[
\dot{s} = \frac{D s}{dt} = \partial_t s + u \cdot \nabla_H s + w \partial_z s \tag{8.15}
\]
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 derivative 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
\[
\frac{D \psi}{dt} = \partial_{t'} \psi + u \cdot \nabla_s \psi + (w - \partial_{t'} z - u \cdot \nabla_s z) \partial_z s \partial_s \psi. \tag{8.16}
\]
Equating this by the individual derivative expressed in the new general vertical coordinate system as visualized in the second equality in (8.14) we get
\[
\dot{s} = (w - \partial_{t'} z - u \cdot \nabla_s z) \partial_z s \equiv \omega \partial_z s, \tag{8.17}
\]
where the identity in (8.17) defines the velocity \( \omega \) by
\[
\omega = w - (\partial_{t'} z + u \cdot \nabla_s z) = w + (\partial_t s + u \cdot \nabla_H s) \partial_z z. \tag{8.18}
\]
\(^1\)Also by many authors referred to as the material derivative
First we observe that if \( s = z \) then \( \omega = w \) as expected. Under these circumstances we must interpret the velocity \( \omega \) as the speed by which the surface \( s \) travels through the fixed \( z \) levels of the geopotential coordinate system. Next we let \( s \) be a material surface. Then the kinematic boundary condition requires \( w = (\partial_t s + \mathbf{u} \cdot \nabla_H s) \), and hence \( \omega = 0 \). This is again 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. Finally, if the \( s \) surface is not a material surface then \( \omega \neq 0 \) is associated with the speed of the fluid particles through the surface \( s \). The difference between \( \omega \) and \( w \) is hence associated with the speed of the surface \( s \) in the geopotential coordinate system. Thus \( \omega \) is simply interpreted as that part of the vertical movement of particles that remains when moving with the surface \( s \).

### 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

\[
\frac{\partial z}{\partial s} p + \rho g = 0. 
\]  
(8.19)

Using the transformation formulas of the previous section we get

\[
\frac{\partial s}{\partial p} + \rho g \frac{\partial z}{\partial s} = 0. 
\]  
(8.20)

We may use this equation to determine the metric factors \( \partial_s z \) and \( \partial_z s \) as follows

\[
\frac{\partial_z z}{\partial s} = -\frac{\partial s}{\partial p}, \quad \text{and} \quad \frac{\partial z}{\partial s} = -\frac{\rho g}{\partial_s p}. 
\]  
(8.21)

We note in passing that if \( s = p \) then (8.20) reduces to

\[
1 + \rho g \frac{\partial_p z}{\partial s} = 0, \quad \text{or} \quad \frac{\partial_p z}{\partial s} = -\frac{1}{\rho g}. 
\]  
(8.22)

#### Mass conservation

Next, we transform the continuity equation

\[
\partial_t \rho + \nabla \cdot (\mathbf{v} \rho) = 0. 
\]  
(8.23)

We first rewrite this equation to yield

\[
\frac{1}{\rho} \frac{\partial \rho}{\partial t} + \nabla_H \cdot \mathbf{u} + \partial_z w = 0. 
\]  
(8.24)
We then make use of the transformation formulas to obtain

\[
\frac{1}{\rho} \frac{D\rho}{dt} + \nabla_H \cdot \mathbf{u} + \partial_z w = - \rho (\partial' \alpha + \mathbf{u} \cdot \nabla_s \alpha + \dot{s} \partial_s \alpha) \\
+ \partial_s \left[ \partial' (\partial_s z) + \nabla_s \cdot (\mathbf{u} \partial_s z) + \partial_s (\dot{s} \partial_s z) \right],
\]

(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

\[
\frac{1}{\rho} \frac{D\rho}{dt} + \nabla_H \cdot \mathbf{u} + \partial_z w = (\partial_s p)^{-1} \left[ \partial' (\partial_s p) + \nabla_s \cdot (\mathbf{u} \partial_s p) + \partial_s (\dot{s} \partial_s p) \right],
\]

(8.26)

and hence the transformed continuity equation reads

\[
\partial' (\partial_s p) + \nabla_s \cdot (\mathbf{u} \partial_s p) + \partial_s (\dot{s} \partial_s p) = 0.
\]

(8.27)

**Energy equation**

If we apply a similar procedure to the tracer equation (1.13) we get

\[
\partial' C + \mathbf{u} \cdot \nabla_s C + \dot{s} \partial_s C = \mathbf{F}_C + S_C
\]

(8.28)

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

\[
\frac{Du}{dt} + f \mathbf{k} \times \mathbf{u} = -\alpha \nabla_H p + \alpha \partial_z \tau + \nabla_H \cdot \mathbf{F}_M^H
\]

(8.29)

where \( \tau \) is the vertical mixing or flux vector, sometimes referred to as 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

\[
\alpha \nabla_H p = \alpha \nabla_s p + g \nabla_s z = \nabla_s M - p \nabla_s \alpha
\]

(8.30)

where

\[
M = \alpha p + gz
\]

(8.31)

is the Montgomery potential (or stream function). In the case \( s = \rho \) the last term in (8.31) 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 (8.31) 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

\[
\alpha \tau = \alpha \partial_z s \partial_s \tau = -g \frac{\partial_s \tau}{\partial_s p} = \partial_p \tau. \tag{8.32}
\]

Recalling that

\[
\frac{D \mathbf{u}}{dt} = \partial_t \mathbf{u} + \mathbf{u} \cdot \nabla_s \mathbf{u} + \dot{s} \partial_s \mathbf{u} \tag{8.33}
\]

and that

\[
\mathbf{u} \cdot \nabla_s \mathbf{u} = \nabla_s \left( \frac{1}{2} \mathbf{u}^2 \right) + \zeta \mathbf{k} \times \mathbf{u} \tag{8.34}
\]

where \( \zeta = k \cdot \nabla_s \times \mathbf{u} \) is the relative vorticity relative to the new coordinate system. Hence the momentum equation becomes

\[
\partial_t \mathbf{u} + \nabla_s \left( \frac{1}{2} \mathbf{u}^2 \right) + (\zeta + f) \mathbf{k} \times \mathbf{u} + \dot{s} \partial_s \mathbf{u} = -\nabla_s M + p \nabla_s \alpha - g \partial_p \tau + \nabla_s \cdot \mathbf{F}_M^H. \tag{8.35}
\]

We may also write this equation in flux form. We then first recombine the second and third term on the left-hand side of (8.35) using (8.34). Next we multiply (8.35) by \( \partial_s p \) and then finally make use of the continuity equation in the form (8.27). We then get

\[
\partial_t (\mathbf{u} \partial_s p) + \nabla_s \cdot (\mathbf{u} \mathbf{u} \partial_s p) + f k \times \mathbf{u} \partial_s p + \partial_s (\dot{s} \mathbf{u} \partial_s p) = -\partial_s p (\nabla_s M + p \nabla_s \alpha) - g \partial_p \tau + \partial_s p \nabla_s \cdot \mathbf{F}_M^H. \tag{8.36}
\]

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 \( \sigma \)-coordinate models. This particular coordinate system is defined by

\[
s = \sigma = \frac{z - \eta}{D} \quad \text{or} \quad z = \sigma D + \eta, \tag{8.37}
\]

where \( D = H + \eta \) is the total depth, \( \eta \) 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

\[
\partial_z s = \partial_z \sigma = \frac{1}{D} \quad \text{and} \quad \partial_s z = \partial_s \sigma = D, \tag{8.38}
\]

which allows us to rewrite the hydrostatic equation to

\[
\partial_s p = -\rho g D. \tag{8.39}
\]

Furthermore we need to know the speed \( \omega \) trough the \( \sigma \) surfaces. Applying (8.37) we get

\[
\omega = w - \sigma \partial_v D - \partial_v \eta - \sigma \mathbf{u} \cdot \nabla_s D - \mathbf{u} \cdot \nabla_s \eta. \tag{8.40}
\]

Thus the mass conservation equation in the form (8.27) becomes

\[
\partial_t' (\rho D) + \nabla_{\sigma} \cdot (\rho D \mathbf{u}) + \partial_\sigma (\dot{\sigma} \rho D) = 0, \tag{8.41}
\]

Separating the effect of the density we get

\[
\frac{D \rho}{dt} + \frac{\rho}{D} [\partial_v D + \nabla_{\sigma} \cdot (D \mathbf{u}) + \partial_\sigma (\dot{\sigma} D)] = 0, \tag{8.42}
\]

We observe using (8.17) that \( \dot{\sigma} = \omega \partial_z \sigma = \omega D^{-1} \). Furthermore we note that \( \partial_v D = \partial_v \eta \). Substitution of these expressions into (8.42), 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

\[
\partial_t' \eta + \nabla_{\sigma} \cdot (D \mathbf{u}) + \partial_\sigma \omega = 0. \tag{8.43}
\]

We note that the remaining equations may be derived from their general expressions in a similar fashion. For instance using (8.39) the momentum equation in the flux form (8.36) becomes

\[
\partial_t' (D \mathbf{u}) + \nabla_s \cdot (D \mathbf{u} \mathbf{u}) + f \mathbf{k} \times D \mathbf{u} + \partial_\sigma (\omega \mathbf{u}) \\
= -D (\nabla_s M + p \nabla_s \alpha) - \frac{1}{\rho_0} \partial_\sigma \tau + D \nabla_s \cdot \mathbf{F}_M. \tag{8.44}
\]
8.3 Terrain-following

GENERAL VERTICAL COORDINATES
Chapter 9

TWO-DIMENSIONAL PROBLEMS

Below we investigate the effect of including more than one-dimension in space on the numerical stability criterion. In particular we study the effect on the finite difference equations relating to the diffusion, advection and shallow water problems. 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

$$\frac{\partial}{\partial t} \theta = \kappa (\frac{\partial^2 \theta}{\partial x^2} + \frac{\partial^2 \theta}{\partial y^2}).$$

(9.1)

Note that since we expand to two dimensions we use the notation $\theta(x_j, y_k, t^n) = \theta^n_{jk}$ as outlined in Section 2.9 page 26. 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

$$\left[\frac{\partial^2 \theta}{\partial x^2}\right]_{jk} = \frac{\theta^n_{j+1,k} - 2\theta^n_{jk} + \theta^n_{j-1,k}}{\Delta x^2},$$

and

$$\left[\frac{\partial^2 \theta}{\partial y^2}\right]_{jk} = \frac{\theta^n_{jk+1} - 2\theta^n_{jk} + \theta^n_{jk-1}}{\Delta y^2},$$

(9.2)

while we replace the first order derivative with respect to time with

$$\left[\frac{\partial \theta}{\partial t}\right]_{jk} = \frac{\theta^{n+1}_{jk} - \theta^n_{jk}}{\Delta t},$$

(9.3)

where $\Delta x, \Delta y$ are the space increments along the $x, y$ axis, respectively, and $\Delta t$ is the time step. Thus we arrive at the scheme

$$\theta^{n+1}_{jk} = \theta^n_{jk} + \kappa \frac{\Delta t}{\Delta x^2} (\theta^n_{j+1,k} - 2\theta^n_{jk} + \theta^n_{j-1,k}) + \kappa \frac{\Delta t}{\Delta y^2} (\theta^n_{jk+1} - 2\theta^n_{jk} + \theta^n_{jk-1}).$$

(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 $\theta^n_{jk}$ 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,

\[
\theta^n_{jk} = \Theta_n e^{\alpha_j \Delta x} e^{\beta_k \Delta y},
\]

where \( \alpha \) and \( \beta \) 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

\[
G = 1 - 2\kappa \frac{\Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) - 2\kappa \frac{\Delta t}{\Delta y^2} (1 - \cos \beta \Delta y).
\]

We observe that this expression is comparable to the one derived for the one-dimensional case, that is (4.31) on page 47, except that we have an additional term due to the two-dimensionality of (9.1). Applying von Neumann’s criterion for numerical stability (4.28) we require \(|G| \leq 1\). Hence

\[
-1 \leq 1 - 2\kappa \frac{\Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) - 2\kappa \frac{\Delta t}{\Delta y^2} (1 - \cos \beta \Delta y) \leq 1.
\]

As in the one-dimensional case we note that the right-hand inequality is trivially satisfied. To satisfy the left-hand inequality we require

\[
\kappa \frac{\Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) + \kappa \frac{\Delta t}{\Delta y^2} (1 - \cos \beta \Delta y) \leq 1.
\]

Since the left-hand side of (9.8) is maximum when \( \cos \alpha \Delta x = \cos \beta \Delta y = -1 \) we note that if

\[
\kappa \frac{\Delta t}{\Delta x^2} + \kappa \frac{\Delta t}{\Delta y^2} \leq \frac{1}{2}
\]

then the stability criterion is satisfied for all possible values of the wavenumbers \( \alpha \) and \( \beta \). Thus the time step limit is

\[
\Delta t \leq \frac{1}{2\kappa \Delta x^2 + \Delta y^2}.
\]

In the special case when the grid is regular (a square grid), that is, \( \Delta x = \Delta y = \Delta s \), we get

\[
\Delta t \leq \frac{\Delta s^2}{4\kappa}.
\]

If we compare (9.11) with the one dimensional case (eq. 4.34 on page 48) 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

\[
\partial_t \theta + u_0 \partial_x \theta + v_0 \partial_y \theta = 0,
\]

(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 first order derivatives with their respective finite difference approximations as outlined in Section 2.7. Thus we get

\[
\frac{\theta^{n+1}_{jk} - \theta^{n-1}_{jk}}{2\Delta t} + u_0 \frac{\theta^n_{j+1k} - \theta^n_{j-1k}}{2\Delta x} + v_0 \frac{\theta^n_{jk+1} - \theta^n_{jk-1}}{2\Delta y} = 0. \tag{9.13}
\]

or

\[
\theta^{n+1}_{jk} = \theta^{n-1}_{jk} + u_0 \frac{\Delta t}{\Delta x} \left( \theta^n_{j+1k} - \theta^n_{j-1k} \right) + v_0 \frac{\Delta t}{\Delta y} \left( \theta^n_{jk+1} - \theta^n_{jk-1} \right). \tag{9.14}
\]

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 \( \lambda \). 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 \( \alpha \) and \( \beta \) 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 here in scalar form. Thus in two dimensions we get

\begin{align}
\frac{\partial}{\partial t} u + \bar{u} \frac{\partial}{\partial x} u + \bar{v} \frac{\partial}{\partial y} u - f v + \partial_x \phi &= 0, \\
\frac{\partial}{\partial t} v + \bar{u} \frac{\partial}{\partial x} v + \bar{v} \frac{\partial}{\partial y} v + f u + \partial_y \phi &= 0, \\
\frac{\partial}{\partial t} \phi + \bar{\phi} \left( \partial_x u + \partial_y v \right) &= 0.
\end{align}

(9.22) \quad (9.23) \quad (9.24)

To analyze the possible wave motions we assume that all the variables are two dimensional waves of frequency \( \omega \). Thus we assume that the solution is

\[ x = x_0 e^{-i\omega t} e^{i(\alpha x + \beta y)}, \]

(9.25)

where \( \alpha \) and \( \beta \) 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 \( x \), that is,

\[ x = \begin{bmatrix} u \\ v \\ \phi \end{bmatrix} \] and \( x_0 = \begin{bmatrix} u_0 \\ v_0 \\ \phi_0 \end{bmatrix}, \]

(9.26)

where \( x_0 \) is the amplitude.

Inserting (9.25) into the linearized equations (9.22) - (9.24) we get

\begin{align}
i(\alpha \bar{u} + \beta \bar{v} - \omega)u - f v + i \alpha \phi &= 0, \\
f u + i(\alpha \bar{u} + \beta \bar{v} - \omega) v + i \beta \phi &= 0, \\
i \bar{\phi} \alpha u + i \bar{\phi} \beta v + i(\alpha \bar{u} + \beta \bar{v} - \omega) \phi &= 0,
\end{align}

(9.27) \quad (9.28) \quad (9.29)

which in turn may be formulated as the homogeneous linear equation,

\[ \mathcal{A} \cdot x = 0, \]

(9.30)

where the tensor \( \mathcal{A} \) is

\[ \mathcal{A} = \begin{bmatrix} i(\alpha \bar{u} + \beta \bar{v} - \omega) & -f & i \alpha \\ f & i(\alpha \bar{u} + \beta \bar{v} - \omega) & i \beta \\ i \bar{\phi} \alpha & \bar{\phi} \beta & i(\alpha \bar{u} + \beta \bar{v} - \omega) \end{bmatrix}. \]

(9.31)

For non-trivial solutions to exists, the determinant of the tensor \( \mathcal{A} \) must be zero, which gives

\[ i(\alpha \bar{u} + \beta \bar{v} - \omega) \left[ i(\alpha \bar{u} + \beta \bar{v} - \omega)^2 - \bar{\phi}(\alpha^2 + \beta^2) + f^2 \right] = 0. \]

(9.32)
As in the one-dimensional case we find that we get three solutions for the frequency $\omega$, namely

\[ \omega_1 = \bar{u}\alpha + \bar{v}\beta, \]  
(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)

where

\[ c_0 = \sqrt{\bar{\phi}} = \sqrt{gH} \]  
(9.36)

is the wave phase speed. The first solution is simply the geostrophic balance as displayed in (1.40) on page 11 with $\phi = gh$, that is,

\[ u = \frac{1}{f}k \times \nabla_H \phi. \]  
(9.37)

We easily derive this interpretation by substituting $\omega_1$ from (9.33) into (9.27) and (9.28), respectively, that is,

\[ 0 - fv + i\alpha\phi = 0, \]  
(9.38)

\[ fu + 0 + i\alpha\phi = 0, \]  
(9.39)

which gives

\[ u = -\frac{1}{f}i\beta\phi \quad \text{and} \quad v = \frac{1}{f}i\alpha\phi \Rightarrow u = -\frac{1}{f}\partial_y\phi \quad \text{and} \quad v = \frac{1}{f}\partial_x\phi. \]  
(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 $\pm \sqrt{\phi(\alpha^2 + \beta^2) + f^2}$ are combined inertia and gravity waves. The inertia part is associated with frequencies $\omega$ proportional to $f$, so called inertial oscillation, while gravity waves are associated with frequencies $\omega = \sqrt{c_0^2(\alpha^2 + \beta^2)}$. Recalling that the dispersion relation for gravity waves is $\omega = c_0\sqrt{\alpha^2 + \beta^2}$ which shows that $c_0$ is the phase speed of gravity waves.

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,

\[ x = \sum_{\alpha = -\infty}^{\infty} \sum_{\beta = -\infty}^{\infty} x_0(\alpha, \beta)e^{i(\alpha x + \beta y - \omega t)}, \]  
(9.41)

where we observe that $x_0(\alpha, \beta)$ 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

\[
\begin{align*}
\frac{u_{jk}^{n+1} - u_{jk}^{n-1}}{2\Delta t} + \frac{u_{j+1k}^{n} - u_{jk}^{n}}{2\Delta x} + \frac{u_{jk}^{n-1} - u_{jk}^{n-1}}{2\Delta y} - f v_{jk}^{n} &= -\frac{\phi_{jk+1}^{n} - \phi_{jk-1}^{n}}{2\Delta x}, & (9.42) \\
\frac{v_{jk}^{n+1} - v_{jk}^{n-1}}{2\Delta t} + \frac{v_{j+1k}^{n} - v_{jk}^{n}}{2\Delta x} + \frac{v_{jk}^{n-1} - v_{jk}^{n-1}}{2\Delta y} + f u_{jk}^{n} &= -\frac{\phi_{jk+1}^{n} - \phi_{jk-1}^{n}}{2\Delta y}, & (9.43) \\
\frac{\phi_{jk}^{n+1} - \phi_{jk}^{n-1}}{2\Delta t} + c_0^2 \frac{u_{j+1k}^{n} - u_{jk}^{n-1}}{2\Delta x} + c_0^2 \frac{v_{jk+1}^{n} - v_{jk}^{n-1}}{2\Delta y} &= 0. & (9.44)
\end{align*}
\]

To investigate the numerical stability of the two-dimensional shallow water problem 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

\[
\begin{align*}
\frac{u_{jk}^{n+1} - u_{jk}^{n-1}}{2\Delta t} - f v_{jk}^{n} + \frac{\phi_{jk+1}^{n} - \phi_{jk-1}^{n}}{2\Delta x} &= 0, & (9.45) \\
\frac{v_{jk}^{n+1} - v_{jk}^{n-1}}{2\Delta t} + f u_{jk}^{n} + \frac{\phi_{jk+1}^{n} - \phi_{jk-1}^{n}}{2\Delta y} &= 0, & (9.46) \\
\frac{\phi_{jk}^{n+1} - \phi_{jk}^{n-1}}{2\Delta t} + c_0^2 \frac{u_{j+1k}^{n} - u_{jk}^{n-1}}{2\Delta x} + c_0^2 \frac{v_{jk+1}^{n} - v_{jk}^{n-1}}{2\Delta y} &= 0. & (9.47)
\end{align*}
\]

The question then arise whether the stability condition changes, 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

\[
x_{jk}^{n} = X_n e^{i(\alpha j \Delta x + \beta k \Delta y)}
\]

where the transpose of the vector \( x_{jk}^{n} \) is \( x_{jk}^{n T} = [u_{jk}^{n}, v_{jk}^{n}, \phi_{jk}^{n}] \) and the transpose of the vector \( X_n \) is \( X_n T = [U_n, V_n, \Phi_n] \). Insertion into (9.45) - (9.47) then gives

\[
\begin{align*}
U_{n+1} - U_{n-1} &= 2f \Delta t V_n - 2i\Phi_n \frac{\Delta t}{\Delta x} \sin \alpha \Delta x, & (9.49) \\
V_{n+1} - V_{n-1} &= -2f \Delta t U_n - 2i\Phi_n \frac{\Delta t}{\Delta y} \sin \beta \Delta y, & (9.50) \\
\Phi_{n+1} - \Phi_{n-1} &= -2ic_0^2 U_n \frac{\Delta t}{\Delta x} \sin \alpha \Delta x - 2ic_0^2 V_n \frac{\Delta t}{\Delta y} \sin \beta \Delta y. & (9.51)
\end{align*}
\]

To find an equation for the growth factor we first 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

\[
U_{n+2} - 2U_n + U_{n+2} = 2f \Delta t (V_{n+1} - V_{n-1}) - 2i(\Phi_{n+1} - \Phi_{n-1}) \frac{\Delta t}{\Delta x} \sin \alpha \Delta x, \tag{9.52}
\]
\[ V_{n+2} - 2V_n + V_{n+2} = -2f \Delta t(U_{n+1} - U_{n-1}) \]
\[ -2i(\Phi_{n+1} - \Phi_{n-1}) \frac{\Delta t}{\Delta y} \sin \beta \Delta y. \]  

(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

\[ U_{n+2} - 2(1 - 2f^2 \Delta t^2)U_n + U_{n+2} = -4if \frac{\Delta t^2}{\Delta y} \sin \beta \Delta y \]
\[ -2i(\Phi_{n+1} - \Phi_{n-1}) \frac{\Delta t}{\Delta y} \sin \alpha \Delta x, \]

(9.54)

\[ V_{n+2} - 2(1 - 2f^2 \Delta t^2)V_n + V_{n+2} = 4if \frac{\Delta t^2}{\Delta x} \sin \alpha \Delta x \]
\[ -2i(\Phi_{n+1} - \Phi_{n-1}) \frac{\Delta t}{\Delta y} \sin \beta \Delta y. \]

(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

\[ \Phi_{n+3} - (1 + 2\lambda)\Phi_{n+1} + (1 + 2\lambda)\Phi_{n-1} - \Phi_{n-3} = 0 \]

(9.57)

where

\[ \lambda = 1 - 2f^2 \Delta t^2 - 2 \left( c_0 \frac{\Delta t}{\Delta x} \right)^2 \sin^2 \alpha \Delta x - 2 \left( c_0 \frac{\Delta t}{\Delta y} \right)^2 \sin^2 \beta \Delta y. \]  

(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,

\[ G^3 - (1 + 2\lambda)G^2 + (1 + 2\lambda)G - 1 = 0. \]  

(9.59)

One solution is \( G_1 = 1 \), while the two other solutions are solutions to the second order equation

\[ G^2 - 2\lambda G + 1 = 0. \]  

(9.60)

Thus

\[ G_{2,3} = \lambda \pm i\sqrt{1 - \lambda^2}. \]  

(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. The condition for this to be true is hence that

\[ |\lambda| \leq 1 \]  

(9.62)
or

\[-1 \leq 1 - 2 f^2 \Delta t^2 - 2 \left( c_0 \frac{\Delta t}{\Delta x} \right)^2 \sin^2 \alpha \Delta x - 2 \left( c_0 \frac{\Delta t}{\Delta y} \right)^2 \sin^2 \beta \Delta y \leq 1. \tag{9.63} \]

The right-hand inequality is no problem. 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 radius. Worst case scenario happens if both \( \sin \alpha \Delta x = \sin \beta \Delta y = 1 \). Hence if

\[
\Delta t \leq \frac{\Delta x}{c_0 \sqrt{1 + \left( \frac{\Delta x}{\Delta y} \right)^2 + \left( \frac{\Delta x}{L_R} \right)^2}} \tag{9.66} \]

then (9.65) is satisfied for all possible wave numbers \( \alpha \) and \( \beta \). In practice the grid size must be small enough to resolve the the Rossby radius. Hence \( \left( \frac{\Delta x}{L_R} \right) \ll 1 \) and the last term in the radicand 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

\[
\Delta t < \frac{\Delta s}{c_0 \sqrt{2}}. \tag{9.68} \]

We are now in a position to compare (9.68) with the similar conditions for the one-dimensional case as displayed in (6.52). We observe, as for the diffusion and advection problem, that including more than one dimension in space results in a more stringent condition because of the appearance of the factor \( \sqrt{2} \) in the denominator. This result therefore appears to be general and 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 accurate schemes? Is it straightforward? 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 21) we note that

\[
\theta^n_{j\pm 1} = \theta^n_j \pm \partial_x \theta^n_j \Delta x + \frac{1}{2} \partial^2_x \theta^n_j \Delta x^2 \pm \frac{1}{6} \partial^3_x \theta^n_j \Delta x^3 + \mathcal{O}(\Delta x^4). \tag{10.2}
\]

Subtracting and solving with respect to \( \partial_x \theta^n_j \) we get

\[
\partial_x \theta^n_j = \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} - \frac{1}{6} \partial^3_x \theta^n_j \Delta x^2 + \mathcal{O}(\Delta x^4), \tag{10.3}
\]

Previously we truncated (10.3) so that the FDA of the first order derivative in space became

\[
[\partial_x \theta]^n_j = \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x}, \tag{10.4}
\]

157
which then has an accuracy of $\mathcal{O}(\Delta x^2)$. Note that the first term we neglect is the term $\frac{1}{6}\partial_x^3\theta_j^n \Delta x^2$. We also notice 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 the Taylor series becomes

$$
\theta_{j+2}^n = \theta_j^n + 2\partial_x \theta_j^n \Delta x + 2\partial_x^2 \theta_j^n \Delta x^2 + \frac{4}{3}\partial_x^3 \theta_j^n \Delta x^3 + \mathcal{O}(\Delta x^4),
$$

which is the same as (10.3) only that $\Delta x$ is replaced by $2\Delta x$. Hence subtracting and solving with respect to $\partial_x \theta_j^n$ we get

$$
\partial_x \theta_j^n = \frac{\theta_{j+2}^n - \theta_{j-2}^n}{4\Delta x} - \frac{2}{3}\partial_x^3 \theta_j^n \Delta x^2 + \mathcal{O}(\Delta x^4).
$$

Note that (10.6) is an equally valid expansion from which a centered, second order FDA may be constructed. All we have to do is to neglect all terms $\mathcal{O}(\Delta x^2)$ and higher in respectively (10.3) and (10.6). In the limit $\Delta x \to 0$ they both tend to $\partial_x \theta$, that is, they are both numerically consistent FDAs. Let us now combine (10.3) and (10.6) linearly while retaining the terms $\mathcal{O}(\Delta x^2)$. We do this by first multiplying (10.3) by a coefficient $a$ and (10.6) by a coefficient $b$ and adding them to get

$$
a \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} + b \frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x} = (a + b)\partial_x \theta_j^n + \frac{1}{6}(a + 4b)\partial_x^3 \theta_j^n \Delta x^2 + \mathcal{O}(\Delta x^4).
$$

The coefficients $a$ and $b$ are then linear weights yet to be found. Solving (10.7) with respect to $(a + b)\partial_x \theta_j^n$ we first get

$$
(a + b)\partial_x \theta_j^n = a \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} + b \frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x} - \frac{1}{6}(a + 4b)\partial_x^3 \theta_j^n \Delta x^2 + \mathcal{O}(\Delta x^4)
$$

If we require $a + 4b = 0$ then the second order term actually vanishes. Furthermore we additionally require $a + b = 1$, which gives $a = \frac{4}{3}$ and $b = -\frac{1}{3}$. We then finally get

$$
\partial_x \theta_j^n = \frac{4}{3} \frac{\theta^n_{j+1} - \theta^n_{j-1}}{2\Delta x} - \frac{1}{3} \frac{\theta^n_{j+2} - \theta^n_{j-2}}{4\Delta x} + \mathcal{O}(\Delta x^4),
$$

which is a fourth order accurate FDA of the first spatial derivative in (10.1). Consequently a centered fourth order in space and second order in time scheme for the advection equation (10.1) is

$$
\frac{\theta_{j+1}^n - \theta_j^{n-1}}{2\Delta t} + u \left\{ \frac{4}{3} \frac{\theta_{j+1}^n - \theta_{j-1}^{n-1}}{2\Delta x} - \frac{1}{3} \frac{\theta_{j+2}^n - \theta_{j-2}^{n-1}}{4\Delta x} \right\} = 0.
$$

or

$$
\theta_{j+1}^n = \theta_j^n - \frac{4}{3} \frac{\Delta t}{\Delta x} \left[ \frac{\theta_{j+1}^n - \theta_{j-1}^{n-1}}{2} \left( \theta_{j+2}^n - \theta_{j-2}^{n-1} \right) \right].
$$

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 $\theta_n^j = \Theta_n e^{\alpha_j \Delta x}$ we first get

$$\theta_{j+1}^n - \theta_{j-1}^n = 2i\Theta_n e^{\alpha_j \Delta x} \sin \alpha \Delta x$$  \hspace{1cm} (10.12)

and

$$\theta_{j+2}^n - \theta_{j-2}^n = 2i\Theta_n e^{\alpha_j \Delta x} \sin 2\alpha \Delta x = 4i\Theta_n e^{\alpha_j \Delta x} \sin \alpha \Delta x \cos \alpha \Delta x.$$  \hspace{1cm} (10.13)

The equation for the growth factor $G = \Theta_{n+1}/\Theta_n$ becomes

$$G^2 + 2i\lambda G - 1 = 0 \hspace{1cm} \lambda = \frac{u \Delta t}{3 \Delta x} \sin \alpha \Delta x (4 - \cos \alpha \Delta x),$$  \hspace{1cm} (10.14)

and hence that

$$G_{1,2} = -i\lambda \pm \sqrt{1 - \lambda^2}.$$  \hspace{1cm} (10.15)

As expected the CTCS schemes for the fourth order scheme returns a growth factor whose absolute value does equal one even though its expression is slightly more complex than for the second order scheme. Thus the fourth order scheme is neutrally stable under the condition that the radical in (10.15) is a positive definite quantity, that is,

$$\frac{1}{3} C |\sin \alpha \Delta x| (4 - \cos \alpha \Delta x) \leq 1$$  \hspace{1cm} (10.16)

where $C = |u| \Delta t/\Delta x$ as before is the Courant number. Since the maximum value of $(4 - \cos \alpha \Delta x)$ is five and the maximum value of $|\sin \alpha \Delta x|$ is one, we finally get

$$C \leq \frac{3}{5} \hspace{1cm} \text{or} \hspace{1cm} \Delta t \leq \frac{3\Delta x}{5|u|},$$  \hspace{1cm} (10.17)

which indeed is more stringent than the condition $C \leq 1,$ or $\Delta t \leq \Delta x/|u|,$ that we obtained for the CTCS or second order, leapfrog scheme.

We then may ask whether the numerical dispersion property is affected. To this end we follow the procedure given in Section 5.11 on page 79. Thus we start by decomposing $\theta_j^n$ into its Fourier components in time and space,

$$\theta_j^n = \Theta_0 e^{i\alpha(j \Delta x - cn \Delta t)}.$$  \hspace{1cm} (10.18)

We then substitute (10.18) into (10.10) and solve with respect to the phase speed $c$ to get

$$c = \frac{1}{\alpha \Delta t} \arcsin \left\{ u \alpha \Delta t \left[ \frac{4}{3} \left( \sin \frac{\alpha \Delta x}{\alpha \Delta x} \right) - \frac{1}{3} \left( \frac{\sin 2 \alpha \Delta x}{2 \alpha \Delta x} \right) \right] \right\}.$$  \hspace{1cm} (10.19)

To leading order in $\alpha \Delta x$ we then obtain\(^1\)

$$c \approx u \left\{ 1 - \frac{4}{5!} (\alpha \Delta x)^4 + \cdots \right\}.$$  \hspace{1cm} (10.20)

\(^1\)Note that $\sin z/z = 1 - z^2/6 + \cdots$ while $\arcsin z = 1 + z^2/6 + \cdots$ for $|z| < 1.$
Recalling (see Section 5.11 on page 79) that the phase speed for the second order CTCS scheme was

\[ c_{2nd} = \frac{1}{\alpha \Delta t} \arcsin \left[ u \alpha \Delta t \left( \frac{\sin \alpha \Delta x}{\alpha \Delta x} \right) \right] \approx u \left\{ 1 - \frac{1}{3!}(\alpha \Delta x)^2 + \cdots \right\} , \tag{10.21} \]

we observe that (remember that \( 0 \leq \alpha \Delta x \leq \pi \)) the fourth order scheme is nearly always superior to the second order scheme.

Finally the question arises about the unphysical mode. To investigate this we first note that the growth factor as given by (10.15) has exactly the same two solutions as in (5.25) on page 68, only that the expression for \( \lambda \) has changed to that listed in (10.14). Following the procedure outlined in Section 5.12 we in fact get exactly the same result, that is,

\[ G_1 = e^{-i\alpha c \Delta t} \quad \text{and} \quad G_2 = (-1)^i e^{i\alpha c \Delta t} \quad \tag{10.22} \]

where \( c = c(\alpha) \) now is the dispersive phase speed defined in (10.19).

This process of constructing higher order finite difference approximations may be continued. For example we note that the scheme

\[ \frac{\theta_{j}^{n+1} - \theta_{j}^{n-1}}{2\Delta t} + u \left\{ \frac{3}{2} \frac{\theta_{j+1}^{n} - \theta_{j-1}^{n}}{2\Delta x} - \frac{3}{5} \frac{\theta_{j+2}^{n} - \theta_{j-2}^{n}}{4\Delta x} + \frac{1}{10} \frac{\theta_{j+3}^{n} - \theta_{j-3}^{n}}{4\Delta x} \right\} = 0. \quad \tag{10.23} \]

is good to \( O(\Delta x^6) \). The dispersion relation becomes

\[ c = \frac{1}{\alpha \Delta t} \arcsin \left\{ u \alpha \Delta t \left[ \frac{4}{3} \frac{\sin \alpha \Delta x}{\alpha \Delta x} - \frac{3}{5} \frac{\sin 2\alpha \Delta x}{2\alpha \Delta x} + \frac{1}{10} \frac{\sin 3\alpha \Delta x}{3\alpha \Delta x} \right] \right\} . \quad \tag{10.24} \]

which to leading order gives

\[ c \approx u \left\{ 1 - \frac{36}{7!}(\alpha \Delta x)^6 + \cdots \right\} . \quad \tag{10.25} \]

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 \( \theta_{j-1}^{n} \), assuming a positive advection velocity, gives

\[ \partial_x \theta_j^n = \frac{\theta_{j}^{n} - \theta_{j-1}^{n}}{\Delta x} - \frac{1}{2} \partial_x^2 \theta_j^n \Delta x + O(\Delta x^2) . \quad \tag{10.26} \]

Similarly expanding \( \theta_{j-2}^{n} \) using Taylor series we get

\[ \partial_x \theta_j^n = \frac{\theta_{j}^{n} - \theta_{j-2}^{n}}{2\Delta x} - \partial_x^2 \theta_j^n \Delta x + O(\Delta x^2) . \quad \tag{10.27} \]
Multiplying (10.26) by $a$ and (10.27) by $b$ and adding we get
\[(a + b)\partial_x \theta^n_j = a\frac{\theta^n_j - \theta^n_{j-1}}{\Delta x} + b\frac{\theta^n_j - \theta^n_{j-2}}{2\Delta x} + \frac{1}{2}(a + 2b)\partial_x^2 \theta^n_j \Delta x + O(\Delta x^2). \tag{10.28}\]
Thus by choosing $a + b = 1$ and $a + 2b = 0$, that is, $a = 2$ and $b = -1$, we get
\[\partial_x \theta^n_j = \frac{1}{2\Delta x}(3\theta^n_j - 4\theta^n_{j-1} + \theta^n_{j-2}) + O(\Delta x^2), \tag{10.29}\]
and hence that a second order in space first order in time upwind scheme reads
\[\theta^{n+1}_j = \theta^n_j + u\frac{\Delta t}{2\Delta x}(3\theta^n_j - 4\theta^n_{j-1} + \theta^n_{j-2}). \tag{10.30}\]

We may also construct a 3rd order upwind scheme by noting that a Taylor expansion of $\theta^n_{j+1}$, $\theta^n_{j-1}$ and $\theta^n_{j-2}$ to fourth order gives
\[
\partial_x \theta^n_j = \frac{\theta^n_{j+1} - \theta^n_j}{\Delta x} - \frac{1}{2}\partial_x^2 \theta^n_j \Delta x - \frac{1}{6}\partial_x^3 \theta^n_j \Delta x^2 + O(\Delta x^3), \tag{10.31}
\]
\[
\partial_x \theta^n_j = \frac{\theta^n_j - \theta^n_{j-1}}{\Delta x} + \frac{1}{2}\partial_x^2 \theta^n_j \Delta x - \frac{1}{6}\partial_x^3 \theta^n_j \Delta x^2 + O(\Delta x^3), \tag{10.32}
\]
\[
\partial_x \theta^n_j = \frac{\theta^n_j - \theta^n_{j-2}}{2\Delta x} + \frac{1}{2}\partial_x^2 \theta^n_j \Delta x - \frac{4}{6}\partial_x^3 \theta^n_j \Delta x^2 + O(\Delta x^3), \tag{10.33}
\]
respectively. 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
\[\partial_x \theta^n_j = \frac{1}{6\Delta x}(2\theta^n_{j+1} + 3\theta^n_j - 6\theta^n_{j-1} + \theta^n_{j-2}) + O(\Delta x^3), \tag{10.34}\]
and finally that a 3rd order upwind scheme for the advection equation reads
\[\theta^{n+1}_j = \theta^n_j + u\frac{\Delta t}{6\Delta x}(2\theta^n_{j+1} + 3\theta^n_j - 6\theta^n_{j-1} + \theta^n_{j-2}). \tag{10.35}\]
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 $\theta$ 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 (leapfrog 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 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 35). Moreover, we simplify the problem by assuming that the problem is one-dimensional in space and that the velocity is constant, that is, \( v = u_0 \). Thus the advection-diffusion equation becomes

\[
\frac{\partial}{\partial t} \theta + u_0 \frac{\partial}{\partial x} \theta = \kappa \frac{\partial^2}{\partial x^2} \theta. \tag{10.36}
\]

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 approximation

\[
\theta_j^{n+1} = \theta_j^{n-1} - \frac{u_0 \Delta t}{\Delta x} (\theta_{j+1}^n - \theta_{j-1}^n) + \frac{2\kappa \Delta t}{\Delta x^2} (\theta_{j+1}^{n-1} - 2\theta_j^{n-1} + \theta_{j-1}^{n-1}). \tag{10.37}
\]

We take note 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 \( \Delta t \). Hence each part is stable in itself. If \( \kappa = 0 \) then the advective part is stable if the Courant number \( C \equiv |u_0| \frac{\Delta t}{\Delta x} \leq 1 \). If \( u_0 = 0 \) the diffusive part is stable under the condition \( \kappa \frac{\Delta t}{\Delta x^2} \leq \frac{1}{4} \). The factor \( \frac{1}{4} \) arises because of the \( 2 \Delta t \) time step used for the diffusive part, and replaces the \( \frac{1}{2} \) factor in (4.34) on page 48.

In the general case with \( u_0 \neq 0 \) and \( \kappa \neq 0 \) we get a modified condition. We again use von Neumann’s method and substitute the single 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

\[
\lambda = \frac{u_0 \Delta t}{\Delta x} \sin \alpha \Delta x \quad \text{and} \quad \lambda_2 = 1 - 4 \frac{\kappa \Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x) \tag{10.39}
\]

are two real numbers. 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

\[
|G| = \sqrt{GG^*} = \sqrt{\lambda^2 + \lambda_2 - \lambda^2} = \sqrt{\lambda_2}. \tag{10.41}
\]
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 obtain a more stringent stability condition to the advection equation when we are adding diffusion.

The solution is thus conditionally stable because $0 \leq \lambda_2 \leq 1$. Moreover the condition $\lambda^2 \leq \lambda_2$ gives

$$C^2 \sin^2 \alpha \Delta x \leq 1 - 4K(1 - \cos \alpha \Delta x)$$  \hspace{1cm} (10.42)

where $C = |u_0| \Delta t / \Delta x$ is the Courant number and $K = \kappa \Delta t / \Delta x^2$. The condition (10.42) may be rewritten to give

$$(C^2 + 4K) \sin^2 \alpha \Delta x = 1 + 4K \cos \alpha \Delta x (1 - \cos \alpha \Delta x)$$  \hspace{1cm} (10.43)

We therefore conclude that the sufficient condition for stability of the combined advection-diffusion scheme (10.37) is

$$C^2 + 4K \leq 1, \text{ or } \frac{(u_0 \Delta t)^2 + 4\kappa \Delta t}{\Delta x^2} \leq 1.$$  \hspace{1cm} (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

$$K \ll C^2.$$  \hspace{1cm} (10.45)
We mentioned earlier (cf. Section 4.6) 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 4.6 on page 52) to approximate the diffusion term, even though it is inconsistent. This is fine as long as the remaining terms in our governing equations are treated by consistent schemes. Thus we proceed by making the following FDA of (10.36),

$$\theta_{j}^{n+1} = \theta_{j}^{n-1} - \frac{u_{0}\Delta t}{\Delta x} (\theta_{j+1}^{n} - \theta_{j-1}^{n}) + 2K (\theta_{j+1}^{n} - \theta_{j-1}^{n}) - 2 \theta_{j}^{n-1} + \theta_{j}^{n},$$  \hspace{1cm} (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

$$(1 + 2K)G^2 - 2\lambda G - (1 - 2K) = 0$$ \hspace{1cm} (10.47)

where

$$\lambda = 2K \cos \alpha \Delta x - i \frac{u_{0}\Delta t}{\Delta x} \sin \alpha \Delta x$$ \hspace{1cm} (10.48)

and thus the growth factor has two solutions given by

$$G_{1,2} = \frac{1}{1 + 2K} \left( \lambda \pm \sqrt{4K^2 + \lambda^2 - 1} \right)$$ \hspace{1cm} (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

$$\theta_{j}^{n+1} = \theta_{j}^{n} - \frac{u_{0}\Delta t}{2\Delta x} (\theta_{j+1}^{n} - \theta_{j-1}^{n}) + \frac{\kappa \Delta t}{\Delta x^2} (\theta_{j+1}^{n} - 2\theta_{j}^{n} + \theta_{j-1}^{n}).$$ \hspace{1cm} (10.50)

The amplification or growth factor then follows the equation

$$G = 1 - i \frac{u_{0}\Delta t}{2\Delta x} \sin \alpha \Delta x - 2 \frac{\kappa \Delta t}{\Delta x^2} (1 - \cos \alpha \Delta x).$$ \hspace{1cm} (10.51)

As shown by Clancy (1981) the scheme is stable provided the two conditions

$$\frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{2}, \text{ and } \frac{|u_{0}|\Delta t}{\kappa} \leq 1$$ \hspace{1cm} (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 3.2 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 (5.1), 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 (5.1). 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 $\Delta x$ 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

$$\Psi(x) = a_0 + \sum_{m=1}^{\infty} a_m \sin(\alpha_m x) + b_m \cos(\alpha_m x) \quad (10.53)$$

where $\alpha_m = m\pi/L$ is the discrete wavenumber, $a_m$ and $b_m$ is the amplitude or energy associated with the wavenumber $\alpha_m$ and $a_0$ is the mean or average value of $\Psi$ 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

$$u(x, t) = \sum_n u_n(t) \sin(\alpha_n x). \quad (10.54)$$
Then nonlinear products will give rise to terms having wavenumbers which are the sum of and difference of the two original wavenumbers, e.g.,

$$\sin(\alpha_1 x) \sin(\alpha_2 x) = \frac{1}{2} \left[ \cos(\alpha_1 - \alpha_2) x - \cos(\alpha_1 + \alpha_2) x \right]$$  \hspace{1cm} (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 $\Delta x$ 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 $\Delta x$.

![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 $\Delta x$. 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.](image)

Points three and four above make us expect \textit{a priori} that even though all initial energy is low wavenumber (long waves), non-linear interactions will eventually provide variance (or energy) 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,\footnote{Note that (6.154) 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.} \hspace{1cm} (10.56)

\[ \partial_t u + u \partial_x u = 0. \]
The difference between (6.154) and the earlier advection equation, e.g., (5.1) on page 63, 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 \( \alpha \) is the wavenumber, that is, \( u^n(x) = u_0 \sin \alpha x \). Then from (6.154), using a scheme that is centered in time, we get

\[
\frac{u^{n+1} - u^{n-1}}{2} = -u_0^2 \sin \alpha x \partial_x \sin \alpha x = -u_0^2 \sin \alpha x \cos \alpha x = -\frac{1}{2} u_0^2 \sin 2\alpha x. \tag{10.57}
\]

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 \( \lambda \) where

\[
\sin \frac{\pi j}{\lambda} = 0 \quad \text{and} \quad \cos \frac{\pi j}{\lambda} = 1.
\]

For the case \( n = 1 \), we obtain relationships among 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,

\[
\begin{align*}
C_{n+1} - C_{n-1} &= 2\lambda S_n(U_n - V) \\
S_{n+1} - S_{n-1} &= 2\lambda C_n(U_n + V) \\
U_{n+1} &= U_{n-1}.
\end{align*}
\]

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

$$G^2 - 2\gamma G + 1 = 0,$$

(10.62)

where $\gamma$ is a real number given by

$$\gamma = 1 + 2\lambda^2(A + V)(B - V).$$

(10.63)

The roots of (10.62) are

$$G_{1,2} = \gamma \pm i\sqrt{1 - \gamma^2}.$$ 

(10.64)

We notice that as long as the radical is real then

$$|G_{1,2}| = \sqrt{\gamma^2 + 1} \equiv 1.$$ 

(10.65)

The $4\Delta x$ wave is therefore neutrally stable provided

$$1 - \gamma^2 \geq 0 \quad \text{or} \quad -1 \leq \gamma \leq 1.$$ 

(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 3.2 and Section 4.6. 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

$$\bar{u}_j^n = (1 - \mu)u_j^n + \frac{1}{2\mu}(u_{j+1}^n - u_{j-1}^n),$$

(10.67)

where $\mu$ is a constant. If the solution is a monochromatic wave, say $u_j = U_n e^{i\omega_j}\Delta x$, then the filtered solution is

$$\bar{u}_j^n = R u_j^n$$

(10.68)
where
\begin{equation}
R = 1 - \mu (1 - \cos \alpha \Delta x) = 1 - 2\mu \sin^2 \left( \frac{\alpha \Delta x}{2} \right)
\end{equation}
(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 \( \alpha = \frac{2\pi}{2\Delta x} \), that is, for the shortest wave resolved in our grid, we get
\begin{equation}
R = 1 - 2\mu.
\end{equation}
(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
\begin{equation}
\bar{u}_j^n = u_j^n + \frac{1}{2} \mu \left( u_{j+1}^n - 2u_j^n + u_{j-1}^n \right).
\end{equation}
(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 \( O(\Delta x^2) \). Thus we get
\begin{equation}
\bar{u}_j^n = u_j^n + \frac{1}{2} \mu \Delta x^2 \left[ \partial_x^2 u \right]_j^n,
\end{equation}
(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 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 27). 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 functions 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.
10.5 The spectral method

**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 18).

We recall from Section 5.1 on page 64 that the one-dimensional advection equation is

\[ \partial_t \phi = -u_0 \partial_x \phi, \quad \text{for} \quad x \in [0, L] \quad \text{and} \quad t > 0 \quad (10.73) \]

where \( L \) is the length of the circumference at a particular latitude. We recall from Section 2.10 that (10.73) is just a special case of the general equation (2.62) on page 27. Hence the linear operator of Section 2.10 is \( \mathcal{H} = -u_0 \partial_x \). Since we will solve (10.73) along a latitude we first conveniently transform to a coordinate system in which

\[ 2\pi x = \xi L \quad \text{or} \quad \xi = \frac{2\pi x}{L} \quad (10.74) \]

where \( \xi \in <0, 2\pi> \) is the new (dimensionless) zonal coordinate. Since

\[ \partial_x \phi = \partial_\xi \phi \partial_\xi \xi \quad (10.75) \]

(10.73) then transforms to

\[ \partial_t \phi = -\gamma \partial_\xi \phi \quad (10.76) \]

where

\[ \gamma = \frac{2\pi u_0}{L} \quad (10.77) \]

is the angular velocity (in units one per second). The cyclic boundary condition is written

\[ \phi(\xi, t) = \phi(\xi + 2\pi m) \quad m = 1, 2, 3, \ldots \quad (10.78) \]

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

\[ \phi(\xi, 0) = f(\xi). \quad (10.79) \]

As outlined at the beginning of Section 5.1 on page 63 the true solution to (10.76) is then

\[ \phi = f(\xi - \gamma t). \quad (10.80) \]

Solving (10.76) 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 165 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

\[ \phi = \sum_\alpha \phi_\alpha(t)e^{i\alpha \xi}, \quad (10.81) \]
where \( \alpha \) is the wave number and the summation is for all possible wavenumbers from \(-\infty\) to \(+\infty\). Solving (10.76) using numerical methods implies that we are band-limited in wavenumber space and hence we must use a truncated version of (10.81), that is,

\[
\phi(\xi, t) = \sum_{l=-l_{\text{max}}}^{l_{\text{max}}} \phi_l(t)e^{i\alpha_l \xi}. \tag{10.82}
\]

where \( l = l_{\text{max}} \) gives the maximum wave number \( \alpha_{l_{\text{max}}} \) 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 \( 0 \leq l \leq l_{\text{max}} \), rather than the full set of expansion coefficients.

We now substitute (10.82) into (10.76) and equate coefficients of the expansion functions. Thus

\[
\partial_t \phi_l = -i\alpha_l \gamma \phi_l \quad \text{for} \quad l = 0(1)l_{\text{max}} \tag{10.83}
\]

giving \( 2l_{\text{max}} + 1 \) equations for the expansion coefficients \( \phi_l \)'s. For this particular case (10.83) can be integrated exactly for each wavenumber \( \alpha_l \) separately to give

\[
\phi_l(t, \alpha_l) = \phi_l(0, \alpha_l)e^{-i\alpha_l \gamma t} \tag{10.84}
\]

where \( \phi_l(0, \alpha_l) \) is the initial condition associated with the wavenumber \( \alpha_l \). If we expand the good function \( f(\xi) \) in terms of a truncated Fourier series, that is,

\[
f(\xi) = \sum_l a_le^{i\alpha_l \xi} = \sum_l \phi_l(0, \alpha_l)e^{i\alpha_l \xi}, \tag{10.85}
\]

we get that \( \phi_l(0, \alpha_l) = a_l \), and hence that the complete solution to (10.76) is

\[
\phi(\xi, t) = \sum_{l=0}^{l_{\text{max}}} a_le^{i\alpha_l(\xi-\gamma t)}, \tag{10.86}
\]

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

\[
\phi_l = \int_0^{2\pi} \sum_m \phi_m e^{i\alpha_m \xi} e^{-i\alpha_l \xi} d\xi. \tag{10.87}
\]

Thus by multiplying (10.82) by the complex conjugate of the expansion functions and integrating in space we get

\[
\phi_l(t, \alpha_l) = A_l \int_0^{2\pi} \phi(\xi, t)e^{-i\alpha_l \xi} d\xi, \tag{10.88}
\]

where \( A_l \) are the normalization factors. Note that (10.88) is the so called direct Fourier transform. The normalization coefficients are determined from the initial condition, or by use of (10.88), that

\[
A_l = \frac{a_l}{\int_0^{2\pi} \phi(\xi, 0)e^{-i\alpha_l \xi} d\xi}. \tag{10.89}
\]
In practice we have at our disposal the grid points values of $\xi$ rather than a continuous function in space. Thus we know $\xi$ at $J+1$ points $\Delta \xi$ apart such that $\xi_j = j \Delta \xi$ where $j = 0, 1, 2, \ldots, J-1, J$ and where $\xi_J = 2\pi$. In this case we think of the truncated Fourier series of $\phi$ as given in (10.82) as representing an interpolating function which exactly fits the values of $\phi$ at the $J+1$ grid points. We then write (10.88) as a discrete direct Fourier transform,

$$
\phi_l(t, \alpha_l) = A'_l \sum_{j=1}^{J} \phi(\xi_j, t) e^{-i \alpha_l \xi_j},
$$

(10.90)

where the normalization coefficients are found by discretization (10.89),

$$
A'_l = \frac{a_l}{\sum_{j=1}^{J} \phi(\xi_j, 0) e^{-i \alpha_l \xi_j}}.
$$

(10.91)

The corresponding discrete inverse Fourier transform is then

$$
\phi(\xi_j, t) = \sum_{l=-l_{\text{max}}}^{l_{\text{max}}} \phi_l(t, \alpha_l) e^{i \alpha_l \xi_j}.
$$

(10.92)

Both (10.90) and (10.92) 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 $j = 0, 1, 2, \ldots, J-1, J$ 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 $J > 2l_{\text{max}} + 1$. Recall that $l_{\text{max}}$ is the number of waves used to compute the direct Fourier transform in (10.92). In addition we must require that the points $\xi_j$ 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 $\phi_0(t)$. We do this by a time stepping procedure, for instance applying a centered in time scheme to (10.83),

$$
\phi_{l+1} = \phi_{l-1} + 2i \alpha_l \gamma \Delta t \phi_{l} \quad l = 1, 2, 3, \ldots, LM
$$

(10.93)

for each wavenumber $\alpha_l$. Using von Neumann’s method we show that numerical stability is ensured provided

$$
|\alpha_l \gamma \Delta t| \leq 1; \quad \forall l.
$$

(10.94)

Since the maximum wavenumber is $\alpha_{LM}$ we require $|\alpha_{LM} \gamma \Delta t| \leq 1$. Moreover, since the maximum dimensionless wavenumber$^3$ 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

$$
C \leq \frac{1}{\pi},
$$

(10.95)

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.

$^3$The maximum dimensional wavenumber is $2\pi/2\Delta x$. 

172
Exercises

1. Use von Neumann’s method (Chapter 4.3) 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.
Appendix A

Introduction to Fortran 2003 via examples

by Gunnar Wollan 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, POM, or other models like them. Depending on your project, you will sometimes have to make changes or additions to an already existing 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 acquire 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
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 Formula Translation, 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 object oriented programming (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.
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 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 2003 compiler.

<table>
<thead>
<tr>
<th>Column number</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A character here means the line is a comment</td>
</tr>
<tr>
<td>2 - 5</td>
<td>Jump address and format number</td>
</tr>
<tr>
<td>6</td>
<td>A character here is a continuation from previous line</td>
</tr>
<tr>
<td>7 - 72</td>
<td>Program code</td>
</tr>
<tr>
<td>73 - 80</td>
<td>Comment</td>
</tr>
</tbody>
</table>

Table A.1: The Fortran 77 (F77) fixed format
A.3.1 Data types in Fortran

In earlier version 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 tab:reald 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.

<table>
<thead>
<tr>
<th>Precision</th>
<th>Sign</th>
<th>Exponent</th>
<th>Significand</th>
<th>Max. value</th>
<th>Min. value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>1</td>
<td>8</td>
<td>23</td>
<td>$2^{128}$</td>
<td>$2^{-126}$</td>
</tr>
<tr>
<td>Double</td>
<td>1</td>
<td>11</td>
<td>52</td>
<td>$2^{1024}$</td>
<td>$2^{-1022}$</td>
</tr>
</tbody>
</table>

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). Note that 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 note that some variables must always 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.
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.

```fortran
REAL :: x ! Declaration of REAL
         ! default length (32bit)
REAL(KIND=8) :: y ! Declaration of REAL
                   ! double precision (64bit)
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.

```fortran
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.

---

4George 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).
A.4 The structure of Fortran

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.

<table>
<thead>
<tr>
<th>CHARACTER</th>
<th>: c1 ! Single character</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER(LEN=80)</td>
<td>: c2 ! String of characters</td>
</tr>
<tr>
<td>CHARACTER, DIMENSION(10)</td>
<td>: c3 ! Array of single characters</td>
</tr>
<tr>
<td>CHARACTER(LEN=80), DIMENSION(10)</td>
<td>: c4 ! Array of character strings (10 elements)</td>
</tr>
</tbody>
</table>

Derived data types

The Fortran 95 syntax for the declaration of a derived datatype can be like the one shown here.

```fortran
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 desirable 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-leap-year. 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, mandatory. 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.

```fortran
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`.

```fortran
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).

```fortran
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.

```fortran
PROGRAM daynumber
  IMPLICIT NONE
  ....
  DO counter = 1, month - 1
    daynr = daynr + months(counter)
  END DO
  daynr = daynr + day
  PRINT*, daynr
END PROGRAM daynumber
```

In order to have a 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 contain a Fortran program written 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 assembly language, that is, instructions written in machine language. We use an open source compiler\(^5\) 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 severe damage.

**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

---

\(^5\)Open source means that the compiler may be downloaded and used free of charge

\(^6\)Note that there are other compilers.
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 degrees 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:

```fortran
!// 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.

The next step is to initialize the Fahrenheit variable and perform the conversion according to the formula. The whole program may be something like,

\[^7\text{In other languages the keyword } \text{float} \text{ is used to declare a floating point variable.}\]
PROGRAM f2c_simple
  IMPLICIT NONE
  !// Declare variables
  REAL :: F ! Fahrenheit variable (floating point)
  REAL :: C ! Centigrade variable C (floating point)
  !// Assign 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 program (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 degrees 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
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
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 \texttt{prompt1} to the terminal window, then read our input and put it in \texttt{answer}. Next it prints the \texttt{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 \texttt{READ} command to read in our answer to \texttt{prompt1}. The construct \texttt{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 \texttt{answer}, performs the conversion and prints the result before it ends the program in a proper way. Thus

\begin{verbatim}
!// 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
\end{verbatim}

\subsection*{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

\(^8\)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.
store them in a one dimensional array or vector of integers. We first recall that the formula for calculating the Fibonacci sequence is

\[ F_{j-1} = F_{j-2} + F_{j-3}, j = 3(1)n \]  

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 length 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

```fortran
!////////////////////////////////////////////////// //////////
!// 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

```fortran
!// 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 the WRITE command makes it possible to avoid or suppress 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

```fortran
!// 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: \texttt{ALLOCATE(sequence(n), STAT=res)}. Then we make an if test using the integer \( \text{res} \) 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 \texttt{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 are 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

```fortran
PROGRAM f2c_file
IMPLICIT NONE
    INTEGER, PARAMETER :: n = 7
....
```

Here \( n \) is declared using the keyword \texttt{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 variable 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
A.5 Sample programs

Introduction to Fortran 2003 via examples

...!

// 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

```fortran
!// 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 Fahrenheit ", 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 \textit{outfile} and gave the reference number \( \text{olun} = 11 \). 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

```fortran
!// 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 specifically ensured that also the \textit{outfile} is formatted by including the argument \texttt{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=ilun,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,

```fortran
!// 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 containing two pairs of seven temperatures in Fahrenheit in two columns formatted following the keyword/argument FMT='FMT='(F4.1,X,F4.1)', that is,

<table>
<thead>
<tr>
<th>68.2</th>
<th>65.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>69.6</td>
<td>63.7</td>
</tr>
<tr>
<td>73.2</td>
<td>66.0</td>
</tr>
<tr>
<td>75.0</td>
<td>68.0</td>
</tr>
<tr>
<td>77.5</td>
<td>70.2</td>
</tr>
<tr>
<td>79.2</td>
<td>71.4</td>
</tr>
<tr>
<td>91.2</td>
<td>73.2</td>
</tr>
</tbody>
</table>

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><B4>
^C3<0>H<C7>D$8<80><DC>s^@L<8D>D$0H<C7>D$0
^@^@^@H<C7>D$H<E0>'N^@H<C7>D$P^D^@^@^@<E8>0<C3>
.....
```

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\(^\text{10}\)

\(^\text{10}\)We have split it into parts here to avoid open spaces
A.5 Sample programs

Introduction to Fortran 2003 via examples

!//////////////////////////////////////////////////// //////////
//
// 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
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. 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.

\footnote{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 FUNCTIONs or SUBROUTINEs.

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 by 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 \( f2c \) \((\text{arg})\), and a second that converts from Celsius to Fahrenheit, which we call \( c2f \) \((\text{arg})\). First we program \( f2c \) \((\text{arg})\),

```fortran
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 \( c2f \) \((\text{arg})\),

```fortran
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\((\text{arg})\) where the datatype of the argument states what kind of function it is. In For-
TRAN 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 an error and the compilation will be aborted. The complete main program is shown below.
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 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

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 Celsius 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
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
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:

```fortran
!////////////////////////////////////////////////////////////////////////
!// SUBROUTINE f2c(F,C)
!////////////////////////////////////////////////////////////////////////
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
   ....
END SUBROUTINE f2c
```

The program continues
A.5 Sample programs

Introduction to Fortran 2003 via examples

......
!// 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 subroutine 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
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 equation 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 model’s 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 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

\[
\partial_t C + \partial_x (uC) = 0,
\]  

(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 \( \nabla \) is the three-dimensional gradient vector.
potential temperature or the contaminant in question, while the second term is the divergence of the advective flux $F_{\text{adv}} = uC$, 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, $\bar{u}$, and a random component, $u'$, such that $u = \bar{u} + u'$ where $\bar{u} = 0$. 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

$$\partial_t(\bar{C} + C') + \partial_x \left[(\bar{u} + u')(\bar{C} + C')\right] = 0.$$  \hspace{1cm} (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

$$\partial_t C + \partial_x (\bar{u}C') = \partial_x \left(u'C'\right).$$ \hspace{1cm} (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,

$$F_{\text{diff}} = -u'C' = -K(x,t)\partial_x C,$$ \hspace{1cm} (B.4)

where $F_{\text{diff}}$ 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/transportation formulas.

\(^2\)In a three-dimensional problem the diffusive flux becomes a vector $\mathbf{F}_{\text{diff}} = -\mathbf{K} \cdot \nabla C$, where $\mathbf{K}$ is a tensor. The right-hand side of (B.3) is then written $\nabla \cdot \mathbf{F}_{\text{diff}}$.  

210
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.
B.3 What is a good model

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 description 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 sim-
ple parameterizations. The range of variations in the input data can be determined from a knowl-
edge 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 parameter-
izations 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 associ-
ated 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 turbu-
lence, 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 biolog-
ical 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 pa-
rameterizations 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 anal-
B.4 Quality assurance procedures

Quality assurance procedures

ysis. 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 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 exercise 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.


Eliassen, A. (1949), The quasi-static equations of motion with pressure as independent variable, Geof. publ., 17(3), 44 pp.


Mesinger, F., and A. Arakawa (1976), Numerical methods used in atmospheric models, GARP Publication Series No. 17, 64 p., World Meteorological Organization, Geneva, Switzerland.


BIBLIOGRAPHY


