Numerical Methods in Fluid Dynamics MPO 662

Instructor Mohamed Iskandarani
MSC 320 x 4045
miskandarani@rsmas.miami.edu

Grades 60% Homework (involve programming)
20% Mid term
20% Term project

Syllabus

1. Introduction
2. Classifications of PDE's and their properties
3. Basics of the finite difference method
4. Finite difference solutions of ODE
5. Finite difference solutions of time-dependent linear PDEs
   (a) advection equation
   (b) heat equation
   (c) Stability and dispersion properties of time differencing schemes
6. Numerical solution of finite difference approximation of elliptic equations
7. Special advection schemes
8. Energetically consistant finite difference schemes
9. The Finite Element Method
10. Additional topics (time permitting)

Background

1. Name

2. Degree Sought (what field)

3. Advisor (if any)

4. Background

5. Scientific Interest

6. Background in numerical modeling

7. Programming experience/language

Reserve List

- Dale B. Haidvogel and Aike Beckmann, *Numerical Ocean Circulation Modeling* Imperial College Press, 1999. (CGFD)
- Dale R. Durran, *Numerical Methods for Wave Equations in Geophysical Fluid Dynamics*, Springer, New York, 1998. (CGFD)
- George J. Haltiner Roger T. Williams, *Numerical Prediction and Dynamic Meteorology*, Wiley, 1980. (CGFD)
- John C. Tannehill, Dale A. Anderson, and Richard H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, Taylor and Francis, 1997. (FDM)
- G. E. Forsythe and W. R. Wasow *Finite-Difference Methods for Partial Differential Equations*, John Wiley and Sons, Inc., New York, 1960. (FDM)
- R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial–Value Problems*, Interscience Publishers (J. Wiley & Sons), New York, 1967.

Useful References

- Gordon D. Smith, *Numerical Solution of Partial Differential Equations : Finite Difference Methods*, Oxford University Press, New York, 1985. (FDM)
- K.W. Morton and D.F. Mayers, *Numerical Solution of Partial Differential Equations : An Introduction*, Cambridge University Press, New York, 1994. (FDM)
- P.J. Roache, *Computational Fluid Dynamics*, Hermosa Publisher, 1972, ISBN 0-913478-05-9. (FDM)
- C.A.J. Fletcher, *Computational Techniques for Fluid Dynamics*, 2 volumes, 2nd ed., Springer-Verlag, New York, 1991-1992. (Num. Sol. of PDE's)
- Roger Peyret and Thomas D. Taylor, *Computational Methods for Fluid Flow*, Springer-Verlag, New York, 1990. (Num. Sol. of PDE's)
- Roger Peyret, *Handbook of Computational Fluid Mechanics*, Academic Press, San Diego, 1996. (QA911 .H347 1996)
- Joel H. Ferziger and M. Peric *Computational Methods For Fluid Dynamics*, Springer-Verlag, New York, 1996.
- R. S. Varga, *Matrix Iterative Analysis*, Prentice–Hall, New York, 1962.
- Bengt Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, 1998. (Spectral methods)
- C. Canuto, M.Y. Hussaini, A. Quarteroni and T.A. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1991. (Spectral Methods)

- John P. Boyd, *Chebyshev and Fourier Spectral Methods* Dover Publications, 2000. (Spectral methods)
- O.C. Zienkiewicz and R.L. Taylor, *The Finite Element Method*, 4^{th} edition, Mc Graw Hill, 1989.
- George Em. Karniadakis and Spencer J. Sherwin, *Spectral $h-p$ Element Methods for CFD*, New York, Oxford University Press, 1999. (Spectral Elements)
- Michel O. Deville, Paul F. Fischer and E.H. Mund, *High-Order Methods for Incompressible Fluid Flow*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2002.

Useful Software

- Plotting Software (e.g. matlab, NCAR Graphics, gnuplot)
- Linear Algebra (e.g. matlab, LAPACK, IMSL)
- Fast Fourier Transforms (e.g. matlab, fftpack, ?)
- Fortran Compiler (debuggers are useful too)

Numerical Methods in Ocean Modeling
Lecture Notes for MPO662

December 2, 2010

Contents

Chapter 1

Introduction

1.1 Justification of CFD

Fluid motion is governed by the Navier-Stokes equations, a set of coupled and nonlinear partial differential equations derived from the basic laws of conservation of mass, momentum and energy. The unknowns are usually the velocity, the pressure and the density (for stratified fluids) and some tracers like temperature and salinity. The analytical paper and pencil solution of these equations is practically impossible save for the simplest of flows. The simplifications can take the form of geometrical simplification (the flow is in a rectangle or a circle), and/or physical simplification (periodicity, homogeneous density, linearity, etc...). Occasionally, it is possible to make headway by using asymptotic analyses technique, and there has been remarkable success in the past O(100) year, like the development of boundary layer theory.

Scientists had to resort to laboratory experiments when theoretical analyses was impossible. Physical complexity can be restored to the system. The answers delivered are, however, usually qualitatively different since dynamical and geometric similitudes are difficult to enforce simultaneously between the lab experiment and the prototype. A prime example is the Reynolds' number similarity which if violated can turn a turbulent flow laminar. Furthermore, the design and construction of these experiments can be difficult (and costly), particularly for stratified rotating flows.

Computational fluid dynamics (CFD) is an additional tool in the arsenal of scientists. In its early days CFD was often controversial, as it involved additional approximation to the governing equations and raised additional (legitimate) issues. Nowadays CFD is an established discipline alongside theoretical and experimental methods. This position is in large part due to the exponential growth of computer power which has allowed us to tackle ever larger and more complex problems.

1.2 Discretization

The central process in CFD is the process of discretization, i.e. the process of taking differential equations with an *infinite* number of degrees of freedom, and reducing it to a system of *finite* degrees of freedom. Hence, instead of determining the solution

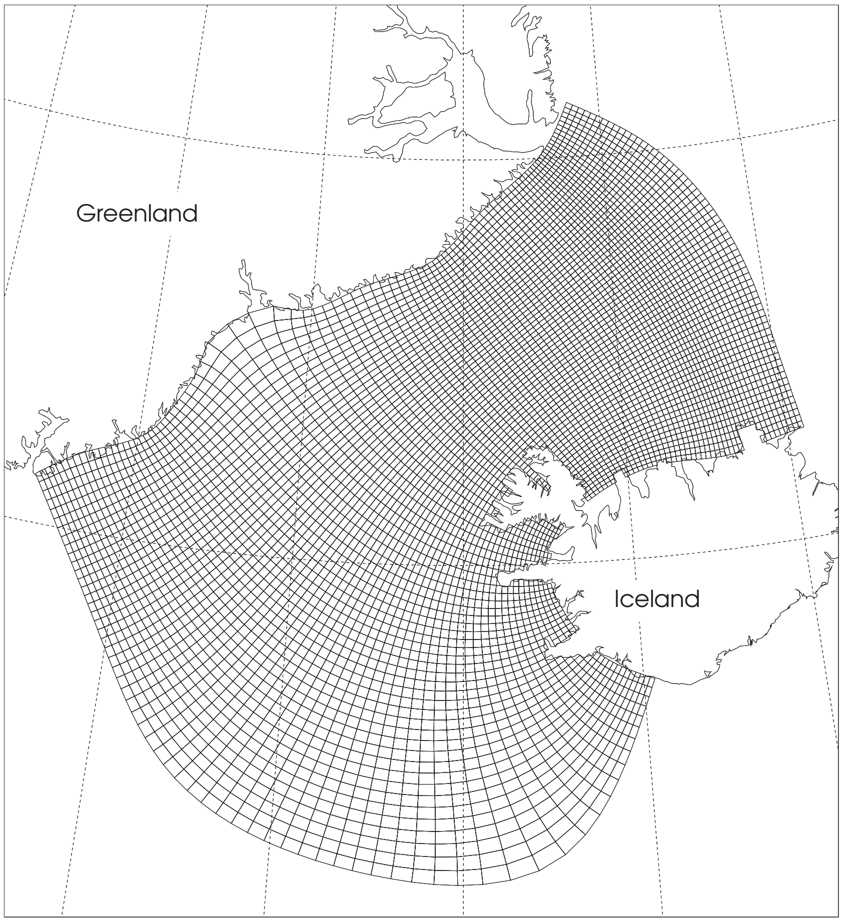


Figure 1.1: Computation grid for a finite difference ocean model

everywhere and for all times, we will be satisfied with its calculation at a finite number of locations and at specified time intervals. The partial differential equations are then reduced to a system of algebraic equations that can be solved on a computer.

Errors creep in during the discretization process. The nature and characteristics of the errors must be controlled in order to ensure that 1) we are solving the correct equations (consistency property), and 2) that the error can be decreased as we increase the number of degrees of freedom (stability and convegence). Once these two criteria are established, the power of computing machines can be leveraged to solve the problem in a numerically reliable fashion.

Various discretization schemes have been developed to cope with a variety of issues. The most notable for our purposes are: finite difference methods, finite volume methods, finite element methods, and spectral methods.

1.2.1 Finite Difference Method

Finite difference replace the infinitesimal limiting process of derivative calculation

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{1.1}$$

with a finite limiting process,i.e.

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta x} + O(\Delta x) \tag{1.2}$$

The term $O(\Delta x)$ gives an indication of the magnitude of the error as a function of the mesh spacing. In this instance, the error is halfed if the grid spacing, Δx is halved, and we say that this is a first order method. Most FDM used in practice are at least second order accurate except in very special circumstances. We will concentrate mostly on finite difference methods since they are still among the most popular numerical methods for the solution of PDE's because of their simplicity, efficiency, low computational cost, and ease of analysis. Their major drawback is in their geometric inflexibility which complicates their applications to general complex domains. These can be alleviated by the use of either mapping techniques and/or masking to fit the computational mesh to the computational domain.

1.2.2 Finite Element Method

The finite element method was designed to deal with problem with complicated computational regions. The PDE is first recast into a variational form which essentially forces the mean error to be small everywhere. The discretization step proceeds by dividing the computational domain into elements of triangular or rectangular shape. The solution within each element is interpolated with a polynomial of usually low order. Again, the unknowns are the solution at the collocation points. The CFD community adopted the FEM in the 1980's when reliable methods for dealing with advection dominated problems were devised.

1.2.3 Spectral Methods

Both finite element and finite difference methods are low order methods, usually of 2nd-4th order, and have local approximation property. By local we mean that a particular collocation point is affected by a limited number of points around it. In contrast, spectral method have global approximation property. The interpolation functions, either polynomials or trigonomic functions are global in nature. Their main benefits is in the rate of convergence which depends on the smoothness of the solution (i.e. how many continuous derivatives does it admit). For infinitely smooth solution, the error decreases exponentially, i.e. faster than algebraic. Spectral methods are mostly used in the computations of homogeneous turbulence, and require relatively simple geometries. Atmospheric model have also adopted spectral methods because of their convergence properties and the regular spherical shape of their computational domain.

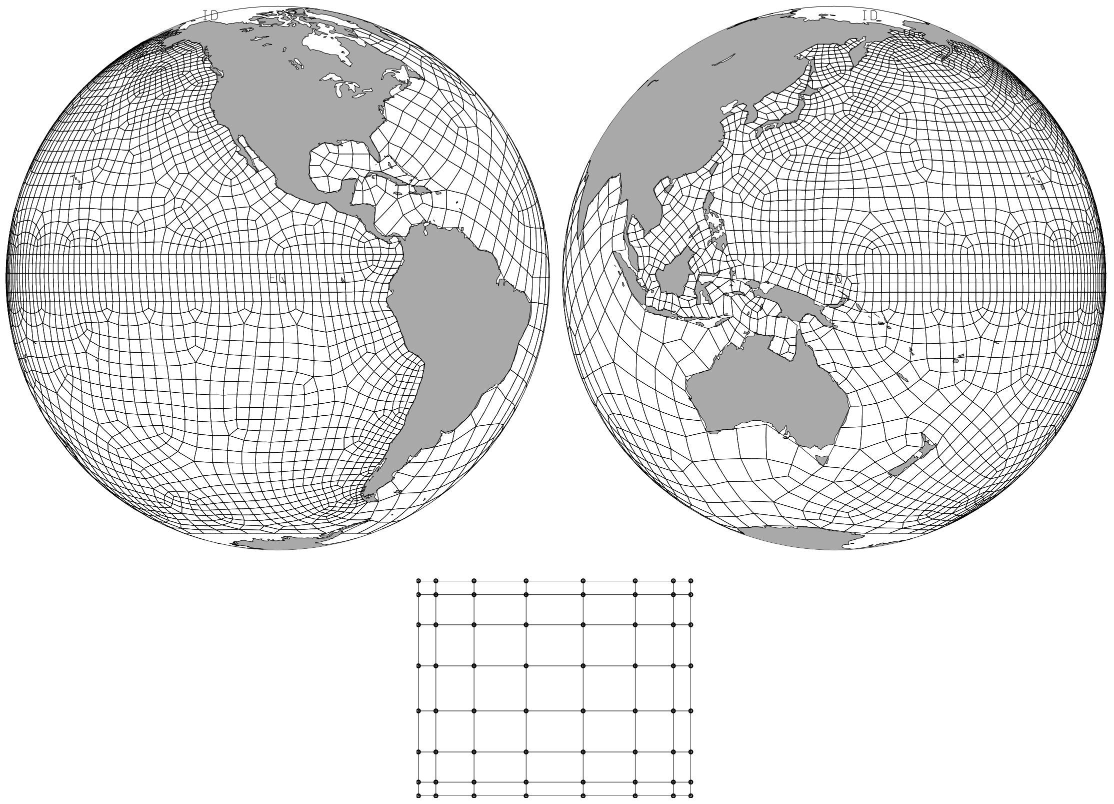


Figure 1.2: Elemental partition of the global ocean as seen from the eastern and western equatorial Pacific. The inset shows the master element in the computational plane. The location of the interpolation points is marked with a circle, and the structuredness of this grid local grid is evident from the predictable adjacency pattern between collocation points.

1.2.4 Finite Volume Methods

Finite volume methods are primarily used in aerodynamics applications where strong shocks and discontinuities in the solution occur. Finite volume method solves an integral form of the governing equations so that local continuity property do not have to hold.

1.2.5 Computational Cost

The CPU time to solve the system of equations differ substantially from method to method. Finite differences are usually the cheapest on a per grid point basis followed by the finite element method and spectral method. However, a per grid point basis comparison is a little like comparing apple and oranges. Spectral methods deliver more accuracy on a per grid point basis than either FEM or FDM. The comparison is more meaningfull if the question is recast as "what is the computational cost to achieve a given error tolerance?". The problem becomes one of defining the error measure which is a complicated task in general situations.

1.3 Initialization and Forcing

The state of a system is determined by its intial state, (conditions at t=0), the forces acting on the system (sources and sinks of momentum, heat, buoyancy), the boundary conditions, and the governing equations. Given this information one can in principle integrate the equations forward to find the state of the system at a future time. Things are not so simple in practice. First, the initial state is seldom known accurately. In spite of advances in measurements and instrumentations, data deficiencies still exist and manifest themselves in either inadequate temporal or spatial coverage or measuring errors. The situation is more diare in the ocean than the atmosphe because the dynamical scales are smaller and require more measurement per unit area, and because of observational difficulties. Furthermore, the fluxes between the ocean and the atmosphere are not well known and constitute another modeling impediment. The discipline of data assimilation is devoted to the task of integrating data and model in optimal fashion. This is topic we will not touch upon in this course.

1.4 Turbulence

Most flows occuring in nature are turbulent, in that they contain energy at all scales ranging from hundred of kilometers to a few centimeters. It is obviously not possible to model all these scales at once. It is often sufficient to model the ”large scale physics” and to relegate the small unresolvable scale to a subgrid model. Subgrid models occupy a large discipline in CFD, we will use the simplest of these models which rely on a simple eddy diffusivity coefficient.

1.5 Examples of system of equations and their properties

The numerical solution of any partial differential equations should only proceed after carefull consideration of the dynamical settings of the problem. A prudent modeler will try to learn as much as he can about the system he is trying to solve, for ortherwise how can he judge the results of his simulations? Errors due to numerical approximation are sometimes notoriously hard to catch and diagnose, and an knoweldge of the expected behavior of the system will go a long way in helping catch these errors. Furthermore, one can often simplify the numerical solution process by taking advantages of special features in the solution, such as symmetry or a reduction of the number of unknowns, to reduce the complexity of the analytical and numerical formulations. In the following sections we consider the case of the two dimensional incompressible Navier Stokes equations to illustrate some of these points.

The incompressible Navier-Stokes equations is an example of a system of partial differential equations governing a large class of fluid flow problems. We will confine ourselves to two-dimensions for simplicity. The primtive form of these equations is:

$$\mathbf{v}_t + \mathbf{v}\cdot\nabla\mathbf{v} = -\frac{1}{\rho_0}\nabla p + \nu\nabla^2\mathbf{v} \qquad \text{(momentum conservation)} \tag{1.3}$$

$$\nabla\cdot\mathbf{v} = 0 \qquad \text{(mass conservation)} \tag{1.4}$$

supplemented with proper boundary and initial conditions. The unknowns are the two components of the velocity $\mathbf{v}$ and the pressure p so that we have 3 unknowns functions to determine. The parameters of the problem are the density and the kinematic viscosity which are assumed constant in this example. Equation (1.3) expresses the conservation of momentum, and equation (1.4), also referred to as the continuity equation, conservation of mass which, for a constant density fluid, amount to conservation of volume. The form of equations (1.3)-(1.4) is called primitive because it uses velocity and pressure as its dependent variables.

In the early computer days memory was expensive, and modelers had to use their ingenuity to reduce the model's complexity as much as possible. Furthermore, the incompressibility constraint complicates the solution process substantially since there is no simple evolution equation for the pressure. The streamfunction vorticity formulation of the two-dimensional Navier Stokes equations addresses both difficulties by enforcing the continuity requirement implicitly, and reducing the number of unknown functions. The streamfunction vorticity formulation introduces other complications in the solution process that we will ignore for the time being. Alas, this is a typical occurence where a cure to one set of concerns raises quite a few, and sometimes, irritating side-effects. The streamfunction is defined as follows:

$$u = -\psi_y, \quad v = \psi_x. \tag{1.5}$$

Any velocity derivable from such a streamfunction is that guaranteed to conserve mass since

$$u_x + v_y = (-\psi_y)_x + (\psi_x)_y = -\psi_{yx} + \psi_{xy} = 0.$$

To simplify the equations further we introduce the vorticity $\zeta = v_x - u_y$ (a scalar in 2D flows), which in terms of the streamfunction is

$$\nabla^2\psi = \zeta. \tag{1.6}$$

We can derive an evolution equation for the vorticity by taking the curl of the momentum equation, Eq. (1.3). The final form after using various vector identities are:

$$\begin{array}{ccccc}
\underbrace{\zeta_t}_{\text{time rate of change}} & + & \underbrace{\mathbf{v}\cdot\zeta}_{\text{advection}} & = & \underbrace{\nu\nabla^2\zeta}_{\text{diffusion}} \\
\dfrac{\Omega}{T} & & \dfrac{U\Omega}{L} & & \dfrac{\nu\Omega}{L^2} \\
1 & & \dfrac{UT}{L} & & \dfrac{\nu T}{L^2} \\
1 & & 1 & & \dfrac{\nu}{UL} = \dfrac{1}{R_e}
\end{array} \tag{1.7}$$

Note that the equation simplifies considerably since the pressure does not appear in the equation (the curl of a gradient is always zero). Equations (1.6) and (1.7) are now a system of two equations in two unknowns: the vorticity and streamfunction. The pressure has disappeared as an unknown, and its role has been assumed by the streamfunction. The two physical processes governing the evolution of vorticity are advection by the flow and diffusion by viscous action. Equation (1.7) is an example of parabolic partial differential equation requiring initial and boundary data for its unique solution. Equation (1.6) is an example of an elliptic partial differential equation. In this instance it is a Poisson equation linking a given vorticity distribution to a streamfunction. Occasionally the term prognostic and diagnostic are applied to the vorticity and streamfunction, respectively, to mean that the vorticity evolves dynamically in time to balance the conservation of momentum, while the streamfunction responds instantly to changes in vorticity to enforce kinematic constraints. A numerical solution of this coupled set of equations would proceed in the following manner: given an initial distribution of vorticity, the corresponding streamfunction is computed from the solution of the Poisson equations along with the associated velocity; the vorticity equation is then integrated in time using the previous value of the unknown fields; the new streamfunction is subsequently updated. The process is repeated until the desired time is reached.

In order to gauge which process dominates, advection or diffusion, the vorticity evolution, we proceed to non-dimensionalize the variables with the following, time, length and velocity scales, T, L and U, respectively. The vorticity scale is then $\Omega = U/L$ from the vorticity definition. The time rate of change, advection and diffusion then scale as Ω/T, $U\Omega/L$, and $\nu\Omega/L^2$ as shown in the third line of equation 1.7. Line 4 shows the relative sizes of the term after multiplying line 3 by T/Ω. If the time scale is chosen to be the advective time scale, i.e. $T = L/U$, then we obtain line 5 which shows a single dimensionless parameter, the Reynolds number, controlling the evolution of ζ. When $R_e \ll 1$ diffusion dominates and the equation reduces to the so called heat equation $\zeta = \nu\nabla^2\zeta$. If $R_e \gg 1$ advection dominates *almost* everywhere in the domain. Notice that dropping the viscous term is problematic since it has the highest order derivative, and hence controls the imposition of boundary conditions. Diffusion then has to become dominant near the boundary through an increase of the vorticity gradient in a thin boundary layers where advection and viscous action become balanced.

What are the implications for numerical solution of all the above analysis. By carefully analysing the vorticity dynamics we have shown that a low Reynolds number simulation requires attention to the viscous operator, whereas advection dominates in high Reynolds number flow. Furthermore, close attention must be paid to the boundary layers forming near the edge of the domain. Further measures of checks on the solution can be obtained by spatially integrating various forms of the vorticity equations to show that energy, kinetic energy here, and enstrophy $\zeta^2/2$ should be conserved in the inviscid case, $R_e = \infty$, when the domain is closed.

Chapter 2

Basics of PDEs

Partial differential equations are used to model a wide variety of physical phenomena. As such we expect their solution methodology to depend on the physical principles used to derive them. A number of properties can be used to distinguish the different type of differential equations encountered. In order to give concrete examples of the discussions to follow we will use as an example the following partial differential equation:

$$au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + f = 0. \tag{2.1}$$

The unknown function in equation (2.1) is the function u which depends on the two independent variables x and y. The coefficients of the equations $a, b, \dots, f$ are yet undefined.

The following properties of a partial differential equation are useful in determining its character, properties, method of analysis, and numerical solution:

Order : The order of a PDE is the order of the highest occuring derivative. The order in equation (2.1) is 2. A more detailed description of the equation would require the specification of the order for each independent variable. Equation 2.1 is second order in both x and y. Most equations derived from physical principles, are usually first order in time, and first or second order in space.

Linear : The PDE is linear if none of its coefficients depend on the unknown function. In our case this is tantamount to requiring that the functions $a, b, \dots, f$ are independent of u. Linear PDEs are important since their linear combinations can be combined to form yet another solution. More mathematically, if u and v are solution of the PDE, the so is $w = \alpha u + \beta v$ where α and β are constants independent of u, x and y. The Laplace equation

$$u_{xx} + u_{yy} = 0$$

is linear while the one dimensional Burger equation

$$u_t + uu_x = 0$$

is nonlinear. The majority of the numerical analysis results are valid for linear equations, and little is known or generalizable to nonlinear equations.

Quasi Linear A PDE is quasi linear if it is linear in its highest order derivative, i.e. the coefficients of the PDE multiplying the highest order derivative depends *at most* on the function and lower order derivative. Thus, in our example, a, b and c may depend on u, u_x and u_y but not on u_{xx}, u_{yy} or u_{xy}. Quasi linear equations form an important subset of the larger nonlinear class because methods of analysis and solutions developed for linear equations can safely be applied to quasi linear equations. The vorticity transport equation of quasi-geostrophic motion:

$$\frac{\partial \nabla^2 \psi}{\partial t} + \left(\frac{\partial \psi}{\partial y} \frac{\partial \nabla^2 \psi}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \nabla^2 \psi}{\partial y} \right) = 0, \tag{2.2}$$

where $\nabla^2 = \psi_{xx} + \psi_{yy}$ is a third order quasi linear PDE for the streamfunction ψ.

2.1 Classification of second order PDEs

The following classification are rooted in the character of the physical laws represented by the PDEs. However, these characteristics can be given a definite mathematical classification that at first sight has nothing to do with their origins. We will attempt to link the PDE's category to the relevant physical roots.

The first question in attempting to solve equation 2.1 is to attempt a transformation of coordinates (the independent variables x and y) in order to simpplify the equation. The change of variable can take the general form:

$$\begin{aligned} x &= x(\xi, \eta) \\ y &= y(\xi, \eta) \end{aligned} \tag{2.3}$$

where ξ and η are the new independent variables. This is equivalent to a change of coordinates. Using the chain rule of differentiation we have:

$$u_x = u_\xi \xi_x + u_\eta \eta_x \tag{2.4}$$
$$u_y = u_\xi \xi_y + u_\eta \eta_y \tag{2.5}$$
$$u_{xx} = u_{\xi\xi} \xi_x^2 + 2u_{\xi\eta} \xi_x \eta_x + u_{\eta\eta} \eta_x^2 + u_\xi \xi_{xx} + u_\eta \eta_{xx} \tag{2.6}$$
$$u_{yy} = u_{\xi\xi} \xi_y^2 + 2u_{\xi\eta} \xi_y \eta_y + u_{\eta\eta} \eta_y^2 + u_\xi \xi_{yy} + u_\eta \eta_{yy} \tag{2.7}$$
$$u_{xy} = u_{\xi\xi} \xi_x \xi_y + u_{\xi\eta} (\xi_x \eta_y + \xi_y \eta_x) + u_{\eta\eta} \eta_x \eta_y + u_\xi \xi_{xy} + u_\eta \eta_{xy} \tag{2.8}$$

Substituting these expression in PDE 2.1 we arrive at the following equation:

$$A u_{\xi\xi} + B u_{\xi\eta} + C u_{\eta\eta} + D u_\xi + E u_\eta + F = 0, \tag{2.9}$$

where the coefficients are given by the following expressions:

$$A = a\xi_x^2 + b\xi_x \xi_y + c\xi_y^2 \tag{2.10}$$
$$B = 2a\xi_x \eta_x + b(\xi_x \eta_y + \xi_y \eta_x) + 2c\xi_y \eta_y \tag{2.11}$$
$$C = a\eta_x^2 + b\eta_x \eta_y + c\eta_y^2 \tag{2.12}$$
$$D = d\xi_x + e\xi_y \tag{2.13}$$
$$E = d\eta_x + e\eta_y \tag{2.14}$$
$$F = f \tag{2.15}$$

The equation can be simplified if ξ and η can be chosen such that $A = C = 0$ which in terms of the transformation factors requires:

$$\left\{ \begin{array}{rcl} a\xi_x^2 + b\xi_x\xi_y + c\xi_y^2 & = & 0 \\ a\eta_x^2 + b\eta_x\eta_y + c\eta_y^2 & = & 0 \end{array} \right. \tag{2.16}$$

Assuming ξ_y and η_y are not equal to zero we can rearrange the above equation to have the form $ar^2 + br + c = 0$ where $r = \xi_x/\xi_y$ or η_x/η_y. The number of roots for this quadratic depends on the sign of the determinant $b^2 - 4ac$. Before considering the different cases we note that the sign of the determinant is independent of the choice of the coordinate system. Indeed it can be easily shown that the determinant in the new system is $B^2-4AC = (b^2-4ac)(\xi_x\eta_y-\xi_y\eta_x)^2$, and hence the same sign as the determinant in the old system since the quantity $(\xi_x\eta_y - \xi_y\eta_x)$ is nothing but the squared Jacobian of the mapping between (x, y) and (ξ, η) space, and the Jacobian has to be one-signed for the mapping to be valid.

2.1.1 Hyperbolic Equation: $b^2 - 4ac > 0$

In the case where $b^2 - 4ac > 0$ equation has two distincts real roots and the equation is called *hyperbolic*. The roots are given by $r = \frac{-b\pm\sqrt{b^2-4ac}}{2a}$. The coordinate transformation required is hence given by:

$$\frac{\xi_x}{\xi_y} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \tag{2.17}$$

$$\frac{\eta_x}{\eta_y} = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{2.18}$$

The interpretation of the above relation can be easily done by focussing on constant ξ surfaces where $\mathrm{d}\xi = \xi_x\ \mathrm{d}x + \xi_y\ \mathrm{d}y = 0$, and hence:

$$\left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_\xi = -\frac{\xi_x}{\xi_y} = \frac{b-\sqrt{b^2-4ac}}{2a} \tag{2.19}$$

$$\left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_\eta = -\frac{\eta_x}{\eta_y} = \frac{b+\sqrt{b^2-4ac}}{2a} \tag{2.20}$$

The roots of the quadratic are hence nothing but the slope of the constant ξ and constant η curves. These curves are called the characteristic curves. They are the preferred direction along which information propagate in a hyperbolic system.

In the (ξ, η) system the equation takes the canonical form:

$$Bu_{\xi\eta} + Du_\xi + Eu_\eta + F = 0 \tag{2.21}$$

The solution can be easily obtained in the case $D = E = F = 0$, and takes the form:

$$u = G(\xi) + H(\eta) \tag{2.22}$$

where G and H are function determined by the boundary and initial conditions.

Example 1 The one-dimensional wave equation:

$$u_{tt} - \kappa^2 u_{xx} = 0, \quad -\infty \leq x \leq \infty \tag{2.23}$$

where κ is the wave speed is an example of a hyperbolic system, since its $b^2 - 4ac = 4\kappa^2 > 0$. The slope of the charasteristic curves are given by

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \pm\kappa, \tag{2.24}$$

which, for constant κ, gives the two family of characteristics:

$$\xi = x - \kappa t, \quad \eta = x + \kappa t \tag{2.25}$$

Initial conditions are needed to solve this problem; assuming they are of the form:

$$u(x,0) = f(x), \quad u_t(x,0) = g(x), \tag{2.26}$$

we get the equations:

$$F(x) + G(x) = f(x) \tag{2.27}$$
$$-\kappa F'(x) + \kappa G'(x) = g(x) \tag{2.28}$$

The second equation can be integrated in x to give

$$-\kappa\left[F(x) - F(x_0)\right] + \kappa\left[G(x) - G(x_0)\right] = \int_{x_0}^{x} g(\alpha)\ \mathrm{d}\alpha \tag{2.29}$$

where x_0 is arbitrary and α an integration variable. We now have two equations in two unknowns, F and G, and the system can be solved to get:

$$F(x) = \frac{f(x)}{2} - \frac{1}{2\kappa}\int_{x_0}^{x} g(\alpha)\ \mathrm{d}\alpha - \frac{F(x_0) - G(x_0)}{2} \tag{2.30}$$
$$G(x) = \frac{f(x)}{2} + \frac{1}{2\kappa}\int_{x_0}^{x} g(\alpha)\ \mathrm{d}\alpha + \frac{F(x_0) - G(x_0)}{2}. \tag{2.31}$$

To obtain the final solution we replace x by $x - \kappa t$ in the expression for F and by $x + \kappa t$ in the expression for G; adding up the resulting expressions the final solution to the PDE takes the form:

$$u(x,t) = \frac{f(x-\kappa t) + f(x+\kappa t)}{2} + \frac{1}{2\kappa}\int_{x-\kappa t}^{x+\kappa t} g(\tau)\ \mathrm{d}\tau \tag{2.32}$$

Figure 2.1 shows the solution of the wave equation for the case where $\kappa = 1$, $g = 0$, and $f(x)$ is a square wave. The time increases from left to right. The succession of plots shows two travelling waves, going in the positive and negative x-direction respectively at the speed $\kappa = 1$. Notice that after the crossing, the two square waves travel with no change of shape.

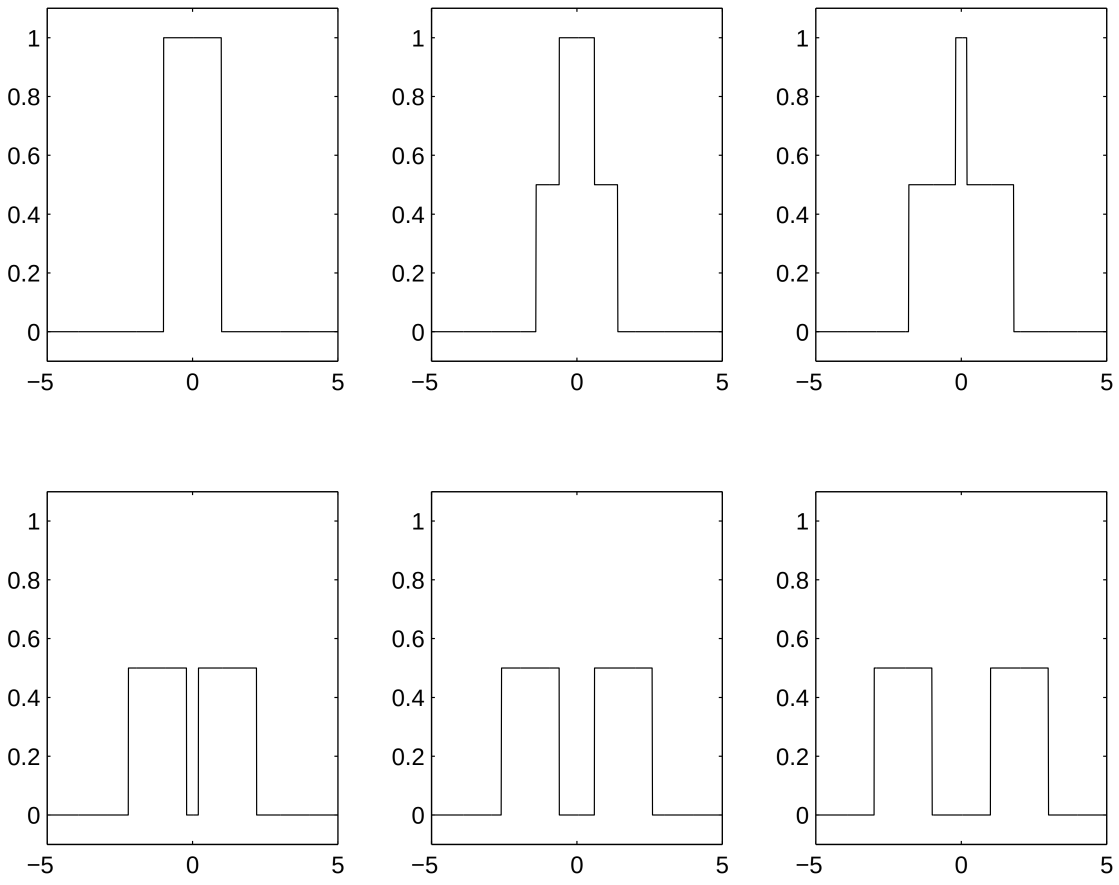


Figure 2.1: Solution to the second order wave equation. The top left figure shows the initial conditions, and the subsequent ones the solution at $t = 0.4$, 0.8, 1.2, 1.6 and 2.0.

2.1.2 Parabolic Equation: $b^2 - 4ac = 0$

If $b^2 - 4ac = 0$ then there is only one double root, and the equation is called *parabolic.* The two characteristic curves then coincide:

$$\left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_\xi = \left.\frac{\mathrm{d}y}{\mathrm{d}x}\right|_\eta = \frac{-b}{2a} \tag{2.33}$$

Since the two characteristic curves coincide, the Jacobian of the mapping $\xi_x\eta_y - \xi_y\eta_x$ vanishes identically. The coefficients of the PDE become $A = B = 0$. The canonical form of the parabolic equation is then

$$Cu_{\eta\eta} + Du_\xi + Eu_\eta + F = 0 \tag{2.34}$$

Example 2 The heat equation is a common example of parabolic equations:

$$u_t = \kappa^2 u_{xx} \tag{2.35}$$

where κ now stands for a diffusion coefficient. The solution in an infinite domain can be obtained by Fourier transforming the PDE to get:

$$\tilde{u}_t = -k^2\kappa^2\tilde{u} \tag{2.36}$$

where $\tilde{u}$ is the Fourier transform of u:

$$\tilde{u}(k,t) = \int_{-\infty}^{\infty} u(x,t)e^{-ikx}\,\mathrm{d}x, \tag{2.37}$$

and k is the wavenumber. The transformed PDE is simply a first order ODE, and can be easily solved with the help of the initial conditions $u(x,0) = u_0(x)$:

$$\tilde{u}(k,t) = \tilde{u}_0 e^{-k^2\kappa^2 t} \tag{2.38}$$

The final solution is obtained by back Fourier transform $u(x,t) = \mathbf{F}^{-1}(\tilde{u})$. The latter can be written as a convolution since the back transforms of $u_0 = \mathbf{F}^{-1}(\tilde{u}_0)$, and $\mathbf{F}^{-1}(e^{-k^2\kappa^2 t}) = \frac{1}{2\kappa\sqrt{\pi t}} e^{\frac{-x^2}{4\kappa^2 t}}$ are known:

$$u(x,t) = \frac{1}{2\kappa\sqrt{\pi t}}\int_{-\infty}^{\infty} u_0(X) e^{\frac{-(x-X)^2}{4\kappa^2 t}}\,\mathrm{d}X \tag{2.39}$$

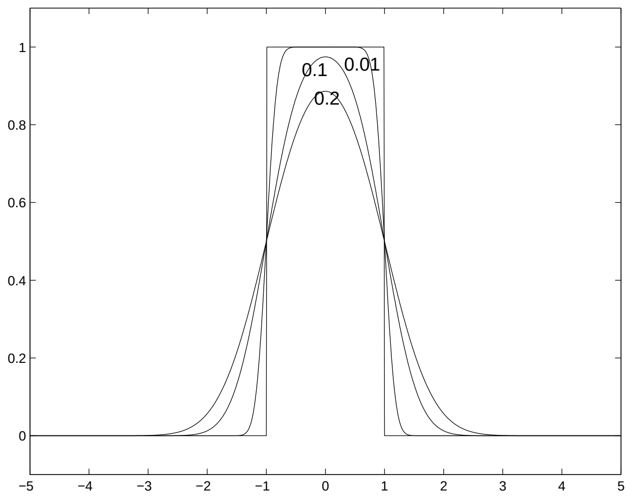


Figure 2.2: Solution to the second order wave equation. The figure shows the initial conditions (the top hat profile), and the solution at times $t = 0.01$, 0.1, 0.2.

As an example we show the solution of the heat equation using the same square initial condition as for the wave equation. The solution can be written in terms of the error function:

$$u(x,t) = \frac{1}{2\kappa\sqrt{\pi t}}\int_{-1}^{1} e^{\frac{-(x-X)^2}{4\kappa^2 t}}\,\mathrm{d}X = \frac{1}{2}\left[\mathrm{erf}\left(\frac{x+1}{2\kappa\sqrt{t}}\right) - \mathrm{erf}\left(\frac{x-1}{2\kappa\sqrt{t}}\right)\right], \tag{2.40}$$

where $\mathrm{erf}(z) = \frac{1}{\sqrt{\pi}}\int_0^z e^{-s^2}\,\mathrm{d}s$. The solution is shown in figure 2.2. Instead of a travelling wave the solution shows a smearing of the two discontinuities as time increases accompanied by a decrease of the maximum amplitude. As its name indicates, the heat equation is an example of a diffusion equation where gradients in the solution tend to be smeared.

2.1.3 Elliptic Equation: $b^2 - 4ac < 0$

If $b^2 - 4ac < 0$ and there is no real roots; the equation is then called *elliptic*. There are no real transformation that can eliminate the second derivatives in ξ and η. However, it is possible to find a transformation that would set $B = 0$ and $A = C = 1$. The canonical form is then:

$$u_{\xi\xi} + u_{\eta\eta} + Du_\xi + Eu_\eta + F = 0 \tag{2.41}$$

Example 3 The Poisson equation in two space dimensions is an example of elliptic PDE:

$$u_{xx} + u_{yy} = 0 \tag{2.42}$$

2.2 Well-Posed Problems

Before attempting to compute numerical a solution to a PDE, we need to find out if its analytical solution makes sense. In other words, we want to find out if enough information is present in the problem formulation to identify the solution. The definition of a well-posed problem addresses this issue. A *well-posed* problem is defined as where the solution satisfies the following properties:

- the solution exists
- the solution is unique
- the solution depends continuously upon the data

Example 4 Consider the system of equations:

$$\left\{\begin{array}{lcl} u_y + v_x &=& 0 \\ v_y + \gamma u_x &=& 0 \end{array}\right. \tag{2.43}$$

The system can be reduced to a second order PDE in 1 variable:

$$\left\{\begin{array}{lcl} u_{yy} - \gamma u_{xx} &=& 0 \\ v_{yy} - \gamma v_{xx} &=& 0 \end{array}\right. \tag{2.44}$$

Clearly, the PDEs are hyperbolic if $\gamma > 0$ and elliptic if $\gamma < 0$. To solve this system of equation we require boundary conditions. To continue with our example, we look for periodic solutions in x and impose the following boundary condition:

$$u(x,0) = \frac{\sin(Nx)}{N^2}, \quad v(x,0) = 0 \tag{2.45}$$

1. **Ellipitic** $\gamma = -\beta^2 < 0$

   The solution for this case is then:

$$u(x,y) = \frac{\sin(Nx)}{N^2}\cosh(\beta N y), \quad v(x,y) = \beta\frac{\cos(Nx)}{N^2}\sinh(\beta N y) \tag{2.46}$$

Notice that even though the PDEs are identical, the two solutions u and v are different because of the boundary conditions. For $N \to \infty$ the boundary conditions become identical, and hence one would expect the solution to become identical. However, it is easy to verify that $|u - v| \to \infty$ for any finite $y > 0$ as $N \to \infty$. Hence small changes in the boundary condition lead to large changes in the solution, and the continuity between the boundary data and the solution has been lost. The problem in this case is that no boundary condition has been imposed on $y \to \infty$ as required by the elliptic nature of the problem.

2. Hyperbolic $\gamma = \beta^2 > 0$
The solution is then

$$u(x,y) = \frac{\sin(Nx)}{N^2}\cos(\beta N y), \quad v(x,y) = -\gamma\frac{\cos(Nx)}{N^2}\sin(\beta N y) \tag{2.47}$$

Notice that in this case $u, v \to 0$ when $N \to \infty$ for any finite value of y.

2.3 First Order Systems

The previous classification considered a single, scalar (1 dependent variable), second order partial differential equation, and two independent variables. In most physical systems there are usually a number of PDEs that must be satisfied simultaneously involving higher order derivative. We must then be able to classify these systems before attempting their solutions. Since systems of PDEs can be recast into first order system, the classification uses the latter approach. For example, the wave equation of the previous section can be cast as two equations in two unknowns:

$$\left\{\begin{array}{lcl} v_t &=& \kappa\eta_x \\ \eta_t &=& \kappa v_x \end{array}\right. \quad \longleftrightarrow \quad \frac{\partial}{\partial t}\begin{pmatrix} v \\ \eta \end{pmatrix} = \begin{pmatrix} 0 & \kappa \\ \kappa & 0 \end{pmatrix}\frac{\partial}{\partial x}\begin{pmatrix} v \\ \eta \end{pmatrix} \tag{2.48}$$

where we have defined $\eta = \kappa u_x$, and $v = u_t$. Note that it is possible to consider v and η as the component of a vector unknown $\mathbf{w}$ and to write the equations in vector notation as shown in equation (2.48). In the next few section we will look primarily on the condition under which a first order system is hyperbolic.

2.3.1 Scalar Equation

A typical equation of this sort is the advection equation:

$$\begin{array}{l} u_t + c u_x = 0, \ 0 \le x \le L \\ u(x, t=0) = u_0(x), \quad u(x=0,t) = u_l(t) \end{array} \tag{2.49}$$

where c is the advecting velocity. Let us define $c = \mathrm{d}x/\mathrm{d}t$, so that the equation becomes:

$$\frac{\partial u}{\partial t}\,\mathrm{d}t + \frac{\partial u}{\partial x}\,\mathrm{d}x = \ \mathrm{d}u = 0 \tag{2.50}$$

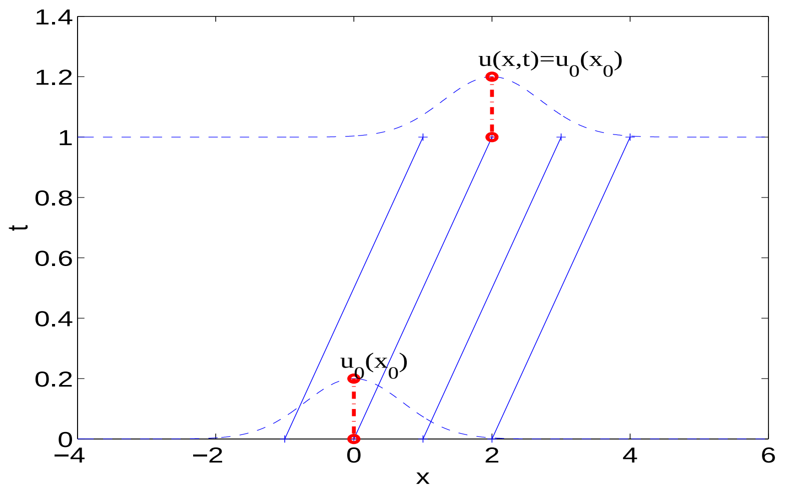


Figure 2.3: Characteristic lines for the linear advection equation. The solid lines are the characteristics emanating from different locations on the initial axis. The dashed line represents the signal at time $t = 0$ and $t = 1$. If the solution at (x, t) is desired, we first need to find the foot of the characteristic $x_0 = x - ct$, and the value of the function there at the initial time is copied to time t.

where du is the total differential of the function u. Since the right hand side is zero, then the function must be constant along the lines $\mathrm{d}x/\mathrm{d}t = c$, and this constant must be equal to the value of u at the initial time. The solution can then written as:

$$u(x, t) = u_0(x_0) \text{ along } \frac{\mathrm{d}x}{\mathrm{d}t} = c \tag{2.51}$$

where x_0 is the location of the foot of the characteristic, the intersection of the characteristic with the $t = 0$ line. The simplest way to visualize this picture is to consider the case where c is constant. The characteristic lines can then be obtained analytically: they are straight lines given by $x = x_0 + ct$. A family of characteristic lines are shown in figure 2.3.1 where c is assumed positive. In this example the information is travelling from left to right at the constant speed c, and the initial hump translates to the right without change of shape.

If the domain is of finite extent, say $0 \leq x \leq L$, and the characteristic intersects the line $x = 0$ (assuming $c > 0$), then a boundary condition is needed on the left boundary to provide the information needed to determine the solution uniquely. That is we need to provide the variation of u at the "inlet" boundary in the form:$u(x = 0, t) = g(t)$. The solution now can be written as:

$$u(x, t) = \begin{cases} u_0(x - ct) & \text{for} \quad x - ct > 0 \\ g(t - x/c) & \text{for} \quad x - ct < 0 \end{cases} \tag{2.52}$$

Not that since the information travels from left to right, the boundary condition is needed at the left boundary and not on the right. The solution would be impossible to determine had the boundary conditions been given on the right boundary $x = L$,

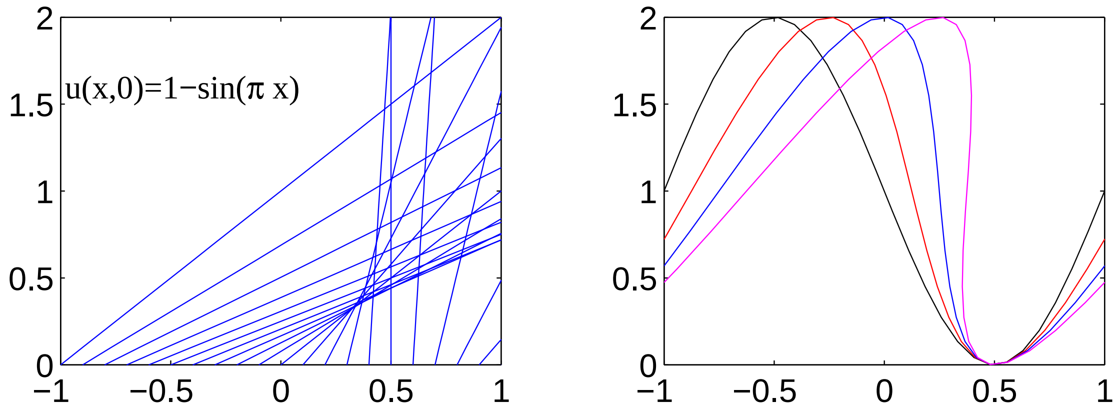


Figure 2.4: Characteristics for Burgers' equation (left panel), and the solution (right panel) at different times for a periodic problem. The black line is the initial condition, the red line the solution at $t = 1/8$, the blue at $t = 1/4$, and the magenta at $t = 3/4$. The solution become discontinuous when characteristics intersects.

the problem would then be ill-posed for lack of proper boundary conditions. The right boundary maybe called an "outlet" boundary since information is leaving the domain. No boundary condition can be imposed there since the solution is determined from "upstream" information. In the case where $c < 0$ the information travels from right to left, and the boundary condition must be imposed at $x = L$.

If the advection speed c varies in x and t, then the characteristics are not necessarily straight lines of constant slope, but are in general curves. Since the slopes of the curves vary, characteristic lines may intersects. These intersection points are places where the solution is discontinuous with sharp jumps. At these locations the slopes are infinite and space and time-derivative become meaningless, i.e. the PDE is not valid anymore. This breakdown occurs usually because of the neglect of important physical terms, such as dissipative terms, that act to prevent true discontinuous solutions.

An example of an equation that can lead to discontinuous solutions is the Burger equation:

$$u_t + uu_x = 0 \tag{2.53}$$

where $c = u$. This equation is nonlinear with a variable advection speed that depend on the solution. The characteristics are given by the lines:

$$\frac{\mathrm{d}x}{\mathrm{d}t} = u \tag{2.54}$$

along which the PDE takes the form $du = 0$. Hence u is constant along characteristics, which means that their slopes are also constant according to equation (2.54), and hence must be straightlines. Even in this nonlinear case the characteristics are straightlines but with varying slopes. The behavior of the solution can become quite complicated as characteristic lines intersect as shown in figure 2.4. The solution of hyperbolic equations in the presence of discontinuities can become quite complicated. We refer the interested reader to Whitham (1974); Durran (1999) for further discussions.

2.3.2 System of Equations in one-space dimension

A system of PDE in one-space dimension of the form

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{w}}{\partial x} = 0 \tag{2.55}$$

where $\mathbf{A}$ is the so called Jacobian matrix is said to be hyperbolic if the matrix $\mathbf{A}$ has a complete set of real eigenvalues. For then one can find a bounded matrix $\mathbf{T}$ whose columns are the eigenvectors of $\mathbf{A}$, such that the matrix $\mathbf{D} = \mathbf{T}^{-1}\mathbf{AT}$ is diagonal. Reminder: A diagonal matrix is one where all entries are zero save for the ones on the main diagonal, and in the case above the diagonal entries are the eigenvalues of the matrix. The system can then be uncoupled by defining the auxiliary variables $\mathbf{w} = \mathbf{T}\hat{\mathbf{w}}$, replacing in the original equation we get the equations

$$\frac{\partial \hat{\mathbf{w}}}{\partial t} + \mathbf{D}\frac{\partial \hat{\mathbf{w}}}{\partial x} = 0 \longleftrightarrow \frac{\partial \hat{w}_i}{\partial t} + \lambda_i \frac{\partial \hat{w}_i}{\partial x} = 0 \tag{2.56}$$

The equation on the left side is written in the vector form whereas the component form on the right shows the uncoupling of the equations. The component form clearly shows how the sytem is hyperbolic and analogous to the scalar form.

Example 5 The linearized equation governing tidal flow in a channel of constant cross section and depth are the shallow water equations:

$$\frac{\partial}{\partial t}\begin{pmatrix} u \\ \eta \end{pmatrix} + \begin{pmatrix} 0 & g \\ h & 0 \end{pmatrix}\frac{\partial}{\partial x}\begin{pmatrix} u \\ \eta \end{pmatrix} = 0, \quad \mathbf{A} = \begin{pmatrix} 0 & g \\ h & 0 \end{pmatrix} \tag{2.57}$$

where u and η are the unknown velocity and surface elevation, g is the gravitational acceleration, and h the water depth. The eigenvalues of the matrix $\mathbf{A}$ can be found by solving the equation:

$$\det \mathbf{A} = 0 \Leftrightarrow \det \begin{vmatrix} -\lambda & g \\ h & -\lambda \end{vmatrix} = \lambda^2 - gh = 0. \tag{2.58}$$

The two *real* roots of the equations are $\lambda = \pm c$, where $c = \sqrt{gh}$. Hence the eigenvalues are the familiar gravity wave speed. Two eigenvectors of the system are $\mathbf{u}_1$ and $\mathbf{u}_2$ corresponding to the positive and negative roots, respectively:

$$\mathbf{u}_1 = \begin{pmatrix} 1 \\ \dfrac{c}{g} \end{pmatrix}, \quad \mathbf{u}_2 = \begin{pmatrix} 1 \\ -\dfrac{c}{g} \end{pmatrix}. \tag{2.59}$$

The eigenvectors are the columns of the transformation matrix $\mathbf{T}$, and we have

$$\mathbf{T} = \begin{pmatrix} 1 & 1 \\ \frac{c}{g} & -\frac{c}{g} \end{pmatrix}, \quad \mathbf{T}^{-1} = \frac{1}{2}\begin{pmatrix} 1 & \frac{g}{c} \\ 1 & \frac{-g}{c} \end{pmatrix}. \tag{2.60}$$

It it easy to verify that

$$\mathbf{D} = \mathbf{T}^{-1}\mathbf{AT} = \begin{pmatrix} c & 0 \\ 0 & -c \end{pmatrix}, \tag{2.61}$$

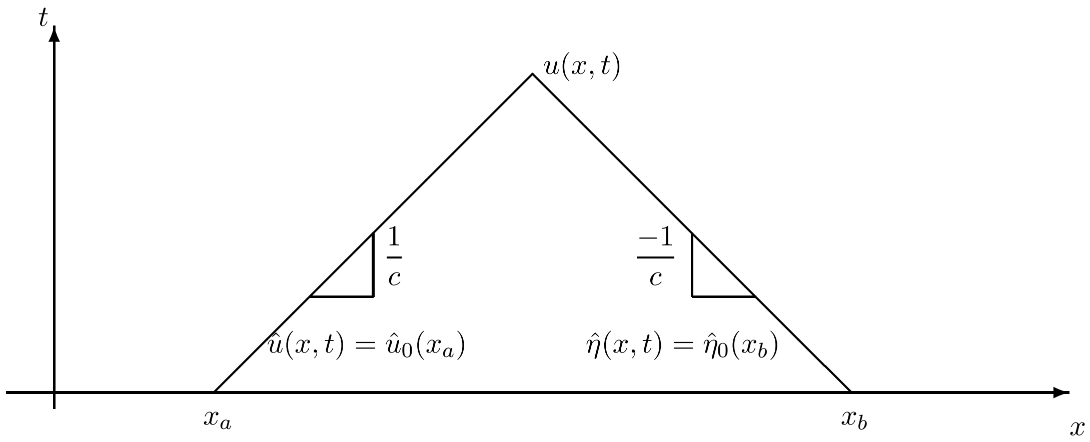

t
$u(x,t)$
$\frac{1}{c}$
$\frac{-1}{c}$
$\hat{u}(x,t) = \hat{u}_0(x_a)$
$\hat{\eta}(x,t) = \hat{\eta}_0(x_b)$
x_a
x_b
x

Figure 2.5: Characteristics for the one-dimensional tidal equations. The new variables $\hat{u}$ and $\hat{\eta}$ are constant along the right, and left going characteristic, respectively. The solution at the point (x, t) can be computed by finding the foot of two characteristic curves at the initial time, x_a and x_b and applying the conservation of $\hat{u}$ and $\hat{\eta}$.

as expected. The new variable are given by

$$\begin{pmatrix} \hat{u} \\ \hat{\eta} \end{pmatrix} = \mathbf{T}^{-1} \begin{pmatrix} u \\ \eta \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 1 & \frac{g}{c} \\ 1 & \frac{-g}{c} \end{pmatrix} \begin{pmatrix} u \\ \eta \end{pmatrix} = \begin{pmatrix} \frac{1}{2}\left(u + \frac{g\eta}{c}\right) \\ \frac{1}{2}\left(u - \frac{g\eta}{c}\right) \end{pmatrix} \tag{2.62}$$

The equations written in the new variables take the component form:

$$\left\{ \begin{array}{rcl} \frac{\partial \hat{u}}{\partial t} + c\frac{\partial \hat{u}}{\partial x} & = & 0 \\ \frac{\partial \hat{\eta}}{\partial t} - c\frac{\partial \hat{\eta}}{\partial x} & = & 0 \end{array} \right. \longleftrightarrow \left\{ \begin{array}{ll} \hat{u} = \text{constant along} & \frac{\mathrm{d}x}{\mathrm{d}t} = c \\ \hat{\eta} = \text{constant along} & \frac{\mathrm{d}x}{\mathrm{d}t} = -c \end{array} \right. \tag{2.63}$$

To illustrate how the characteristic information can be used to determine the solution at any time, given the initial conditions, we consider the case of an infinite domain, and calculate the intersection of the two characteristic curves meeting at (x, t) with the axis $t = 0$. If we label the two coordinate x_a and x_b as shown in the figure, then we can set up the two equations:

$$\left\{ \begin{array}{rcl} u + \frac{g\eta}{c} & = & u_a + \frac{g\eta_a}{c} \\ u - \frac{g\eta}{c} & = & u_b - \frac{g\eta_b}{c} \end{array} \right. \longleftrightarrow \left\{ \begin{array}{rcl} u & = & \frac{u_a + u_b}{2} + g\frac{\eta_a - \eta_b}{2c} \\ \eta & = & c\frac{u_a - u_b}{2g} + \frac{\eta_a + \eta_b}{2} \end{array} \right. \tag{2.64}$$

2.3.3 System of equations in multi-space dimensions

A system of PDE in two-space dimension can be written in the form:

$$\frac{\partial \mathbf{w}}{\partial t} + \mathbf{A}\frac{\partial \mathbf{w}}{\partial x} + \mathbf{B}\frac{\partial \mathbf{w}}{\partial y} = 0 \tag{2.65}$$

In general it is not possible to define a similarity transformation that would diagonalize all matrices at once. To extend the definition of hyperbolicity to multi-space dimension, the following procedure can be used for the case where **A** and **B** are constant matrices. First we define the Fourier transform of the dependent variables with respect to x, y and t:

$$\mathbf{w} = \hat{\mathbf{w}} e^{i(kx+ly-\omega t)} \tag{2.66}$$

where $\hat{\mathbf{w}}$ can be interpreted as the vector of Fourier amplitudes, and k and l are wavelength in the x and y direction respectively, and ω is the frequency. This Fourier representation is then substitute it in the partial differential equation to obtain:

$$\left[k\mathbf{A} + l\mathbf{B} - \omega I\right] \hat{\mathbf{w}} = 0, \tag{2.67}$$

where **I** is the identity matrix. Equation (2.67) has the form of an eigenvalue problem, where ω represent the eigenvalues of the matrix $k\mathbf{A} + l\mathbf{B}$. The system is classified as hyperbolic if and only if the eigenvalues ω are real for real choices of the wavenumber vector (k, l). The extension above hinges on the matrices **A** and **B** being constant in space and time; in spite of its limitation it does show intuitively how the general behavior of wave-like solution can be extended to multiple spatial dimensions.

For the case where the matrices are not constant, the definition can be extended by requiring that the exitence of bounded matrices **T** such that the matrix $\mathbf{T}^{-1}(k\mathbf{A} + l\mathbf{B})\mathbf{T}$ is a diagonal matrix with real eigenvalues for all points within an neighborhood of (x, y).

Chapter 3

Finite Difference Approximation of Derivatives

3.1 Introduction

The standard definition of derivative in elementary calculus is the following

$$u'(x) = \lim_{\Delta x \to 0} \frac{u(x + \Delta x) - u(x)}{\Delta x} \tag{3.1}$$

Computers however cannot deal with the limit of $\Delta x \to 0$, and hence a *discrete* analogue of the continuous case need to be adopted. In a discretization step, the set of points on which the function is defined is finite, and the function value is available on a discrete set of points. Approximations to the derivative will have to come from this discrete table of the function.

Figure 3.1 shows the discrete set of points x_i where the function is known. We will use the notation $u_i = u(x_i)$ to denote the value of the function at the i-th node of the computational grid. The nodes divide the axis into a set of intervals of width $\Delta x_i = x_{i+1} - x_i$. When the grid spacing is fixed, i.e. all intervals are of equal size, we will refer to the grid spacing as Δx. There are definite advantages to a constant grid spacing as we will see later.

3.2 Finite Difference Approximation

The definition of the derivative in the continuum can be used to approximate the derivative in the discrete case:

$$u'(x_i) \approx \frac{u(x_i + \Delta x) - u(x_i)}{\Delta x} = \frac{u_{i+1} - u_i}{\Delta x} \tag{3.2}$$

where now Δx is finite and small but not necessarily infinitesimally small, i.e. . This is known as a *forward Euler* approximation since it uses forward differencing. Intuitively, the approximation will improve, i.e. the error will be smaller, as Δx is made smaller.

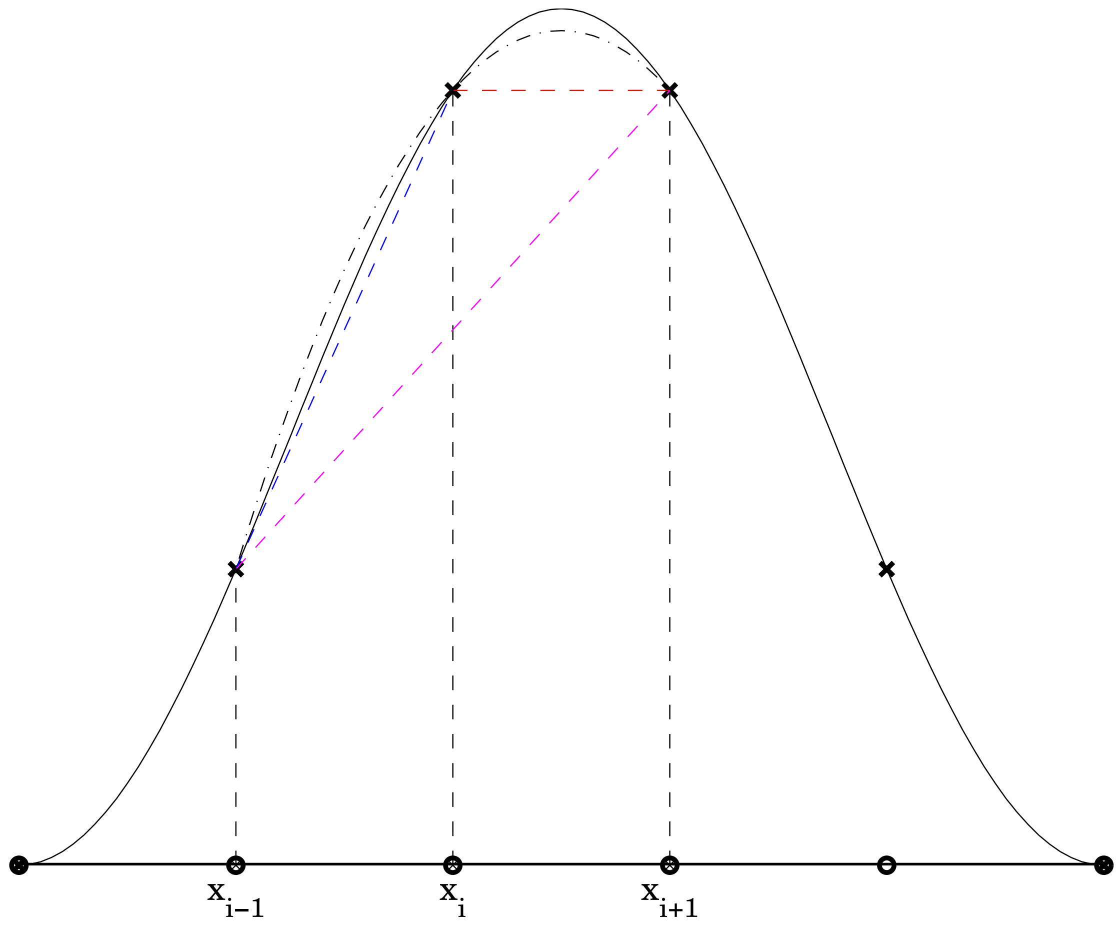


Figure 3.1: Computational grid and example of backward, forward, and central approximation to the derivative at point x_i. The dash-dot line shows the centered parabolic interpolation, while the dashed line show the backward (blue), forward (red) and centered (magenta) linear interpolation to the function.

The above is not the only approximation possible, two equally valid approximations are:
backward Euler:

$$u'(x_i) \approx \frac{u(x_i) - u(x_i - \Delta x)}{\Delta x} = \frac{u_i - u_{i-1}}{\Delta x} \tag{3.3}$$

Centered Difference

$$u'(x_i) \approx \frac{u(x_i + \Delta x) - u(x_i - \Delta x)}{2\Delta x} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} \tag{3.4}$$

All these definitions are equivalent in the continuum but lead to different approximations in the discrete case. The question becomes which one is better, and is there a way to quantify the error committed. The answer lies in the application of Taylor series analysis. We briefly describe Taylor series in the next section, before applying them to investigate the approximation errors of finite difference formulae.

3.3 Taylor series

Starting with the identity:

$$u(x) = u(x_i) + \int_{x_i}^{x} u'(s)\ \mathrm{d}s \tag{3.5}$$

Since $u(x)$ is arbitrary, the formula should hold with $u(x)$ replaced by $u'(x)$, i.e.,

$$u'(x) = u'(x_i) + \int_{x_i}^{x} u''(s)\ \mathrm{d}s \tag{3.6}$$

Replacing this expression in the original formula and carrying out the integration (since $u(x_i)$ is constant) we get:

$$u(x) = u(x_i) + (x - x_i)u'(x_i) + \int_{x_i}^{x}\int_{x_i}^{x} u''(s)\ \mathrm{d}s\ \mathrm{d}s \tag{3.7}$$

The process can be repeated with

$$u''(x) = u''(x_i) + \int_{x_i}^{x} u'''(s)\ \mathrm{d}s \tag{3.8}$$

to get:

$$u(x) = u(x_i) + (x - x_i)u'(x_i) + \frac{(x - x_i)^2}{2!}u''(x_i) + \int_{x_i}^{x}\int_{x_i}^{x}\int_{x_i}^{x} u'''(s)\ \mathrm{d}s\ \mathrm{d}s\ \mathrm{d}s \tag{3.9}$$

This process can be repeated under the assumption that $u(x)$ is sufficiently differentiable, and we find:

$$u(x) = u(x_i) + (x - x_i)u'(x_i) + \frac{(x - x_i)^2}{2!}u''(x_i) + \cdots + \frac{(x - x_i)^n}{n!}u^(n)(x_i) + R_{n+1} \tag{3.10}$$

where the remainder is given by:

$$R_{n+1} = \int_{x_i}^{x}\cdots\int_{x_i}^{x} u^{(n+1)}(x)(\ \mathrm{d}s)^{n+1} \tag{3.11}$$

Equation 3.10 is known as the Taylor series of the function $u(x)$ about the point x_i. Notice that the series is a polynomial in $(x - x_i)$ (the signed distance of x to x_i), and the coefficients are the (scaled) derivatives of the function *evaluated* at x_i.

If the $(n + 1)$-th derivative of the function u has minimum m and maximum M over the interval $[x_i x]$ then we can write:

$$\int_{x_i}^{x}\cdots\int_{x_i}^{x} m(\ \mathrm{d}s)^{n+1} \quad \le R_{n+1} \le \quad \int_{x_i}^{x}\cdots\int_{x_i}^{x} M(\ \mathrm{d}s)^{n+1} \tag{3.12}$$

$$m\frac{(x - x_i)^{n+1}}{(n + 1)!} \quad \le R_{n+1} \le \quad M\frac{(x - x_i)^{n+1}}{(n + 1)!} \tag{3.13}$$

which shows that the remainder is bounded by the values of the derivative and the distance of the point x to the expansion point x_i raised to the power $(n + 1)$. If we further assume that $u^{(n+1)}$ is continuous then it must take all values between m and M that is

$$R_{n+1} = u^{(n+1)}(\xi)\frac{(x - x_i)^{n+1}}{(n + 1)!} \tag{3.14}$$

for some ξ in the interval $[x_i x]$.

3.3.1 Taylor series and finite differences

Taylor series have been widely used to study the behavior of numerical approximation to differential equations. Let us investigate the forward Euler with Taylor series. To do so, we expand the function u at x_{i+1} about the point x_i:

$$u(x_i + \Delta x_i) = u(x_i) + \Delta x_i \left.\frac{\partial u}{\partial x}\right|_{x_i} + \frac{\Delta x_i^2}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} + \frac{\Delta x_i^3}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \ldots \tag{3.15}$$

The Taylor series can be rearranged to read as follows:

$$\frac{u(x_i + \Delta x_i) - u(x_i)}{\Delta x_i} - \left.\frac{\partial u}{\partial x}\right|_{x_i} = \underbrace{\frac{\Delta x_i}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} + \frac{\Delta x_i^2}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \ldots}_{\text{Truncation Error}} \tag{3.16}$$

where it is now clear that the forward Euler formula (3.2) corresponds to truncating the Taylor series after the second term. The right hand side of equation (3.16) is the error committed in terminating the series and is referred to as the **truncation error**. The tuncation error can be defined as the difference between the partial derivative and its finite difference representation. For sufficiently smooth functions, i.e. ones that possess continuous higher order derivatives, and sufficiently small Δx_i, the first term in the series can be used to characterize the order of magnitude of the error. The first term in the truncation error is the product of the second derivative evaluated at x_i and the grid spacing Δx_i: the former is a property of the function itself while the latter is a numerical parameter which can be changed. Thus, for finite $\frac{\partial^2 u}{\partial x^2}$, the numerical approximation depends lineraly on the parameter Δx_i. If we were to half Δx_i we ought to expect a linear decrease in the error for sufficiently small Δx_i. We will use the "big Oh" notation to refer to this behavior so that T.E. $\sim O(\Delta x_i)$. In general if Δx_i is not constant we pick a representative value of the grid spacing, either the average of the largest grid spacing. Note that in general the exact truncation error is not known, and all we can do is characterize the behavior of the error as $\Delta x \to 0$. So now we can write:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{u_{i+1} - u_i}{\Delta x_i} + O(\Delta x) \tag{3.17}$$

The taylor series expansion can be used to get an expression for the truncation error of the backward difference formula:

$$u(x_i - \Delta x_{i-1}) = u(x_i) - \Delta x_{i-1}\left.\frac{\partial u}{\partial x}\right|_{x_i} + \frac{\Delta x_{i-1}^2}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} - \frac{\Delta x_{i-1}^3}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \ldots \tag{3.18}$$

where $\Delta x_{i-1} = x_i - x_{i-1}$. We can now get an expression for the error corresponding to backward difference approximation of the first derivative:

$$\frac{u(x_i) - u(x_i - \Delta x_{i-1})}{\Delta x_{i-1}} - \left.\frac{\partial u}{\partial x}\right|_{x_i} = \underbrace{-\frac{\Delta x_{i-1}}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_{x_i} + \frac{\Delta x_{i-1}^2}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \ldots}_{\text{Truncation Error}} \tag{3.19}$$

It is now clear that the truncation error of the backward difference, while not the same as the forward difference, behave similarly in terms of order of magnitude analysis, and is linear in Δx:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{u_i - u_{i-1}}{\Delta x_i} + O(\Delta x) \tag{3.20}$$

Notice that in both cases we have used the information provided at just two points to derive the approximation, and the error behaves linearly in both instances.

Higher order approximation of the first derivative can be obtained by combining the two Taylor series equation (3.15) and (3.18). Notice first that the high order derivatives of the function u are all evaluated at the same point x_i, and are the same in both expansions. We can now form a linear combination of the equations whereby the leading error term is made to vanish. In the present case this can be done by inspection of equations (3.16) and (3.19). Multiplying the first by Δx_{i-1} and the second by Δx_i and adding both equations we get:

$$\frac{1}{\Delta x_i + \Delta x_{i-1}}\left[\Delta x_{i-1}\frac{u_{i+1}-u_i}{\Delta x_i} + \Delta x_i \frac{u_i - u_{i-1}}{\Delta x_{i-1}}\right] - \left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{\Delta x_{i-1}\Delta x_i}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \ldots \tag{3.21}$$

There are several points to note about the preceding expression. First the approximation uses information about the functions u at three points: x_{i-1}, x_i and x_{i+1}. Second the truncation error is T.E. $\sim O(\Delta x_{i-1}\Delta x_i)$ and is second order, that is if the grid spacing is decreased by 1/2, the T.E. error decreases by factor of 2^2. Thirdly, the previous point can be made clearer by focussing on the important case where the grid spacing is constant: $\Delta x_{i-1} = \Delta x_i = \Delta x$, the expression simplifies to:

$$\frac{u_{i+1} - u_{i-1}}{2\Delta x} - \left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{\Delta x^2}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_{x_i} + \ldots \tag{3.22}$$

Hence, for an equally spaced grid the centered difference approximation converges quadratically as $\Delta x \rightarrow 0$:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} + O(\Delta x^2) \tag{3.23}$$

Note that like the forward and backward Euler difference formula, the centered difference uses information at only two points but delivers twice the order of the other two methods. This property will hold in general whenever the grid spacing is constant and the computational **stencil**, i.e. the set of points used in approximating the derivative, is symmetric.

3.3.2 Higher order approximation

The Taylor expansion provides a very useful tool for the derivation of higher order approximation to derivatives of any order. There are several approaches to achieve this. We will first look at an expendient one before elaborating on the more systematic one. In most of the following we will assume the grid spacing to be constant as is usually the case in most applications.

Equation (3.22) provides us with the simplest way to derive a fourth order approximation. An important property of this centered formula is that its truncation error contains only odd derivative terms:

$$\frac{u_{i+1}-u_{i-1}}{2\Delta x} = \frac{\partial u}{\partial x} + \frac{\Delta x^2}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{\Delta x^4}{5!}\frac{\partial^5 u}{\partial x^5} + \frac{\Delta x^6}{7!}\frac{\partial^7 u}{\partial x^7} + \ldots + \frac{\Delta x^{2m}}{(2m+1)!}\frac{\partial^{(2m+1)} u}{\partial x^{(2m+1)}} + \ldots \tag{3.24}$$

The above formula can be applied with Δx replace by $2\Delta x$, and $3\Delta x$ respectively to get:

$$\frac{u_{i+2}-u_{i-2}}{4\Delta x} = \frac{\partial u}{\partial x} + \frac{(2\Delta x)^2}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{(2\Delta x)^4}{5!}\frac{\partial^5 u}{\partial x^5} + \frac{(2\Delta x)^6}{7!}\frac{\partial^7 u}{\partial x^7} + O(\Delta x^8) \tag{3.25}$$

$$\frac{u_{i+3}-u_{i-3}}{6\Delta x} = \frac{\partial u}{\partial x} + \frac{(3\Delta x)^2}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{(3\Delta x)^4}{5!}\frac{\partial^5 u}{\partial x^5} + \frac{(3\Delta x)^6}{7!}\frac{\partial^7 u}{\partial x^7} + O(\Delta x^8) \tag{3.26}$$

It is now clear how to combine the different estimates to obtain a fourth order approximation to the first derivative. Multiplying equation (3.24) by 2^2 and substracting it from equation (3.25), we cancel the second order error term to get:

$$\frac{8(u_{i+1}-u_{i-1})-(u_{i+2}-u_{i-2})}{12\Delta x} = \frac{\partial u}{\partial x} - \frac{4\Delta x^4}{5!}\frac{\partial^5 u}{\partial x^5} - \frac{20\Delta x^6}{7!}\frac{\partial^7 u}{\partial x^7} + O(\Delta x^8) \tag{3.27}$$

Repeating the process for equation but using the factor 3^2 and substracting it from equation (3.26), we get

$$\frac{27(u_{i+1}-u_{i-1})-(u_{i+3}-u_{i-3})}{48\Delta x} = \frac{\partial u}{\partial x} - \frac{9\Delta x^4}{5!}\frac{\partial^5 u}{\partial x^5} - \frac{90\Delta x^6}{7!}\frac{\partial^7 u}{\partial x^7} + O(\Delta x^8) \tag{3.28}$$

Although both equations (3.27) and (3.28) are valid, the latter is not used in practice since it does not make sense to disregard neighboring points while using more distant ones. However, the expression is useful to derive a sixth order approximation to the first derivative: multiply equation (3.28) by 9 and equation (3.28) by 4 and substract to get:

$$\frac{45(u_{i+1}-u_{i-1})-9(u_{i+2}-u_{i-2})+(u_{i+3}-u_{i-3})}{60\Delta x} = \frac{\partial u}{\partial x} + \frac{36\Delta x^6}{7!}\frac{\partial^7 u}{\partial x^7} + O(\Delta x^8) \tag{3.29}$$

The process can be repeated to derive higher order approximations.

3.3.3 Remarks

The validity of the Taylor series analysis of the truncation error depends on the existence of higher order derivatives. If these derivatives do not exist, then the higher order approximations cannot be expected to hold. To demonstrate the issue more clearly we will look at specific examples.

Example 6 The function $u(x) = \sin \pi x$ is infinitely smooth and differentiable, and its first derivative is given by $u_x = \pi \cos \pi x$. Given the smoothness of the function we expect the Taylor series analysis of the truncation error to hold. We set about verifying this claim in a practical calculation. We lay down a computational grid on the interval $-1 \le x \le 1$ of constant grid spacing $\Delta x = 2/M$. The approximation points are then $x_i = i\Delta x - 1$,

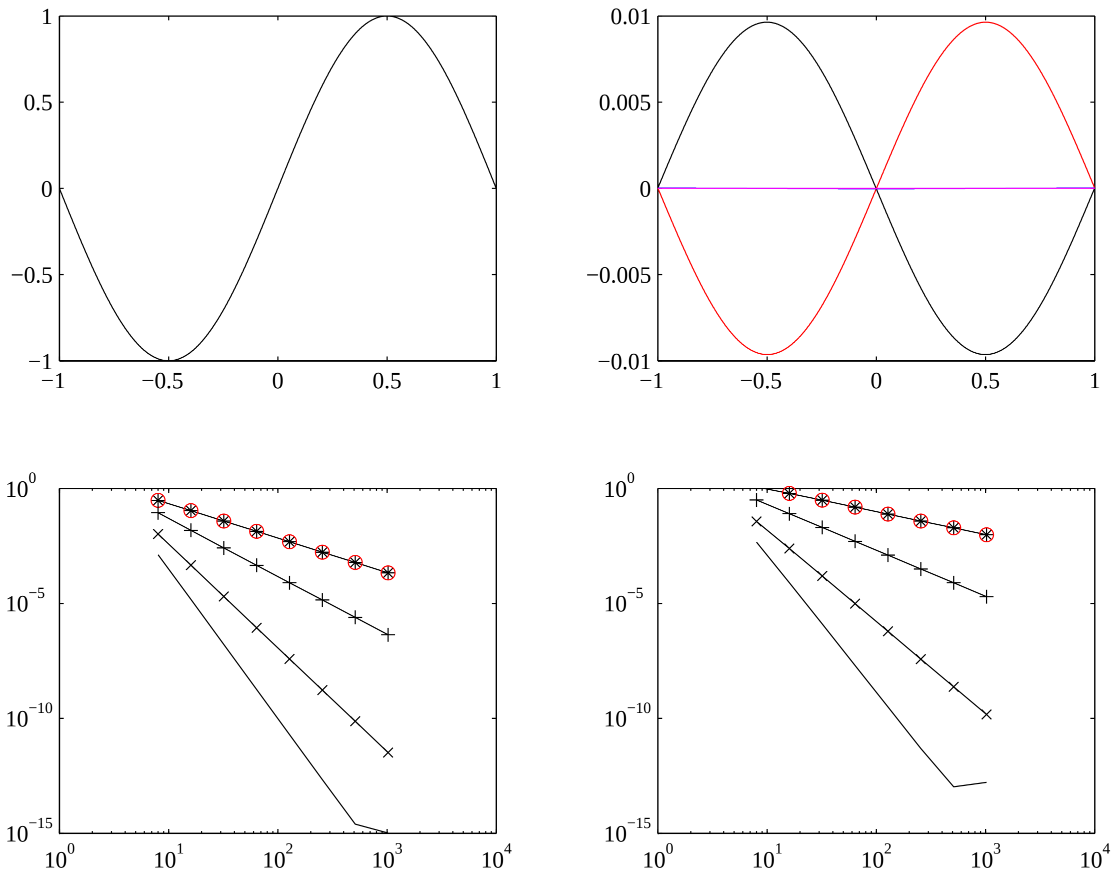


Figure 3.2: Finite difference approximation to the derivative of the function $\sin \pi x$. The top left panel shows the function as a function of x. The top right panel shows the spatial distribution of the error using the Forward difference (black line), the backward difference (red line), and the centered differences of various order (magenta lines) for the case $M = 1024$; the centered difference curves lie atop each other because their errors are much smaller then those of the first order schemes. The lower panels are convergence curves showing the rate of decrease of the rms and maximum errors as the number of grid cells increases.

$i = 0, 1, \ldots, M$. Let ϵ be the error between the finite difference approximation to the first derivative, $\tilde{u}_x$, and its analytical derivative u_x:

$$\epsilon_i = \tilde{u}_x(x_i) - u_x(x_i) \tag{3.30}$$

The numerical approximation $\tilde{u}_x$ will be computed using the forward difference, equation (3.17), the backward difference, equation (3.20), and the centered difference approximations of order 2, 4 and 6, equations (3.22), (3.27, and (3.29). We will use two measures to characterize the error ϵ_i, and to measure its rate of decrease as the number of grid points is increased. One is a bulk measure and consists of the root mean square error, and the other one consists of the maximum error magnitude. We will use the following notations for the rms and max errors:

$$\|\epsilon\|_2 = \Delta x \left(\sum_{i=0}^{M} \epsilon_i^2 \right)^{\frac{1}{2}} \tag{3.31}$$

$$\|\epsilon\|_\infty = \max_{0 \le i \le M} (|\epsilon_i|) \tag{3.32}$$

The right panel of figure 3.2 shows the variations of ϵ as a function of x for the case $M = 1024$ for several finite difference approximations to u_x. For the first order schemes the errors peak at $\pm 1/2$ and reaches 0.01. The error is much smaller for the higher order centered difference scheme. The lower panels of figure 3.2 show the decrease of the rms error ($\|\epsilon\|_2$ on the left), and maximum error ($\|\epsilon\|_\infty$ on the right) as a function of the number of cells M. It is seen that the convergence **rate** increases with an increase in the order of the approximation as predicted by the Taylor series analysis. The slopes on this log-log plot are -1 for forward and backward difference, and -2, -4 and -6 for the centered difference schemes of order 2, 4 and 6, respectively. Notice that the maximum error decreases at the same rate as the rms error even though it reports a higher error. Finally, if one were to gauge the efficiency of using information most accurately, it is evident that for a given M, the high order methods achieve the lowest error.

Example 7 We now investigate the numerical approximation to a function with finite differentiability, more precisely, one that has a discontinuous third derivative. This function is defined as follows:

	$u(x)$	$u_x(x)$	$u_{xx}(x)$	u_{xxx}
$x < 0$	$\sin \pi x$	$\pi \cos \pi x$	$-\pi^2 \sin \pi x$	$-\pi^3 \cos \pi x$
$0 < x$	$\pi x e^{-x^2}$	$\pi(1 - 2x^2)e^{-x^2}$	$2\pi x(2x^2 - 3)e^{-x^2}$	$-2\pi(3 - 12x^2 + 4x^4)e^{-x^2}$
$x = 0$	0	π	0	$-\pi^3, -6\pi$

Notice that the function and its first two derivatives are continuous at $x = 0$, but the third derivative is discontinuous. An examination of the graph of the function in figure 3.3 shows a curve, at least visually (the so called eye-ball norm). The error distribution is shown in the top right panel of figure 3.3 for the case $M = 1024$ and the fourth order centered difference scheme. Notice that the error is very small except for the spike near the discontinuity. The error curves (in the lower panels) show that the second order centered difference converges faster then the forward and backward Euler scheme, but

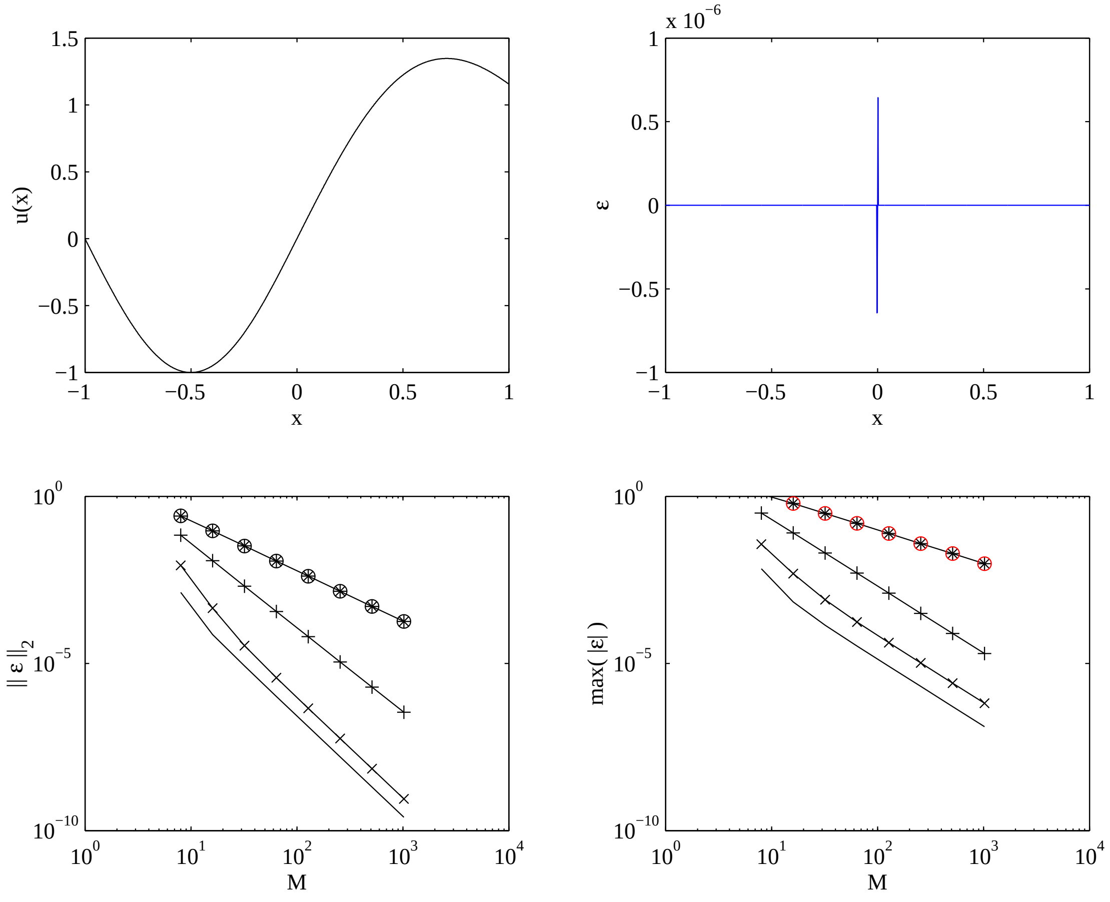


Figure 3.3: Finite difference approximation to the derivative of a function with discontinuous third derivative. The top left panel shows the function $u(x)$ which, to the eyeball norm, appears to be quite smooth. The top right panel shows the spatial distribution of the error ($M = 1024$) using the fourth order centered difference: notice the spike at the discontinuity in the derivative. The lower panels are convergence curves showing the rate of decrease of the rms and maximum errors as the number of grid cells increases.

that the convergence rates of the fourth and sixth order centered schemes are no better then that of the second order one. This is a direct consequence of the discontinuity in the third derivative whereby the Taylor expansion is valid only up to the third term. The effects of the discontinuity are more clearly seen in the maximum error plot (lower right panel) then in the mean error one (lower left panel). The main message of this example is that for functions with a finite number of derivatives, the Taylor series prediction for the high order schemes does not hold. Notice that the error for the fourth and sixth order schemes are lower then the other 3, but their **rate** of convergence is the same as the second order scheme. This is largely coincidental and would change according to the function.

3.3.4 Systematic Derivation of higher order derivative

The Taylor series expansion provides a systematic way of deriving approximation to higher order derivatives of any order (provided of course that the function is smooth enough). Here we assume that the grid spacing is uniform for simplicity. Suppose that the stencil chosen includes the points: x_j such that $i-l \leq j \leq i+r$. There are thus l points to the left and r points to the right of the point i where the derivative is desired for a total of $r+l+1$ points. The Taylor expansion is:

$$u_{i+m} = u_i + \frac{(m\Delta x)}{1!}u_x + \frac{(m\Delta x)^2}{2!}u_{xx} + \frac{(m\Delta x)^3}{3!}u_{xxx} + \frac{(m\Delta x)^4}{4!}u_{xxxx} + \frac{(m\Delta x)^5}{5!}u_{xxxxx} + \ldots \tag{3.33}$$

for $m=-l,\ldots,r$. Multiplying each of these expansions by a constant a_m and summing them up we obtain the following equation:

$$\begin{aligned}
\sum_{m=-l,m\neq 0}^{r} a_m u_{i+m} - \left(\sum_{m=-l,m\neq 0}^{r} a_m\right) u_i \;=\; & \left(\sum_{m=-l,m\neq 0}^{r} m a_m\right) \frac{\Delta x}{1!}\left.\frac{\partial u}{\partial x}\right|_i \\
+ & \left(\sum_{m=-l,m\neq 0}^{r} m^2 a_m\right) \frac{\Delta x^2}{2!}\left.\frac{\partial^2 u}{\partial x^2}\right|_i \\
+ & \left(\sum_{m=-l,m\neq 0}^{r} m^3 a_m\right) \frac{\Delta x^3}{3!}\left.\frac{\partial^3 u}{\partial x^3}\right|_i \\
+ & \left(\sum_{m=-l,m\neq 0}^{r} m^4 a_m\right) \frac{\Delta x^4}{4!}\left.\frac{\partial^4 u}{\partial x^4}\right|_i \\
+ & \left(\sum_{m=-l,m\neq 0}^{r} m^5 a_m\right) \frac{\Delta x^5}{5!}\left.\frac{\partial^5 u}{\partial x^5}\right|_i \\
+ & \ldots
\end{aligned} \tag{3.34}$$

It is clear that the coefficient of the k-th derivative is given by $b_k = \sum_{m=-l,m\neq 0}^{r} m^k a_m$. Equation (3.34) allows us to determine the $r+l$ coefficients a_m according to the derivative desired and the order desired. Hence if the first order derivative is needed at fourth order accuracy, we would set b_1 to 1 and $b_{2,3,4}=0$. This would provide us with four equations, and hence we need at least four points in order to determine its solution uniquely. More generally, if we need the k-th derivative then the highest derivative to be neglected must be of order $k+p-1$, and hence $k+p-1$ points are needed. The equations will then have the form:

$$b_q = \sum_{m=-l, m\neq 0}^{r} m^q a_m = \delta_{qk}, q = 1, 2, \ldots, k+p-1 \tag{3.35}$$

where δ_{qk} is the Kronecker delta $\delta_{qk} = 1$ is $q = k$ and 0 otherwise. For the solution to exit and be unique we must have: $l + r = k + p$. Once the solution is obtained we can determine the leading order truncation term by calculating the coefficient multiplying the next higher derivative in the truncation error series:

$$b_{k+1} \sum_{m=-l, m\neq 0}^{r} m^{k+p} a_m. \tag{3.36}$$

Example 8 As an example of the application of the previous procedure, let us fix the stencil to $r = 1$ and $l = -3$. Notice that this is an off-centered scheme. The system of equation then reads as follows in matrix form:

$$\begin{pmatrix} -3 & -2 & -1 & 1 \\ (-3)^2 & (-2)^2 & (-1)^2 & (1)^2 \\ (-3)^3 & (-2)^3 & (-1)^3 & (1)^3 \\ (-3)^4 & (-2)^4 & (-1)^4 & (1)^4 \end{pmatrix} \begin{pmatrix} a_{-3} \\ a_{-2} \\ a_{-1} \\ a_1 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix} \tag{3.37}$$

If the first derivative is desired to fourth order accuracy, we would set $b_1 = 1$ and $b_{2,3,4} = 0$, while if the second derivative is required to third order accuracy we would set $b_{1,3,4} = 0$ and $b_2 = 1$. The coefficients for the first example would be:

$$\begin{pmatrix} a_{-3} \\ a_{-2} \\ a_{-1} \\ a_1 \end{pmatrix} = \frac{1}{12} \begin{pmatrix} -1 \\ 12 \\ -18 \\ 3 \end{pmatrix} \tag{3.38}$$

3.3.5 Discrete Operator

Operators are often used to describe the discrete transformations needed in approximating derivatives. This reduces the lengths of formulae and can be used to derive new approximations. We will limit ourselves to the case of the centered difference operator:

$$\delta_{nx} u_i = \frac{u_{i+\frac{n}{2}} - u_{i-\frac{n}{2}}}{n\Delta x} \tag{3.39}$$

$$\delta_x u_i = \frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{\Delta x} = u_x + O(\Delta x^2) \tag{3.40}$$

$$\delta_{2x} u_i = \frac{u_{i+1} - u_{i-1}}{2\Delta x} = u_x + O(\Delta x^2) \tag{3.41}$$

The second order derivative can be computed by noticing that

$$\delta_x^2 u_i = \delta_x(\delta_x u_i) = \delta_x(u_x + O(\Delta x^2) \tag{3.42}$$

$$\delta_x \left(\frac{u_{i+\frac{1}{2}} - u_{i-\frac{1}{2}}}{\Delta x} \right) = u_{xx} + O(\Delta x^2) \tag{3.43}$$

$$\frac{1}{\Delta x} \left(\delta_x(u_{i+\frac{1}{2}}) - \delta_x(u_{i-\frac{1}{2}}) \right) = u_{xx} + O(\Delta x^2) \tag{3.44}$$

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2}) = u_{xx} + O(\Delta x^2) \tag{3.45}$$

The truncation error can be verified by going through the formal Taylor series analysis.

Another application of operator notation is the derivation of higher order formula. For example, we know from the Taylor series that

$$\delta_{2x} u_i = u_x + \frac{\Delta x^2}{3!} u_{xxx} + O(\Delta x^4) \tag{3.46}$$

If I can estimate the third order derivative to second order then I can substitute this estimate in the above formula to get a fourth order estimate. Applying the δ_x^2 operator to both sides of the above equation we get:

$$\delta_x^2(\delta_{2x} u_i) = \delta_x^2(u_x + \frac{\Delta x^2}{3!} u_{xxx} + O(\Delta x^4)) = u_{xxx} + O(\Delta x^2) \tag{3.47}$$

Thus we have

$$\delta_{2x} u_i = u_x + \frac{\Delta x^2}{3!} \delta_x^2 [\delta_{2x} u_i + O(\Delta x^2)] \tag{3.48}$$

Rearranging the equation we have:

$$u_x|_{x_i} = \left(1 - \frac{\Delta x^3}{3!} \delta_x^2 \right) \delta_{2x} u_i + O(\Delta x^4) \tag{3.49}$$

3.4 Polynomial Fitting

Taylor series expansion are not the only means to develop finite difference approximation. An another approach is to rely on polynomial fitting such as splines (which we will not discuss here), and Lagrange interpolation. We will concentrate on the later in the following section.

Lagrange interpolation consists of fitting a polynomial of a specified defree to a given set of (x_i, u_i) pairs. The slope at the point x_i is approximated by taking the derivative of the polynomial at the point. The approach is best illustrate by looking at specific examples.

3.4.1 Linear Fit

The linear polynomial:

$$L_1(x) = \frac{x - x_i}{\Delta x} u_{i+1} - \frac{x - x_{i+1}}{\Delta x} u_i, \quad x_i \le x \le x_{i+1} \tag{3.50}$$

The derivative of this function yields the forward difference formula:

$$u_x|_{x_i} = \left.\frac{\partial L_1(x)}{\partial x}\right|_{x_i} = \frac{u_{i+1} - u_i}{\Delta x} \tag{3.51}$$

A Taylor series analysis will show this approximation to be linear. Likewise if a linear interpolation is used to interpolate the function in $x_{i-1} \leq x \leq x_i$ we get the backward difference formula.

3.4.2 Quadratic Fit

It is easily verified that the following quadratic interpolation will fit the function values at the points x_i and $x_{i\pm1}$:

$$L_2(x) = \frac{(x - x_i)(x - x_{i+1})}{2\Delta x^2} u_{i-1} - \frac{(x - x_{i-1})(x - x_{i+1})}{\Delta x^2} u_i + \frac{(x - x_{i-1})(x - x_i)}{2\Delta x^2} u_{i+1} \tag{3.52}$$

Differentiating the functions and evaluating it at x_i we can get expressions for the first and second derivatives:

$$\left.\frac{\partial L_2}{\partial x}\right|_{x_i} = \frac{u_{i+1} - u_{i-1}}{2\Delta x} \tag{3.53}$$

$$\left.\frac{\partial^2 L_2}{\partial x^2}\right|_{x_i} = \frac{u_{i+1} - 2u_i + u_{i-1}}{\Delta x^2} \tag{3.54}$$

Notice that these expression are identical to the formulae obtained earlier. A Taylor series analysis would confirm that both expression are second order accurate.

3.4.3 Higher order formula

Higher order fomula can be develop by Lagrange polynomials of increasing degree. A word of caution is that high order Lagrange interpolation is practical when the evaluation point is in the **middle of the stencil**. High order Lagrange interpolation is notoriously noisy near the end of the stencil when equal grid spacing is used, and leads to the well known problem of Runge oscillations Boyd (1989). Spectral methods that do not use periodic Fourier functions (the usual "sin" and "cos" functions) rely on **unevenly** spaced points.

To illustrate the Runge phenomenon we'll take the simple example of interpolating the function

$$f(x) = \frac{1}{1 + 25x^2} \tag{3.55}$$

in the interval $|x| \leq 1$. The Lagrange interpolation using an equally spaced grid is shown in the upper panel of figure 3.4, the solid line refers to the exact function f while the dashed-colored lines to the Lagrange interpolants of different orders. In the center of the interval (near $x = 0$, the difference between the dashed lines and the solid black line decreases quickly as the polynomial order is increased. However, near the edges of the interval, the Lagrangian interpolants oscillates between the interpolation points.

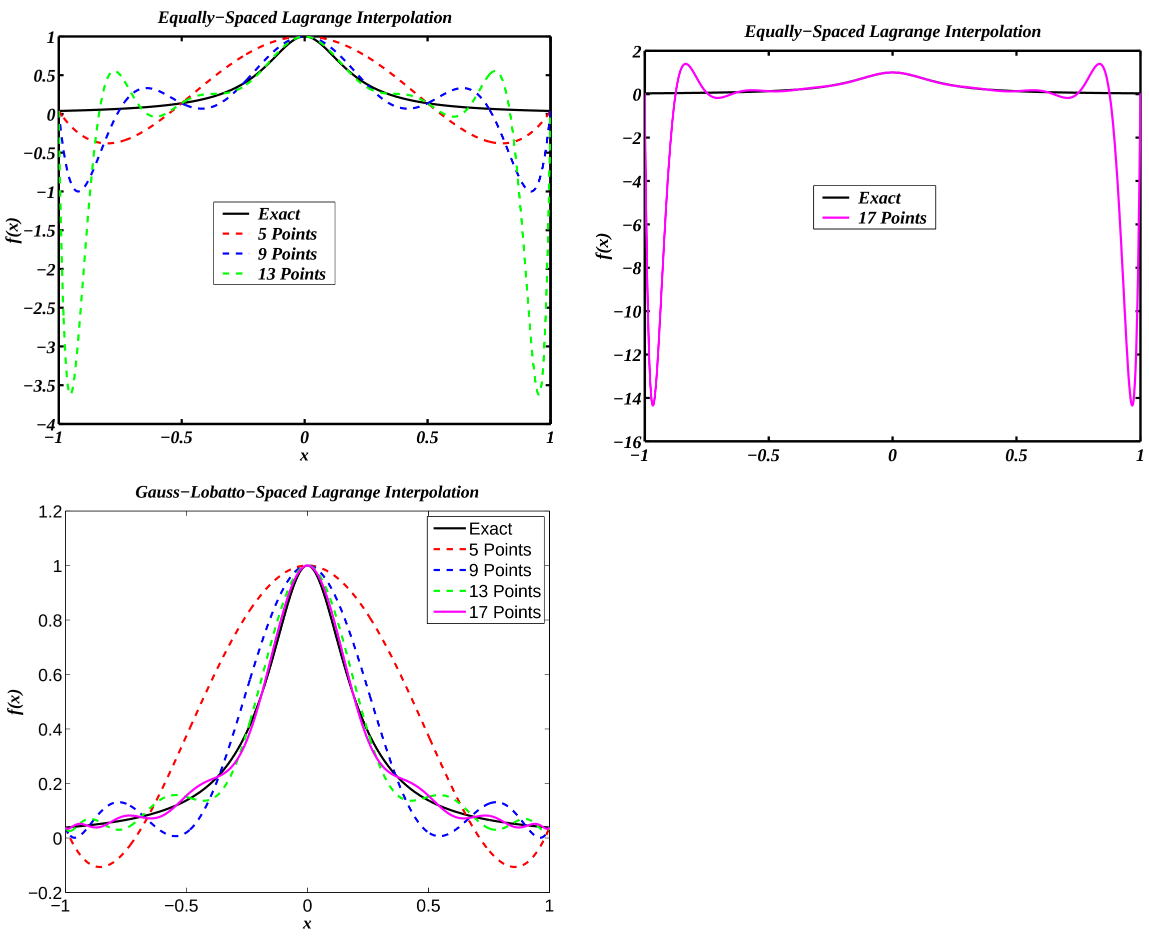


Figure 3.4: Illustration of the Runge phenomenon for equally-spaced Lagrangian interpolation (upper figures). The right upper figure illustrate the worsening amplitude of the oscillations as the degree is increased. The Runge oscillations are suppressed if an unequally spaced set of interpolation point is used (lower panel); here one based on Gauss-Lobatto roots of Chebyshev polynomials. The solution black line refers to the exact solution and the dashed lines to the Lagrangian interpolants. The location of the interpolation points can be guessed by the crossing of the dashed lines and the solid black line.

At a fixed point near the boundary, the oscillations' amplitude becomes bigger as the polynomial degree is increased: the amplitude of the 16 order polynomial reaches of value of 17 and has to be plotted separately for clarity of presentation. This is not the case when a non-uniform grid is used for the interpolation as shown in the lower left panel of figure 3.4. The interpolants approach the true function in the center and at the edges of the interval. The points used in this case are the Gauss-Lobatto roots of the Chebyshev polynomial of degree $N-1$, where N is the number of points.

3.5 Compact Differencing Schemes

A major disadvantage of the finite difference approach presented earlier is the widening of the computational stencil as the order of the approximation is increased. These large stencils are cumbersome near the edge of the domain where no data is available to perform the differencing. Fortunately, it is possible to derive high-order finite difference approximation with compact stencils at the expense of a small complication in their evalution: implicit differencing schemes (as opposed to explicit schemes) must be used. Here we show how these schemes can be derived.

3.5.1 Derivation of 3-term compact schemes

The Taylor series expansion of u_{i+m}, where $u_{i+m} = u(x_i + m\Delta x)$ about point x_i can be written as

$$u_{i+m} = \sum_{n=0}^{\infty} \frac{(m\Delta x)^n}{n!} u^{(n)} \tag{3.56}$$

where $u^{(n)}$ is the n-th derivative of u with respect to x at x_i, with m being an arbitrary number. From this expression it is easy to obtain the following sum and difference

$$u_{i+m} \pm u_{i-m} = \sum_{n=0}^{\infty} \left((1 \pm (-1)^n\right) \frac{(m\Delta x)^n}{n!} u^{(n)} \tag{3.57}$$

$$\frac{u_{i+m} + u_{i-m}}{2} = \sum_{n=0,2,4}^{\infty} \frac{(m\Delta x)^n}{n!} u^{(n)} \tag{3.58}$$

$$\frac{u_{i+m} - u_{i-m}}{2m\Delta x} = \sum_{n=0,2,4}^{\infty} \frac{(m\Delta x)^n}{(n+1)!} u^{(n+1)} \tag{3.59}$$

These expansion apply to arbitraty functions u as long as the expansion is valid; so they apply in particular to their derivatives. In case we substitute $u^{(1)}$ for u in the summation expansion we obtain an expression for the expansion of the derivatives:

$$\frac{u^{(1)}_{i+m} + u^{(1)}_{i-m}}{2} = \sum_{n=0,2,4}^{\infty} \frac{(m\Delta x)^n}{n!} u^{(n+1)} \tag{3.60}$$

Consider centered expansions of the following form

$$\alpha u^{(1)}_{i-1} + u^{(1)}_i + \alpha u^{(1)}_{i+1} = a_1 \delta_{2x} u_i + a_2 \delta_{4x} u_i + a_3 \delta_{6x} u_i \tag{3.61}$$

where α, and the a_i are unknown constants. The Taylor series expansion of the left and right hand sides can be matched as follows

$$u^{(1)}_i + 2\alpha \sum_{n=0,2,4}^{\infty} \frac{1}{n!} \Delta x^n u^{(n+1)} = \sum_{n=0,2,4}^{\infty} \frac{a_1 + 2^n a_2 + 3^n a_3}{(n+1)!} \Delta x^n u^{(n+1)} \tag{3.62}$$

or

$$\sum_{n=0,2,4}^{\infty} \frac{(a_1 + 2^n a_2 + 3^n a_3) - (n+1)(\delta_{n0} + 2\alpha)}{(n+1)!} \Delta x^n u^{(n+1)} = 0 \tag{3.63}$$

Here δ_{n0} refers to the Kronecker delta: $\delta_{nm} = 0$ for $n \neq m$ and 1 if $n = m$. This leads to the following constraints on the constants a_i and α:

$$a_1 + a_2 + a_3 = 1 + 2\alpha \quad \text{for } n = 0 \tag{3.64}$$

$$a_1 + 2^n a_2 + 3^n a_3 = 2(n+1)\alpha \quad \text{for } n = 2, 4, \dots, N \tag{3.65}$$

with a leading truncation error of the form:

$$\frac{a_1 + 2^{N+2}a_2 + 3^{N+2}a_3 - 2(N+3)\alpha}{(N+3)!}\Delta x^{N+2}u^{(N+3)} \tag{3.66}$$

Since we only have a handful of parameters we cannot hope to satisfy all these constraints for all n. However, we can derive progressively better approximation by matching higher-order terms. Indeed with 4 paramters at our disposal we can only satisfy 4 constraints. Let us explore some of the possible options.

3.5.2 Families of Fourth order schemes

The smallest computational stencil is obtained by setting $a_2 = a_3 = 0$, in which case only 2 parameters are left to maximize the accuracy of the scheme, and only 2 constraints can be imposed:

$$\left\{ \begin{array}{ccccc} a_1 & - & 2 & \alpha = & 1 \\ a_1 & - & 2\frac{3!}{2!} & \alpha = & 0 \end{array} \right. \quad \text{with solution} \quad \left\{ \begin{array}{ccc} \alpha & = & \frac{1}{4} \\ a_1 & = & \frac{3}{2} \end{array} \right. \tag{3.67}$$

A family of fourth order schemes can be obtained if we allow a wider stencil on the right hand side of the equations, and allow $a_2 \neq= 0$. This family of schemes can be generated by the single parameter α

$$\left\{ \begin{array}{ccccc} a_1 & + & & a_2 = & 1 + 2\alpha \\ a_1 & + & 4 & a_2 = & 6\alpha \end{array} \right. \quad \text{with solution} \quad \left\{ \begin{array}{ccc} a_1 & = & \frac{2}{3}(\alpha + 2) \\ a_2 & = & \frac{1}{3}(4\alpha - 1) \end{array} \right. \tag{3.68}$$

The leading terms in the truncation error would then be

$$TE = \frac{a_1 + 2^4 a_2 - 2 \cdot 5\alpha}{5!}\Delta x^4 u^{(5)} + \frac{a_1 + 2^6 a_2 - 2 \cdot 7\alpha}{7!}\Delta x^6 u^{(7)} \tag{3.69}$$

$$= \frac{4(3\alpha - 1)}{5!}\Delta x^4 u^{(5)} + \frac{4(18\alpha - 5)}{7!}\Delta x^6 u^{(7)} \tag{3.70}$$

This family of compact scheme can be made unique by, for example, requiring the scheme to be sixth-order and setting $\alpha = 1/3$; the leading truncation error term would then be $4/(7!)\Delta x^6 u^{(7)}$.

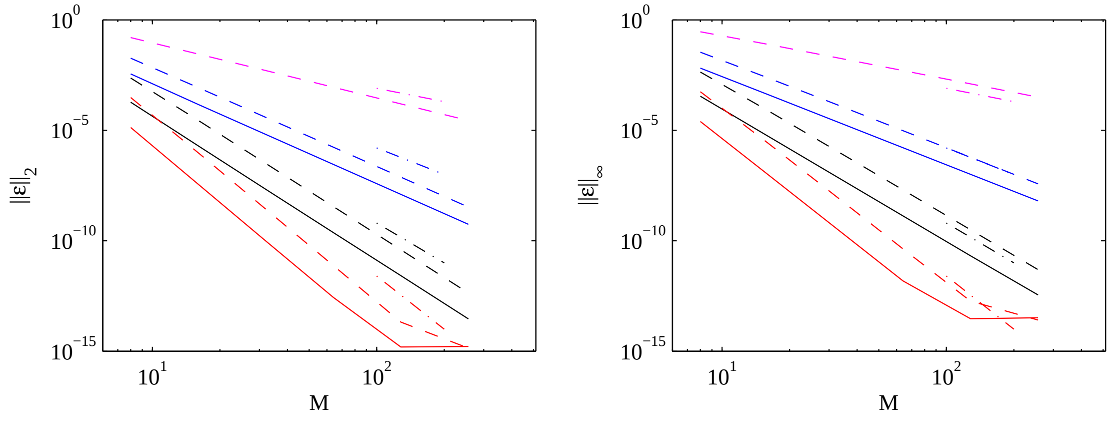


Figure 3.5: Convergence curves for the compact (solid ligns) and explicit centered difference (dashed lines) schemes for the sample function $u = \sin \pi x$, $|x| \leq 1$. The dash-dot lines refers to lines of slope -2,-4,-6, and -8. The left panel shows the 2-norm of the error while the right panel shows the ∞-norm.

3.5.3 Families of Sixth order schemes

Allowing the stencil on the right hand sides to expand, $a_3 \neq 0$, we can generate families of sixth-order schemes easily. The constraints are given by

$$\left\{\begin{array}{ccccccc} a_1 & + & a_2 & + & a_3 & = & 1+2\alpha \\ a_1 & + & 2^2 a_2 & + & 3^2 a_3 & = & 6\alpha \\ a_1 & + & 2^4 a_2 & + & 3^4 a_3 & = & 10\alpha \end{array}\right. \quad \text{with solution} \quad \left\{\begin{array}{ccc} a_1 & = & \dfrac{\alpha+9}{6} \\ a_2 & = & \dfrac{32\alpha-9}{15} \\ a_3 & = & \dfrac{-3\alpha+1}{10} \end{array}\right. \tag{3.71}$$

The leading terms in the truncation error would then be

$$TE = \frac{a_1 + 2^6 a_2 + 3^6 a_3 - 2\cdot 7\alpha}{7!}\Delta x^6 u^{(7)} + \frac{a_1 + 2^8 a_2 + 3^8 a_3 - 2\cdot 9\alpha}{9!}\Delta x^8 u^{(9)} \tag{3.72}$$

$$= \frac{12(-8\alpha+3)}{7!}\Delta x^6 u^{(7)} + \frac{72(-20\alpha+7)}{9!}\Delta x^8 u^{(9)} \tag{3.73}$$

Again, the formal order of the scheme can be made eighth order if one chooses $\alpha = 3/8$.

3.5.4 Numerical experiments

Figure 3.5 illustrate the convergence of the various compact scheme presented here for the sample function $u = \sin \pi x$. For comparison we have shown in dashed lines the convergence curves for the explicit finite difference schemes presented earlier. The dashed-dot lines are reference curves with slopes -2,-4,-6 and -8, respectively. For this infinitly differentiable function, the various schemes achieve their theoretically expected convergence order as the order fo the scheme is increased. It should be noted that, although, the convergence curves of the two approaches are parallel (the same slope), the errors of the compact schemes are lower then those of their explicit differene counter-parts.

Chapter 4

Application of Finite Differences to ODE

In this chapter we explore the application of finite differences in the simplest setting possible, namely where there is only one independent variable. The equations are then referred to as ordinary differential equations (ODE). We will use the setting of ODE's to introduce several concepts and tools that will be useful in the numerical solution of partial differential equations. Furthermore, time-marching schemes are almost universally reliant on finite difference methods for discretization, and hence a study of ODE's is time well spent.

4.1 Introduction

Here we derive how an ODE may be obtained in the process of solving numerically a partial differential equations. Let us consider the problem of solving the following PDE:

$$u_t + cu_x = \nu u_{xx}, \quad 0 \leq x \leq L \tag{4.1}$$

subject to periodic boundary conditions. Equation (4.1) is an advection diffusion equation with c being the advecting velocity and ν the viscosity coeffficient. We will take c and ν to be positive constants. The two independent variables are t for time and x for space. Because of the periodicity, it is sensible to expand the unknown function in a Fourier series:

$$u = \sum_{k=-\infty}^{\infty} \hat{u}_n(t) e^{ik_n x} \tag{4.2}$$

where $\hat{u}_n$ are the complex amplitudes and depend only on the time variable, whereas e^{ikx} are the Fourier functions with wavenumber k_n. Because of the periodicity requirement we have $k_n = 2\pi n/L$ where n is an integer. The Fourier functions form what is called an orthonormal basis, and can be determined as follows: multiply the two sides of equations (4.2) by $e^{-ik_m x}$ where m is integer and integrate over the interval $[0\pi]$ to get:

$$\int_0^L u e^{-ik_m x} \, \mathrm{d}x = \sum_{k=-\infty}^{\infty} \hat{u}_n(t) \int_0^L e^{i(k_n - k_m)x} \, \mathrm{d}x \tag{4.3}$$

Now notice that the integral on the right hand side of equation (4.3) satisfies the **orthogonality** property:

$$\int_0^L e^{i(k_n-k_m)x}\,\mathrm{d}x = \begin{cases} \dfrac{e^{i(k_n-k_m)L}-1}{i(k_n-k_m)} = \dfrac{e^{i2\pi(n-m)L}-1}{i(k_n-k_m)} = 0, \quad n \neq m \\ L, \quad n = m \end{cases} \tag{4.4}$$

The role of the integration is to pick out the $m-th$ Fourier component since all the other integrals are zero. We end up with the following expression for the Fourier coefficients:

$$\hat{u}_m = \frac{1}{L}\int_0^L u(x)e^{-ik_mx}\,\mathrm{d}x \tag{4.5}$$

Equation (4.5) would allow us to calculate the Fourier coefficients for a known function u. Note that for a real function, the Fourier coefficients satisfy

$$\hat{u}_{-n} = \hat{u}_n^* \tag{4.6}$$

where the * superscript stands for the complex conjugate. Thus, only the positive Fourier components need to be determined and the negative ones are simply the complex conjugates of the positive components.

The Fourier series (4.2) can now be differentiated term by term to get an expression for the derivatives of u, namely:

$$u_x = \sum_{k=-\infty}^{\infty} ik_n\hat{u}_n(t)e^{ik_nx} \tag{4.7}$$

$$u_{xx} = \sum_{k=-\infty}^{\infty} -k_n^2\hat{u}_n(t)e^{ik_nx} \tag{4.8}$$

$$u_t = \sum_{k=-\infty}^{\infty} \frac{\mathrm{d}\hat{u}_n}{\mathrm{d}t}e^{ik_nx} \tag{4.9}$$

Replacing these expressions for the derivative in the original equation and collecting terms we arrive at the following equations:

$$\sum_{k=-\infty}^{\infty}\left[\frac{\mathrm{d}\hat{u}_n}{\mathrm{d}t} + (ick_n + \nu k_n^2)\hat{u}_n\right]e^{ik_nx} = 0 \tag{4.10}$$

Note that the above equation has to be satisfied for all x and hence its Fourier amplitudes must be zero for all x (just remember the orthogonality property (4.4), and replace u by zero). Each Fourier component can be studied separately thanks to the linearity and the constant coefficients of the PDE.

The governing equation for the Fourier amplitude is now

$$\frac{\mathrm{d}\hat{u}}{\mathrm{d}t} = \underbrace{-(ick + \nu k^2)}_{\kappa}\,\hat{u} \tag{4.11}$$

where we have removed the subscript n to simplify the notation, and have introduced the complex number κ. The solution to this simple ODE is:

$$\hat{u} = \hat{u}_0 e^{\kappa t} \tag{4.12}$$

where $\hat{u_0} = \hat{u}(t=0)$ is the Fourier amplitude at the initial time. Taking the ratio of the solution between time t and $t+\Delta t$, we can get see the expected behavior of the solution between two consquetive times:

$$\frac{\hat{u}(t+\Delta t)}{\hat{u}(t)} = e^{\kappa \Delta t} = e^{\mathcal{R}e(\kappa)\Delta t} e^{i\mathcal{I}m(\kappa)\Delta t} \tag{4.13}$$

where $\mathcal{R}e(\kappa)$ and $\mathcal{I}m(\kappa)$ refer to the real and imaginary parts of κ. It is now clear to follow the evolution of the amplitude of the Fourier components:

$$|\hat{u}(t+\Delta t)| = |\hat{u}(t)| e^{\mathcal{R}e(\kappa)\Delta t} \tag{4.14}$$

The analytical solution predicts an exponential decrease if $\mathcal{R}e(\kappa) < 0$, an exponential increase if $\mathcal{R}e(\kappa) > 0$, and a constant amplitude if $\mathcal{R}e(\kappa) = 0$. The imaginary part of κ influences only the phase of the solution and decreases by an amount $\mathcal{I}m(\kappa)\Delta t$. We now turn to the issue of devising numerical solution to the ODE.

4.2 Forward Euler Approximation

Let us approximate the time derivative in (4.11) by a forward difference approximation (the name forward Euler is also used) to get:

$$\frac{u^{n+1} - u^n}{\Delta t} \approx \kappa u^n \tag{4.15}$$

where the superscript indicates the time level:$u^n = u(n\Delta t)$, and where we have removed the ˆ for simplicity. Equation (4.15) is an **explicit** approximation to the original differential equation since no information about the unknown function at the future time $(n+1)\Delta t$ has been used on the right hand side of the equation. In order to derive the error committed in the approximation we rely again on Taylor series. Expanding u^{n+1} about time level $n\Delta t$s, and inserting in the forward difference expression (4.15) we get:

$$u_t - \kappa u = \underbrace{-\frac{\Delta t}{2} u_t - \frac{\Delta t^2}{3!} u_{tt}}_{\text{truncation error } \sim O(\Delta t)} \tag{4.16}$$

The terms on the right hand side are the truncation errors of the forward Euler approximation. The formal definition of the truncation error is that it is the difference between the analytical and approximate representation of the differential equation. The leading error term (for sufficiently small Δt) is linear in Δt and hence we expect the errors to decrease linearly. Most importantly, the approximation is **consistent** in that the truncation error goes to zero as $\Delta t \to 0$.

Given the initial condition $u(t = 0) = u_0$ we can advance the solution in time to get:

$$\begin{aligned}
&u^1 = (1 + \kappa\Delta t)u^0 \\
&u^2 = (1 + \kappa\Delta t)u^1 = (1 + \kappa\Delta t)^2 u^0 \\
&u^3 = (1 + \kappa\Delta t)u^2 = (1 + \kappa\Delta t)^3 u^0 \\
&\vdots \\
&u^n = (1 + \kappa\Delta t)u^{n-1} = (1 + \kappa\Delta t)^n u^0
\end{aligned} \tag{4.17}$$

Let us study what happens when we let $\Delta t \to 0$ for a fixed time integration $t_n = n\Delta t$. The only factor we need to worry about is the numerical amplification factor:$(1 + \kappa\Delta t)$:

$$\lim_{\Delta t\to 0}(1 + \kappa\Delta t)^{\frac{t_n}{\Delta t}} = \lim_{\Delta t\to 0} e^{\frac{t_n \ln(1+\kappa\Delta t)}{\Delta t}} = \lim_{\Delta t\to 0} e^{\frac{t_n(\kappa\Delta t - \kappa^2\Delta t^2 + \ldots)}{\Delta t}} = e^{\kappa t_n} \tag{4.18}$$

where we have used the logarithm Taylor series $\ln(1 + \epsilon) = \epsilon - \epsilon^2 + \ldots$, assuming that $\kappa\Delta t$ is small. Hence we have proven convergence of the numerical solution to the analytic solution in the limit $\Delta t \to 0$. The question is what happens for finite Δt?

Notice that in analogy to the analytic solution we can define an amplification factor associated with the numerical solution, namely:

$$A = \frac{u^n}{u^{n-1}} = |A|e^{i\theta} \tag{4.19}$$

where θ is the argument of the complex number A. The amplitude of A will determine whether the numerical solution is amplifying or decaying, and its argument will determine the change in phase. The numerical amplification factor should mimic the analytical amplification factor, and should lead to an anologous increase or decrease of the solution. For small $\kappa\Delta t$ it can be seen that A is just the first term of the Taylor series expansion of $e^{\kappa\Delta t}$ and is hence only first order accurate. Let us investigate the magnitude of A in terms of κ, a problem parameter, and Δt the numerical parameter, we have:

$$|A|^2 = AA^* = 1 + 2\mathcal{R}e(\kappa)\Delta t + |\kappa|^2\Delta t^2 \tag{4.20}$$

We focus in particular for the condition under which the amplitude factor is less then 1. The following condition need then to be fullfilled (assuming $\Delta t > 0$):

$$\Delta t \leq -2\frac{\mathcal{R}e(\kappa)}{|\kappa|^2} \tag{4.21}$$

There are two cases to consider depending on the sign of $\mathcal{R}e(\kappa)$. If $\mathcal{R}e(\kappa) > 0$ then $|A| > 1$ for $\Delta t > 0$, and the finite difference solution will grow like the analytical solution. For $\mathcal{R}e(\kappa) = 0$, the solution will also grow in amplitude whereas the analytical solution predicts a neutral amplification. If $\mathcal{R}e(\kappa) < 0$, then $|A| > 1$ for $\Delta t > -2\frac{\mathcal{R}e(\kappa)}{|\kappa|^2}$ whereas the analytical solution predicts a decay. The moral of the story is that the numerical solution can behave in unexpected ways. We can rewrite the amplification factor in the following form:

$$|A|^2 = AA^* = [\mathcal{R}e(z) + 1]^2 + [\mathcal{I}m(z)]^2 \tag{4.22}$$

where $z = \kappa\Delta t$. The above equation can be interpreted as the equation for a circle centered at $(-1, 0)$ in the complex plane with radius $|A|^2$. Thus z must be within the unit circle centered at $(-1, 0)$ for $|A|^2 \leq 1$.

4.3 Stability, Consistency and Convergence

Let us denote by u the exact analytical solution to the differential equation, and by U the numerical solution obtained using a finite difference scheme. The quantity $\|U - u\|$ is a norm of the error. we have the following definitions:

- **Convergence** A scheme is called to converge to $O(\Delta t^p)$ if $\|U - u\| = O(\Delta t^p)$ as $\Delta t \to 0$, where p is a positive constant.
- **Truncation Error** The local difference between the difference approximation and the differential equations. It is the error introduced if the exact solution is plugged into the difference equations.
- **Consistency** The finite difference approximation is called consistent with the differential equation if the truncation error goes to zero when the numerical parameters are made arbitrarily small.
- **Stability** A method is called stable if there is a constant C independent of the time step or the number of time steps such that:

  $$\|U^n\| < C\|U^0\| \tag{4.23}$$

  Equation (4.23) is a very loose constraint on the numerical approximation to guarantee convergence. This constraint allows the solution to grow in time (indeed the solution can still grow exponentially fast), but rules out growth that depends on the number of time steps or the step size. In situations where the analytical solution is known not to grow, it is entirely reasonable to put the restriction:

  $$\|U^n\| < \|U^0\| \tag{4.24}$$

4.3.1 Lax Richtmeyer theorem

The celebrated Lax-Richtmeyer theorem links the notion of consistency and stability for linear differential equations. It maintains that for linear differential equations, a consistent finite difference approximation converges to the true solution if the scheme is stable. The converse is also true in that a convergent and consistent numerical solution must be stable. We will show a here simplified version of the Lax-Richtmeyer equivalence theorem to highlight the relationships between consistency stability, and justify the constraints we place on the finite difference approximations. A general form for the integration of the ODE U takes the form:

$$U^n = AU^{n-1} + b^{n-1} \tag{4.25}$$

where A is the multiplicative factor and b a source sink term that does not depend on u. If the exact solution is plugged into the above recursive formula we get:

$$u^n = Au^{n-1} + b^{n-1} + T^{n-1}\Delta t \tag{4.26}$$

where T^{n-1} is the truncation error at time n. Substracting the two equations from each others, and invoking the linearity of the process, we can derive an equation for the evolution of the error in time, namely:

$$e^n = Ae^{n-1} - T^{n-1}\Delta t \tag{4.27}$$

where $e^n = U^n - u^n$ is the total error at time $t_n = n\Delta t$. The reapplication of this formula to e^{n-1} transforms it to:

$$e^n = A(Ae^{n-2} - T^{n-2}\Delta t) - T^{n-1}\Delta t = (A)^2 e^{n-2} - \Delta t\left[AT^{n-2} + T^{n-1}\right] \tag{4.28}$$

where $(A)^2 = A.A$, and the remainder of the superscript indicate time levels. Repeated application of this formula shows that:

$$e^n = (A)^n e^0 - \Delta t\left[(A)^{n-1}T^0 + (A)^{n-2}T^1 + \ldots + AT^{n-2} + T^{n-1}\right] \tag{4.29}$$

Equation (4.29) shows that the error at time level n depends on the initial error, on the history of the truncation error, and on the discretization through the factor A. We will now attempt to bound this error and show that this possible if the truncation error can be made arbitrarily small (the consistency condition), and if the scheme is stable according to the definition shown above. A simple application of the triangle inequality shows that

$$|e^n| \leq |A^n|\;|e^0| + \Delta t\left[\left|(A)^{n-1}\right|\;\left|T^0\right| + \left|(A)^{n-2}\right|\;\left|T^1\right| + \ldots + |A|\;\left|T^{n-2}\right| + \left|T^{n-1}\right|\right] \tag{4.30}$$

Now we define $T = \max|T^m|$ for all $0 \leq m \leq n-1$, that is T is the maximum norm of the truncation error encountered during the course of the calculation, then the right hand side of the above inequality can be bounded again:

$$|e^n| \leq |A^n|\;|e^0| + \Delta t T\left[\sum_{m=0}^{n-1}|(A)^m|\right] \tag{4.31}$$

In order to proceed further we need to introduce also the maximum bound on the amplification factor and all its powers. So let us assume that the scheme is **stable**, i.e. there is a positive constant C, independent of Δt and n, such that

$$\max\left(|A^m|\right) \leq C, \text{ for } 0 \leq m \leq n \tag{4.32}$$

Since the individual entries in the summation are smaller then C and the sum involves n terms, the sum must be smaller then nC. The inequality (4.32) is then bounded above by $C|E^0| + nT\Delta tC$, and we arrive at:

$$|e^n| \leq C|e^0| \;+\; t_n\,T\,C \tag{4.33}$$

where $t_n = n\Delta t$ is the final integration time. The right hand side of (4.33) can be made arbitrarily small by the following argument. First, the initial condition is known and so the initial error is (save for round-off errors) zero. Second, since the scheme is consistent, the maximum truncation error T can be made arbitrarily small by choosing smaller and smaller Δt. The end results is that the bound on the error can be made arbitrarily small if the approximation is **stable** and **consistent**. Hence the scheme converges to the solution as $\Delta t \to 0$.

4.3.2 Von Neumann stability condition

Here we derive a practical bound on the amplification factor $|A|$ based on the criteria used in the derivation of the equivalence theorem $|A^m| \le C$:

$$
\begin{aligned}
|A^m| = |A|^m &\le C && (4.34)\\
|A| &\le C^{\frac{1}{m}} = e^{\frac{\Delta t \ln C}{t_m}} && (4.35)\\
|A| &\le 1 + O(\Delta t) && (4.36)
\end{aligned}
$$

where we have used the Taylor series for the exponential in arriving at the final expression. This is the least restrictive condition on the amplification factor that will permit us to bound the error growth for finite times. Thus the modulus of the amplification factor maybe greater then 1 by an amount proportional to positive powers of $|\Delta t|$. This gives plenty of latitude for the numerical solution to grow, but will prevent this growth from depending on the time step or the number of time steps.

In practice, the stability criterion is too generous, particularly when we know the solution is bounded. The growth of the numerical solution should be bounded at all times by setting $C = 1$. In this case the Von Neumann stability criterion reduces to

$$|A| \le 1 \qquad (4.37)$$

4.4 Backward Difference

The backward difference formula to the ODE is

$$\left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{t_{n+1}} \approx \frac{u^{n+1} - u^n}{\Delta t} = \kappa^{un+1} \qquad (4.38)$$

This is an example of an **implicit** method since the unknown u^{n+1} has been used in evaluating the slope of the solution on the right hand side; this is not a problem to solve for u^{n+1} in this scalar and linear case. For more complicated situations like a nonlinear right hand side or a system of equations, a nonlinear system of equations may have to be inverted. It is easy to show via Taylor series analysis that the truncation error for the backward difference scheme is $O(\Delta t)$ and the scheme is hence consistent and of first order. The numerical solution can be updated according to:

$$u^{n+1} = \frac{u^n}{1 - \kappa \Delta t} \qquad (4.39)$$

and the amplification factor is simply $A = 1/(1 - \kappa\Delta t)$. Its magnitude is given by:

$$|A|^2 = \frac{1}{1 - 2\mathcal{R}e(\kappa)\Delta t + |\kappa|^2 \Delta t^2} \qquad (4.40)$$

The condition under which this amplification factor is bounded by 1 is

4.5 Backward Difference

The backward difference formula to the ODE is

$$\left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{t_{n+1}} \approx \frac{u^{n+1}-u^n}{\Delta t} = \kappa^{un+1} \tag{4.41}$$

This is an example of an **implicit** method since the unknown u^{n+1} has been used in evaluating the slope of the solution on the right hand side; this is not a problem to solve for u^{n+1} in this scalar and linear case. For more complicated situations like a nonlinear right hand side or a system of equations, a nonlinear system of equations may have to be inverted. It is easy to show via Taylor series analysis that the truncation error for the backward difference scheme is $O(\Delta t)$ and the scheme is hence consistent and of first order. The numerical solution can be updated according to:

$$u^{n+1} = \frac{u^n}{1-\kappa\Delta t} \tag{4.42}$$

and the amplification factor is simply $A = 1/(1-\kappa\Delta t)$. Its magnitude is given by:

$$|A|^2 = \frac{1}{1-2\mathcal{R}e(\kappa)\Delta t + |\kappa|^2\Delta t^2} \tag{4.43}$$

The condition under which this amplification factor is bounded by 1 is

$$\Delta t \geq 2\frac{\mathcal{R}e(\kappa)}{\kappa^2} \tag{4.44}$$

again depend on the sign of $\mathcal{R}e(\kappa)$. If $\mathcal{R}e(\kappa) < 0$, then $|A| < 1$ for all $\Delta t > 0$; this is an instance of unconditional stability. The numerical solution is also damped when $\mathcal{R}e(\kappa) = 0$ whereas the analytical solution is neutral. The numerical amplitude factor can be rewritten as:

$$[1-\mathcal{R}e(z)]^2 + [\mathcal{I}m(z)]^2 = \frac{1}{|A|^2} \tag{4.45}$$

and shows contours of constant amplitude factors to be circles centered at (1,0) and of radius $1/|A|$.

4.6 Trapezoidal Scheme

The trapezoidal scheme is an an example of second order scheme that uses only two time levels. It is based on applying the derivative at the intermediate time $n+\frac{1}{2}$, and using a centered difference formula with step size $\Delta t/2$. The derivation is as follows:

$$\left.\frac{\mathrm{d}u}{\mathrm{d}t}\right|_{t_{n+\frac{1}{2}}} = \kappa u^{n+\frac{1}{2}} \tag{4.46}$$

$$\frac{u^{n+1}-u^n}{\Delta t} + O(\Delta t^2) = \kappa\frac{u^{n+1}+u^n}{2} + O(\Delta t^2) \tag{4.47}$$

using simple Taylor series expansions about time $n+\frac{1}{2}$. The truncation error is $O(\Delta t^2)$ and the method is hence second order accurate. It is implicit since u^{n+1} is used in the evaluation of the right hand side. The unkown function can be updated as:

$$u^{n+1} = \frac{1+\frac{\kappa\Delta t}{2}}{1-\frac{\kappa\Delta t}{2}}u^n \tag{4.48}$$

The amplification factor is

$$A = \frac{1+\frac{\kappa\Delta t}{2}}{1-\frac{\kappa\Delta t}{2}}, \quad |A|^2 = \frac{1+\mathcal{R}e(\kappa\Delta t)+\frac{|\kappa^2|\Delta t^2}{4}}{1-\mathcal{R}e(\kappa\Delta t)+\frac{|\kappa^2|\Delta t^2}{4}} \tag{4.49}$$

The condition for $|A| < 1$ is simply

$$\mathcal{R}e(\kappa) \leq 0 \tag{4.50}$$

The scheme is hence unconditionally stable for $\mathcal{R}e(\Delta t) < 0$ and neutrally stable ($|A| = 1$) for $\mathcal{R}e(\kappa) = 0$.

Example 9 To illustrate the application of the different scheme we proceed to evaluate numerically the solution of $u_t = -iu$ with the initial condition $u = 1$. The analytical solution in the complex u plane is a circle that starts at $(1, 0)$ and proceeds counterclockwise. We then proceed to compute the numerical solutions using the forward, backward and trapezoidal schemes.

The modulus of the forward Euler solution cycles outward and is indicative of the unstable nature of the scheme. The backward Euler solution cycles inward indicative of a numerical solution that is too damped. Finally, the trapezoidal scheme has neutral amplification: its solution remains on the unit circle and tracks the analytical solution quite closely. Notice however, that the trapezoidal solution seem to lag behind the analytical solution, and this lag seems to increase with time. This is symptomatic of lagging *phase errors*.

4.6.1 Phase Errors

Stability is primarily concerned with the modulus of the amplification factor. However, the accuracy of the numerical scheme depends on the amplitude and phase errors of the scheme. The phase error can be analyzed by inspecting the argument of the amplification factor, the θ term in equation (4.19). The analytical change of phase for our model problem is $\theta_e = \mathcal{I}m(\kappa)\Delta t$, the ratio of the numerical and analytical phase is called the relative phase error:

$$R = \frac{\theta}{\theta_e} \tag{4.51}$$

When $R > 1$ the numerical phase exceeds the analytical phase and we call the scheme accelerating; if $R < 1$ the scheme is decelerating. For the forward differencing scheme the relative phase is given by:

$$R = \frac{1}{\mathcal{I}m(z)}\tan^{-1}\left(\frac{\mathcal{I}m(z)}{1+\mathcal{R}e(z)}\right) \tag{4.52}$$

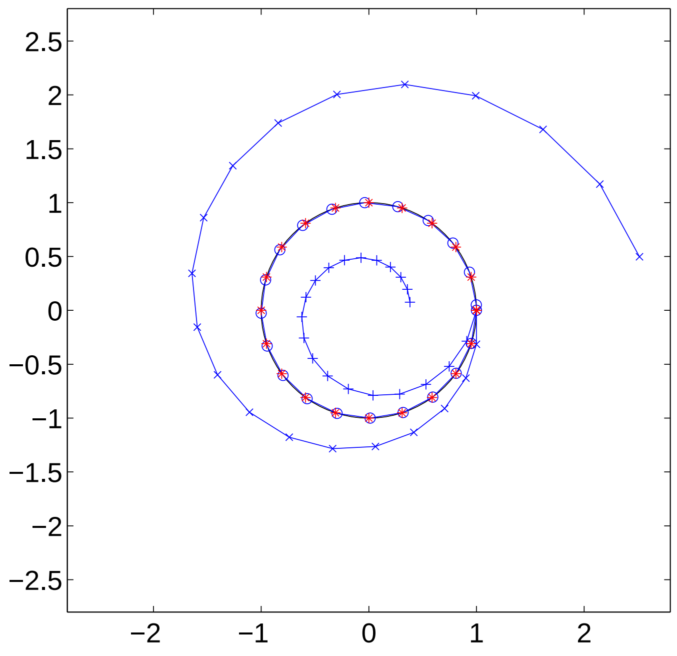


Figure 4.1: Solution of the oscillation equation using the forward (x), backward (+) and trapezoidal schemes (○). The analytical solution is indicated by a red asterisk.

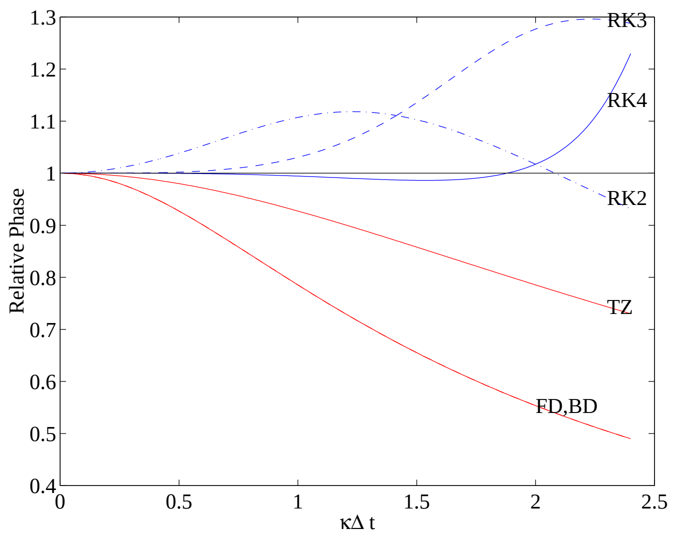


Figure 4.2: Phase errors for several two-level schemes when $\mathcal{R}e(\kappa) = 0$. The forwad and backward differencing schemes (FD and BD) have the same decelerating relative phase. The Trapezoidal scheme (TZ) has lower phase error for the same $\kappa\Delta t$ as the 2 first order schemes. The Runge Kutta schemes of order 2 and 3 are accelerating. The best performance is for RK4 which stays closest to the analytical curve for the largest portion of the spectrum.

In general it is hard to get a simple formula for the phase error since the expressions often involve the tan^{-1} functions with complicated arguments. Figure 4.2 shows the relative phase as a function of $\kappa\Delta t$ for the case where $\mathcal{R}e(\kappa) = 0$ for several time integration schemes. The solid black line ($R = 1$) is the reference for an exact phase. The forward, backward and trapezoidal differencing have negative phase errors (and hence the schemes are decelerating), while the RK schemes (to be presented below) have an accelerating phase.

4.7 Higher Order Methods

4.7.1 Multi Stage (Runge Kutta) Methods

One approach to increasing the order of the calculations without using information at previous time levels is to introduce intermediate stages in the calculations. The most popular of these approaches is referred to as the Runge Kutta methods. We will illustrate their derivation for the second order scheme. For generality, we will assume that the ODE takes the form

$$\frac{\mathrm{d}u}{\mathrm{d}t} = f(u,t), u(0) = u_0 \tag{4.53}$$

The derivation of the second order Runge Kutta method starts with the expression:

$$\begin{aligned} u^{(1)} &= u^n + a_{21}\Delta t &(4.54)\\ u^{n+1} &= u^n + b_1\Delta t f(u^n, t_n) + b_2\Delta t f(u^{(1)}, t_n + c_2\Delta t) &(4.55)\end{aligned}$$

where a_{21}, b_1, b_2 and c_2 are constant that will be determined on the basis of accuracy. Variants of this approach has the constants determined on the basis of stability considertaion, but in the following we follow the accuracy criterion. The key to determining these constants is to match the Taylor series expansion for the ODE with that of the approximation. Expanding u as a Taylor series in time we get:

$$u^{n+1} = u^n + \Delta t\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{\Delta t^2}{2!}\frac{\mathrm{d}^2u}{\mathrm{d}t^2} + O(\Delta t^3) \tag{4.56}$$

Notice that the ODE provides the information necessary to compute the derivative in the Taylor series. Thus we have:

$$\begin{aligned} \frac{\mathrm{d}u}{\mathrm{d}t} &= f(u,t) &(4.57)\\ \frac{\mathrm{d}^2u}{\mathrm{d}t^2} &= \frac{\mathrm{d}f}{\mathrm{d}t} = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}\frac{\mathrm{d}u}{\mathrm{d}t} &(4.58)\\ &= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}f &(4.59)\\ \frac{\mathrm{d}^3u}{\mathrm{d}t^3} &= \frac{\mathrm{d}f_t}{\mathrm{d}t} + \frac{\mathrm{d}f_u}{\mathrm{d}t}f + f_u\frac{\mathrm{d}f}{\mathrm{d}t} &(4.60)\\ &= f_{tt} + f_{ut}f + f_{ut}f + f_{uu}f^2 + f_uf_t + f_u^2f &(4.61)\\ &= f_{tt} + 2f_{ut}f + f_{uu}f^2 + f_uf_t + f_u^2f &(4.62)\end{aligned}$$

Replacing these two derivatives in the Taylor series expression we get:

$$u^{n+1} = u^n + \Delta t f(u^n, t_n) + \frac{\Delta t^2}{2!}[f_u f + f_t]_{t_n} + \frac{\Delta t^3}{3!}[f_{tt} + 2f_{ut}f + f_{uu}f^2 + f_u f_t + f_u^2 f] + O(\Delta t^4) \tag{4.63}$$

We now turn to expanding the expression for the proposed difference equations. We have to proceed carefully to include the effects of changes in u and t in our expansion. We start by expanding the last term of equation (4.55) about the variable u^n.

$$f(u^n + a_{21}\Delta t f, t_n + c_2\Delta t) = f(u^n, t_n + c_2\Delta t) + a_{21}\Delta t f f_u + \frac{(a_{21}\Delta t f)^2}{2!} f_{uu} + O(\Delta t^3) \tag{4.64}$$

Now each term in expansion (4.64) is expanded in the t variable about time t_n to get:

$$\begin{aligned} f(u^n, t_n + c_2\Delta t) &= f(u^n, t^n) + c_2\Delta t f_t + \frac{(c_2\Delta t)^2}{2!} f_{tt} + O(\Delta t^2) && (4.65)\\ f_u(u^n, t_n + c_2\Delta t) &= f_u(u^n, t_n) + f_{ut}(u^n, t_n) c_2 \Delta t + O(\Delta t^2) && (4.66)\\ f_{uu}(u^n, t_n + c_2\Delta t) &= f_{uu}(u^n, t_n) + O(\Delta t) && (4.67) \end{aligned}$$

Substituting these expressions in expansion (4.64) we get the two-variable Taylor series expansion for f. The whole expression is then inserted in (4.55) to get:

$$\begin{aligned} u^{n+1} &= u^n + (b_2 + b_1) f \Delta t + (b_2 a_{21} f f_u + b_2 c_2 f_t)\Delta t^2 && (4.68)\\ &+ \left(b_2 \frac{c_2^2}{2} f_{tt} + b_2 a_{21} c_2 f f_{ut} + \frac{b_2 a_{21}^2}{2} f_{uu}\right)\Delta t^3 + O(\Delta t^3) \end{aligned}$$

Matching the expansions (4.68) and (4.63) term by term we get the following equations for the different constants.

$$\left\{\begin{aligned} b_2 + b_1 &= 1\\ 2b_2 a_{21} &= 1\\ 2b_2 c_2 &= 1 \end{aligned}\right. \tag{4.69}$$

A solution can be found in term of the parameter b_2, and it is as follows:

$$\left\{\begin{aligned} b_1 &= 1 - b_2\\ a_{21} &= \tfrac{1}{2b_2}\\ c_2 &= \tfrac{1}{2b_2} \end{aligned}\right. \tag{4.70}$$

A family of second order Runge Kutta schemes can be obtained by varying b_2. Two common choices are

- **Midpoint rule** with $b_2 = 1$, so that $b_1 = 0$ and $a_{21} = c_2 = \frac{1}{2}$. The schemes becomes:

$$\begin{aligned} u^{(1)} &= u^n + \frac{\Delta t}{2} f(u^n, t_n) && (4.71)\\ u^{n+1} &= u^n + \Delta t f(u^{(1)}, t_n + \frac{\Delta t}{2}) && (4.72) \end{aligned}$$

The first phase of the midpoint rule is a forward Euler half step, followed by a centered approximation at the mid-time level.

- **Heum rule** $b_2 = \frac{1}{2}$ and $a_{21} = c_2 = 1$.

$$u^{(1)} = u^n + \Delta t f(u^n, t_n) \tag{4.73}$$

$$u^{n+1} = u^n + \frac{\Delta t}{2}[f(u^{(1)}, t_n + \Delta t) + f(u^n, t_n)] \tag{4.74}$$

  The first step is forward Euler full step followed by a centered step with the averaged sloped.

Higher order Runge Kutta schemes can be derived by introducing additional stages between the two time levels; their derivation is however very complicated (for more information see Butcher (1987) and see Dormand (1996) for a more readable account). Here we limit ourselves to listing the algorithms for common third and fourth order Runge Kutta schemes.

- **RK3**

$$\begin{array}{ll} q_1 = \Delta t f(u^n, t_n) & u^{(1)} = u^n + \frac{q_1}{3} \\ q_2 = \Delta t f(u^{(1)}, t_n + \frac{\Delta t}{3}) - \frac{5q_1}{9} & u^{(2)} = u^{(1)} + \frac{15q_2}{16} \\ q_3 = \Delta t f(u^{(2)}, t_n + \frac{\Delta t}{3}) - \frac{153q_2}{128} & u^{n+1} = u^{(2)} + \frac{8q_3}{15} \end{array} \tag{4.75}$$

- **RK4** The fourth order RK scheme is the most well-known for its accuracy and large stability region. It is:

$$\begin{array}{ll} q_1 = & \Delta t f(u^n, t_n) \\ q_2 = & \Delta t f(u^n + \frac{q_1}{2}, t_n + \frac{\Delta t}{2}) \\ q_3 = & \Delta t f(u^n + \frac{q_2}{2}, t_n + \frac{\Delta t}{2}) \\ q_4 = & \Delta t f(u^n + q_3, t_n + \Delta t) \\ u^{n+1} = & u^n + \frac{q_1 + 2q_2 + 2q_3 + q_4}{6} \end{array} \tag{4.76}$$

4.7.2 Remarks on RK schemes

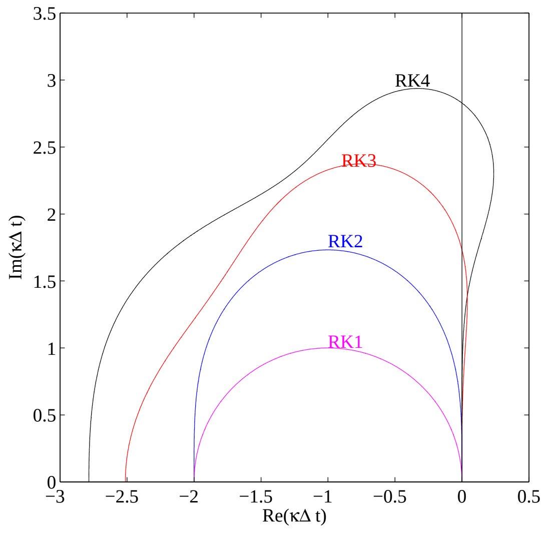
The following is a quick summary of the RK properties of different orders.

1. Implicit Runge Kutta time steps are possible.

2. Runge Kutta offers high order integration with only information at two time levels. Automatic step size control is easy since the order of the method does not depend on maintaining the same step size as the calculation proceeds. Several strategies are then possible to maximize accuracy and reduce CPU cost.

3. For order less then or equal to 4, only one stage is required per additional order, which is optimal. For higher order, it is known that the number of stages exceeds the order of the method. For example, a fifth order method requires 6 stages, and an eight order RK scheme requires a minimum of 11 stages.

4. Runge Kutta schemes require multiple evaluation of the right hand side per time step. This can be quite costly if a large system of simultaneous ODE's is involved. Alternatives to the RK steps are multi-level schemes.



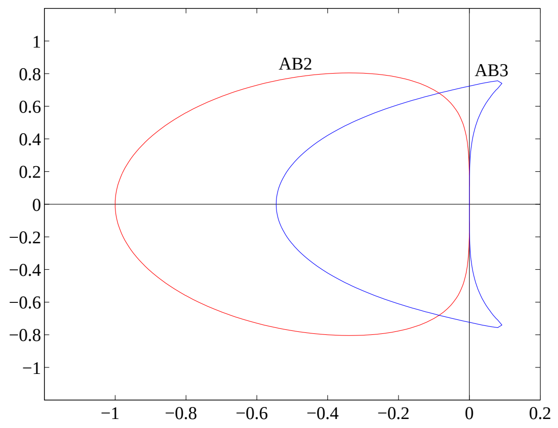


Figure 4.3: Stability region for the Runge-Kutta methods of order 1, 2, 3, and 4 (left figure), and for the Adams Bashforth schmes of order 2 and 3 (right figure). The RK2 and AB2 stability curves are tangent to the imaginary axis at the origin, and hence the method are not stable for purely imaginary $\kappa\Delta t$.

5. A new family of Runge Kutta scheme was devised in recent years to cope with the requirements of Total Variations Diminishing (TVD) schemes. For second order methods, the Heum scheme is TVD. The third order TVD Runge Kutta scheme is

$$\begin{array}{ll} q_1 = \Delta t f(u^n, t_n) & u^{(1)} = u^n + \frac{q_1}{3} \\ q_2 = \Delta t f(u^{(1)}, t_n + \Delta t) & u^{(2)} = \dfrac{3}{4}u^n + \dfrac{1}{4}u^{(1)} + \dfrac{1}{4}q_2 \\ q_3 = \Delta t f(u^{(2)}, t_n + \frac{\Delta t}{2}) & u^{n+1} = \dfrac{1}{3}u^n + \dfrac{2}{3}u^{(2)} + \dfrac{2}{3}q_3 \end{array} \tag{4.77}$$

4.7.3 Multi Time Levels Methods

The Runge-Kutta methods achieve their accuracy by evaluating the right hand side function at intermediate time levels. The cost of this accuracy is a multiple evaluation of f for a an integration of size Δt. This may prove to be expensive if we are looking at complicated right hand sides and/or systems of ODEs. An alternative is to use information prior to t_n to increase the order of accuracy at t_{n+1}.

Leap Frog scheme

The leap frog scheme consists of using a centered difference in time at level n:

$$u^{n+1} = u^{n-1} + 2\Delta t f(u^n, t_n) \tag{4.78}$$

It is easy to show that the truncation error is of size $O(\Delta t^2)$. Moreover, unlike the trapezoidal scheme, it is explicit in the unknown u^{n+1}, and hence does not involve nonlinear complications, nor systems of equations. For our model equation, the trapezoidal scheme takes the form:

$$u^{n+1} = u^{n-1} + 2\kappa\Delta t u^n \tag{4.79}$$

The determination of the amplification factor is complicated by the fact that two time levels are involved in the calculations. Nevertheless, let us assume that the amplification factor is the same for each time step, i.e. $u^n = Au^{n-1}$, and $u^{n+1} = Au^n$. We then arrive at the following equation:

$$A^2 - 2zA - 1 = 0 \tag{4.80}$$

There are two solutions to this quadratic equation:

$$A_{\pm} = z \pm \sqrt{1+z^2} \tag{4.81}$$

In the limit of good resolution, $|z| \rightarrow 0$, we have $A_+ \rightarrow 1$ and $A_- \rightarrow -1$. The numerical solution is capable of behaving in two different ways, or modes. The mode associated with A_+ is referred to as the physical mode because it approximates the solution to the original differential equation. The mode associated with A_- is referred to as the *computational mode* since it arises solely as an artifact of the numerical procedure.

The origin of the computational mode can be traced back to the fact that the leap-frog scheme is an approximation to a higher order equation that requires more initial conditions then necessary for the original ODE. To show this consider the trivial case of $\kappa = 0$ where the analytical solution is simply given by $u_a = u_0$; here u_0 is the initial condition. The amplitude factors for the leap-frog schemes are $A_+ = 1$ and $A_- = -1$, and hence the computational mode is expected to keep its amplitude but switch sign at every time step. Applying the leap-frog scheme we see that all even time levels will have the correct value: $u^2 = u^4 = \ldots = u_0$. The odd time levels will be contaminated by error in estimating the second initial condition needed to jump start the calculations. If $u^1 = u_0 + \epsilon$ where ϵ is the initial error committed, the solution at all odd time levels will then be $u^{2n+1} = u_0 + \epsilon$. The numerical solution for the present simple case can be written entirely in terms of the physical (initial condition) and computational (initial condition error) modes:

$$u^n = u^0 + \frac{\epsilon}{2} \; - \; (-1)^n \frac{\epsilon}{2} \tag{4.82}$$

Absolute stability requires that $|A|_{\pm} \leq 1$; notice however that the product of the two roots is $A_+A_- = -1$, which implies that $|A_+||A_-| = 1$. Hence, if one root, say A_+ is stable $|A_+| < 1$, the other one must be unstable with $|A_-| = 1/|A_+| > 1$; the only exception is when both amplification factor have a neutral amplification $|A_+| = |A_-| = 1$. For real z, $Im(z) = 0$, one of the two roots has modulus exceeding 1, and the scheme is always unstable. Let us for a moment assume that $z = i\lambda$, we then have: $A = i\lambda + \sqrt{1-\lambda^2}$. If $\lambda \leq 1$ then the quantity under the square root sign is positive and we have two roots such that $|A_+| = |A_-| = 1$. To make further progress on studying the stability of the leap frog scheme, let $z = \sinh(w)$ where w is a complex number. Using the identity $\cosh^2 w - \sinh^2 w = 1$ we arrive at the expression $A_{\pm} = \sinh w \pm \cosh w$. Setting $w = a + ib$ where a, b are real, subsituting in the previous expression for the amplification factor, and calculating its modulus we get: $|A_{\pm}| = e^{\pm a}$. Hence $a = 0$ for both amplification factors to be stable. The region of stability is hence $z = i \sin b$ where b is real, and is confined to the unit slit along the imaginary axis $|\mathcal{I}m(z)| \leq 1$.

The leap frog scheme is a popular scheme to integrate PDE's of primarily hyperbolic type in spite of the existence of the computational mode. The reason lies primarily in its neutral stability and good phase properties. The control of the computational mode can be effectively achieved either with a Asselin time filter (see Durran (1999)) or by discarding periodically the solution at level $n-1$ and taking a two time level scheme.

Multi-Step schemes

A family of multi-step schemes can built upon interpolating the right hand side of the ODE in the interval $[t_n t_{n+1}]$ and performing the integral. The derivation starts from the exact solution to the ODE:

$$u^{n+1} = u^n + \int_{t_n}^{t_{n+1}} f(u,t)\ \mathrm{d}t \tag{4.83}$$

Since the integrand is unknown in $[t_n t_{n+1}]$ we need to find a way to approximate it given information at specific time levels. A simple way to achieve this is to use a polynomial that interpolates the integrand at the points (t_k, u^k), $n-p \leq k \leq n$, where the solution is known. If we write:

$$f(u,t) = V_p(t) + E_p(t) \tag{4.84}$$

where V_p is the polynomial approximation and E_p the error associated with it, then the numerical scheme becomes:

$$u^{n+1} = u^n + \int_{t_n}^{t_{n+1}} V_p(t)\ \mathrm{d}t + \int_{t_n}^{t_{n+1}} E_p(t)\ \mathrm{d}t \tag{4.85}$$

If the integration of V_p is performed exactly, then the only approximation errors present are due to the integration of the interpolation error term; this term can be bounded by $\max(|E_p|\Delta t$.

The explicit family of Adams Bashforth scheme relies on Lagrange interpolation. Specifically,

$$\begin{aligned}
V_p(t) &= \sum_{k=0}^{p} h_k^p(t) f^{n-p}, && (4.86)\\
h_k^p(t) &= \prod_{m=0, m\neq k}^{p} \frac{t - t_{n-m}}{t_{n-k} - t_{n-m}} && (4.87)\\
&= \frac{t - t_n}{t_{n-k} - t_n} \cdots \frac{t - t_{n-(k-1)}}{t_{n-k} - t_{n-(k-1)}} \frac{t - t_{n-k-1}}{t_{n-k} - t_{n-k-1}} \cdots \frac{t - t_{n-p}}{t_{n-k} - t_{n-p}} && (4.88)
\end{aligned}$$

It is easy to verify that $h_k^p(t)$ is a polynomial of degree $p-1$ in t, and that $h_k^p(t_{n-m}) = 0$ for $m \neq k$ and $h_k^p(t_{n-k}) = 1$. These last two properties ensures that $V_p(t_{n-k}) = f^{n-k}$. The error associated with the Lagrange interpolation with $p+1$ points is $O(\Delta t^{p+1})$. Inserting the expressions for V_p in the numerical scheme, and we get:

$$u^{n+1} = u^n + \sum_{k=0}^{p} f^{n-k} \left(\int_{t_n}^{t_{n+1}} h_k^p(t)\ \mathrm{d}t \right) \quad + \Delta t O(\Delta t^{p+2}) \tag{4.89}$$

Note that the error appearing in the above formula is only the local error, the global error is one order less, i.e. it is $O(\Delta t^{p+1})$.

We illustrate the application of this procedure by considering the derivation of its second and third order variants. The second order scheme requires $p = 1$. Hence, we write:

$$V_1(t) = \frac{t - t_{n-1}}{t_n - t_{n-1}} f^n + \frac{t - t_n}{t_{n-1} - t_n} f^{n-1} \tag{4.90}$$

The integral on the interval $[t_n t_{n+1}]$ is

$$\int_{t_n}^{t_{n+1}} V_1(t)\,\mathrm{d}t = \int_{t_n}^{t_{n+1}} \frac{t - t_{n-1}}{t_n - t_{n-1}}\,\mathrm{d}t f^n + \int_{t_n}^{t_{n+1}} \frac{t - t_n}{t_{n-1} - t_n}\,\mathrm{d}t f^{n-1} \tag{4.91}$$

$$= \Delta t \left(\frac{3}{2} f^n - \frac{1}{2} f^{n-1}\right) \tag{4.92}$$

The final expression for the second order Adams Bashforth formula is:

$$u^{n+1} = u^n + \Delta t \left(\frac{3}{2} f^n - \frac{1}{2} f^{n-1}\right) + O(\Delta t^3) \tag{4.93}$$

A third order formula can be designed similarly. Starting with the quadratic interpolation polynomial $V_2(t)$:

$$\begin{aligned} V_2(t) &= \frac{[t - t_{n-1}][t - t_{n-2}]}{[t_n - t_{n-1}][t_n - t_{n-2}]} f^n + \frac{[t - t_n][t - t_{n-2}]}{[t_{n-1} - t_n][t_{n-1} - t_{n-2}]} f^{n-1} \\ &+ \frac{[t - t_n][t - t_{n-1}]}{[t_{n-2} - t_n][t_{n-2} - t_{n-1}]} f^{n-2} \end{aligned} \tag{4.94}$$

Its integral can be evaluated and plugged into equation (4.89) to get:

$$u^{n+1} = u^n + \Delta t \left(\frac{23}{12} f^n - \frac{16}{12} f^{n-1} + \frac{5}{12} f^{n-2}\right) \tag{4.95}$$

The stability of the AB2 scheme can be easily determined for the sample problem. The amplification factors are the roots to the equation

$$A^2 - \left(1 + \frac{3}{2} z\right) A + \frac{z}{2} = 0 \tag{4.96}$$

and the two roots are:

$$A_{\pm} = \frac{1}{2}\left[\left(1 + \frac{3}{2} z\right) \pm \sqrt{\left(1 + \frac{3}{2} z\right)^2 - 2z}\,\right]. \tag{4.97}$$

Like the leap frog scheme AB2 suffers from the existence of a computational mode. In the limit of good resolution, $z \rightarrow 0$, we have $A_+ \rightarrow 1$ and $A_+ \rightarrow 0$; that is the computational mode is heavily damped. figure 4.4 shows the modulus of the physical and computational modes for $\mathcal{R}e(z) = 0$ and $Im(z) < 1$. The modulus of the computational mode amplitude factor is quite small for the entire range of z considered. On the other hand the physical mode is unstable for purely imaginary z as the modulus of its amplification factor exceeds

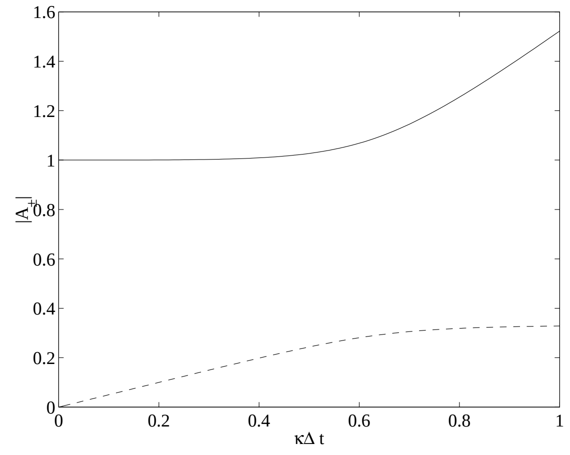


Figure 4.4: Modulus of amplification factor for the physical and computational modes of AB2 when $\mathcal{R}e(\kappa) = 0$.

1. Note however, that a series expansion of $|A_+|$ for small $z = i\lambda\Delta t$ shows that $|A_+| = 1 + (\lambda\Delta t^4)/4$ and hence the instability grows very slowly for sufficiently small Δt. It can be anticipated that AB3 will have one physical mode and two computational modes since its stability analysis leads to a third order equation for the amplification factor. Like AB2, AB3 strongly damps the two computational modes; it has the added benefit of providing conditional stability for $Im(z) \neq 0$. The complete stability regions for AB2 and AB3 is shown in the right panel of figure 4.3.

Like all multilevel schemes there are some disadvantages to Adams Bashforth methods. First a starting method is required to jump start the calculations. Second the stability region shrinks with the order of the method. The good news is that although AB2 is unstable for imaginary κ, its instability is small and tolerable for finite integration time. The third order Adams Bashforth scheme on the other hand includes portion of the imaginary axis, which makes AB3 quite valuable for the integration of advection like operators. The main advantage of AB schemes over Runge-Kutta is that they require but one evaluation of the right hand side per time step and use a similar amount of storage.

4.8 Strongly Stable Schemes

Occasionally we are interested in enlarging the stability region as much as possible, while maitaining a high convergence order. The lowest order scheme of that sort is the backward difference formula. We now look for equivalent higher order formula. The common thread is to evaluate the derivative term at the next time level. The taylor series expansion of u^{n-k} about time level u^{n+1} is:

$$u^{n-k+1} = u^{n+1} - \frac{(k\Delta t)}{1!}\frac{\mathrm{d}u}{\mathrm{d}t} + \frac{(k\Delta t)^2}{2!}\frac{\mathrm{d}^2u}{\mathrm{d}t^2} - \frac{(k\Delta t)^3}{3!}\frac{\mathrm{d}^3u}{\mathrm{d}t^3} + \frac{(k\Delta t)^4}{4!}\frac{\mathrm{d}^4u}{\mathrm{d}t^4} + \dots \qquad (4.98)$$

p	a_1	a_2	a_3	a_4	$\sum_{k=1}^{p} ka_k$
1	1				1
2	4/3	−1/3			2/3
3	18/11	−9/11	2/11		6/11
4	48/25	−36/25	16/25	−3/25	12/25

Table 4.1: Coefficients of the Backward Difference Formula of order 1, 2, 3 and 4.

where $k = 1, \dots, p$. Multiplying each of these expansion by a coefficient a_k and adding the individual terms, we get:

$$\begin{aligned}\sum_{k=1}^{p} a_k u^{n-k+1} &= \left(\sum_{k=1}^{p} a_k\right) u^{n+1} - \left(\sum_{k=1}^{p} k a_k\right) \frac{(k\Delta t)}{1!}\frac{\mathrm{d}u}{\mathrm{d}t} + \left(\sum_{k=1}^{p} k^2 a_k\right) \frac{(k\Delta t)^2}{2!}\frac{\mathrm{d}^2u}{\mathrm{d}t^2} \\ &- \left(\sum_{k=1}^{p} k^3 a_k\right) \frac{(k\Delta t)^3}{3!}\frac{\mathrm{d}^3u}{\mathrm{d}t^3} + \left(\sum_{k=1}^{p} k^4 a_k\right) \frac{(k\Delta t)^2}{4!}\frac{\mathrm{d}^4u}{\mathrm{d}t^4} + \dots \\ &- \left(\sum_{k=1}^{p} k^m a_k\right) \frac{(k\Delta t)^m}{m!}\frac{\mathrm{d}^mu}{\mathrm{d}t^m} + \dots \end{aligned} \tag{4.99}$$

For a p-th order expression, we require the higher order derivative, 2 through p, to vanish; this yields $p-1$ homogeneous algebraic equations. For a non-trivial solution we need to append one more conditions which we choose to be that the sum of the unknown coefficient is equal to one. This yields the following system of equations

$$\sum_{k=1}^{p} a_k = 1 \tag{4.100}$$

$$\sum_{k=1}^{p} k^m a_k = 0, m = 2, 3, \dots, p \tag{4.101}$$

for the p coefficients a_1, a_2, ..., a_p. In matrix form we have

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 2^2 & 3^2 & \dots & p^2 \\ 1 & 2^3 & 3^3 & \dots & p^3 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 2^p & 3^p & \dots & p^p \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_p \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{4.102}$$

The solution of this system for p equal to 2, 3 and 4 is shown in table 4.1. The corresponding expressions are:

$$u^{n+1} = u^n + \Delta t\ u_t|^{n+1} + O(\Delta t) \tag{4.103}$$

$$u^{n+1} = \frac{4}{3}u^n - \frac{1}{3}u^{n-1} + \frac{2}{3}\Delta t\ u_t|^{n+1} + O(\Delta t^2) \tag{4.104}$$

$$u^{n+1} = \frac{18}{11}u^n - \frac{9}{11}u^{n-1} + \frac{2}{11}u^{n-2} + \frac{6}{11}\Delta t\ u_t|^{n+1} + O(\Delta t^3) \tag{4.105}$$

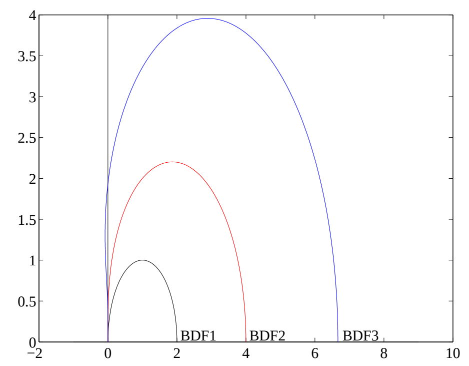


Figure 4.5: Stability regions for the Backward Differencing schemes of order 1, 2 and 3. The schemes are unstable within the enclosed region and stable everywhere else. The instability regions grow with the order. The stability regions are symmetric about the real axis

$$u^{n+1} = \frac{48}{25}u^n - \frac{36}{25}u^{n-1} + \frac{16}{25}u^{n-2} - \frac{3}{25}u^{n-3} + + \frac{12}{25}\Delta t\ u_t|^{n+1} + O(\Delta t^4) \quad (4.106)$$

Notice that we have shown explicitly the time level at which the time derivative is approximated. The BDF's scheme lead to **implicit** expressions to update the solution at time level u^{n+1}. Like the Adams-Bashforth formula the BDF schemes require a starting method. They also generate computational modes whose number depends on how many previous time levels have been used. Their most important advantage is their stability regions in the complex plane which are much larger then equivalent explicit schemes.

4.8.1 Stability of BDF

We investigate the stability of the BDF schemes for the simple case where $f(u,t) = \kappa u$. It has already been shown that the backward difference scheme is stable in the entire complex plane save for the inside of the unit circle centered at $(1, 0)$. The equation for the BDF2 amplification factor is easily derived:

$$\left(1 - \frac{2}{3}z\right)A^2 - \frac{4}{3}A + \frac{1}{3} = 0 \quad (4.107)$$

and admits the two roots:

$$A_\pm = \frac{2 \pm \sqrt{1+2z}}{3-2z} \quad (4.108)$$

The positive roots is the physical mode while the negative root is the computational mode. In the limit of $z \rightarrow 0$, we have $A_+ \rightarrow 1$ and $A_- \rightarrow 1/3$, the computational mode is hence naturally damped. Figure 4.5 shows the stability regions for the BDF schemes. The contours of $|A| = 1$ are shown in the figure. The schemes are unstable within the regions shown and stable outside it. The instability region grows with increasing order.

4.9 Systems of ODEs

The equations to be solved can form a system of equations:

$$\frac{\mathrm{d}\mathbf{u}}{\mathrm{d}t} = L\mathbf{u} \tag{4.109}$$

where now $\mathbf{u}$ represents a vector of unknown and L is a matrix. The preceding schemes can be all made to work with the system of equations by treating all the components in a consistent matter. The major problem is not computational per se, but conceptual and concerns the stability of a system of equation. For example, a backward differentiation of the system leads to the following set of equations for the unknowns at the next time level:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} = L\mathbf{u}^{n+1}, \text{ or } \mathbf{u}^{n+1} = P\mathbf{u}^n, \quad P = I - \Delta t L \tag{4.110}$$

If we denote the exact solution of the system as $\mathbf{v}$, then the error between the numerical and exact solutions is given by $\mathbf{e}^n = \mathbf{u}^n - \mathbf{v}(t_n)$. A number of vector norms are usefull to measure the error, namely, the 1, 2 or $|infty$-norms. The numerical solution converges if $\|\mathbf{e}\| \to 0$ as $\Delta t \to 0$. The concept of stability also carries over. It is clear that the solution, for a linear system at least, evolves as $\mathbf{u}^n = P^n\mathbf{u}^0$. And hence the solution will remain bounded if

$$\|\mathbf{u}^n\| = \|P^n\mathbf{u}^0\| \leq C\|\mathbf{u}^0\| \tag{4.111}$$

We now have to worry about the effect of the amplification matrix P. The problem with the stability analysis is that it is hard to relate $\|P^n\|$ and $\|P\|$. The matrix norms guarantee that $\|P^n\| \leq \|P\|^n$. Hence requiring that $\|P\| < 1$ will ensure stability. This is a sufficient condition but not a necessary condition. Since the spectral radius is a lower bound on the different matrix norms, it is necessary to require $\rho(P) \leq 1$. If P can be made diagonal, such as when it possess a complete set of linearly independent eigenvectors, then the requirement $\rho(P) \leq 1$ is sufficient and necessary.

Chapter 5

Numerical Solution of PDE's

5.1 Introduction

Suppose we are given a well-posed problem that consists of a partial differential equation

$$\frac{\partial u}{\partial t} = Lu \tag{5.1}$$

where L is a differential operator, initial conditions

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}) \tag{5.2}$$

and appropriate boundary conditions. We are now interested in devising a numerical scheme based on finite difference method to solve the above well-posed problem.

Let $v(x,t)$ be the exact solution of the problem. The numerical solution of these equations via finite differences requires us to replace the continuous derivatives by discrete approximations, and to confine ourselves with the solution of the problem at a discrete set of space and time points. Hence the numerical solution, denoted by u will be determined at the discrete space points $x_j = j\Delta x$, and time points $n\Delta t$. We will use the notation $u_j^n = u(x_j, t_n)$. The approximation must be **consistent**, **stable**, and **convergent** to be useful in modeling physical problems. We will turn to the issue of defining these important concepts shortly.

A simple example of this class of problem is the scalar advection equation in a single space dimension

$$u_t + cu_x = 0, \tag{5.3}$$

where c is the advection speed. In this case $L = -cu_x$, and an appropriate boundary conditions is to specify the value of u at the upstream boundary. To make the discussion more concrete let us illustrate the discretization process for the case mentioned above. For simplicity we assume that c is constant and positive. A simple finite difference scheme that would advance the solution in time for time level n to $n+1$ is to use a Forward Euler scheme for the time derivative, and a backward Euler scheme for the space derivative. We get the following approximation to the PDE at point (x_j, t^n).

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_j^n - u_{j-1}^n}{\Delta x} = 0 \tag{5.4}$$

Equation 5.4 provides a simple formula for updating the solution at time level $n+1$ from the values at time n:

$$u_j^{n+1} = (1-\mu)u_j^n + \mu u_{j-1}^n, \text{ where } \mu = \frac{c\Delta t}{\Delta x} \tag{5.5}$$

The variable μ is known as the Courant number and will figure prominently in the study of the stability of the scheme. Equation 5.5 can be written as a matrix operation in the following form:

$$\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}^{n+1} = \begin{pmatrix} 1 & & & & & & & \\ \mu & 1-\mu & & & & & & \\ & \ddots & \ddots & & & & & \\ & & \mu & 1-\mu & & & & \\ & & & \mu & 1-\mu & & & \\ & & & & \mu & 1-\mu & & \\ & & & & & \ddots & \ddots & \\ & & & & & & \mu & 1-\mu & \\ & & & & & & & \mu & 1-\mu \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix}^{n} \tag{5.6}$$

where we have assumed that the boundary condition is given by $u(x_1, t) = u_0(x_1)$.

The following legitimate question can now be asked:

1. **Consistency:** Is the discrete equation (5.4) a correct approximation to the continuous form, eq. (5.3), and does this discrete form reduce to the PDE in the limit of $\Delta t, \Delta x \to 0$.

2. **Convergence** Does the numerical solution $u_j^n \to v_j^n$ as $\Delta t, \Delta x \to 0$.

3. **Errors** What are the errors committed by the approximation, and how should one expect them to behave as the numerical resolution is increased.

4. **Stability** Does the numerical solution remained bounded by the data specifying the problem? or are the numerical errors increasing as the computations are carried out.

We will now turn to the issue of defining these concepts more precisely, and hint to the role they play in devising finite difference schemes. We will return to the issue of illustrating their applications in practical situations later.

5.1.1 Convergence

Let $e_j^n = u_j^n - v_j^n$ denote the error between the numerical and analytical solutions of the PDE at time $n\Delta t$ and point $j\Delta x$. If this error tends to 0 as the grid and time steps are decreased, the finite difference solution **converges** to the analytical solution. Moreover, a finite difference scheme is said to be convergent of order (p, q) if $\|e\| = O(\Delta t^p, \Delta x^q)$ as $\Delta t, \Delta x \to 0$.

5.1.2 Truncation Error

If the analytical solution is inserted in the finite difference scheme, we expect a small residual to remain. This residual characterizes the error in approximating the continuous form by a discrete form. By performing a Taylor series analysis we can derive an expression of this residual in terms of higher order derivatives of the solution.

5.1.3 Consistency

Loosely speaking the notion of consistency addresses the problem of whether the finite difference approximation is really representing the partial differential equations. We say that a finite difference approximation is **consistent** with a differential equation if the FD equations converge to the original equations as the time and space grids are refined. Hence, if the truncation error goes to zero as the time and space grids are refined we conclude that the scheme is consistent.

5.1.4 Stability

The notion of stability is a little more complicated to define. Our primary concern here is to make sure that numerical errors do not swamp the analytical solution. One way to ensure that is to require the solution to remain bounded by the initial data. Hence a definition of stability is to require the following

$$\|u^n\| \leq C\|u^0\| \tag{5.7}$$

where C is a positive constant that may depend on the final integration time t_n but not on the time nor on the space increments. Notice that this definition of stability is very general and does not refer to the behavior of the continuum equation. If the latter is known to preserve the norm of the solution, then the more restrictive condition

$$\|u^n\| \leq \|u^0\| \tag{5.8}$$

is more practical, particularly for finite Δt.

5.1.5 Lax-Richtmeyer Equivalence theorem

The Lax-Richtmeyer equivalence theorem ties these different notions together. It states the following "*Given a properly-posed linear initial value problem, and a finite difference approximation to it that satisfies the consistency condition, stability is the necessary and sufficient condition for convergence.* This theorem's value is that it guarantees convergence provided two simpler conditions are satisfied, namely consistency and stability. These two are considerable easier to check for general type problems then convergence.

5.2 Truncation Error

The analysis of the truncation error for the simple advection equation will be presented here. Inserting the exact solution v in the finite difference form 5.4, we get:

$$\frac{v_j^{n+1} - v_j^n}{\Delta t} + c\frac{v_j^n - v_{j-1}^n}{\Delta x} = 0 \tag{5.9}$$

The Taylor series, in time and space yield the following:

$$v_j^{n+1} = v_j^n + \Delta t\, v_t|_j^n + \frac{\Delta t^2}{2}\, v_{tt}|_j^n + O(\Delta t^3) \tag{5.10}$$

$$v_{j-1}^{n} = v_j^n - \Delta x\, v_x|_j^n + \frac{\Delta x^2}{2}\, v_{xx}|_j^n + O(\Delta x^3) \tag{5.11}$$

Substituting these expressions in equation 5.9 we get:

$$[v_t + cv_x] = \underbrace{\left[-\frac{\Delta t}{2} v_{tt} + \frac{c\Delta x}{2} v_{xx}\right] + O(\Delta t^2, \Delta x^2)}_{\text{T.E.}} \tag{5.12}$$

The terms on the left hand side of eq. 5.12 are the original PDE; all terms on the right hand side are part of the truncation error. They represent the residual by which the exact solution fails to satisfy the difference equation. For sufficiently small Δt and Δx, the leading term in the truncation series is linear in both Δt and Δx. Notice also that one can regard equation 5.12 as the true partial differential equation represented by the finite difference equation for *finite* Δt and Δx. The analysis of the different terms appearing in the truncation error series can give valuable insight into the behavior of the numerical approximation, and forms the basis of the **modified equation** analysis. We will return to this issue later. For now it is sufficient to notice that the truncation error tends to 0 as $\Delta t, \Delta x \to 0$, and hence the finite difference approximation is consistent.

5.3 The Lax Richtmeyer theorem

A general formula for the evolution of the finite difference solution is the following:

$$\mathbf{u}^n = \mathbf{A}\mathbf{u}^{n-1} + \mathbf{b}^{n-1} \tag{5.13}$$

where $\mathbf{A}$ is the evolution matrix, and $\mathbf{b}$ is a vector containing forcing terms and the effects of boundary conditions. The vector $\mathbf{u}^n$ holds the vector of solution values at time n. The truncation error at a specific time level can be obtained by applying the above matrix operation to the vector of exact solution values:

$$\mathbf{v}^n = \mathbf{A}\mathbf{v}^{n-1} + \mathbf{b}^{n-1} + \mathbf{z}^{n-1}\Delta t \tag{5.14}$$

where $\mathbf{z}$ is the vector of truncation error at time level $n-1$. Substracting equation 5.14 from 5.13, we get an evolution equation for the error, namely:

$$\mathbf{e}^n = \mathbf{A}\mathbf{e}^{n-1} + \mathbf{z}^{n-1}\Delta t \tag{5.15}$$

Equation 5.15 shows that the error at time level n is made up of two parts. The first one is the evolution of the error inherited from the previous time level, the first term on the right hand side of eq. 5.15, and the second part is the truncation error committed at the present time level. Since, this expression applies to a generic time level, the same expression holds for $\mathbf{e}^{n-1}$:

$$\mathbf{e}^{n-1} = \mathbf{A}\mathbf{e}^{n-2} + \mathbf{z}^{n-2}\Delta t \tag{5.16}$$

where we have assumed that the matrix $\mathbf{A}$ does not change with time to simplify the discussion (this is tantamount to assuming constant coefficients for the PDE). By repeated application of this argument we get:

$$\begin{aligned}
\mathbf{e}^n &= \mathbf{A}^2\mathbf{e}^{n-2} + \left(\mathbf{A}\mathbf{z}^{n-2} + \mathbf{z}^{n-1}\right)\Delta t && (5.17)\\
&= \mathbf{A}^3\mathbf{e}^{n-3} + \left(\mathbf{A}^2\mathbf{z}^{n-3} + \mathbf{A}\mathbf{z}^{n-2} + \mathbf{z}^{n-1}\right)\Delta t && (5.18)\\
&\vdots\\
&= \mathbf{A}^n\mathbf{e}^0 + \left(\mathbf{A}^n\mathbf{z}^0 + \mathbf{A}^{n-1}\mathbf{z}^1 + \ldots + \mathbf{A}\mathbf{z}^{n-2} + \mathbf{z}^{n-1}\right)\Delta t && (5.19)
\end{aligned}$$

Equation 5.19 shows that the error growth depends on the truncation error at all time levels, and on the discretization through the matrix $\mathbf{A}$. We can use the triangle inequality to get a bound on the norm of the error. Thus,

$$\|\mathbf{e}^n\| \leq \|\mathbf{A}^n\|\,\|\mathbf{e}^0\| + \left(\|\mathbf{A}^n\|\,\|\mathbf{z}^0\| + \|\mathbf{A}^{n-1}\|\,\|\mathbf{z}^1\| + \ldots + \|\mathbf{A}\|\,\|\mathbf{z}^{n-2}\| + \|\mathbf{z}^{n-1}\|\right)\Delta t \tag{5.20}$$

In order to make further progress we assume that the norm of the truncation error at any time is bounded by a constant ϵ such that

$$\epsilon = \max_{0\leq m\leq n-1}\left(\|\mathbf{z}^m\|\right) \tag{5.21}$$

The right hand side of inequality 5.20 can be bounded by

$$\|\mathbf{e}^n\| \leq \|\mathbf{A}^n\|\,\|\mathbf{e}^0\| + \epsilon\Delta t\left(\sum_{m=0}^{n-1}\|\mathbf{A}^m\|\right) \tag{5.22}$$

The initial errors and the subsequent truncation errors are thus modulated by the evolution matrices $\mathbf{A}^m$. In order to prevent the unbounded growth of the error norm as $n \to \infty$, we need to put a limit on the norm of the these matrices. This is in effect the **stability** property needed for convergence:

$$\|\mathbf{A}^m\| \leq C = \max_{1\leq m\leq n}\left(\|\mathbf{A}^m\|\right) \tag{5.23}$$

where C is a constant independent of n, Δt and Δx. The sum in bracket can be bounded by the factor nC; the final expression becomes:

$$\|\mathbf{e}^n\| \leq C\left(\|\mathbf{e}^0\| + t_n\epsilon\right) \tag{5.24}$$

where $t_n = n\Delta t$ is the final integration time. When $\Delta x \to 0$, the initial error $\|\mathbf{e}^n\|$ can be made as small as desired. Furthermore, by consistency, the truncation error $\epsilon \to 0$ when $\Delta t, \Delta x \to 0$. The global error is hence guarateed to go to zero as the computational grid is refined, and the scheme is convergent.

5.4 The Von Neumann Stability Condition

The sole requirements we have put on the scheme for convergence are consistency and stability. The latter took the form:

$$\|\mathbf{A}^m\| \le C \tag{5.25}$$

where C is independent of Δt, Δx and n. By the matrix norm properties we have:

$$\|\mathbf{A}^m\| \le \|\mathbf{A}\|^m \tag{5.26}$$

hence it is sufficient to require that $\|\mathbf{A}\|^m \le C$, or that

$$\|\mathbf{A}\| \le C^{\frac{1}{m}} = e^{\frac{\Delta t}{t_m}\ln C} = 1 + \frac{\ln C}{t_m}\Delta t + \ldots = 1 + O(\Delta t) \tag{5.27}$$

The Von neumann stability condition is hence that

$$\|\mathbf{A}\| \le 1 + O(\Delta t) \tag{5.28}$$

Note that this stability condition does not make any reference on whether the continuous (exact) solution grows or decays in time. Furthermore, the stability condition is established for finite integration times with the limit $\Delta t \to 0$. In practical computations the computations are necessarily carried out with a small but finite Δt, and it is frequently the case that the evolution equation puts a bound on the growth of the solution. Since the numerical solution and its errors are subject to the same growth factors via the matrix $\mathbf{A}$, it is reasonable, and in most cases essential to require the stronger condition $\|\mathbf{A}\| \le 1$ for stability for non growing solutions.

A final practical detail still needs to be ironed out, namely what norm should be used to measure the error? From the properties of the matrix norm it is immediately clear that the spectral radius $\rho(\mathbf{A}) \le \|\mathbf{A}\|$, hence $\rho(\mathbf{A}) \le 1$ is a necessary condition for stability but not sufficient. There are classes of matrices $\mathbf{A}$ where it is sufficient, for example those that posess a complete set of linear eigenvectors such as those that arise from the discretization of hyperbolic equation. If the 1 or ∞-norms are used the condition for stability becomes sufficient.

Example 10 In the case of the advection equation, the matrix $\mathbf{A}$ given in equation 5.6 has norm:

$$\|\mathbf{A}\|_1 = |\mathbf{A}\|_\infty = |\mu| + |1-\mu| \tag{5.29}$$

For stability we thus require that $|\mu| + |1-\mu| \le 1$. Two cases need to be considered:

1. $0 \le \mu \le 1$: $\|\mathbf{A}\| = \mu + 1 - \mu = 1$, stable.

2. $\mu < 0$: $\|\mathbf{A}\| = 1 - 2\mu > 1$, unstable.

3. $\mu > 1$: $\|\mathbf{A}\| = 1 + 2\mu > 1$, unstable.

The scheme is hence guaranteed to converge when $0 \le \mu \le 1$.

5.5 Von Neumann Stability Analysis

Matrix analysis can be difficult to carry out for complicated PDEs, particularly since it requires us to know the entire spectrum, or norm of the matrix before we can derive useful stability criteria. Von Neumann devised a substantially easier stability test, one that does involve matrices per se but can be reduced to evaluating scalars. The idea is to restrict attention to periodic problems and to consider the Fourier modes of the solution. Since the solution is periodic it can be expanded into a Fourier series of the form:

$$u_j^n = \hat{u}_k^n e^{ikx_j} \tag{5.30}$$

where k is the wavenumber and $\hat{u}_k^n$ is its (complex) Fourier amplitude. This expression can then be inserted back in the finite difference equation, and an expression for the amplification factor A can be obtained, where A depends on k, Δx and Δt. Stability of every Fourier mode will guarantee the stability of the entire solution, and hence $|A| \leq 1$ for all Fourier modes is the necessary and sufficient stability condition for non-growing solutions.

Example 11 Inserting the Fourier series in the finite difference approximation for the advection equation we end up with the following equation:

$$\hat{u}_k^{n+1} e^{ikx_j} = (1-\mu)\hat{u}_k^n e^{ikx_j} + \mu \hat{u}_k^n e^{ikx_{j-1}} \tag{5.31}$$

Since $x_{j-1} = x_j - \Delta x$, the exponential factor drops out of the picture and we end up with the following expression for the growth of the Fourier coefficients:

$$\hat{u}_k^{n+1} = \underbrace{\left[(1-\mu) + \mu e^{-ik\Delta x}\right]}_{A} \hat{u}_k^n \tag{5.32}$$

The expression in bracket is nothing but the amplification factor for Fourier mode k. Stability requires that $|A| < 1$ for all k.

$$\begin{aligned}
|A|^2 &= AA^* = \left[(1-\mu) + \mu e^{-ik\Delta x}\right]\left[(1-\mu) + \mu e^{ik\Delta x}\right] && (5.33)\\
&= (1-\mu)^2 + \mu(1-\mu)(e^{ik\Delta x} + e^{-ik\Delta x}) + \mu^2 && (5.34)\\
&= 1 - 2\mu + 2\mu(1-\mu)\cos k\Delta x + 2\mu^2 && (5.35)\\
&= 1 - 2\mu(1-\cos k\Delta x) + 2\mu^2(1-\cos k\Delta x) && (5.36)\\
&= 1 - 4\sin^2\frac{k\Delta x}{2}(1-\mu)\mu && (5.37)
\end{aligned}$$

It is now clear that $|A|^2 \leq 1$ if $\mu(1-\mu) > 0$, i.e. $0 \leq \mu \leq 1$. It is the same stability criterion derived via the matrix analysis procedure.

5.6 Modified Equation

The truncation error series used to establish the consistency of the scheme can be used to extract additional information about the expected behavior of the numerical scheme.

This is motivated by the observation that the finite difference scheme is in fact solving a perturbed form of the original equation. Equations 5.9 and 5.12 establish that the FTBS scheme approximates the advection equation to first order, $O(\Delta t, \Delta x)$ to the advection equation. They also show that FTBS approximates the following equation to second order in time and space:

$$[v_t + cv_x] = \left[-\frac{\Delta t}{2} v_{tt} + \frac{c\Delta x}{2} v_{xx}\right] + O(\Delta t^2, \Delta x^2). \tag{5.38}$$

The second term on the right hand side of equation 5.38 has the form of a diffusion like operator, and hence we expect it to lead to a gradual decrease in the amplitude of the solution. The interpretation of the time derivative term is not simple. The way to proceed is to derive an expression for v_{tt} in terms of the spatial derivative. This is achieved by differentiating equation 5.38 once with respect to time and once with respect to space to obtain:

$$\begin{aligned} v_{tt} + cv_{xt} &= -\frac{\Delta t}{2} v_{ttt} + \frac{c\Delta x}{2} v_{txx} + O(\Delta t^2 + \Delta x^2) && (5.39)\\ v_{tx} + cv_{xx} &= -\frac{\Delta t}{2} v_{ttx} + \frac{c\Delta x}{2} v_{xxx} + O(\Delta t^2 + \Delta x^2) && (5.40) \end{aligned}$$

Multiplying equation 5.40 by $-c$ and adding it to equation 5.39 we get:

$$v_{tt} = c^2 v_{xx} \frac{\Delta t}{2} \left(-v_{ttt} + cv_{txx}\right) + \frac{c\Delta x}{2} \left(v_{xxt} - cv_{xxx}\right) + O(\Delta t^2, \Delta x^2) \tag{5.41}$$

Inserting this first order approximation to $v_t t$ back in equation 5.38 we obtain the following **modified equation**.

$$v_t + cv_x = \frac{c}{2} \left(\Delta x - c\Delta t\right) v_{xx} + O(\Delta x^2, \Delta x \Delta t, \Delta t^2) \tag{5.42}$$

Equation 5.42 is more informative then its earlier version, equation 5.38. It tells us that the leading error term in $\Delta t, \Delta x$ behaves like a second order spatial derivative whose coefficient is given by the pseudo, or numerical, viscosity ν_n, where

$$\nu_n = \frac{c}{2} \left(\Delta x - c\Delta t\right). \tag{5.43}$$

If $\nu_n > 0$ we expect the solution to be damped to leading order, the numerical scheme behaves as a advection-diffusion equation, one whose viscous coefficient is purely an artifact of the finite difference discretization. If the numerical viscosity is negative, $\nu_n < 0$, the solution will be amplified exponentially fast. The stability condition that $\nu_n > 0$ is nothing but the usual stability criterion we have encountered earlier, namely that $c > 0$ and $\mu = c\Delta t/\Delta x < 1$.

A more careful analysis of the truncation error that includes higher powers of $\Delta t, \Delta x$ yields the following form:

$$v_t + cv_x = \frac{c\Delta x}{2}(1-\mu) v_{xx} - \frac{c\Delta x^2}{6}(2\mu^2 - 3\mu + 1) v_{xxx} + O(\Delta t^3, \Delta t^2 \Delta x, \Delta t \Delta x^2, \Delta x^3) \tag{5.44}$$

The third derivative term is indicative of the presence of dispersive errors in the numerical solution; the magnitude of these errors is amplified by the coefficient multiplying the third order derivative. This coefficient is always negative in the stability region $0 \leq \mu \leq 1$. One can expect a lagging phase error with respect to the analytical solution. Notice also that the coefficients of the higher order derivative on the right hand side term go to zero for $\mu = 1$. This "ideal" value for the time step makes the scheme at least third order accurate according to the modified equation; in fact it is easy to convince one self on the basis of the characteristic analysis that the exact solution is recovered.

Notice that the derivation of the modified equation uses the Taylor series form of the finite difference scheme, equation 5.9, rather then the original partial differential equations to derive the estimates for the high order derivative. This is essential to account for the discretization errors. The book by Tannehill et all 1997 discusses a systematic procedure for deriving the higher order terms in the modified equation.

Chapter 6

Numerical Solution of the Advection Equation

6.1 Introduction

We devote this chapter to the application of the notions discussed in the previous chapter to investigate several finite difference schemes to solve the simple advection equation. This equation was taken as an example to illustrate the abstract concepts that frame most of the discussion on finite difference methods from a theoretical perspective. These concepts we repeat are consistency, convergence and stability. We will investigate several common schemes found in the literature, and we will investigate their amplitude and phase errors more closely.

6.2 Donor Cell scheme

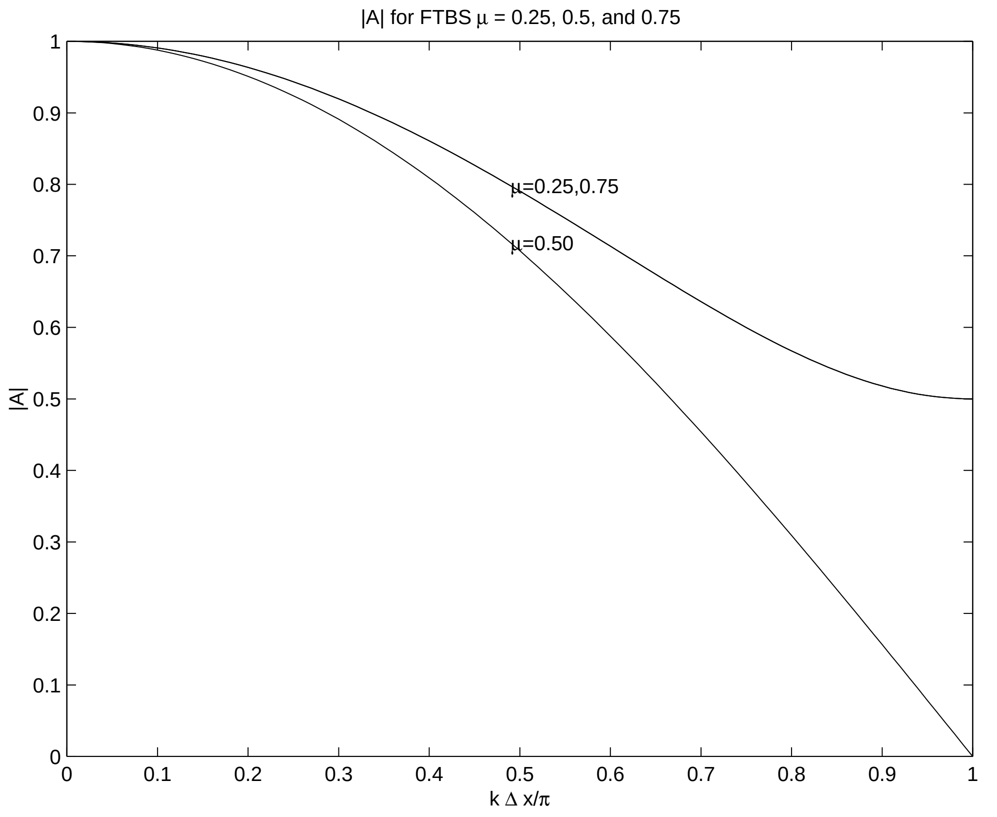
The donor cell scheme is essentially the FTBS scheme seen earlier. The only distinguishing feature of the donor-cell scheme is that it allows the switching of the spatial finite difference according to the sign of the advecting velocity c. A compact way of writing the scheme is:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + \frac{c + |c|}{2}\frac{u_j^n - u_{j-1}^n}{\Delta x} + \frac{c - |c|}{2}\frac{u_{j+1}^n - u_j^n}{\Delta x} = 0 \tag{6.1}$$

For $c > 0$ the scheme simplifies to a FTBS, and for $c < 0$ it becomes a FTFS (forward time and forward space) scheme. Here we will consider solely the case $c > 0$ to simplify things. Figure 6.1 shows plots of the amplification factor for the donor cell scheme. Prior to discussing the figures we would like to make the following remarks.

6.2.1 Remarks

1. The scheme is **conditionally stable** since the time step cannot be chosen independently of the spatial discretization and must satisfy $\Delta t \leq \Delta t_{\max} = c/\Delta x$.



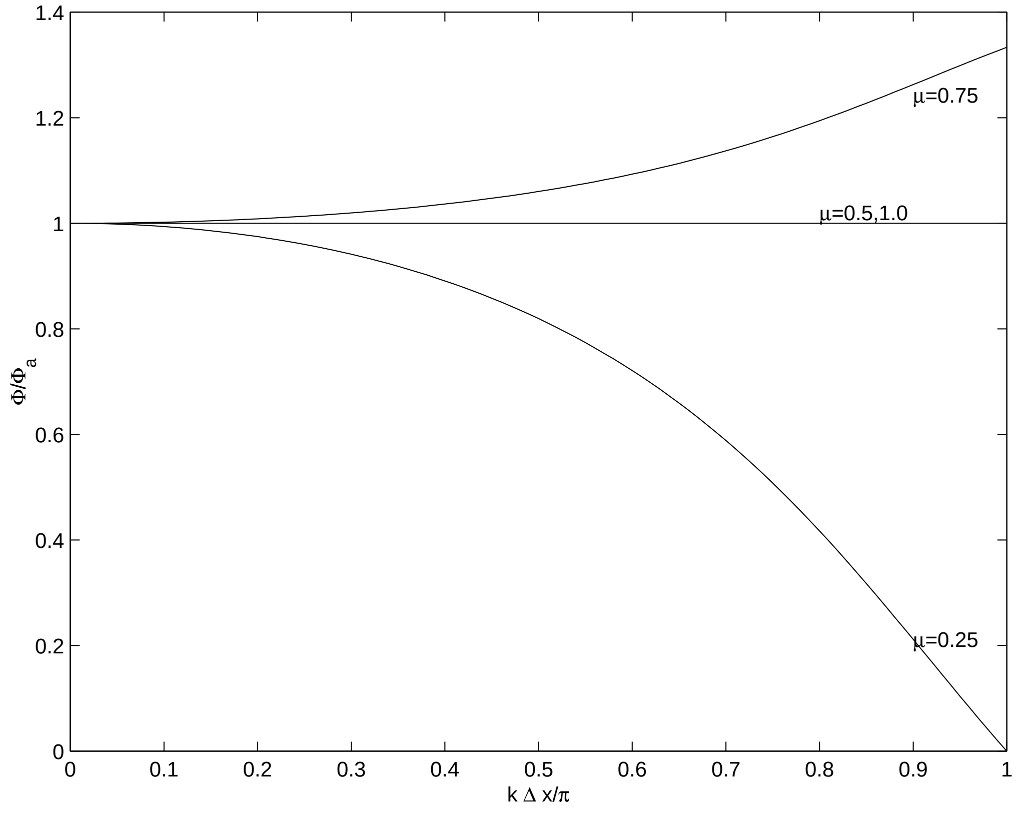


Figure 6.1: Amplitude and phase diagram of the donor cell scheme as a function of the wavenumber

2. The wavelength appearing in the Von-Neumann stability analysis has not been specified yet. Small values of k correspond to very long wavelegths, i.e. Fourier modes that are well represented on the computational grid. Large values of k correspond to very short wavelength. This correspondence is evident by the expression $k\Delta x = 2\pi\Delta x/\lambda$, where λ is the wavelength of the Fourier mode. For example, a twenty kilometers wave represented on a grid with $\Delta x = 2$ kilometers would have 10 points per wavelength and its $k\Delta x = 2\pi 2/10 = 2\pi/5$.

3. There is an lower limit on the value of the shortest wave representable on a discrete grid. This wave has a wavelength equal to $2\Delta x$ and takes the form of a see-saw function; its $k\Delta x = \pi$. Any wavelength shorter then this limit will be **aliased** into a longer wavelegth. This phenomenon is similar to the one encountered in the Fourier analysis of time series where the **Nyquist** limit sets a lower bound to the smallest measurable wave period.

4. In the previous chapter we have focussed primarily on the magnitude of the amplification factor as it is the one that impacts the issue of stability. However, additional information is contained in the expression for the amplification factor that relates to the dispersive properties of the finite difference scheme. The *analytical* expression for the amplification factor for a Fourier mode is

$$A_a = e^{-ick\Delta t}. \tag{6.2}$$

Thus the analytical solution expects a unit amplification per time step, $|A_a| = 1$, and a change of phase of $\phi_a = -ck\Delta t = -\mu k\Delta x$. The amplification factor for the donor cell scheme is however:

$$\begin{aligned} A &= |A|e^{i\phi}, && (6.3)\\ |A| &= 1 - \mu(1-\mu)4\sin^2\frac{k\Delta x}{2}, && (6.4)\\ \phi &= \tan^{-1}\frac{\mu\sin k\Delta x}{1-\mu(1-\cos k\Delta x)} && (6.5)\end{aligned}$$

where ϕ is the argument of the complex number A. The ratio of ϕ/ϕ_a gives the relative error in the phase. A ratio less then 1 means that the numerical phase error is less then the analytical one, and the scheme is decelerating, while a ratio greater then indicates an accelerating scheme. We will return to phase errors later when we look at the dispersive properties of the scheme.

5. The donor cell scheme for c positive can be written in the form:

$$u_j^{n+1} = (1-\mu)u_j^n + \mu u_{j-1}^n \tag{6.6}$$

which is a linear, convex (for $0 \le \mu \le 1$), combination of the two values at the previous time levels upstream of the point (j, n). Since the two factors are positive we have

$$\min(u_j^n, u_{j-1}^n) \le u_j^{n+1} \le \max(u_j^n, u_{j-1}^n), \tag{6.7}$$

In plain words the value at the next time level cannot exceed the maximum of the two value upstream, nor be less then the minimum of these two values. This is referred to as the **monotonicity** property. It plays an important role in devising scheme which do not generate spurious oscillation because of under-resolved gradients. We will return to this point several times when discussing dispersive errors and special advection schemes.

Figure 6.1 shows $|A|$ and ϕ/ϕ_a for the donor cell scheme as a function of $k\Delta x$ for several values of the Courant number μ. The long waves (small $k\Delta x$ are damped the least for $0 \leq \mu \leq 1$ whereas high wave numbers $k\Delta x \to 0$ are damped the most. The most vigorous damping occurs for the shortest wavelength for $\mu = 1/2$, where the donor-cell scheme reduces to an average of the two upstream value, the amplification factor magnitude is then $|A| = 0$, i.e. $2\Delta x$ waves are eliminated after a single time step. The amplification curves are symmetric about $\mu = 1/2$, and damping lessens as μ becomes smaller for a fixed wavelength. The dispersive errors are small for long waves; they are decelerating for all wavelengths for $\mu < 1/2$ and accelerating for $1/2 \leq \mu \leq 1$; they reach their peak acceleration for $\mu = 3/4$.

6.3 Backward time centered space (BTCS)

In this scheme the terms in the equations are evaluated at time $(n+1)$. For a two time level scheme this translates into a backward euler difference for the time derivative. We use a centered difference in space to increase the order of the spatial approximation. This leads to the equations:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_{j+1}^{n+1} - u_{j-1}^{n+1}}{2\Delta x} = 0 \tag{6.8}$$

6.3.1 Remarks

1. **truncation error** The Taylor series analysis (expansion about time level $n+1$) leads to the following equation:

$$u_t + cu_x = -\left(-\frac{\Delta t}{2}u_{tt} + \frac{c\Delta x^2}{3}u_{xxx} + \frac{\Delta t^2}{6}u_{ttt}\right) + O(\Delta t^3, \Delta x^4) \tag{6.9}$$

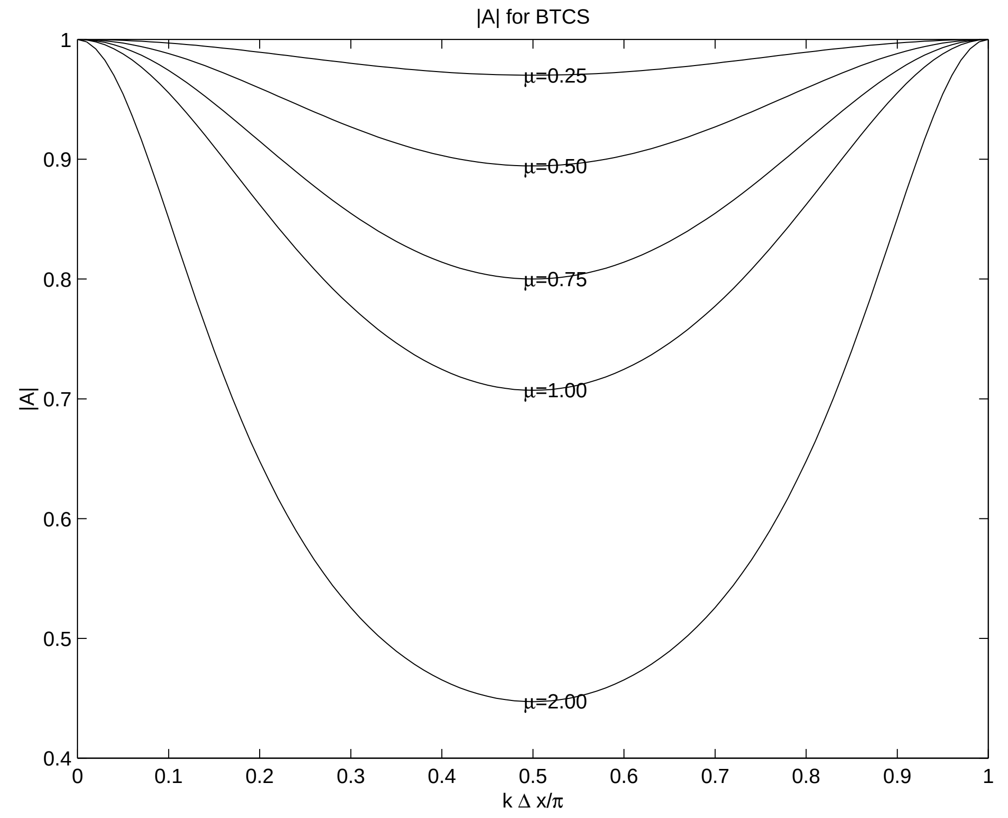
   The leading truncation error term is $O(\Delta t, \Delta x^2)$, and hence the scheme is first order in time and second order in space. Moreover, the truncation error goes to zero for $\Delta t, \Delta x \to 0$, and hence the scheme is **consistent**.

2. The Von Neumann stability analysis leads to the following amplification factor:

$$A = \frac{1 - i\mu \sin k\Delta x}{1 + \mu^2 \sin^2 k\Delta x} \tag{6.10}$$

$$|A| = \frac{1}{\sqrt{1 + \mu^2 \sin^2 k\Delta x}} < 1, \text{ for all } \mu, k\Delta x \tag{6.11}$$

$$\frac{\phi}{\phi_a} = \frac{\tan^{-1}(-\mu \sin k\Delta x)}{-\mu k\Delta x} \tag{6.12}$$



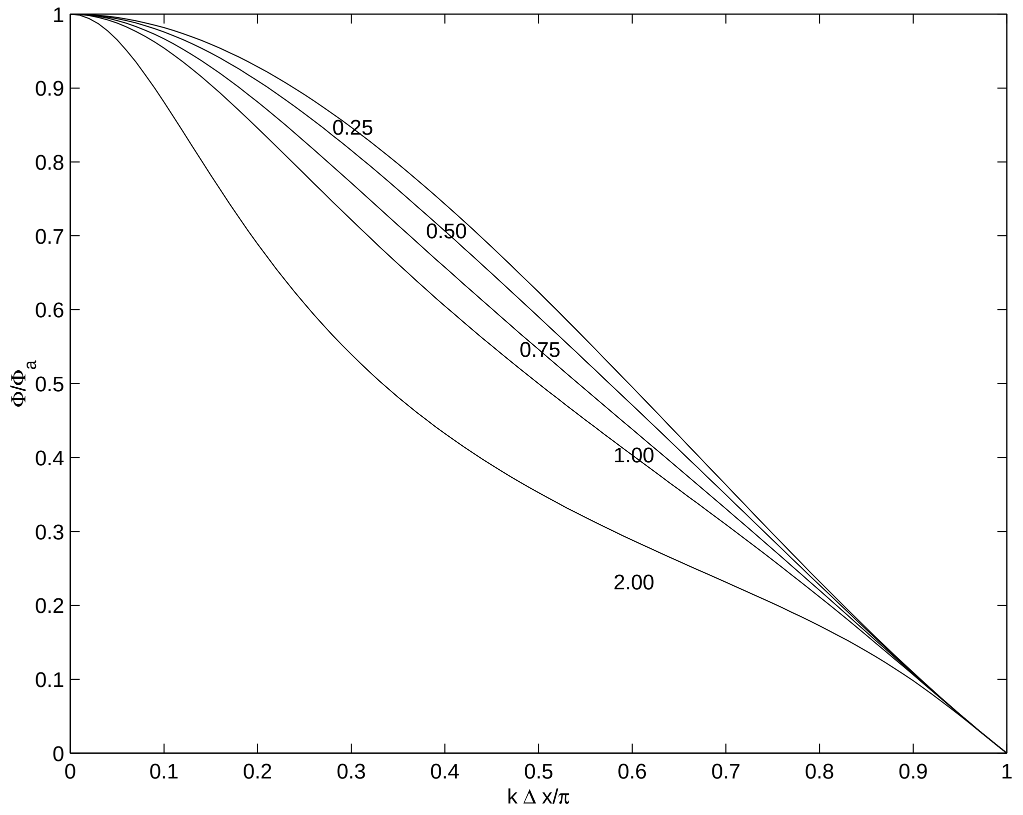


Figure 6.2: Amplitude and phase diagrams of the BTCS scheme as a function of the wavenumber

The scheme is **unconditionally stable** since $|A| < 1$ irrespective of the time step Δt. By the Lax-Richtmeyer theorem the consistency and stability of the scheme guarantee it is also **convergent**.

3. The modified equation for BTCS is

$$u_t + cu_x = \frac{c^2\Delta t}{2}u_{xx} - \left(\frac{c\Delta x^2}{6} + \frac{c^3}{6}\Delta t^2\right)u_{xxx} + \dots \tag{6.13}$$

The numerical viscosity is hence always positive and lends the scheme its stable and damping character. Notice that the damping increasing with increasing c and Δt.

4. Notice that the scheme cannot update the solution values a grid point at a time, since the values u_j^{n+1} and $u_{j\pm1}^{n+1}$ are unknown and must be determined simultaneously. This is an example of an **implicit** scheme which requires the inversion of a system of equation. Segregating the unknowns on the left hand side of the equation we get:

$$-\frac{\mu}{2}u_{j-1}^{n+1} + u_j^{n+1} + \frac{\mu}{2}u_{j+1}^{n+1} = u_j^n \tag{6.14}$$

which consititutes a matrix equation for the vector of unknowns at the next time level. The equation in matrix forms are:

$$\begin{pmatrix} & & & & & \\ \frac{-\mu}{2} & 1 & \frac{\mu}{2} & & & \\ & \frac{-\mu}{2} & 1 & \frac{\mu}{2} & & \\ & 0 & \frac{-\mu}{2} & 1 & \frac{\mu}{2} & \\ & & & & & \end{pmatrix} \begin{pmatrix} u_1 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_N \end{pmatrix}^{n+1} = \begin{pmatrix} u_1 \\ \vdots \\ u_{j-1} \\ u_j \\ u_{j+1} \\ \vdots \\ u_N \end{pmatrix}^{n} \tag{6.15}$$

The special structure of the matrix is that the only non-zero entries are those along the diagonal, and on the first upper and lower diagonals. This special structure is referred to as a tridiagonal matrix. Its inversion is far cheaper then that of a full matrix and can be done in $O(N)$ addition and multiplication through the **Thomas** algorithm for tridiagonal matrices; in contrast, a full matrix would require $O(N^3)$ operations. Finally, the first and last rows of the matrix have to be modified to take into account boundary conditions. We will return to the issue of boundary conditions later.

Figures 6.2 shows the magnitude of the amplification factor $|A|$ for several Courant numbers. The curves are symmetric about $k\Delta x = \pi/2$. The high and low wave numbers are the least damped whereas the intermediate wave numbers are the most damped. The departure of $|A|$ from 1 deteriorates with increasing μ. Finally the scheme is decelarating for all wavenumbers and Courant numbers, and the deceleration deteriorates for the shorter wavelengths.

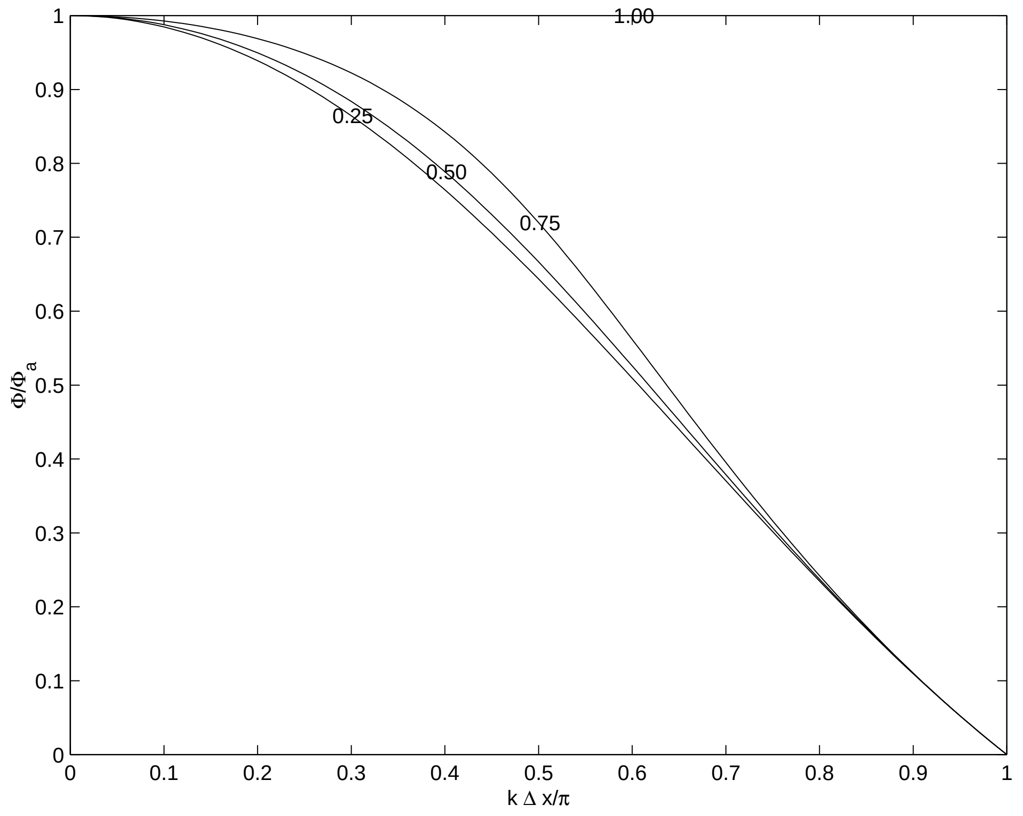


Figure 6.3: Phase diagrams of the CTCS scheme as a function of the wavenumber

6.4 Centered time centered space (CTCS)

A simple and popular explicit second order scheme in space and time is the centered time and centered space scheme. This is a **three time level** scheme and takes the form:

$$\frac{u_j^{n+1} - u_j^{n-1}}{2\Delta t} + c\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} = 0 \qquad (6.16)$$

6.4.1 Remarks

1. **truncation error** The Taylor series analysis leads to the following equation:

$$u_t + cu_x = -\left(-\frac{\Delta t^2}{3}u_{ttt} + \frac{c\Delta x^2}{3}u_{xxx}\right) + O(\Delta t^4, \Delta x^4) \qquad (6.17)$$

The leading truncation error term is $O(\Delta t^2, \Delta x^2)$, and hence the scheme is second order in time and space. Moreover, the truncation error goes to zero for $\Delta t, \Delta x \rightarrow 0$, and hence the scheme is **consistent**.

2. The Von Neumann stability analysis leads to a quadratic equation for the amplification factor: $A^2 + 2\mu \sin k\Delta x A - 1 = 0$. Its two solutions are:

$$\begin{aligned} A_{\pm} &= -i\mu \sin k\Delta x \pm \sqrt{1-\mu^2 \sin^2 k\Delta x} && (6.18)\\ |A_{\pm}| &= 1,\ \text{for all } |\mu| < 1 && (6.19)\\ \frac{\phi}{\phi_a} &= \frac{1}{-\mu k\Delta x} \tan^{-1} \frac{-\mu \sin k\Delta x}{\sqrt{1-\mu^2 \sin^2 k\Delta x}} && (6.20) \end{aligned}$$

   The scheme is **conditionally stable** for $|\mu| < 1$, and its amplification is **neutral** since $|A| = 1$ within the stability region. An attribute of the CTCS scheme is that its amplification factor mirror the neutral amplification of the analytical solution. By the Lax-Richtmeyer theorem the consistency and stability of the scheme guarantee it is also **convergent**.

3. The modified equation for CTCS is

$$u_t + cu_x = \frac{c\Delta x^2}{6}(\mu^2 - 1)u_{xxx} - \frac{c\Delta x^4}{120}(9\mu^4 - 10\mu^2 + 1)u_{xxxxx} + \dots \qquad (6.21)$$

   The even derivative are absent from the modified equation indicating the total absence of numerical dissipation. The only errors are dispersive in nature due to the presence of odd derivative in the modified equation.

4. The model requires a starting procedure to kick start the computations. It also has a computational mode that must be damped.

5. Figures 6.3 shows the phase errors for CTCS for several Courant numbers. All wave numbers are decelerating and the shortest wave are decelerated more then the long waves.

6.5 Lax Wendroff scheme

The idea behind the Lax-Wendroff scheme is to keep the simplicity of two time level schemes while attempting to increase the order of accuracy in space and time to second order. This is possible if derivatives in time are translated to derivatives in space. The Taylor series in time about time level n is:

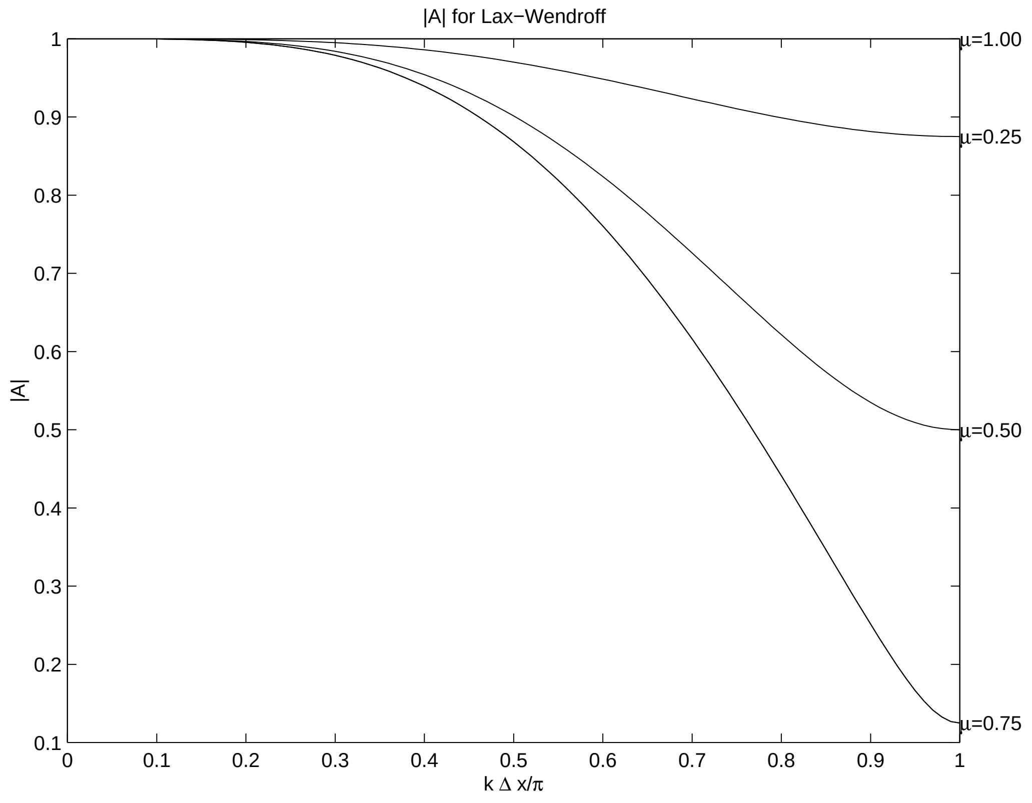
$$u_j^{n+1} = u_j^n + \Delta t u_t + \frac{\Delta t^2}{2} u_{tt} + \frac{\Delta t^3}{6} u_{ttt} \qquad (6.22)$$

From the PDE we know that $u_t = -cu_x$. What we need to complete the second order accuracy in time is a second order expression for u_{tt}. This can be obtained by differentiating the advection equation with respect to time to yield:

$$u_{tt} = -cu_{xt} = -c(u_t)_x = -c(-cu_x)_x = c^2 u_{xx} \qquad (6.23)$$

The Taylor series in time takes the form:

$$u_j^{n+1} = u_j^n - c\Delta t u_x + \frac{c^2 \Delta t^2}{2} u_{xx} + \frac{\Delta t^3}{6} u_{ttt} \qquad (6.24)$$



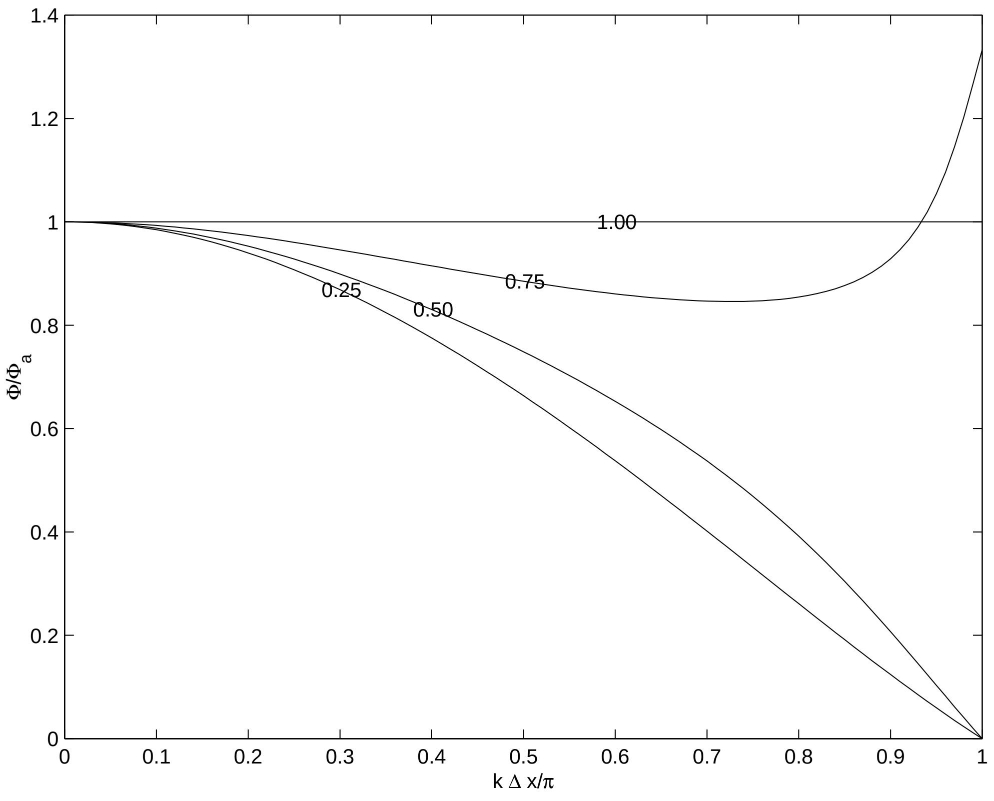


Figure 6.4: Amplitude and phase diagrams of the Lax-Wendroff scheme as a function of the wavenumber

All that remains to be done is to use high order approximations for the spatial derivatives u_x and u_xx. We use centered derivatives for both terms as they are second order accurate to get the final expression:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x} + \frac{c^2\Delta t}{2}\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} = 0 \tag{6.25}$$

6.5.1 Remarks

1. **truncation error** The Taylor series analysis (expansion about time level $n+1$ leads to the following equation:

$$u_t + cu_x = -\left(-\frac{\Delta t^2}{3}u_{ttt} + \frac{c\Delta x^2}{3}u_{xxx}\right) + O(\Delta t^4, \Delta x^4) \tag{6.26}$$

   The leading truncation error term is $O(\Delta t^2, \Delta x^2)$, and hence the scheme is second order in time and space. Moreover, the truncation error goes to zero for $\Delta t, \Delta x \to 0$, and hence the scheme is **consistent**.

2. The Von Neumann stability analysis leads to:

$$\begin{aligned} A &= 1 - \mu^2(1 - \cos k\Delta x) - i\mu \sin k\Delta x && (6.27)\\ |A|^2 &= [1 - \mu^2(1 - \cos k\Delta x)]^2 + \mu^2 \sin^2 k\Delta x, && (6.28)\\ \frac{\phi}{\phi_a} &= \frac{1}{-\mu k\Delta x}\tan^{-1}\frac{-\mu \sin k\Delta x}{1 - \mu^2(1 - \cos k\Delta x)} && (6.29) \end{aligned}$$

   The scheme is **conditionally stable** for $|\mu| < 1$. By the Lax-Richtmeyer theorem the consistency and stability of the scheme guarantee it is also **convergent**.

3. The modified equation for Lax Wendroff is

$$u_t + cu_x = \frac{c\Delta x^2}{6}(\mu^2 - 1)u_{xxx} - \frac{c\Delta x^3}{8}\mu(1 - \mu^2)u_{xxxx} + \dots \tag{6.30}$$

4. Figures 6.4 shows the amplitude and phase errors for the Lax Wendroff schemes. The phase errors are predominantly lagging, the only accelerating errors are those of the short wave at relatively high values of the Courant number.

6.6 Numerical Dispersion

Consistency and stability are the first issues to consider when contemplating the solution of partial differential equations. They address the theoretical questions of convergence in the limit of improving resolution. They should not be the last measure of performance, however, as other error measures can be of equal importance. For hyperbolic equations, where wave dynamics are important, it is critical to consider the distortion of wave propagation by the numerical scheme. Although, we have looked at the phase characteristic of the scheme derived so far, it was hard to get an intuitive feel for the impact on the wave propagation characteristics. The aim of this section is to address the issue of the numerical dispersion relation, and to compare it to the dispersion relation of the continuous equations. We start with the latter.

6.6.1 Analytical Dispersion Relation

The analytical dispersion relation for the wave equation can be obtained by looking for periodic solutions in space and time of the form $\tilde{u}e^{i(kx-\omega t)}$ where k is the wavenumber and ω the corresponding frequency. Inserting this expression in equation 5.3 we get the dispersion relation:

$$\omega = ck \tag{6.31}$$

The associate phase speed, C_p, and group velocity, C_g, of the system is as follows

$$C_p = \frac{\omega}{k} = c \tag{6.32}$$

$$C_g = \frac{\partial\omega}{\partial k} = c \tag{6.33}$$

The two velocities are constant and the system is **non-dispersive**, i.e. all waves travels with the same phase speed regardless of wavenumber. The group velocity is also constant in the present case and reflects the speed of energy propagation. One can anticipate that this property will be violated in the numerical discretization process based on what we know of the phase error plots; there it was shown that phase errors are different for the different wave number. We will make this assertion clearer by looking at the **numerical dispersion** relation.

6.6.2 Numerical Dispersion Relation: Spatial Differencing

To keep the algebra tractable, we assume that only the spatial dimension is discretized and the time dimenion is kept continuous. The semi-discrete form of the following schemes are:

1. **Centered second order scheme**

$$u_t + \frac{u_{j+1} - u_{j-1}}{2\Delta x} = 0 \tag{6.34}$$

2. **Centered fourth order scheme**

$$u_t + \frac{8(u_{j+1} - u_{j-1}) - (u_{j+2} - u_{j-2})}{12\Delta x} = 0 \tag{6.35}$$

3. **Centered sixth order scheme**

$$u_t + \frac{45(u_{j+1} - u_{j-1}) - 9(u_{j+2} - u_{j-2}) + (u_{j+3} - u_{j-3})}{60\Delta x} = 0 \tag{6.36}$$

4. **Donor cell**

$$u_t + \frac{u_j - u_{j-1}}{\Delta x} = 0 \tag{6.37}$$

5. **Third order upwind**

$$u_t + \frac{2u_{j+1} + 3u_j - 6u_{j-1}) + u_{j-2}}{6\Delta x} = 0 \tag{6.38}$$

The dispersion for these numerical scheme can be derived also on the basis of periodic solution of the form $u_j = \tilde{u}e^{i(kx_j - \sigma)}$. The biggest difference is of course that the Fourier expansion is discrete in space. The following expression for the phase velocity can be derived for the different schemes:

$$\begin{array}{ll}
\text{CD2} & \dfrac{\sigma}{k} = c\,\dfrac{\sin k\Delta x}{k\Delta x} \\
\text{CD4} & \dfrac{\sigma}{k} = c\,\dfrac{8\sin k\Delta x - \sin 2k\Delta x}{6k\Delta x} \\
\text{CD6} & \dfrac{\sigma}{k} = c\,\dfrac{\sin 3k\Delta x - 9\sin 2k\Delta x + 45\sin k\Delta x}{30k\Delta x} \\
\text{Donor} & \dfrac{\sigma}{k} = c\,\dfrac{\sin k\Delta x - i(1-\cos k\Delta x)}{k\Delta x} \\
\text{Third Upwind} & \dfrac{\sigma}{k} = c\,\dfrac{(8\sin k\Delta x - \sin 2k\Delta x) - i2(1-\cos k\Delta x)^2}{k\Delta x}
\end{array} \tag{6.39}$$

Several things stand out in the numerical dispersion of the various schemes. First, all of them are dispersive, and hence one expects that a wave form made up of the sum of individual Fourier components will evolve such that the fast travelling wave will pass the slower moving ones. Second, all the centered difference scheme show a real frequency, i.e. they introduce no amplitude errors. The off-centered schemes on the other hand have real and imaginary parts. The former influences the phase speed whereas the former influences the amplitude. The amplitude decays if $\mathcal{I}m(\sigma) < 0$, and increases for $\mathcal{I}m(\sigma) > 0$. Furthermore, the upwind biased schemes have the same real part as the next higher order centered scheme; thus their dispersive properties are as good as the higher order centered scheme except for the damping associated with their imaginary part (this is not necessarily a bad things at least for the short waves).

Figure 6.5 shows the dispersion curve for the various scheme discussed in this section versus the analytical dispersion curve (the solid straight line). One can immediately see the impact of higher order spatial differencing in improving the propagation characteristics of the intermediate wavenumber range. As the order is increased, the dispersion curves rise further towards the analytical curve, particularly near $k\Delta x > \pi/2$, hence a larger portion of the spectrum is propagating correctly. The lower panel shows the impact of biasing the differencing towards the upstream side. The net effect is the introduction of numerical dissipation. The latter is strongest for the short waves, and decreases with the order of the scheme.

Figure 6.6 shows the phase speed (upper panel) and group velocity (lower panel) of the various schemes. Again it is evident that a larger portion of the wave spectrum is propagating correctly whereas as the order is increased. None of the schemes allows the shortest wave to propagate. The same trend can be seen for the group velocity plot. However, the impact of the numerical error is more dramatic there since the short waves have negative group velocities, i.e. they are propagating in the opposite direction. This trend worsens as the accuracy is increased.

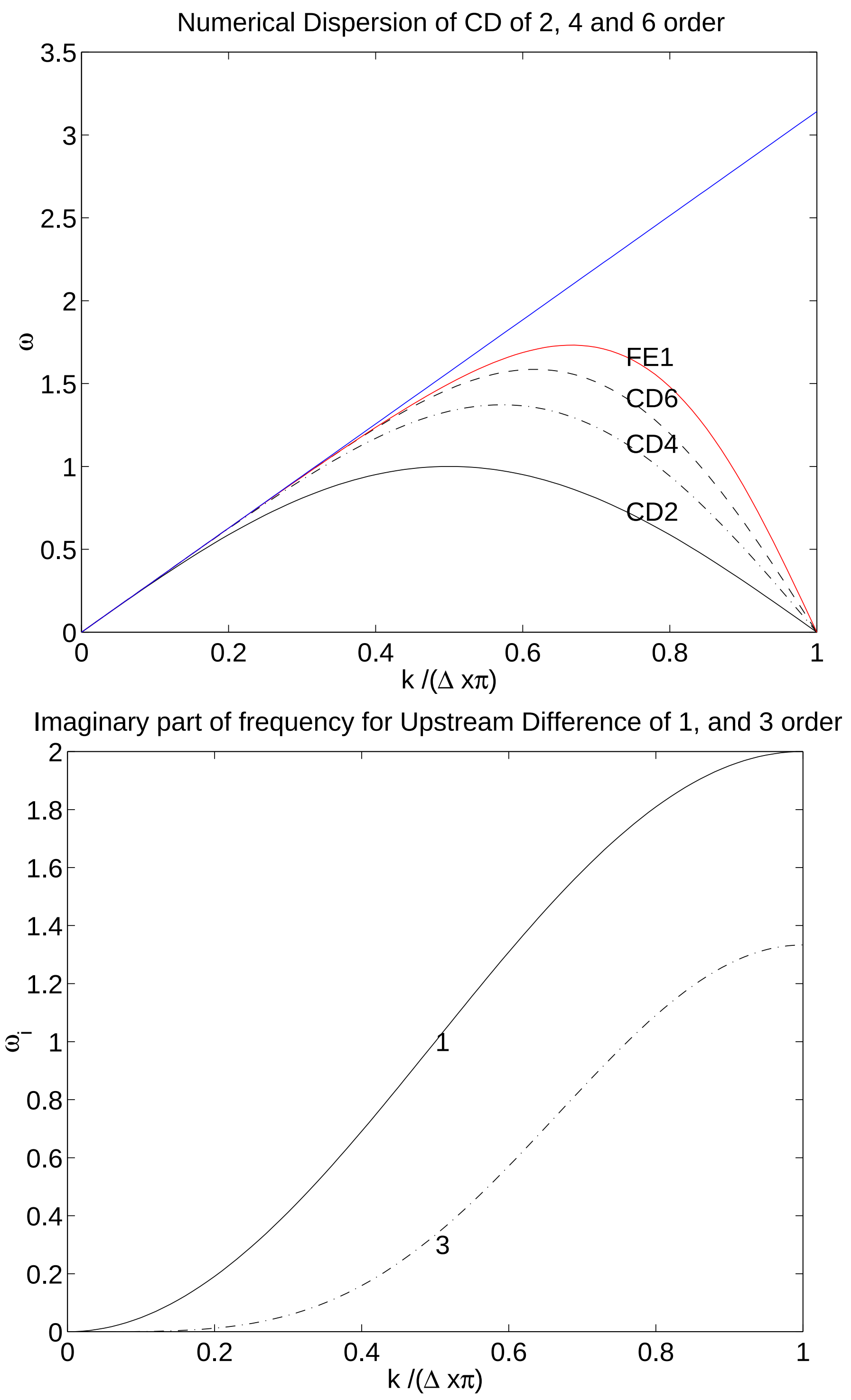


Figure 6.5: Dispsersion relation for various semi-discrete schemes. The upper panel shows the real part of the frequency whereas the bottom panel shows the imaginary part for the first and third order upwind schemes. The real part of the frequency for these two schemes is identical to that of the second and fourth order centered schemes.

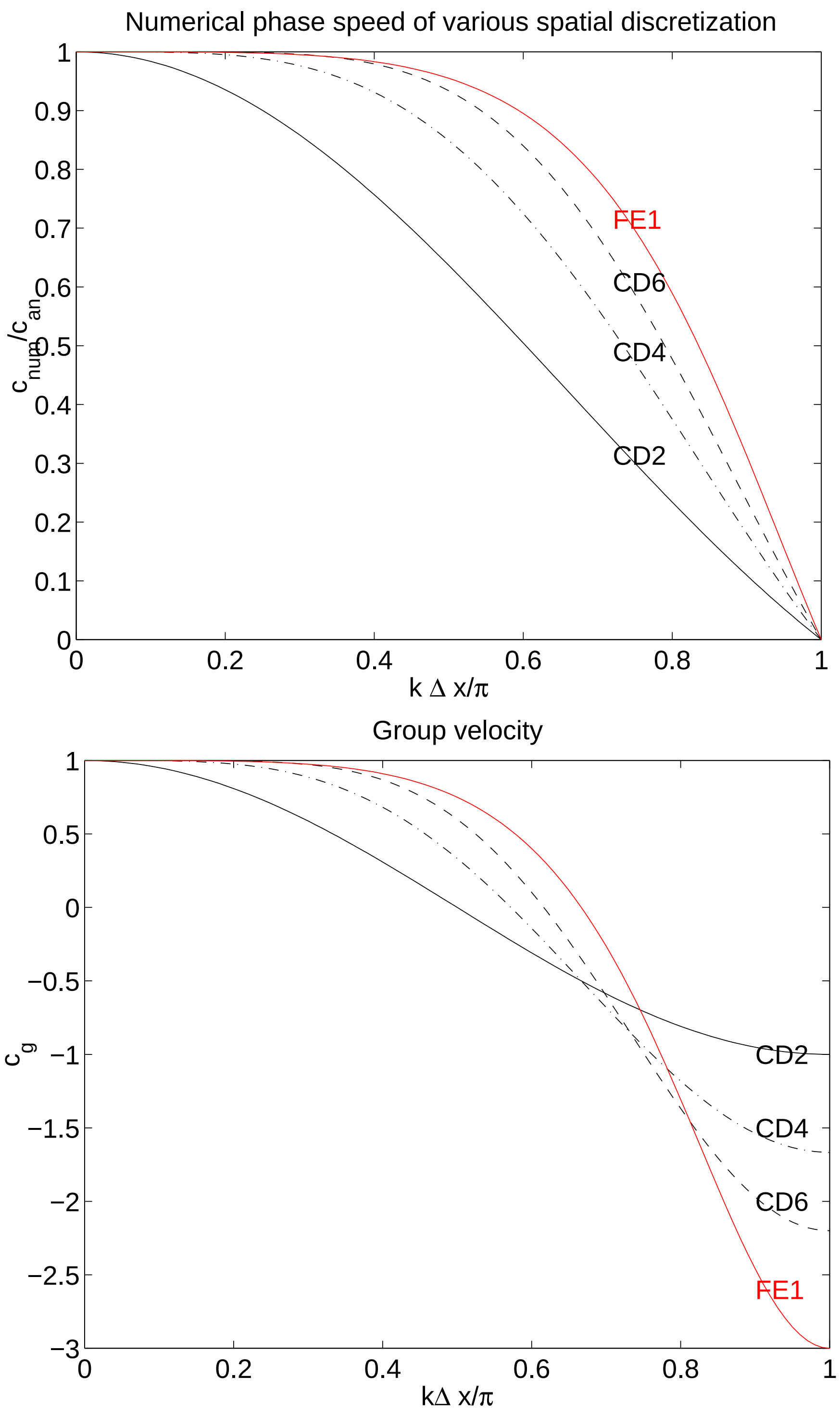


Figure 6.6: Phase speed (upper panel) and Group velocity (lower panel) for various semi-discrete schemes.

Chapter 7

Numerical Dispersion of Linearized SWE

This chapter is concerned with the impact of FDA and variable staggering on the fidelity of wave propagation in numerical models. We will use the shallow water equations as the model equations on which to compare various approximations. These equations are the simplest for describing wave motions in the ocean and atmosphere, and they are simple enough to be tractable with pencil and paper. By comparing the dispersion relation of the continuous and discrete systems, we can decide which scales of motions are faithfully represented in the model, and which are distorted. Conversely the diagrams produced can be used to decide on the number of points required to resolve specific wavelengths. The two classes of wave motions encountered here are inertia-gravity waves, and Rossby waves. The main reference for the present work is Dukowicz (1995).

The plan is to look at dynamical system of increasing complexity in order to highlight various aspects of the discrete systems. We start by looking at 1D versions of the linearized shallow water equations, and unstaggered and staggered versions of the discrete approximation; in particular we constrast these two approaches for several high order centered difference scheme and show the superiority of the staggered system. Second we look at the impact of including a second spatial dimensional and include rotation but restrict ourselves to second order schemes; the discussion is instead focussed on the various staggering on the dispersive properties. Lastly we look at the dispersive relation for the Rossby waves.

7.1 Linearized SWE in 1D

Since we are interested in applying Fourier analysis to study wave propagations, we need to linearize the equations and hold the coefficients of the PDE to be constant. For the shallow water equations, they are:

$$u_t + g\eta_x = 0 \tag{7.1}$$

$$\eta_t + Hu_x = 0 \tag{7.2}$$

7.1.1 Centered FDA on A-grid

The straight forward approach to discretizing the shallow water equation in space is to replace the continuous partial derivatives by their discrete counter-parts. The main question is what impact do the choice of variable staggering have on the dispersion relationship. We start by looking at the case where u and η are co-located. We also restrict ourselves at centered approximation to the spatial derivatives which have the form:

$$u_x|_j = \frac{\sum_{m=1}^{M} \alpha_m (u_{j+m} - u_{j-m})}{2\Delta x} + O(\Delta x^{2M}) \tag{7.3}$$

where M is the width of the stencil; the α_m are coefficients that can be obtained from the Taylor series expansions (see equations 3.24,3.27, and 3.29. The order of the neglected term on an equally spaced grid is $O(\Delta x)^{2M}$. A similar representation holds for the η derivative. The semi-discrete form of the equation is then:

$$u_t|_j + g\frac{\sum_{m=1}^{M} \alpha_m (\eta_{j+m} - \eta_{j-m})}{2\Delta x} = 0 \tag{7.4}$$

$$\eta_t|_j + H\frac{\sum_{m=1}^{M} \alpha_m (u_{j+m} - u_{j-m})}{2\Delta x} = 0 \tag{7.5}$$

To compute the numerical dispersion associated with the spatially discrete system, we need to look at periodic solution in space and time, and thus we set

$$\begin{pmatrix} u_j \\ \eta_j \end{pmatrix} = \begin{pmatrix} \hat{u} \\ \hat{\eta} \end{pmatrix} e^{i(kx_j - \sigma t)} \tag{7.6}$$

Hence we have for the time derivative $u_t = -\sigma \hat{u} e^{i(kx_j - \sigma t)}$, and for the spatially discrete derivative

$$u_{j+m} - u_{j-m} = \hat{u}\left[e^{i(kx_{j+m} - \sigma t)} - e^{i(kx_{j-m} - \sigma t)}\right] = \hat{u} e^{i(kx_j - \sigma t)} 2i \sin mk\Delta x; \tag{7.7}$$

Hence the FDA of the spatial derivative has the following expression

$$\sum_{m=1}^{M} \alpha_m (u_{j+m} - u_{j-m}) = 2i\ \hat{u}\ e^{i(kx_j - \sigma t)} \sum_{m=1}^{M} \alpha_m \sin mk\Delta x \tag{7.8}$$

Similar expressions can be written for the η variables. Inserting the periodic solutions in 7.5 we get the homogeneous system of equations for the amplitudes $\hat{u}$ and $\hat{\eta}$:

$$-i\sigma\hat{u} + gi\left(\sum_{m=1}^{M} \alpha_m \frac{\sin mk\Delta x}{\Delta x}\right)\hat{\eta} = 0 \tag{7.9}$$

$$Hi\left(\sum_{m=1}^{M} \alpha_m \frac{\sin mk\Delta x}{\Delta x}\right)\hat{u} - i\sigma\hat{\eta} = 0 \tag{7.10}$$

For non-trivial solution we require the determinant of the system to be equal to zero, a condition that yields the following dispersion relation:

$$\sigma = \pm c\left(\sum_{m=1}^{M} \alpha_m \frac{\sin mk\Delta x}{\Delta x}\right) \tag{7.11}$$

where $c = \sqrt{gH}$ is the gravity wave speed of the continuous system. The phase speed is then

$$C_{A,M} = c\left(\sum_{m=1}^{M} \alpha_m \frac{\sin mk\Delta x}{k\Delta x}\right) \tag{7.12}$$

and clearly the departure of the term in bracket from unity determines the FDA phase fidelity of a given order M. We thus have the following relations for schemes of order 2, 4 and 6:

$$C_{A,2} = c\frac{\sin k\Delta x}{k\Delta x} \tag{7.13}$$

$$C_{A,4} = c\frac{8\sin k\Delta x - \sin 2k\Delta x}{6k\Delta x} \tag{7.14}$$

$$C_{A,6} = c\frac{45\sin k\Delta x - 9\sin 2k\Delta x + \sin 3k\Delta x}{30k\Delta x} \tag{7.15}$$

7.1.2 Centered FDA on C-grid

When the variables are staggered on a C-grid, the spatially discrete equations take the following form

$$u_t|_{j+\frac{1}{2}} + g\frac{\sum_{m=0}^{M}\beta_m\left(\eta_{j+\frac{1}{2}+\frac{1+2m}{2}} - \eta_{j+\frac{1}{2}-\frac{1+2m}{2}}\right)}{\Delta x} = 0 \tag{7.16}$$

$$\eta_t|_j + H\frac{\sum_{m=0}^{M}\beta_m(u_{j+\frac{1+2m}{2}} - u_{j-\frac{1+2m}{2}})}{\Delta x} = 0 \tag{7.17}$$

where β_m's are the differentiation coefficients on a staggered grid. These can be obtained from applying the expansion in 3.24 to grids of spacing $(2*m+1)\Delta x/2$ with $m = 0, 1, 2, \ldots$ to get:

$$\frac{u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}}{\Delta x} = \left.\frac{\partial u}{x}\right|_j + \frac{(\Delta x/2)^2}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{(\Delta x/2)^4}{5!}\frac{\partial^5 u}{\partial x^5} + \frac{(\Delta x/2)^6}{7!}\frac{\partial^7 u}{\partial x^7} \tag{7.18}$$

$$\frac{u_{j+\frac{3}{2}} - u_{j-\frac{3}{2}}}{3\Delta x} = \left.\frac{\partial u}{x}\right|_j + \frac{(3\Delta x/2)^2}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{(3\Delta x/2)^4}{5!}\frac{\partial^5 u}{\partial x^5} + \frac{(3\Delta x/2)^6}{7!}\frac{\partial^7 u}{\partial x^7} \tag{7.19}$$

$$\frac{u_{j+\frac{5}{2}} - u_{j-\frac{5}{2}}}{5\Delta x} = \left.\frac{\partial u}{x}\right|_j + \frac{(5\Delta x/2)^2}{3!}\frac{\partial^3 u}{\partial x^3} + \frac{(5\Delta x/2)^4}{5!}\frac{\partial^5 u}{\partial x^5} + \frac{(5\Delta x/2)^6}{7!}\frac{\partial^7 u}{\partial x^7} \tag{7.20}$$

The fourth order approximation can be obtained by combining these expressions to yield:

$$\frac{9}{8}\frac{u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}}{\Delta x} - \frac{1}{8}\frac{u_{j+\frac{3}{2}} - u_{j-\frac{3}{2}}}{3\Delta x} = \left.\frac{\partial u}{x}\right|_j - 9\frac{(\Delta x/2)^4}{5!}\frac{\partial^5 u}{\partial x^5} - 90\frac{(\Delta x/2)^6}{7!}\frac{\partial^7 u}{\partial x^7} \tag{7.21}$$

$$\frac{25}{24}\frac{u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}}{\Delta x} - \frac{1}{24}\frac{u_{j+\frac{5}{2}} - u_{j-\frac{5}{2}}}{5\Delta x} = \left.\frac{\partial u}{x}\right|_j - 25\frac{(\Delta x/2)^4}{5!}\frac{\partial^5 u}{\partial x^5} - 650\frac{(\Delta x/2)^6}{7!}\frac{\partial^7 u}{\partial x^7} \tag{7.22}$$

Finally the above two expressions can be combined to yield the sixth-order approximation:

$$\frac{450}{384}\frac{u_{j+\frac{1}{2}} - u_{j-\frac{1}{2}}}{\Delta x} - \frac{25}{128}\frac{u_{j+\frac{3}{2}} - u_{j-\frac{3}{2}}}{3\Delta x} + \frac{9}{384}\frac{u_{j+\frac{5}{2}} - u_{j-\frac{5}{2}}}{5\Delta x} = \left.\frac{\partial u}{x}\right|_j + 150\frac{(\Delta x/2)^6}{7!}\frac{\partial^7 u}{\partial x^7} \tag{7.23}$$

Going back to the dispersion relation, we now look for periodic solution of the form:

$$u_{j+\frac{1}{2}} = \hat{u}e^{i(kx_{j+\frac{1}{2}}-\sigma t)} \text{ and } \eta_j = \hat{\eta}e^{i(kx_j-\sigma t)} \tag{7.24}$$

which when replaced in the FDA yields the following

$$\begin{aligned}
\sum_{m=0}^{M}\beta_m(u_{j+\frac{1+2m}{2}} - u_{j-\frac{1+2m}{2}}) &= \hat{u}e^{i(kx_j-\sigma t)}\sum_{m=0}^{M}\beta_m\left[e^{i\frac{1+2m}{2}k\Delta x} - e^{-i\frac{1+2m}{2}k\Delta x}\right] && (7.25)\\
&= \hat{u}e^{i(kx_j-\sigma t)}2i\sum_{m=0}^{M}\beta_m\sin\frac{1+2m}{2}k\Delta x && (7.26)\\
\sum_{m=0}^{M}\beta_m(\eta_{j+\frac{1}{2}+\frac{1+2m}{2}} - \eta_{j+\frac{1}{2}-\frac{1+2m}{2}}) &= \hat{\eta}e^{i\left(kx_{j+\frac{1}{2}}-\sigma t\right)}\sum_{m=0}^{M}\beta_m\left[e^{i\frac{1+2m}{2}k\Delta x} - e^{-i\frac{1+2m}{2}k\Delta x}\right] && (7.27)\\
&= \hat{\eta}e^{i\left(kx_{j+\frac{1}{2}}-\sigma t\right)}2i\sum_{m=0}^{M}\beta_m\sin\frac{1+2m}{2}k\Delta x && (7.28)
\end{aligned}$$

Inserting these expressions in the FDA for the C-grid we obtain the following equations after eliminating the exponential factors to get the dispersion equations:

$$\begin{aligned}
-i\sigma\hat{u} + g2i\left(\sum_{m=0}^{M}\beta_m\frac{\sin\frac{1+2m}{2}k\Delta x}{\Delta x}\right)\hat{\eta} &= 0 && (7.29)\\
H2i\left(\sum_{m=1}^{M}\beta_m\frac{\sin\frac{1+2m}{2}k\Delta x}{\Delta x}\right)\hat{u} - i\sigma\hat{\eta} &= 0 && (7.30)
\end{aligned}$$

The frequency and the phase speed are then given by

$$\sigma_{C,M} = \pm c\sum_{m=0}^{M}\beta_m\frac{\sin\frac{1+2m}{2}k\Delta x}{\Delta x/2}\sigma_{C,M} = \pm c\sum_{m=0}^{M}\beta_m\frac{\sin\frac{1+2m}{2}k\Delta x}{k\Delta x/2} \tag{7.31}$$

We thus have the following phase speeds for schemes of order 2,4 and 6

$$\begin{aligned}
\sigma_{C,2} &= c\frac{\sin\frac{k\Delta x}{2}}{k\Delta x/2} && (7.32)\\
\sigma_{C,4} &= c\left[\frac{27\sin\frac{k\Delta x}{2} - \sin 3\frac{k\Delta x}{2}}{24k\Delta x/2}\right] && (7.33)\\
\sigma_{C,6} &= c\left[\frac{2250\sin\frac{k\Delta x}{2} - 125\sin 3\frac{k\Delta x}{2} + 9\sin 5\frac{k\Delta x}{2}}{1920k\Delta x/2}\right] && (7.34)
\end{aligned}$$

Figure 7.1 compares the shallow water phase speed for the staggered and unstaggered configuration for various order of centered difference approximations. The un-staggered schemes display a familiar pattern: by increasing order the phase speed in the intermediate wavelengths is improved but there is a rapid deterioration for the marginally resolved waves $k\Delta x \geq 0.6$. The staggered scheme on the other hand displays a more accurate representation of the phase speed for the entire spectrum. Notice that the second order

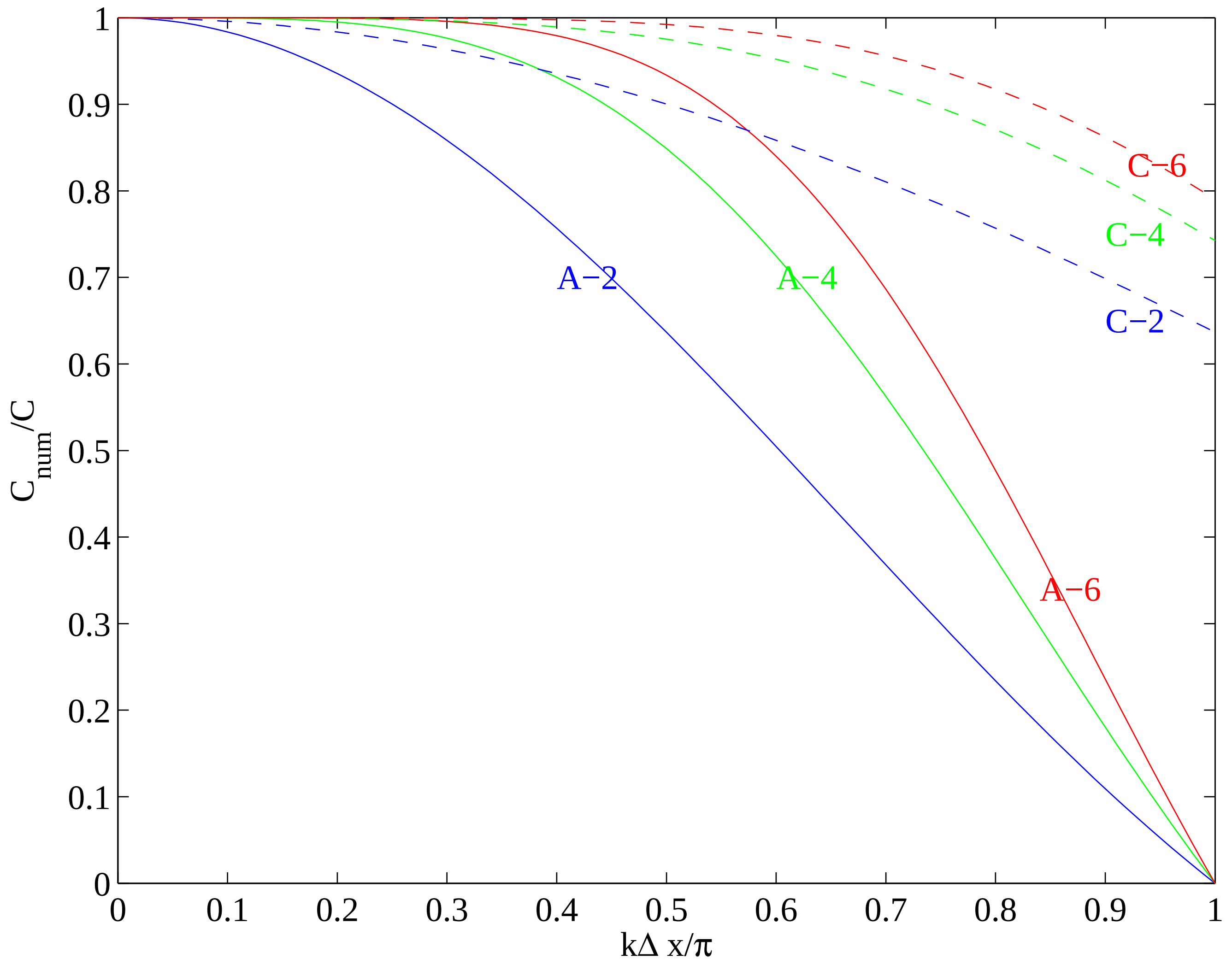


Figure 7.1: Phase speed of the spatially discrete linearized shallow water equation. The solid lines show the phase speed for the A-grid configuration for centered schemes of order 2, 4 and 6, while the dashed lines show the phase speed of the staggered configuration for orders 2, 4 and 6.

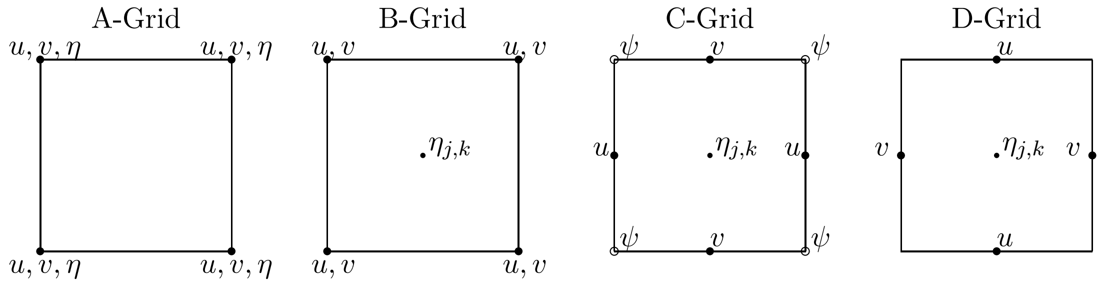


Figure 7.2: Configuration of 4 Arakawa grids

staggered approximation provides a phase fidelity which is comparable to the the fourth order approximation in the intermediate wavelengths $0.2\pi \leq k\Delta x \leq 0.6\pi$ and superior for wavelengths $k\Delta x \geq 0.6$. Finally, and most importantly the unstaggered scheme possess a null mode where $C = 0$ which could manifest itself as a non-propagating $2\Delta x$ spurious mode; the staggered schemes do not have a null mode.

7.2 Two-Dimensional SWE

Here we carry out the dispersion relationship for the two-dimensional shallow water equaions in the presence of rotation. We shall consider the two cases of flow on an f-plane and flow on a β-plane. We will also consider various grid information that include the Arakawa grids A, B, C, and D.

7.2.1 Inertia gravity waves

The linearized equations are given by

$$u_t - fv + g\eta_x = 0 \tag{7.35}$$
$$v_t + fu + g\eta_y = 0 \tag{7.36}$$
$$\eta_t + H(u_x + v_y) = 0 \tag{7.37}$$

Assuming periodic solutions in time and space of the form

$$(u, v, \eta) = (\hat{u}, \hat{v}, \hat{\eta})e^{i(kx+ly-\omega t)},$$

where (k, l) are wavenumbers in the $x - y$ directions, we obtain the following eigenvalue problem for the frequency σ:

$$\begin{vmatrix} -i\sigma & -f & gik \\ f & -i\sigma & gil \\ iHk & iHl & -i\sigma \end{vmatrix} = -i\omega\left[-\omega^2 + f^2 + c^2(k^2 + l^2)\right] = 0 \tag{7.38}$$

Here $c = \sqrt{gH}$ is the gravity wave speed. The non-inertial roots can be written in the following form:

$$\sigma^2 = 1 + a^2\left[(kd)^2 + (ld)^2\right] \tag{7.39}$$

where $\sigma = \omega/f$ is a non-dimensional frequency, d is the grid spacing assumed uniform in both directions and a is the ratio of the Rossby number to the grid spacing, i.e. it is the number of points per rossby radius:

$$a = \frac{R_o}{d} = \frac{gH}{fd}. \tag{7.40}$$

Although the continuous dispersion does not depend on a grid spacing, it is useful to write it in the above form for comparison with the numerical dispersion relations. The numerical dispersion for the various grids are given by Dukowicz (1995)

$$\sigma_A^2 = 1 + a^2\left[\sin^2 kd + \sin^2 ld\right] \tag{7.41}$$

$$\sigma_B^2 = 1 + 2a^2\left[1 - \cos kd \cos ld\right] \tag{7.42}$$

$$\sigma_C^2 = \cos^2\frac{kd}{2}\cos^2\frac{ld}{2} + 4a^2\left[\sin^2\frac{kd}{2} + \sin^2\frac{ld}{2}\right] \tag{7.43}$$

$$\sigma_D^2 = \cos^2\frac{kd}{2}\cos^2\frac{ld}{2} + a^2\left[\cos^2\frac{kd}{2}\sin^2 ld + \sin^2 kd\cos^2\frac{ld}{2}\right] \tag{7.44}$$

7.3 Rossby waves

The Rossby dispersion relation are given by

$$\sigma = -a^2 kd\left\{1 + a^2\left[(kd)^2 + (ld)^2\right]\right\}^{-1} \tag{7.45}$$

$$\sigma_A = -a^2 \sin kd\, \cos ld\left\{1 + a^2\left[\sin^2 kd + \sin^2 ld\right]\right\}^{-1}; \tag{7.46}$$

$$\sigma_B = -a^2 \sin kd\, \left\{1 + 2a^2\left[1 - \cos kd \cos ld\right]\right\}^{-1} \tag{7.47}$$

$$\sigma_C = -a^2 \sin kd\, \cos^2\frac{ld}{2}\left\{\cos^2\frac{kd}{2}\cos^2\frac{ld}{2} + 4a^2\left[\sin^2\frac{kd}{2} + \sin^2\frac{ld}{2}\right]\right\}^{-1} \tag{7.48}$$

$$\sigma_D = -a^2 \sin kd\, \cos^2\frac{ld}{2}\left\{1 + 4a^2\left[\sin^2\frac{kd}{2} + \sin^2\frac{ld}{2}\right]\right\}^{-1}; \tag{7.49}$$

where the frequency is now normalized by βd. The normalized Rossby wave frequencies, and their relative error for the various grids are displayed in figures 7.8-7.12 for various Rossby radius parameters a. Since contour plots are hard to read we also supply line-plots for special l values $l = 0$ and $l = k$ in figure 7.13. From these plots one can conclude the following:

1. All grid configurations have a null mode at $k\Delta x = \pi$
2. The C and D grids have a null mode for all zonal wavenumber when $ld = \pi$.
3. for $a \geq 2$ the B,C and D grids perform similarly for the resolved portion of the spectrum $kd \leq 2\pi/5$.

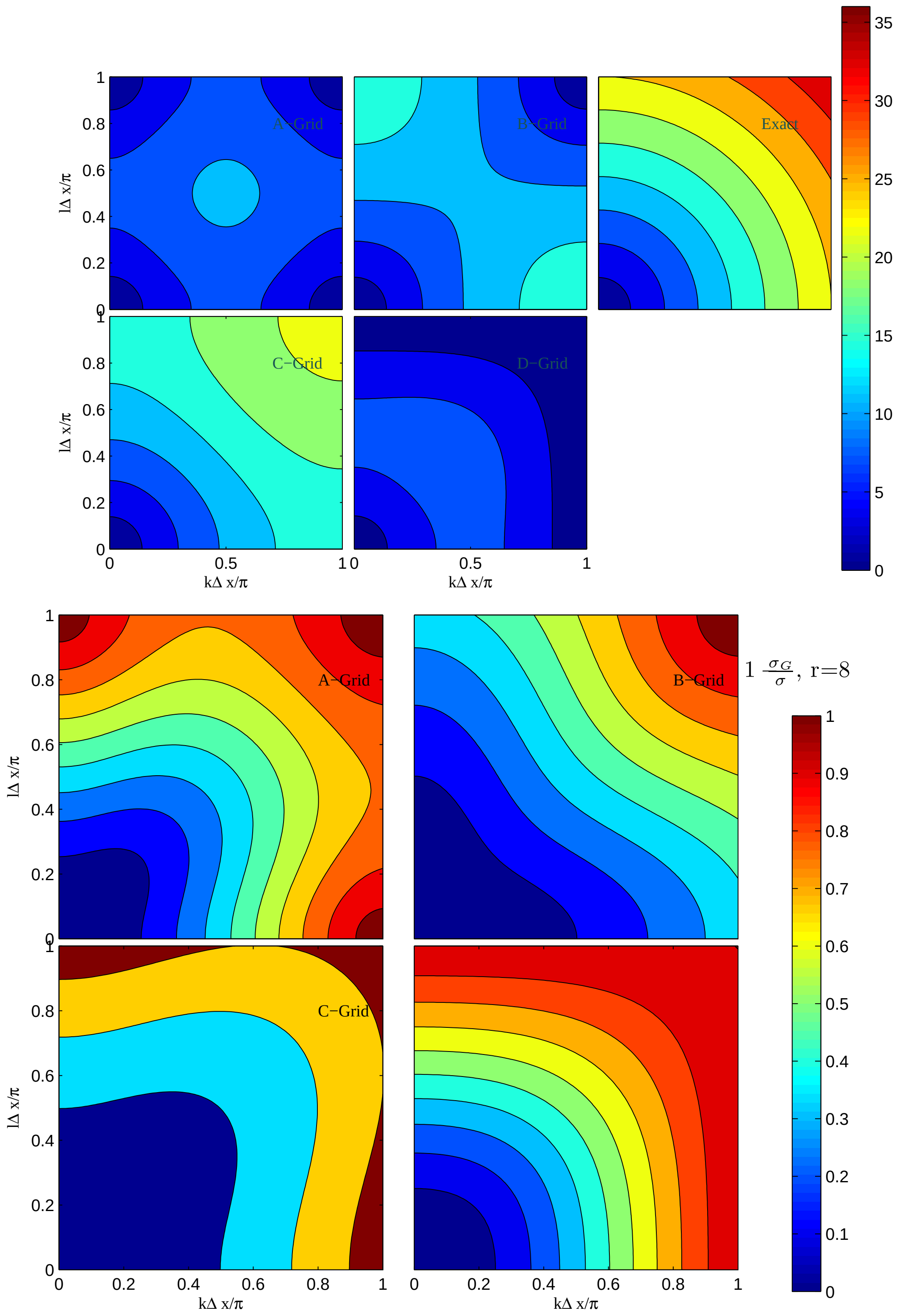


Figure 7.3: Comparison of the dispersion relation on the Arakawa A, B, C and D grids. The top figure shows the dispersion relation while the bottom one shows the relative error. The parameter a=8.

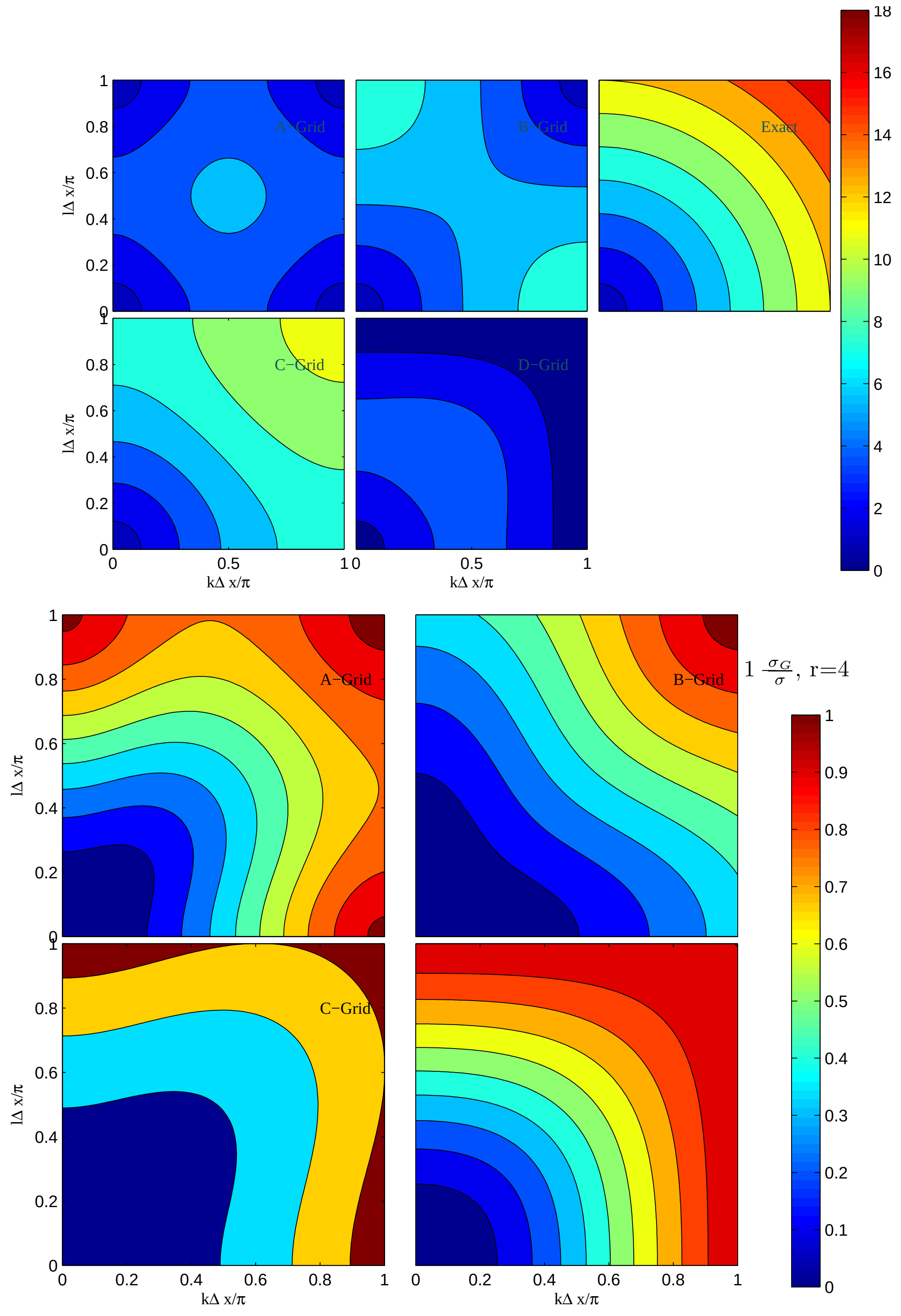
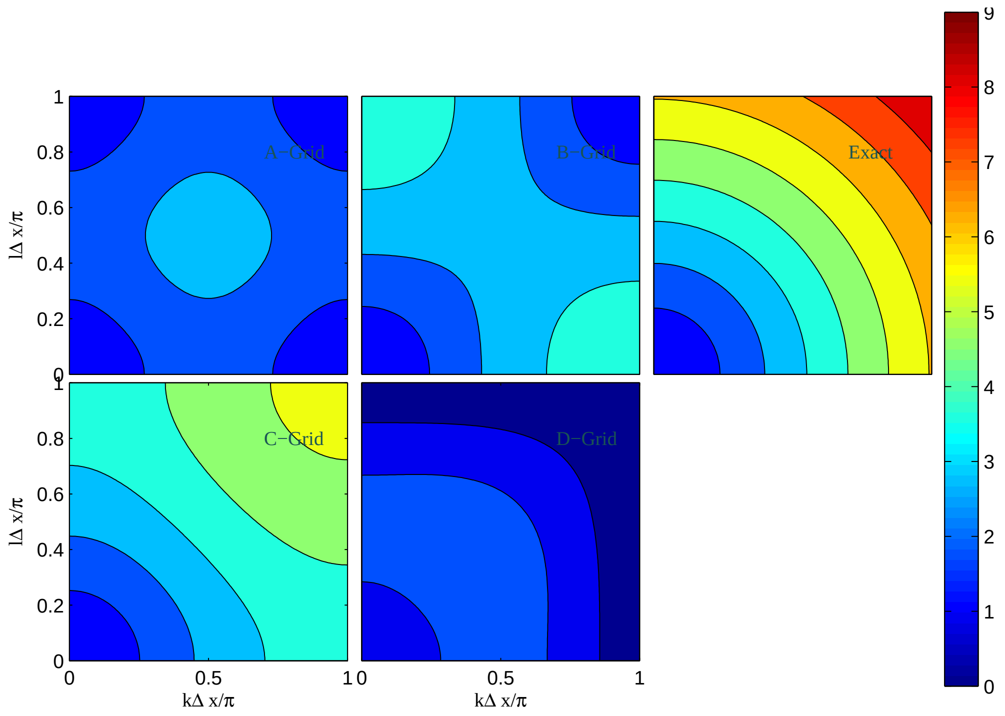


Figure 7.4: Same as 7.3 but for a=4.



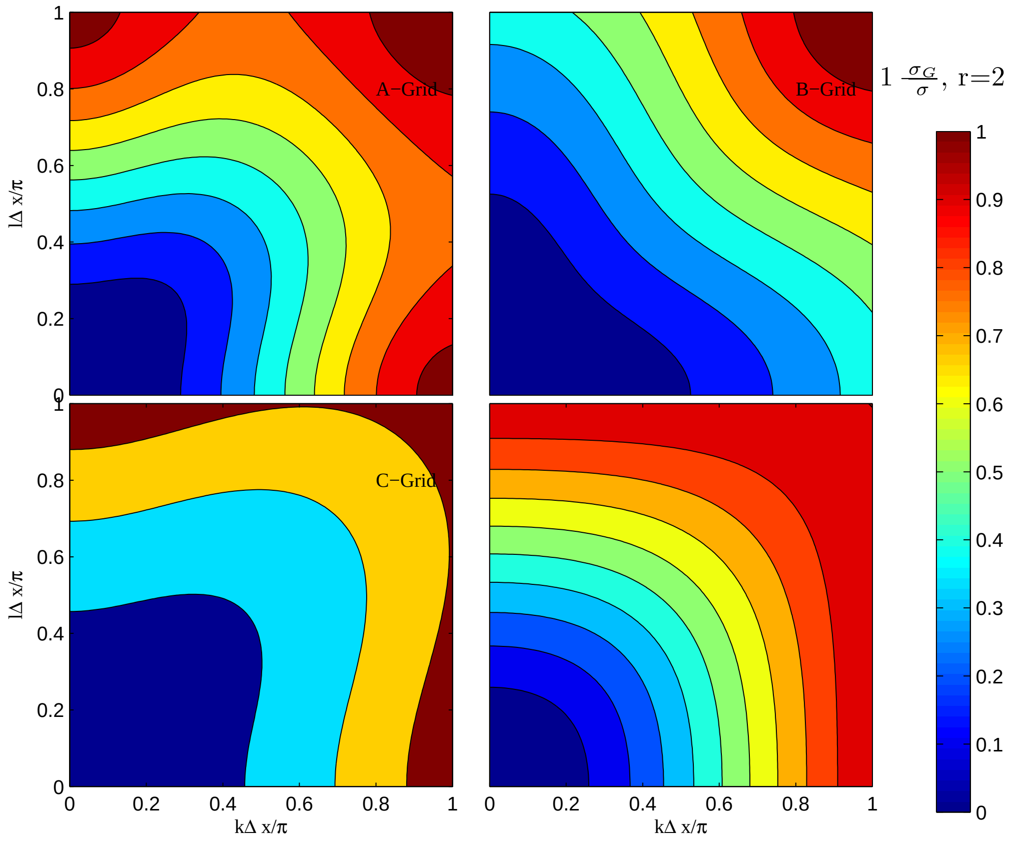


Figure 7.5: Same as 7.3 but for a=2.

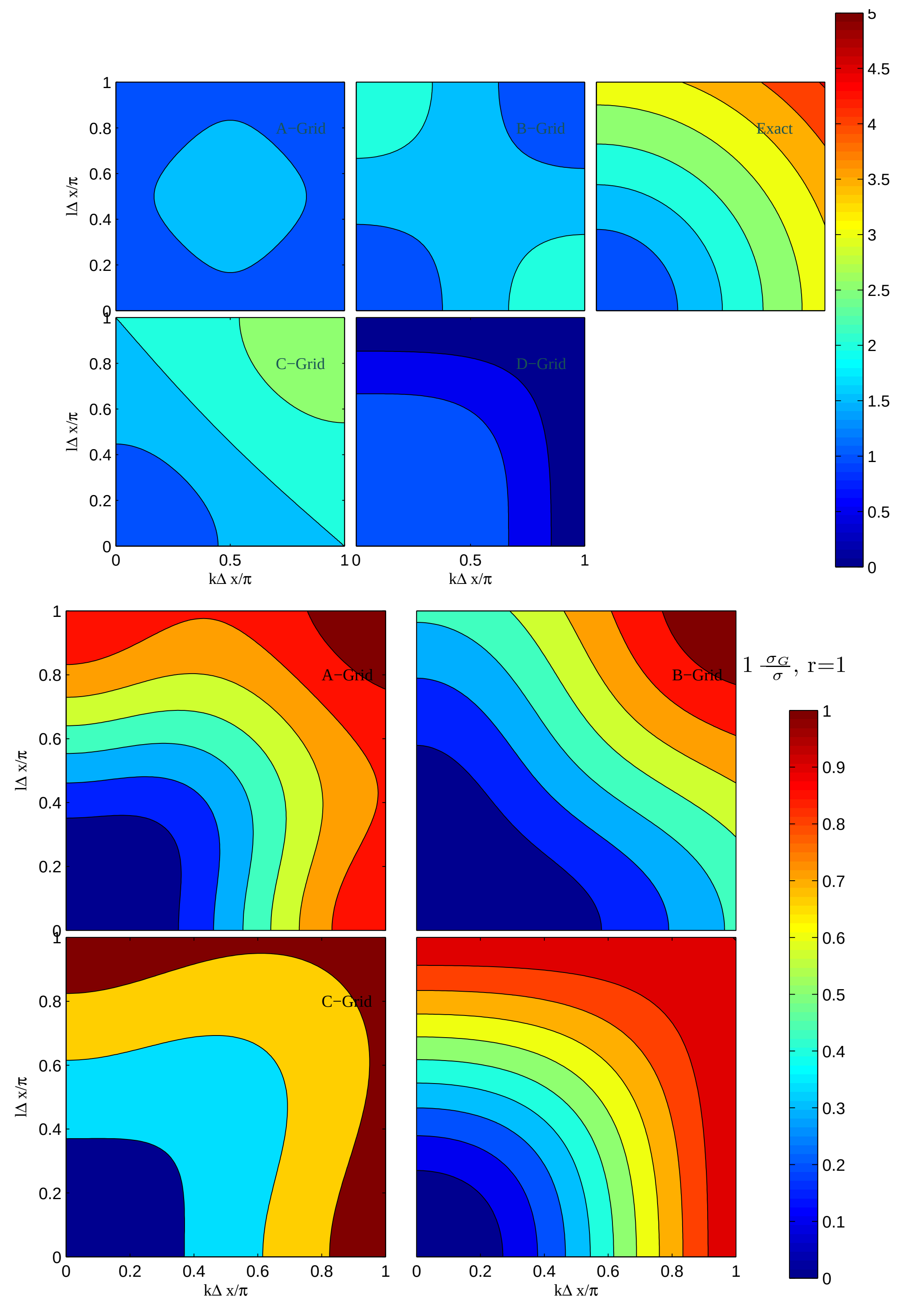


Figure 7.6: Same as 7.3 but for a=1.

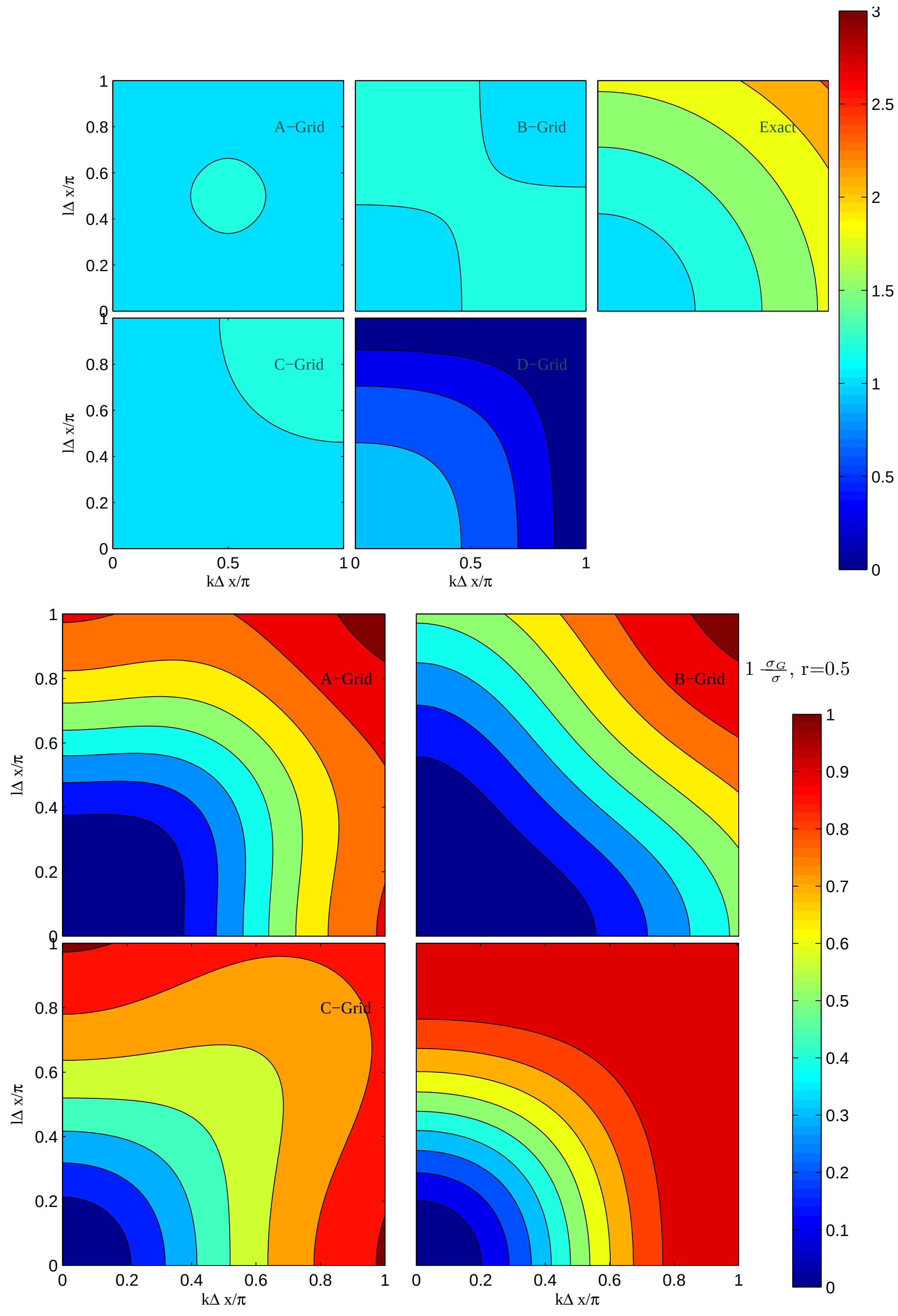


Figure 7.7: Same as 7.3 but for a=1/2.

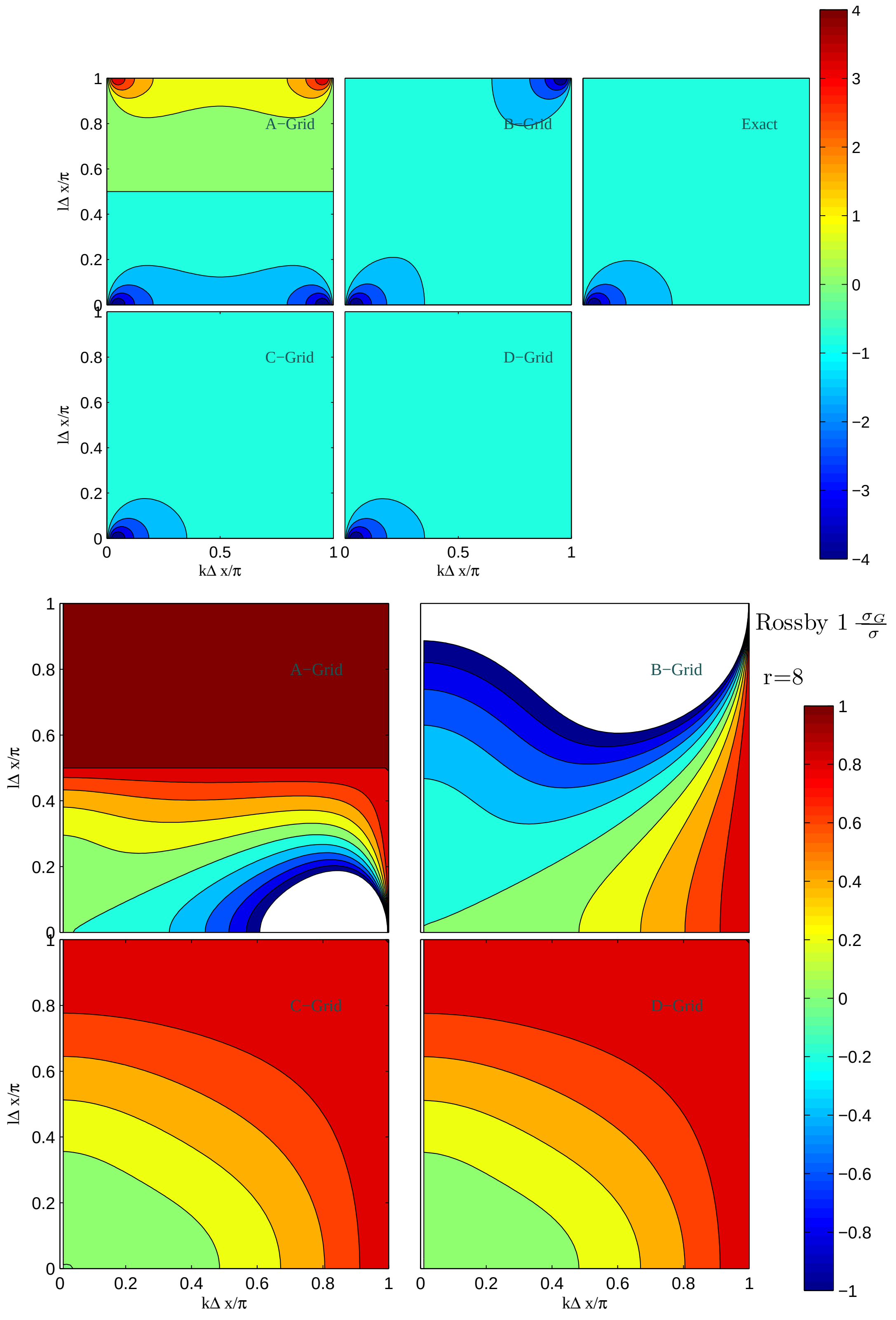


Figure 7.8: Comparison of Rossby wave dispersion for the different Arakawa grids. The top figures shows the dispersion while the bottom ones show the relative error. Here a=8.

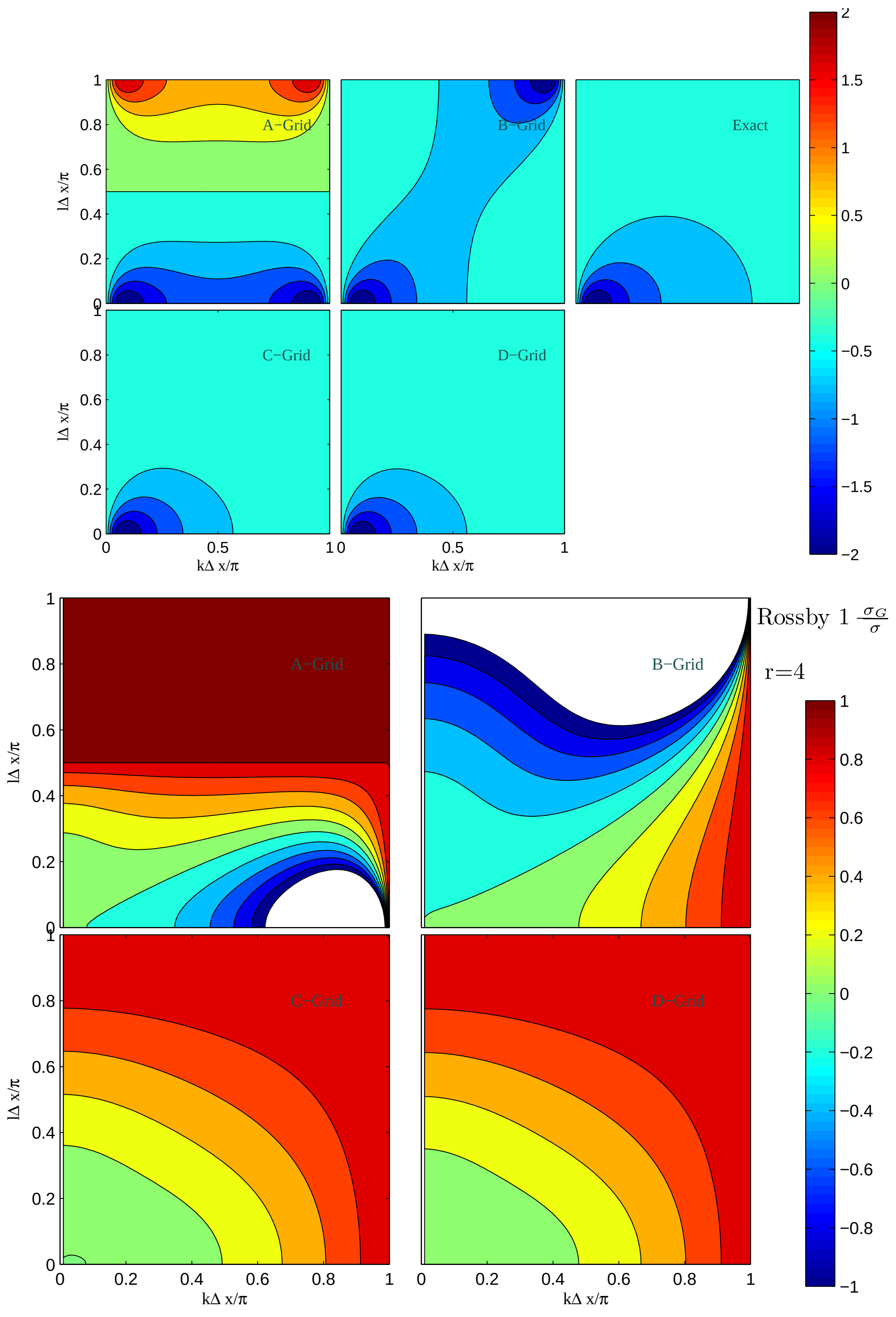
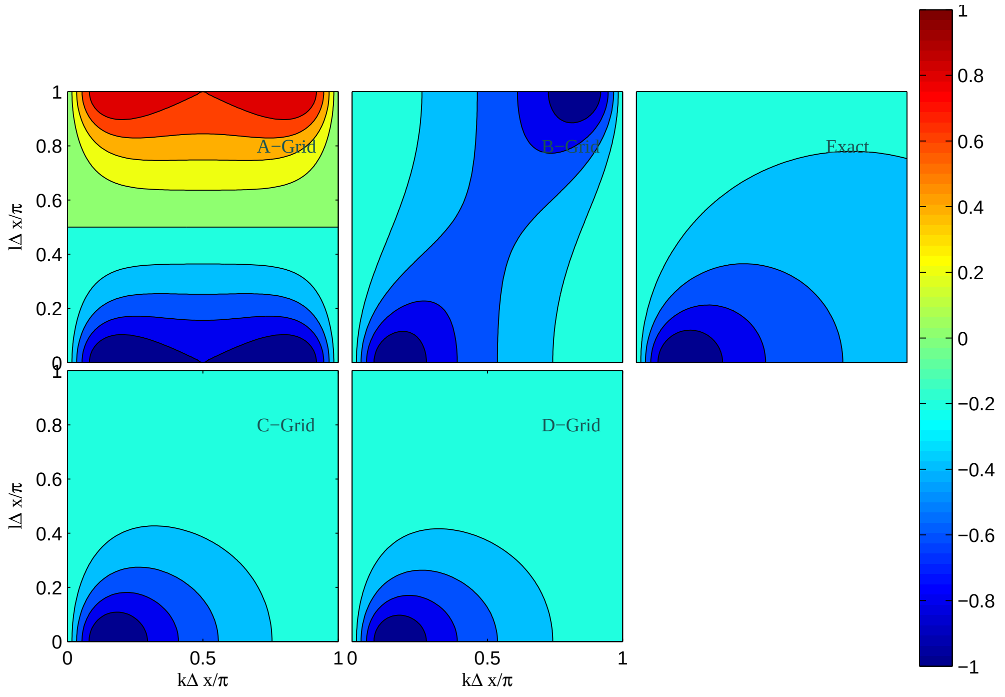


Figure 7.9: Same as figure 7.8 but for a=4.



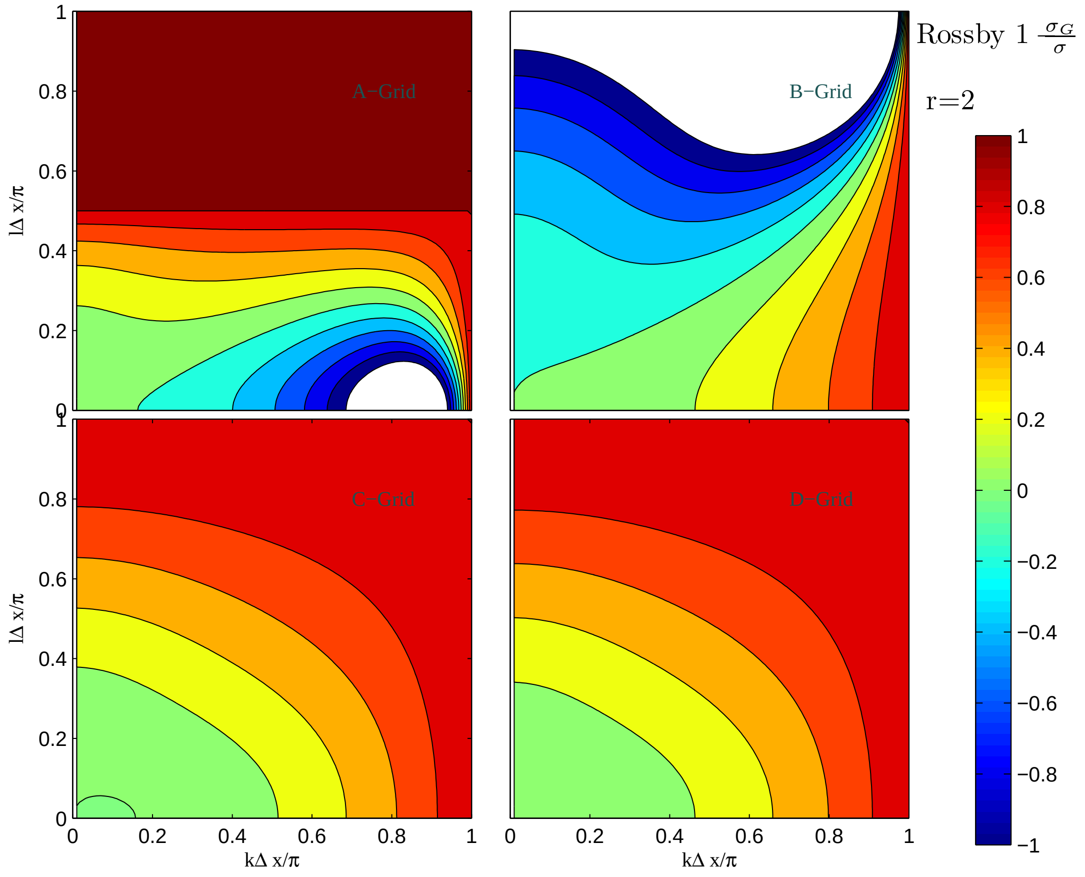
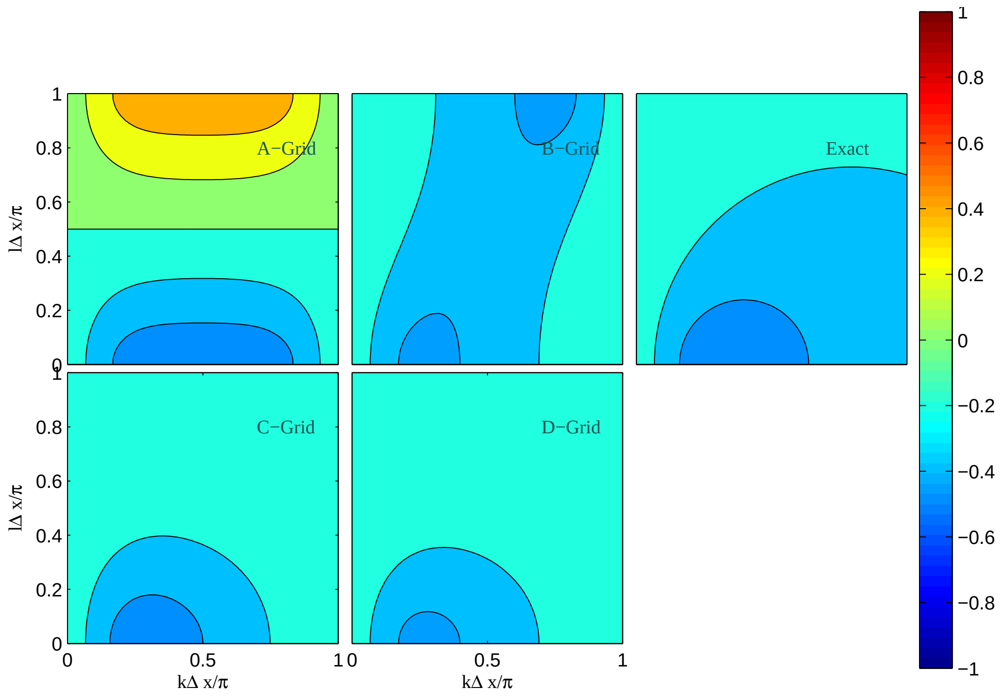


Figure 7.10: Same as figure 7.8 but for a=2.



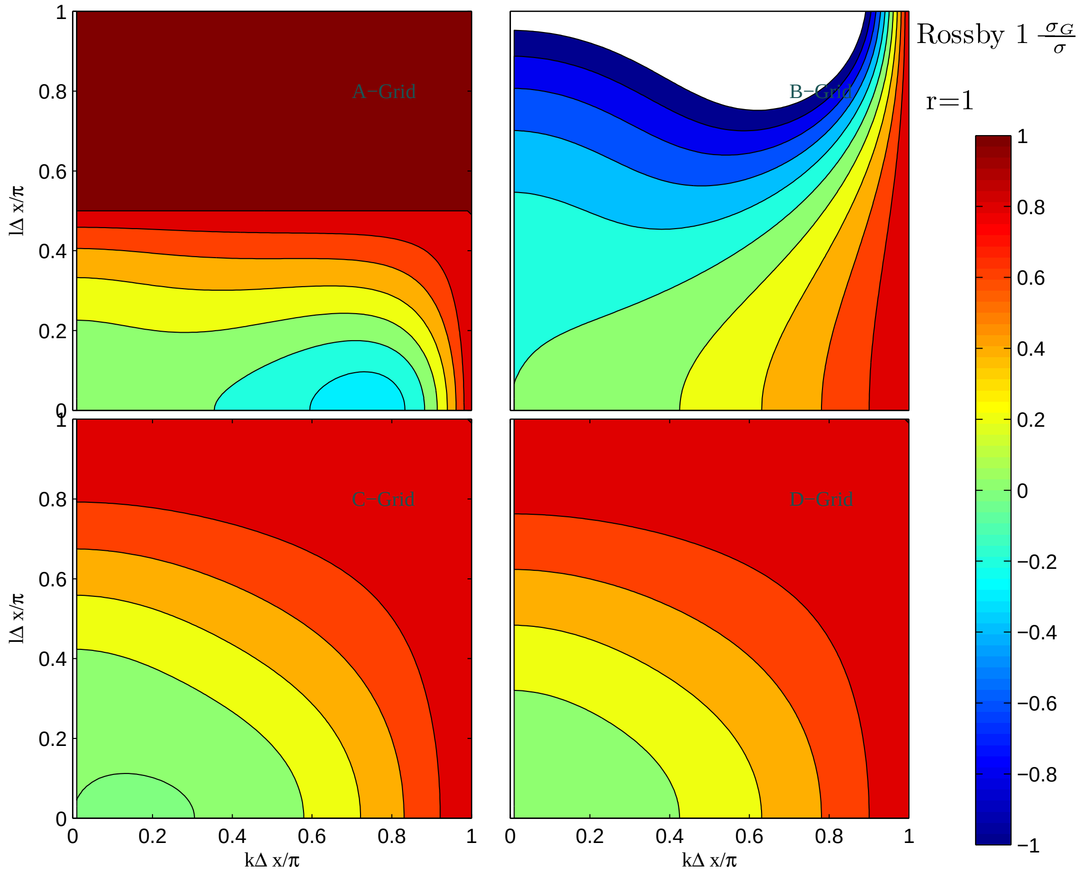


Figure 7.11: Same as figure 7.8 but for a=1.

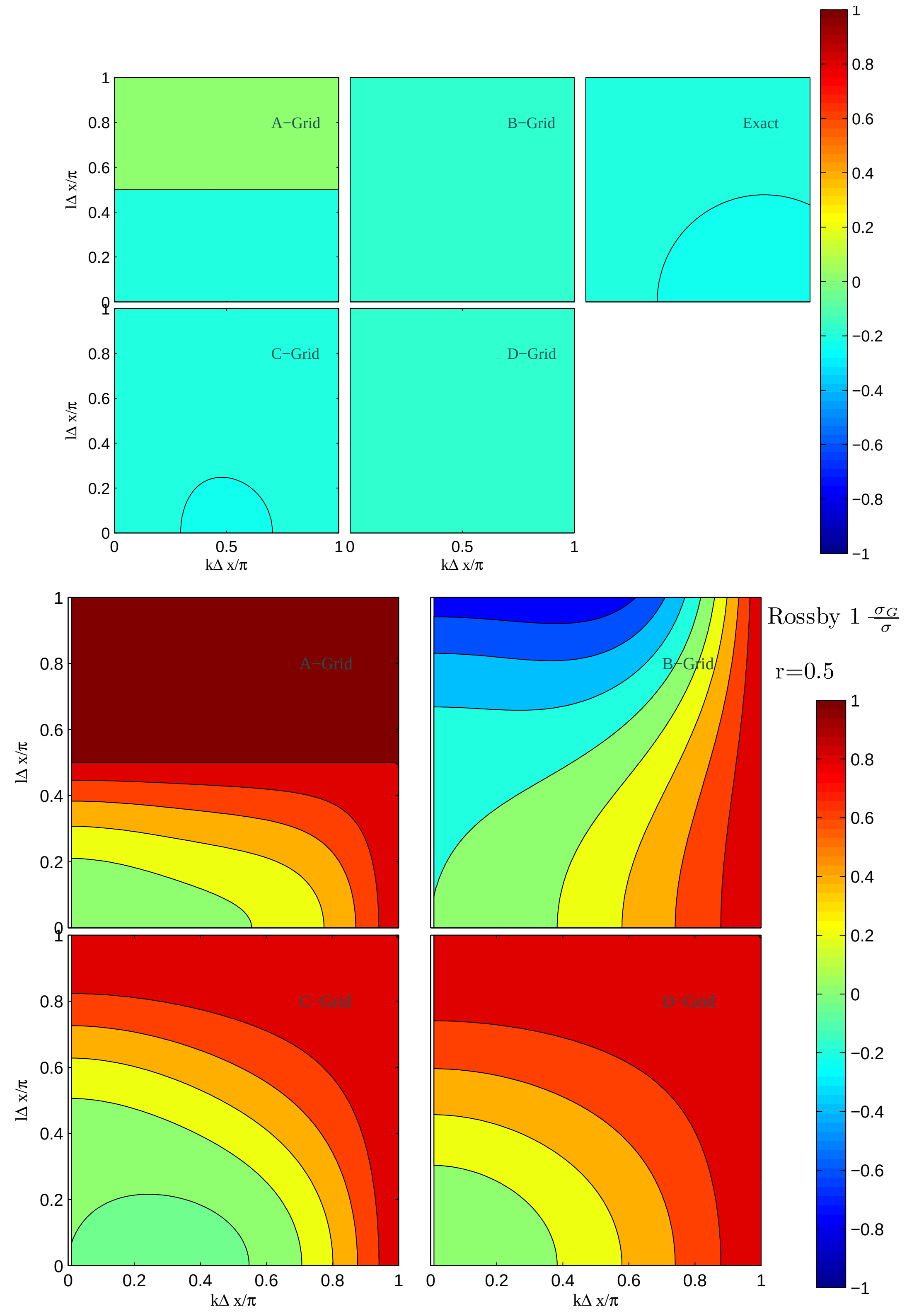


Figure 7.12: Same as figure 7.8 but for a=1/2.

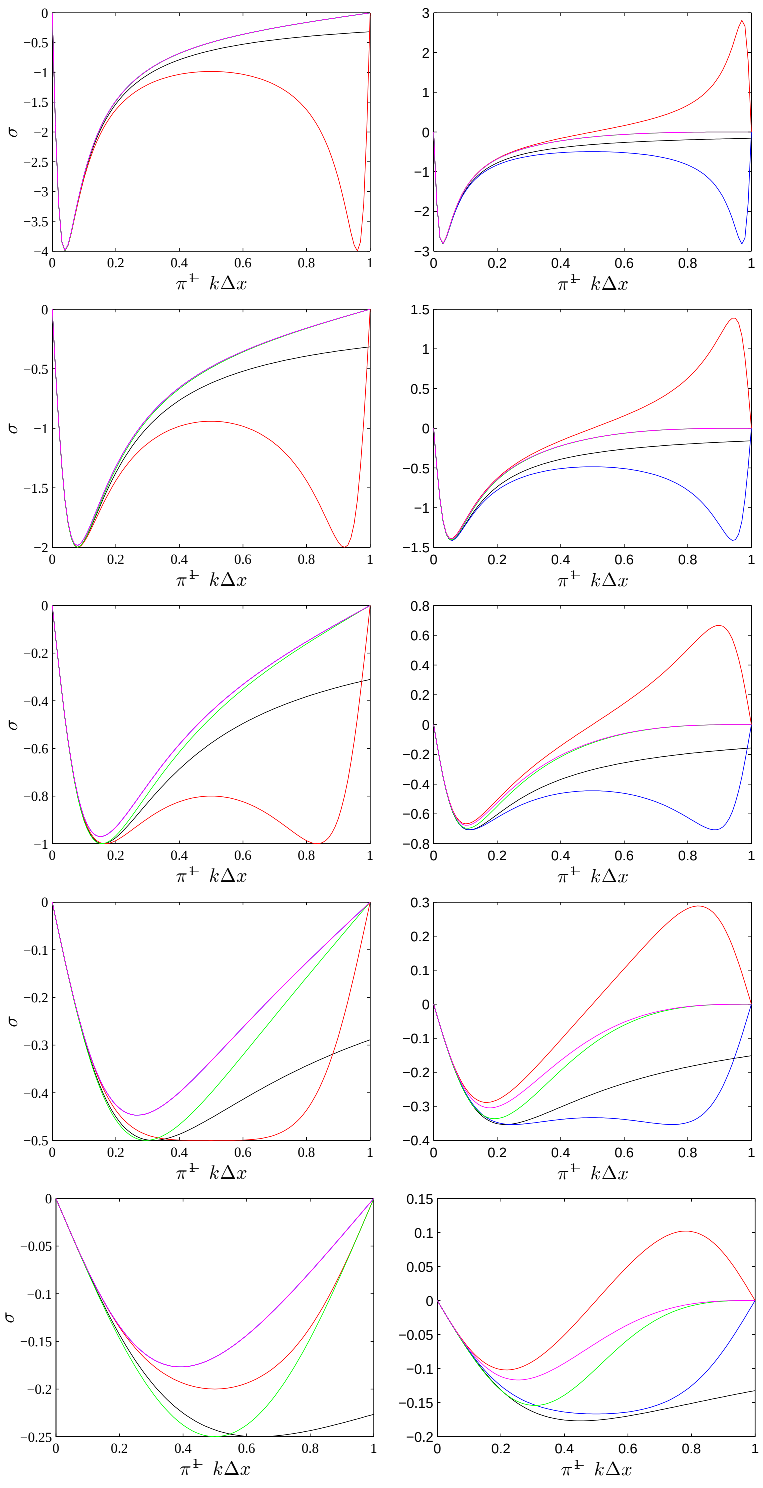


Figure 7.13: Rossby wave frequency σ versus $k\Delta x$ for, from top to bottom $a =$, 8,4,2, 1 and 1/2. The left figures show the case $l = 0$ and the right figures the case $l = k$. The black line refers to the continuous case and the colored line to the A (red), B (blue), C (green), and D (magenta) grids.

Chapter 8

Solving the Poisson Equations

The textbook example of an elliptic equation is the Poisson equation:

$$\nabla^2 u = f, \mathbf{x} \in \Omega \tag{8.1}$$

subject to appropriate boundary conditions on $\partial\Omega$, the boundary of the domain. The right hand side f is a known function. We can approximate the above equation using standard second order finite differences:

$$\frac{u_{j+1,k} - 2u_{j,k} + uj - 1, k}{\Delta^2 x} + \frac{u_{j,k+1} - 2u_{j,k} + uj, k - 1}{\Delta^2 y} = f_{j,k} \tag{8.2}$$

The finite difference representation 8.2 of the Poisson equation results in a coupled system of algebraic equations that must be solved simultaneously. In matrix notation the system can be written in the form $Ax = b$, where x represents the vector of unknowns, b represents the right hand side, and A the matrix representing the system. Boundary conditions **must** be applied prior to solving the system of equations.

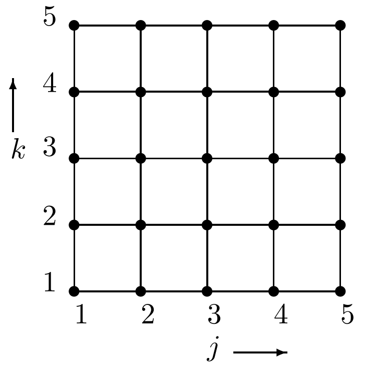


Figure 8.1: Finite Difference Grid for a Poisson equation.

Example 12 For a square domain divided into 4x4 cells, as shown in figure 8.1, subject to Dirichlet boundary conditions on all boundaries, there are 9 unknowns $u_{j,k}$, with $(j,k) = 1,2,3$. The finite difference equations applied at these points provide us with the system:

$$\begin{pmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{pmatrix} \begin{pmatrix} u_{2,2} \\ u_{3,2} \\ u_{4,2} \\ u_{2,3} \\ u_{3,3} \\ u_{4,3} \\ u_{2,4} \\ u_{3,4} \\ u_{4,4} \end{pmatrix} = \Delta \begin{pmatrix} f_{2,2} \\ f_{3,2} \\ f_{4,2} \\ f_{2,3} \\ f_{3,3} \\ f_{4,3} \\ f_{2,4} \\ f_{3,4} \\ f_{4,4} \end{pmatrix} - \begin{pmatrix} u_{2,1} + u_{1,2} \\ u_{3,1} \\ u_{4,1} \\ 0 \\ 0 \\ 0 \\ u_{2,5} \\ u_{3,5} \\ u_{4,5} + u_{5,4} \end{pmatrix} \tag{8.3}$$

where $\Delta x = \Delta y = \Delta$. Notice that the system is symmetric, and pentadiagonal (5 non-zero diagonal). This last property precludes the efficient solution of the system using the efficient tridiagonal solver.

The crux of the work in solving elliptic PDE is the need to update the unknowns simultaneously by inverting the system $Ax = b$. The solution methodologies fall under 2 broad categories:

1. **Direct solvers:** calculate the solution $x = A^{-1}b$ exactly (up to round-off errors). These methods can be further classified as:

   (a) **Matrix Methods:** are the most general type solvers and work for arbitrary non-singular matrices A. They work by factoring the matrix into a lower and upper triangular matrices that can be easily inverted. The most general and robust algorithm for this factorization is the Gaussian elimination method with partial pivoting. For symmetric real system a slightly faster version relies on the Cholesky algorithm. The main drawback of matrix methods is that their storage cost CPU cost grow rapidly with the increase of the number of points. In particular, the CPU cost grows as $O(M^3)$, where N is the number of unknowns. If the grid has N points in each direction, then for a 2D problem this cost scales like N^6, and like N^9 for $3D$ problems.

   (b) **FFT** also referred to as fast solvers. These methods take advantage of the structure of separable equations to diagonalize the system using Fast Fourier Transforms. The efficiency of the method rests on the fact that FFT costs grow like $N^2 \ln N$ in 2D, a substantial reduction compared to the N^6 cost of matrix methods.

2. **Iterative Methods** calculate an approximation to the solution that minimizes the norm of the residual vector $r = b - Ax$. There is a large number of iterative solvers; the most efficient ones are those that have a fast convergence rate and low CPU cost per iteration. Often times it is necessary to exploit the structure of the

equations to reduce the CPU cost and accelerate convergence. Here we mention a few of the more common iterative schemes:

(a) Fixed point methods: Jacobi and Gauss-Seidel Methods)

(b) Multigrid methods

(c) Krylov Method: Preconditioned Conjugate Gradient (PCG)

8.1 Iterative Methods

We will discuss mainly the fixed point iterations methods and (maybe) PCG methods.

8.1.1 Jacobi method

The solution of the Poisson equation can be viewed as the steady state solution to the following parabolic equation:

$$u_t = \nabla^2 u - f \tag{8.4}$$

At steady state, $t \to \infty$ the left hand side of the equation goes to zero and we recover the original Poisson equation. The artifice of introducing a pseudo-time t is usefull because we can now use an explicit method to update the unknowns $u_{j,k}$ individually without solving a system of equation. Using a FTCS approximation we have:

$$u^{n+1}_{j,k} = u^n_{j,k} + \frac{\Delta t}{\Delta^2}\left(u^n_{j+1,k} + u^n_{j-1,k} + u^n_{j,k+1} + u^n_{j,k-1} - 4u^n_{j,k}\right) - \Delta t f_{j,k} \tag{8.5}$$

where we have assumed $\Delta x = \Delta y = \Delta$. At steady state, after an infinite number of iteration, the solution satifies

$$u_{j,k} = u_{j,k} + \frac{\Delta t}{\Delta^2}\left(u_{j+1,k} + u_{j-1,k} + u_{j,k+1} + u_{j,k-1} - 4u^n_{j,k}\right) - \Delta t f_{j,k} \tag{8.6}$$

Forming the difference of equations 8.6 and 8.5 we get:

$$e^{n+1}_{j,k} = e^n_{j,k} + \frac{\Delta t}{\Delta^2}\left(e^n_{j+1,k} + e^n_{j-1,k} + e^n_{j,k+1} + e^n_{j,k-1} - e^n_{j,k}\right), \tag{8.7}$$

where $e^n_{j,k} = u^n_{j,k} - u_{j,k}$. Thus, the error will evolve according to the amplification factor associated with the FDE 8.7. We note that the initial conditions for our pseudo-time are not important for the convergence analysis; of course we would like to start our initial guess as close as possible to the right solution. Second, since we are not interested in the transient solution, the time-accuracy is not relevant easier. As a matter of fact, we would like to use the largest time step possible that will lead us to steady state. A Von-Neumann stability analysis shows that the stability limit is $\Delta t \leq \frac{1}{4}\Delta^2$, substituting this time step in the equation we obtain the **Jacobi** algorithm:

$$u^{n+1}_{j,k} = \frac{u^n_{j+1,k} + u^n_{j-1,k} + u^n_{j,k+1} + u^n_{j,k-1}}{4} - \frac{\Delta^2}{4} f_{j,k} \tag{8.8}$$

A Von-Neumann analysis shows that the amplification factor for this method is given

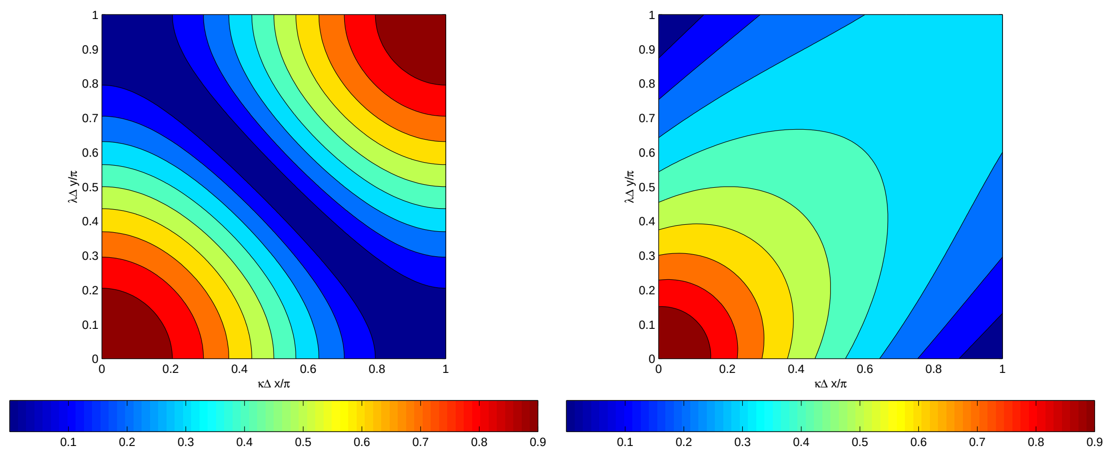


Figure 8.2: Magnitude of the amplification factor for the Jacobi method (left) and Gauss-Seidel method (right) as a function of the (x, y) Fourier components.

by

$$G = \frac{\cos \kappa\Delta x + \cos \kappa\Delta y}{2} \tag{8.9}$$

where (κ, λ) are the wavenumbers in the (x, y) directions. A plot of $|G|$ is shown in figure 8.2. It is clear that the shortest $(\kappa\Delta x \to \pi)$ and longest $(\kappa\Delta x \to 0)$ error components are damped the least. The intermediate wavelengths $(\kappa\Delta x = \pi/2)$ are damped most effectively.

8.1.2 Gauss-Seidel method

A simple modification to the Gauss-Seidel method can improve the storage and convergence rate of the Jacobi method. Note that in Jacobi, only values at the previous iterations are used to update the solution. An improved algorithm can be obtained if the most recent value is used. Assuming that we are sweeping through the grid by increased j and k indeces, then the Gauss-Seidel method can be written in the form:

$$u_{j,k}^{n+1} = \frac{u_{j+1,k}^{n} + u_{j-1,k}^{n+1} + u_{j,k+1}^{n} + u_{j,k-1}^{n+1}}{4} - \frac{\Delta^2}{4} f_{j,k} \tag{8.10}$$

The major advantages of this scheme are that only one time level need be stored (the values can be updated on the fly), and the convergence rate can be improved substantially (double the rate of the Jacobi method). The latter point can be quantified by looking at the error amplification which now takes the form:

$$G = \frac{e^{i\alpha} + e^{i\beta}}{4 - (e^{-i\alpha} + e^{-i\beta})} \quad |G|^2 = \frac{1 + \cos(\alpha - \beta)}{9 - 4(\cos\alpha + \cos\beta) + \cos(\alpha - \beta)} \tag{8.11}$$

where $\alpha = \kappa\Delta x$, and $\beta = \lambda\Delta y$. A plot of $|G|$ versus wavenumbers is shown in figure 8.2, and clearly shows the reduction in the area where $|G|$ is close to 1. Notice that unlike the Jacobi method, the smallest wavelengths are damped at the rate of $1/3$ at every time step. The error components that are damped the least are the long ones: $\alpha, \beta \to 0$.

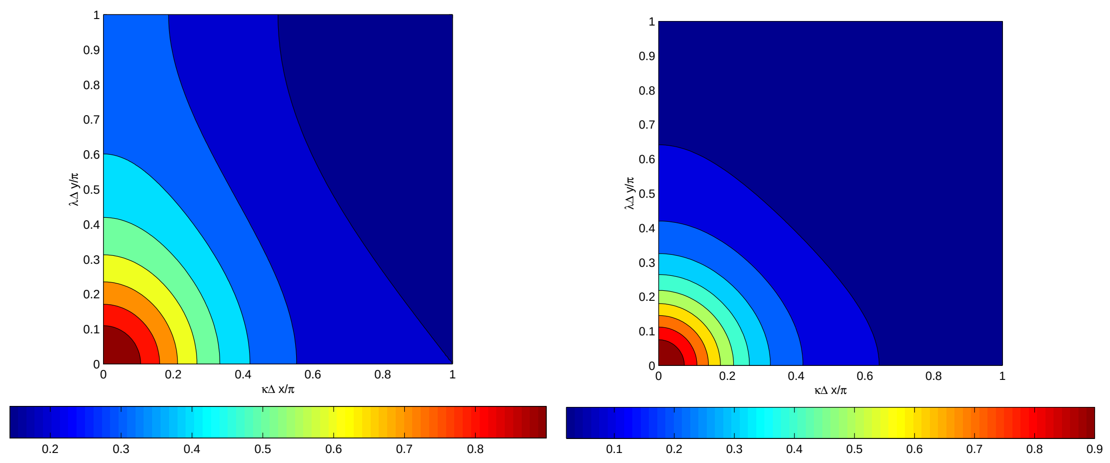


Figure 8.3: Magnitude of the amplification factor for the Gauss-Seidel by rows (left) and Gauss-Seidel method by rows and columns (right) as a function of the (x, y) Fourier components.

8.1.3 Successive Over Relaxation (SOR) method

A correction factor can be added to the update of the Gauss-Seidel method in order to improve the convergence rate. Let $u^*_{j,k}$ denote the temporary value obtained from the Gauss-Seidel step; then an improved estimate of the solution is

$$u^{n+1}_{j,k} = u^n_{j,k} + \omega(u^*_{j,k} - u^n_{j,k}) \tag{8.12}$$

where ω is the correction factor. For $\omega = 1$ we revert to the Gauss-Seidel update, for $\omega < 1$ the correction is under-relaxed, and for $\omega > 1$ the correction is over-relaxed. For convergence, it can be shown that $1 \leq \omega \leq 2$. The optimal ω, ω_o, can be quite hard to compute and depends on the number of points in each direction and the boundary conditions applied. Analytic values for ω_o can be obtained for a Dirichlet problem:

$$\omega_o = 2\frac{1-\sqrt{1-\beta}}{\beta}, \quad \beta = \left[\frac{\cos\frac{\pi}{M-1} + \frac{\Delta x^2}{\Delta y^2}\cos\frac{\pi}{N-1}}{1+\frac{\Delta x^2}{\Delta y^2}}\right]^2 \tag{8.13}$$

where M and N are the number of points in the x and y directions, respectively.

8.1.4 Iteration by Lines

A closer examination of the Gauss-Seidel method in equation 8.10 reveals that an efficient algorithm, relying on tridiagonal solvers, can be produced if the iteration is changed to:

$$u^{n+1}_{j,k} = \frac{u^{n+1}_{j+1,k} + u^{n+1}_{j-1,k} + r^2(u^n_{j,k+1} + u^{n+1}_{j,k-1})}{2(1+r^2)} - \frac{\Delta x^2}{4} f_{j,k} \tag{8.14}$$

where $r = \Delta x/\Delta y$ is the aspect ratio of the grid. Notice that 8.14 has 3 unknowns only at row j since $u^{n+1}_{j,k-1}$ would be known from either a boundary condition or a previous iteration, and $u^n_{j,k+1}$ is still lagged in time. Hence a simple tridiagonal solver can be used to update the rows one-by-one. The amplification factor for this variation on the Gauss-Seidel method is given by:

$$|G|^2 = \frac{r^4}{[2(1+r^2-\cos\alpha)]^2 + [2(1+r^2-\cos\alpha)]2\cos\beta + r^4} \tag{8.15}$$

A plot of $|G|$ for $r = 1$ is shown in figure 8.3. The areas with small $|G|$ have expanded with resepect to those shown in figure 8.2. In order to symmetrize the iterations along the two directions, it is natural to follow a sweep-by-row by a sweep-by-columns. The amplification factor for this iteration is shown in the left panel of figure 8.3 and show a substantial reduction in error amplitude for all wavelenths except the longest ones.

Example 13 In order to illustrate the efficiency of the different methods outline above we solve the following Laplace equation

$$\begin{aligned}
\nabla^2 u &= 0, 0 \le x, y \le 1 && (8.16)\\
u(0,y) &= u(1,y) = 0 && (8.17)\\
u(x,1) &= \sin \pi x && (8.18)\\
u(x,1) &= e^{-16(x-\frac{1}{4})^2} \quad \sin \pi x && (8.19)
\end{aligned}$$

We divide the unit square into $M \times N$ grid points and we use the following methods:Jacobi, Gauss-Seidel, SOR, SOR by line in the x-direction, and SOR by line in both directions. We monitor the convergence history with the rms change in u from one iteration to the next:

$$\|\epsilon\|_2 = \frac{1}{MN}\left[\sum_{j,k}(u^{n+1}_{jk} - u^n_{jk})^2\right]^{\frac{1}{2}} \tag{8.20}$$

The stopping criterion is $\|\epsilon\|_2 < 10^{-13}$, and we limit the maximum number of iterarions to 7,000. We start all iterations with $u = 0$ (save for the bc) as an initial guess. The convergence history is shown in figure 8.4 for $M = N = 65$. The Jacobi and Gauss-Seidel have similar convergence history except near the end where Gauss-Seidel is converging faster. The SOR iterations are the fastest reducing the number of iterations required by a factor of 100 almost. We have used the optimal relaxation factor since it is was computable in our case. The SOR iterations are also quite similar showing a slow decrease of the error in the initial stages but very rapid decrease in the final stages. The criteria for the selection of an iteration algorithm should not rely solely on the algorithm's rate of convergence; it should also the operation count needed to complete each iteration. The convergence history for the above example shows that the 2-way line SOR is the most efficient per iterations. However, table 8.1 shows the total CPU time is cheapest for the point-SOR. Thus, the overhead of the tridiagonal solver is not compensated by the higher efficiency of the SOR by line iterations. Table 8.1 also shows that, where applicable, the FFT-based fast solvers are the most efficient and cheapest.

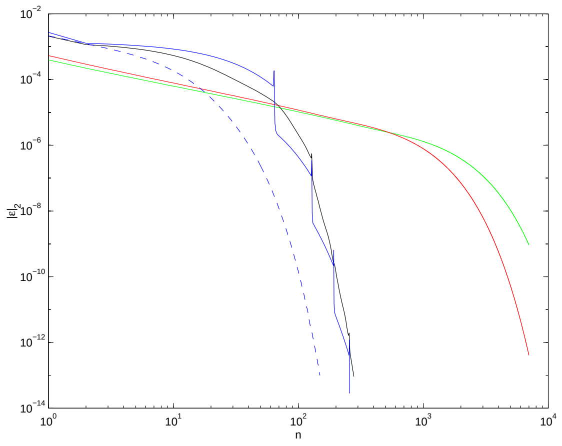


Figure 8.4: Convergence history for the Laplace equation. The system of equation is solved with: Jacobi (green), Gauss-Seidel (red), SOR (black), Line SOR in x (solid blue), and line SOR in x and y (dashed blue). Here $M = N = 65$.

	33	65	129
Jacobi	0.161	0.682	2.769
Gauss-Seidel	0.131	2.197	10.789
SOR	0.009	0.056	0.793
SOR-Line	0.013	0.164	1.291
SOR-Line 2	0.014	0.251	1.403
FFT	0.000	0.001	0.004

Table 8.1: CPU time in second to solve Laplace equation versus the number of points (top row).

8.1.5 Matrix Analysis

The relaxation schemes presented above are not restricted to the Poisson equation but can be re-intrepeted as specific instances of a larger class of schemes. We present the matrix approach in order to unify the different schemes presented. Let the matrix A be split into

$$A = N - P \tag{8.21}$$

where N and P are matrices of the same order as A. The system of equations becomes:

$$Nx = Px + b \tag{8.22}$$

Starting with an arbitrary vector $x^{(0)}$, we define a sequence of vectors $x^{(v)}$ by the recursion

$$Nx^{(n)} = Px^{(n-1)} + b, n = 1, 2, 3, ... \tag{8.23}$$

It is now clear what kind of restrictions need to be imposed on the matrices in order to solve for x, namely: the matrix N must be non-singular: $\det(N) \neq 0$, and the matrix N must be easily invertible so that computing y from $Ny = z$ is computationally efficient. In order to study how fast the iterations are converging to the correct solution, we introduce the matrix $M = N^{-1}P$, and the error vectors $e^{(n)} = x^{(n)} - x$. Substracting equation 8.22 from equation 8.23, we obtain an equation governing the evolution of the error, thus:

$$e^{(n)} = Me^{(n-1)} = M^2 e^{(n-2)} = \ldots = M^n e^{(0)} \tag{8.24}$$

where $e^{(0)}$ is the initial error. Thus, it is clear that a sufficient condition for convergence, i.e. that $\lim_{n\to\infty} e^{(n)} = 0$, is that $\lim_{n\to\infty} M^n = O$. This is also necessary for the method to converge for all $e^{(0)}$. The condition for a matrix to be convergent is that its spectral radius $\rho(M) < 1$. (Reminder: the spectral radius of a matrix M is defined as the maximum eigenvalue in magnitude: $\rho(M) = \max_i |\lambda_i|$). Since computing the eigenvalues is difficult usually, and since the spectral radius is a lower bound for any matrix norm, we often revert to imposing conditions on the matrix norm to enforce convergence; thus

$$\rho(M) \leq \|M\| < 1. \tag{8.25}$$

In particular, it is common to use either the 1- or infinity-norms since these are the simplest to calculate.

The spectral radius is also useful in defining the rate of convergence of the method. In fact since, using equation 8.24, one can bound the norm of the error by:

$$\|e^{(n)}\| \leq \|M^n\| \|e^{(0)}\| \tag{8.26}$$
$$\|e^{(n)}\| \leq [\rho(M)]^n \|e^{(0)}\| \tag{8.27}$$

Thus the number of iteration needed to reduce the initial error by a factor of α is $n \geq \ln \alpha / \ln[\rho(M)]$. Thus, a small spectral radius reduces the number of iterations (and hence CPU cost) needed for convergence.

Jacobi Method

The Jacobi method derived for the Poisson equation can be generalized by defining the matrix N as the diagonal of matrix A:

$$N = D, \quad P = A - D \tag{8.28}$$

The matrix $D = a_{ij}\delta_{ij}$, where δ_{ij} is the Kronecker delta. The matrix $M = D^{-1}(D - A) = I - D^{-1}A$ In component form the update takes the form:

$$x_i^n = \frac{1}{a_{ii}} \sum_{\substack{j=1 \\ j \neq i}}^{K} a_{ij} x_j^{n-1} \tag{8.29}$$

The procedure can be employed if $a_{ii} \neq 0$, i.e. all the diagonal elements of A are different from zero. The rate of convergence is in general difficult to obtain since the eigenvalues are not easily available. However, the infinity and/or 1-norm of M can be easily obtained:

$$\begin{aligned} \rho(M) &\leq \min(\|M\|_1, \|M\|_\infty) && (8.30) \\ \|M\|_1 &= \max j \sum_{\substack{i=1 \\ i \neq j}} \left| \frac{a_{ij}}{a_{ii}} \right| < 1 && (8.31) \\ \|M\|_\infty &= \max i \sum_{\substack{j=1 \\ j \neq i}} \left| \frac{a_{ij}}{a_{jj}} \right| < 1 && (8.32) \\ & && (8.33) \end{aligned}$$

Gauss-Seidel Method

A change of splitting leads to the Gauss-Seidel method. Thus we split the matrix into a lower triangular matrix, and an upper triangular matrix:

$$N = \begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{K1} & a_{K2} & \cdots & a_{KK} \end{pmatrix}, \quad P = N - A \tag{8.34}$$

A slightly different form of writing this splitting is as $A = D + L + U$ where D is again the diagonal part of A, L is a strictly lower triangular matrix, and U is a strictly upper triangular matrix; here $N = D + L$. The matrix notation for the SOR iteration is a little complicated but can be computed:

$$\begin{aligned} x^m &= Mx^{m-1} + (I + \alpha D^{-1}L)^{-1}\alpha D^{-1}b && (8.35) \\ M &= (I + \alpha D^{-1}L)^{-1}[(1-\alpha)I - \alpha D^{-1}U] && (8.36) \end{aligned}$$

8.2 Krylov Method-CG

Consider the system of equations $Ax = b$, where the matrix A is a symmetric positive definite matrix. The solution of the system of equations is equivalent to minimizing the functional:

$$\Phi(x) = \frac{1}{2}x^T Ax - x^T b. \tag{8.37}$$

The extremum occurs for $\frac{\partial \Phi}{\partial x} = Ax - b = 0$, thanks to the symmetry of the matrix, and the positivity of the matrix shows that this extremum is a minimum, i.e. $\frac{\partial^2 \Phi}{\partial x^2} = A$. The iterations have the form:

$$x_k = x_{k-1} + \alpha p_k \tag{8.38}$$

where x_k is the k^{th} iterate, α is a scalar and p_k are the search directions. The two parameters at our disposal are α and p. We also define the residual vector $r_k = b - Ax_k$. We can now relate $\Phi(x_k)$ to $\Phi(x_{k-1})$:

$$\begin{aligned}\Phi(x_k) &= \frac{1}{2}x_k^T Ax_k - x_k^T b \\ &= \Phi(x_{k-1}) + \alpha x_{k-1}^T Ap_k + \left(\frac{\alpha^2}{2}p_k^T Ap_k - \alpha p_k^T b\right)\end{aligned} \tag{8.39}$$

For an efficient iteration algorithm, the 2nd and 3rd terms on the right hand side of equation 8.39 have to be minimized separately. The task is considerably simplified if we require the search directions p_k to be A-orthogonal to the solution:

$$x_{k-1}^T Ap_k = 0. \tag{8.40}$$

The remaining task is to choose α such that the last term in 8.39 is minimized. It is a simple matter to show that the optimal α occurs for

$$\alpha = \frac{p_k^T b}{p_k^T Ap_k}, \tag{8.41}$$

and that the new value of the functional will be:

$$\Phi(x_k) = \Phi(x_{k-1}) - \frac{1}{2}\frac{(p_k^T b)^2}{p_k^T Ap_k}. \tag{8.42}$$

We can use the orthogonality requirement 8.40 to rewrite the above two equations as:

$$\alpha = \frac{p_k^T r_{k-1}}{p_k^T Ap_k}, \quad \Phi(x_k) = \Phi(x_{k-1}) - \frac{1}{2}\frac{(p_k^T r_{k-1})^2}{p_k^T Ap_k}. \tag{8.43}$$

The remaining task is defining the iteration is to determine the algorithm needed to update the search vectors p_k; the latter must satisfy the orthogonality condition 8.40, and must maximum the decrease in the functional. Let us denote by P_k the matrix formed by the $(k-1)$ column vectors p_i, then since the iterates are linear combinations of the search vectors,we can write:

$$x_{k-1} = \sum_{i=1}^{k-1} \alpha_i p_i = P_{k-1} y \tag{8.44}$$

$$P_{k-1} = \begin{bmatrix} p_1 & p_2 & \dots & p_{k-1} \end{bmatrix} \tag{8.45}$$

$$y = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{k-1} \end{pmatrix} \tag{8.46}$$

We note that the solution vector x_{k-1} belongs to the space spanned by the search vectors $p_i, i = 1, \dots, k-1$. The orthogonality property can now be written as $y^T P_{k-1}^T A p_k = 0$. This property is easy to satisfy if the new search vector p_k is A-orthogonal to all the previous search vectors, i.e. if $P_{k-1}^T A p_k = 0$. The algorith can now be summarized as follows: First we initialize our computations by defining our initial guess and its residual; second we perform the following iterations:

```
while $\|r_k\| < \epsilon$:
    1. Choose $p_k$ such that $p_i^T A p_k = 0, \forall i < k$, and maximize $p_k^T r_{k-1}$.
    2. Compute the optimal $\alpha_k = \frac{p_k^T r_{k-1}}{p_k^T A p_k}$
    3. Update the guess $x_k = x_{k-1} + \alpha_k p_k$, and residual $r_k = r_{k-1} - \alpha_k A p_k$
end
```

A vector p_k which is A-orthogonal to all previous search, and such that $p_k^T r_{k-1} \neq 0$, vectors can always be found. Note that if $p_k^T r_{k-1} = 0$, then the functional does not decrease and the minimum has been reached, i.e. the system has been solved. To bring about the largest decrease in $\Phi(x_k)$, we must maximize the inner product $p_k^T r_{k-1}$. This can be done by minimizing the angle between the two vectors p_k and r_{k-1}, i.e. minimizing $\|r_{k-1} - p_k\|$.

Consider the following update for the search direction:

$$p_k = r_{k-1} - A P_{k-1} z_{k-1} \tag{8.47}$$

where z_{k-1} is chosen to minimize $J = \|r_{k-1} - A P_{k-1} z\|^2$. It is easy to show that the minimum occurs for

$$P_{k-1}^T a^T A P_{k-1} z = P_{k-1}^T A^T r_{k-1}, \tag{8.48}$$

and under this condition $p_k^T A P_{k-1} = 0$, and $\|p_k - r_k\|$ is minimized. We have the following property:

$$P_k^T r_k^T = 0, \tag{8.49}$$

i.e. the search vectors are orthogonal to the residual vectors. We note that

$$\text{Span}\{p_1, p_2, \dots, p_k\} = \text{Span}\{r_0, r_1, \dots, r_{k-1}\} = \text{Span}\{b, Ab, \dots, A^{k-1} b\} \tag{8.50}$$

i.e. these different basis sets are spanning the same vector space. The final steps in the conjugate gradient algorithm is that the search vectors can be written in the simple form:

$$\begin{aligned} p_k &= r_{k-1} + \beta_k p_{k-1}, && (8.51)\\ \beta_k &= -\frac{p_{k-1}^T A r_{k-1}}{p_{k-1}^T A p_{k-1}}, && (8.52)\\ \alpha_k &= -\frac{r_{k-1}^T r_{k-1}}{p_k^T A p_k} && (8.53) \end{aligned}$$

The conjugate gradient algorithm can now be summarized as

```
Initialize: $r = b - Ax$, $p = r$, $\rho = \|r\|^2$.
while $\rho < \epsilon$:
    $k \leftarrow k + 1$
    $w \leftarrow Ap$
    $\alpha = \frac{\|r\|^2}{p^T w}$
    update guess: $x \leftarrow x + \alpha p$
    update residual: $r \leftarrow r - \alpha w$
    new residual norm: $\rho' \leftarrow \|r\|^2$
    update search direction: $\beta = \rho'/\rho$, $p \leftarrow r + \beta p$
    update residual norm: $\rho \leftarrow \rho'$.
end
```

It can be shown that the error in the CG algorithm after k iteration is bounded by:

$$\|x_k - x\|_A \leq 2\|x_0 - x\|_A \left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k \tag{8.54}$$

where $\kappa(A)$ is the condition number,

$$\kappa(A) = \|A\|\|A^{-1}\| = \frac{\max_i(|\lambda_i|)}{\min_i(|\lambda_i|)}, \tag{8.55}$$

the ratio of maximum eigenvalue to minimum eigenvalue. The error estimate uses the A-norm: $\|w\|_A = \sqrt{w^T A w}$. Note that for very large condition numbers, $1 \ll \kappa$, the rate of residual decrease approaches 1:

$$\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right) \approx 1 - \frac{2}{\sqrt{\kappa}} \approx 1 \tag{8.56}$$

Hence the number of iterations needed to reach convergence increases. For efficient iterations κ must be close to 1, i.e. the eigenvalues cluster around the unit circle. The problem becomes one of converting the original problem $Ax = b$ into $\tilde{A}\tilde{x} = \tilde{b}$ with $\kappa(\tilde{A}) \approx 1$.

8.3 Direct Methods

8.3.1 Periodic Problem

We will be mainly concerned with FFT based direct methods. These are based on the efficiency of the FFT to diagonalize the matrix A trough the transformation $D = Q^{-1}AQ$, where Q is the unitary matrix made up of the eigenvectors of A. These eigenvector depend on the shape, boundary conditions of the problem. The method is applicable to seperable elliptic problems only. For a doubly periodic problem, we can write that:

$$u_{jk} = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{u}_{mn}\, e^{-i\frac{2\pi jm}{M}}\, e^{-i\frac{2\pi kn}{N}} \tag{8.57}$$

where $\hat{u}_{mn}$ are the Discrete Fourier Coefficients. A similar expression can be written for the right hand side function f. Replace the Fourier expression in the original Laplace equation we get:

$$\frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{u}_{mn}\, e^{-i\frac{2\pi jm}{M}}\, e^{-i\frac{2\pi kn}{N}} \left(e^{-i\frac{2\pi m}{M}} + e^{i\frac{2\pi m}{M}} e^{-i\frac{2\pi n}{N}} + e^{i\frac{2\pi n}{N}}\right) = \Delta^2 \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} \hat{f}_{mn} e^{-i\frac{2\pi jm}{M}} e^{-i\frac{2\pi kn}{N}} \tag{8.58}$$

Since the Fourier functions form an orthogonal basis, the Fourier coefficients should match individually. Thus, one can obtain the following expression for the unknowns $\hat{u}_{mn}$:

$$\hat{u}_{mn} = \frac{\Delta^2 \hat{f}_{mn}}{2\left(\cos\frac{2\pi m}{M} + \cos\frac{2\pi n}{N} - 2\right)},\ m = 0, 1, \ldots, M-1,\ n = 0, 1, \ldots, N-1 \tag{8.59}$$

8.3.2 Dirichlet Problem

For a Dirichlet problem with homogeneous boundary conditions on all boundaries, the following expansion satisfies the boundary conditions identically:

$$u_{jk} = \frac{1}{MN}\sum_{m=1}^{M-1}\sum_{n=1}^{N-1} \hat{u}_{mn}\ \sin\frac{\pi jm}{M}\ \sin\frac{\pi kn}{N} \tag{8.60}$$

Again, the sine basis function are orthogonal and hence the Fourier coefficients can be computed as

$$\hat{u}_{mn} = \frac{\Delta^2 \hat{f}_{mn}}{2\left(\cos\frac{\pi m}{M} + \cos\frac{\pi n}{N} - 2\right)},\ m = 1, \ldots, M-1,\ n = 1, \ldots, N-1 \tag{8.61}$$

Again, the efficiency of the method rests on the FFT algorithm. Specialized routines for sine-transforms are available.

Chapter 9

Nonlinear equations

Linear stability analysis is not sufficient to establish the stability of finite difference approximation to nonlinear PDE's. The nonlinearities add a severe complications to the equations by providing a continuous source for the generation of small scales. Here we investigate how to approach nonlinear problems, and ways to mitigate/control the growth of nonlinear instabilities.

9.1 Aliasing

In a constant coefficient linear PDE, no new Fourier components are created that are not present either in the initial and boundary conditions conditions, or in the forcing functions. This is not the case if nonlinear terms are present or if the coefficients of a linear PDE are not constant. For example, if two periodic functions: $\phi = e^{ik_1 x_j}$ and $\psi = e^{ik_2 x_j}$, are multiplied during the course of a calculation, a new Fourier mode with wavenumber $k_1 + k_2$ is generated:

$$\phi\psi = e^{i(k_1+k_2)x_j}. \tag{9.1}$$

The new wave generated will be shorter then its parents if $k_{1,2}$ have the same sign, i.e. $\frac{2\pi}{k_1+k_2} < \frac{2\pi}{k_{1,2}}$. The representation of this new wave on the finite difference grid can become problematic if its wavelength is smaller then twice the grid spacing. In this case the wave can be mistaken for a longer wave via **aliasing**.

Aliasing occurs because a function defined on a discrete grid has a limit on the shortest wave number it can represent; all wavenumbers shorter then this limit appear as a long wave. The shortest wavelength representable of a finite difference grid with step size Δx is $\lambda_s = 2\Delta x$ and hence the largest wavenumber is $k_{\max} = 2\pi/\lambda_s = \pi/\Delta x$. Figure 9.1 shows an example of a long and short waves aliased on a finite difference grid consisting of 6 cells. The solid line represents the function $\sin\frac{6\pi x}{4\Delta x}$ and is indistinguishable from the function $\sin\frac{\pi x}{4\Delta x}$ (dashed line): the two functions coincide at all points of the grid (as marked by the solid circles). This coincidence can be explained by re-writing each Fourier mode as:

$$e^{ikx_j} = e^{ikj\Delta x} = e^{ikj\Delta x}\left[e^{i2\pi n}\right]^j = e^{i\left(k+\frac{2\pi n}{\Delta x}\right)j\Delta x}, \tag{9.2}$$

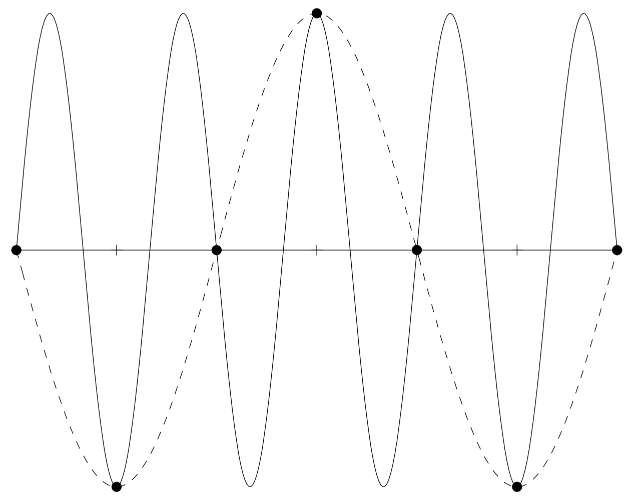


Figure 9.1: Aliasing of the function $\sin\frac{6\pi x}{4\Delta x}$ (solid line) and the function $sin\frac{6\pi x}{4\Delta x}$ (dashed line). The two functions have the same values on the FD grid $j\Delta x$.

where $n = 0, \pm 1, \pm 2, \ldots$ Relation 9.2 is satisfied at all the FD grid points $x_j = j\Delta x$; it shows that all waves with wavenumber $k + \frac{2\pi n}{\Delta x}$ are indistinguishable on a finite difference grid with grid size Δx. In the case shown in figure 9.1, the long wave has length $4\Delta x$ and the short wave has length $4\Delta x/3$, so that the equation 9.2 applies with $n = -1$.

Going back to the example of the quadratic nonlinearity $\phi\psi$, although the individual functions, ϕ and ψ, are representable on the FD grid, i.e. $|k_{1,2}| \leq \pi/\Delta x$, their product may not be since now $|k_1 + k_2| \leq 2\pi/\Delta x$. In particular, if $\pi/\Delta x \leq |k_1 + k_2| \leq 2\pi/\Delta x$, the product will be unresolvable on the discrete grid and will be aliased into wavenumber $\tilde{k}$ given by

$$\tilde{k} = \begin{cases} k_1 + k_2 - \frac{2\pi}{\Delta x}, & \text{if } k_1 + k_2 > \frac{\pi}{\Delta x} \\ k_1 + k_2 + \frac{2\pi}{\Delta x}, & \text{if } k_1 + k_2 < -\frac{\pi}{\Delta x} \end{cases} \tag{9.3}$$

Note that very short waves are aliased to very long waves: $\tilde{k} \to 0$ when $|k_{1,2}| \to \frac{\pi}{\Delta x}$. The aliasing wavenumber can be visualized by looking at the wavenumber axis shown in

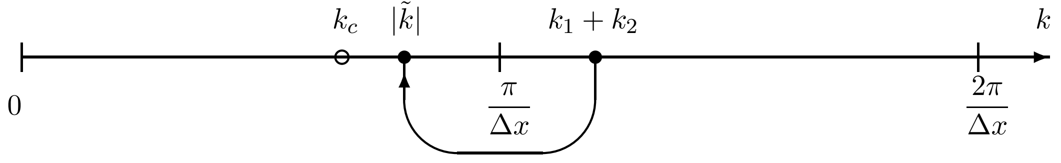


Figure 9.2: Folding of short waves into long waves.

figure 9.2; note that $k_1 + k_2$ and $|\tilde{k}|$ are symmetrically located about the point $\frac{\pi}{\Delta x}$. There exist a cut-off wavenumber k_c whereby all longer wavelength are aliased to $|\tilde{k}| > kc$; thus since $k < k_c$, then $(k_1 + k_2) < 2k_c$. Thus if $k_1 + k_2 > k_{\max}$ the product will be aliased into $|\tilde{k}| = k_{\max} - (2k_c - k_{\max})$, and the latter must satisfy $|\tilde{k}| > k_c$, and we end up with

$$k_c < \frac{2}{3}k_{\max} \tag{9.4}$$

For a finite difference grid this is equivalent to $k_c < \frac{2\pi}{3\Delta x}$.

9.2 1D Burger equation

The 1D Burger equation is the simplest model of nonlinearities as found in the Navier-Stokes momentum equations:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0, \quad 0 \le x \le L \tag{9.5}$$

Multiplying the above equation by u^m where $m \ge 0$ we can derive the following conservation law:

$$\frac{\partial u^{m+1}}{\partial t} + \frac{1}{m+2}\frac{\partial u^{m+2}}{\partial x} = 0. \tag{9.6}$$

The above equation is a conservation equation for all moments of the solution; in particular for $m = 0$, we have the momentum conservation, and for $m = 1$ energy conservation. The spatial integral yields:

$$\frac{\partial\left(\int_L u^{m+1}dx\right)}{\partial t} + \frac{u^{m+2}|_L - u^{m+2}|_0}{m+2} = 0. \tag{9.7}$$

which shows that the global budget of u^{m+1} depends on the boundary values, and is zero for periodic boundary conditions.

We will explore the impact of spatial differencing on the continuum conservation properties of the energy (second moment). A central difference scheme of the advective form of the equation, equation 9.5, yields:

$$\frac{\partial u_j}{\partial t} = -u_j\frac{u_{j+1} - u_{j-1}}{2\Delta x}. \tag{9.8}$$

Multiplying by u_j and summing over the interval we get:

$$\frac{\partial \sum_{j=0}^N u_j^2/2}{\partial t} = -\frac{1}{2}\sum_{j=0}^{N}(u_j u_{j+1} - u_j u_{j-1}). \tag{9.9}$$

Notice that the terms within the summation sign do not cancel out, and hence energy is not conserved. Likewise, a finite difference approximation to the conservative form:

$$\frac{\partial u_j}{\partial t} = -\frac{u_{j+1}^2 - u_{j-1}^2}{4\Delta x}, \tag{9.10}$$

is not conserving as its discrete energy equation attests:

$$\frac{\partial \sum_{j=0}^N u_j^2/2}{\partial t} = -\frac{1}{4}\sum_{j=0}^{N}(u_j u_{j+1}^2 - u_j u_{j-1}^2). \tag{9.11}$$

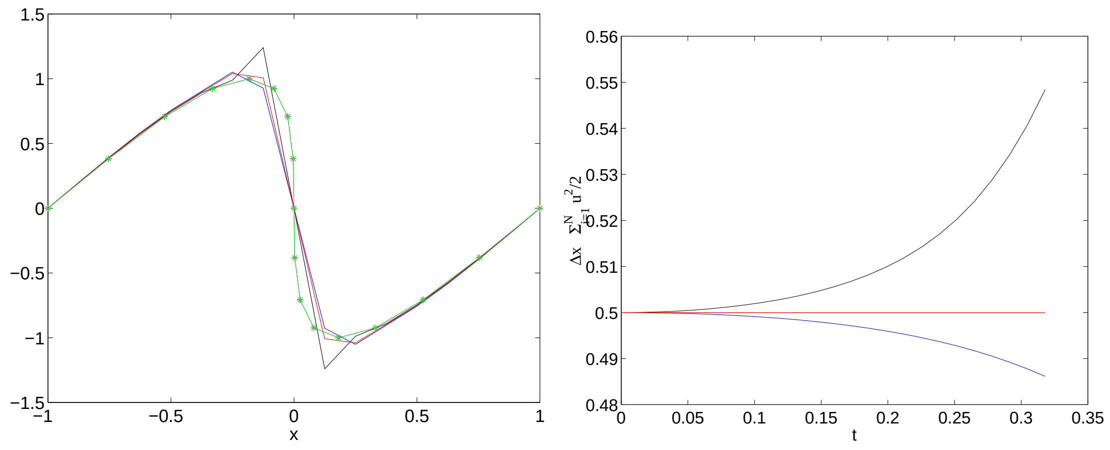


Figure 9.3: Left: Solution of the inviscid Burger equation at $t = 0.318 < 1/\pi$ using the advective form (black), momentum conserving form (blue), and energy conserving form (red); the analytical solution is superimposed in Green. The initial conditions are $u(x, 0) = -\sin \pi x$, the boundary conditions are periodic, the time step is $\Delta t = 0.01325$, and $\Delta x = 2/16$; RK4 was used for the time integration. Right: Energy budget for the different Burger schemes: red is the energy conserving, blue is the momentum conserving, and black is the advective form.

The only energy conserving available for the Burger equation is the following:

$$\frac{\partial u_j}{\partial t} = -\frac{u_{j+1} + u_j + u_{j-1}}{3} \frac{u_{j+1} - u_{j-1}}{2\Delta x}. \tag{9.12}$$

where the advection velocity is a three-term average of the velocity at the central point. Its discrete energy equation is given by:

$$\frac{\partial \sum_{j=0}^{N} u_j^2/2}{\partial t} = -\frac{1}{6} \sum_{j=0}^{N} \left[u_j u_{j+1}(u_j + u_{j+1}) - u_j u_{j-1}(u_j + u_{j-1}) \right] \tag{9.13}$$

where the term inside the summation sign does cancel out. Figure 9.3 shows solutions of the Burger equations using the 3 schemes listed above. The advective form, shown in black, does not conserve energy and exhibits oscillations near the front region. The oscillations are absent in the both the flux and energy conserving forms. The flux form, equation 9.10, exhibits a decrease in the energy and a decrease in the amplitude of the waves. Note that the solution is shown just prior to the formation of the shock at time $t = 0.318 < 1/\pi$.

9.3 Quadratic Conservation

It is obvious that building quadratic conserving schemes depends highly on the system of equations considered. The remaining sections will be devoted to equations commonly found in the CFD/oceanic literature. We will concentrate on the following system of equations:

$$\mathbf{v}_t + \mathbf{v}\cdot\nabla\mathbf{v} + g\nabla\eta = \mathbf{f} \tag{9.14}$$

$$\alpha\eta_t + \nabla\cdot(h\mathbf{v}) = 0 \tag{9.15}$$

Equation 9.14 is the momentum equation and 9.15 is the continuity equation in a fluid moving with velocity $\mathbf{v}$ and subject to a pressure η. The parameter α controls the compressibility of the system; for $\alpha = 0$ we recover the incompressible equations, and for $\alpha = 1$ the shallow water equations. The parameter h is the thickness of the fluid layer, and $\mathbf{f}$ is a term lumping momentum sources and sinks (including dissipation).

The system 9.14-9.15 imply conservation laws for energy, vorticity and enstrophy in the incompressible case, and for energy, potential vorticity and potential enstrophy in the compressible case, when the source terms on the right hand sides are identically zero. The question is: Is it possible to enforce the conservation of these higher order quantities (mainly quadratic in the unknown variables) in the finite difference approximation? We will look in particular at energy/enstrophy conservation in non-divergent shallow water flow, and on energy/potential vorticity conservation in divergent shallow water equations.

We first begin by defining the following operators:

$$\delta_x u = \frac{u(x+\frac{\Delta x}{2}) - u(x-\frac{\Delta x}{2})}{\Delta x} \tag{9.16}$$

$$\overline{u}^x = \frac{u(x+\frac{\Delta x}{2}) + u(x-\frac{\Delta x}{2})}{2} \tag{9.17}$$

The operators defined above are nothing but the centered difference and averaging operators. It is easy to show that the following relationships holds for any 2 functions a and b defined on a finite difference grids:

$$\delta_x(\overline{a}^x) = \overline{\delta_x a}^x \tag{9.18}$$

$$\overline{a}^x\delta_x b = \delta_x(ab) - \overline{b}^x\delta_x a \tag{9.19}$$

$$a\delta_x b = \delta_x(\overline{a}^x b) - \overline{b\delta_x a}^x \tag{9.20}$$

$$\overline{ab}^x = \overline{a}^x\overline{b}^x + \frac{\Delta x^2}{4}(\delta_x a)\ (\delta_x b) \tag{9.21}$$

$$\overline{\overline{a}^x b}^x = a\overline{b}^x + \delta_x\left(\frac{\Delta x^2}{4}\, b\,\delta_x a\right) \tag{9.22}$$

The first relationship shows that the differencing and averaging operators commute, the second and third relations are the finite difference equivalent of product differentiation rules, and the fourth and fifth are the finite difference equivalent of product averaging. It is easy to show the additional relationships

$$\overline{a}^x\delta_x a = \delta_x\left(\frac{a^2}{2}\right) \tag{9.23}$$

$$a\delta_x\overline{a}^x = \delta_x\left(\frac{\widetilde{a^2}^x}{2}\right),\quad \widetilde{a^2}^x = a\left(x+\frac{\Delta x}{2}\right)\ a\left(x-\frac{\Delta x}{2}\right) \tag{9.24}$$

$$\frac{\widetilde{a}^{2^x}}{2} = (\overline{a}^x)^2 - \frac{\overline{a^2}^x}{2} \tag{9.25}$$

9.4 Nonlinear advection equation

The advection equation

$$\frac{\mathrm{D}T}{\mathrm{D}t} = T_t + \mathbf{v}\cdot\nabla T = 0 \tag{9.26}$$

of a tracer T by a flow field $\mathbf{v}$ that is divergence-free:

$$\nabla\cdot\mathbf{v} = 0, \tag{9.27}$$

is equivalent to the following conservation law:

$$T_t + \nabla\cdot(\mathbf{v}T) = 0. \tag{9.28}$$

Equation 9.26 is called the advective form, and equation 9.28 is called the flux (or conservative) form. The two forms are equivalent in the continuum provided the flow is divergence free. Note that the above statements holds regardless of the linearity/dimensionality of the system. Integration of the flux form over the domain Ω shows that

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\int_\Omega T\mathrm{d}V\right) = -\int_{\partial\Omega}\mathbf{n}\cdot\mathbf{v}T\mathrm{d}S \tag{9.29}$$

where $\mathbf{n}$ is the unit outward normal, $\partial\Omega$ the boundary of the flow domain, $\mathrm{d}S$ is an elemental surface on this boundary, and the boundary integral is the amount of T entering Ω. Equation 9.29 shows that the total inventory of T and Ω depends on the amount of T entering through the boundary. In particular the total budget of T should be constant if the boundary is closed $\mathbf{v}\cdot\mathbf{n} = 0$ or if the domain is periodic.

The above conservation laws imply higher order conservation. To wit, equation 9.26, can be multiplied by T^m (where $m \geq 0$) and the following equation can be derived:

$$\frac{\partial T^{m+1}}{\partial t} + \mathbf{v}\cdot\nabla T^{m+1} = 0, \tag{9.30}$$

i.e. the conservation law imply that all moments of T^m are also conserved. Since the above equation has the same form as the original one, the total inventory of T^m will also be conserved under the same conditions as equation 9.29.

9.4.1 FD Approximation of the advection term

For a general flow field, the FD of the advection form will not conserve the first moment, while the FD of the flux form will. This is easy to see since the flux is:

$$\nabla\cdot(\mathbf{v}T) = \delta_x(uT) + \delta_y(vT). \tag{9.31}$$

The relevant question is: is it possible to come up with a finite difference scheme that will conserve both the first and second moment? Let us look at the following approximation of the flux form

$$\nabla\cdot(\mathbf{v}T) = \delta_x(\overline{u}^x\overline{T}^x) + \delta_y(\overline{v}^y\overline{T}^y). \tag{9.32}$$

Can the term $T\nabla\cdot(\mathbf{v}T)$ be also written in flux form (for then it will be conserved upon summation). We concentrate on the x-component for simplicity:

$$\begin{aligned}
T\delta_x\left(\overline{u}^x\overline{T}^x\right) &= \delta_x\left[\overline{u}^x\left(\overline{T}^x\right)^2\right] - \overline{\overline{u}^x\overline{T}^x\delta_x T}^x && (9.33)\\
&= \delta_x\left[\overline{u}^x\left(\overline{T}^x\right)^2\right] - \overline{\overline{u}^x\delta_x\left(\frac{T^2}{2}\right)}^x && (9.34)\\
&= \delta_x\left[\overline{u}^x\left(\overline{T}^x\right)^2\right] - \overline{\delta_x\left(u\frac{T^2}{2}\right) - \frac{\overline{T^2}^x}{2}\delta_x u}^x && (9.35)\\
&= \delta_x\left[\overline{u}^x\left(\overline{T}^x\right)^2\right] - \delta_x\left(\overline{u}^x\frac{\overline{T^2}^x}{2}\right) - \frac{\Delta x^2}{4}\delta_x\left(\delta_x u\,\delta_x\frac{T^2}{2}\right) + \frac{T^2}{2}\delta_x\overline{u}^x\\
&+ \frac{\Delta x^2}{4}\delta_x\left(\delta_x u\,\delta_x\frac{T^2}{2}\right) && (9.36)\\
&= \delta_x\left[\frac{\widetilde{T^2}^x}{2}\overline{u}^x\right] - \frac{T^2}{2}\delta_x\left(\overline{u}^x\right) && (9.37)
\end{aligned}$$

Equality 9.33 follows from property 9.20, 9.34 from 9.23, 9.35 from 9.19. The second and third terms of equation 9.35 can rewritten with the help of equations 9.21 and 9.24, respectively. The third and fifth terms on the right hand side of equation 9.36 cancel. The final equation 9.37 is obtained by combining the first and second terms of equation 9.36 (remember the operators are linear), and using equation 9.25. A similar derivation can be carries out for the y-component of the divergence:

$$T\delta_y\left(\overline{v}^y\overline{T}^y\right) = \delta_y\left[\frac{\widetilde{T^2}^y}{2}\overline{v}^y\right] - \frac{T^2}{2}\delta_y\left(\overline{v}^y\right) \qquad (9.38)$$

Thus, the semi-discrete second moment conservation becomes:

$$\frac{\partial T^2/2}{\partial t} = -\delta_x\left[\frac{\widetilde{T^2}^x}{2}\overline{u}^x\right] - \delta_y\left[\frac{\widetilde{T^2}^y}{2}\overline{v}^y\right] + \frac{T^2}{2}\left[\delta_x\left(\overline{u}^x\right) + \delta_y\left(\overline{v}^y\right)\right] \qquad (9.39)$$

The first and second term in the semi-discrete conservation equation are in flux form, and hence will cancel out upon summation. The third term on the right hand side is nothing but the discrete divergence constraint. Thus, the second order moment of T will be conserved provided that the velocity field is *discretely* divergence-free.

The following is a FD approximation to $\mathbf{v}\cdot\nabla T$ consistent with the above derivation:

$$\begin{aligned}
u\frac{\partial T}{\partial x} &= \frac{\partial uT}{\partial x} - T\frac{\partial u}{\partial x} && (9.40)\\
&= \delta_x\left(\overline{u}^x\overline{T}^x\right) - T\delta_x\left(\overline{u}^x\right) && (9.41)\\
&= T\delta_x\left(\overline{u}^x\right) + \overline{\overline{u}^x\delta_x T}^x - T\delta_x\left(\overline{u}^x\right) && (9.42)\\
&= \overline{\overline{u}^x\delta_x T}^x && (9.43)
\end{aligned}$$

Thus, we have

$$\mathbf{v}\cdot\nabla T = \overline{\overline{u}^x\delta_x T}^x + \overline{\overline{v}^y\delta_y T}^y \qquad (9.44)$$

9.5 Conservation in vorticity streamfunction formulation

Nonlinear instabilities can develop if energy is falsely generated and persistently channeled towards the shortest resolvable wavelengths. Arakawa Arakawa (1966); Arakawa and Lamb (1977) devised an elegant method to eliminate these artificial sources of energy. His methodology is based on the streamfunction-vorticity formulation of two dimensional, divergence-free fluid flows. The continuity constraint can be easily enforced in 2D flow by introducing a streamfunction, ψ, such that $\mathbf{v} = \mathbf{k} \times \nabla\psi$; in component form this is:

$$u = -\psi_y, \; v = \psi_x \tag{9.45}$$

The vorticity $\zeta = \nabla \times \mathbf{v}$ reduces to

$$\zeta = v_x - u_y = \psi_x x + \psi_y y = \nabla^2\psi \tag{9.46}$$

and the vorticity advection equation can be obtained by taking the curl of the momentum equation, thus:

$$\frac{\partial \nabla^2\psi}{\partial t} = J(\nabla^2\psi, \psi) \tag{9.47}$$

where J stand for the Jacobian operator:

$$J(a, b) = a_x b_y - b_x a_y \tag{9.48}$$

The Jacobian operator possesses some interesting properties

1. It is anti-symmetric, i.e.
$$J(b, a) = -J(a, b) \tag{9.49}$$

2. The Jacobian can be written in the useful forms:
$$\begin{aligned} J(a, b) &= \nabla a \cdot \mathbf{k} \times \nabla b && (9.50)\\ &= \nabla \cdot (\mathbf{k} \times a\nabla b) && (9.51)\\ &= -\nabla \cdot (\mathbf{k} \times b\nabla a) && (9.52) \end{aligned}$$

3. The integral of the Jacobian over a closed domain can be turned into a boundary integral thanks to the above equations
$$\int_\Omega J(a, b)dA = \int_{\partial\Omega} a\frac{\partial b}{\partial s}ds = -\int_{\partial\Omega} b\frac{\partial a}{\partial s}ds \tag{9.53}$$
where s is the tangential direction to the boundary. Hence, the integral of the Jacobian vanishes if either a or b is constant along $\partial\Omega$. In particular, if the boundary is a streamline or a vortex line, the Jacobian integral vanishes. The area-averaged vorticity is hence conserved.

4. The following relations hold:
$$\begin{aligned} aJ(a, b) &= J(\frac{a^2}{2}, b) && (9.54)\\ bJ(a, b) &= J(a, \frac{b^2}{2}) && (9.55) \end{aligned}$$

Thus, the area integrals of $aJ(a,b)$ and $bJ(a,b)$ are zero if either a or b are constant along the boundary.

It is easy to show that enstrophy, $\zeta^2/2$, and kinetic energy, $|\nabla\psi|^2/2$, are conserved if the boundary is closed. We would like to investigate if we can conserve vorticity, energy and enstrophy in the discrete equations. We begin first by noting that the Jacobian in the continuum form can be written in one of 3 ways:

$$\begin{aligned}
J(\zeta,\psi) &= \zeta_x\psi_y - \zeta_y\psi_x && (9.56)\\
&= (\zeta\psi_y)_x - (\zeta\psi_x)_y && (9.57)\\
&= (\psi\zeta_x)_y - (\psi\zeta_y)_x && (9.58)
\end{aligned}$$

We can thus define 3 centered difference approximations to the above definitions:

$$\begin{aligned}
J_1(\zeta,\psi) &= \delta_x\overline{\zeta}^x\delta_y\overline{\psi}^y - \delta_y\overline{\zeta}^y\delta_x\overline{\psi}^x && (9.59)\\
J_2(\zeta,\psi) &= \delta_x\left(\overline{\zeta\delta_y\overline{\psi}^y}^x\right) - \delta_y\left(\overline{\zeta\delta_x\overline{\psi}^x}^y\right) && (9.60)\\
J_3(\zeta,\psi) &= \delta_y\left(\overline{\psi\delta_x\overline{\zeta}^x}^y\right) - \delta_x\left(\overline{\psi\delta_y\overline{\zeta}^y}^x\right) && (9.61)
\end{aligned}$$

It is obvious that J_2 and J_3 will conserve vorticity since they are in flux form; J_1 can also be shown to conserve the first moment since:

$$\begin{aligned}
J_1(\zeta,\psi) &= \delta_x\left[\delta_y\overline{\psi}^{xy}\overline{\zeta}^x\right] - \overline{\overline{\zeta}^x\delta_x\delta_y\overline{\psi}^y}^x - \delta_y\left[\delta_x\overline{\psi}^{xy}\overline{\zeta}^y\right] + \overline{\overline{\zeta}^y\delta_y\delta_x\overline{\psi}^x}^y && (9.62)\\
&= \delta_x\left[\delta_y\overline{\psi}^{xy}\overline{\zeta}^x - \frac{\Delta x^2}{4}\delta_x\delta_y\overline{\psi}^y\,\delta_x\zeta\right] - \delta_y\left[\delta_x\overline{\psi}^{xy}\overline{\zeta}^y - \frac{\Delta y^2}{4}\delta_y\delta_x\overline{\psi}^x\,\delta_y\zeta\right] && (9.63)
\end{aligned}$$

The last equation above shows that J_1 can indeed be written in flux form, and hence vorticity conservation is ensured. Now we turn our attention to the conservation of quadratic quantities, namely, kinetic energy and enstrophy. It is easy to show that J_2 conserves kinetic energy since:

$$\begin{aligned}
\psi J_2(\zeta,\psi) &= \psi\delta_x\left(\overline{\zeta\delta_y\overline{\psi}^y}^x\right) - \psi\delta_y\left(\overline{\zeta\delta_x\overline{\psi}^x}^y\right) && (9.64)\\
&= \delta_x\left(\overline{\psi}^x\overline{\zeta\delta_y\overline{\psi}^y}^x\right) - \delta_y\left(\overline{\psi}^y\overline{\zeta\delta_x\overline{\psi}^x}^y\right) - \overline{\overline{\zeta\delta_y\overline{\psi}^y}^x\delta_x\psi}^x + \overline{\overline{\zeta\delta_x\overline{\psi}^x}^y\delta_y\psi}^y && (9.65)\\
&= \delta_x\left[\overline{\psi}^x\overline{\zeta\delta_y\overline{\psi}^y}^x - \frac{\Delta x^2}{4}\delta_x\psi\,\delta_x\left(\zeta\delta_y\overline{\psi}^y\right)\right]\\
&- \delta_y\left[\overline{\psi}^y\overline{\zeta\delta_x\overline{\psi}^x}^y - \frac{\Delta y^2}{4}\delta_y\psi\,\delta_y\left(\zeta\delta_x\overline{\psi}^x\right)\right] && (9.66)
\end{aligned}$$

Similarly, it can be shown that the average of J_1 and J_2 conserves enstrophy:

$$\zeta\frac{J_1+J_2}{2} = \delta_x\left[\overline{\zeta}^x\overline{\zeta\delta_y\overline{\psi}^y}^x - \frac{\Delta x^2}{4}\delta_x\zeta\;\delta_x\left(\zeta\delta_y\overline{\psi}^y\right)\right] - \delta_y\left[\overline{\zeta}^y\overline{\zeta\delta_x\overline{\psi}^x}^y - \frac{\Delta y^2}{4}\delta_y\zeta\;\delta_y\left(\zeta\delta_x\overline{\psi}^x\right)\right] \tag{9.67}$$

Notice that the finite difference Jacobians satisfy the following property:

$$J_1(\psi,\zeta) = -J_1(\zeta,\psi) \tag{9.68}$$
$$J_2(\psi,\zeta) = -J_3(\zeta,\psi). \tag{9.69}$$

Hence, from equation 9.66 $\zeta J_3(\zeta,\psi)$ can be written in flux form, and from equation 9.67 $\psi\frac{J_1+J_3}{2}$ can also be written in flux form. These results can be tabulated:

energy conserving	enstrophy conserving
J_2	J_3
$\frac{J_1+J_3}{2}$	$\frac{J_1+J_2}{2}$

Notice that any linear combination of the energy conserving schemes will also be energy conserving, likewise for the enstrophy conserving forms. Thus, it is possible to find an energy and enstrophy conserving Jacobian if we can find two constants α and β such that:

$$J_A = \alpha J_2 + (1-\alpha)\frac{J_1+J_3}{2} = \beta J_3 + (1-\beta)\frac{J_1+J_2}{2} \tag{9.70}$$

Equating like terms in J_1, J_2 and J_3 we can solve the system of equation. The final result can be written as:

$$J_A = \frac{J_1+J_2+J_3}{3} \tag{9.71}$$

Equation 9.71 defines the Arakawa Jacobian, named in honor of Akio Arakawa who proposed it first. The expression for J_A in terms of the FD computational stencil is a little complicated. We give the expression for a square grid spacing:$\Delta x = \Delta y$.

$$\begin{aligned}
12\Delta x\Delta y J_A(\zeta,\psi) &= (\zeta_{j+1,k}+\zeta_{j,k})(\psi_{j+1,k+1}+\psi_{j,k+1}-\psi_{j+1,k-1}-\psi_{j,k-1})\\
&+ (\zeta_{j,k+1}+\zeta_{j,k})(\psi_{j-1,k+1}+\psi_{j-1,k}-\psi_{j+1,k+1}-\psi_{j+1,k})\\
&+ (\zeta_{j-1,k}+\zeta_{j,k})(\psi_{j-1,k-1}+\psi_{j,k-1}-\psi_{j-1,k+1}-\psi_{j,k-1})\\
&+ (\zeta_{j,k-1}+\zeta_{j,k})(\psi_{j+1,k-1}+\psi_{j+1,k}-\psi_{j+1,k-1}-\psi_{j-1,k})\\
&+ (\zeta_{j+1,k+1}+\zeta_{j,k})(\psi_{j,k+1}-\psi_{j+1,k})\\
&+ (\zeta_{j-1,k+1}+\zeta_{j,k})(\psi_{j-1,k}-\psi_{j,k+1})\\
&+ (\zeta_{j-1,k-1}+\zeta_{j,k})(\psi_{j,k-1}-\psi_{j-1,k})\\
&+ (\zeta_{j+1,k-1}+\zeta_{j,k})(\psi_{j+1,k}-\psi_{j,k-1})
\end{aligned} \tag{9.72}$$

Note, the terms in $\zeta_{j,k}$ cancel out, the expression for J_A can use $\pm\zeta_{j,k}$ or no value at all.

An important property of the Arakawa Jacobian is that it inhibits the pile up of energy at small scales, a consequence of conserving enstrophy and energy. Since both quantities are conserved, so is their ratio which can be used to define an average wavenumber κ:

$$\kappa^2 = \frac{\int_A \|\nabla\psi\|^2 dA}{\int_A (\nabla^2\psi)^2 dA}. \tag{9.73}$$

For the case of a periodic problem, the relationship between the above ratio and wavenumbers can be easily demonstrated by expanding the streamfunction and vorticity in terms

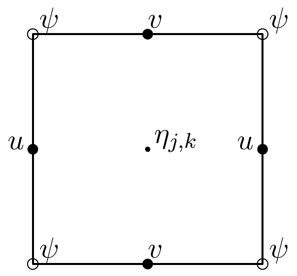


Figure 9.4: Configuration of unknowns on an Arakawa C-Grid. The C-Grid velocity points $u_{i+\frac{1}{2}}$ and $v_{i,j+\frac{1}{2}}$ are located a distance $d/2$ to the left and top, respectively, of pressure point $\eta_{i,j}$.

of the Fourier components:

$$\psi = \sum_m \sum_n \hat{\psi}_{m,n} e^{imx} e^{iny} \tag{9.74}$$

$$\zeta = -\sum_m \sum_n (m^2+n^2) \hat{\psi}_{m,n} e^{imx} e^{iny} \tag{9.75}$$

where $\hat{\psi}_{m,n}$ are the complex Fourier cofficients and the computational domain has been mapped into the square domain $[0, 2\pi]^2$. Using the orthogonality of the Fourier modes, it is easy to show that the ratio κ becomes

$$\kappa^2 = \frac{\sum_m \sum_n (m^2+n^2)^2 |\hat{\psi}_{m,n}|^2}{\sum_m \sum_n (m^2+n^2) |\hat{\psi}_{m,n}|^2} \tag{9.76}$$

The implication of equation 9.76 is that there can be no one-way cascade of energy in wavenumber space; if some "local" cascading takes place from one part of the spectrum to the other; there must be a compensating shift of energy in another part.

9.6 Conservation in primitive equations

The Arakawa Jacobian enforces enstrophy and energy conservation when the vorticity-streamfunction formulation is discretized. There are situatiions (in the presence of islands, for example) where the vorticity-streamfunction formulation is not appropriate and one must revert to discretizing the primitive equations (momentum and volume conservation). The question becomes what is the appropriate FD discretization of the nonlinear momentum advection that can guarantee conservation of kinetic energy and enstrophy? The process to obtaining these differencing schemes is to backtrap the steps that lead to the derivation of the Jacobian operator in the vorticity equation. The Jacobian in the vorticity equation arises from the cross-differentiation of the nonlinear advection terms:

$$\frac{\partial \mathbf{v}\cdot\nabla v}{\partial x} - \frac{\partial \mathbf{v}\cdot\nabla u}{\partial y} = \frac{\partial}{\partial x}\left(u\frac{\partial v}{\partial x} - u\frac{\partial u}{\partial y}\right) + \frac{\partial}{\partial y}\left(v\frac{\partial v}{\partial x} - v\frac{\partial v}{\partial y}\right) \tag{9.77}$$

$$= \frac{\partial}{\partial x}\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) - \frac{\partial}{\partial y}\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) \tag{9.78}$$

In order to make use of the results obtained using the vorticity-streamfunction form, it is usefull to introduce a fictitious streamfunction in the primitive variables using a staggered grid akin to the Arakawa C-grid shown in figure 9.4. The staggered velocity are defined with respect to the "fictitious streamfunction":

$$u = -\delta_y\psi, \quad v = \delta_x\psi, \quad \zeta = \delta_x v - \delta_y u \tag{9.79}$$

The energy conserving Jacobian can thus be written as:

$$J_2(\zeta,\psi) = -\delta_x\left(\overline{\overline{u}^y\delta_x v}^x + \overline{\overline{v}^y\delta_y v}^y\right) + \delta_y\left(\overline{\overline{u}^x\delta_x u}^x + \overline{\overline{v}^x\delta_y u}^y\right) \tag{9.80}$$

Comparing equations 9.80 and 9.77 we can deduce the following energy conserving momentum advection operators:

$$\left\{\begin{array}{rcl} \mathbf{v}\cdot\nabla u &=& \overline{\overline{u}^x\delta_x u}^x + \overline{\overline{v}^x\delta_y u}^y \\ \mathbf{v}\cdot\nabla v &=& \overline{\overline{u}^y\delta_x v}^x + \overline{\overline{v}^y\delta_y v}^y \end{array}\right. \tag{9.81}$$

In a similar manner, the enstrophy conserving Jacobian can be re-written as:

$$\frac{J_1+J_2}{2} = -\delta_x\left(\overline{u}^{xy}\delta_x\overline{v}^x + \overline{v}^{yy}\delta_y\overline{v}^y\right) + \delta_y\left(\overline{u}^{xx}\delta_x\overline{u}^x + \overline{v}^{xy}\delta_y\overline{u}^y\right), \tag{9.82}$$

and we can deduce the following enstrophy-conserving operators:

$$\left\{\begin{array}{rcl} \mathbf{v}\cdot\nabla u &=& \overline{u}^{xx}\delta_x\overline{u}^x + \overline{v}^{xy}\delta_y\overline{u}^y \\ \mathbf{v}\cdot\nabla v &=& \overline{u}^{xy}\delta_x\overline{v}^x + \overline{v}^{yy}\delta_y\overline{v}^y \end{array}\right. \tag{9.83}$$

If either 9.81 or 9.83 is used in the momentum equation, and if the flow is discretely divergence-free, then energy or enstrophy is conserved in the same manner as it is in the vorticity equation through J_2 or $\frac{J_1+J_2}{2}$. Stated differently, only the divergent part of the velocity field is capable of creating or destroying energy or enstrophy, in perfect analogy to the behavior of the continuous equations.

We would like to have an operator that conserves both energy and enstrophy, which means converting J_3. This is considerably harder. We skip the derivation and show the result, see Arakawa and Lamb for details Arakawa and Lamb (1977)):

$$\nabla\cdot\mathbf{v}u = \frac{2}{3}\left[\delta_x\left(\overline{u}^{xyy}\overline{u}^x\right) + \delta_y\left(\overline{v}^{xyy}\overline{u}^y\right)\right] + \frac{1}{3}\left[\delta_{x'}\left(\overline{u'}^y\overline{u}^{x'}\right) + \delta_{y'}\left(\overline{v'}^y\overline{u}^{y'}\right)\right] \tag{9.84}$$

$$\nabla\cdot\mathbf{v}v = \frac{2}{3}\left[\delta_x\left(\overline{u}^{xxy}\overline{v}^x\right) + \delta_y\left(\overline{v}^{xxy}\overline{v}^y\right)\right] + \frac{1}{3}\left[\delta_{x'}\left(\overline{u'}^x\overline{v}^{x'}\right) + \delta_{y'}\left(\overline{v'}^x\overline{v}^{y'}\right)\right] \tag{9.85}$$

$$u' = -\delta_{y'}\psi = \frac{\overline{u}^x + \overline{v}^y}{\sqrt{2}} \tag{9.86}$$

$$v' = \delta_{x'}\psi = \frac{-\overline{u}^x + \overline{v}^y}{\sqrt{2}} \tag{9.87}$$

The (x', y') coordinate system is rotated 45 degrees counterclockwise to the (x, y) coordinate system; i.e. it is in the diagonal directions w.r.t. to the original axis.

9.7 Conservation for divergent flows

So far we have dealt mostly with advection operators which handle conservation laws appropriate for divergence-free flows. There are situations, such as flows over obstacles, where the divergent velocity plays a significant role. Under such conditions, it might be important to incorporate conservation laws appropriate to divergent flows. In the following we will consider primarily the shallow water equations in a rotating frame:

$$\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\nabla\mathbf{v} + f\mathbf{k}\times\mathbf{v} + g\nabla\eta = 0 \tag{9.88}$$

$$\frac{\partial h}{\partial t} + \nabla\cdot(h\mathbf{v}) = 0 \tag{9.89}$$

where the fluid depth h is the sum of the resting layer thickness H and the surface displacement η. One of the more important conservation principles is the conservation of potential vorticity, q:

$$\frac{\partial q}{\partial t} + \mathbf{v}\cdot\nabla q = 0, \quad q = \frac{\zeta + f}{h} \tag{9.90}$$

The latter is derived by re-writing the nonlinear advection of momentum in the form

$$\mathbf{v}\cdot\nabla\mathbf{v} = \nabla\frac{\mathbf{v}\cdot\mathbf{v}}{2} - \mathbf{v}\times\zeta\mathbf{k} \tag{9.91}$$

prior to taking the curl of the momentum equation 9.88 to arrive at:

$$\frac{\partial \zeta + f}{\partial t} + \nabla\cdot[(\zeta+f)\mathbf{v}] = 0. \tag{9.92}$$

The potential vorticity conservation equation is obtained after expanding equation 9.92 and using the continiuity equation 9.89. Equation 9.92 is the flux form of equation 9.90, and shows that the area average of hq is conserved if the domain is closed.

The best configuration to solve the shallow water equations is that of the C-grid, figure 9.4; the vorticity, streamfunction, and potential vorticity are collocated. In terms of the discrete operators we have

$$\zeta = \delta_x v - \delta_y u, \quad q = \frac{\zeta + f}{\overline{\eta}^{xy}} \tag{9.93}$$

The finite difference discretization of the continuity equation on the C-grid takes the form:

$$\frac{\partial h}{\partial t} + \delta_x U + \delta_y V = 0 \tag{9.94}$$

where $U = \overline{h}^x u$ and $V = \overline{h}^y v$.

The purpose is to formulate FD expressions for the momentum advection terms compatible with the conservation of PV, equation 9.90. With this in mind, we start by averaging equation 9.94 with ^{xy}, to bring it to the q collocation points, and multiply the resulting equation by q to obtain:

$$\frac{\partial q\overline{h}^{xy}}{\partial t} + \delta_x\left(\overline{q}^x\overline{U}^{xy}\right) + \delta_y\left(\overline{q}^y\overline{V}^{xy}\right) = \overline{h}^{xy}\left[\frac{\partial q}{\partial t} + \frac{1}{\overline{h}^{xy}}\left(\overline{\overline{U}^{xy}\delta_x q}^x + \overline{\overline{V}^{xy}\delta_y q}^y\right)\right] \tag{9.95}$$

Equation 9.95 is a FD expression of the identity

$$\frac{\partial qh}{\partial t} = \nabla \cdot (qh\mathbf{v}) = h\frac{\mathrm{D}q}{\mathrm{D}t} \tag{9.96}$$

If the right hand side of 9.95 is a FD expression for the PV equation 9.90, the left hand side is a FD analogue to the vorticity equation 9.92. Carrying the steps backwards we have the following component forms for $-\mathbf{v} \times (\zeta + f)\mathbf{k}$:

$$\begin{array}{cc} \text{Continuum} & \text{Discrete} \\ \begin{pmatrix} -v(\zeta+f) \\ u(\zeta+f) \end{pmatrix} & \begin{pmatrix} -\overline{V}^{xy}\overline{q}^{y} \\ \overline{U}^{xy}\overline{q}^{x} \end{pmatrix} \end{array}$$

The remaining task is to find suitable forms for the $\nabla\frac{\mathbf{v}\cdot\mathbf{v}}{2}$. The choices available are either squaring the space-averaged velocity components, or averaging the squared velocity. The latter however leads to a straightforward FD analogue of the kinetic energy and is therefore preferred. This leads to the following PV-conserving momentum advection and Coriolis force operators:

$$\mathbf{v}\cdot\nabla u - fv = \frac{1}{2}\delta_x\left(\overline{u^2}^x + \overline{v^2}^y\right) - \overline{v}^{xy}\overline{q}^y \tag{9.97}$$

$$\mathbf{v}\cdot\nabla v + fu = \frac{1}{2}\delta_y\left(\overline{u^2}^x + \overline{v^2}^y\right) + \overline{u}^{xy}\overline{q}^x \tag{9.98}$$

It can be shown that the above operator also conserves potential enstrophy $hq^2/2$.

The derivation of schemes that conserve both PV and kinetic energy is very complex. Arakawa and Lamb Arakawa and Lamb (1981); Arakawa and Hsu (1981) did derive such a differencing scheme. Here we quote the final result:

$$\frac{1}{2}\delta_x\left(\overline{u^2}^x + \overline{v^2}^y\right) - \overline{\overline{V}^y\overline{q}^{xy}}^x - \frac{1}{48}\delta'_x\left((\delta'_y V)\delta'_x\delta'_y q\right) + \frac{1}{12}\delta'_x\left(\overline{U}^x\delta'_y\overline{q}^x\right) + \frac{1}{12}\overline{\delta'_x U\;\overline{\delta'_y\overline{q}^x}^x}^x \tag{9.99}$$

$$\frac{1}{2}\delta_y\left(\overline{u^2}^x + \overline{v^2}^y\right) - \overline{\overline{U}^x\overline{q}^{xy}}^y + \frac{1}{48}\delta'_y\left((\delta'_x U)\delta'_x\delta'_y q\right) - \frac{1}{12}\delta'_y\left(\overline{V}^y\delta'_x\overline{q}^y\right) - \frac{1}{12}\overline{\delta'_y V\;\overline{\delta'_x\overline{q}^y}^y}^y \tag{9.100}$$

where δ' is the discrete differential operator without division by grid distance.

Chapter 10

Special Advection Schemes

10.1 Introduction

This chapter deals with specialized advection schemes designed to handle problems where in addition to consistency, stability and conservation, additional constraints on the solution must be satisfied. For example, biological or chemical concentration must be non-negative for phsical reason; however, numerical errors are capable of generating negative values which are simply wrong and not amenable to physical interpretation. These negative values can impact the solution adversely, particularly if there is a feed back loop that exacerbate these spurious values by increasing their unphysical magnitude. An example of a feed back loop is a reaction term valid for only positive values leading to moderate growth or decay of the quantity in question; whereas negative values lead to unstable exponential growth. Another example is the equation of state in ocean models which intimately ties salt, temperature, and density. This equation is empirical in nature and is valid for specific ranges of temperature, salt, and density; and the results of out of range inputs to this equation are unpredictable and lead quickly to instabilities in the simulation.

The primary culprit in these numerical artifacts is the advection operator as it is the primary means by which tracers are moved around in a fluid environment. Molecular diffusion is usually too weak to account for much of the transport, and what passes for turbulent diffusion has its roots in "vigorous" advection in straining flow fields. Advection transports a tracer from one place to another without change of shape, and as such preserves the original extrema (maxima and minima) of the field for long times (in the absence of other physical mechanism). Problems occur when the gradient are too steep to be resolved by the underlying computational grid. Examples include true discontinuities, such as shock waves or tidal bore, or pseudi-discontinuities such as narrow temperature or salt fronts that are too narrow to be resolved on the grid, (a few hundered meters whereas the computational grid is of the order of kilometers).

A number of special advection schemes were devised to address some or all of these issues. They are known generically as Total Variation Diminishing (TVD) schemes. They occupy a prominent place in the study and numerical solution of hyperbolic equations like the Euler equations of gas dynamics or the shallow water equations. Here we confine ourselves to the pure advection equation, a scalar hyperbolic equation.

10.2 Monotone Schemes

The properties of the pure advection operator to preserve the original extrema of the advected field is referred to as the monotonicity property. Consider an initially discretized initial condition of the form $T_j^0 \geq T_{j+1}^0$, then a scheme is called monotone if

$$T_j^n \geq T_{j+1}^n \tag{10.1}$$

for all j and n. A general advection scheme can be written in the form:

$$T_j^{n+1} = \sum_{k=-p}^{q} \alpha_k T_{j+k}^n \tag{10.2}$$

where the α_k are coefficients that depend on the specific scheme used. A linear scheme is one where the coefficients α_k are independent of the solution T_j. For a scheme to be monotone with respect to the T_k^n, we need the condition

$$\frac{\partial T_j^{n+1}}{\partial T_{j+k}^n} \geq 0 \tag{10.3}$$

Godunov has shown that the only *linear* monotonic scheme is the first order (upstream) donor cell scheme. All high-order linear schemes are not monotonic and will permit spurious extrema to be generated. High-order schemes must be *nonlinear* in order to preserve monotonicity.

10.3 Flux Corrected Transport (FCT)

The FCT algorithm was originally proposed by Boris and Book Boris and Book (1973, 1975, 1976) and later modified and generalized to multidimensions by Zalesack Zalesak (1979). Here we present the Zalesak version as it is the more common one and flexible one. We will first consider the scheme in one-dimension before we consider its two-dimensional extension.

10.3.1 One-Dimensional

Consider the advection of a tracer in one-dimension written in conservation form:

$$T_t + (uT)_x = 0 \tag{10.4}$$

subject to appropriate initial and boundary conditions. The spatially integrated form of this equation lead to the following:

$$\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} T_t \, \mathrm{d}x + f|_{x_{j+\frac{1}{2}}} - f|_{x_{j-\frac{1}{2}}} = 0 \tag{10.5}$$

where $f|_{x_{j+\frac{1}{2}}} = [uT]_{x_{j-\frac{1}{2}}}$ is the flux out of the cell j. This equation is nothing but the restatement of the partial differential equation as the rate at which the budget of T in cell j increases according to the advective fluxes in and out of the cell. As a matter of fact the above equation can be reintrepeted as a finite volume method if the integral is replaced by $\frac{\partial \overline{T}_j}{\partial t}\Delta x$ where $\overline{T}_j$ refers to the average of T in cell j whose size is Δx. We now have:

$$\frac{\partial \overline{T}_j}{\partial t} + \frac{f_{j+\frac{1}{2}} - f_{j-\frac{1}{2}}}{\Delta x} = 0 \tag{10.6}$$

If the analytical flux is now replaced by a numerical flux, F, we can generate a family of discrete schemes. If we choose an upstream biased scheme where the value within each cell is considered constant, i.e. $F_{j+\frac{1}{2}} = u_{j+\frac{1}{2}}T_j$ for $u_{j+\frac{1}{2}} > 0$ and $F_{j+\frac{1}{2}} = u_{j+\frac{1}{2}}T_{j+1}$ for $u_{j+\frac{1}{2}} < 0$ we get the donor cell scheme. Note that the two cases above can be re-written (and programmed) as:

$$F_{j+\frac{1}{2}} = \frac{u_{j+\frac{1}{2}} + |u_{j+\frac{1}{2}}|}{2}T_j + \frac{u_{j+\frac{1}{2}} - |u_{j+\frac{1}{2}}|}{2}T_{j+1} \tag{10.7}$$

The scheme will be monotone if we were to advance in time stably using a forward Euler method. If on the other hand we choose to approximate T_j at the cell edge as the average of the two cells:

$$F_{j+\frac{1}{2}} = u_{j+\frac{1}{2}}\frac{T_j + T_{j+1}}{2} \tag{10.8}$$

we obtained the second order centered in space scheme. Presumably the second order scheme will provide a more accurate solution in those regions where the advected profile is *smooth* whereas it will create spurious oscillations in regions where the solution is "rough".

The idea behind the flux corrected transport algorithm is to use a combination of the higher order flux and the lower order flux to prevent the generation of new extrema. The algorithm can be summarized as follows:

1. compute low order fluxes $F^L_{j+\frac{1}{2}}$.

2. compute high order fluxes $F^H_{j+\frac{1}{2}}$, e.g. second order interpolation of T to cell edges or higher.

3. Define the anti-diffusive flux $A_{j+\frac{1}{2}} = F^H_{j+\frac{1}{2}} - F^L_{j+\frac{1}{2}}$. This flux is dubbed anti-diffuse because the higher order fluxes attempt to correct the over diffusive effects of the low order fluxes.

4. Update the solution using the low order fluxes to obtain a first order diffused but monotonic approximation:

$$T^d_j = T^n_j - \frac{F^L_{j+\frac{1}{2}} - F^L_{j-\frac{1}{2}}}{\Delta x}\Delta t \tag{10.9}$$

5. Limit the anti-diffusive flux so that the corrected solution will be free of extrema not found in T^n_j or T^d_j. The limiting is effected through a factor: $A^c_{j+\frac{1}{2}} = C_{j+\frac{1}{2}}A_{j+\frac{1}{2}}$, where $0 \leq C_{j+\frac{1}{2}} \leq 1$.

6. Apply the anti-diffusive flux to get the corrected solution

$$T_j^{n+1} = T_j^d - \frac{A^c_{j+\frac{1}{2}} - A^c_{j-\frac{1}{2}}}{\Delta x}\Delta t \tag{10.10}$$

Notice that for $C = 0$ the anti-diffusive fluxes are not applied, and we end up with $T_j^{n+1} = T_j^d$; while for $C = 1$, they are applied at full strength.

10.3.2 One-Dimensional Flux Correction Limiter

In order to elucidate the role of the limiter we expand the last expression in terms of the high and low order fluxes to obtain:

$$T_j^{n+1} = T_j^n - \frac{\left[C_{j+\frac{1}{2}}F^H_{j+\frac{1}{2}} + \left(1 - C_{j+\frac{1}{2}}\right)F^L_{j+\frac{1}{2}}\right] - \left[C_{j-\frac{1}{2}}F^H_{j-\frac{1}{2}} + \left(1 - C_{j-\frac{1}{2}}\right)F^L_{j-\frac{1}{2}}\right]}{\Delta x}\Delta t \tag{10.11}$$

The term under the bracket can thus be interpreted as a weighed average of the low and high order flux; and the weights depend on the *local* smoothness of the solution. Thus for a rough neighborhood we should choose $C \to 0$ to avoid oscillations, while for a smooth neighborhood $C = 1$ to improve accuracy. As one can imagine the power and versatility of FCT lies in the algorithm that prescribe the limiter. Here we prescribe the Zalesak limiter.

1. Optional step designed to eliminate correction near extrema. Set $A_{j+\frac{1}{2}} = 0$ if:

$$A_{j+\frac{1}{2}}\left(T^d_{j+1} - T^d_j\right) < 0 \text{ and } \begin{cases} A_{j+\frac{1}{2}}\left(T^d_{j+2} - T^d_{j+1}\right) < 0 \\ \text{or} \\ A_{j+\frac{1}{2}}\left(T^d_j - T^d_{j-1}\right) < 0 \end{cases} \tag{10.12}$$

2. Evaluate the range of permissible values for T_j^{n+1}:

$$\begin{cases} T_j^{\max} = \max\left(T^n_{j-1}, T^n_j, T^n_{j+1}, T^d_{j-1}, T^d_j, T^d_{j+1}\right) \\ T_j^{\min} = \min\left(T^n_{j-1}, T^n_j, T^n_{j+1}, T^d_{j-1}, T^d_j, T^d_{j+1}\right) \end{cases} \tag{10.13}$$

3. Compute the anti-diffusive fluxes P_j^+ going into cell j:

$$P_j^+ = \max\left(0, A_{j-\frac{1}{2}}\right) - \min\left(0, A_{j+\frac{1}{2}}\right) \tag{10.14}$$

These incoming fluxes will increase T_j^{n+1}.

4. Compute the maximum permissible incoming flux that will keep $T_j^{n+1} \leq T_j^{\max}$. From the corrective step in equation 10.10 this given by

$$Q_j^+ = \left(T_j^{\max} - T_j^d\right)\frac{\Delta x}{\Delta t} \tag{10.15}$$

5. Compute limiter required so that the extrema in cell j are respected:

$$R_j^+ = \begin{cases} \min\left(1, \dfrac{Q_j^+}{P_j^+}\right) & \text{if} \quad P_j^+ > 0 \\ 0 & \text{if} \quad P_j^+ = 0 \end{cases} \tag{10.16}$$

6. Steps 3, 4 and 5 must be repeated so as to ensure that the lower bound on the solution $T_j^{\min} \le T_j^{n+1}$. So now we define the anti-diffusive fluxes away from cell j:

$$P_j^- = \max\left(0, A_{j+\frac{1}{2}}\right) - \min\left(0, A_{j-\frac{1}{2}}\right) \tag{10.17}$$

$$Q_j^- = \left(T_j^d - T_j^{\min}\right)\frac{\Delta x}{\Delta t} \tag{10.18}$$

$$R_j^- = \begin{cases} \min\left(1, \dfrac{Q_j^-}{P_j^-}\right) & \text{if} \quad P_j^- > 0 \\ 0 & \text{if} \quad P_j^- = 0 \end{cases} \tag{10.19}$$

7. We now choose the limiting factors so as enforce the extrema constraints simultaneously on adjacent cells.

$$C_{j+\frac{1}{2}} = \begin{cases} \min\left(R_{j+1}^+, R_j^-\right) & \text{if } A_{j+\frac{1}{2}} > 0 \\ \min\left(R_j^+, R_{j+1}^-\right) & \text{if } A_{j+\frac{1}{2}} < 0 \end{cases} \tag{10.20}$$

10.3.3 Properties of FCT

Figure 10.1 shows the advection of a function using a centered differencing formula of order 8 using 50, 100, 200, 400 and 2000 points around a periodic domain of length 20. The red and blue curves show the analytical and the numerical solution, respectively. The time-stepping is based on an RK3 scheme with a fixed time step of $\Delta t = 10^{-3}$. The function is composed of multiple shapes to illustrate the strength and weaknesses of different numerical discretization on various solution profiles with different levels of discontinuities: square and triangular waves, a truncated inverted parabola with a smooth maximum, and an infinitely smooth narrow Gaussian. The discontinuities in the profile are challenging for high-order method and this is reflected in the solutions obtained in 10.1 where Gibbs oscillations pollute the entire solution and at all resolutions. One can anticipate that in the limit of infinite resolution the amplitude of these Gibbs oscillations will reach a finite limit independent of the grid size. The top solution with 50 points is severely under-resolved (even the analytical solution does not look smooth since it is drawn on the numerical grid). The situation improves dramatically with increased resolution for the smooth profiles where the smooth peaks are now well-represented using $\Delta x = 20/200 = 0.1$. The Gibbs oscillations seen riding on the smooth peaks are caused by the spurious dispersion of grid scale noise. The resolution increase does not pay off for the square wave where the numerical solution exhibit noise at all resolutions.

In contrast to the high-order solution, figure 10.2 shows the results of the same computations using a donor-cell scheme to compute the fluxes. As anticipated from our

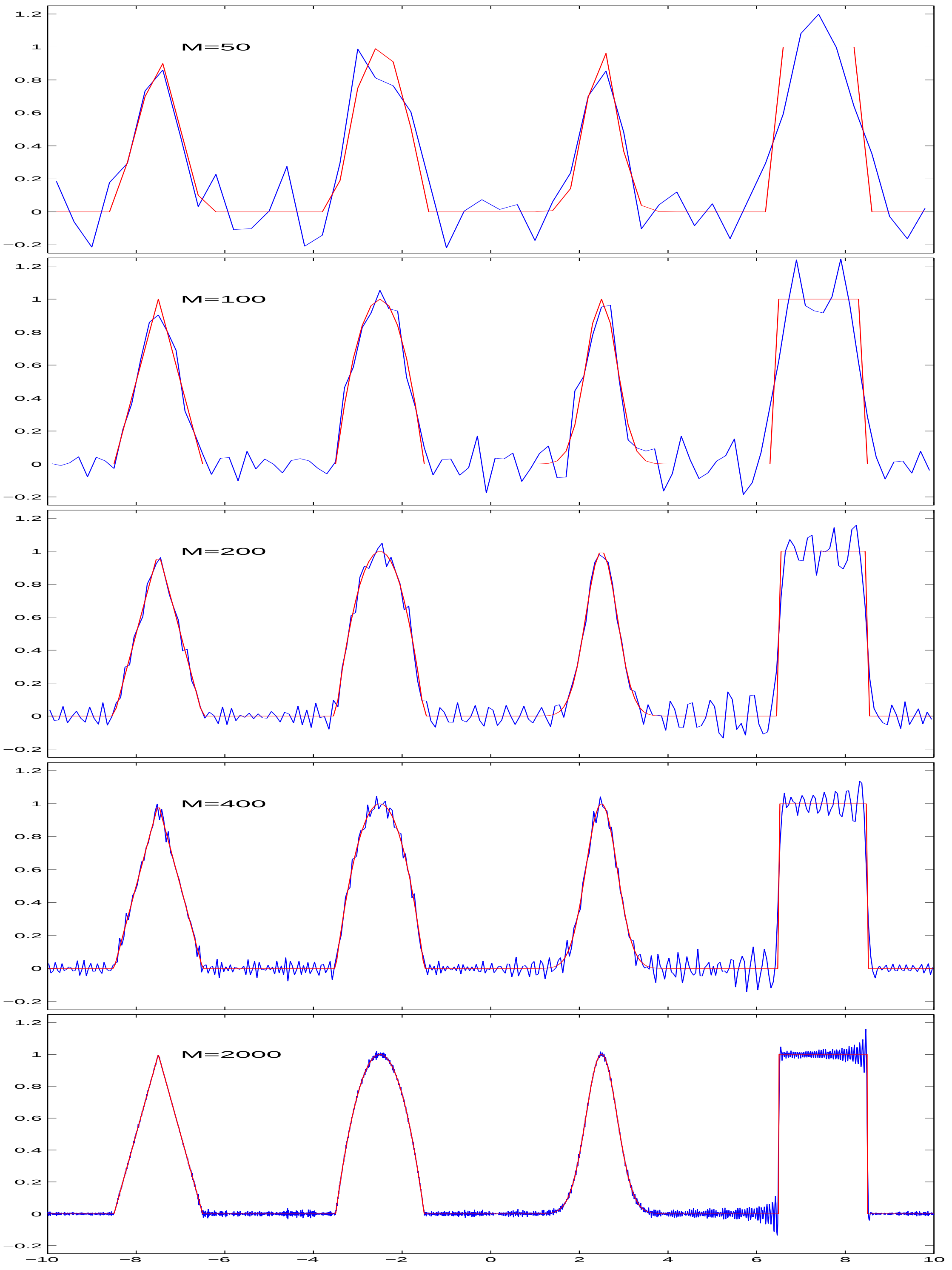


Figure 10.1: Uncorrected Centered Difference schemes of 8th order with RK3, $\Delta t = 10^{-3}$.

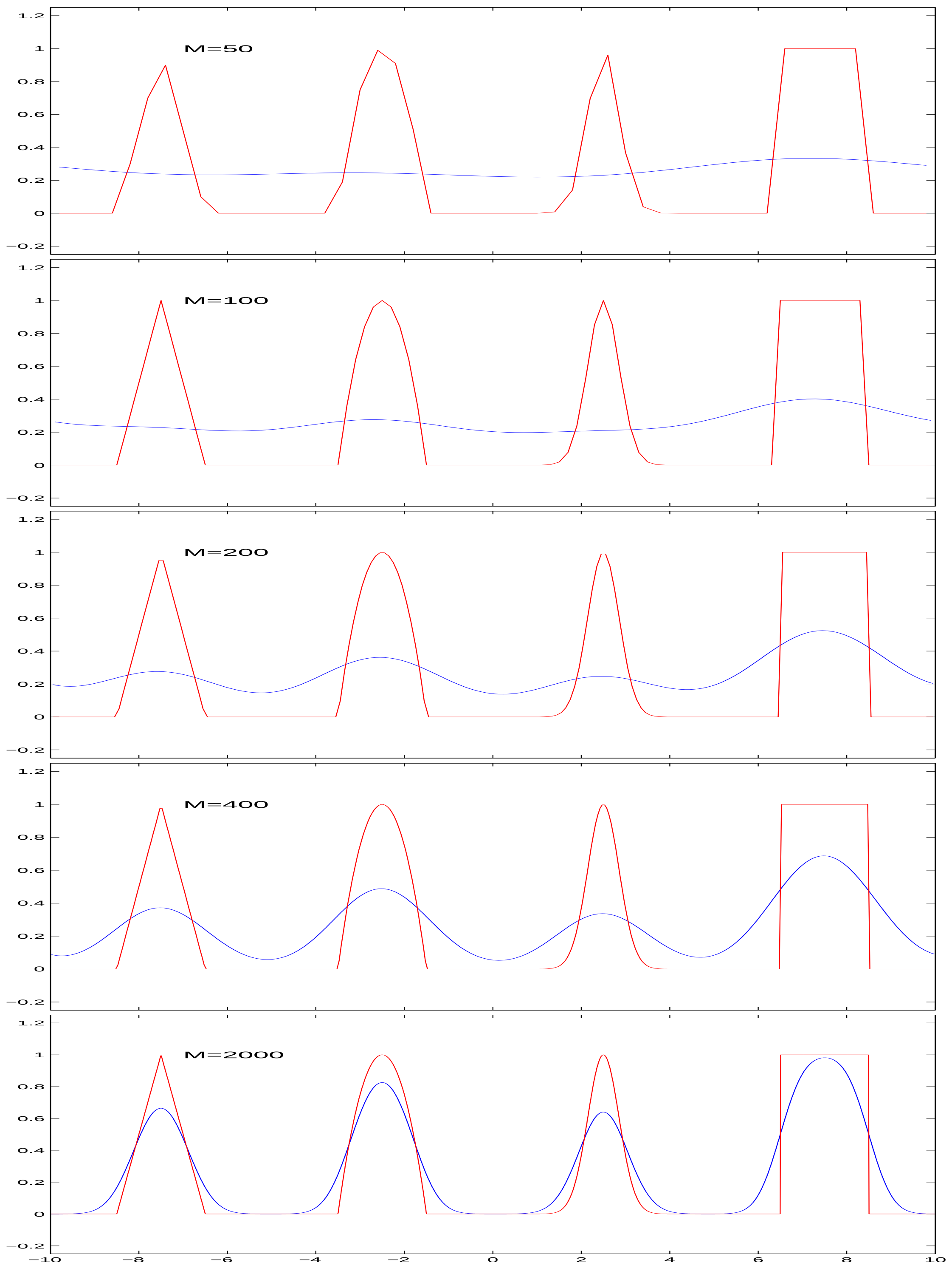


Figure 10.2: Uncorrected Centered Difference schemes of 1st order with RK1, $\Delta t = 10^{-3}$.

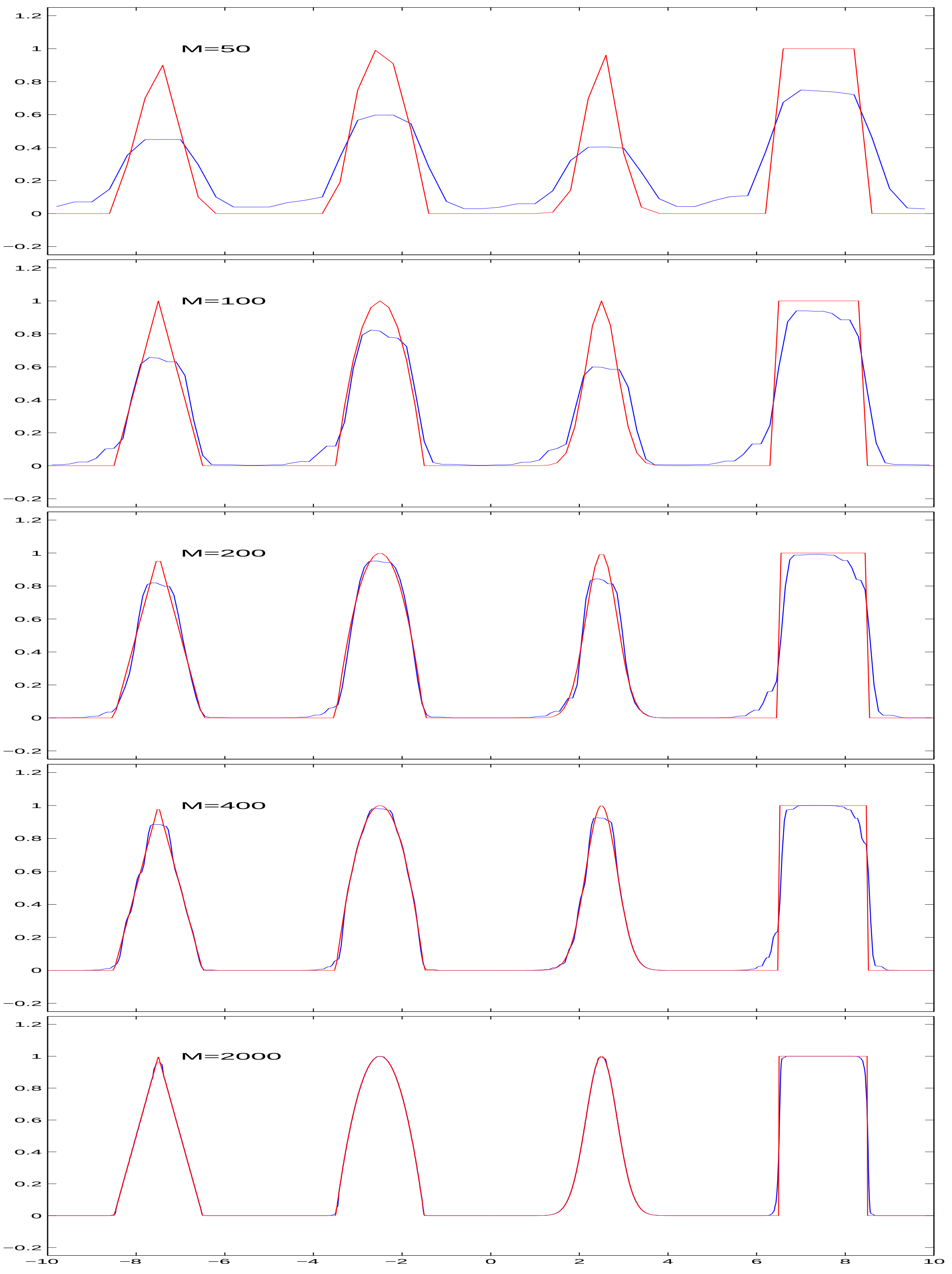


Figure 10.3: FCT-Centered Difference schemes of 8th order with RK3, $\Delta t = 10^{-3}$.

theoretical discussion of the donor-cell scheme the errors are primarily dissipative. At coarse resolution the signals in the solution have almost disappeared and the solution has been smeared excessively. Even at the highest resolution used here the smooth peaks have lost much of their amplitudes while the discontinuities (of various strengthes) have also been smeared. On the up-side the donor-cell scheme has delivered solutions that are oscillations-free and that respect the TVD properties of the continuous PDE.

The FCT approach, as well as other limiting methods, aim to achieve a happy medium between diffusive but oscillation-free and high-order but oscillatory. Figure 10.3 illustrate FCT's benefits (and drawbacks). First the oscillations have been eliminate at all resolutions. Second, at the coarseest resolution, there is a significant loss of peak accompanied by the so-called terracing effect where the limiter tends to flatten smooth peaks, and to introduce intermediate "step" in the solution where previously there was none. This effect continues well-into the well resolved regime of $\Delta x = 20/400$ where the smooth Gaussian peak has still not recovered its full amplitude and is experiencing a continuiing terracing effect. This contrasts with the uncorrected solution where the smooth peak was recoved with a more modest resolution of $\Delta x = 20/200$. This is a testament to an overactive FCT that cannot distinguish between smooth peaks and discontinuities. As a result smooth peaks that ought not to be limited are flattened out. It is possible to mitigate an overzealous limiter by appending a "discriminator" that can turn off the limiting near smooth extrema. In general the discriminator is built upon checking whether the second derivative of the solution is one-signed in neighborhoods where the solution's slope changes sign. Furthermore, the terracing effect can be eliminated by introducing a small amount of scale selective dissipation flux (essentially a hyperviscosity). For further details check out Shchepetkin and McWilliams (1998) and Zalesak (2005).

The FCT procedure has advantages and disadvantages. The primary benefit of the procedure is that it is a practical procedure to prevent the generation of spurious extrema. It is also flexible in defining the high order fluxes and the extrema of the fields. Most importantly the algorithm can be extended to multiple dimensions in a relatively straightforward manners. The disadvantages is that the procedure is costly in CPU compared to unlimited method, but hey nothing comes free.

10.3.4 Two-Dimensional FCT

In two dimensions the advection equation takes the form $T_t + f_x + g_y = 0$, where $f = uT$ and $g = vT$. The FCT algorithm takes the same form as before after taking account of the extra flux G and extra spatial dimension. We thus have the low order solution:

$$T^d_{j,k} = T^n_{j,k} - \Delta t \frac{F^L_{j+\frac{1}{2},k} - F^L_{j-\frac{1}{2},k}}{\Delta x} - \Delta t \frac{G^L_{j,k+\frac{1}{2}} - G^L_{j,k-\frac{1}{2}}}{\Delta y} \tag{10.21}$$

and the following anti-diffusive fluxes:

$$A_{j+\frac{1}{2},k} = F^H_{j+\frac{1}{2},k} - F^L_{j+\frac{1}{2},k}, \quad A_{j,k+\frac{1}{2}} = G^H_{j,k+\frac{1}{2}} - G^L_{j,k+\frac{1}{2}} \tag{10.22}$$

The extrema of the solution are defined in two passes over the data. First we set $T^{\max}_{j,k} = \max(T^n_{j,k}, T^d_{j,k})$. The final permissible values are determined by computing the extrema of the previous fields in the neighboring cells:

$$\begin{cases} T^{\max}_{j,k} = \max\left(T^{\max}_{j,k}, T^{\max}_{j\pm1,k}, T^{\max}_{j,k\pm1}\right) \\ T^{\min}_{j,k} = \min\left(T^{\min}_{j,k}, T^{\min}_{j\pm1,k}, T^{\min}_{j,k\pm1}\right) \end{cases} \tag{10.23}$$

Finally, the incoming and outgoing fluxes in cell (j,k) are given by:

$$P^{+}_{j,k} = \max\left(0, A_{j-\frac{1}{2},k}\right) - \min\left(0, A_{j+\frac{1}{2},k}\right) + \max\left(0, A_{j,k-\frac{1}{2}}\right) - \min\left(0, A_{j,k+\frac{1}{2}}\right) \tag{10.24}$$

$$P^{-}_{j,k} = \max\left(0, A_{j+\frac{1}{2},k}\right) - \min\left(0, A_{j-\frac{1}{2},k}\right) + \max\left(0, A_{j,k+\frac{1}{2}}\right) - \min\left(0, A_{j,k-\frac{1}{2}}\right) \tag{10.25}$$

10.3.5 Time-Differencing with FCT

A last practical issue with the implementation of FCT is the choice of time-differencing for the combined low and high order scheme. In order to preserve the monotone property the low order scheme must rely on a first order Forward Euler scheme in time. For the high order flux it is desirable to increase the order of the time-differencing to match that of the spatial differencing, and to increase the time-accuracy at least in smooth regions. There is an additional wrinkle to this dilemna in that whereas Forward Euler in time is stable for the donor-cell scheme, it is unconditionally unstable for centered in space spatial differences.

The resolution of this dilemna can be addressed in several ways. One approach is to use a Runge-Kutta type approach and to use the low and high order fluxes at each of the substages of the Runge Kutta scheme. For the RK scheme to be stable to both spatial differences; we need at least a third order scheme (their stability region includes portions of the imaginary axis and the left hand complex planes). The drawback of such an approach is an increase in CPU time since multiple evaluations of the right hand sides is required. The CPU cost is exacerbated if the FCT limiter is applied at each of the sub-steps.

Another approach consists of using a multi-level methods like the leap-frog trapezoidal scheme. The low flux is first updated using the solution at time n. The high order fluxes at time n are obtained with a traditional leap-frog trapezoidal step that does not involve the low order solution (i.e. no limiter is applied): first the leap-frog step is applied:

$$\hat{T}^{n+1} = T^{n-1} - \Delta t \frac{F^H_{j+\frac{1}{2}}(T^n) - F^H_{j-\frac{1}{2}}(T^n)}{\Delta x} \tag{10.26}$$

A second order estimate of the function values at the mid-time level is calculated:

$$T^{n+\frac{1}{2}} = \frac{T^n + \hat{T}^{n+1}}{2} \tag{10.27}$$

and then used to compute the high order fluxes

$$F^H_{j+\frac{1}{2}} = F^H_{j+\frac{1}{2}}\left(T^{n+\frac{1}{2}}\right). \tag{10.28}$$

It is these last high order fluxes that are limited using the FCT algorithm.

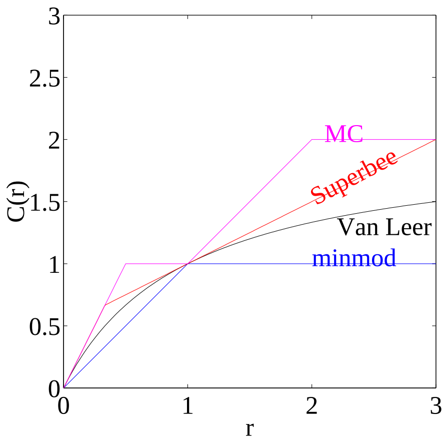


Figure 10.4: Graph of the different limiters as a function of the slope ratio r.

10.4 Slope/Flux Limiter Methods

The slope/flux limiter class of methods uses a similar approach to the flux correction method, in that a low order and a high order flux is used to eliminate spurious oscillations. The slope-limiter schemes however do not involve the computations of the temporary diffused value. The limiting is instead applied directly to the flux based on the values of the solution's gradients. That the final flux used in the scheme takes the form:

$$F_{j+\frac{1}{2}} = F^L_{j+\frac{1}{2}} + C_{j+\frac{1}{2}} \left(F^H_{j+\frac{1}{2}} - F^L_{j+\frac{1}{2}} \right) \tag{10.29}$$

where the limiting factor $C = C(r)$ is a function of the slope ratio in neighboring cells:

$$r_{j+\frac{1}{2}} = \frac{T_j - T_{j-1}}{T_{j+1} - T_j}. \tag{10.30}$$

For slowly varying smooth function the slope ratio is close to 1; the slopes change sign near an extremum and the ratio is negative. A family of method can be generated based entirely on the choice of limiting function; here we list a few of the possible choices:

$$\begin{array}{ll} \text{MinMod:} & C(r) = \max(0, \min(1, r)) \\ \text{Superbee:} & C(r) = \max(0, \min(1, 2r), \min(2, r)) \\ \text{Van Leer:} & C(r) = \frac{r+|r|}{1+|r|} \\ \text{MC:} & C(r) = \max(0, \min(2r, \frac{1+2r}{2}, 2) \end{array} \tag{10.31}$$

The graph of these limiters is shown in figure 10.4. The different functions have a number of common features. All limiters set $C = 0$ near extrema ($r \leq 0$). They all asymptote to 2, save for the minmod limiter which asymptotes to 1, when the function changes rapidly ($r \gg 1$). The minmod limiter is the most stringent of the limiters and prevents the solution gradient from changing quickly in neighboring cells; this limiter is known as being diffusive. The other limiters are more lenient, the MC one being the most lenient, and permit the gradients in neighboring cells to be twice as large as the one in the neighboring cell. The Van Leer limiter is the smoothest of the limiters and asymptotes to 2 for $r \rightarrow \infty$.

10.5 MPDATA

The Multidimensional Positive Definite Advection Transport Algorithm (MPDATA) was presented by Smolarkiewicz (1983) as an algorithm to preserve the positivity of the field throughout the simulation. The motivation behind his work is that chemical tracers must remain positive. Non-oscillatory schemes like FCT are positive definite but are deemed too expensive, particularly since oscillations are tolerable as long as they did not involve negative values. MPDATA is built on the monotone donor cell scheme and on its modified equation. The latter is used to determine the diffusive errors in the scheme and to correct for it near the zero values of the field. The scheme is presented here in its one-dimensional form for simplicity. The modified equation for the donor cell scheme where the fluxes are defined as in equaiton 10.7 is:

$$\frac{\partial T}{\partial t} + \frac{\partial uT}{\partial x} = \frac{\partial}{\partial x}\left(\kappa \frac{\partial T}{\partial x}\right) + O(\Delta x^2) \tag{10.32}$$

where κ is the numerical diffusion generated by the donor cell scheme:

$$\kappa = \frac{|u|\Delta x - u^2 \Delta t}{2} \tag{10.33}$$

The donor cell scheme will produce a first etimate of the field which is guranteed to be non-negative if the initial field is initially non-negative. This estimate however, is too diffused; and must be corrected to eliminate these first order errors. MPDATA data achieves the correction by casting the second order derivatives in the modified equation 10.32 as another transport step with a pseudo-velocity $\tilde{u}$:

$$\frac{\partial T}{\partial t} = -\frac{\partial \tilde{u} T}{\partial x}, \quad \tilde{u} = \begin{cases} \dfrac{\kappa}{T}\dfrac{\partial T}{\partial x} & T > 0 \\ 0 & T = 0 \end{cases} \tag{10.34}$$

and re-using the donor cell scheme to discretize it. The velocity $\tilde{u}$ plays the role of an anti-diffusion velocity that tries to compensate for the diffusive error in the first step. The correction step takes the form:

$$\begin{aligned}
\tilde{u}_{j+\frac{1}{2}} &= \left(\left|u_{j+\frac{1}{2}}\right| \Delta x - u^2_{j+\frac{1}{2}} \Delta t\right) \frac{T^d_{j+1} - T^d_j}{(T^d_{j+1} + T^d_j + \epsilon)\Delta x} && (10.35) \\
\tilde{F}_{j+\frac{1}{2}} &= \frac{\tilde{u}_{j+\frac{1}{2}} + \left|\tilde{u}_{j+\frac{1}{2}}\right|}{2} T^d_j + \frac{\tilde{u}_{j+\frac{1}{2}} - \left|\tilde{u}_{j+\frac{1}{2}}\right|}{2} T^d_{j+1} && (10.36) \\
T^{n+1}_j &= \tilde{T}_j - \frac{\tilde{F}_{j+\frac{1}{2}} - \tilde{F}_{j-\frac{1}{2}}}{\Delta x} \Delta t && (10.37)
\end{aligned}$$

where T^d_j is the diffused solution from the donor-cell step, and ϵ is a small positive number, e.g. 10^-15, meant to prevent the denominator from vanishing when $T^d_j =$

$T^d_{j+1} = 0$. The second donor cell step is stable provided the original one is too; and hence the correction does not penalize the stability of the scheme. The procedure to derive the two-dimensional version of the scheme is similar; the major difficulty is in deriving the modified equation and the corresponding anti-diffusion velocity. It turns out that the x-component of the anti-diffusion velocity remains the same while the y-components takes a similar form with u replaced by v, and Δx by Δy.

10.6 WENO schemes in vertical

We explore the application of WENO methodology to compute in the vertical. The WENO methodology is based on a reconstruction of function values from cells averages using different stencils, and on combining the different estimates so as to maximime accuracy while minimizing the impact of nonsmooth stencils. We briefly describe the steps of a WENO calculation below, the details can be found in Shu (1998); Jiang and Shu (1996). Note that the diffusive term requires the calculation of the derivative of the function. This can be also done accurately with a WENO scheme after the reconstruction of T; the caveat for high order accuracy is that the grid spacing Δz must vary smoothly. We first take up the reconstruction step and dwell on differentiation later. The question is of course always how much should we pay for an accurate calculation of the vertical diffusion term.

10.6.1 Function reconstruction

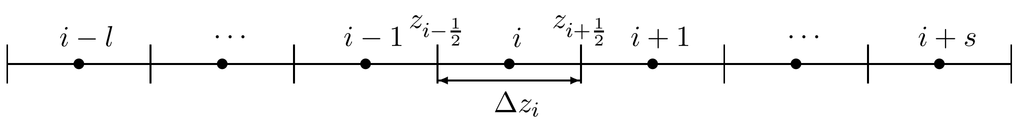


Figure 10.5: Sketch of the stencil $S(i;k,l)$. This stencil is associated with cell i, has left shift l, and contains $k = l + s + 1$ cells.

We first focus on the issue of computing function values from cell averages. We divide the vertical into a number of finite volumes which we also refer to as cells, and we define cells, cell centers and cells sizes by:

$$I_i = \left[z_{i-\frac{1}{2}}, z_{i+\frac{1}{2}}\right] \tag{10.38}$$

$$z_i = \frac{z_{i-\frac{1}{2}} + z_{i+\frac{1}{2}}}{2} \tag{10.39}$$

$$\Delta z_i = z_{i+\frac{1}{2}} - z_{i-\frac{1}{2}} \tag{10.40}$$

The reconstruction problem can be stated as follows: Given the cell averages of a function $T(z)$:

$$\overline{T}^z_i = \frac{1}{\Delta z_i}\int_{z_{i-\frac{1}{2}}}^{z_{i+\frac{1}{2}}} v(z')\,\mathrm{d}z', i = 1,2,\ldots,N \tag{10.41}$$

find a polynomial $p_i(z)$, of degree at most $k-1$, for each cell i, such that it is a k-th order accurate approximation to the function $T(z)$ inside I_i:

$$p_i(z) = T(z) + O(\Delta z^k), \quad z \in I_i, i = 1, 2, \dots, N \tag{10.42}$$

The polynomial $p_i(z)$ interpolates the function within cells. It also provides for a *discontinuous* interpolation at cell boundaries since a cell boundary is shared by more then one cell; we thus write:

$$T^+_{i-\frac{1}{2}} = p_i(z_{i-\frac{1}{2}}), \quad T^-_{i+\frac{1}{2}} = p_i(z_{i+\frac{1}{2}}) \tag{10.43}$$

Given the cell I_i and the order of accuracy k, we first choose a stencil, $S(i; k, l)$, based on I_i, l cells to the left of I_i and s cells to the right of I_i with $l + s + 1 = k$. $S(i)$ consists of the cells:

$$S(i) = \{I_{i-l}, I_{i-l+1}, \dots, I_{i+s}\} \tag{10.44}$$

There is a unique polynomial $p(z)$ of degree $k - 1 = l + s$, whose cell average in each of the cells in $S(i)$ agrees with that of $T(z)$:

$$\overline{T}^z_j = \frac{1}{\Delta z_j}\int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} p(z')\ \mathrm{d}z', \quad j = i-l, \dots, i+s. \tag{10.45}$$

The polynomial in question is nothing but the derivative of the Lagrangian interpolant of the function $T(z)$ at the cell boundaries. To see this, we look at the primitive function of $T(z)$:

$$\mathcal{T}(z) = \int_{-\infty}^{z} T(z')\ \mathrm{d}z', \tag{10.46}$$

where the choice of lower integration limit is immaterial. The function $\mathcal{T}(z)$ at cell edges can be expressed in terms of the cell averages:

$$\mathcal{T}(z_{i+\frac{1}{2}}) = \sum_{j=-\infty}^{i} \int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} T(z')\ \mathrm{d}z' = \sum_{j=-\infty}^{i} \overline{T}^z_j \Delta z_j \tag{10.47}$$

Thus, the cell averages define the primitive function at the cell boundaries. If we denote the unique polynomial of degree at most k which interpolates $\mathcal{T}$ at the $k+1$ points: $z_{i-l-\frac{1}{2}}, \dots, z_{i+s+\frac{1}{2}}$, by $P(z)$, and denote its derivative by $p(z)$, it is easy to verify that:

$$\begin{aligned}
\frac{1}{\Delta z_j}\int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} p(z')\ \mathrm{d}z' &= \frac{1}{\Delta z_j}\int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} P'(z')\ \mathrm{d}z' && (10.48)\\
&= \frac{P(z_{j+\frac{1}{2}}) - P(z_{j-\frac{1}{2}})}{\Delta z_j} && (10.49)\\
&= \frac{\mathcal{T}(z_{j+\frac{1}{2}}) - \mathcal{T}(z_{j-\frac{1}{2}})}{\Delta z_j} && (10.50)\\
&= \frac{1}{\Delta z_j}\int_{z_{j-\frac{1}{2}}}^{z_{j+\frac{1}{2}}} T(z')\ \mathrm{d}z' && (10.51)\\
&= \overline{T}^z_j, \quad j = i-l, \dots, i+s && (10.52)
\end{aligned}$$

This implies that $p(z)$ is the desired polynomial. Standard approximation theory tells us that $P'(z) = T'(z) + O(\Delta z^k), z \in I_i$, which is the accuracy requirement.

The construction of the polynomial $p(z)$ is now straightforward. We can start with the Lagrange intepolants on the $k+1$ cell boundary and differentiate with respect to z to obtain:

$$p(z) = \sum_{m=0}^{k} \sum_{j=0}^{m-1} \overline{T}^z_{i-l+j} \Delta z_{i-l+j} \left(\frac{\displaystyle\sum_{\substack{n=0\\ n\neq m}}^{k} \prod_{\substack{q=0\\ q\neq m,n}}^{k} \left(z - z_{i-l+q-\frac{1}{2}}\right)}{\displaystyle\prod_{\substack{n=0\\ n\neq m}}^{k} \left(z_{i-l+m-\frac{1}{2}} - z_{i-l+n-\frac{1}{2}}\right)} \right) \tag{10.53}$$

The order of the outer sums can be exchanged to obtain an alternative form which maybe computationally more practical:

$$p(z) = \sum_{j=0}^{k-1} C_{lj}(z) \overline{T}^z_{i-l+j} \tag{10.54}$$

where $C_{lj}(z)$ is given by:

$$C_{lj}(z) = \Delta z_{i-l+j} \left(\sum_{m=j+1}^{k} \frac{\displaystyle\sum_{\substack{n=0\\ n\neq m}}^{k} \prod_{\substack{q=0\\ q\neq m,n}}^{k} \left(z - z_{i-l+q-\frac{1}{2}}\right)}{\displaystyle\prod_{\substack{n=0\\ n\neq m}}^{k} \left(z_{i-l+m-\frac{1}{2}} - z_{i-l+n-\frac{1}{2}}\right)} \right) \tag{10.55}$$

The coefficient C_{lj} need not be computed at each time step if the computational grid is fixed, instead they can be precomputed and stored to save CPU time. The expression for the C_{lj} simplifies (because many terms vanish) when the point z coincide with a cell edge and/or when the grid is equally spaced ($\Delta z_j = \Delta z, \forall j$).

ENO reconstruction

The accuracy estimate holds only if the function is smooth inside the entire stencil $S(i; k, l)$ used in the interpolation. If the function is not smooth Gibbs oscillations appear. The idea behind ENO reconstruction is to vary the stencil $S(i; k, l)$, by changing the left shift l, so as to choose a discontinuity-free stencil; this choice of $S(i; k, l)$ is called an "adaptive stencil". A smoothness criterion is needed to choose the smoothest stencil, and ENO uses Newton divided differences. The stencil with the smoothest Newton divided difference is chosen.

ENO properties:

1. The accuracy condition is valid for any cell which does not contain a discontinuity.

2. $P_i(z)$ is monotone in any cell I_i which *does* contain a discontinuity.

3. The reconstruction is Total Variation Bounded (TVB as opposed to TVD), that is there is a function $Q(z)$ satisfying $Q(z) = P_i(z) + O(\Delta z_i^{k+1}), z \in I_i$, such that $TV(Q) \leq TV(T)$.

ENO disadvantages:

1. The choice of stencil is sensitive to round-off errors near the roots of the solution and its derivatives.

2. The numerical flux is not smooth as the stencil pattern may change at neighboring points.

3. In the stencil choosing process k stencils are considered covering $2k-1$ cells but only one of the stencils is used. If information from all cells are used one can potentially get $2k-1$-th order accuracy in smooth regions.

4. ENO stencil choosing is not computationally efficient because of the repeated use of "if" structures in the code.

10.6.2 WENO reconstruction

WENO attempts to address the disadvantages of ENO, primarily a more efficient use of CPU time to gain accuracy in smooth region without sacrificing the TVB property in the presence of discontinuity. The basic idea is to use a convex combination of all stencils used in ENO to form a better estimate of the function value. Suppose the k candidate stencils $S(i;k,l), l = 0,\ldots,k-1$ produce the k different estimates:

$$T^l_{j+\frac{1}{2}} = \sum_{j=0}^{k-1} C_{lj}\overline{T}^z_{i-l+j}, \quad l = 0,\ldots,k-1 \tag{10.56}$$

then the WENO estimate is

$$T_{j+\frac{1}{2}} = \sum_{l=0}^{k-1} \omega_l T^l_{j+\frac{1}{2}}. \tag{10.57}$$

where ω_l are the weights satisfying the following requirements for consistency and stability:

$$\omega_l \geq 0, \quad \sum_{l=0}^{k-1} \omega_l = 1 \tag{10.58}$$

Furthermore, when the solution has a discontinuity in one or more of the stencils we would like the corresponding weights to be essentially 0 to emulate the ENO idea. The weights should also be smooth functions of the cell averages. The weights described below are in fact C^∞.

Shu et al propose the following forms for the weights:

$$\omega_l = \frac{\alpha_l}{\sum\limits_{n=0}^{k-1} \alpha_n}, \quad \alpha_l = \frac{d_l}{(\epsilon + \beta_l)^2} \quad l = 0,\ldots,k-1 \tag{10.59}$$

Here, the d_l are the factor needed to maximize the accuracy of the estimate, i.e. to make the truncation error $O(\Delta z^{2k-1})$. Note that the weights must be as close to d_l in smooth regions, actually we have the requirement that $\omega_l = d_l + O(\Delta z^k)$. The factor ϵ is introduced to avoid division by zero, a value of $\epsilon = 10^{-6}$ seems standard. Finally, β_l are the smoothness indicators of stencils $S(i;k,l)$. These factors are responsible for the success of WENO; they also account for much of the CPU cost increase over traditional methods. The requirements for the smoothness indicator are that $\beta_l = O(\Delta z^2)$ in smooth regions and $O(1)$ in the presence of discontinuities. This translates into $\alpha_l = O(1)$ in smooth regions and $O(\Delta z^4)$ in the presence of discontinuities. The smoothness measures advocated by Shu et al look like weighed H^{k-1} norms of the interpolating functions:

$$\beta_l = \sum_{n=1}^{k-1} \int_{z_{i-\frac{1}{2}}}^{z_{i+\frac{1}{2}}} \Delta z^{2n-1} \left(\frac{\partial^n p_l}{\partial z^n}\right)^2 \,\mathrm{d}z \tag{10.60}$$

The right hand side is just the squares of the scaled L_2 norms for all derivatives of the polynomial p_l over the interval $[z_{i-\frac{1}{2}}, z_{i+\frac{1}{2}}]$. The factor Δz^{2n-1} is introduced to remove any Δz dependence in the derivatives in order to preserve self similarity; the smoothness indicator are the same regardless of the underlying grid. The smoothness indicators for the case $k = 2$ are:

$$\beta_0 = (\overline{T}^z_{i+1} - \overline{T}^z_i)^2, \quad \beta_1 = (\overline{T}^z_i - \overline{T}^z_{i-1})^2 \tag{10.61}$$

Higher order formulae can be found in Shu (1998); Balsara and Shu (2000). The formulae given here have a one-point upwind bias in the optimal linear stencil suitable for a problem with wind blowing from left to right. If the wind blows the other way, the procedure should be modified symemetrically with respect to $z_{i+\frac{1}{2}}$.

10.6.3 ENO and WENO numerical experiments

Figure 10.6 (left panel) shows the convergence rates of ENO interpolation for the function $\sin 2\pi x$ for $-\frac{1}{2} \le x \le \frac{1}{2}$. Two sets of experiments were conducted. One set the shift to 0 so the interpolation is right tilted, and the other to $(k-1)/2$ where k is the polynomial order so that the stencil is centered. The two sets of experiments overlap for $k = 2, 3$. The convergence rates for both experiments are the same, although the centered stencils yield a lower error. The WENO reconstruction effectively doubles the convergence rates by using a convex combination of all stencils used in the reconstruction.

I have coded up a WENO advection scheme that can use variable order space interpolation (up to order 9), and up to 3rd order Runge-Kutta stepping. I have also experimented with the scheme in 1D. Figure 10.7 shows the advection of Shchepetkin's narrow profile (top left), wide profile (top right), and hat profile, (bottom left). The high order WENO5-RK3 scheme has less dissipation, and better phase properties than the WENO3-RK2 scheme. For the narrow Gaussian hill the peak is well preserved and the profile is narrower; it is indistinguishable from the analytical solution for the wider profile. Finally, although the scheme does not enforce TVD there is no evidence of dispersive ripples in the case of the hat profile; there are however small negative values.

I have tried to implement the shape preserving WENO scheme proposed by Suresh and Huynh (1997) and Balsara and Shu (2000). Their limiting attempts to preserve

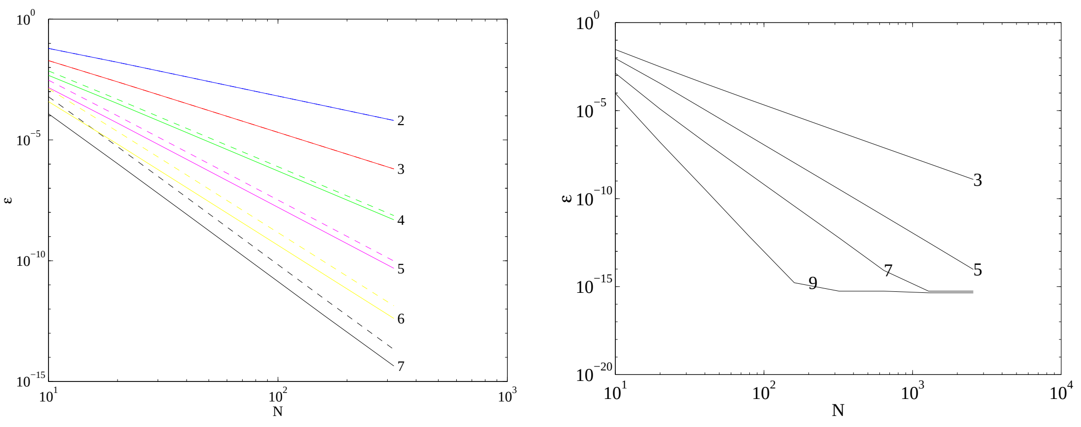


Figure 10.6: Convergence Rate (in the maximum norm) for ENO (left panel) and WENO (right panel) reconstuction. The dashed curves are for a left shift set to 0, while the solid curve are for centered interpolation. The numbers refer to the interpolation order

the high order accuracy of WENO near discontinuities and smoth extrema, and as such include a peak discriminator that picks out smooth extrema from discontinuous ones. As such, I think the scheme will fail to preserve the peaks of the original shape and will allow some new extrema to be generated. This is because there is no full proof discrimator. Consider what happens to a square wave advected by a uniform velocity field. The discontinuity is initially confined to 1 cell; the discriminator will rightly flag it as a discontinuous extremum and will apply the limiter at full strength. Subsequentally, numerical dissipation will smear the front across a few cells and the front width will occupy a wider stencil. The discriminator, which works by comparing the curvature at a fixed number of cells, will fail to pick the widening front as a discontinuity, and will permit out of range values to be mistaken for permissible smooth extrema.

In order to test the effectiveness of the limiter, I have tried the 1D advection of a square wave using the limited and unlimited WENO5 (5-th order) coupled with RK2. Figure 10.8 compares the negative minimum obtained with the limited (black) and unlimited (red) schemes; the x-axis represent time (the cone has undergone 4 rotations by the end of the simulation). The different panels show the result using 16, 32, 64 and 128 cells. The trend in all cases is similar for the unlimited scheme: a rapid growth of the negative extremum before it reaches a quasi-steady state. The trend for the limited case is different. Initially, the negative values are suppressed the black curves starting away from time 0. This is initial period increases with better resolution. After the first negative values appear, there is a rapid deterioration in the minimum value before reaching a steady state. This steady state value decreases with the number of points, and becomes negligeable for the 128 cell case. Finally, note that unlimited case produces a slightly better minimum for the case of the 16 cells, but does not improve substantially as the number of points is increased. For this experiment, the interval is the unit interval and the hat profile is confined to $|x| < 1/4$; the time step is held fix at $\Delta t = 1/80$, so the Courant number increases with the number of cells used.

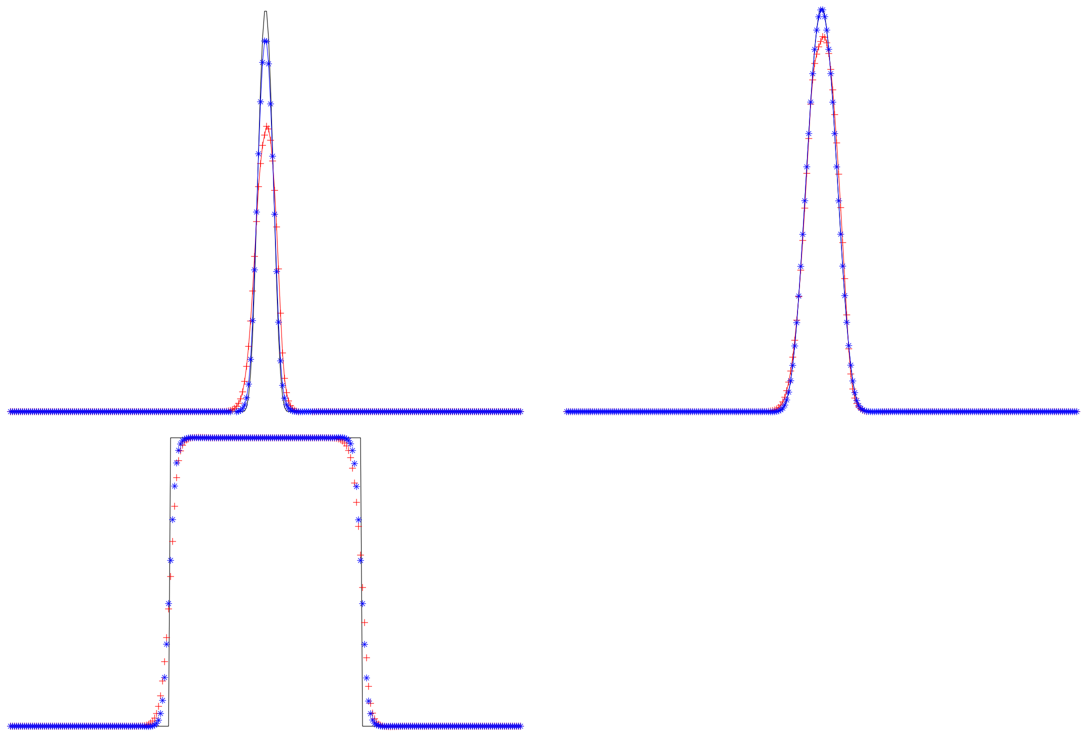


Figure 10.7: Advection of several Shchepetkin profiles. The black solid line refers to the analytical solution, the red crosses to the WENO3 (RK2 time-stepping), and the blue stars to WENO5 with RK3. The WENO5 is indistiguishable from the analytical solution for the narrow profile

I have repeated the experiments for the narrow profile case (Shchepetkin's profile), and confirmed that the limiter is indeed able to supress the generation of negative value, even for a resolution as low as 64 cells (the reference case uses 256 cells). The discriminator, however, allows a very slight and occasional increase of the peak value. By in large, the limiter does a good job. The 2D cone experiments with the limiters are shown in the Cone section.

10.7 Utopia

The uniform third order polynomial interpolation algorithm was derived explicitly to be a multi-dimension, two-time level, conservative advection scheme. The formulation is based on a finite volume formulation of the advection equation:

$$(\Delta V \overline{T})_t + \int_{\partial \Delta V} \mathbf{F} \cdot \mathbf{n} \, \mathrm{d}S = 0 \tag{10.62}$$

where $\overline{T}^z$ is the average of T over the cell ΔV and $\mathbf{F} \cdot \mathbf{n}$ are the fluxes passing through the surfaces $\partial \Delta V$ of the control volume. A further integral in time reduces the solution

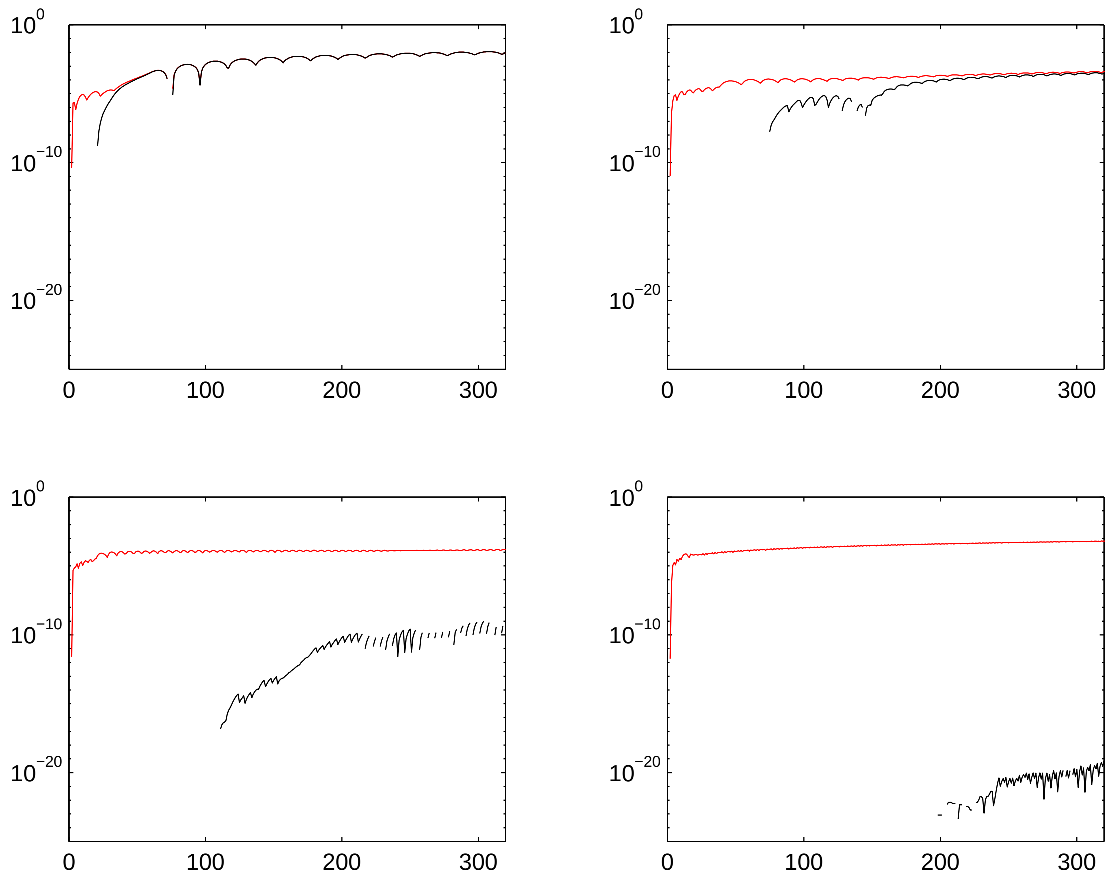


Figure 10.8: Negative minima of unlimited (red) and limited (black) WENO scheme on a square hat profile. Top left 16 points, top right 32 points, bottom left 64 points, and bottom rights 64 points.

to the following:

$$\overline{T}^{n+1} = \overline{T}^{n} + \frac{1}{\Delta V}\int_0^{\Delta t}\int_{\partial V} \mathbf{F}\cdot\mathbf{n}\,\mathrm{d}S\,\mathrm{d}t = 0 \tag{10.63}$$

A further definition will help us interpret the above formula. If we let the time-average flux passing the surfaces bounding ΔV as $\mathcal{F}\Delta t = \int_0^{\Delta t}\mathbf{F}\,\mathrm{d}t$ we end up with the following two-time level expression:

$$\overline{T}^{n+1} = \overline{T}^{n} + \frac{\Delta t}{\Delta V}\int_{\partial V} \mathcal{F}\cdot\mathbf{n}\,\mathrm{d}S \tag{10.64}$$

UTOPIA takes a Lagrangian point of view in tracing the fluid particle crossing each face. The situation is depicted in figure 10.9 where the particles crossing the left face of a rectangular cell, is the area within the quadrilateral ABCD; this is effectively the contribution of edge AD to the boundary integral $\Delta t\int_{\partial V_{AD}}\mathcal{F}\cdot\mathbf{n}\,\mathrm{d}S$. UTOPIA makes the assumption that the advecting velocity is locally constant across the face in space

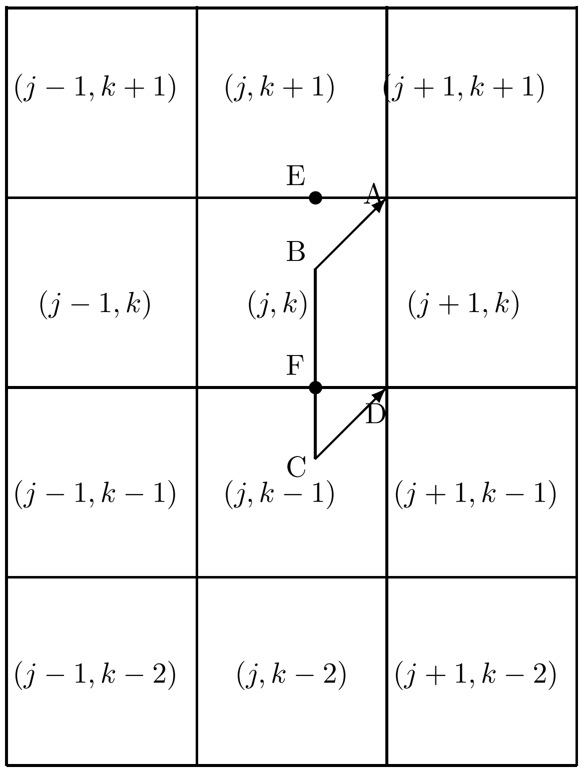


Figure 10.9: Sketch of the particles entering cell (j, k) through its left edge $(j - \frac{1}{2}, k)$ assuming positive velocity components u and v.

and time; this amount to approximating the curved edges of the area by straight lines as shown in the figure. The distance from the left edge to the straight line BC is $u\Delta t$, and can be expressed as $p\Delta x$ where p is the courant number for the x-component of the velocity. Likewise, the vertical distance between point B and edge $k + \frac{1}{2}$ is $v\Delta t = q\Delta y$, where q is the Courant number in the y direction.

We now turn to the issue of computing the integral of the boundary fluxes; we will illustrate this for edge AD of cell (j, k). Owing to UTOPIA's Lagrangian estimate of the flux we have:

$$\frac{\Delta t}{\Delta V} \int_{\partial V_{AD}} \mathcal{F} \cdot \mathbf{n} \, \mathrm{d}S = \frac{1}{\Delta x \Delta y} \int_{ABCD} T \, \mathrm{d}x \, \mathrm{d}y. \tag{10.65}$$

The right hand side integral is in area integral of T over portions of upstream neighboring cells. Its evaluation requires us to assume a form for the spatial variations of T. Several choices are available and Leonard et al Leonard et al. (1995) discusses several options. UTOPIA is built on assuming a quadratic variations; for cell (j, k), the interpolation is:

$$\begin{aligned} T_{j,k}(\xi,\eta) &= \overline{T}_{j,k} - \frac{1}{24}\left(\overline{T}_{j+1,k} + \overline{T}_{j,k-1} + \overline{T}_{j-1,k} + \overline{T}_{j,k+1} - 4\overline{T}_{j,k}\right) \\ &+ \frac{\overline{T}_{j+1,k} - \overline{T}_{j-1,k}}{2}\xi + \frac{\overline{T}_{j+1,k} - 2\overline{T}_{j,k} + \overline{T}_{j-1,k}}{2}\xi^2 \\ &+ \frac{\overline{T}_{j,k+1} - \overline{T}_{j,k-1}}{2}\eta + \frac{\overline{T}_{j,k+1} - 2\overline{T}_{j,k} + \overline{T}_{j,k-1}}{2}\eta^2. \end{aligned} \tag{10.66}$$

Here, ξ and η are scaled local coordinates:

$$\xi = \frac{x}{\Delta x} - i \quad \eta = \frac{y}{\Delta y} - j. \tag{10.67}$$

so that the center of the box is located at $(0, 0)$ and the left and right edges at $(\pm\frac{1}{2}, 0)$, respectively. The interpolation formula is designed such as to produce the proper cell-averages when the function is integrated over cells (j, k), $(j\pm1, k)$ and $(j, k\pm1)$. The area integral in equation 10.65 must be broken into several integral: First, the area ABCD stradles two cells, $(j - 1, k)$ and $(j - 1, k - 1)$, with two different interpolation for T; and second, the trapezoidal area integral can be simplified. We now have

$$\frac{1}{\Delta x \Delta y} \int_{ABCD} T \, \mathrm{d}x \, \mathrm{d}y = I_1(j, k) - I_2(j, k) + I_2(j, k - 1) \tag{10.68}$$

where the individual contributions from each area are given by:

$$I_1(j,k) = \int_{AEFD} T_{j,k}(\xi,\eta) \, \mathrm{d}\eta \, \mathrm{d}\xi = \int_{\frac{1}{2}-u}^{\frac{1}{2}} \int_{\frac{1}{2}-u}^{\frac{1}{2}} T_{j,k}(\xi,\eta) \, \mathrm{d}\eta \, \mathrm{d}\xi \tag{10.69}$$

$$I_2(j,k) = \int_{AEB} T_{j,k}(\xi,\eta) \, \mathrm{d}\eta \, \mathrm{d}\xi = \int_{\frac{1}{2}-u}^{\frac{1}{2}} \int_{\eta_{AB}(\xi)}^{\frac{1}{2}} T_{j,k}(\xi,\eta) \, \mathrm{d}\eta \, \mathrm{d}\xi. \tag{10.70}$$

The equation for the line AB is:

$$\eta_{AB}(\xi) = \frac{1}{2} + \frac{q}{p}\left(\xi - \frac{1}{2}\right) \tag{10.71}$$

$$\begin{aligned}
F_{j+\frac{1}{2},k} &= \frac{(f_{j,k+1}-2f_{j,k}+f_{j,k-1})-(f_{j,k}-2f_{j,k-1}+f_{j,k-2})}{24}uv^3 \\
&+ \frac{f_{j,k-1}-2f_{j,k}+f_{j,k+1}}{6}uv^2 \\
&+ \left[\frac{f_{j+1,k}-2f_{j,k}+f_{j-1,k}-f_{j+1,k-1}+2f_{j,k-1}-f_{j-1,k-1}}{8}u^2\right. \\
&\quad -\frac{(f_{j+1,k}-f_{j+1,k-1})-(f_{j,k}-f_{j,k-1})}{3}u \\
&\quad +\frac{(f_{j,k-2}-2f_{j,k-1}+f_{j,k})+2(f_{j+1,k}-f_{j+1,k-1})}{12} \\
&\quad \left.+\frac{2(f_{j,k+1}-f_{j,k-1})+(f_{j,k}-f_{j-1,k})-(f_{j,k-1}-f_{j-1,k-1})}{12}\right]uv \\
&+ \left[\left(\frac{f_{j+1,k}+f_{j,k}}{2}-\frac{f_{j+1,k}-2f_{j,k}+f_{j-1,k}}{6}\right)\right. \\
&\quad \left.-\frac{f_{j+1,k}-f_{j,k}}{2}u+\frac{f_{j+1,k}-2f_{j,k}+f_{j-1,k}}{6}u^2\right]u \qquad (10.72)
\end{aligned}$$

Figure 10.10: Flux for the finite volume form of the utopia algorithm.

using the local coordinates of cell (j,k). Performing the integration is rather tedious; the output of a symbolic computer program is shown in figure 10.10

A different derivation of the UTOPIA scheme can be obtained if we consider the cell values are function values and **not** cell averages. The finite difference form is then given by the equation shown in figure 10.11.

10.8 Lax-Wendroff for advection equation

We explore the application of the Lax-Wendroff procedure to compute high-order, two-time level approximation to the advection diffusion equation written in the form:

$$T_t + \nabla\cdot(\mathbf{u}T) = 0. \qquad (10.74)$$

The starting point is the time Taylor series expansion which we carry out to fourth order:

$$T^{n+1} = T^n + \frac{\Delta t}{1!}T_t + \frac{\Delta t^2}{2!}T_{tt} + \frac{\Delta t^3}{3!}T_{ttt} + \frac{\Delta t^4}{4!}T_{tttt} + O(\Delta t^5). \qquad (10.75)$$

The next step is the replacement of the time-derivative above with spatial derivatives using the original PDE. It is easy to derive the following identities:

$$\begin{aligned}
T_t &= -\nabla\cdot[\mathbf{u}T] && (10.76)\\
T_{tt} &= -\nabla\cdot[\mathbf{u}_tT+\mathbf{u}T_t] \\
T_{tt} &= -\nabla\cdot[\mathbf{u}_tT-\mathbf{u}\nabla\cdot(\mathbf{u}T)] && (10.77)\\
T_{ttt} &= -\nabla\cdot[\mathbf{u}_{tt}T-2\mathbf{u}_t\nabla\cdot(\mathbf{u}T)-\mathbf{u}\nabla\cdot(\mathbf{u}_tT)+\mathbf{u}\nabla\cdot(\mathbf{u}\nabla\cdot(\mathbf{u}_tT))] && (10.78)
\end{aligned}$$

$$\begin{aligned}
F_{j+\frac{1}{2},k} &= \frac{f_{m,n-1} - 3f_{m,n-2} + 3f_{m,n+1} - f_{m,n}}{24} uv^3 \\
&+ \frac{f_{m,n-1} + f_{m,n+1} - 2f_{m,n}}{6} uv^2 \\
&+ \left[\frac{-5f_{m,n-1} - 3f_{m+1,n-1} + 3f_{m,n+1} + 3f_{m+1,n} + f_{m-1,n-1} - f_{m-1,n} + f_{m,n-2} + f_{m,n}}{16} \right. \\
&\quad + \frac{-f_{m+1,n} + f_{m,n} + f_{m+1,n-1} - f_{m,n-1}}{3} u \\
&\quad \left. + \frac{+f_{m+1,n} + f_{m-1,n} - f_{m+1,n-1} - 2f_{m,n} + 2f_{m,n-1} - f_{m-1,n-1}}{8} u^2 \right] uv \\
&+ \left[\frac{+f_{m,n+1} - 3f_{m-1,n} + 16f_{m,n} + f_{m,n-1} + 9f_{m+1,n}}{24} \right. \\
&\quad \left. + \frac{f_{m,n} - f_{m+1,n}}{2} u + \frac{f_{m+1,n} - 2f_{m,n} + f_{m-1,n}}{6} u^2 \right] u \qquad (10.73)
\end{aligned}$$

Figure 10.11: Flux of UTOPIA when variables represent function values. This is the finite difference form of the scheme

Chapter 11

Finite Element Methods

The discretization of complicated flow domains with finite difference methods is quite cumbersome. Their straightfoward application requires the flow to occur in logically rectangular domains, a constraint that severely limit their capabilities to simulate flows in realistic geometries. The finite element method was developed in large part to address the issue of solving PDE's in arbitrarily complex regions. Briefly, the FEM method works by dividing the flow region into cells referred to as element. The solution within each element is approximated using some form of interpolation, most often polynomial interpolation; and the weights of the interpolation polynomials are adjusted so that the residual obtained after applying the PDE is minimized. There are a number of FE approaches in existence today; they differ primarily in their choice of interpolation procedure, type of elements used in the discretization; and the sense in which the residual is minimized. Finlayson Finlayson (1972) showed how the different approaches can be unified via the perspective of the Mean Weighed Residual (MWR) method.

11.1 MWR

Consider the following problem: find the function u such that

$$L(u) = 0 \tag{11.1}$$

where L is a linear operator defined on a domain Ω; if L is a differential operator, appropriate initial and boundary conditions must be supplied. The continuum problem as defined in equation 11.1 is an infinite dimensional problem as it requires us to find u at every point of Ω. The essence of numerical discretization is to turn this infinite dimensional system into a finite dimensional one:

$$\tilde{u} = \sum_{j=0}^{N} \hat{u}_j \phi(\mathbf{x}) \tag{11.2}$$

Here $\tilde{u}$ stands for the approximate solution of the problem, $\hat{u}$ are the $N+1$ degrees of freedom available to minimize the error, and the ϕ's are the interpolation or trial functions. Equation 11.2 can be viewed as an expansion of the solution in term of a basis function defined by the functions ϕ. Applying this series to the operator L we obtain

$$L(\tilde{u}) = R(\mathbf{x}) \tag{11.3}$$

where $R(\mathbf{x})$ is a residual which is different then zero unless $\tilde{u}$ is the exact solution of the equation 11.1. The degrees of freedom $\hat{u}$ can now be chosen to minimize R. In order to determine the problem uniquely, I can impose $N+1$ constraints. For MWR we require that the inner product of R with a $N+1$ test functions v_j to be orthogonal:

$$(R, v_j) = 0, j = 0, 1, 2, \ldots, N. \tag{11.4}$$

Recalling the chapter on linear analysis; this is equivalent to saying that the projection of R on the set of functions v_j is zero. In the case of the inner product defined in equation 12.13 this is equivalent to

$$\int_\Omega R v_j \, \mathrm{d}\mathbf{x} = 0, j = 0, 1, 2, \ldots, N. \tag{11.5}$$

A number of different numerical methods can be derived by assigning different choices to the test functions.

11.1.1 Collocation

If the test functions are defined as the Dirac delta functions $v_j = \delta(\mathbf{x}-\mathbf{x}_j)$, then constraint 11.4 becomes:

$$R(\mathbf{x}_j) = 0 \tag{11.6}$$

i.e. it require the residual to be identically zero on the collocation points $\mathbf{x}_j$. Finite differences can thus be cast as a MWR with collocation points defined on the finite difference grid. The residual is free to oscillate between the collocation points where it is pinned to zero; the oscillations amplitude will decrease with the number of of collocation points if the residual is a smooth function.

11.1.2 Least Square

Setting the test functions to $v_j = \frac{\partial R}{\partial \hat{u}_j}$ is equivalent to minimizing the norm of the residual $\|R\|^2 = (R, R)$. Since the only parameters available in the problem are $\hat{u}_j$, this is equivalent to finding $\hat{u}_j$ that minimize $\|R\|^2$. This minimum occurs for $\hat{u}_j$ such that

$$\begin{aligned}
\frac{\partial}{\partial \hat{u}_j}\left(\int_\Omega R^2 \, \mathrm{d}\mathbf{x}\right) &= 0 && (11.7)\\
\left(\int_\Omega 2R\frac{\partial R}{\partial \hat{u}_j} \, \mathrm{d}\mathbf{x}\right) &= 0 && (11.8)\\
\left(R, \frac{\partial R}{\partial \hat{u}_j}\right) &= 0 && (11.9)
\end{aligned}$$

11.1.3 Galerkin

In the Galerkin method the test functions are taken to be the same as the trial functions, so that $v_j = \phi_j$. This is the most popular choice in the FE community and will be the one we concentrate on. There are varitions on the Galerkin method where the test functions are perturbation of the trial functions. This method is usually referred as the Petrov-Galerkin method. The perturbations are introduced to improve the numerical stability of the scheme; for example to introduce upwinding in the solution of advection dominated flows.

11.2 FEM example in 1D

We illustrate the application of the FEM method by focussing on a specific problem. Find $u(x)$ in $x \in [0, 1]$ such that

$$\frac{\partial^2 u}{\partial x^2} - \lambda u + f = 0 \tag{11.10}$$

subject to the boundary conditions

$$\begin{aligned} u(x = 0) &= 0, & (11.11)\\ \frac{\partial u}{\partial x} &= q & (11.12) \end{aligned}$$

Equation 11.11 is a Dirichlet boundary conditions and is usually referred to as an essential boundary condition, while equation 11.12 is usually referred to as a natural boundary conditions. The origin of these terms will become clearer shortly.

11.2.1 Weak Form

In order to cast the equation into a residual formulation, we require that the inner product with suitably chosen test functions v is zero:

$$\int_0^1 \left(\frac{\partial^2 u}{\partial x^2} - \lambda u + f \right) v \,\, \mathrm{d}x = 0 \tag{11.13}$$

The only condition we impose on the test function is that it is zero on those portions of the boundary where Dirichlet boundary conditions are applied; in this case $v(0) = 0$. Equation 11.13 is called the strong form of the PDE as it requires the second derivative of the function to exist and be integrable. Imposing this constraint in geometrically complex region is difficult, and we seek to reformulate the problem in a "weak" form such that only lower order derivatives are needed. We do this by integrating the second derivative in equation 11.13 by part to obtain:

$$\int_0^1 \frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \lambda uv \,\mathrm{d}x = \int_0^1 fv \,\, \mathrm{d}x + \left[v\frac{\partial u}{\partial x} \right]_1 - \left[v\frac{\partial u}{\partial x} \right]_0 \tag{11.14}$$

The essential boundary conditions on the left edge eliminates the third term on the right hand side of the equation since $v(0) = 0$, and the Neumann boundary condition at the right edge can be substituted in the second term on the right hand side. The final form is thus:

$$\int_0^1 \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \lambda uv\right)\ \mathrm{d}x = \int_0^1 fv\ \ \mathrm{d}x + qv(1) \tag{11.15}$$

For the weak form to be sensible, we must require that the integrals appearing in the formulation be finite. The most severe restriction stems from the first order derivatives appearing on the left hand side of 11.15. For this term to be finite we must require that the functions $\frac{\partial u}{\partial x}$ and $\frac{\partial v}{\partial x}$ be integrable, i.e. piecewise continuous with finite jump discontinuities; this translates into the requirement that the functions u and v be continuous, the so-called $\mathbf{C^0}$ **continuity** requirement.

11.2.2 Galerkin form

The solution is approximated with a finite sum as:

$$u(x) = \sum_{i=0}^{N} \hat{u}_i \phi_i \tag{11.16}$$

and the test functions are taken to be $v = \phi_j$, $j = 1, 2, \ldots, N$. The trial functions ϕ_i must be chosen such that $\phi_{i>0}(0) = 0$, in accordance with the restriction that $v(0) = 0$. We also set, without loss of generality, $\phi_0(0) = 1$, the first term of the series is then nothing but the value of the function at the edge where the Dirichlet condition is applied: $u(0) = \hat{u}_0$. The substitution of the expansion and test functions into the weak form yield the following N system of equations in the $N + 1$ variables $\hat{u}_i$:

$$\sum_{i=0}^{N} \int_0^1 \left(\frac{\partial \phi_i}{\partial x}\frac{\partial \phi_j}{\partial x} + \lambda\phi_i\phi_j\ \mathrm{d}x\right)\hat{u}_i = \int_0^1 f\phi_j\ \ \mathrm{d}x + q\phi_j(1) \tag{11.17}$$

In matrix form this can be re-written as

$$\sum_{i=0}^{N} K_{ji}u_i = b_j,\ \ K_{ji} = \int_0^1 \frac{\partial \phi_j}{\partial x}\frac{\partial \phi_i}{\partial x} + \lambda\phi_i\phi_j\ \ \mathrm{d}x,\ \ b_j = \int_0^1 f\phi_j\ \ \mathrm{d}x + q\phi_j(1) \tag{11.18}$$

Note that the matrix K is symmetric: $K_{ji} = K_{ij}$, so that only half the matrix entries need to be evaluated. The Galerkin formulation of the weak variational statement 11.15 will always produce a symmetric matrix regardless of the choice of expansion function; the necessary condition for the symmetry is that the left hand side operator in equation 11.15 be symmetric with respect to the u and v variables. The matrix K is usually referred to as the **stiffness** matrix, a legacy term dating to the early application of the finite element method to solve problems in solid mechanics.

11.2.3 Essential Boundary Conditions

The series has the N unknowns $\hat{u}_{1\leq i\leq N}$, thus the matrix equation above must be modified to take into account the boundary condition. We do this by moving all known qunatities to the right hand side, and we end up with the following system of equations:

$$\sum_{i=1}^{N} K_{ji}u_i = c_j,\ c_j = bj - K_{j0}u_0,\ j = 1, 2, \ldots, N \tag{11.19}$$

Had the boundary condition on the right edge of the domain been of Dirichlet type, we would have to add the following restrictions on the trial functions $\phi_{2\leq i\leq N-1}(1) = 0$. The imposition of Dirichlet conditions on both sides is considerably simplified if we further request that $\phi_0(1) = \phi_N(0) = 0$ and $\phi_0(0) = \phi_N(1) = 1$. Under these conditions $\hat{u}_0 = u(0) = u_0$ and $\hat{u}_N = u(1) = u_N$. We end up with the following $(N-1)\times(N-1)$ system of algebraic equations

$$\sum_{i=1}^{N-1} K_{ji}u_i = c_j,\; c_j = bj - K_{j0}u_0 - KjNu_N,\; j = 1,2,\ldots,N-1 \tag{11.20}$$

11.2.4 Choice of interpolation and test functions

To complete the discretization scheme we need to specify the type of interpolation functions to use. The choice is actually quite open save for the restriction on using **continuous functions** (to integrate the first order derivative terms), and imposing the Dirichlet boundary conditions. There are two aspects to choosing the test functions: their locality and their interpolation properties.

If the functions ϕ_i are defined over the entire domain, they are termed **global** expansion functions. Such functions are most often used in spectral and pseudo-spectral methods. They provide a very accurate representation for functions that are smooth; in fact the rate of convergence increases faster then any finite power of N if the solution is infinitely smooth, a property known as exponential convergence. This property is lost if the solution has finite continuity. The main drawback of global expansion functions is that the resulting matrices are full and tightly couple all degrees of freedom. Furthermore, the accurate representation of local features, such as fronts and boundary layers, requires long series expasions with substantial increase in the computational cost.

Finite element methods are based on **local** expansion functions: the domain is divided into elements wherein the solution is expanded into a finite series. The functions ϕ_i are thus non-zero only within one element, and zero elsewhere. This local representation of the function is extremely useful if the solution has localized features such as boundary layers, local steep gradient, etc... The resulting matrices are sparser and hence more efficient solution schemes become possible. The most popular choice of expansion function is the linear interpolation function, commonly referred to the hat functions which we will explore later on. Higher order expansion are also possible, in particular the spectral element method chooses expansion that are high order polynomial within each element.

The nature of the expansion function refers to the nature of the expansion coefficients. If a **spectral** representation is chosen, then the unknowns become the generalized Fourier (or spectral) coefficients. Again this is a common choice for spectral methods. The most common choice of expansion functions in finite element methods are **Lagrangian** interpolation functions, i.e. functions that interpolated the solution at specified points x_j also referred to as **collocation** points; in FEM these points are also referred to as **nodes**. Lagrangian interpolants are chosen to have the following property:

$$\phi_j(x_i) = \delta_{ij} \tag{11.21}$$

where δ_{ij} is the Kronecker delta. The interpolation function ϕ_j is zero at all points $x_{i\neq j}$; at point x_j, $\phi_j(x_j) = 1$. Each interpolation function is associated with one collocation

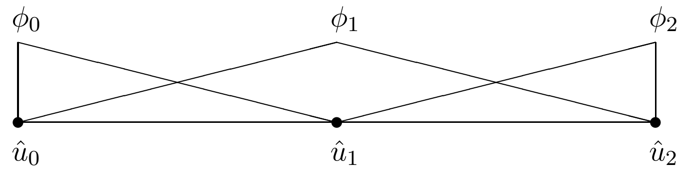


Figure 11.1: 2-element discretization of the interval $[0, 1]$ showing the interpolation functions

point. If our expansion functions are Lagrange interpolants, then the coefficients $\hat{u}_i$ represent the value of the function at the collocation points x_j:

$$u(x_j) = u_j = \sum_{i=0}^{N} \hat{u}_i \phi(x_j) = \hat{u}_j, \quad j = 0, 1, \ldots, N \tag{11.22}$$

We will omit the circumflex accents on the coefficients whenever the expansion functions are Lagrangian interpolation functions. The use of Lagrangian interpolation simplifies the imposition of the C^0 continuity requirement, and the function values at the collocation points are obtained directly without further processing.

There are other expansion functions in use in the FE community. For example, Hermitian interpolation functions are used when the solution and its derivatives must be continuous across element boundaries (the solution is then C^1 continuous); or Hermitian expansion is used to model infinite elements. These expansion function are usually reserved for special situations and we will not address them further.

In the following 3 sections we will illustrate how the FEM solution of equation 11.15 proceeds. We will approach the problem from 3 different perspectives in order to highlight the algorithmic steps of the finite element method. The first approach will consider a small size expansion for the approximate solution, the matrix equation can then be written and inverted manually. The second approach repeats this procedure using a longer expansion, the matrix entries are derived but the solution of the larger system must be done numerically. The third approach considers the same large problem as number two above; but introduces the local coordinate and numbering systems, and the mapping between the local and global systems. This local approach to constructing the FE stiffness matrix is key to its success and versatility since it localizes the computational details to elements and subroutines. A great variety of local finite element approximations can then be introduced at the local elemental level with little additional complexity at the global level.

11.2.5 FEM solution using 2 linear elements

We illustrate the application of the Galerkin procedure for a 2-element discretization of the interval $[0, 1]$. Element 1 spans the interval $[0, \frac{1}{2}]$ and element 2 the interval $[\frac{1}{2}, 1]$ and we use the following interpolation procedure:

$$u(x) = u_0\phi_0(x) + u_1\phi_1(x) + u_2\phi_2(x) \tag{11.23}$$

where the interpolation functions and their derivatives are tabulated below

	$\phi_0(x)$	$\phi_1(x)$	$\phi_2(x)$	$\frac{\partial\phi_0}{\partial x}$	$\frac{\partial\phi_1}{\partial x}$	$\frac{\partial\phi_2}{\partial x}$
$0 \le x \le \frac{1}{2}$	$1-2x$	$2x$	0	-2	2	0
$\frac{1}{2} \le x \le 1$	0	$2(1-x)$	$2x-1$	0	-2	2

(11.24)

and shown in figure 11.1. It is easy to verify that the ϕ_i are Lagrangian interpolation functions at the 3 collocations points $x = 0, 1/2, 1$, i.e. $\phi_i(x_j) = \delta_{ij}$. Furthermore, the expansion functions are continuous across element interfaces, so that the C^0 continuity requirement is satisfied), but their derivates are discontinuous. It is easy to show that the interpolation 11.23 amounts to a linear interpolation of the solution within each element.

Since the boundary condition at $x = 0$ is of Dirichlet type, we need only test with functions that satisfy $v(0) = 0$; in our case the functions ϕ_1 and ϕ_2 are the only candidates. Notice also that we have only 2 unknowns u_1 and u_2, u_0 being known from the Dirichlet boundary conditions; thus only 2 equations are needed to determine the solution. The matrix entries can now be determined. We illustrate this for two of the entries, and assuming λ is constant for simplicity:

$$K_{10} = \int_0^1 \left(\frac{\partial\phi_1}{\partial x}\frac{\partial\phi_0}{\partial x} + \lambda\phi_1\phi_0\right)\, \mathrm{d}x = \int_0^{\frac{1}{2}} [-4 + \lambda(2x - 4x^2)]\, \mathrm{d}x = -2 + \frac{\lambda}{12} \tag{11.25}$$

Notice that the integral over the entire domain reduces to an integral over a single element because the interpolation and test functions ϕ_0 and ϕ_1 are non-zero only over element 1. This property that localizes the operations needed to build the matrix equation is key to the success of the method.

The entry K_{11} requires integration over both elements:

$$\begin{aligned} K_{11} &= \int_0^1 \left(\frac{\partial\phi_1}{\partial x}\frac{\partial\phi_1}{\partial x} + \lambda\phi_1\phi_1\right)\, \mathrm{d}x && (11.26)\\ &= \int_0^{\frac{1}{2}} [4 + \lambda 4x^2]\, \mathrm{d}x + \int_{\frac{1}{2}}^1 [4 + \lambda 4(1-x)^2]\, \mathrm{d}x && (11.27)\\ &= \left(2 + \frac{2\lambda}{12}\right) + \left(2 + \frac{2\lambda}{12}\right) = 4 + \frac{4\lambda}{12} && (11.28)\end{aligned}$$

The remaining entries can be evaluated in a similar manner. The final matrix equation takes the form:

$$\begin{pmatrix} -2+\frac{\lambda}{12} & 4+\frac{4\lambda}{12} & -2+\frac{\lambda}{12} \\ 0 & -2+\frac{\lambda}{12} & 2+\frac{2\lambda}{12} \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \tag{11.29}$$

Note that since the value of u_0 is known we can move it to the right hand side to obtain the following system of equations:

$$\begin{pmatrix} 4+\frac{4\lambda}{12} & -2+\frac{\lambda}{12} \\ -2+\frac{\lambda}{12} & 2+\frac{2\lambda}{12} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} b_1 + \left(2 - \frac{\lambda}{12}\right) u_0 \\ b_2 \end{pmatrix} \tag{11.30}$$

whose solution is given by

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{\Delta} \begin{pmatrix} 2+\frac{2\lambda}{12} & 2-\frac{\lambda}{12} \\ 2-\frac{\lambda}{12} & 4+\frac{4\lambda}{12} \end{pmatrix} \begin{pmatrix} b_1 + \left(2-\frac{\lambda}{12}\right) u_0 \\ b_2 \end{pmatrix} \tag{11.31}$$

where $\Delta = 8(1+\frac{\lambda}{12})^2 - (\frac{\lambda}{12}-2)^2$ is the determinant of the matrix. The only missing piece is the evaluation of the right hand side. This is easy since the function f and the flux q are known. It is possible to evaluate the integrals directly if the global expression for f is available. However, more often that not, f is either a complicated function, or is known only at the collocation points. The interpolation methodology that was used for the unknown function can be used to interpolate the forcing functions and evalute their associated integrals. Thus we write:

$$\begin{aligned} b_j &= \int_0^1 f\phi_j \;\mathrm{d}x + q\phi_j(1) && (11.32)\\ &= \int_0^1 \sum_{i=0}^{2} (f_i\phi_i)\phi_j \;\mathrm{d}x + q\phi_j(1) && (11.33)\\ &= \sum_{i=0}^{2}\left(\int_0^1 \phi_i\phi_j \;\mathrm{d}x\right) f_i + q\phi_j(1) && (11.34)\\ &= \frac{1}{12}\begin{pmatrix} 1 & 4 & 1 \\ 0 & 1 & 2 \end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \\ f_2 \end{pmatrix} + \begin{pmatrix} 0 \\ q \end{pmatrix} && (11.35) \end{aligned}$$

The final solution can thus be written as:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{\Delta} \begin{pmatrix} 2+\frac{2\lambda}{12} & 2-\frac{\lambda}{12} \\ 2-\frac{\lambda}{12} & 4+\frac{4\lambda}{12} \end{pmatrix} \begin{pmatrix} \frac{f_0+4f_1+f_2}{12} + \left(2-\frac{\lambda}{12}\right) u_0 \\ \frac{f1+2f_2}{12} + q \end{pmatrix} \tag{11.36}$$

If $u_0 = 0$, $\lambda = 0$ and $f = 0$, the analytical solution to the problem is $u = qx$. The finite element solution yields:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} 0 \\ q \end{pmatrix} = \begin{pmatrix} \frac{q}{2} \\ q \end{pmatrix} \tag{11.37}$$

which is the exact solution of the PDE. The FEM procedure produces the exact result because the solution to the PDE is linear in x. Notice that the FE solution is exact at the interpolation points $x = 0, 1/2, 1$ *and* inside the elements.

If $f = -1$, and the remaining parameters are unchanged the exact solution is quadratic $u_e = x^2/2 + (q-1)x$, and the finite element solution is

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} \frac{-1}{2} \\ q-\frac{1}{4} \end{pmatrix} = \begin{pmatrix} \frac{4q-3}{8} \\ q-\frac{1}{2} \end{pmatrix} \tag{11.38}$$

Notice that the FE procedure yields the exact value of the solution at the 3 interpolation points. The errors committed are due to the interpolation of the function within the

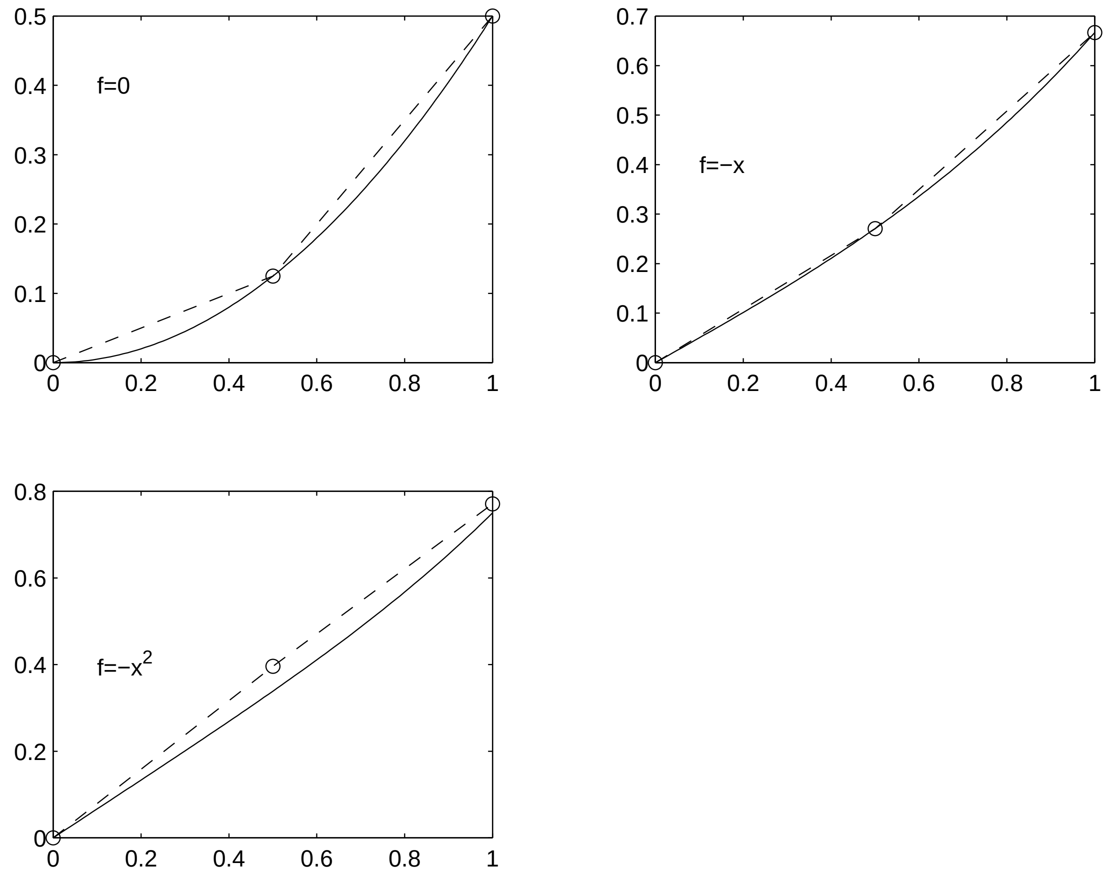


Figure 11.2: Comparison of analytical (solid line) and FE (dashed line) solutions for the equation $u_x x + f = 0$ with homogeneous Dirichlet condition on the left edge and Neumann condition $u_x = 1$ on the right edge. The circles indicate the location of the interpolation points. Two linear finite elements are used in this example.

elements; the solution is quadratic whereas the FE interpolation provide only for a linear variation.

For $f = -x$, the exact solution is $u_e = x^3/6 + (q - 1/2)x$, and the FE solution is:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} \frac{-1}{4} \\ q - \frac{5}{24} \end{pmatrix} = \begin{pmatrix} \frac{24q-11}{48} \\ q - \frac{1}{3} \end{pmatrix} \tag{11.39}$$

The solution is again exact at the interpolation points by in error within the element due to the linear interpolation. This fortuitous state of affair is due to the exact evaluation of the forcing term f which is also exactly interpolated by the linear functions.

For $f = -x^2$, the exact solution is $u = x^4/12 + (q - 1/2)x$, and the FE solution is:

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 2 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} \frac{-1}{12} \\ q - \frac{9}{48} \end{pmatrix} = \begin{pmatrix} \frac{48q-10}{96} \\ q - \frac{22}{96} \end{pmatrix} \tag{11.40}$$

This time the solution is in error at the interpolation points also. Figure 11.2 compare the analytical and FE solutions for the 3 cases after setting $q = 1$.

11.2.6 FEM solution using N linear elements

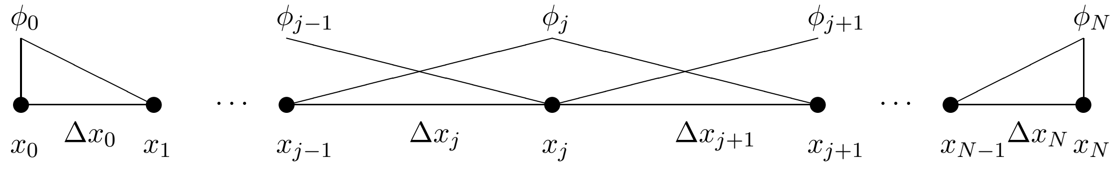


Figure 11.3: An element partition of the domain into N linear elements. The element edges are indicated by the filled dots.

In order to decrease the error further for cases where f is a more complex function, we need to increase the number of elements. This will increase the size of the matrix system and the computational cost. We go through the process of developing the stiffness equations for this system since it will be useful for the understanding of general FE concepts. Suppose that we have divided the interval into N elements (not necessarily of equal size), then interpolation formula becomes:

$$u(x) = \sum_{i=0}^{N} u_i \phi_i(x) \tag{11.41}$$

Element number j, shown in figure 11.3, spans the interval $[x_{j-1}, x_j]$, for $j = 1, 2, \ldots, N$; its left neighbor is element $j - 1$ and its right number is element $j + 1$. The length of each element is $\Delta x_j = x_j - x_{j-1}$. The linear interpolation function associated with node

j is

$$\phi_j(x) = \begin{cases} 0 & x < x_{j-1} \\ \frac{x-x_{j-1}}{x_j-x_{j-1}} & x_{j-1} \le x \le x_j \\ \frac{x_{j+1}-x}{x_{j+1}-x_j} & x_j \le x \le x_{j+1} \\ 0 & x_{j+1} < x \end{cases} \tag{11.42}$$

Let us now focus on building the stiffness matrix row by row. The j^{th} row of K corresponds to setting the test function to ϕ_j. Since the function ϕ_j is non-zero only on the interval $[x_{j-1}, x_{j+1}]$, the integral in the stiffness matrix reduces to an integration over that interval. We thus have:

$$K_{ji} = \int_{x_{j-1}}^{x_{j+1}} \left(\frac{\partial \phi_j}{\partial x}\frac{\partial \phi_i}{\partial x} + \lambda \phi_i \phi_j \right) \, \mathrm{d}x, \tag{11.43}$$

$$b_j = \int_{x_{j-1}}^{x_{j+1}} f\phi_j \, \mathrm{d}x + q\phi_j(1) \tag{11.44}$$

Likewise, the function ϕ_i is non-zero only on elements i and $i+1$, and hence $K_{ji} = 0$ unless $i = j-1, j, j+1$; the system of equation is hence **tridiagonal**. We now derive explicit expressions for the stiffness matrix entries for $i = j, j \pm 1$.

$$K_{j,j-1} = \int_{x_{j-1}}^{x_j} \left(\frac{\partial \phi_j}{\partial x}\frac{\partial \phi_{j-1}}{\partial x} + \lambda \phi_{j-1}\phi_j \right) \, \mathrm{d}x, \tag{11.45}$$

$$= \int_{x_{j-1}}^{x_j} \left(-\frac{1}{(x_j - x_{j-1})^2} + \lambda \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x_j - x}{x_j - x_{j-1}} \right) \, \mathrm{d}x, \tag{11.46}$$

$$= -\frac{1}{\Delta x_j} + \lambda \frac{\Delta x_j}{6} \tag{11.47}$$

The entry for $K_{j,j+1}$ can be deduced automatically by using symmetry and applying the above formula for $j+1$; thus:

$$K_{j,j+1} = K_{j+1,j} = -\frac{1}{\Delta x_{j+1}} + \lambda \frac{\Delta x_{j+1}}{6} \tag{11.48}$$

The sole entry remaining is $i = j$; in this case the integrals spans the elements i and $i+1$

$$K_{j,j} = \int_{x_{j-1}}^{x_j} \left(\frac{\partial \phi_j}{\partial x}\frac{\partial \phi_j}{\partial x} + \lambda \phi_j \phi_j \right) \, \mathrm{d}x + \int_{x_j}^{x_{j+1}} \left(\frac{\partial \phi_j}{\partial x}\frac{\partial \phi_j}{\partial x} + \lambda \phi_j \phi_j \right) \, \mathrm{d}x, \tag{11.49}$$

$$= \frac{1}{\Delta x_j^2} \int_{x_{j-1}}^{x_j} \left[1 + \lambda (x - x_{j-1})^2 \right] \, \mathrm{d}x, + \frac{1}{\Delta x_{j+1}^2} \int_{x_j}^{x_{j+1}} \left[1 + \lambda (x_{j+1} - x)^2 \right] \mathrm{d}x, \tag{11.50}$$

$$= \left(\frac{1}{\Delta x_j} + \lambda \frac{2\Delta x_j}{6} \right) + \left(\frac{1}{\Delta x_{j+1}} + \lambda \frac{2\Delta x_{j+1}}{6} \right)$$

Note that all entries in the matrix equations are identical except for the rows associated with the end points where the diagonal entries are different. It is easy to show that we must have:

$$K_{0,0} = \left(\frac{1}{\Delta x_1} + \lambda \frac{2\Delta x_1}{6} \right) \tag{11.51}$$

$$K_{N,N} = \left(\frac{1}{\Delta x_N} + \lambda \frac{2\Delta x_N}{6} \right) \tag{11.52}$$

The evaluation of the right hand sides leads to the following equations for b_j:

$$\begin{aligned} b_j &= \int_{x_{j-1}}^{x_{j+1}} f\phi_j \ \mathrm{d}x + q\phi_j(1) && (11.53)\\ &= \sum_{i=j-1}^{j+1} \int_{x_{j-1}}^{x_{j+1}} \phi_i\phi_j \ \mathrm{d}x f_i + q\phi_N(1)\delta_{Nj} && (11.54)\\ &= \frac{1}{6}\left[\Delta x_j f_{j-1} + 2(\Delta x_j + \Delta x_{j+1})f_j + \Delta x_{j+1} f_{j+1}\right] + q\phi_N(1)\delta_{Nj} && (11.55)\end{aligned}$$

Again, special consideration must be taken when dealing with the edge points to account for the boundary conditions properly. In the present case b_0 is not needed since a Dirichlet boundary condition is applied on the left boundary. On the right boundary the right hand side term is given by:

$$b_N = \frac{1}{6}\left[\Delta x_j f_{N-1} + 2(\Delta x_N f_j\right] + q \tag{11.56}$$

Note that the flux boundary condition affects only the last entry of the right hand side.

If the grid spacing is constant, a typical of the matrix equation is:

$$\left(-\frac{1}{\Delta x} + \frac{\lambda\Delta x}{6}\right)u_{j-1} + 2\left(\frac{1}{\Delta x} + \frac{2\lambda\Delta x}{6}\right)u_j + \left(-\frac{1}{\Delta x} + \frac{\lambda\Delta x}{6}\right)u_{j+1} = \frac{\Delta x}{6}\left(f_{j-1} + 4f_j + f_{j+1}\right) \tag{11.57}$$

For $\lambda = 0$ it is easy to show that the left hand side reduces to the centered finite difference approximation of the second order derivative. The finite element discretization produces a more complex approximation for the right hand side involving a weighing of the function at several neighboring points.

11.2.7 Local stiffness matrix and global assembly

We note that in the previous section we have built the stiffness matrix by constantly referring to a global node numbering system and a global coordinate system across all elements. Although this is practical and simple in one-dimensional problems, it becomes very tedious in two and three dimensions, where elements can have arbitrary orientation with respect to the global coordinate system. It is thus useful to transform the computations needed to a local coordinate system and a local numbering system in order to simplify/automate the building of the stiffness matrix. We illustrate these local entities in the one-dimensional case since they are easiest to grasp in this setting.

For each element we introduce a local coordinate system that maps the element j defined over $x_{j-1} \le x \le x_j$ into $-1 \le \xi \le 1$. The following linear map is the simplest transformation that accomplishes that:

$$\xi = 2\frac{x - x_{j-1}}{\Delta x_j} - 1 \tag{11.58}$$

This linear transformation maps the point x_{j-1} into $\xi = -1$ and x_j into $\xi = 1$; its inverse is simply

$$x = \Delta x_j \frac{\xi + 1}{2} + x_{j-1} \tag{11.59}$$

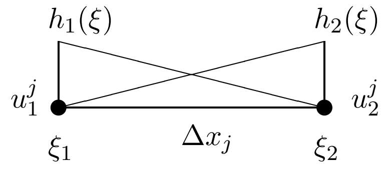


Figure 11.4: Local coordinate system and local numbering system

We also introduce a local numbering system so the unknown can be identified locally. The superscript j, whenever it appears, indicate that a local numbering system is used to refer to entities defined over element j. In the present instance the u_1^j refers to the global unknown u_{j-1} and u_2^j refers to the global unknown u_j. Finally, the global expansion functions, ϕ_j are transformed into local expansion functions so that the interpolation of the solution u within element j is:

$$u^j(\xi) = u_1^j h_1(\xi) + u_2^j h_2(\xi) \tag{11.60}$$

where the functions $h_{1,2}$ are the local Lagrangian functions

$$h_1(\xi) = \frac{1-\xi}{2}, \quad h_2(\xi) = \frac{1+\xi}{2}, \tag{11.61}$$

It is easy to show that h_1 should be identified with the right limb of the global function ϕ_{j-1} while h_2 should be identified with the left limb of global function $\phi_j(x)$.

The operations that must be carried out in the computational space include differentiation and integration. The differentiation in physical space is evaluated with the help of the chain rule:

$$\frac{\partial u^j}{\partial x} = \frac{\partial u^j}{\partial \xi}\frac{\partial \xi}{\partial x} = \frac{\partial u^j}{\partial \xi}\frac{2}{\Delta x_j} \tag{11.62}$$

where $\frac{\partial \xi}{\partial x}$ is the metric of mapping element j from physical space to computational space. For the linear mapping used here this metric is constant. The derivative of the function in computational space is obtained from differentiating the interpolation formula 11.60:

$$\frac{\partial u^j}{\partial \xi} = u_1^j \frac{\partial h_1}{\partial \xi} + u_2^j \frac{\partial h_2}{\partial \xi} \tag{11.63}$$

$$= \frac{u_2^j - u_1^j}{2} \tag{11.64}$$

For the linear interpolation functions used in this example, the derivative is constant throught the element.

We know introduce the local stiffness matrices which are the contributions of the local element integration to the global stiffness matrix:

$$K_{m,n}^j = \int_{x_{j-1}}^{x_j} \left(\frac{\partial h_m}{\partial x}\frac{\partial h_n}{\partial x} + \lambda h_m(\xi) h_n(\xi) \right) \,\mathrm{d}x, \; (m,n) = 1,2 \tag{11.65}$$

Notice that the local stiffness matrix has a small dimension, 2×2 for the linear interpolation function, and is symmetric. We evaluate these integrals in computational space by breaking them up into 2 pieces $D^j_{m,n}$ and $M^j_{m,n}$ defined as follows:

$$D^j_{m,n} = \int_{x_{j-1}}^{x_j} \frac{\partial h_m}{\partial x}\frac{\partial h_n}{\partial x}\,\mathrm{d}x = \int_{-1}^{1} \frac{\partial h_m}{\partial \xi}\frac{\partial h_n}{\partial \xi}\left(\frac{\partial \xi}{\partial x}\right)^2 \frac{\partial x}{\partial \xi}\ \mathrm{d}\xi \tag{11.66}$$

$$M^j_{m,n} = \int_{x_{j-1}}^{x_j} h_m h_n\,\mathrm{d}x = \int_{-1}^{1} h_m h_n \frac{\partial x}{\partial \xi}\ \mathrm{d}\xi \tag{11.67}$$

The integrals in physical space have been replaced with integrals in computational space in which the metric of the mapping appears. For the linear mapping and interpolation function, these integrals can be easily evaluated:

$$M^j = \frac{\Delta x_j}{2}\int_{-1}^{1}\begin{pmatrix} (1-\xi)^2 & (1-\xi^2) \\ (1-\xi^2) & (1+\xi)^2 \end{pmatrix} \mathrm{d}\xi = \frac{\Delta x_j}{6}\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \tag{11.68}$$

Similarly, the matrix D^j can be shown to be:

$$D^j = \frac{1}{2\Delta x_j}\int_{-1}^{1}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \mathrm{d}\xi = \frac{1}{\Delta x_j}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \tag{11.69}$$

The local stiffness matrix is $K^j = D^j + \lambda M^j$. The matrix M^j appears frequently in FEM, it is usually identified with a time-derivative term (absent here), and is referred to as the **mass matrix**.

Having derived expressions for the local stiffness matrix, what remains is to map them into the global stiffness matrix. The following relationships hold between the global stiffness matrix and the local stiffness matrices:

$$K_{j,j-1} = K^j_{2,j} \tag{11.70}$$

$$K_{j,j} = K^j_{2,2} + K^{j+1}_{1,1} \tag{11.71}$$

$$K_{j,j+1} = K^{j+1}_{1,2} \tag{11.72}$$

The left hand sides in the above equations are the global entries while those on the right hand sides are the local entries. The process of adding up the local contribution is called the **stiffness assembly**. Note that some global entries require contributions from different elements.

In practical computer codes, the assembly is effected most efficiently by keeping track of the map between the local and global numbers in an array: `imap(2,j)` where `imap(1,j)` gives the global node number of local node 1 in element j, and `imap(2,j)` gives the global node number of local node 2 in element j. For the one-dimensional case a simple straightforward implementation is shown in the first loop of figure 11.5 where P stands for the number of collocation points within each element. For linear interpolation, $P = 2$. The scatter, gather and assembly operations between local and global nodes can now be easily coded as shown in the second, and third loops of figure 11.5.

11.2.8 Quadratic Interpolation

With the local approach to stiffness assembly, it is now simple to define more complicated local approximations. Here we explore the possibility of using quadratic interpolation to

```
integer, parameter :: N=10    ! number of elements
integer, parameter :: P=3     ! number of nodes per element
integer :: Nt=N*(P-1)+1       ! total number of nodes
integer :: imap(P,N)          ! connectivity
real*8  :: ul(P,N),vl(P,N)    ! local variables
real*8  :: u(Nt), v(Nt)       ! global variables
real*8  :: Kl(P,P,N)          ! local stiffness matrix
real*8  :: K(Nt,Nt)           ! global stiffness matrix

!         Assign Connectivity in 1D
do j = 1,N                    ! element loop
  do m = 1,P                  ! loop over collocation points
    imap(m,j) = (j-1)*(P-1)+m ! assign global node numbers
  enddo
enddo
!         Gather/Scatter operation
do j = 1,N                    ! element loop
  do m = 1,P                  ! loop over local node numbers
    mg = imap(m,j)            ! global node number of node m in element j
    ul(m,j) = u(mg)           ! local gathering operation
    v(mg) = vl(m,j)           ! local scattering
  enddo
enddo
!         Assembly operation
 K(1:Nt,1:Nt) = 0             ! global stiffness matrix
 do j = 1,N                   ! element loop
   do n = 1,P
     ng = imap(n,j)           ! global node number of local node n
     do m = 1,P
       mg = imap(m,j)         ! global node number of local node m
       K(mg,ng) = K(mg,ng) + Kl(m,n,j)
     enddo
   enddo
 enddo
```

Figure 11.5: Gather, scatter and stiffness assembly codes.

improve our solution. The local interpolation takes the form

$$\begin{aligned} u^j(\xi) &= u_1^j h_1(\xi) + u_2^j h_2(\xi) + u_3^j h_3(\xi) && (11.73)\\ h_1(\xi) &= -\xi\frac{1-\xi}{2} && (11.74)\\ h_2(\xi) &= 1-\xi^2 && (11.75)\\ h_3(\xi) &= \xi\frac{1+\xi}{2} && (11.76)\end{aligned}$$

It is easy to verify that $h_i(\xi)$ are Lagrangian interpolants at the collocation points $\xi_i = -1, 0, 1$. These functions are shown in top right panel of figure 11.6. Notice that there are now 3 degrees of freedom per elements, and that the interpolation function associated with the interior node does not interact with the interpolation functions defined in other elements. The local matrices can be evaluated analytically:

$$M^j = \frac{\Delta x_j}{30}\begin{pmatrix} 4 & 2 & -1\\ 2 & 16 & 2\\ -1 & 2 & 4\end{pmatrix}, \quad D^j = \frac{1}{3\Delta x_j}\begin{pmatrix} 7 & -8 & 1\\ -8 & 16 & -8\\ 1 & -8 & 7\end{pmatrix}, \qquad (11.77)$$

The assembly procedure can now be done as before with the proviso that the local node numbers m runs from 1 to 3. In the present instance the global system of equation is pentadiagonal and is more expensive to invert then the tridiagonal system obtained with the linear interpolation functions. One would expect improved accuracy, however.

11.2.9 Spectral Interpolation

Generalizing the approach to higher order polynomial interpolation is possible. As the degree of the interpolating polynomial increases, however, the well-known Runge phenomenon rears its ugly head. This phenomenon manifests itself in oscillations near the edge of the interpolation interval. This can be cured by a more judicial choice of the collocation points. This is the approach followed by the spectral element method, where the polynomial interpolation is still cast in terms of high order Lagrangian interpolation polynomials but the collocation points are chosen to be the Gauss-Lobatto roots of special polynomials. The most common polynomials used are the Legendre polynomials since their Gauss-Lobatto roots possess excellent interpolation and quadrature properties. The Legendre spectral interpolation takes the form

$$\begin{aligned} u^j(\xi) &= \sum_{m=1}^{P} u_m^j h_m(\xi) && (11.78)\\ h_m(\xi) &= \frac{-(1-\xi^2)L'_{P-1}(\xi)}{P(P-1)L_{P-1}(\xi_m)(\xi-\xi_m)} = \prod_{n=1,n\neq m}^{P}\frac{\xi-\xi_n}{\xi_m-\xi_n}, \quad m = 1,2,\ldots,P && (11.79)\end{aligned}$$

L_{P-1} denotes the Legendre polynomial of degree $(P-1)$ and L'_{P-1} denotes its derivative. The P collocation points ξ_n are the P Gauss-Lobatto roots of the Legendre polynomials of degree $P-1$, i.e. they are the roots of the equation:

$$(1-\xi_n^2)L'_{P-1}(\xi_n) = 0, \qquad (11.80)$$

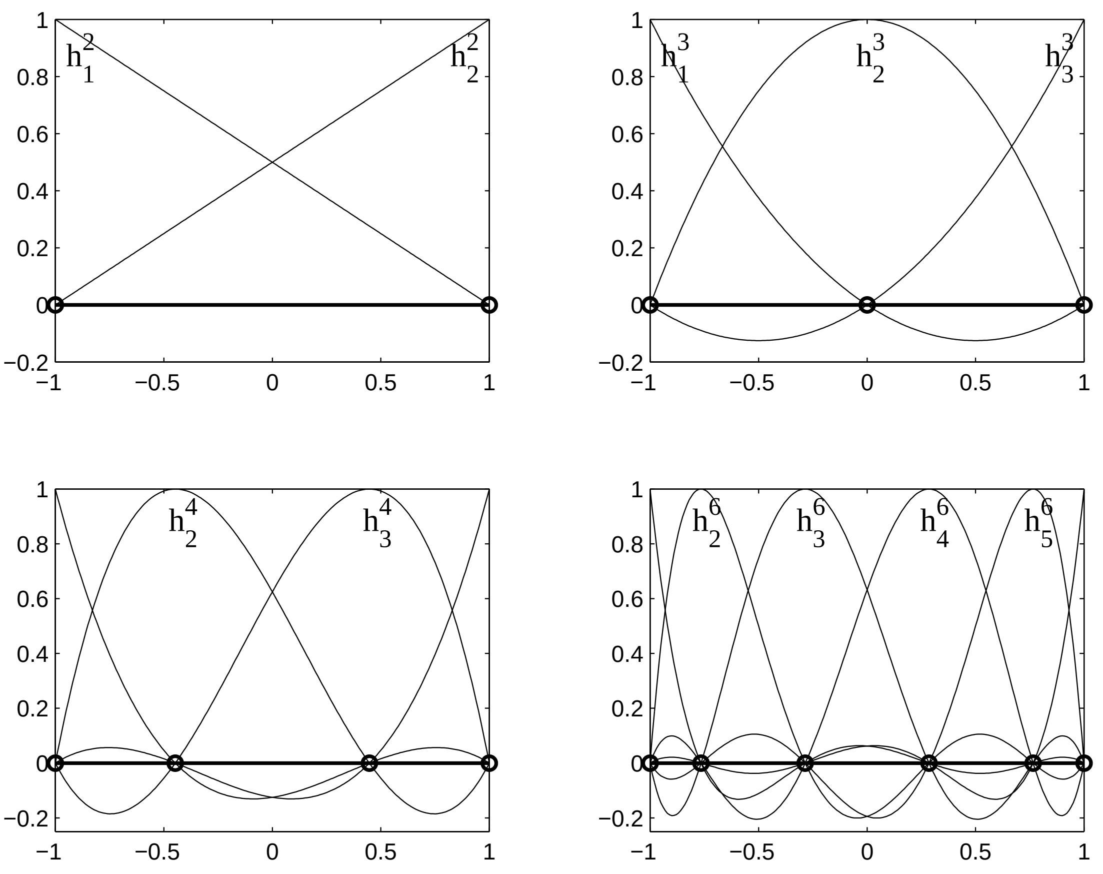


Figure 11.6: Plot of the Lagragian interpolants for different polynomial degrees: linear (top left), quadratic (top right), cubic (bottom left), and fifth (bottom right). The collocation points are shown by circles, and are located at the Gauss-Lobatto roots of the Legendre polynomial. The superscript indicates the total number of collocation points, and the subscript the collocation point with which the polynomial interpolant is associated.

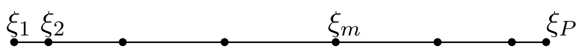


Figure 11.7: Distribution of collocation points in a spectral element. In this example there are 8 collocation points (polynomial of degree 7).

and are shown in figure 11.7. Equation 11.79 shows the two different forms in we can express the Lagragian interpolant; the traditional $\prod$ notation expresses h_m as a product of $P-1$ factors chosen so as to guarantee the Lagragian property; the second form is particular to the choice of Legendre Gauss-Lobatto points Boyd (1989); Canuto et al. (1988). It is easy to show that $h_m(\xi_n) = \delta_{mn}$, and they are polynomials of degree $P-1$. Note that unlike the previous cases the collocation points are not equally spaced within each element but tend to cluster more near the boundary of the element. Actually the collocation spacing is $O(1/(P-1)^2)$ near the boundary and $O(1/(P-1))$ near the center of the element. These functions are shown in figure 11.6 for $P = 4$ and 6. The $P-2$ internal points are localized to a single element and do not interact with the interpolation function of neighboring elements; the edge interpolation polynomials have support in two neighboring elements.

The evaluation of the derivative of the solution at specified points η_n is equivalent to:

$$u'(\eta_n) = \sum_{m=1}^{P} h'_m(\eta_n) u_m \tag{11.81}$$

and can be cast in the form of a matrix vector product, where the matrix entries are the derivative of the interpolation polynomials at the specified points η_n.

The only problem arises from the more complicated form of the integration formula. For this reason, it is common to evaluate the integrals numerically using high order Gaussian quadrature; see section 11.2.10. Once the local matrices are computed the assembly procedure can be performed with the local node numbering m running from 1 to P.

11.2.10 Numerical Integration

Although it is possible to evaluate the integrals analytically for each interpolation polynomial, the task becomes complicated and error prone. Furthermore, the presence of variable coefficients in the equations may complicate the integrands and raise their order. The problem becomes compounded in multi-dimensional problems. It is thus customary to revert to numerical integration to fill in the entries of the different matrices.

Gauss quadrature estimates the definite integral of a function with the weighed sum of the function evaluated at specified points called **quadrature points**:

$$\int_{-1}^{1} g(\xi)\ \mathrm{d}\xi = \sum_{p=1}^{Q} g(\xi_p^G)\omega_p + R_Q \tag{11.82}$$

where Q is the order of the quadrature formula and ξ_p^G are the Gauss quadrature points; the superscript is meant to *distinguish* the quadrature points from the collocation points. R_Q is the remainder of approximating the integral with a weighed sum; it is usually proportional to a high order derivative of the function g.

Gauss Quadrature

The Gauss quadrature is one of the most common quadrature formula used. Its quadrature points are given by the roots of the Q^{th} degree Legendre polynomial; its weights and remainder are:

$$\omega_p^G = \frac{2}{(1-\xi_p^2)[L'_Q(\xi_p^G)]^2},\ p = 1, 2, \ldots, Q \tag{11.83}$$

$$R_Q = \frac{2^{2Q+1}(Q!)^4}{(2Q+1)[(2Q)!]^3} \left.\frac{\partial^{2Q} g}{\partial \xi^{2Q}}\right|_{\xi},\ |\xi| < 1 \tag{11.84}$$

Gauss quadrature omits the end points of the interval $\xi = \pm 1$ and considers only interior points. Notice that if the integrand is a polynomial of degree $2Q-1$; its $2Q$-derivative is zero and the remainder vanishes identically. Thus a Q point Gauss quadrature integrates all polynomials of degree **less then** $2Q$ exactly.

Gauss-Lobatto Quadrature

The Gauss-Lobatto quadrature formula include the end points in their estimate of the integral. The roots, weights, and remainder of a Q-order Gauss-Lobatto quadrature are:

$$\left[1 - \left(\xi_p^{GL}\right)^2\right] L'_{Q-1}(\xi_p^{GL}) = 0 \tag{11.85}$$

$$\omega_p^{GL} = \frac{2}{(1-\xi_p^2)[L'_Q(\xi_p)]^2},\ p = 1, 2, \ldots, Q \tag{11.86}$$

$$R_Q = \frac{-Q(Q-1)^3 2^{2Q-1}[(Q-2)!]^4}{(2Q-1)[(2Q-2)!]^3} \left.\frac{\partial^{2Q-2} g}{\partial \xi^{2Q}}\right|_{\xi},\ |\xi| < \tag{11.87}$$

A Q point Gauss-Lobatto quadrature of order Q integrates exactly polynomials of degree less or equal to $2Q-3$.

Quadrature order and FEM

The most frequently encountered integrals are those associated with the mass matrix, equation 11.67, and diffusion matrix, equation 11.66. We will illustrate how the order of integration can be determined to estimate these integrals accurately. We assume for the time being that the interpolation polynomial h_m is of degree $P-1$ (there are P collocation points within each element), and the metric of the mapping is constant. The integrand in equation 11.66 is of degree $2(P-2)$, and the in equation 11.67 is of degree $2(P-1)$.

If Gauss quadrature is used and exact integration is desired then the order of the quadrature must be $Q > P-2$ in order to evaluate 11.66 exactly and $Q > P$ in order to evaluate 11.67 exactly. Usually a single quadrature rule is chosen to effect all integrations needed. In this case $Q = P+1$ will be sufficient to evaluate the matrices M and D in equations 11.67-11.66. Higher quadrature with $Q > P+1$ may be required if additional terms are included in the integrand; for example when the metric of the mapping is not constant. Exact evaluation of the integrals using Gauss-Lobatto quadrature requires more points since it is less accurate then Gauss quadrature: $Q \geq (2P+3)/2$ Gauss-Lobatto points are needed to compute the mass matrix exactly.

Although the order of Gauss quadrature is higher, it is not always the optimal choice; other considerations may favor Gauss-Lobatto quadrature and reduced (inexact) integration. Consider a quadrature rule where the collocation and quadrature points are identical, such a rule is possible if one chooses the Gauss-Lobatto quadrature of order P, where P is the number of points in each element; then $\xi_m = \xi_m^{GL}$ for $m = 1, \ldots, P$. The evaluation of the local mass matrix becomes:

$$\begin{aligned}
M^j_{m,n} &= \int_{-1}^{1} h_m(\xi) h_n(\xi) \frac{\partial x}{\partial \xi}\, \mathrm{d}\xi && (11.88)\\
&\approx \sum_{p=1}^{P} \left[h_m(\xi_p) h_m(\xi_p) \left.\frac{\partial x}{\partial \xi}\right|_{\xi_p} \omega_p \right] && (11.89)\\
&= \sum_{p=1}^{P} \left[\delta_{m,p}\delta_{n,p} \left.\frac{\partial x}{\partial \xi}\right|_{\xi_p} \omega_p \right] && (11.90)\\
&= \delta_{m,n} \left.\frac{\partial x}{\partial \xi}\right|_{\xi_m} \omega_m && (11.91)
\end{aligned}$$

Equation 11.91 shows that mass matrix becomes diagonal when the quadrature and collocation points are identical. This rule applies equatlly well had we chosen the Gauss points for quadrature and collocation. However, the Gauss-Lobatto roots are preferred since they simplify the imposition of C^0 continuity across elements (there are formulation where C^0 continuity is not necessary, Gauss quadrature and collocation becomes sensible). The implication of a diagonal mass matrix is profound for it simplifies considerably the computations of time-dependent problems. As we will see later, the time-integration requires the inversion of the mass matrix, and this task is infinitely easier when the mass matrix is diagonal. The process of reducing the mass matrix to diagonal is occasionally referred to as **mass lumping**. One should be carefull when low order finite elements are used to build the mass matrix as the reduced quadrature introduces error. For low order interpolation (linear and quadratic) mass lumping reduces the accuracy of the finite element method substantially; the impact is less pronounced for higher order interpolation; the rule of thumb is that mass lumping is terrible for linear element and has little impact for $P > 3$.

Example 14 We solve the 1D problem: $u_{xx} - 4u = 0$ in $0 \leq x \leq 1$ subject to the boundary conditions $u(0) = 0$, and $u_x = 1$. The analytical solution is $u = \sinh 2x/(2\cosh 2)$. The rms error between the finite element solution and the analytical solution is shown in figures 11.8 as a function of the number of degrees of freedom. The plots show the error for the linear (red) and quadratic (blue) interpolation. The left panel shows a semi logarithmic plot to highlight the exponential convergence property of the spectral element method, while the right panel shows a log-log plot of the same quantities to show the algebraic decrease of the error as a function of resolution for the linear and quadratic interpolation as evidenced by the straight convergence lines. The slopes of the convergence curves for the spectral element method keeps increasing as the number of degrees of freedom is increased. This decrease is most pronounced when the degrees of freedom as added as increased spectral interpolation within each element as opposed to increasing the number of elements. Note that the spectral element computations used

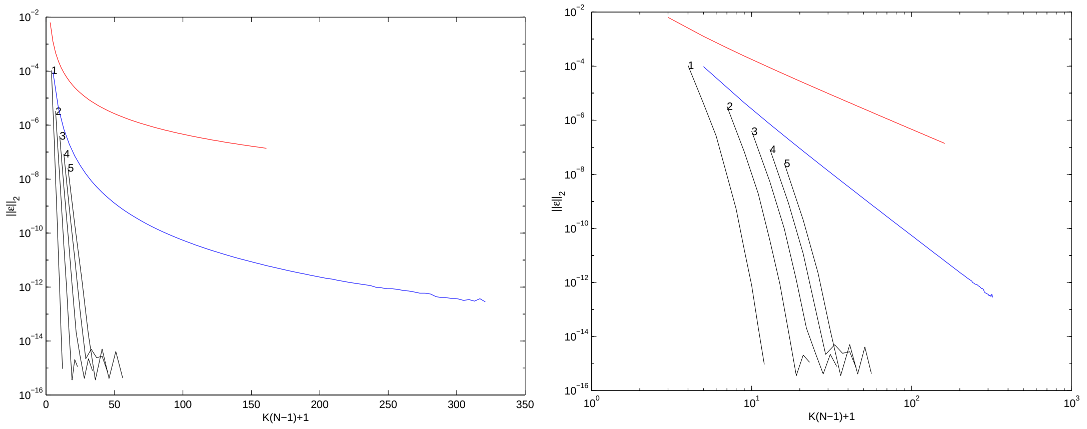


Figure 11.8: Convergence curves for the finite element solution of $u_{xx} - 4u = 0$. The red and blue lines show the solutions obtained using linear and quadratic interpolation, respectively, using exact integration; the black lines show the spectral element solution.

Gauss-Lobatto quadrature to evaluate the integrals, whereas exact integration was used for linear and quadratic finite elements. The inexact quadrature does not destroy the spectral character of the method.

11.3 Mathematical Results

The following mathematical results are presented without proof given the substantial mathematical sophistication in their derivation.

11.3.1 Uniqueness and Existence of continuous solution

The existence and uniqueness of the solution to the weak form is guaranteed by the Lax-Milgram theorem:

Theorem 1 *Lax-Milgram Let V be a Hilbert space with norm $\| \; \|_V$, consider the bilinear form $\mathcal{A}(u,v) : V \times V \longrightarrow R$, and the bounded linear functional $\mathcal{F}(v) : V \longrightarrow R$. If the bilinear form is bounded and coercive,i.e. there exists positive constants ρ and β such that*

- *continuity of $\mathcal{A}$:* $|\mathcal{A}(u,v)| \leq \beta \|u\|_V \, \|v\|_V \quad \forall u,v \in V$
- *coercivity of $\mathcal{A}$:* $\rho \|u\|_V \leq \mathcal{A}(u,u) \quad \forall u \in V$

Then there exists a unique $\hat{u} \in V$ such that

$$\mathcal{A}(\hat{u},v) = \mathcal{F}(v) \quad \forall v \in V \tag{11.92}$$

The continuity condition guarantees that the operator $\mathcal{A}$ is bounded: $|\mathcal{A}(u,v)| \leq \beta \|u\|^2{}_V$. This, in combination with the coercivity condition guarantees that $\mathcal{A}$ is norm equivalent to $\|\|_V$. The above theorem guarantees the existence and uniqueness of the continuous solution. We take the issue of the discrete solution in the following section.

11.3.2 Uniqueness and Existence of continuous solution

The infinite-dimensional continuous solution $\hat{u}$, must be approximated with a finite dimensional approximation u_N where N characterizes the dimensionality (number of degrees of freedom) of the discrete solution. Let $V_N \subset V$ be a finite dimensional subspace of V providing a dense coverage of V so that in the limit $N \to \infty$, $\lim_{N\to\infty} V_N = V$.

Since V_N is a subset of V the condition of the Lax-Milgram theorem are fullfilled and hence a unique solution exists for the discrete problem. The case where the discrete space $V_N \subset V$ (V_N is a subset of V) is called the **conforming** case. The proofs of existence and uniqueness follow from the simple fact that V_N is a subset of V. Additionally, for the Galerkin approximation the following **stability** condition is satisfied:

$$\|u_N\|_V \leq C\|f\| \tag{11.93}$$

where C is a positive constant independent of N. One has moreover that:

$$\|u_n - \hat{u}\|_V \leq \frac{\beta}{\rho} \inf_{v\in V_N} \|\hat{u} - v\|\|_V \tag{11.94}$$

The inequality (11.93) shows that the V-norm of the numerical solution is bounded by the L_2-norm of the forcing function f (the data). This is essentially the stability criterion of the Lax-Richtmeyer theorem. Inequality (11.94) says that the V-norm of the error is bounded by the smallest error possible $v \in V_N$ to describe $\hat{u} \in V$. By making further assumptions about the smoothness of the solution $\hat{u}$ it is possible to devise error estimates in terms of the size of the elements h. The above estimate guarantees that the left hand side of (11.94) goes to zero as $N \to \infty$ since $V_N \to V$. Hence the numerical solution **converges** to the true solution. According to the Lax-Richtmeyer equivalence theorem, since two conditions (stability and convergence) hold, the third condition (consistency) must follow; the discretization is hence also **consistent**.

11.3.3 Error estimates

Inequality (11.94) provide the mean to bound the error in the numerical solution. Let $\hat{u}_I = I_N\hat{u}$ be the interpolant of $\hat{u}$ in V_N. An upper bound on the approximation error can be obtained since

$$\|u_n - \hat{u}\|_V \leq \frac{\beta}{\rho} \inf_{v\in V_N} \|v - \hat{u}\|\|_V \leq C\|\hat{u}_I - \hat{u}\|V \tag{11.95}$$

Let h reflect the charateristic size of an element (for a uniform discretization in 1D, this would be equivalent to Δx). One expects $\|u_n - \hat{u}\|_V \to 0$ as $h \to 0$. The rate at which that occurs depends on the smothness of the solution $\hat{u}$.

For linear intepolation, we have the following estimate:

$$\|u_n - \hat{u}\|_{H^1} \leq C\, h\, |\hat{u}|_{H^2}, \quad |\hat{u}|_{H^2} = \left(\int_0^1 u_{xx} \mathrm{d}x\right)^{\frac{1}{2}} \tag{11.96}$$

where $|\hat{u}|_{H^2}$ is the so-called H^2 semi-norm, (essentially a measure of the "size" of the second derivative), and C is a generic positive constant independent of h. If the solution admits integrabale second derivatives, then the H^1-norm of the error decays linearly with the grid size h. The L_2-norm of the error however decreases quadratically according to:

$$\|u_n - \hat{u}\|_{H^1} \le \tilde{C}\, h^2\, |\hat{u}|_{H^2} |\hat{u}|_{H^2} = \left(\int_0^1 (u_{xx})^2 \mathrm{d}x\right)^{\frac{1}{2}} \tag{11.97}$$

The difference between the rate of convergences of the two error norms is due to the fact that the H^1-norm takes the derivatives of the function into account. That the first derivative of $\hat{u}$ are approximated to first order in h while $\hat{u}$ itself is approximated to second order in h.

For an interpolation using polynomials of degree k the L_2-norm of the error is given by

$$\|u_n - \hat{u}\| \le \hat{C}\, h^{k+1}\, |\hat{u}|_{H^{k+1}} \qquad |\hat{u}|_{H^{k+1}} = \left[\int_0^1 \left(\frac{\mathrm{d}^{k+1}u}{\mathrm{d}x^{k+1}}\right)^2 \mathrm{d}x\right]^{\frac{1}{2}} \tag{11.98}$$

provided that the solution is smooth enough i.e. the $k+1$ derivative of the solution is square-integrable.

For the spectral element approximation using a single element, the error depends on N and the regularity of the solution. If $\hat{u} \in H^m$ with $m > 0$, then the approximation error in the L_2 norm is bounded by

$$\|u_n - \hat{u}\| \le C\, N^{-m}\, \|\hat{u}\|_{H^m} \tag{11.99}$$

The essential difference between the p-version of the finite element method and the spectral element method lies in the exponent of the error (note that $h\ N^{-1}$). In the p-case the exponent of N is limited by the degree of the interpolating polynomial. In the spectral element case it is limited by the smoothness of the solution. If the latter is infinitely smooth then $m \approx N$ and hence the decay is exponential N^{-N}.

11.4 Two Dimensional Problems

The extension of the finite element methods to two-dimensional elliptic problems is straightforward and follows the same lines as for the one-dimensional examples, namely: transformation of the PDE to a variational problem, restriction of this problem to a finite dimensional space (effectively, the Galerkin step), discretization of the geometry and spatial interpolation, assembly of the stiffness matrix, imposition of boundary conditions, and finally solution of the linear system of equations. Two dimensional problems have more complicated geometries then one-dimensional problems. The issue of geometry discretization becomes important and this will be explored in the present section. We will take as our sample PDE the following problem:

$$\nabla^2 u - \lambda u + f = 0,\ \mathbf{x} \in \Omega \tag{11.100}$$

$$u(\mathbf{x}) = u_b(\mathbf{x}),\ \mathbf{x} \in \Gamma_D \tag{11.101}$$

$$\nabla u \cdot \mathbf{n} = q,\ \mathbf{x} \in \Gamma_N \tag{11.102}$$

where Ω is the region of interest with $\partial\Omega$ being its boundary, Γ_D is that portion of $\partial\Omega$ where Dirichlet conditions are imposed and Γ_N are those portions where Neumann conditions are imposed. We suppose the $\Gamma_D + \Gamma_N = \partial\Omega$.

The variational statements comes from multiplying the PDE by test functions $v(\mathbf{x})$ that satisfy $v(\mathbf{x} \in \Gamma_D) = 0$, integrating over the domain Ω, and applying the boundary conditions; the variational statement boils down to:

$$\int_\Omega (\nabla u \cdot \nabla v + \lambda uv)\ \mathrm{d}A = \int_\Omega fv\ \mathrm{d}A\ +\ \int_{\Gamma_N} vq\ \mathrm{d}s,\ \forall v \in H_0^1 \tag{11.103}$$

where H_0^1 is the space of square integrable functions on Ω that satisfy homogenous boundary conditions on Γ_D; $\mathrm{d}s$ is the arclength along the boundary. The Galerkin formulation comes from restricting the test functions to a finite set and interpolation functions to a finite set:

$$u = \sum_{i=1}^{N} u_i \phi_i(\mathbf{x}),\ \ v = \phi_j, j = 1, 2, \dots, N \tag{11.104}$$

where the $\phi_i(\mathbf{x})$ are now two dimensional interpolation functions (here we restrict ourselves to Lagrangian interpolants). The Galerkin formulations becomes find u_i such that

$$\sum_{i=1}^{N} \int_\Omega (\nabla\phi_i \cdot \nabla\phi_i + \lambda\phi_i\phi_j)\ \mathrm{d}A u_i = \int_\Omega f\phi_j\ \mathrm{d}A\ +\ \int_{\Gamma_N} \phi_j q\ \mathrm{d}s,\ j = 1, 2, \dots, N \tag{11.105}$$

We thus recover the matrix formulation $Ku = b$ where the stiffness matrix and forcing functions are given by:

$$K_{ji} = \int_\Omega (\nabla\phi_i \cdot \nabla\phi_i + \lambda\phi_i\phi_j)\ \mathrm{d}A \tag{11.106}$$

$$b_j = \int_\Omega f\phi_j\ \mathrm{d}A\ +\ \int_{\Gamma_N} \phi_j q\ \mathrm{d}s \tag{11.107}$$

In two space dimensions we have a greater choice of element topology (shape) then in the simplistic 1D case. Triangular elements, the simplex elements of 2D space, are very common since they have great flexibility, and allow the discretization of very complicated domains. In addition to triangular elements, quadrilaterals elements with either straight or curved edges are extensively used. In the following, we explore the interpolation functions for each of these elements, and the practical issues needed to be overcome in order to compute the integrals of the Galerkin formulation.

11.4.1 Linear Triangular Elements

One of the necessary ingredient of FE computations is the localization of the computations to an element which requires the development of a natural coordinate system in which to perform the computations. For the triangle, the natural coordinate system is the area coordinate shown in figure 11.9. The element is identified by the three nodes i, j and k forming its vertices; let P be a point inside the triangle with coordinates $\mathbf{x}$. By connecting P to the three vertices i, j, and k, we can divide the elements into three

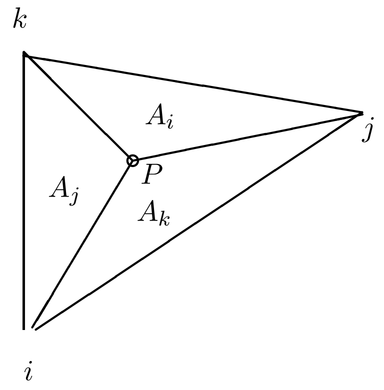


Figure 11.9: Natural area coordinate system for triangular elements.

small triangles with areas A_i, A_j and A_k, respectively with $A_i + A_j + A_k = A$, where A is the area of the original element. Notice that if the point P is located along edge $j - k$, $A_i = 0$, whereas if its located at i we have $A_i = A$ and $A_j = A_k = 0$; the nodes j and k have similar property. This natural division of the element into 3 parts of similar structure allows us to define the area coordinate of point P as:

$$a_i = \frac{A_i}{A},\ a_j = \frac{A_j}{A},\ a_k = \frac{A_k}{A},\ a_i + a_j + a_k = 1 \tag{11.108}$$

The area of a triangle is given by the determinant of the following matrix:

$$A = \frac{1}{2}\begin{vmatrix} 1 & x_i & y_i \\ 1 & x_j & y_j \\ 1 & x_k & y_k \end{vmatrix} \tag{11.109}$$

The other areas A_i, A_j and A_k can be obtained similarly; their dependence on the coordinate (x, y) of the point P is linear. It is now easy to verify that if we set the local interpolation functions to $\phi_i = a_i$ we obtain the linear Lagrangian interpolant on the triangle, i.e. that $\phi_i(\mathbf{x}_m) = \delta_{i,m}$ where m can tak the value i, j, or k. The linear Lagrangian interpolant for point i can be easily expressed in terms of the global coordinate system (x, y):

$$\phi_i(x, y) = \frac{1}{2A}(\alpha_i + \beta_i x + \gamma_i y) \tag{11.110}$$

$$\alpha_i = x_j y_k - x_k y_j \tag{11.111}$$

$$\beta_i = y_j - y_k \tag{11.112}$$

$$\gamma_i = -(x_j - x_k) \tag{11.113}$$

The other interpolation function can be obtained with a simple permutation of indices. Note that the derivative of the linear interpolation functions are constant over the element. The linear interpolation now takes the simple form:

$$u(\mathbf{x}) = u_i\phi_i(\mathbf{x}) + u_j\phi_j(\mathbf{x}) + u_k\phi_k(\mathbf{x}) \tag{11.114}$$

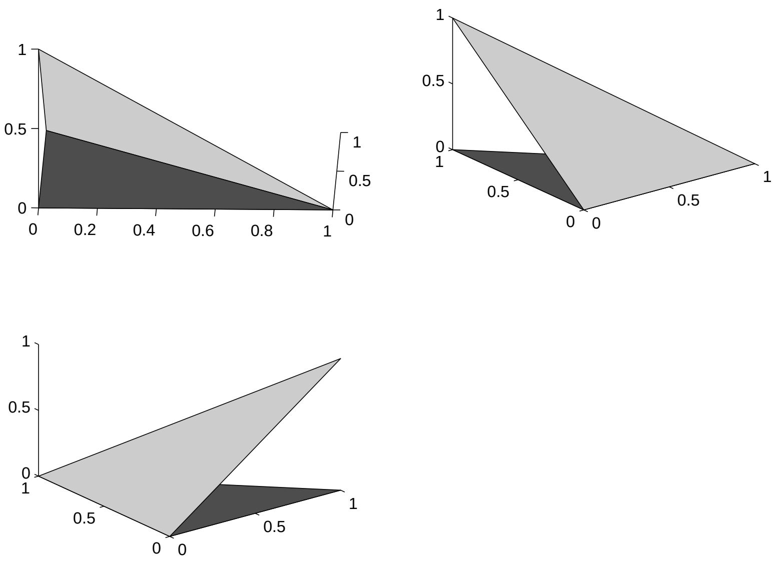


Figure 11.10: Linear interpolation function over linear triangular elements. The triangle is shown in dark shades. The three interpolation functions are shown in a light shade and appear as inclined planes in the plot.

where u_i, u_j and u_k are the values of the solution at the nodes i, j and k. The interpolation formula for the triangle guarantees the continuity of the solution across element boundaries. Indeed, on edge $j-k$ for example, the interpolation does not involve node i and is essentially a linear combination using the functions values at node j and k, thus ensuring continuity. The linear interpolation function for the triangle are shown in figure 11.10 as a three-dimensional plot.

The usefullness of the area coordinates stems from the existence of the following integration formula over the triangle:

$$\int_A a_i^p a_j^q a_k^r \, \mathrm{d}A = 2A \frac{p!q!r!}{(a+b+c+2)!} \tag{11.115}$$

where the notation $p! = 12\ldots p$ stands for the factorial of the integer p. It is now easy to verify that the *local* mass matrix is now given by

$$M^e = \int_A \begin{pmatrix} \phi_i\phi_i & \phi_i\phi_j & \phi_i\phi_k \\ \phi_j\phi_i & \phi_j\phi_j & \phi_j\phi_k \\ \phi_k\phi_i & \phi_k\phi_j & \phi_k\phi_k \end{pmatrix} \mathrm{d}A = \frac{A}{12}\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix} \tag{11.116}$$

The entries of the matrix arising from the discretization of the Laplace operator are easy to compute since the gradients of the interpolation and test functions are constant over an element; thus we have:

$$D^e \quad = \quad \int_A \begin{pmatrix} \nabla\phi_i\cdot\nabla\phi_i & \nabla\phi_i\cdot\nabla\phi_j & \nabla\phi_i\cdot\nabla\phi_k \\ \nabla\phi_j\cdot\nabla\phi_i & \nabla\phi_j\cdot\nabla\phi_j & \nabla\phi_j\cdot\nabla\phi_k \\ \nabla\phi_k\cdot\nabla\phi_i & \nabla\phi_k\cdot\nabla\phi_j & \nabla\phi_k\cdot\nabla\phi_k \end{pmatrix} \mathrm{d}A \tag{11.117}$$

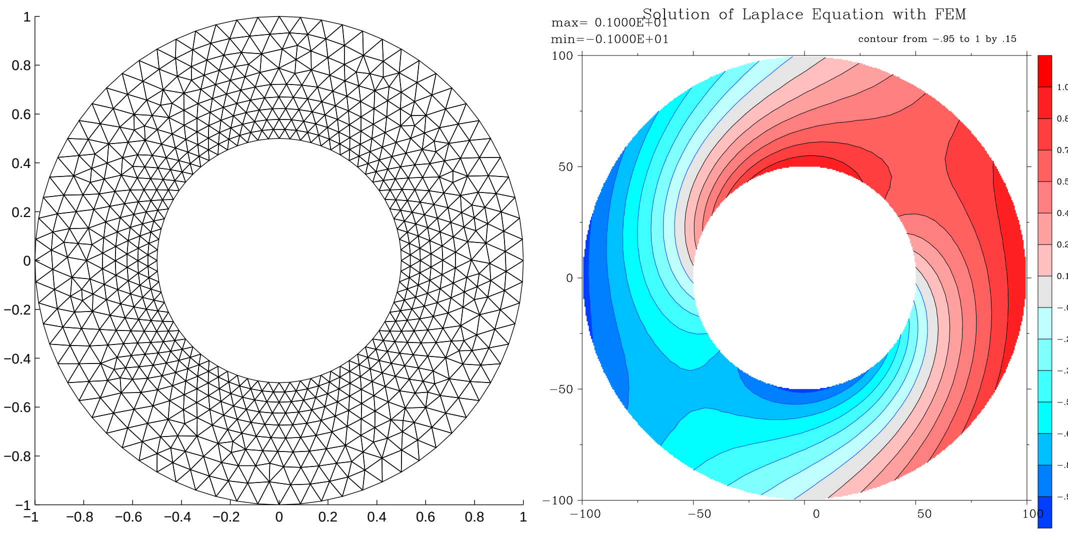


Figure 11.11: FEM grid and contours of the solution to the Laplace equation in a circular annulus.

$$= \frac{1}{4A}\begin{pmatrix} \beta_i\beta_i + \gamma_i\gamma_i & \beta_i\beta_j + \gamma_i\gamma_j & \beta_i\beta_k + \gamma_i\gamma_k \\ \beta_j\beta_i + \gamma_j\gamma_i & \beta_j\beta_j + \gamma_j\gamma_j & \beta_j\beta_k + \gamma_j\gamma_k \\ \beta_k\beta_i + \gamma_k\gamma_i & \beta_k\beta_j + \gamma_k\gamma_j & \beta_k\beta_k + \gamma_k\gamma_k \end{pmatrix} \tag{11.118}$$

As an example of the application of the FEM element method we solve the Laplace equation in a circular annulus subject to Dirichlet boundary conditions using a FEM with linear triangular elements. The FEM grid and contours of the solution are shown in figure 11.11. The grid contains 754 nodes and 1364 elements. The boundary conditions were set to $\cos\theta$ on the outer radius and $\sin\theta$ on the inner radius, where θ is the azimuthal angle. The inner and outer radii are 1/2 and 1, respectively. The contour lines were obtained by interpolating the FEM solution to a high resolution (401x401) strutured grid prior to contouring it.

11.4.2 Higher order triangular elements

It is possible to define higher order interpolation formula within an element without changing the shape of an element. For example the quadratic triangular elements with collocation points at the triangle vertices and mid-points of edges are given by

$$\begin{aligned} u(\mathbf{x}) &= u_i\phi_i^2(\mathbf{x}) + u_j\phi_j^2(\mathbf{x}) + u_k\phi_k^2(\mathbf{x}) \\ &+ u_{i-j}\phi_{i-j}^2(\mathbf{x}) + u_{j-k}\phi_{j-k}^2(\mathbf{x}) + u_{k-i}\phi_{k-i}^2(\mathbf{x}) && (11.119) \\ \phi_i^2 &= a_i(a_i - 1) && (11.120) \\ \phi_j^2 &= a_j(a_j - 1) && (11.121) \\ \phi_k^2 &= a_k(a_k - 1) && (11.122) \\ \phi_{i-j}^2 &= 4a_ia_j && (11.123) \end{aligned}$$

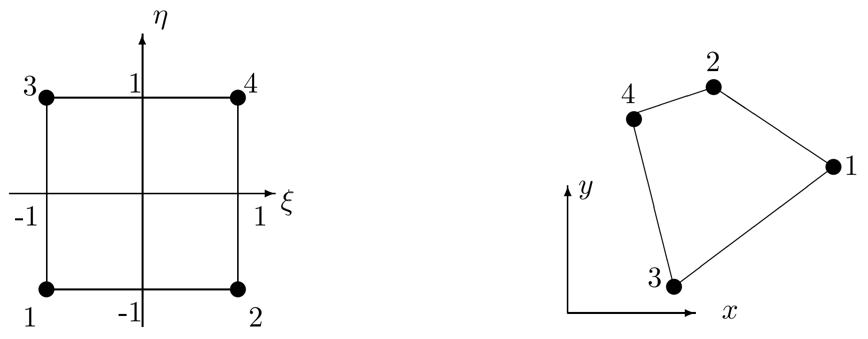


Figure 11.12: Mapping of a quadrilateral between the unit square in computational space (left) and physical space (right).

$$\phi^2_{j-k} = 4a_j a_k \tag{11.124}$$

$$\phi^2_{k-i} = 4a_k a_i \tag{11.125}$$

where $i-k$ denotes the midpoint of edge $i-k$. There are 6 degrees of freedom associated with each quadratic triangular element. Although it is possible to define even higher order interpolation in the triangle by proceeding as before; it is not so simple to implement. The alternative is to use a mapping to a structured computational space and define the interpolation and collocation function in that space. It is important to choose the collocation points appropriately in order to avoid Gibbs oscillations as the polynomial degree increases. Spectral triangular elements are the optimal choice with this regard. We will not discuss spectral element triangles here; we refer the interested reader to Karniadakis and Sherwin (1999).

11.4.3 Quadrilateral elements

Change of Coordinates

The derivation of the interpolation and integration formula for quadrilateral finite elements follows the line of the previous section. The main task resides in defining the local coordinate system in the master element shown in figure 11.12. For straight edged elements the mapping between physical and computational space can be easily effected through the following bilinear map:

$$\mathbf{x}(\xi,\eta) = \frac{1-\xi}{2}\left(\frac{1-\eta}{2}\mathbf{x}_1 + \frac{1+\eta}{2}\mathbf{x}_3\right) + \frac{1+\xi}{2}\left(\frac{1-\eta}{2}\mathbf{x}_2 + \frac{1+\eta}{2}\mathbf{x}_4\right) \tag{11.126}$$

In order to derive all the expressions needed to express the Galerkin integrals in computational space, it is useful to introduce a basis in computational space $(\mathbf{e}_\xi, \mathbf{e}_\eta)$ tangential to the coordinate lines (ξ, η); we denote by (x, y) the coordinate in physical space. Let $\mathbf{r} = x\mathbf{i} + y\mathbf{j}$ denotes a vector pointing to point P located inside the element. We define the basis vectors vectors tangent to the coordinate lines as

$$\mathbf{e}_\xi = \frac{\partial \mathbf{r}}{\partial \xi} = x_\xi \mathbf{i} + y_\xi \mathbf{j}, \quad \mathbf{e}_\eta = \frac{\partial \mathbf{r}}{\partial \eta} = x_\eta \mathbf{i} + y_\eta \mathbf{j} \tag{11.127}$$

where $\mathbf{r}$ denotes the position vector of a point P in space and $(\mathbf{i},\mathbf{j})$ forms an othonormal basis in the physical space. Inverting the above relationship one obtains

$$\mathbf{i} = \frac{y_\eta}{J}\mathbf{e}_\xi - \frac{y_\xi}{J}\mathbf{e}_\eta, \quad \mathbf{j} = -\frac{x_\eta}{J}\mathbf{e}_\xi + \frac{x_\xi}{J}\mathbf{e}_\eta$$

where $J = x_\xi y_\eta - x_\eta y_\xi$ is the Jacobian of the mapping. The norms of $\mathbf{e}_\xi$ and $\mathbf{e}_\eta$ are given by

$$|\mathbf{e}_\xi|^2 = \mathbf{e}_\xi \cdot \mathbf{e}_\xi = (x_\xi)^2 + (y_\xi)^2, \quad |\mathbf{e}_\eta|^2 = \mathbf{e}_\eta \cdot \mathbf{e}_\eta = (x_\eta)^2 + (y_\eta)^2$$

The basis in the computational plane is orthogonal if $\mathbf{e}_\xi \cdot \mathbf{e}_\eta = x_\xi x_\eta + y_\xi y_\eta = 0$; in general the basis is not orthogonal unless the element is rectangular.

It is now possible to derive expression for length and area segements in computational space. These are needed in order to compute boundary and area integrals arising from the Galerkin formulation. Using the definition $(\,\mathrm{d}s)^2 = \,\mathrm{d}\mathbf{r}\cdot\,\mathrm{d}\mathbf{r}$ with $\mathrm{d}\mathbf{r} = \mathbf{r}_\xi\,\mathrm{d}\xi + \mathbf{r}_\eta\,\mathrm{d}\eta$, we have:

$$(\,\mathrm{d}s)^2 = |\mathbf{e}_\xi\,\mathrm{d}\xi + \mathbf{e}_\eta\,\mathrm{d}\eta|^2 = |\mathbf{e}_\xi|^2\,\mathrm{d}\xi^2 + |\mathbf{e}_\eta|^2\,\mathrm{d}\eta^2 + 2\mathbf{e}_\xi\cdot\mathbf{e}_\eta\,\mathrm{d}\xi\,\mathrm{d}\eta \tag{11.128}$$

The differential area of a curved surface is defined as the area of its tangent plane approximation (in 2D, the area is always flat.) The area of the parallelogram defined by the vectors $\mathrm{d}\xi\mathbf{e}_\xi$ and $\mathrm{d}\eta\mathbf{e}_\eta$ is

$$\mathrm{d}A = ||\,\mathrm{d}\xi\mathbf{e}_\xi \times\,\mathrm{d}\eta\mathbf{e}_\eta|| = \left\|\begin{array}{ccc} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_\xi & y_\xi & 0 \\ x_\eta & y_\eta & 0 \end{array}\right\| \mathrm{d}\xi\,\mathrm{d}\eta = |x_\xi y_\eta - x_\eta y_\xi|\,\mathrm{d}\xi\,\mathrm{d}\eta = |J|\,\mathrm{d}\xi\,\mathrm{d}\eta \tag{11.129}$$

after using the definition of $(\mathbf{e}_\xi, \mathbf{e}_\eta)$ in terms of $(\mathbf{i},\mathbf{j})$.

Since $x = x(\xi,\eta)$ and $y = y(\xi,\eta)$, the derivative in physical space can be expressed in terms of derivatives in computational space by using the chain rule of differentiation; in matrix form this can be expressed as:

$$\begin{pmatrix} u_x \\ u_y \end{pmatrix} = \begin{pmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{pmatrix} \begin{pmatrix} u_\xi \\ u_\eta \end{pmatrix} \tag{11.130}$$

Notice that the chain rule involves the derivatives of ξ,η in terms of x,y whereas the bilinear map readily delivers the derivatives of x,y with respect to ξ,η. In order to avoid inverting the mapping from physical to computational space we derive expressions for $\nabla\xi$, and $\nabla\eta$ in terms of x_ξ, x_η, etc... Applying the chain rule to x and y we obtain (noticing that the two variables are independent) the system of equations:

$$\begin{pmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{pmatrix} \begin{pmatrix} x_\xi & y_\xi \\ x_\eta & y_\eta \end{pmatrix} = \begin{pmatrix} x_x & y_x \\ x_y & y_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{11.131}$$

The solution is

$$\begin{pmatrix} \xi_x & \eta_x \\ \xi_y & \eta_y \end{pmatrix} = \frac{1}{J}\begin{pmatrix} y_\eta & -y_\xi \\ -x_\eta & x_\xi \end{pmatrix}, \quad J = x_\xi y_\eta - x_\eta y_\xi \tag{11.132}$$

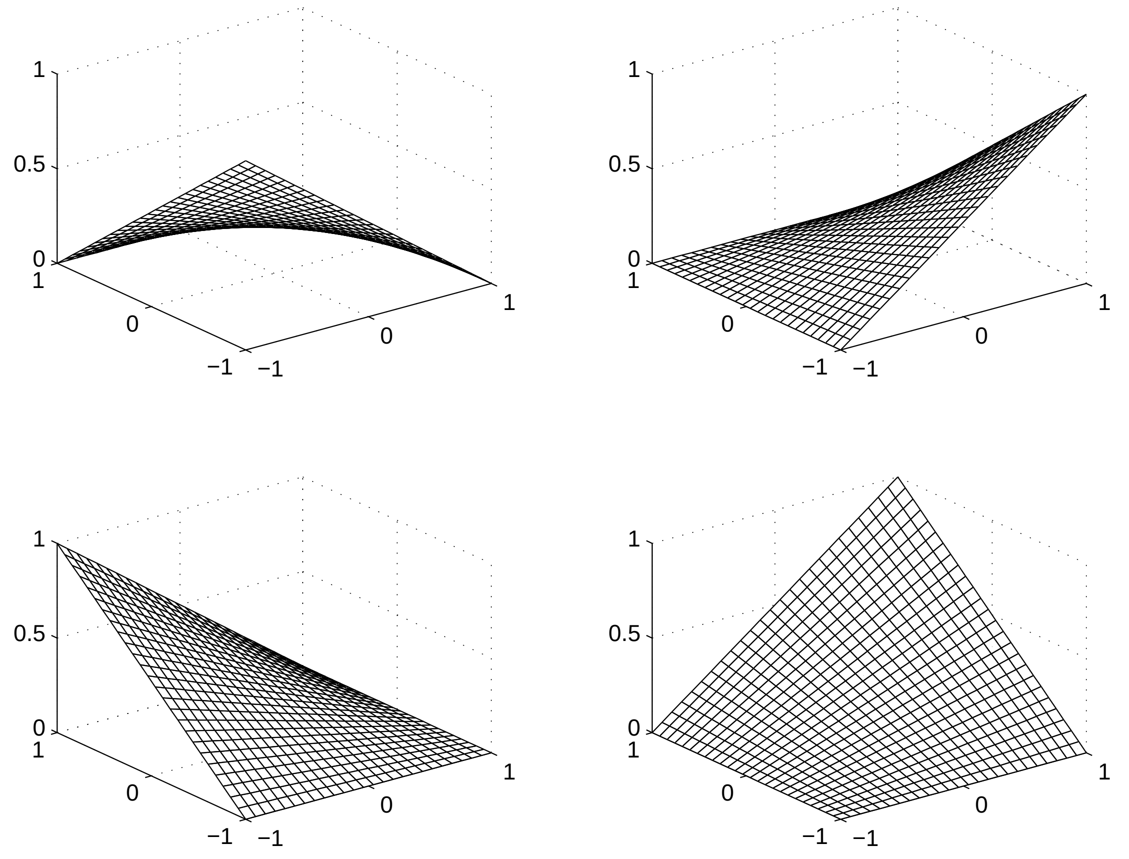


Figure 11.13: Bilinear shape functions in quadrilateral elements, the upper left hand corner is $h_1(\xi)h_1(\eta)$, the upper right hand side panel shows $h_2(\xi)h_1(\eta)$, the lower left panel shows $h_1(\xi)h_2(\eta)$ and lower right shows $h_2(\xi)h_1(\eta)$.

For the bilinear map of equation 11.126, the metrics and Jacobian can be easily computed by differentiation:

$$x_\xi = \frac{1-\eta}{2}\frac{x_2-x_1}{2} + \frac{1+\eta}{2}\frac{x_4-x_1}{2} \qquad y_\xi = \frac{1-\eta}{2}\frac{y_2-y_1}{2} + \frac{1+\eta}{2}\frac{y_4-y_1}{2} \tag{11.133}$$

$$x_\eta = \frac{1-\xi}{2}\frac{x_3-x_1}{2} + \frac{1+\xi}{2}\frac{x_4-x_2}{2} \qquad y_\eta = \frac{1-\xi}{2}\frac{y_3-y_1}{2} + \frac{1+\xi}{2}\frac{y_4-y_2}{2} \tag{11.134}$$

The remaining expression can be obtained simply by plugging in the the various expressions derived earliear.

11.4.4 Interpolation in quadrilateral elements

The interpolation of the solution within quadrilateral elements is easily accomplished if tensorized product of one-dimensional Lagrangian interpolants are used to build the two-dimensional formula. For the bilinear map shown in figure 11.12, for example we can use the collocation points located at the vertices of the quadrilateral to define the following interpolation:

$$u(\xi,\eta) = \sum_{m=1}^{4} u_m \phi_m(\xi,\eta) = u_1\phi_1 + u_2\phi_2 + u_3\phi_3 + u_4\phi_4 \tag{11.135}$$

where the two-dimensional Lagrangian interpolants are tensorized product of the one-dimensional interpolants defined in equation 11.61:

$$\begin{array}{ll} \phi_1(\xi,\eta) = h_1(\xi)h_1(\eta), & \phi_2(\xi,\eta) = h_2(\xi)h_1(\eta), \\ \phi_3(\xi,\eta) = h_1(\xi)h_2(\eta), & \phi_4(\xi,\eta) = h_2(\xi)h_2(\eta), \end{array} \tag{11.136}$$

and shown in figure 11.13. Note that the bilinear interpolation function above satisfy the C^0 continuity requirement. This can be easily verified by noting first that the interpolation along an edge involves only the collocation points along that edge, hence neighboring elements sharing an edge will interpolate the solution identically if the value of the function on the collocation points is unique. Another important feature of the bilinear interpolation is that, unlike the linear interpolation in triangular element, it contains the term of second degree: $\xi\eta$. Hence the interpolation within an element is non-linear; it is only linear along edges.

Before proceeding further, we introduce a new notation for interpolation with quadrilaterals to explicitely bring out the tensorized form of the interpolation functions. This is accomplished by breaking the single two-dimensional index m in equation 11.135 into two one-dimensional indices (i,j) such that $m = (j-1)2 + i$, where $(i,j) = 1,2$. The index i runs along the ξ direction and the index j along the η direction; thus $m = 1$ becomes identified with $(i,j) = (1,1)$, $m = 2$ with $(i,j) = (2,1)$, etc... The interpolation formula can now be written as

$$u(\xi,\eta) = \sum_{j=1}^{2}\sum_{i=1}^{2} u_{ij} h_i(\xi) h_j(\eta) \tag{11.137}$$

where u_{ij} are the function values at point (i,j).

With this notation in hand, it is now possible to write down arbitrarily high order Lagrangian interpolation in 2D using the tensor product formula. Thus, a 1-D interpolation using N points per 1D element can be extended to 2D via:

$$u(\xi,\eta) = \sum_{j=1}^{N}\sum_{i=1}^{N} u_{ij} h_i^N(\xi) h_j^N(\eta) \tag{11.138}$$

The superscript N has been introduced on the Lagragian interpolants to stress that they are polynomials of degree $N-1$ and use N collocation points per direction. The collocation points using 7^{th} degree interpolation polynomials is shown in figure 11.14.

Note, finally, that **sum factorization** algorithms must be used to compute various quantities on structured sets of points p,q in order to reduce the computational overhead. For example, the derivative of the function at point (p,q), can be computed as:

$$u_\xi|_{\xi_p,\eta_q} = \sum_{j=1}^{N}\left(\sum_{i=1}^{N} u_{ij} \left.\frac{\mathrm{d}h_i^N}{\mathrm{d}\xi}\right|_{\xi_p}\right) h_j^N(\eta_q). \tag{11.139}$$

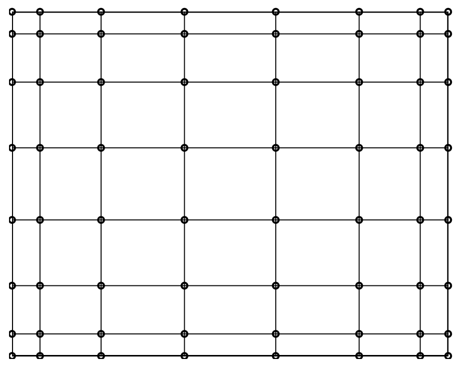


Figure 11.14: Collocation points within a spectral element using 8 collocation points per direction to interpolate the solution; there are 8^2=64 points in total per element.

First the term in parenthesis is computed and saved in a temporary array, second, the final expression is computed and saved; this essentially reduces the operation from $O(N^4)$ to $O(N^3)$. Further reduction in operation count can be obtained under special circumstances. For instance, if η_q happens to be a collocation point, then $h_j^N(\eta_q) = \delta_{jq}$ and the formula reduces to a single sum:

$$u_\xi|_{\xi_p,\eta_q} = \sum_{i=1}^{N} u_{iq} \left.\frac{\mathrm{d}h_i^N}{\mathrm{d}\xi}\right|_{\xi_p} \tag{11.140}$$

11.4.5 Evaluation of integrals

The integrals needed to build the stiffness matrix for two dimensional quadrilateral elements are a bit more complicated then those encountered in triangular elements. This is primarily due to the lack of magic integration formula, and the more complex (non-constant) mapping between physical and computational space. We start by considering the calculation of the elemental mass matrix $M_{m,n} = \int_A \phi_m \phi_n \, \mathrm{d}A$; in computational space and using the 2-index notation introduced earlier, this integral becomes:

$$M^e_{ij,kl} = \int_{-1}^{1}\int_{-1}^{1} h_i(\xi)\, h_j(\eta)\, h_k(\xi)\, h_l(\eta)\, |J| \;\; \mathrm{d}\xi \;\; \mathrm{d}\eta \tag{11.141}$$

where $m = (j-1)P + i$, and $n = (k-1)P + l$. In order to bypass the tediousness of evaluating integrals analytically for each term that may be present in the Galerkin formulation, it is common practice to evaluate the integrals with Gauss quadrature. The order of the quadrature needed depends of course on the polynomial degree of the integrand and on whether exact integration is required.

We now proceed to determine the quadrature order needed to integrate the mass matrix exactly. In the case of the bilinear map of equation 11.126, the Jacobian varies bilinearly over the element, hence it is linear in the variable ξ and η. Assuming the Lagrange interpolation uses P points in each direction, the integrand in the mass matrix is a polynomial of degree $2(P-1)+1$ in each of the variables ξ and η. Thus, Gauss quadrature of order Q will evaluate the integrals exactly provided $Q \geq P$:

$$M_{ij,kl} = \sum_{m=1}^{Q} \sum_{n=1}^{Q} h_i(\xi_m^G)\, h_j(\eta_n^G)\, h_k(\xi_m^G)\, h_l(\eta_n^G)\, |J_{mn}| \omega_m^G \omega_n^G \tag{11.142}$$

where J_{mn} denotes the Jacobian at the Gauss quadrature points (ξ_m^G, η_n^G), and ω_m^G are the Gauss quadrature weights. The only required operations are hence the evaluation of the Lagrangian interpolants at the points ξ_m^G, and summations of the terms in 11.142. If Gauss-Lobatto integration is used, the roots and weights become those appropriate for the Gauss-Lobatto quadrature; the number of quadrature points needed to evaluate the integral exactly increases to $Q \geq P + 1$.

Like the one-dimensional case, the mass matrix can be rendered diagonal if inexact (but accurate enough) integration is acceptable. If the quadrature points and collocation points coincide, $h_k(\xi_m^G) = \delta_{im}$, and the expression for the mass matrix entries reduces to:

$$M_{ij,kl} = \delta_{ik} \delta_{jl} \omega_k \omega_l |J_{kl}| \tag{11.143}$$

The integrals involved in evaluating the discrete Laplace operator, the matrix D, is substantially more complicated in the present instance. Here we continue with our reliance on Gauss quadrature and derive the terms that must be computed. Keeping with our 2 index notation, we have

$$D_{ij,kl} = \int_A \nabla \phi_{ij} \cdot \nabla \phi_{kl} \;\; \mathrm{d}A. \tag{11.144}$$

Most of the work comes from having to express the inner product in the integrand in computational space:

$$\begin{aligned}
\nabla \phi_{ij} \cdot \nabla \phi_{kl} &= \frac{\partial \phi_{ij}}{\partial x} \frac{\partial \phi_{kl}}{\partial x} + \frac{\partial \phi_{ij}}{\partial y} \frac{\partial \phi_{kl}}{\partial y} && (11.145) \\
&= \left(\frac{\partial \phi_{ij}}{\partial \xi} \nabla \xi + \frac{\partial \phi_{ij}}{\partial \eta} \nabla \eta \right) \cdot \left(\frac{\partial \phi_{kl}}{\partial \xi} \nabla \xi + \frac{\partial \phi_{kl}}{\partial \eta} \nabla \eta \right) && (11.146) \\
&= \frac{\partial \phi_{ij}}{\partial \xi} \frac{\partial \phi_{kl}}{\partial \xi} \nabla \xi \cdot \nabla \xi + \frac{\partial \phi_{ij}}{\partial \eta} \frac{\partial \phi_{kl}}{\partial \eta} \nabla \eta \cdot \nabla \eta \\
&+ \left(\frac{\partial \phi_{ij}}{\partial \xi} \frac{\partial \phi_{kl}}{\partial \eta} + \frac{\partial \phi_{ij}}{\partial \eta} \frac{\partial \phi_{kl}}{\partial \xi} \right) \nabla \xi \cdot \nabla \eta. && (11.147)
\end{aligned}$$

Setting $\phi_{ij} = h_i(\xi) h_j(\eta)$ and $\phi_{kl} = h_k(\xi) h_l(\eta)$, and evaluating the integrals using Gauss quadrature, we get:

$$\begin{aligned}
D_{ij,kl} &= \sum_{n=1}^{Q} \sum_{m=1}^{Q} h_i'(\xi_m)\, h_j(\eta_n)\, h_k'(\xi_m)\, h_l(\eta_n) \;\; [\nabla \xi \cdot \nabla \xi |J|]_{m,n} \omega_m \omega_n \\
&+ \sum_{n=1}^{Q} \sum_{m=1}^{Q} h_i(\xi_m)\, h_j'(\eta_n)\, h_k(\xi_m)\, h_l'(\eta_n) \;\; [\nabla \eta \cdot \nabla \eta |J|]_{m,n} \omega_m \omega_n \\
&+ \sum_{n=1}^{Q} \sum_{m=1}^{Q} h_i'(\xi_m)\, h_j(\eta_n)\, h_k(\xi_m)\, h_l'(\eta_n) \;\; [\nabla \xi \cdot \nabla \eta |J|]_{m,n} \omega_m \omega_n \\
&+ \sum_{n=1}^{Q} \sum_{m=1}^{Q} h_i(\xi_m)\, h_j'(\eta_n)\, h_k'(\xi_m)\, h_l(\eta_n) \;\; [\nabla \xi \cdot \nabla \eta |J|]_{m,n} \omega_m \omega_n \qquad (11.148)
\end{aligned}$$

where the expressions in bracket are evaluated at the quadrature points (ξ_m, η_n) (we have omitted the superscript G from the quadrature points). Again, substantial savings can be achieved if the Gauss-Lobatto quadrature of the same order as the inteporlation polynomial is used. The expression for $D_{ij,kl}$ reduces to:

$$\begin{aligned}
D_{ij,kl} &= \delta_{jl} \sum_{m=1}^{Q} h_i'(\xi_m)\, h_k'(\xi_m)\ \left[\nabla\xi \cdot \nabla\xi |J|\right]_{m,j} \omega_m \omega_j \\
&+ \delta_{ik} \sum_{n=1}^{Q} h_j'(\eta_n)\, h_l'(\eta_n)\ \left[\nabla\eta \cdot \nabla\eta |J|\right]_{i,n} \omega_i \omega_n \\
&+ h_i'(\xi_k)\, h_l'(\eta_j)\ \left[\nabla\xi \cdot \nabla\eta |J|\right]_{k,j} \omega_k \omega_j \\
&+ h_k'(\xi_i)\, h_j'(\eta_l)\ \left[\nabla\xi \cdot \nabla\eta |J|\right]_{i,l} \omega_i \omega_l
\end{aligned} \tag{11.149}$$

11.5 Time-dependent problem in 1D: the Advection Equation

Assume we attempt to solve the 1D constant-coefficient advection equation:

$$u_t + cu_x = 0, x \in \Omega \tag{11.150}$$

subject to appropriate initial and boundary conditions. The variational statement that solves the above problem is: Find u such that

$$\int_\Omega (u_t + cu_x) v dx = 0, \forall v \tag{11.151}$$

The Galerkin formulation reduces the problem to a finite dimensional space by replacing the solution with a finite expansion of the form

$$u = \sum_{i=1}^{N} u_i(t)\phi_i(x) \tag{11.152}$$

and setting the test functions to $v = \phi_j$, $j = 1, 2, \ldots, N$. Note that unlike the steady state problems encountered earlier u_i, the value of the function at the collocation points, depends on time. Replacing the finite expansion in the variational form we get the following ordinary differential equation (ODE)

$$M\frac{\mathrm{d}u}{\mathrm{d}t} + Cu = 0 \tag{11.153}$$

where u is the vector of solution values at the collocation points, M is the mass matrix, and C the matrix resulting from discretization of the advection term using the Galerkin procedure. The entries of M and C are given by:

$$M_{j,i} = \int_0^L \phi_i \phi_j \ \mathrm{d}x \tag{11.154}$$

$$C_{j,i} = \int_0^L c\frac{\partial \phi_i}{\partial x}\phi_j \ \mathrm{d}x \tag{11.155}$$

We follow the procedure outlined in section 11.2.7 to build the matrices M and C. We also start by looking at linear interpolation functions as those defined in equation 11.60. The local mass matrix is given in equation 11.68; here we derive expressions for the advection matrix C assuming the advective velocity c is constant. and advection matrices are

$$C_{ji} = \int_{x_{j-1}}^{x_j} h_j \frac{\mathrm{d}h_i}{\mathrm{d}x} dx = c\int_{-1}^{1} h_i(\xi)\frac{\mathrm{d}h_j(\xi)}{\mathrm{d}\xi} d\xi, \;\; C = \frac{c}{2}\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \tag{11.156}$$

After stiffness assembly, the global system of equations becomnes:

$$\frac{\Delta x}{6}\frac{\mathrm{d}}{\mathrm{d}t}\left(u_{j-1} + 4u_j + u_{j+1}\right) + \frac{c}{2}(-u_{j-1} + u_{j+1}) = 0 \tag{11.157}$$

The approximation of the advective term using linear finite element has resulted in a centered-difference approximation for that term; whereas the time-derivative term has produced the mass matrix. Notice that any integration scheme, *even an explicit ones* like leap-frog, would necessarily require the inversion of the mass matrix. Thus, one can already anticipate that the computational cost of solving the advection using FEM will be higher then a similarly configured explicit finite difference method. For linear elements in 1D, the mass matrix is tridiagonal and the increase in cost is minimal since tridiagonal solvers are very efficient. Quadratic elements in 1D lead to a pentadiagonal system and is hence costlier to solve. This increased cost may be justifiable if it is compensated by a sufficient increase in accuracy. In multi-dimensions, the mass matrix is not tridiagonal but has only limited bandwidth that depends on the global numbering of the nodes; thus even linear elements would require a full matrix inversion.

Many solutions have been proposed to tackle the extra-cost of the full matrix. One solution is to use reduced integration using Gauss-Lobatto quadrature which as we saw in the Gaussian quadrature section leads immediately to a diagonal matrix; this procedure is often referred to as mass lumping. For low order elements, mass lumping degrades significantly the accuracy of the finite element method, particularly in regards to its phase properties. For the 1D advection equation mass lumping of linear elements is equivalent to a centered difference approximation. For high order interpolation, i.e. higher then degree 3, the loss of accuracy due to the inexact quadrature is tolerable, and is of the same order of accuracy as the interpolation formula. Another alternative revolves around the use of discontinuous test functions and is appropriate for the solution of mostly hyperbolic equations; this approach is dubbed the Discontinuous Galerkin method, and will be examined in a following section.

The system of ODE can now be integrated using one of the time-stepping algorithms, for example second order leap-frog, third order Adams-Bashforth, or one of the Runge-Kutta schemes. For linear finite elements, the stability limit can be easily studied with the help of Von Neumann stability analysis. For example, it is easy to show that a leap-frog scheme applied to equation will result in a stability limit of the form $\mu = c\Delta t/\Delta x < 1/\sqrt{3}$; and hence is much more restrictive then the finite difference scheme which merely requires that the Courant number be less then 1. However, an examination of the phase property of the linear FE scheme will reveal its superiority over centered differences.

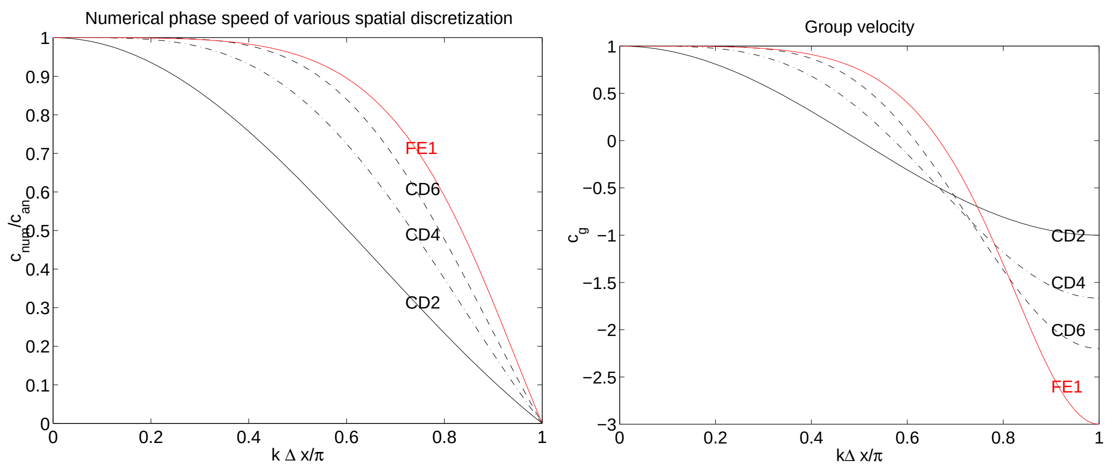


Figure 11.15: Comparison of the dispersive properties of linear finite elements with centered differences, the left panel shows the ratio of numerical phase speed to analytical phase speed, and the right panel shows the ratio of the group velocity

We study the phase properties implied by the linear finite element discretization by looking for the periodic solution, in space and time, of the system of equation 11.157: $u(x,t) = e^{i(kx_j - \sigma t)}$. We thus get the following dispersion relationship and phase speed:

$$\sigma = \frac{3c}{\Delta x}\frac{\sin k\Delta x}{2+\cos k\Delta x} \tag{11.158}$$

$$\frac{c_F}{c} = \frac{3\sin k\Delta x}{k\Delta x(2+\cos k\Delta x)} \tag{11.159}$$

The numerical phase speed should be contrasted to the one obtaiend from centered second and fourth order finite differences:

$$\frac{c_{CD2}}{c} = \frac{\sin k\Delta x}{k\Delta x} \tag{11.160}$$

$$\frac{c_{CD4}}{c} = \frac{1}{3}\left(4\frac{\sin k\Delta x}{k\Delta x} - \frac{\sin 2k\Delta x}{2k\Delta x}\right) \tag{11.161}$$

Figure 11.15 compares the dispersion of linear finite element with that of centered difference schemes of 2, 4 and 6 order. It is immediately apparent that the FE formulation yields more accurate phase speed at all wavenumbers, and that the linear interpolation is equivalent, if not slightly better then, a sixth-order centered FD approximation; in particular the intermediate to short waves travel slightly faster then in FD. The group velocity, shown in the right panel of figure 11.15, shows similar results for the long to intermediate waves. The group velocity of the short wave are, however, in serious errors for the FE formulation; in particular they have a negative phase speed and propagate upstream of the signal at a faster speed then the finite difference schemes. A mass-lumped version of the FE method would collapse the FE curve onto that of the centered second order method.

11.5.1 Numerical Example

As an example of the application of the FEM to the the inviscid 1D advection equation we solve the following problem:

$$u_t + u_x = 0 \text{ on } -1 \leq x \leq 1,\ u(-1,t) = 0,\ u(x,0) = e^{256\left(x+\frac{1}{2}\right)^2} \tag{11.162}$$

The initial condition consists of an infinitely smooth Gaussian hill centered at $x = -1/2$. The solution at time $t = 1$ should be the same Gaussian hill but centered at $x = 1/2$. The FEM solutions are shown in figure 11.16 where the consistent mass (exact integration) and "lumped" mass solutions are compared, for various interpolation orders; the number of elements was kept fixed and the convergence study can be considered an p-refinement strategy. The number of element was chosen so that the hill is barely resolved for linear elements (the case $m = 2$). It is obvious that the solution improves rapidly as m (the number of interpolation points within an element) is increased. The lumped mass solution is very poor indeed for the linear case where its poor dispersive properties have generated substantial oscillatory behavior. The situation improves substantially when the interpolation is increased to quadratic; the biggest discrepancy occuring near the upstream foot of the hill. The lumped and consistent mass solutions are indistinguishable for $m = 5$.

One may object that the improvements are solely due to the increased resolution. We have repeated the experiment above trying to keep the total number of degrees of freedom fixed; in this case we are trying to isolate the effects of improved interpolation order. The first case considered was an under-resolved case using 61 total degrees of freedom, and the comparison is shown in figure 11.17. The figure shows indeed the improvements in the dispersive characteristics of the lumped mass approximation as the degree of the polynomial is increased. The consistent and lumped solutions overlap over the main portion of the signal for $m \geq 4$ but differ over the (numerically) dispersed waves trailing the hill. These are evidence that the two approximations are still under-resolved.

The solution in the resolved case is shown in figure 11.18 where the total number of degrees of freedom is increased to 121. In this case the two solution overlap for $m \geq 3$. The two solutions are indistinguishable for $m = 5$ over the entire domain, evidence that the solution is now well-resolved. Notice that the dispersive error of the lumped mass in the linear interpolation case is still quite in error and further increase in the number of elements is required; these errors are entirely due to mass-lumping as the consitent mass solution is free of dispersive ripples.

The lesson of the previous example is that the dispersive properties of the lumped-mass solution is quite good provided enough resolution is used. The number of elements needed to reach the resolution threshold decreases dramatically as the polynomial order is increased. The dispersive properties of the lumped-mass linear interpolation FEM seem to be the worse, and seem to require a very-well resolved solution before the effect of the mass-lumping is eliminated. One then has to weigh the increased cost of a mass-lumped large calculation versus that of a smaller consistent-mass calculation to decided whether lumping is cost-effective or not.

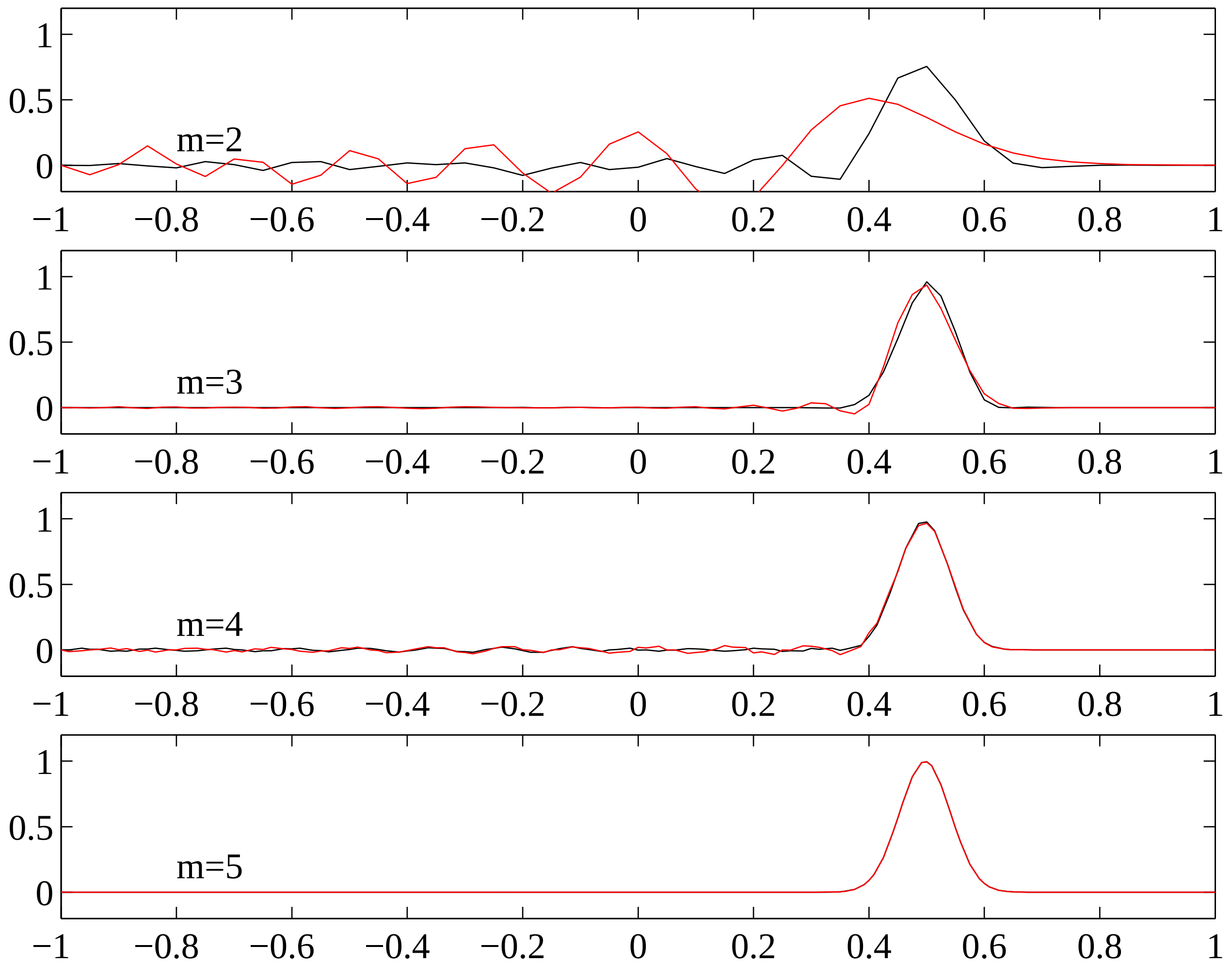


Figure 11.16: Solution of the advection equation with FEM. The black lines show the result of the consistent mass matrix calculation and the red lines show that of the "lumped" mass matrix calculation. The discretization consisted of 40 equally spaced elements on the interval [0 1], and a linear (m=2), quadratic (m=3), cubic (m=4) and quartic (m=5) interpolation. The time stepping is done with a TVD-RK3 scheme (no explicit dissipation included).

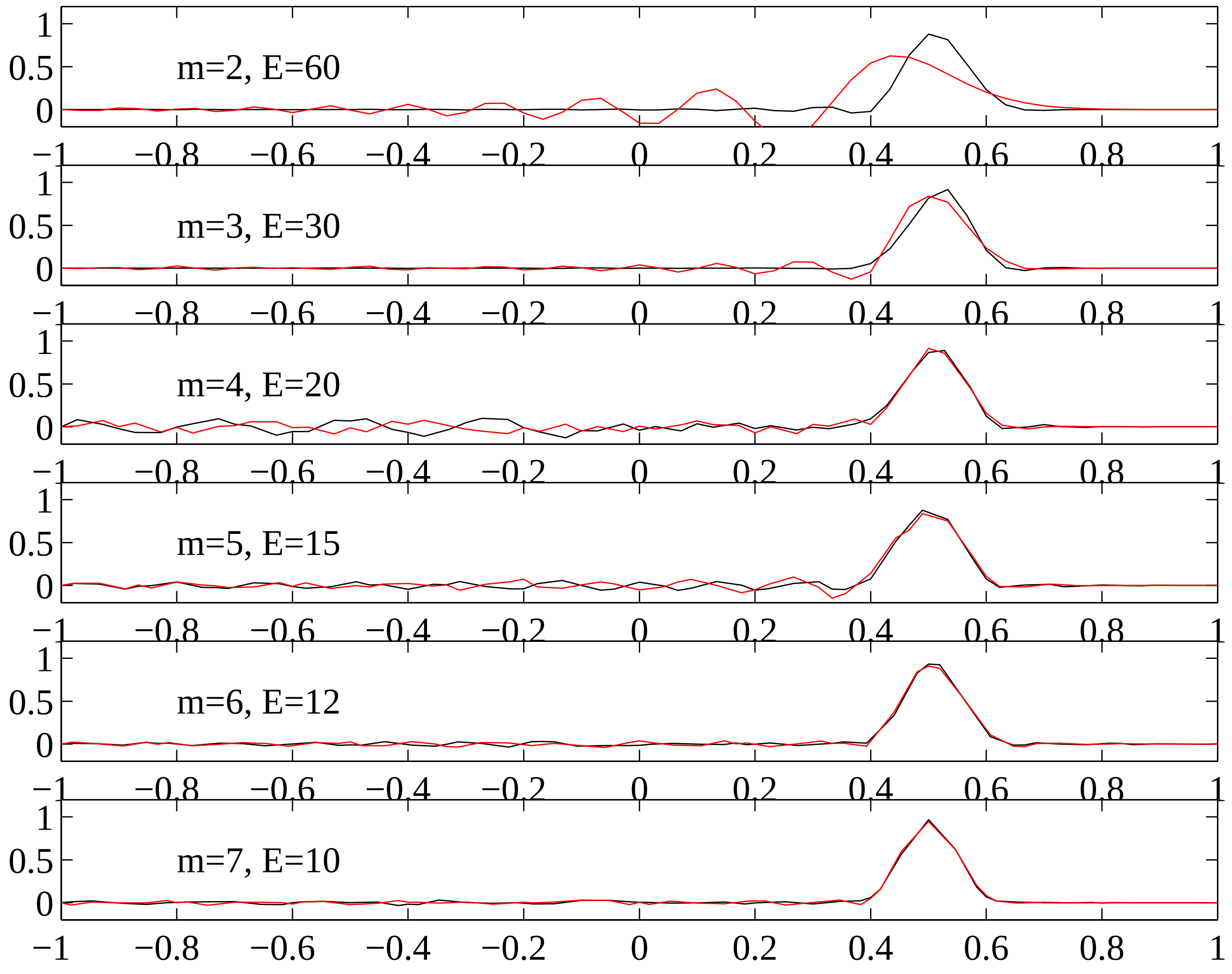


Figure 11.17: Consistent versus lumped solution of the advection equation with FEM for a coarse resolution with the total number of degrees of freedom fixed, $N \approx 61$. The black and red lines refer to the consistent and lumped mass solutions, respectively.

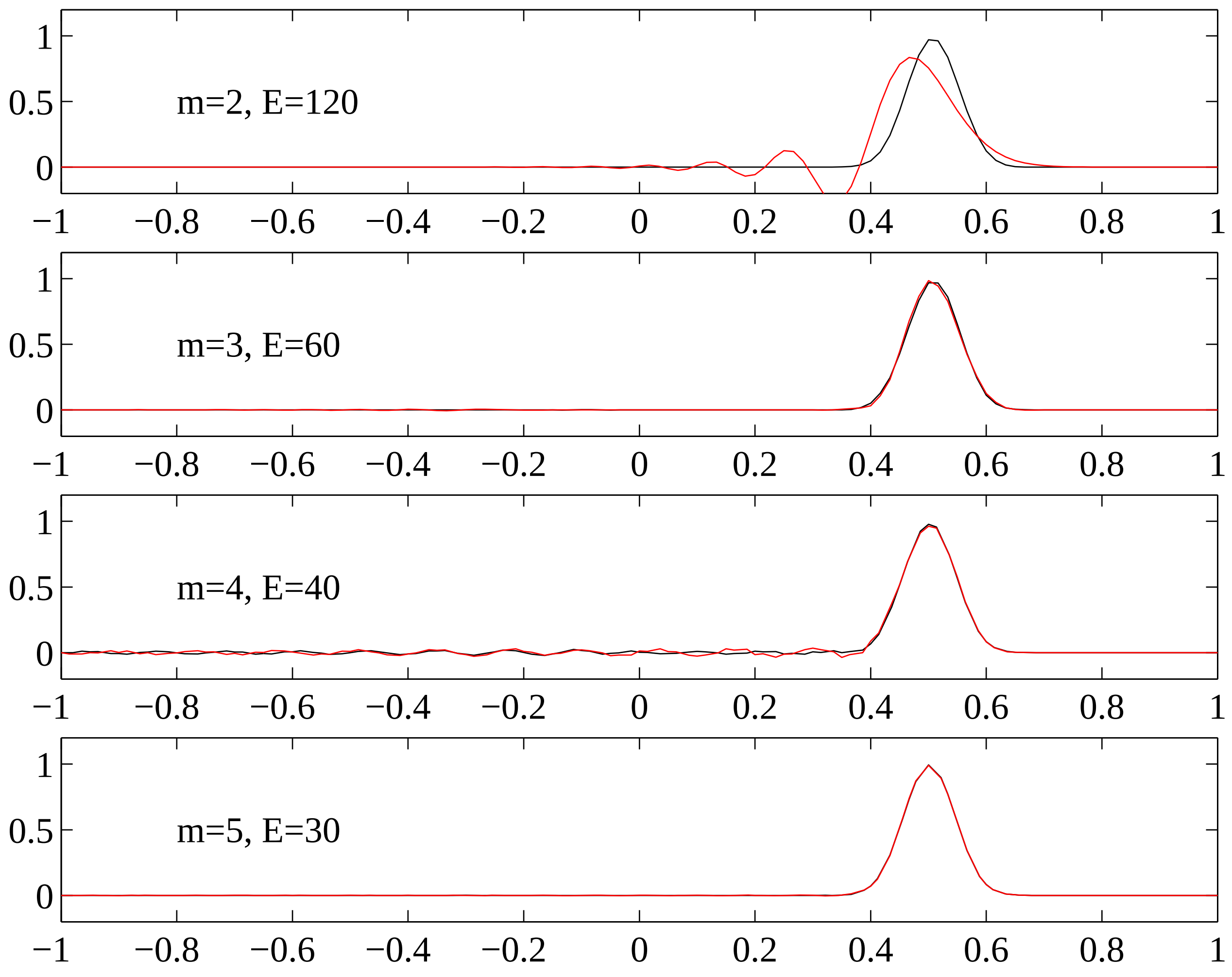


Figure 11.18: Consistent versus lumped solution of the advection equation with FEM for a coarse resolution with the total number of degrees of freedom fixed, $N \approx 121$. The black and red lines refer to the consistent and lumped mass solutions, respectively.

11.6 The Discontinuous Galerkin Method (DGM)

A major drawback of the continuous Galerkin method is the requirement to maintain C^0 continuity; a requirement that leads to a tight coupling between neighboring elements. In particular, it causes the mass matrix to be global which necessitates matrix inversion to time-step the solution. There are classes of problem where the C^0 continuity is not necessary; test and trial functions can then be made discontinuous, and the solution process becomes considerably more cost effective. This new formulation of the Galerkin method has been dubbed the Discontinuous Galerkin Method (DGM). It is most suitable to compute the solution to problems governed primarily by hyperbolic (like the pure advection equation or the shallow water equation) or predominantly hyperbolic (like the advection-diffusion equation with high Peclet number).

In the following we describe the formulation of DGM for the simple case of a pure advection equation:

$$T_t + \mathbf{v} \cdot \nabla T = 0 \tag{11.163}$$

where $\mathbf{v}$ is the advective velocity field. If $\mathbf{v}$ is divergence-free, that is $\nabla \cdot \mathbf{v} = 0$, the advection equation can be written in the conservative form

$$T_t + \cdot\nabla \mathbf{F} = 0, \quad \mathbf{F} = \mathbf{v}T \tag{11.164}$$

where $\mathbf{F}$ is the flux and is a function of T; equation 11.164 is written in the form of a conservation law. We suppose that the domain of interest has been divided into elements. Then the following variational statement applies inside each element:

$$\int_E (T_t + \cdot\nabla \mathbf{F}) w \; \mathrm{d}V = 0 \tag{11.165}$$

where w are the test functions, and E is the element of interest. If w is sufficiently smooth, we can integrate the divergence term by part using the Gauss theorem to obtain:

$$\int_E (T_t w - \mathbf{F} \cdot \nabla w) \; \mathrm{d}V + \int_{\partial E} w \mathbf{F} \cdot \mathbf{n} \; \mathrm{d}S = 0 \tag{11.166}$$

where ∂E is the boundary of element E, $\mathbf{n}$ is the unit outward normal. The boundary integral represent a weighed sum of the fluxes leaving/entering the element. The discretization steps consist of replacing the infinite dimensional space of the test functions by a finite space, and representing the solution by a finite expansion T_h. Since T_h is discontinuous at the edges of elements, we must also replace the flux $\mathbf{F}(T)$ by a *numerical* flux $\mathbf{G}$ that depends on the values of T_h outside and inside the element:

$$\mathbf{G} = \mathbf{G}(T^i, T^o) \tag{11.167}$$

where T^i and T^o represent the values of the function on the edge as seen from inside element, and from the neighboring element, respectively. Physical intuition dictates that the right flux is that obtained by following the characteristic (Lagrangian trajectory). For the advection equation that means the outside value is used if $\mathbf{n} \cdot \mathbf{v} < 0$ and the inside value is used if $\mathbf{n} \cdot \mathbf{v} > 0$.

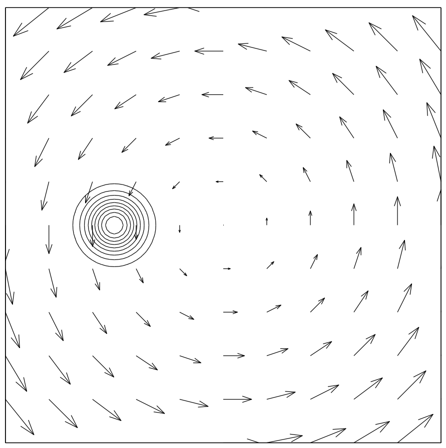


Figure 11.19: Flow field and initial condition for Gaussian Hill experiment

Proceeding as before and defining the local/elemental approximation to T_h as

$$T(\mathbf{x}) = \sum_{i=1}^{N} T_i(t)\phi_i(\mathbf{x}) \tag{11.168}$$

and the degrees of freedom T_i are determined by the following ODE:

$$\begin{aligned} M\frac{\mathrm{d}T}{\mathrm{d}t} &+ \mathcal{G} = 0 &(11.169)\\ M_{ji} &= \int_E \phi_i\phi_j \,\mathrm{d}V &(11.170)\\ G_j &= \int_E \phi_i \mathbf{G}\cdot\mathbf{n}\,\mathrm{d}S &(11.171)\end{aligned}$$

where M is the *local* mass matrix; no assembly is required. The great value of DGM is that the variational statement operates one element at a time, and hence the mass matrix arising from the time-derivative is purely local. The only linkage to neighboring elements comes from considering the fluxes along the boundary. Thus, even if the matrix M is full, it is usually a small system that can be readily inverted, and the process of time integration becomes considerably cheaper. Another benefit of DGM is that it satisfies the local conservation property so dear to many oceanographic/coastal modeler, since the fluxes defined on each edge are unique. Finally, by dropping the continuity requirements it becomes quite feasible to build locally and dynamically adaptive solution strategies without worrying about global continuity of the function. In the next section we compare the performance of the DGM with that of the continuous formulation on several problems; all simulations will be performed using the spectral element interpolation.

11.6.1 Gaussian Hill Experiment

The first experiment is designed to establish the convergence property of the different methods. To this end we choose the classical problem of advecting a passive tracer in the unit square domain ($0 \leq x, y \leq 1$) by a rotating non-divergent flow given by:

$$u = -\omega(y - 1/2),\ v = \omega(x - 1/2). \tag{11.172}$$

where ω is set to 2π. The initial condition is infinitely smooth and given by a Gaussian distribution

$$\phi = e^{-\frac{r^2}{l^2}},\ r = \sqrt{\left(x - \frac{1}{4}\right)^2 + \left(y - \frac{1}{2}\right)^2} \tag{11.173}$$

with an e-folding length scale $l = 1/16$. Periodic boundary conditions are imposed on all sides. All models were integrated for one rotation which time the solution should be identical to the initial condition.

The convergence curves for the l_2 norm of the error are displayed in 11.20 for the different formulations. The time integration consists of RK4 for traditional Galerkin method (which we will refer to as the CGM), and DGM; the time step was chosen so that spatial errors are dominant. The convergence curves for CGM, and DGM are similar and indicate the exponential decrease of the error as the spectral truncation is increased for a constant elemental partition.

In order to compare the benefits of h versus p refinements, we plot in the right panels of figure 11.20 the l_2 error versus the total number of collocation points in each direction. The benefits of p-refinement for this infinitely smooth problem is readily apparent for CG: Given a fixed number of collocation points, the error is smallest for the smallest number of elements, and hence highest spectral truncation. (The number of collocation points is given by $K(N-1)+1$, where N is the number of points per element and K is the number of elements). The situation is not as clear for DGM where the different curves tend to overlap. This is probably due to the discontinuous character of the interpolation: adjacent elements are holding duplicate information about the solution, and the total number of degrees of freedom grows like KN.

11.6.2 Cone Experiment

The second experiment is designed to establish the behavior of the different methods in the presence of "mild" discontinuities. The flow field is the same as for the Gaussian hill, but the initial profile consists of a cone:

$$u(x, y, t = 0) = \max(0, 1 - 8r),\quad r = \sqrt{\left(x - \frac{1}{4}\right)^2 + \left(y - \frac{1}{2}\right)^2}. \tag{11.174}$$

The cone has a peak at $r = 0$ which decreases linearly to 0; there is a slope discontinuity at $r = 1/8$. The initial conditions contours are similar to the one depicted in figure 11.19. The same numerical parameters were used as for the Gaussian Hill problem.

The presence of the discontinuity ruins the spectral convergence property of the spectral element method. This is born out in the convergence curves (not shown) which display a $1/N$ convergence rate only in the l_2 norm for a fixed number of elements; h refinement is a more effective way to reduce the errors in the present case. In the following, we compare the performance of the different schemes using a single resolution, 10x10 elemental partition with 6 points per element. Figure 11.21 compares the solution

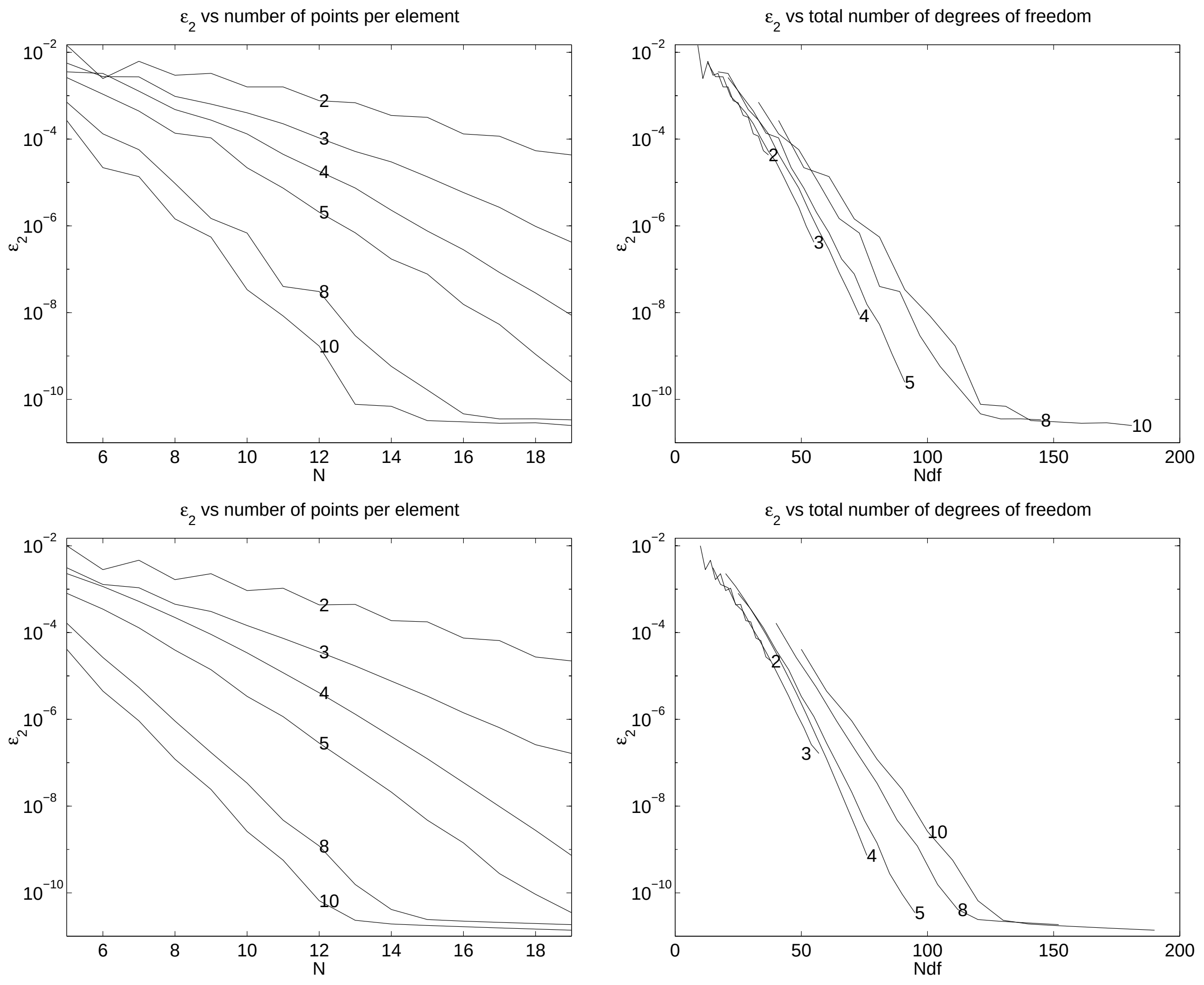


Figure 11.20: Convergence curve in the L_2 norm for the Gaussian Hill initial condition using, from top to bottom, CGM, and DG. The labels indicate the number of elements in each direction. The abcissa on the left graphs represent the spectral truncation, and on the right the total number of collocation points.

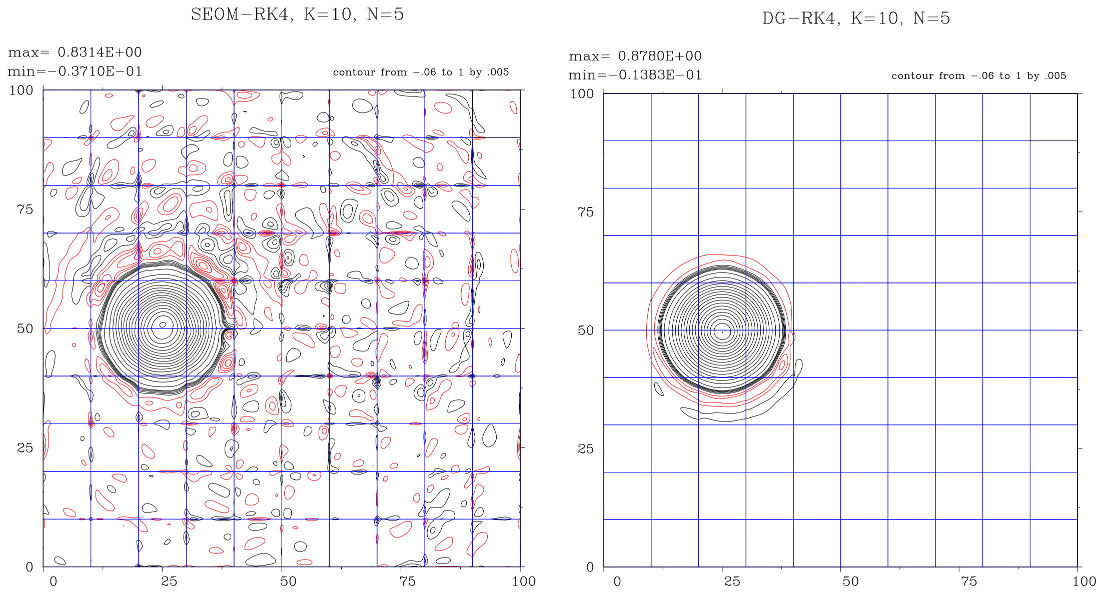


Figure 11.21: Contour lines of the rotating cone problem after one revolution for CG (left), and DG (right), using the 10×6 grid. The contours are irregularly spaced to highlight the Gibbs oscillations.

for the 4 schemes at the end of the simulations. We note that the contour levels are irregularly spaced and were chosen to highlight the presence of Gibbs oscillations around the 0-level contour.

For CG, the oscillation are present in the entire computational region, and have peaks that reaches -0.03. Although the DG solution exhibits Gibbs oscillations also, these oscillations are confined to the immediate neighborhood of the cone. Their largest amplitude is one third that observed with CG. Further reduction of these oscillation require the use of some form of dissipation, e.g. Laplacian, high order filters, or slope limiters. We observe that CG shows a similar decay in the peak amplitude of the cone with DG doing a slightly better job at preserving the peak amplitude. Figure 11.22 shows the evolution of the largest negative T as a function of the grid resolution for CG and DG. We notice that the DG simulation produces smaller negative values, up to a factor of 5, than CG at the same resolution.

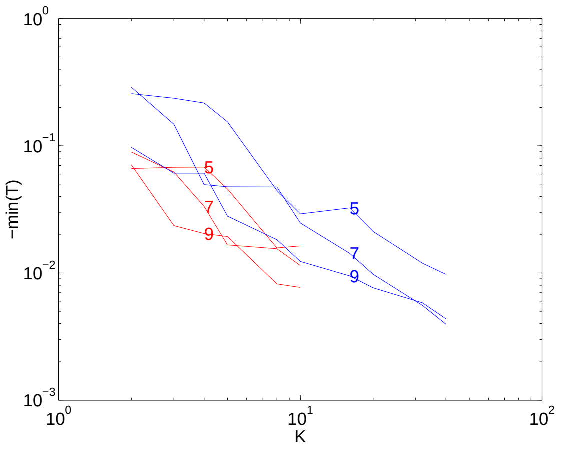


Figure 11.22: Min(T) as a function of the number of element at the end of the simulation. The red lines are for DG and the blue lines for CG. The number of points per element is fixed at 5, 7 and 9 as indicated by the lables.

Chapter 12

Linear Analysis

12.1 Linear Vector Spaces

In the following we will use the convention that bold roman letter, such as $\mathbf{x}$, denote vectors, greek symbols denote scalars (real or complex) and capital roman letters denote operators.

12.1.1 Definition of Abstract Vector Space

We call a set V of vectors a linear vector space V if the following requirements are satisfied:

1. We can define an addition operation, denoted by '+', such that for any 2 elements of the vector space $\mathbf{x}$ and $\mathbf{y}$, the result of the operation $\mathbf{z} = (\mathbf{x} + \mathbf{y})$ belongs to V. We say that the set V is *closed under addition*. Furthermore the addition must have the following properties:

   (a) commutative $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$

   (b) associative $(\mathbf{x} + \mathbf{y}) + \mathbf{z} = \mathbf{x} + (\mathbf{y} + \mathbf{z})$

   (c) neutral element: there exist a null zero vector $\mathbf{0}$ such that $\mathbf{x} + \mathbf{0} = \mathbf{x}$

   (d) for each vector $\mathbf{x} \in V$ there exist a vector $\mathbf{y}$ such that $\mathbf{x} + \mathbf{y} = \mathbf{0}$, we denote this vector by $\mathbf{y} = -\mathbf{x}$.

2. We can define a scalar multiplication operation defined between any vector $\mathbf{x} \in V$ and a scalar α such that $\alpha\mathbf{x} \in V$, i.e. V is closed under scalar multiplication. The following properties must also hold for 2 scalars α and β, and any 2 vectors $\mathbf{x}$ and $\mathbf{y}$:

   (a) $\alpha(\beta\mathbf{x}) = (\alpha\beta)\mathbf{x}$

   (b) Distributive scalar addition: $(\alpha + \beta)\mathbf{x} = \alpha\mathbf{x} + \beta\mathbf{x}$

   (c) Distributive vector addition: $\alpha(\mathbf{x} + \mathbf{y}) = \alpha\mathbf{x} + \alpha\mathbf{y}$

   (d) $1\mathbf{x} = \mathbf{x}$

   (e) $0x = \mathbf{0}$

12.1.2 Definition of a Norm

In order to provide the abstract vector space with the sense of length and distance, we define the norm or length of a vector $\mathbf{x}$, as the number $\|\mathbf{x}\|$. In order for this number to make sense as a distance we put the following restrictions on the definition of the norm:

1. $\|\alpha\mathbf{x}\| = |\alpha|\ \|\mathbf{x}\|$
2. Positivity $\|\mathbf{x}\| > 0, \forall\mathbf{x} \neq \mathbf{0}$, and $\|\mathbf{x} = 0\| \Leftrightarrow \mathbf{x} = \mathbf{0}$
3. Triangle or Minkowski inequality $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$

With the help of the norm we can define now the **distance** between 2 vectors $\mathbf{x}$ and $\mathbf{y}$ as $\|\mathbf{x}-\mathbf{y}\|$, i.e. the norm of their difference. So 2 vectors are equal or identical if their distance is zero. Furthermore, we can now talk about the **convergence** of a vector sequence. Specifically, we say that a vector sequence $\mathbf{x}_n$ converges to $\mathbf{x}$ as $n \to \infty$ if for any $\epsilon > 0$ there is an N such that $\|\mathbf{x}_n - \mathbf{x}\| < \epsilon\ \ \forall n > N$.

12.1.3 Definition of an inner product

It is generally usefull to introduce the notion of angle between 2 vectors $\mathbf{x}$ and $\mathbf{y}$. We thus define the scalar $(\mathbf{x}, \mathbf{y})$ which must satisfy the following properties

1. conjugate symmetry: $(\mathbf{x}, \mathbf{y}) = \overline{(\mathbf{y}, \mathbf{x})}$, where the overbar denotes the complex conjugate.
2. linearity: $(\alpha\mathbf{x} + \beta\mathbf{y}, \mathbf{z}) = \alpha(\mathbf{x}, \mathbf{z}) + \beta(\mathbf{y}, \mathbf{z})$
3. positiveness: $(\mathbf{x}, \mathbf{x}) > 0,\ \forall\mathbf{x} \neq \mathbf{0}$, and $(\mathbf{x}, \mathbf{x}) = 0$ if $\mathbf{x} = \mathbf{0}$.

The above properties imply the **Schwartz inequality**:

$$|(\mathbf{x}, \mathbf{y})| \leq \sqrt{(\mathbf{x}, \mathbf{x})}\ \sqrt{(\mathbf{y}, \mathbf{y})} \tag{12.1}$$

This inequality suggest that the inner product $\sqrt{(\mathbf{x}, \mathbf{x})}$ can be defined as a norm, so that $\|\mathbf{x}\| = (\mathbf{x}, \mathbf{x})^{1/2}$ With the definition of a norm we call 2 vectors orthogonal iff $(\mathbf{x}, \mathbf{y}) = 0$. Moreover iff $(\mathbf{x}, \mathbf{y}) = \|\mathbf{x}\|\ \|\mathbf{y}\|$ the 2 vectors are colinear or aligned.

12.1.4 Basis

A set of vectors $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ different from $\mathbf{0}$ are called a basis for the vector space V if they have the following 2 properties:

1. **Linear independence**:
$$\sum_{i=1}^{N} \alpha_i \mathbf{e}_i = \mathbf{0} \Leftrightarrow \alpha_i = 0 \forall i \tag{12.2}$$
If at least one of the α_i is non-zero, the set is called linearly dependent, and one of the vectors can be written as a linear combination of the others.

2. **Completeness** Any vector $\mathbf{z} \in V$ can be written as a linear combination of the basis vectors.

The number of vectors needed to form a basis is called the dimension of the space V. Put in another way, V is **N dimensional** if it contains a set of N independent vectors but no set of $(N+1)$ of independent vectors. If N vectors can be found for each N, no matter how large, we say that the vector space is infinite dimensional.

A basis is very usefull since it allows us to describe any element $\mathbf{x}$ of the vector space. Thus any vector $\mathbf{x}$ can be "expanded" in the basis $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$ as:

$$\begin{aligned} \mathbf{x} &= \sum_i^N \alpha_i \mathbf{e}_i \\ &= \alpha_1\mathbf{e}_1 + \alpha_2\mathbf{e}_2 + \ldots + \alpha_n\mathbf{e}_n \end{aligned} \tag{12.3}$$

This representaion is also unique by the independence of the basis vectors. The question of course if how to find out the coefficients α_i which are nothing but the coordinates of $\mathbf{x}$ in $\mathbf{e}_i$. We can take the inner product of both sides of the equations to come up with the following linear system of algebraic equations for the α_i:

$$\begin{array}{lllll} \alpha_1(\mathbf{e}_1,\mathbf{e}_1) & +\alpha_2(\mathbf{e}_2,\mathbf{e}_1) & +\ldots & +\alpha_n(\mathbf{e}_n,\mathbf{e}_1) & = (\mathbf{x},\mathbf{e}_1) \\ \alpha_1(\mathbf{e}_1,\mathbf{e}_2) & +\alpha_2(\mathbf{e}_2,\mathbf{e}_2) & +\ldots & +\alpha_n(\mathbf{e}_n,\mathbf{e}_2) & = (\mathbf{x},\mathbf{e}_2) \\ \vdots & & & & \\ \alpha_1(\mathbf{e}_1,\mathbf{e}_n) & +\alpha_2(\mathbf{e}_2,\mathbf{e}_n) & +\ldots & +\alpha_n(\mathbf{e}_n,\mathbf{e}_n) & = (\mathbf{x},\mathbf{e}_n) \end{array} \tag{12.4}$$

The coupling between the equations makes it hard to compute the coordinates of a vector in a general basis, particularly for large N. Suppose however that the basis set is mutually orthogonal, that is every every vector $\mathbf{e}_i$ is orthogonal to every other vector $\mathbf{e}_j$, that is $(\mathbf{e}_i, \mathbf{e}_j) = 0$ for all $i \neq j$. Another way of denoting this mutual orthogonality is by using the Kronecker delta function, δ_{ij}:

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases} \tag{12.5}$$

The orthogonality property can then be written as $(\mathbf{e}_i, \mathbf{e}_j) = \delta_{ij}(\mathbf{e}_j, \mathbf{e}_j)$, and the basis is called orthogonal. For an orthogonal basis the system reduces to the uncoupled (diagonal) system

$$(\mathbf{e}_j, \mathbf{e}_j)\alpha_j = (\mathbf{x}, \mathbf{e}_j) \tag{12.6}$$

and the coordinates can be computed easily as

$$\alpha_j = \frac{(\mathbf{x}, \mathbf{e}_j)}{(\mathbf{e}_j, \mathbf{e}_j)} = \frac{(\mathbf{x}, \mathbf{e}_j)}{\|\mathbf{e}_j\|^2} \tag{12.7}$$

The basis set can be made **orthonormal** by normalizing the basis vectors (i.e. rescaling $\mathbf{e}_j$ by such that their norm is 1), then $\alpha_j = (\mathbf{x}, \mathbf{e}_j)$.

12.1.5 Example of a vector space

Consider a set V whose elements are defined by N tuples, i.e. each element $\mathbf{x}$ of V is identified by N scalar fields $(\xi_1, \xi_2, \ldots, \xi_n)$ Let us define the addition of 2 vectors $\mathbf{x}$, defined as above, and $\mathbf{y}$ defined by the N-tuples $(\eta_1, \eta_2, \ldots, \eta_n)$ as the vector $\mathbf{z}$ whose N-tuples are $\sigma_i = \xi_i + \eta_i$. Furthermore, we define the vector $\mathbf{z} = \alpha a$ by its N-tuples $\sigma_i = \alpha\xi_i$. It is then easy to verify that this space endowed with the vector addition and scalar multiplication defined above fullfill the requirements of a vector space.

A norm for this vector space can be easily defined by the so-called p norm where

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{N} |\xi|^p\right)^{\frac{1}{p}} \tag{12.8}$$

A particularly usefull norm is the 2-norm ($p = 2$), also, called the Euclidean norm. Other usefull norms are the 1-norm ($p = 1$) and the infinity norm:

$$\|\mathbf{x}\|_\infty = \lim_{p\to\infty} \|\mathbf{x}\|_p = \max_{1\le j\le N} |\xi_j|. \tag{12.9}$$

An inner product for this vector space can be defined as:

$$(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{N} \xi_i \overline{\eta}_i \tag{12.10}$$

It can be easily verified that this inner product satifies all the needed requirements. Furthermore, the norm introduced by this inner product is nothing but the 2-norm mentioned above.

An orthonormal basis for the vector space V is given by;

$$\begin{aligned}
\mathbf{e}_1 &= (1, 0, 0, 0, \ldots, 0) \\
\mathbf{e}_2 &= (0, 1, 0, 0, \ldots, 0) \\
&\vdots \\
\mathbf{e}_n &= (0, 0, 0, 0, \ldots, 1)
\end{aligned} \tag{12.11}$$

is a complete and independent vector set. Thus V is N-dimensional.

12.1.6 Function Space

Spaces where the vectors are functions occupy an important place in linear analysis. Their properties as linear spaces are harder to manipulate as they are infinite dimensional spaces. For example the space of all continuous functions defined on the interval $a \le t \le b$ and which we denote by $\mathcal{C}(a, b)$ is a linear vector space where vector addition and scalar multiplication are defined in an obvious way. The inner product on this vector space is the continuous analogue of the inner product defined for the N-tuple space. Suppose for a moment that the functions $\mathbf{x}(t)$ and $\mathbf{y}(t)$ are defined on N equally spaced points on the interval $[a, b]$ (i.e. discrete space) by their pointwise values $\mathbf{x}_i$ and $\mathbf{y}_i$ at the points t_i, and let us define the discrete inner product as the Riemann type sum:

$$(\mathbf{x}, \mathbf{y}) = \frac{b-a}{N} \sum_{i=1}^{N} \mathbf{x}_i \overline{\mathbf{y}}_i \tag{12.12}$$

In the limit N tends to infinity the above discrete sum become

$$(\mathbf{x}, \mathbf{y}) = \lim_{N \to \infty} \frac{b-a}{N} \sum_{i=1}^{N} \mathbf{x}_i \overline{\mathbf{y}}_i = \int_a^b \mathbf{x}(t) \overline{\mathbf{y}}(t)\, dt. \tag{12.13}$$

Two functions are said to be orthogonal if $(\mathbf{x}, \mathbf{y}) = 0$.

Similarly we can define the p-norm of a function as:

$$\|\mathbf{x}\|_p = \int_a^b |x(t)|^p \, dt \tag{12.14}$$

The 1-norm, 2-norm and ∞ norms follow by setting $p = 1, 2.$ and ∞.

The main difficulties with function spaces are twofold. First they are infinite dimensional and thus require an infinite set of vectors to define a basis; proving completeness is hence difficult. Two, functions are usually defined in a continuum where limits may be in or outside the vector space. Some interesting issue arise. Consider for example the sequence of functions

$$\mathbf{x}(t) = \begin{cases} 0, & -1 \le t \le 0 \\ nt, & 0 \le t \le \frac{1}{n} \\ 1, & \frac{1}{n} \le t \le 1 \end{cases} \tag{12.15}$$

defined on $\mathcal{C}(-1, 1)$. This sequence converges to the step function, also known as the Heaviside function, $H(t)$ which does not belong to $\mathcal{C}(-1, 1)$ since it is discontinuous. That although the sequence $\mathbf{x}_n$ is in $\mathcal{C}(-1, 1)$, its limit as $n \to \infty$ is not; we say that the space does not contain its limit points and hence is not closed. This is akin to the space $\mathcal{C}(-1, 1)$ having holes in it. This is rather unfortunate as closed spaces are more tractable.

It is possible to create a closed space by changing the definition of the space slightly to that of the Lebesgue space $\mathcal{L}_2(a, b)$, namely the space of functions that are square integrable on the interval $[a, b]$, i.e.

$$\|\mathbf{x}\|_2 = \int_a^b |x(t)|^2 \, dt < \infty. \tag{12.16}$$

$\mathcal{L}_2(a, b)$ is an example of a **Hilbert space** – a closed inner product space with the $\|\mathbf{x}\| = (\mathbf{x}, \mathbf{x})^{\frac{1}{2}}$ – a closed inner product space with the $\|\mathbf{x}\| = (\mathbf{x}, \mathbf{x})^{\frac{1}{2}}$.

The issue of defining a basis function for a function space is complicated by infinite dimension of the space. Assume I have an infinite set of linearly independent vectors, if I remove a single element of that set, the set is still infinite but clearly cannot generate the space. It turns out that it is possible to prove completeness but we will defer the discussion until later. For the moment, we assume that it is possible to define such a basis. Furthermore, this basis can be made to be orthogonal. The Legendre polynomials P_n are an orthogonal set of functions over the interval $[-1, 1]$, and the trigonometric functions e^{inx} are orthogonal over $[-\pi, \pi]$.

Suppose that an orthogonal and complete basis $\mathbf{e}_i(t)$ has been defined, then we can expand a vector in this basis function:

$$\mathbf{x} = \sum_{i=1}^{\infty} \alpha_i \mathbf{e}_i \tag{12.17}$$

The above expansion is referred to as a generalized Fourier series, and the α_i are the Fourier coefficients of $\mathbf{x}$ in the basis $\mathbf{e}_i$. We can also follow the procedure outlined for the finite-dimensional spaces to compute the α_i's by taking the inner product of both sides of the expansion. The determination of the coordinate is, again, particularly easy if the basis is orthogonal

$$\alpha_i = \frac{(\mathbf{x}, \mathbf{e}_i)}{(\mathbf{e}_i, \mathbf{e}_i)} \tag{12.18}$$

In particular, if the basis is orthonormal $\alpha_i = (\mathbf{x}, \mathbf{e}_i)$.

In the following we show that the Fourier coefficients are the best approximation to the function in the 2-norm, i.e. the coefficients α_i minimize the norm $\|\mathbf{x} - \sum_i \alpha_i \mathbf{e}_i\|_2$. We have:

$$\begin{aligned} \|\mathbf{x} - \sum_i \alpha_i \mathbf{e}_i\|_2^2 &= (\mathbf{x} - \sum_i \alpha_i \mathbf{e}_i, \mathbf{x} - \sum_i \alpha_i \mathbf{e}_i) & (12.19) \\ &= (\mathbf{x}, \mathbf{x}) - \sum_i \overline{\alpha}_i (\mathbf{x}, \mathbf{e}_i) - \sum_i \alpha_i (\mathbf{e}_i, \mathbf{x}) + \sum_i \sum_j \alpha_i \alpha_j (\mathbf{e}_i, \mathbf{e}_j) & (12.20) \end{aligned}$$

The orthogonality of the basis functions can be used to simply the last term on the right hand side to $\sum_i |\alpha_i|^2$. If we furthermore define $a_i = (\mathbf{x}, \mathbf{e}_i)$ we have:

$$\begin{aligned} \|\mathbf{x} - \sum_i \alpha_i \mathbf{e}_i\|_2^2 &= \|\mathbf{x}\|_2 + \sum_i [\alpha_i \overline{\alpha}_i - a_i \overline{\alpha}_i - \overline{a}_i \alpha_i] & (12.21) \\ &= \|\mathbf{x}\|_2 + \sum_i [(a_i - \alpha_i)(\overline{a}_i - \overline{\alpha}_i) - \overline{a}_i] & (12.22) \\ &= \|\mathbf{x}\|_2 + \sum_i |a_i - \alpha_i|^2 - \sum_i |a|_i^2 & (12.23) \end{aligned}$$

Note that since the first and last terms are fixed, and the middle term is always greater or equal to zero, the left hand side can be minimized by the choice $\alpha_i = a_i = (\mathbf{x}, \mathbf{e}_i)$. The minimum norm has the value $||x||_2 - \sum_i |a_i|^2$. Since this value must be always positive then

$$\sum_i |a_i| \leq ||x||_2, \tag{12.24}$$

A result known as the Bessel inequality. If the basis set is complete and the minimum norm tend to zero as the number of basis functions increases to infinity, we have:

$$\sum_i |a_i| = ||x||_2, \tag{12.25}$$

which is known as the Paserval equality; this is a generalized "Pythagorean Theorem".

12.1.7 Pointwise versus Global Convergence

Consider the 2 functions $\mathbf{x}(t)$ and $\mathbf{y}(t)$ defined on $[-1, 1]$ as follows:

$$\mathbf{x}(t) = |t| \tag{12.26}$$

$$\mathbf{y}(t) = \begin{cases} |t|, & \text{for } |t| > 0 \\ 1, & x = 0 \end{cases} \tag{12.27}$$

Both functions belong to $\mathcal{C}(-1, 1)$ and are identical for all t except $t = 0$. If we use the 2-norm to judge the distance between the 2 functions, we get that $\|\mathbf{x} - \mathbf{y}\|_2 = 0$, hence the functions are the same. However, in the maximum norm, the 2 functions are not identical since $\|\mathbf{x} - \mathbf{y}\|_\infty = 1$. This example makes apparent that the choice of norms is critical in deciding whether 2 functions are the same or different, 2 functions that may be considered identical in one norm can become different in an another norm. This is simply an apparent contradiction and reflects the fact that different norms measure different things. The 2-norm for example looks at the global picture and asks if the 2 functions are the same over the interval; this is the so-called mean-square convergence. The infinity norm on the other hand measures pointwise convergence.

12.2 Linear Operators

An operator or (transformation) L, we mean a mapping from one vector space, called the domain $\mathcal{D}$, to another vector space called the range $\mathcal{R}$. L thus describes functions operating on vectors. As with functions, we require L to be uniquely valued although not necessarily one-to-one, i.e. we may have $L\mathbf{x} = L\mathbf{y} = \mathbf{z}$ for $\mathbf{x} \neq \mathbf{y}$. However, $L\mathbf{x} = L\mathbf{y} = \mathbf{z}$ implies $\mathbf{x} = \mathbf{y}$, we say that L is one-to-one, i.e. each vector $\mathbf{z}$ in $\mathcal{R}$ has a single corresponding $\mathbf{x}$ such that $L\mathbf{x} = \mathbf{z}$. Two operators A and B are equal if they have the same domain and if $A\mathbf{x} = B\mathbf{y}, \forall \mathbf{x}, \mathbf{y}$. Finally, we say that I is the identity operator if $I\mathbf{x} = \mathbf{x}, \forall \mathbf{x}$ and $\emptyset$ is the null operator if $\emptyset\mathbf{x} = \mathbf{0}\forall\mathbf{x}$.

Operation addition and multiplication by a scalar can be defined. The operator $C = A + B$ is defined as $C\mathbf{x} = A\mathbf{x} + B\mathbf{y}$, and $C = \alpha A$ is defined as $C\mathbf{x} = \alpha(A\mathbf{x})$. The product of two operators $C = AB$ is defined as $C\mathbf{x} = A(B\mathbf{x})$. We can easily see that operator addition commutes (since vector addition must commute), whereas AB need not equal BA, when they do, we say that A and B commute.

We call an operator linear if

$$L(\alpha\mathbf{x} + \beta\mathbf{y}) = \alpha L\mathbf{x} + \beta L\mathbf{y} \tag{12.28}$$

An operator is bounded if there is a positive constant c such that $\|L\mathbf{x}\| < c, \forall \mathbf{x} \in \mathcal{D}$. The smallest suitable bound is called the norm of the operator, thus:

$$\|L\| = \text{lub}_{\mathbf{x}\neq\mathbf{0}} \frac{\|L\mathbf{x}\|}{\|\mathbf{x}\|} \tag{12.29}$$

An adjoint to the operator L is the operator L^* such that

$$(L\mathbf{x}, \mathbf{y}) = (\mathbf{x}, L^*\mathbf{y}), \forall \mathbf{x}, \mathbf{y} \tag{12.30}$$

It is easy to show the following properties:

$$\begin{aligned}(L^*)^* &= L && (12.31)\\ (A+B)^* &= A^* + B^* && (12.32)\\ (AB)^* &= B^* + A^* && (12.33)\end{aligned}$$

If $L^* = L$ we call the operator **self adjoint**.

12.3 Eigenvalues and Eigenvectors

Non trivial solutions $\mathbf{x} \neq \mathbf{0}$ to the equation

$$L\mathbf{x} = \lambda\mathbf{x}, \tag{12.34}$$

are called eigenvectors, the scalars λ are called eigenvalues. The statement above asks essentially if there are special vectors which when transformed by L produce parallel vectors, the ratio of lengths of these two vectors is the eigenvalue. The above equation can be restated as find the non-trivial solution to the homogeneous equation

$$(L - \lambda I)\mathbf{x} = \mathbf{0}. \tag{12.35}$$

Usually the eigenvalues and eigenvectors occur in pairs. If the operator L is a matrix the λ's can be determined by solving the characteristic equations $\det(L - \lambda I) = 0$.

The following results are very important:

1. The eigenvalues of a self adjoint operator are all real, and the eigenvectors corresponding to distinct eigenvalues are mutually orthogonal.

2. For self-adjoint operator L on a finite dimensional domain V, k mutually orthogonal eigenvectors can be found for each eigenvectors of multiplicity k.

3. The 2 properties above imply that the eigenvectors of a self-adjoint operators form a basis for the finite-dimensional space V. The situation is substantially more complicated for infinite dimensional spaces, and is taken care of by the Sturm Liouville theory.

12.4 Sturm-Liouville Theory

We will change our notation and drop the bold face of the vector notation. Given the 2 functions f and g on the interval $a \leq t \leq b$, we define first an inner product of the form:

$$(f,g) = \int_a^b f(t)g(t)w(t)\,dt \tag{12.36}$$

where $w(t) > 0$ on $a < t < b$ (this is a more general inner product that the one defined earlier which correspond to $w(t) = 1$). Let the operator L be defined as follows:

$$\begin{aligned}Ly &= \frac{1}{w(t)}\left\{\frac{d}{dt}\left[p(t)\frac{dy}{dt}\right] + r(t)\right\} && (12.37)\\ \alpha y(a) + \beta y'(a) &= 0, \gamma y(b) + \delta y'(b) = 0 && (12.38)\end{aligned}$$

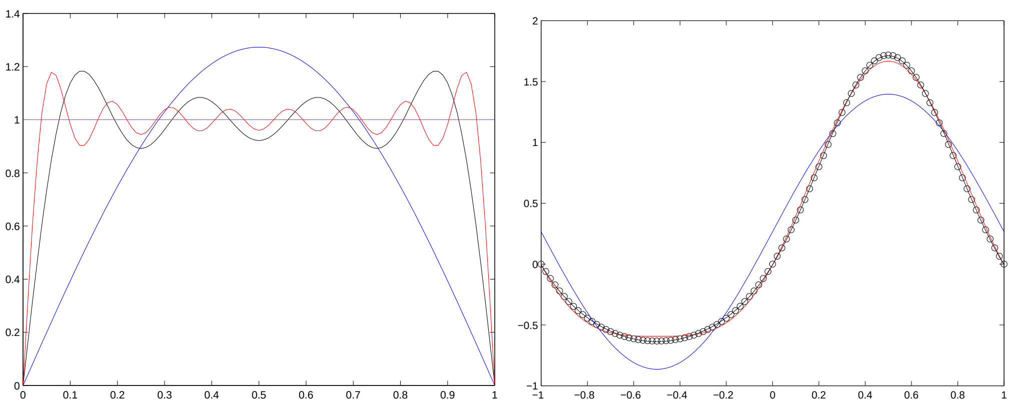


Figure 12.1: Left: step function fit $f = 1$; blue line is for $N = 1$, black line for $N = 7$, and red line for $N = 15$. Right: Fourier expansion to $f = e^{\sin \pi x}$, blue line is for $N = 1$, red for $N = 2$, and black for $N = 3$; the circles show the function f

In the present case L is a differential operator subject to homogeneous boundary conditions. It is easy to show that the operator L is self adjoint under the inner product defined in 12.36. Furthermore, the eigenvalue problem defined by

$$Ly + \lambda y = 0 \tag{12.39}$$

is called a Sturm-Liouville system. Sturm-Liouville systems have the crucial theorem that

Theorem 2 *If both $p(t)$ and $w(t)$ are analytic and positive over the interval $[a, b]$, where a and b are finite, then the eigenfunctions of the Sturm-Liouville system are complete over $\mathcal{L}_2(a, b)$.*

Completeness is also known to hold under other circumstances. For example if $p(t)$ vanishes at one or both end points, then the boundary conditions can be replaced with the requirement that y or y' be finite there. If it happens that $p(a) = p(b)$ we can replace the boundary conditions by periodicity conditions $y(a) = y(b)$ and $y'(a) = y'(b)$.

Note carefully that the domain of the Sturm Liouville system is the set of function that are twice differentiable and that satisfy certain boundary conditions, and yet the theorem asserts completeness over a much broader space $\mathcal{L}_2$, which contains functions that need not be even continuous and that need not satify the boundary conditions.

The various special forms of the Sturm Liouville problems gives rise to various commonly known Fourier series. For example the choice of $w = p = 1$ and $r = 0$ gives rise to the Fourier trigonometric series. For $w = 1$, $p = 1 - t^2$ we get the Fourier Legendre series, etc...

Example 15 The function $y(t) = 1$ on $0 \le t \le 1$ can be expanded in the trigonometrics series $\sum_{m=1}^{N} f_m \sin(m\pi x)$. It is easy to verify that

$$(\sin m\pi x, \sin k\pi x) = \int_0^1 \sin k\pi x \, \sin m\pi x dx = \frac{1}{2}\delta_{km} \tag{12.40}$$

m	A_m	B_m
0	0.000000000000000	0.532131755504017
1	1.13031820798497	$4.097072104153782\times10^{-17}$
2	$4.968164449421373\times10^{-18}$	-0.271495339534077
3	$-4.433684984866381\times10^{-02}$	$-5.790930955238388\times10^{-17}$
4	$-3.861917048379140\times10^{-17}$	$5.474240442093724\times10^{-03}$
5	$5.429263119140263\times10^{-04}$	$2.457952426063564\times10^{-17}$
6	$-3.898111864745672\times10^{-16}$	$-4.497732295430222\times10^{-05}$
7	$-3.198436462370136\times10^{-06}$	$5.047870580187706\times10^{-17}$
8	$3.363168791013840\times10^{-16}$	$1.992124806619515\times10^{-07}$
9	$1.103677177269183\times10^{-08}$	$-1.091272736411650\times10^{-16}$
10	$-4.748619297285671\times10^{-17}$	$-5.505895970193617\times10^{-10}$
11	$-2.497959777896152\times10^{-11}$	$1.281761515677824\times10^{-16}$

Table 12.1: Fourier coefficients of example 16

and that the Fourier coefficients are given by: $f_m = 0$, for even m, and $f_m = \frac{4}{m\pi}$, for odd m. Figure 12.1 illustrates the convergence of the series as the number of Fourier functions retained, N, increases.

Example 16 The function $f = e^{\sin \pi x}$ is periodic over the interval $[-1, 1]$ and can be expanded into a Fourier series of the following form:

$$f(x) = \sum_{m=0}^{N} A_m \cos m\pi x \; + \; B_m \sin m\pi x \tag{12.41}$$

The Fourier coefficients can be determined from:

$$A_m = \frac{\int_{-1}^{1} e^{\sin \pi x} \cos m\pi x dx}{\int_{-1}^{1} \cos^2 m\pi x dx}, \quad B_m = \frac{\int_{-1}^{1} e^{\sin \pi x} \sin m\pi x dx}{\int_{-1}^{1} \sin^2 m\pi x dx} \tag{12.42}$$

Since the integrals cannot be evaluated analytically, we compute them numerically using a very high order methods. The first few Fourier coefficients are listed in table 12.1. Notice in particular the rapid decrease of $|A_m|$ and $|B_m|$ as m increases, and the fact that with 3 Fourier modes the series expansion and the original function are visually identical.

12.5 Application to PDE

The method of separation variables relies on the Sturm Liouville theory to generate a basis made up of eigenfunctions for the function space where the solution is sought. The problem boils down to finding the Fourier coefficients of the solution in that basis. The type of eigenfunctions used depends on the geometry of the domain, the boundary conditions and the partial differential equations.

Example 17 Let us take the example of the Laplace equation $\nabla^u = 0$ defined on the rectangular domain $0 \leq x \leq a$ and $0 \leq y \leq b$, and subject to the boundary conditions that $u = 0$ on all boundaries except the top boundary where $u(x, y = b) = v(x)$. Separation of variables assumes that the solution can be written as the product of functions that depend on a single independent variable: $u(x, y) = X(x)Y(y)$. When this trial solution is substituted in the PDE, we can derive the identity:

$$\frac{X_{xx}}{X} = -\frac{Y_{yy}}{Y} \tag{12.43}$$

Since the left hand side is a function of x-only, and the right hand side a function of y only, and the equality must hold for arbitrary x and y the two ratios must be equal to a constant which we set to $-\lambda^2$, and we end up with the 2 equations

$$X_{xx} + \lambda^2 X = 0 \tag{12.44}$$
$$Y_{yy} - \lambda^2 Y = 0 \tag{12.45}$$

Notice that the 2 equations are in the form of a Sturm-Liouville problem. The solutions of the above 2 systems are the following set of functions:

$$X = A\cos\lambda x + B\sin\lambda x \tag{12.46}$$
$$Y = C\cosh\lambda y + D\sinh\lambda y \tag{12.47}$$

where A, B, C and D are integration constants. Applying the boundary conditions at $x = 0$ and $y = 0$, we deduce that $A = C = 0$. The boundary condition at $x = b$ produces the equation

$$B\sin\lambda a = 0 \tag{12.48}$$

The solution $B = 0$ is rejected since this would result in the trivial solution $u = 0$, and hence we require that $\sin\lambda a = 0$ which yields the permissible values of λ:

$$\lambda_n = \frac{n\pi}{a},\ n = 1, 2, \ldots. \tag{12.49}$$

There are thus an infinite number of possible λ which we have tagged by the subscript n, with correponding $X_n(x)$, $Y_n(x)$, and unknown constants. Since the problem is linear the sum of these solution is also a solution and hence we set

$$u(x, y) = \sum_{n=1}^{\infty} E_n \ \sin\lambda_n x \ \sinh\lambda_n y \tag{12.50}$$

where we have set $E_n = B_n * D_n$. The last unused boundary condition determines the constants E_n with the equation

$$v(x) = \sum_{n=1}^{\infty} E_n \ \sin\lambda_n x \ \sinh\lambda_n b \tag{12.51}$$

It is easy to show that the functions $\sin\lambda_n x$ are orthogonal over the interval $0 \leq x \leq a$, i.e.

$$\int_a^b \sin\lambda_n x \ \sin\lambda_m x \, dx = \delta_{nm}\frac{a}{2} \tag{12.52}$$

which leads to

$$E_n = \frac{2}{a}\int_a^b \sin\lambda_n x \; v(x)\,dx. \tag{12.53}$$

The procedure outlined above can be reintrepeted as follows: the $\sin\lambda_n x$ are the eigenfunctions of the Sturm Liouville problem 12.44 and the eigenvalues are given by 12.49; the basis function is complete and hence can be used to generate the solution as in equation 12.50; the coordinates of the solution in that basis are determined by 12.53.

The eigenvalues and eigenfunctions depend on the partial differential equations, the boundary conditions, and the shape of the domain. The next few examples illustrate this dependence.

Example 18 The heat equation $u_t = \nu(u_{xx} + u_{yy})$ in the same domain as example 17 subject to homogeneous Dirichlet boundary conditions on all sides will generate the eigenvalues eigenfunction pairs $\sin\left(\frac{m\pi x}{a}\right)$, in the x-direction and $\sin\left(\frac{n\pi y}{b}\right)$ in the y-direction. The solution can be written as the double series:

$$u(x,y,t) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} A_{mn}\sin\frac{m\pi x}{a}\sin\frac{n\pi y}{b}e^{-\alpha_{mn}t}, \quad \alpha_{mn} = -\left(\frac{m^2\pi^2}{a^2} + \frac{n^2\pi^2}{b^2}\right) \tag{12.54}$$

Changing the boundary conditions to homogeneous Neumann conditions in the y-direction will change the eigenfunctions to $\cos(\frac{n\pi y}{b})$. If the boundary conditions is homogeneous Neumann at the bottom and Dirichlet at the top, the eigenpairs in the y-directions become $\cos\left(\frac{(2n+1)\pi y}{b}\right)$.

Example 19 Solution of the wave equation $u_{tt} = c^2\nabla^2 u$ in the disk $0 \le r \le a$ using cylindrical coordinates will produce the following set of equations in each independent variable:

$$\begin{aligned} T_{tt} + \kappa^2 c^2 T &= 0 && (12.55)\\ \Theta_{\theta\theta} + \lambda^2\Theta &= 0 && (12.56)\\ r^2 R_{rr} + rR_r + (\kappa^2 r^2 - \lambda^2)R &= 0 && (12.57)\end{aligned}$$

Since the domain is periodic in the azimuthal direction, we should expect a periodic solution and hence λ must be an integer. The radial equation is nothing but the Bessel equation in the variable κr, its solutions are given by $R = A_n J_n(\kappa r) + B_n Y_n(\kappa r)$. $B_n = 0$ must be imposed if the solution is to be finite at $r = 0$. The eigenvalues κ are determined by imposing a boundary condition at $r = a$. For a homogeneous Dirichlet conditions, the eigenvalues are determined by the roots ξ_{mn} of the Bessel functions $J_n(\xi_m) = 0$, and hence $\kappa_{mn} = \xi_{mn}/a$. Note that κ_{mn} is the radial wavenumber. The solution can now be expanded as:

$$\begin{aligned} u(r,\theta,t) = \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} J_n(\kappa_{mn}r)\,[(A_{mn}\cos n\theta + B_{mn}\sin n\theta)\cos\sigma_{mn}t \quad + \\ (C_{mn}\cos n\theta + D_{mn}\sin n\theta)\sin\sigma_{mn}t] \qquad (12.58)\end{aligned}$$

where $\sigma_{mn} = \kappa_{mn} c$ is the time frequency. The integration constants must be determined from the initial conditions of which there must be 2 since we have a second derivative in time. In the present examples the radial eigenfunctions are given the Bessel functions of the first kind and order n and the eigenvalues are determined by the roots of the J_n. Notice that the Bessel equation is also a Sturm Liouville problem and hence the basis $J_n(\kappa_{mn} r)$ must be complete and orthogonal. Periodicity in the azimuthal direction yields the trigonometric functions, and quantizes the eigenvalues to the set of integers.

Chapter 13

Rudiments of Linear Algebra

13.1 Vector Norms and Angles

We need to generalize the notion of distance and angles for multi-dimensional systems. These notions are intuitive for two and three dimensional vectors and provide a basis for the generalization. This generalization does not prescribe a single formula for a norm or distance but lists the properties that must be satisfied for a measure to be called a distance or angle.

13.1.1 Vector Norms

A vector norm is defined as the measure of a vector in real-number space, i.e. it is a function that associated a real (positive) number for each member of the vector space. The norm of a vector $\mathbf{u}$ is denoted by $\|\mathbf{u}\|$ and must satisfy the following properties:

1. **positivity**: $\|\mathbf{u}\| \geq 0$, and if $\|\mathbf{u}\| = 0$, then $\mathbf{u} = \mathbf{0}$. All norms are positive and only the null vector has 0 norm.

2. **Scalar multiplication**: $\|\alpha\mathbf{u}\| = |\alpha|\|\mathbf{u}\|$, for any scalar α.

3. **triangle inequality**: $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ for any two vectors $\mathbf{u}$ and $\mathbf{v}$.

For a vector $\mathbf{u} = (u_1, u_2, \ldots, u_N)$ an operator that defines a norm is the so-called L_p norm defined as:

$$\|\mathbf{u}\|_p = \left(\sum_{i=1}^{N} |u_i|^p\right)^{\frac{1}{p}} \tag{13.1}$$

The following are frequently used values for p:
1-norm: $p = 1$, $\|\mathbf{u}\|_1 = \sum_{i=1}^{N} |u_i|$;
2-norm: $p = 2$, $\|\mathbf{u}\|_2 = \sqrt{\sum_{i=1}^{N} |u_i|^2}$;
max-norm: $p = \infty$, $\|\mathbf{u}\|_\infty = \max_i |u_i|$.

13.1.2 Inner product

An inner product is an operation that associates a real number between any two vectors $\mathbf{u}$ and $\mathbf{v}$. It is usually denoted by $(\mathbf{u}, \mathbf{v})$ or $\mathbf{u} \cdot \mathbf{v}$; and has the following properties:

1. **commutative**: $(\mathbf{u}, \mathbf{v}) = (\mathbf{v}, \mathbf{u})$.
2. **linear** under scalar multiplication: $(\alpha\mathbf{u}, \mathbf{v}) = \alpha(\mathbf{u}, \mathbf{v})$.
3. **linear** under vector addition: $(\mathbf{u}, \mathbf{v} + \mathbf{w}) = (\mathbf{u}, \mathbf{v}) + (\mathbf{u}, \mathbf{v})$.
4. **positivity** $(\mathbf{u}, \mathbf{u}) \geq 0$, and $(\mathbf{u}, \mathbf{u}) = 0$ implies $\mathbf{u} = 0$.

The norm and inner product definition allow us to defined the cosine of an angle between two vectors as:

$$\cos\theta = \frac{(\mathbf{u}, \mathbf{v})}{\|\mathbf{u}\|\,\|\mathbf{v}\|} \tag{13.2}$$

The properties of the inner product allows it to defined a vector norm, often called the inner-product induced norm: $\|\mathbf{u}\| = (\mathbf{u}, \mathbf{u})$.

13.2 Matrix Norms

Properties of a matrix norm:

- **positivity**: $\|L\| > 0 \forall L$, $\|L\| = 0$ implies $L = 0$.
- **scalar multiplication**: $\|\alpha L\| = |\alpha| \|L\|$, where α is a scalar
- **triangle inequality**: $\|L + M\| \leq \|L\| + \|M\|$
- $\|LM\| \leq \|L\| \|M\|$

Since matrices and vectors occur together, it is reasonable to put some conditions on their respective norms. Matrix and vector norms are compatible if:

$$\|Lu\| \leq \|L\|\,\|u\| \forall \|u\| \neq 0 \tag{13.3}$$

It is possible to use a vector norm to define a matrix norm, the so-called, subordinate norm:

$$\|L\| = \max_{u} \frac{\|Lu\|}{\|u\|} = \max_{\|u\|=1} \|Lu\| \tag{13.4}$$

Here are some common matrix norms some satisfy the compatibility and can be regarded as subordinate norms:

1. **1-norm** $\|L\|_1 = \max_j(\sum_{i=1}^{N} |l_{ij}|)$ This is also referred to as the maximum column sum.
2. ∞**-norm** $\|L\|_\infty = \max_i(\sum_{j=1}^{N} |l_{ij}|)$ This is also referred to as the maximum row sum.
3. **2-norm** $\|L\|_2 = \sqrt{\rho(L^T L)}$ where ρ is the spectral radius (see below). If the matrix L is symmetric $L^T = L$, then $\|L\|_2 = \sqrt{\rho(L^2)} = \rho(L)$.

13.3 Eigenvalues and Eigenvectors

Eigenvalues and eigenvectors of a matrix L are the pairs $(\lambda,\mathbf{u})$ such that

$$L\mathbf{u} = \lambda\mathbf{u}, \quad \mathbf{u} \neq 0 \tag{13.5}$$

The eigenvalues can be determined by rewriting the definition as $(L - \lambda_i I)\mathbf{u}_i = 0$, and requiring that the solutions to this equation have non-trivial solution. The condition is that the determinant of the matrix be zero: $\det(L - \lambda I) = 0$. This is the characteristic equation of the system. For an $N \times N$ matrix, the characteristic equation is an n-degree polynomial that admits n complex roots. These roots are precisely the eigenvalues of the system.

Assuming, the eigenvalues of the matrix L are λ_i and $\mathbf{u}_i$, then the following properties hold:

- The transpose of the matrix, L^T has the same eigenvalues and eigenvectors as the matrix L.
- The matrix $L^2 = LL$ has the same eigenvectors as L and has the eigenvalues λ_i^2.
- If the inverse of the matrix exist, L^{-1}, then its eigenvectors are the same as those of L and its eigenvalues are $1/\lambda_i$.
- If $B = a_0 + a_1 L + a_2 L^2 + \ldots + a_p L^p$, is a polynomial in L, then it has the same eigenvectors as L and its eigenvalues are $a_0 + a_1\lambda + a_2\lambda^2 + \ldots + a_p\lambda^p$.
- If L is a real symmetric matrix, $(L^T = L)$, then eigenvectors corresponding to distinct eigenvalues are mutually orthogonal, and all its eigenvalues are real.

13.4 Spectral Radius

The **spectral** radius, $\rho(L)$, of a matrix L is given by its largest eigenvalue in magnitude:

$$\rho(L) = \max_i(|\lambda_i|) \tag{13.6}$$

The spectral radius is the lowest for all compatible matrix norms. The proof is simple. Let the matrix L have the eigenvalues and eigenvectors λ_i, $\mathbf{u}_i$. Then

$$\underbrace{\|L\mathbf{u}_i\|}_{\|\lambda_i\mathbf{u}_i\| = |\lambda_i|\,\|\mathbf{u}_i\|} \leq \|L\|\,\|\mathbf{u}_i\| \tag{13.7}$$

and hence $|\lambda_i| \leq \|L\|$. This result holds for all eigenvalues, and in particular for the eigenvalue with the largest magnitude, i.e. the spectral radius:

$$\rho(L) = \max_i(|\lambda_i|) \leq \|L\| \tag{13.8}$$

The above result holds for all matrix norms. For the case of the 1 or ∞-norms, we have **Gershgorin** first theorem, that the spectral radius is less then the largest sum of the absolute values of the row or columns entries, namely: $\rho(L) \leq \|L\|_1$ and $\rho(L) \leq \|L\|_\infty$.

Gershgorin's second theorem puts a limit on where to find the eigenvalues of a matrix in the complex plane. Each eigenvalue is within a circle centered at $l_i i$, the diagonal entry of the matrix, and of radius R:

$$|\lambda_i - l_{i,i}| \leq \sum_{j=1,j\neq i}^{N} |l_{ij}| = |l_{i,1}| + |l_{2,i}| + \dots |l_{i,i-1}| + |l_{i,i+1}| + \dots |l_{i,N}| \tag{13.9}$$

13.5 Eigenvalues of Tridiagonal Matrices

A tridiagonal matrix is a matrix whose only non-zero entries are on the main diagonal, and on the first upper and lower diagonals.

$$L = \begin{pmatrix} l_{1,1} & l_{1,2} & 0 & 0 & 0 \\ l_{2,1} & l_{2,2} & l_{2,3} & 0 & \vdots \\ 0 & l_{3,2} & l_{3,3} & l_{3,4} & 0 \\ \vdots & & & \ddots & \\ 0 & \cdots & 0 & l_{N,N-1} & l_{N,N} \end{pmatrix} \tag{13.10}$$

Tridiagonal matrices occur often in the discretization of one-dimensional partial differential equations. The eigenvalues of some of these tridiagonal matrices can sometimes be derived for constant coefficients PDE. In this case the matrix takes the form:

$$L = \begin{pmatrix} a & b & & & & \\ c & a & b & & & \\ & c & a & b & & \\ & & & \ddots & & \\ & & & c & a & b \\ & & & & c & a \end{pmatrix} \tag{13.11}$$

The eigenvalues and corresponding eigenvectors are given by

$$\lambda_i = a + 2\sqrt{bc}\cos\frac{i\pi}{N+1}, \quad u_i = \begin{pmatrix} \left(\frac{c}{b}\right)^{\frac{1}{2}} \sin\frac{i\pi}{N+1} \\ \left(\frac{c}{b}\right)^{\frac{2}{2}} \sin\frac{2i\pi}{N+1} \\ \vdots \\ \left(\frac{c}{b}\right)^{\frac{j}{2}} \sin\frac{ji\pi}{N+1} \\ \vdots \\ \left(\frac{c}{b}\right)^{\frac{N}{2}} \sin\frac{Ni\pi}{N+1} \end{pmatrix} \tag{13.12}$$

for $i = 1, 2, \dots, N$.

For periodic partial differential equations, the tridiagonal system is often of the form

$$L = \begin{pmatrix} a & b & & & & c \\ c & a & b & & & \\ & c & a & b & & \\ & & & \ddots & & \\ & & & c & a & b \\ b & & & & c & a \end{pmatrix} \tag{13.13}$$

The eigenvalues are then given by:

$$\lambda_i = a + (b+c)\cos\frac{2\pi(i-1)}{N} + i(c-b)\sin\frac{2\pi(i-1)}{N} \tag{13.14}$$

A final useful result is the following. If a real tridiagonal matrix has either all its off-diagonal element positive or all its off-diagonal element negative, then all its eigenvalues are real.

Chapter 14

Fourier series

In this chapter we explore the issues arising in expressing functions as Fourier series.

14.1 Continuous Fourier series

As shown earlier, the set of functions

$$\phi_k(x) = e^{ikx}, k = 0, \pm 1, \pm 2, \dots \tag{14.1}$$

forms an orthogonal basis function over the interval $[-\pi, \pi]$:

$$\int_{-\pi}^{\pi} \phi_j \overline{\phi}_k \mathrm{d}x = 2\pi\delta_{jk} \tag{14.2}$$

where the overbar denotes the complex conjugate. The Fourier series of the function u is defined as:

$$Su = \sum_{k=-\infty}^{\infty} \hat{u}_k \phi_k. \tag{14.3}$$

It represents the formal expansion of u in the Fourier orthogonal system. The Fourier coefficients $\hat{u}_k$ are:

$$\hat{u}_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} u(x) e^{-ikx} \, \mathrm{d}x. \tag{14.4}$$

It is also possible to re-write the Fourier series in terms of trigonometric functions by using the identities:

$$\cos\theta = \frac{e^{i\theta} + e^{-i\theta}}{2}, \;\; \sin\theta = \frac{e^{i\theta} - e^{-i\theta}}{2i}, \tag{14.5}$$

The Fourier series become:

$$Su = a_0 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx) \tag{14.6}$$

The Fourier coefficients of the trigonometric series are related to those of the complex exponential series by

$$\hat{u}_k = a_{|k|} - i b_{|k|}. \tag{14.7}$$

If u is a real-valued functions, a_k and b_k are real numbers, and $\hat{u}_{-k} = \overline{\hat{u}}_k$. Often it is unnessary to use the full Fourier expansion. If u is an even function, i.e. $u(-x) = u(x)$ then all the sine coefficients, b_k, are zero, and the series becomes what is called a cosine-series. Likewise, if the function u is odd, $u(-x) = -u(x)$, $b_k = 0$ and the expansion becomes a sine-series.

14.2 Discrete Fourier series

In practical applications, numerical methods based on Fourier series cannot be implemented in precisely the same way as suggested in the earlier section. For example, the Fourier coefficients of an arbitrary function may be too difficult to calculate using equation 14.4, either the integral is too complicated to evaluate analytically, or the function u is only known at a discrete set of points; thus an efficient way must be found to convert the function u from physical space to spectral space. Furthermore, nonlinearities can complicate significantly the application of spectral methods. The key to overcoming these difficulties is the use of a *discrete* Fourier series and its associated discrete Fourier transform.

Consider the set of N points $x_j = 2\pi j/N$, for $j = 0, 1, \dots, N-1$. We define the *discrete Fourier coeffcients* of a complex valued function u in $0 \le x \le 2\pi$ as

$$\tilde{u}_n = \frac{1}{N}\sum_{j=0}^{N-1} u_j e^{-inx_j}, \quad n = -N/2, -N/2+1, \dots, N/2-1. \tag{14.8}$$

Notice that $\tilde{u}_{\pm N/2} = \sum_{j=0}^{N-1} u_j e^{\mp i\frac{N}{2}\frac{2\pi j}{N}} = \sum_{j=0}^{N-1}(-1)^j u_j$, and so $\tilde{u}_{\frac{N}{2}} = \tilde{u}_{-\frac{N}{2}}$. The inversion of the definition can be done easily, multiply equation 14.8 by e^{inx_k} and sum the resulting series over n to obtain:

$$\sum_{n=0}^{N-1} \tilde{u}_n e^{inx_k} = \sum_{n=0}^{N-1}\frac{1}{N}\sum_{j=0}^{N-1} u_j e^{in(x_k - x_j)} = \frac{1}{N}\sum_{j=0}^{N-1} u_j \left[\sum_{n=0}^{N-1} e^{in(x_k-x_j)}\right] \tag{14.9}$$

The last sum can be written as as geometric series with factor $e^{i(x_k-x_j)}$. Its sum can then be expressed analytically as:

$$\sum_{n=0}^{N-1} e^{in(x_k-x_j)} = 1 + e^{i(x_k-x_j)} + e^{i2(x_k-x_j)} + \dots + e^{i(N-1)(x_k-x_j)} \tag{14.10}$$

$$= \frac{1 - e^{iN(x_k-x_j)}}{1 - e^{i(x_k-x_j)}} = \frac{1 - e^{i2\pi(k-j)}}{1 - e^{i2\pi(k-j)/N}} \tag{14.11}$$

There are two cases to consider: if $k = j$ then all the terms in the series are equal to 1 and hence sum up to N, if $k \ne j$ then the numerator in the last fraction is equal to 0. Thus we can write $\sum_{n=0}^{N-1} e^{in(x_k-x_j)} = N\delta_{jk}$. The set of functions e^{inx} is said to be *discretely orthogonal.* This property can be substituted in equation 14.9 to obtain the inversion formula:

$$\sum_{n=0}^{N-1} \tilde{u}_n e^{inx_k} = \sum_{j=0}^{N-1} u_j \delta_{jk} = u_k \tag{14.12}$$

14.2.1 Fourier Series For Periodic Problems

The structure of the discrete Fourier series can be justified easily for periodic problems. In the continuous case they would take the form

$$u(x) = \sum_{n=-\infty}^{\infty} \hat{u}_n e^{-ik_n x} \tag{14.13}$$

Here, and unlike the integral form, the wavenumbers k are not continuous but are quantizied on account of periodicity. If the domain is $0 \le x \le a$ then the wavenumbers are given by

$$k_n = \frac{n\pi}{a}, \quad n = 0, 1, 2, 3, \dots \tag{14.14}$$

In the discrete case the domain would be divide into an equally-spaced set of points $x_j = j\Delta x$ with $\Delta x = a/N$, where $N+1$ is the number of points. The discrete Fourier series would then take the form

$$u(x_j) = u_j = \sum_{n=-N_{\max}}^{N_{\max}} \hat{u}_n e^{-ik_n x_j} \tag{14.15}$$

where $N_{\max}$ is the maximum wave mode that can be represented on the discrete grid. Note that $k_n x_j = \frac{2\pi n}{a} jaN = \frac{2\pi n j}{N}$. Since the smallest wavelength is $2\Delta x$ the maximum wavenumber is then

$$k_{\max} = \frac{2\pi}{2\Delta x} = \frac{2\pi N_{\max}}{a} \text{ so that } N_{\max} = \frac{a}{2\Delta x} = \frac{N}{2} \tag{14.16}$$

Furthermore we have that

$$e^{-ik_{\pm N_{\max}} x_j} = e^{\mp i \frac{2\pi}{N}\frac{N}{2} j} = e^{\pm i\pi j} = (-1)^j \tag{14.17}$$

Hence the two waves are identical and it is enough to retain the amplitude of one only. The discrete Fourier series can then take the form:

$$u(x_j) = u_j = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{u}_n e^{-i\frac{2\pi n}{N}} \tag{14.18}$$

We now have the parity between the N degrees of freedom in physical space u_j and the N degrees of freedom in Fourier space. Further manipulation can reduce the above expression in the standard form presented earlier:

$$\begin{aligned} u_j &= \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \hat{u}_n e^{-i\frac{2\pi n j}{N}} && (14.19) \\ &= \sum_{n=-\frac{N}{2}}^{-1} \hat{u}_n e^{-i\frac{2\pi n j}{N}} + \sum_{n=0}^{\frac{N}{2}-1} \hat{u}_n e^{-i\frac{2\pi n j}{N}} && (14.20) \end{aligned}$$

$$= \sum_{n=-\frac{N}{2}}^{-1} \hat{u}_n e^{-i\frac{2\pi nj}{N}} e^{-i2\pi j} + \sum_{n=0}^{\frac{N}{2}-1} \hat{u}_n e^{-i\frac{2\pi nj}{N}} \tag{14.21}$$

$$= \sum_{n=-\frac{N}{2}}^{-1} \hat{u}_n e^{-i\frac{2\pi(n+N)j}{N}} + \sum_{n=0}^{\frac{N}{2}-1} \hat{u}_n e^{-i\frac{2\pi nj}{N}} \tag{14.22}$$

$$= \sum_{n=\frac{N}{2}}^{N-1} \hat{u}_{n-N} e^{-i\frac{2\pi nj}{N}} + \sum_{n=0}^{\frac{N}{2}-1} \hat{u}_n e^{-i\frac{2\pi nj}{N}} \tag{14.23}$$

$$= \sum_{n=0}^{N-1} \tilde{u}_n e^{-i\frac{2\pi nj}{N}} \tag{14.24}$$

where the new tilded coefficients are related to the old (hatted) coefficients by

$$\begin{cases} \tilde{u}_n = \hat{u}_n & 0 \le n \le \frac{N}{2} - 1 \\ \tilde{u}_n = \hat{u}_{n-N} & \frac{N}{2} - 1 \le n \le N - 1 \end{cases} \tag{14.25}$$

Sine transforms

For problems that have homogeneous Dirichlet boundary conditions imposed, one expands the solution in terms of sine functions:

$$u_j = \sum_{n=1}^{N-1} \hat{u}_n \sin\frac{n\pi x_j}{L} = \sum_{n=1}^{N-1} \hat{u}_n \sin\frac{n\pi j\Delta x}{L} = \sum_{n=1}^{N-1} \hat{u}_n \sin\frac{n\pi j}{N} \tag{14.26}$$

It is easy to verify that $u_0 = u_N = 0$ no matter the values of $\hat{u}_n$. To derive the inversion formula, multiply the above sum by $\sin\frac{m\pi j}{N}$ and sum over j to get:

$$\begin{aligned} \sum_{j=1}^{N-1} u_j \sin\frac{m\pi j}{N} &= \sum_{j=1}^{N-1} \left(\hat{u}_n \sum_{n=1}^{N-1} \sin\frac{n\pi j}{N}\right) \sin\frac{m\pi j}{N} \\ &= \sum_{n=1}^{N-1} \hat{u}_n \left(\sum_{j=1}^{N-1} \sin\frac{n\pi j}{N} \sin\frac{m\pi j}{N}\right) \\ &= \sum_{n=1}^{N-1} \frac{\hat{u}_n}{2} \left(\sum_{j=1}^{N-1} \cos\frac{(n-m)\pi j}{N} - \cos\frac{(n+m)\pi j}{N}\right) \\ &= \sum_{n=1}^{N-1} \frac{\hat{u}_n}{4} \left(\sum_{j=1}^{N-1} e^{i\frac{(n-m)\pi j}{N}} + e^{-i\frac{(n-m)\pi j}{N}} - e^{i\frac{(n+m)\pi j}{N}} - e^{-i\frac{(n+m)\pi j}{N}}\right) \end{aligned} \tag{14.27}$$

The terms in the inner series have the form

$$S_k = r + r^2 + \ldots + r^k = r(1 + r + \ldots + r^{k-1}) = r\frac{r^k - 1}{r - 1} = \frac{r^{k+1} - r}{r - 1} = \frac{r^{k+1} - 1}{r - 1} - 1 \tag{14.28}$$

which for our trigonometric series become

$$S(p) = \sum_{j=1}^{N-1} e^{i\frac{p\pi j}{N}} = \frac{e^{i\frac{p\pi N}{N}} - e^{i\frac{p\pi}{N}}}{e^{i\frac{p\pi}{N}} - 1} = \frac{(-1)^p - e^{i\frac{p\pi}{N}}}{e^{i\frac{p\pi}{N}} - 1} \tag{14.29}$$

with $p = \pm(n \pm m)$. Note also that $S(0) = (N-1)$. Furthermore we have that

$$\begin{aligned} S(p) + S(-p) &= \frac{(-1)^p - e^{i\frac{p\pi}{N}}}{e^{i\frac{p\pi}{N}} - 1} + \frac{(-1)^p - e^{-i\frac{p\pi}{N}}}{e^{-i\frac{p\pi}{N}} - 1} && (14.30) \\ &= \frac{(-1)^p \left[2\cos\frac{p\pi}{N} - 2\right] - \left[2 - 2\cos\frac{p\pi}{N}\right]}{2 - 2\cos\frac{p\pi}{N}} && (14.31) \\ &= -1 - (-1)^p && (14.32) \end{aligned}$$

We can now estimate the four different sums:

$$S(n-m) + S(-n+m) - S(n+m) - S(-n-m) = (-1)^{n+m} - (-1)^{n-m} = 0 \tag{14.33}$$

since if $n \pm m$ are even if n and m are both either odd or even, and $n \pm m$ is odd if one is odd and the other even. For the case where $m = n$ we have

$$S(n-m) + S(-n+m) - S(n+m) - S(-n-m) = 2(N-1) + 1 + (-1)^{2n} = 2N \tag{14.34}$$

In compact notation we would write

$$S(n-m) + S(-n+m) - S(n+m) - S(-n-m) = 2N\delta_{nm} \tag{14.35}$$

replacing the above expression in the discrete inner product we get

$$\hat{u}_n = \frac{2}{N} \sum_{j=1}^{N-1} u_j \sin\frac{\pi n j}{N} \tag{14.36}$$

Chapter 15

Programming Tips

15.1 Introduction

The implementation of numerical algorithm requires familiarity with a number of software packages and utilities. Here is a short list of the minimum required to get started in programming:

1. Basics of the Operating System such as file manipulation. The RSMAS library has *UNIX: the basics*. The library has also a bunch of electronic books on the subject. Two titles I came across are: *Unix Unleashed*, and *Learning the Unix Operating System: Nutshell Handbook*

2. Text editor to write the computer program. Most Unix books would have a short tutorial on using either `vi` or `emacs` for editing text files. There are a number of simpler visual editors too such as jed. Web sites for `vi` or its close cousin `vim` are:

   - `http://www.asu.edu/it/fyi/dst/helpdocs/editing/vi/`
     This is actually a very concise and fast introduction to `vi`. Start with it and then go to the other web sites for more in-depth information.
   - `http://docs.freebsd.org/44doc/usd/12.vi/paper.html`
   - `http://www.vim.org/`

   To learn about `emacs`, and its visual counterpart, `xemacs`, visit

   - `http://www.math.utah.edu/lab/unix/emacs.html`
     This seems like a good and brief introduction so that you can be editing files with simple commands.
   - `http://cmgm.stanford.edu/classes/unix.emacs.html`
   - `http://www.lib.chicago.edu/keith/tcl`course/emacs-tutorial.html-
   - `http://www.xemacs.org/`
     This is the Grapher User Interface version of `emacs`. It is much like notepad in WINDOWS in that the commands can be entered visually.

3. Knowledge of a programming language. Fortran is still the preferred language for numerical programming, although C and C++ have been used in certain applications requiring sophisticated data structures. Section 15.2 has an example to introduce various elements of the language.
4. Compiler to turn the program into machine instructions. Section 15.3 discusses compiler issues and how to use the compiler options in helping you to track errors in the coding.
5. Debugger to track down bugs in a computer program.
6. Visualization software to plot the results of computations.
7. Various mathematical libraries such as LAPACK for linear algebra routines and FFTPACK for Fast Fourier Transform.

15.2 Fortran Example

The following is a simple fortran 90 code that makes use of simple language syntax such as declaring variable, looping, opening a file, and writing data out. You should type in the program to practice the editor commands, compile it and run it. Then use a visualization package to plot the results.

```
!
! This is a sample fortran program to introduce the language.
! It divides the interval [-1 1] into M points and computes the
! functions sin(x) and cos(x).  The results are written to the
! terminal and to a file called waves.dat
!
! Comments are marked with an exclamation mark; the compiler
! ignores all characters after the "!" sign.
!
! A fortran statement that does not fit into a single line can
! be continued on the next one by terminating it with a "&" sign.
program waves           ! name of program unit

  implicit none         ! prevents compiler from assigning default types
                        ! and forces user to declare every variable

!.Variable Declaration starts here
  integer, parameter :: M=21 ! declares the value of a constant
                             ! that does not change throughout
                             ! the calculation
  integer :: i               ! declares an integer counter

  real, parameter :: xmin=-1.0, xmax=1.0 ! single precision constants
  real :: f(M)               ! real array with M entries
  real :: pi,x,y,dx
!.End of Variable Declaration

!            Executable statements are below


! Unit 6 is the terminal also called stdout
  write(6,*)'Hello World'

! Open a file to write out the data.
  open(unit=9,           & ! file unit is number 9
       file='waves.out', & ! output file name is waves.out
       form='formatted', & ! data written in ASCII
       status='unknown', & ! create file if it does not exit already
       action='write')     ! file meant for writing

  pi = 2.0*asin(1.0)    ! initialize pi
  dx = (xmax-xmin)/real(M-1)    ! grid-size
  do i = 1,M            ! counter: starts at 1, increments by 1 and ends at M
!...indent statements within loop for clarity
    x = (i-1)*dx + xmin ! location on interval
    y = sin(pi*x)       ! compute function 1
    f(i) = cos(pi*x)    ! compute function 2
    write(6,*) x,y      ! write two columns to terminal
    write(9,*) x,y,f(i) ! write three columns to file
  enddo                 ! end of do loop must be marked.

  close(9)              ! close file (optional)

  write(6,*)'Done'

  stop
end program waves
!
!
! Compiling the program and creating an executable called waves:
!      $ f90 waves.f90 -o waves
! If "-o waves" is ommited the executable will be called a.out
! by default.  The fortan 90 compiler (f90) may have a different name
! on your system. Possible alternatives are pgf90 (Portland Group
! compiler), ifc (Intel Fortran Compiler), and xlf90 (on IBMs).
!
!
! Running the program
!      $ waves
```

```
!
!
! Expected Terminal output is:
! Hello World
!  -1.000000      8.7422777E-08
! -0.9000000     -0.3090170
! -0.8000000     -0.5877852
! -0.7000000     -0.8090170
! -0.6000000     -0.9510565
! -0.5000000      -1.000000
! -0.4000000     -0.9510565
! -0.3000000     -0.8090171
! -0.2000000     -0.5877852
! -9.9999964E-02 -0.3090169
!  0.0000000E+00  0.0000000E+00
!  0.1000000      0.3090171
!  0.2000000      0.5877854
!  0.3000001      0.8090171
!  0.4000000      0.9510565
!  0.5000000       1.000000
!  0.6000000      0.9510565
!  0.7000000      0.8090169
!  0.8000001      0.5877850
!  0.9000000      0.3090170
!   1.000000     -8.7422777E-08
! Done
!
!
! Visualizing results with matlab:
!      $ matlab
!> z = load('waves.out');   % read file into array z
!> size(z)                  % get dimensions of z
!> plot(z(:,1),z(:,2),'k')  % plot second column versus first in black
!> hold on;                 % add additional lines
!> plot(z(:,1),z(:,3),'r')  % plot third column versus first in red
!> xlabel('x');             % add labels to x-axis.
!> ylabel('f(x)');          % add labels to y-axis.
!> title('sine and cosine curves');   % add title to plot
!> legend('sin','cos',0)    % add legend
!> print -depsc waves       % save results to a color
!>                          % encapsulated postscript file
!>                          % called waves.eps. The extension
!>                          % eps will be added automatically.
! Viewing the postscript file:
!      $ ghostscript waves.eps
```

```
!
! Printing the file to color printer
!     $ lpr -Pmpocol waves.eps
```

15.3 Debugging and Validation

. Errors are invariably introduced when implementing a numerical algorithm in a computer code. There are actually two kinds of errors to watch for:

1. **Algorithmic Errors**: are conceptual in nature and are independent of the actual computer code. These kinds of errors can usually be studied theoretically when the algorithm is devised. Examples include stability and convergence of a numerical scheme to its mathematically continuous form. It is usually very hard to solve algorithmic problems with computer experimentation. The latter can only be a guide and/or a confirmation of theory.

2. **Programming Errors**: are introduced when translating an algorithm into actual computer code. These errors are often referred to as bugs and there are techniques that makes tracking them simpler. This is what we will concentrate on in this section.

15.3.1 Programming Rules

The following are rules of thumb devised to help a programmer write "good" code. The primary advice is to write clear readable code that is easy to validate and maintain. Other advice towards that goal are:

- Write modular programs with clearly defined functional unit. By separating the different steps of an algorithm, it becomes easier to "abstract" the code, and to make connections with the theory.

- Each modular unit should be validated. By gaining confidence in the basic working of the building units it becomes easier to cobble together more complicated programs. Moreover, some basic units can be reused in new programs without having to rewrite them.

- Comment the program to explain the different steps. Choose meaningfull names for variables. Document the input and output of subroutines and their basic tasks.

- Write clear code, and do not worry about efficiency in either CPU or memory. Modern computers are vastly superior to the ones common during the early years of computing. Memory and CPU speed constraints forced programmers to use coding tricks that obfuscated the code.

- Improve the readability of the code. Do not be afraid to leave white spaces. Indent do loops, logical statements, and functional units so they are easy to identify.

- Last but not least make sure you are using the appropriate algorithm for what you intend to do.

15.3.2 Coding tips and compiler options

If you write programs in the "right" way, you will have few syntax errors, a good compiler (called with the right options) will flag them for you, and you will correct them easily. It is important to emphasize that most compilers are rather lenient by default, and must be invoked with special options in order to get many useful warnings.

Some of these compiler options enable compile-time checks, i.e. the compiler will spot errors during the translation of the code to machine language and prior to executing the program. Examples include: disabling implicit type declarations, and flagging standard language violations. Others options enable various run-time checks (when the code is actually executed) such as: checking array bounds, trapping floating-point exceptions, and special variable initializations that are supposed to produce floating-point exceptions if uninitialized variables are used.

The following is a list of programming tips that will help minimize the number of bugs in the code.

- Do not used implicitly declared types. In fortran variables whose name start with the letters `i,j,k,l,m,n` are integers by default and all others are reals by default. Use the statement `implicit none` in your code to force the compiler to list every undeclared variable, and to catch mistyped variables. It is also possible to do that using compiler options (see the manual pages for the specific options, on UNIX machines it is usually `-u`).

- Make sure every variable is initialized properly before using it, and do not assume that the compiler does it automatically. Some compiler will allow you to initialize variables to a bogus value, a Nan (short for not a number, so that the code trips if that variable is used in an operation before overwriting the Nan.

- Use the `include header.h` statement to make sure that common blocks are identical across program units. The common declaration would thus go into a separate file (`header.h`) which can be subsequently included in other subroutines without retyping it.

- Check that the argument list of a `call` statement matches in type and number the argument list of the subroutine declaration.

- **Remarks on the pitfalls list** The improved syntax of Fortran 90 eliminates some common programming pitfalls. Note that a Fortran 90 compiler will often only be able to help you if you make use of the stricter checking features of the new standard:

  1. IMPLICIT NONE
  2. Explicit interfaces
  3. INTENT attributes
  4. PRIVATE attributes for module-wide data that should not be accessible to the outside

- Use list files to catch compiler report. When compiling the compiler throws rapidly a list of errors at you, being out of context they are hard to understand, and even harder to remember when you return to the editor. Using two windows one for editing and one for debugging is helpful. You can also ask the compiler to generate a LIST FILE, that you can look at (or print) in a separate window.

- Use modules and interface to double check the argument lists of calling subroutines.

15.3.3 Run time errors and compiler options

Some bugs cannot be caught at compile time and produce errors only when the code is executed. The compiler switches can help you catch some common bugs. Here is again a list of tips.

1. Do not optimize the code in the first run. Rather compile it with a debugging option (usually `-g`) to produce trace back information. The program can then let you know the statement number that caused the fatal error.

2. Do array bound checking. The code will crash if you are trying to access memory beyond that available for an array. The common flag for this is `-C` but changes from compiler to compiler. This flag will slow down the performance of the code. However, you first concern should be a correct code rather then fast code.

3. Some floating point operations can result in a Nan. The code should stop then and issue an error report. Various switches trap different floating point exceptions. You want to catch division by zero, overflows (where the number is too large to be represent in the machine precision), underflows (similar but for very small numbers). Underflows are not as problematic as overflows and can usually be ignored. Check the manual for the right compiler switches.

4. Test routines individually to check if they are working according to their specification. Try first "trivial" cases where you know the answers to check the results you get.

5. Use print statements liberally to spot check the value of variables.

6. Use a symbolic debugger to trace the execution of your program.

The first rule about debugging is staying cool, treat the misbehaving program as an intellectual exercise, or as a detective work.

1. You got some strange results that made you think there is a bug. Think again, are you sure they are not the correct output for some special input?

2. If you are not sure what causes the bug, DON'T try semi-random code modifications, that's seldom works. Your aim should be to gather as much as possible information!

3. If you have a modular program, each part does a clearly defined task, so properly placed 'debug statements' can ISOLATE the malfunctioning procedure/code-section.
4. If you are familiar with a debugger use it, but be careful not to be carried away by the many options and start playing.

15.3.4 Some common pitfalls

- **Data-types**
  1. Using implicit variable declarations
  2. Using a non-intrinsic function without a type declaration
  3. Incompatible argument lists in `call` and the routine definition
  4. Incompatible declarations of a common block
  5. Using constants and parameters of incorrect (smaller) type
  6. Assuming that untyped integer constants get typed properly
  7. Assuming intrinsic conversion-functions take care of result type
- **Arithmetic**
  1. Using constants and parameters of incorrect type
  2. Uncareful use of automatic type promotions
  3. Assuming that dividing two integers will give a floating-point result.
  4. Assuming integer exponents (e.g. `2**`(3)-) are computed as floating-point numbers
  5. Using floating-point comparisons tests, `.EQ.` and `.NE.` are particularly risky
  6. Loops with a `REAL` or `DOUBLE PRECISION` control variable
  7. Assuming that the `MOD` function with `REAL` arguments is exact
  8. Assuming real-to-integer assignment will work in all cases
- **Miscellaneous**
  1. Code lines longer than the allowed maximum
  2. Common blocks losing their values while the program runs
  3. Aliasing of dummy arguments and common block variables, or other dummy arguments in the same subprogram invocation
  4. Passing constants to a subprogram that modifies them
  5. Bad DO-loop parameters (see the DO loops chapter)
  6. TABs in input files - what you see is not what you get!
- **General**

1. Assuming variables are initialized to zero
2. Assuming variables keep their value between the execution of a RETURN statement and subsequent invocations
3. Letting array indexes go out of bounds
4. Depending on assumptions about the computing order of subexpressions
5. Assuming short-circuit evaluation of expressions
6. Using trigonometric functions with large arguments on some machines
7. Inconsistent use of physical units

Chapter 16

Debuggers

A debugger is a very useful tool for developing codes, and in finding and fixing bugs introduced in the code development. Most compiler providers include a debugger as part of their software bundle. Since we are using the Portland Group compiler for the class, the discussion here will center on its debugger primarily, even though a lot of the information applies equally to other compiler/debugger systems. Check the manual for the specific compiler/debugger in case of problems.

16.1 Preparing the code for debugging

A debuggable executable must have extra information embedded in it for debugging purposes. The debugger then makes use of this information to report the values of variables, and allow the programmer to follow the execution of the program line by line. Compiler options must be used to instruct the compiler to embed this information in the object files. Here is a useful subset of these options pertinent to the PG compiler:

1. `-g` Generate information for the debugger, this is necessary on most computers to enable the printing of useful debugging information. Avoid using any optimization in the code development phase (so avoid the `-O` options). Occasionaly you want to debug with optimization on, the `-gopt` would then be necessary.

2. `-C` or `-Mbounds`: generate code to check array bounds

3. `-Ktrap=list-of-options`: helps trap floating point problems. The list is a comma separates list of strings that controls which floating point operations to catch. These include:

   (a) `-Ktrap=divz`: trap divide by zero.
   (b) `-Ktrap=ovef`: trap floating point overflows.
   (c) `-Ktrap=unf`: trap underflow (number too small to be representable).
   (d) `-Ktrap=inv`: trap invalid operands (e.g. square root of negative numbers).

   A commonly useful subset is to set -Ktrap=divz,inv,ovf

4. `-Mchkptr`: Check if code mistakenly references NULL pointers.

5. `-Mchkstk`: Check the stack for available space upon entry to and before the start of a parallel region. Useful when many private variables are declared.

Compiler	PGI	Intel	gcc	IBM	Pathscale
Enable Debugging	-g	-g	-g	-g	-g
Bounds Checking	-C	-CB	-C	-C	
Uninitialized variables	NA	-ftrapuv		-qinitauto	
Floating point trap	-Ktrap	-fpe0	-ffpe-trap	-qflttrp	
Floating point stack	-Mchkstk	-fpstkchk			

16.2 Running the debugger

The command `pgdbg executable` wil run the code `executable` under the control of the debugger. The default is to start the debugger under a Graphical User Interface (GUI). If you have a slow connection, the GUI can slow down your work because of the graphics overhead. Most debugger include a command, text line interface. For PG it is invoked with the `-text` option". The command will start the debugger which in turn will "load" the executable and all its pertinent information. The debugger then passes control to the user and waits for further instructions. Most debuggers come with an on-line help facility to allow users to learn the command list interactively, the command to initiate is usually called `help`. Here we list briefly a subset of these commands and that have proved to be most useful for the author.

The debugger GUI is fairly intuitive; it will open up at least one window to display source lines, and another one for I/O operations. The GUI and text versions of the debugger can be controlled via command lines. Here we will cover some the most useful one, and we refer the user to the manual for further information.

1. `run` Will cause the code to start executing.

2. `list 10,40` will list the lines 10,40, `list` without argument will list lines 10 to 20 from the current statement.

3. `stop at xyz` will put a breakpoint at line xyz. Execution will stop at the line so the user can examine variables.

4. `print sn` will print the value of the variable sn.

5. `assign sn=expression` assigns the expression `expression` to the variable sn.

6. `whatis sn` will report the data type of variable sn.

7. `where` will report the current statement.

8. `step` will execute the next source line, including stepping into a function or subroutine.

9. `stepi` will execute a single intruction (as opposed to the entire source line. step ¡count¿ will execute ¡count¿ instructions.

10. `display` list the expressions being printed at breakpoints. `display <exp1>,<exp2>` prints `<exp1>` and `<exp2>` at every breakpoint.

11. `next` will cause the debugge to skip over a function or subroutine, i.e. executing it in its entirety.

12. `continue` will cause the execution to resume from the point it stopped to the next breakpoint.

Bibliography

Arakawa, A., 1966. Computational design for long term numerical integration of the equations of fluid motion: two-dimensional incompressible flow. part i. Journal of Computational Physics 1 (1), 119–143.

Arakawa, A., Hsu, Y.-J., 1981. Energy conserving and potential-enstrophy dissipating schemes for the shallow-water equations. Monthly Weather Review 118, 1960–1969.

Arakawa, A., Lamb, V. R., 1977. Computational Design of the Basic Dynamical Processes of the UCLS general circulation model. Vol. 17. Academic Press, New York, p. 174.

Arakawa, A., Lamb, V. R., 1981. A potential enstrophy and energy conserving scheme for the shallow water equations. Monthly Weather Review 109, 18–36.

Balsara, D. S., Shu, C.-W., 2000. Monotonicity preserving weighed essentially non-oscillatory schemes with increasingly high order of accuracy. Journal of Computational Physics 160, 405–452.

Boris, J. P., Book, D. L., 1973. Flux corrected transport, i: Shasta, a fluid transport algorithm that works. Journal of Computational Physics 11, 38–69.

Boris, J. P., Book, D. L., 1975. Flux corrected transport, ii: Generalization of the method. Journal of Computational Physics 18, 248.

Boris, J. P., Book, D. L., 1976. Flux corrected transport, iii: Minimum error fct algorithms. Journal of Computational Physics 20, 397.

Boyd, J. P., 1989. Chebyshev and Fourier Spectral Methods. Lecture Notes in Engineering. Springer-Verlag, New York.

Butcher, J. C., 1987. The Numerical Analysis of Ordinary Differential Equations. John Wiley and Sons Inc., NY.

Canuto, C., Hussaini, M. Y., Quarteroni, A., Zang, T. A., 1988. Spectral Methods in Fluid Dynamics. Springer Series in Computational Physics. Springer-Verlag, New York.

Dormand, J. R., 1996. Numerical Methods for Differential Equations, A Computational Approach. CRC Press, NY.

Dukowicz, J. K., 1995. Mesh effects for rossby waves. Journal of Computational Physics 119, 188–194.

Durran, D. R., 1999. Numerical Methods for Wave Equations in Geophysical Fluid Dynamics. Springer, New York.

Finlayson, B. A., 1972. The Method of Weighed Residuals and Variational Principles. Academic Press.

Jiang, C.-S., Shu, C.-W., 1996. Efficient implementation of weighed eno schemes. Journal of Computational Physics 126, 202–228.

Karniadakis, G. E., Sherwin, S. J., 1999. Spectral/*hp* Element Methods for CFD. Oxford University Press.

Leonard, B. P., MacVean, M. K., Lock, A. P., 1995. The flux integral method fo multidimensional convection and diffusion. Applied Mathematical Modelling 19, 333–342.

Shchepetkin, A. F., McWilliams, J. C., 1998. Quasi-monotone advection schemes based on explicit locally adaptive dissipation. Montlhy Weather Review 126, 1541–1580.

Shu, C.-W., 1998. Essentially non-oscillatory and weighed essentially non-oscillatory schemes for hyperbolic conservation laws. Springer, New York, p. 325.

Suresh, A., Huynh, H. T., 1997. Accurate monotonicity preserving schemes with runge-kutta time stepping. Journal of Computational Physics 136, 83–99.

Whitham, G. B., 1974. Linear and Nonlinear Waves. Wiley-Interscience, New York.

Zalesak, S. T., 1979. Fully multidimensional flux-corrected transport algorithms for fluids. Journal of Computational Physics 31, 335–362.

Zalesak, S. T., 2005. The design of flux-corrected transport algorithms for structured grids. In: Kuzmin, D., Löhner, R., Turek, S. (Eds.), Flux-Corrected Transport. Springer, pp. 29–78.