IMPERIAL

Computational Partial Differential Equations (CPDEs) MATH60025/70025

Spring Term 2: 2024-2025

live document

(updated March 18, 2025)

MATH60025/70025: Computational Partial Differential Equations

Lecturer: Dr Shahid Mughal

The lecture notes are an adaptation of earlier notes (2018) on the course developed by Professor J. Mestel, Mathematics Department, Imperial College London.

Email : s.mughal@imperial.ac.uk
Office : Department of Mathematics, Huxley 734
Assessment : 100% by projects
Lectures : 26 + Surgery hours

Module Summary

This module will introduce a variety of computational approaches for solving partial differential equations, focusing mostly on finite difference methods. Students will gain experience implementing the methods and writing/modifying short programs in Matlab or another programming language of their choice. Applications will be drawn from problems arising in areas such as Mathematical Biology and Fluid Dynamics.

Prerequisites : None, other than the core modules from years 1 & 2, but Programming ability in Python is required – MATH50008 module *Partial Differential Equations in Action* will provide some background. You will be expected to produce and/or modify programs in (say) Python or any other language of your choice.

Module Content :
There is some flexibility as to the exact module content and ordering, but the following should be very close:

1. Introduction: How can we solve PDEs on a computer? Finite Difference Methods. Basic types and Classification of PDEs. Well-posedness and the importance of Boundary Conditions.
2. Parabolic equations: Explicit & Implicit Schemes. Maximum principle analysis.
3. Elliptic equations: Iterative methods: How can they be made faster? Jacobi, Gauss-Seidel, relaxation techniques. Multigrid methods and motivation and implementation.
4. Hyperbolic equations: characteristics, up-winding, Lex-Wendroff schemes. Non-reflecting boundary conditions, perfectly matched layers (PML).
5. Combinations, Extensions and Applications: e.g. nonlinearities, advection/diffusion and Navier-Stokes equations. Magnetic Induction and heat transport equations. Domain decomposition, operator splitting.

Intended Learning Outcomes :

- Appreciate the physical and mathematical differences between different types of PDEs;
- Outline a theoretical approach to testing the stability of a given algorithm;
- Determine the order of convergence of a given algorithm;
- Demonstrate familiarity with the implementation and rationale of multigrid methods;
- Develop finite-difference based software for use on research level problems;
- Use numerical methods, as an alternative to analytical approaches, to solve mathematical problems with a physical underpinning.

Programming and Coding Proficiency : This course does **NOT** assess your programming/coding abilities. It is expected that you will already be proficient in a programming language like Python, Fortran, C or Matlab. If you have **NO experience in programming you should think wisely as to whether this course is for you** – it is possible to learn programming as you go along, but you may find this quite challenging particularly due to time constraints of submitting your assessed work.

Philosophy of the Module : Most of the world and universe as we know it, can be described by mathematical tools, usually in a multi-variable context, governed by Partial Differential Equations (PDEs). However, the techniques for solving PDEs analytically are few and most real phenomena are described by very complex coupled sets of PDEs where analytical solutions are simply not possible. Yet nowadays we have incredible computer power, and we can obtain accurate approximations to solutions to most physically relevant problems. An understanding of analytical techniques really should be accompanied by the ability to find numerical solutions when required. This module is important for those considering a future in research, but also for those considering getting a job in a scientific & technical fields (including Finance) – these almost invariably involve solving some form of PDE or sets of PDEs with a computer.

It is moderately easy to write inefficient code which solves simple problems adequately. This skill should not be under-rated - many problems you will encounter can be dealt with effectively by a "quick and dirty" approach. Yet, if you want to be able to solve important problems quickly, or difficult problems at all, a good understanding of numerical techniques is vital, and is a well-valued talent. This is what this module aims to engender. At its conclusion, you should be able to make a decent stab at previously unsolved Research-level problems.

Assessment : The module will be assessed solely by projects. The plan is as follows: A relatively short project, worth 25%will be set early on, and returned to you with comments. You will be committed to completing the module on submission of 25%. A final project will be set in week 8, with an additional project released for the Mastery (MSc, MSci). These projects should be submitted electronically, via Blackboard.

1. **CW 1 :** 25% released 28 January, submission 1pm, 10 February.
2. **CW 2 :** 40% released 14 February, submission 1pm, 3 March.

3. **CW 3 :** 35% released 6 March, submission 1pm, 20 March.
4. **CW 4 :** 20% released 6 March, submission 1pm, 4 April **(Mastery)**.

The assessment for each CW is based upon you solving a given problem having written your code, and followed up by a **detailed technical report** which outlines your findings. You should not expect a high mark, if you only submit your code but fail to submit the written report.

During the module, various Matlab codes will be demonstrated in lectures and released on Blackboard. Most of you will choose to modify these codes for the problems set in the projects, but it is quite permissible to write your programs in any language which runs on the Imperial College machines, e.g. Python, C, Matlab, Fortran.

The projects will involve demonstrating that your codes work to the expected accuracy, obtaining solutions to set problems and discussing the results. The projects may build on one another slightly; you may be able to use your codes from the first two projects as part of the final project. The final project will be at a high level, the kind of problem which borders on Research level. At the end of the module you should be able to solve difficult problems using the various techniques covered.

More topics will be covered in lectures that will/may be of direct of use in the Projects, just as some topics do not get assessed in examinations.

General Comments on Project Modules :
Most Maths modules are assessed mainly by a summer exam. Some memory is required - in 2-3 hours you demonstrate what you have learned from the lectures and the many hours of revision. In project modules, memory is not a factor, and you can spend a very long time on them if you choose. As a result, average marks on project modules tend to be a little higher. Nevertheless, 100% is not achievable without deep understanding, and you should not expect perfection. In the past, some students have spent too long on their projects and neglected revision of other topics – I can only caution you against this. The Senior Tutor may advise you not to attend too many project modules.

Collaboration :
We do not, of course, wish to discourage discussion between you on any of the mathematical and computational issues underlying this module. However, plagiarism considerations are very important in project modules. The projects really must be produced independently and any help you received MUST be acknowledged in your submissions. You must adhere strictly to the plagiarism guidelines you have been given – if you are in any doubt about what is permitted you should seek clarification from the Senior Tutor, Dr Ford. The College penalties are exceptionally severe for breaches and this can jeopardise your entire degree. Please do be sensible about this.

Support Classes :
Every now and again, problem sessions will occur in lectures. There will also be surgery sessions in office hours. The timings of these will be discussed in lectures.

Recommended Books :

See the complete Reading list on Blackboard, but the following are pretty good.

1. LeVeque, Randall J., *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. **(Core)**
2. Tannehill, John C., *Computational fluid mechanics and heat transfer*.
3. Smith, G. D., *Numerical solution of partial differential equations : finite difference methods*.
4. LeVeque, Randall J., *Finite Volume Methods for Hyperbolic Problems*.
5. Toro, Eleuterio F., *Riemann Solvers and Numerical Methods for Fluid Dynamics*, 3rd edition.
6. Ames, William F., *Numerical Methods for Partial Differential Equations* , 2nd edition, Academic press (1977).

Acknowledgment : *Some of the lecture notes comprise material sourced from the above references.*

MATH60025/70025: Computational Partial Differential Equations

Contents

1 Brief Motivation

The entire universe and phenomenon we come across on a daily basis ranging from physical, chemical, biological, mechanical, economical to weather forecasting can be described by some form of systems of Partial Differential Equations (PDEs). Only a limited set of PDEs can be solved exactly. For example, Laplace equation widely arises in physics problems ranging from electrical, magnetic and gravitational potentials, steady-state temperatures, and in hydrodynamics (to name a few). You may recall from earlier courses how to solve the Laplace equation for $u(x,\, y)$ in a rectangle,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \; \in [\, 0 < x < a, \;\; 0 < y < b \,], \tag{1.1}$$

with the boundary conditions

$$u(a, y) = 1 \;\; \text{for} \;\; y \neq (0, b); \quad u(0, y) = u(x, 0) = u(x, b) = 0. \tag{1.2}$$

Using separation of variables and Fourier series, you can show that

$$u = \frac{2}{\pi} \sum_{n=1}^{\infty} \frac{1 - (-1)^n}{n \sinh(n\pi a/b)} \sinh\left(\frac{n\pi x}{b}\right) \sin\left(\frac{n\pi y}{b}\right). \tag{1.3}$$

That's fine, but what does the solution look like? You'll almost certainly want to contour it on a computer. To do this you have to truncate the series to a finite number of terms, and so only plot an approximation to the exact solution, as shown in Figs. 1.1 and 1.2. How many terms in the series are adequate to obtain a desired accuracy? Why is this better than obtaining an approximation to the entire problem on a computer from the start?

The above equation (1.1) may be solvable exactly on square, rectangular or circular boundaries as shown in Fig. 1.3a,b, however if the boundaries or boundary conditions become complex (Fig. 1.3c,d) either the exact analytic solution is not possible (in most cases) or some considerable work has to be done to find the exact analytical solution – which even then may well require some form of numerical computation for the solution to be make sense.

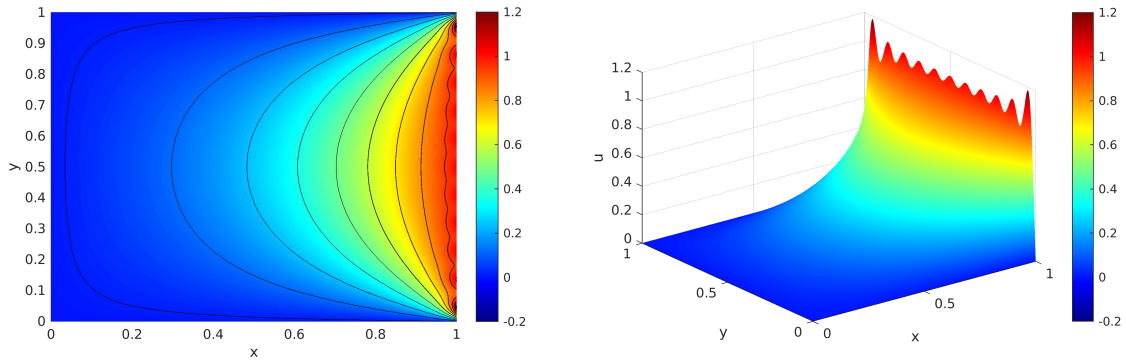

Figure 1.1: Typical solution of Eqn. 1.1 – two-dimensional plot, on truncation of Eqn. 1.3 to a finite number ($n = 21$) of terms.

Numerical solutions can be found for a much wider variety of problems, including systems of **nonlinear** PDEs. Numerical computation is commonplace today in virtually every aspect

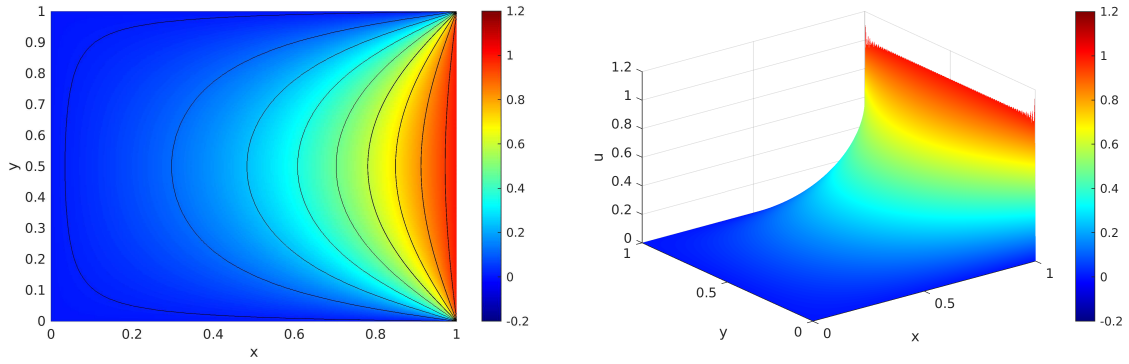

Figure 1.2: Solution of Eqn. 1.1 – surface plot, on truncation of Eqn. 1.3 to a larger but still finite ($n = 161$) terms.

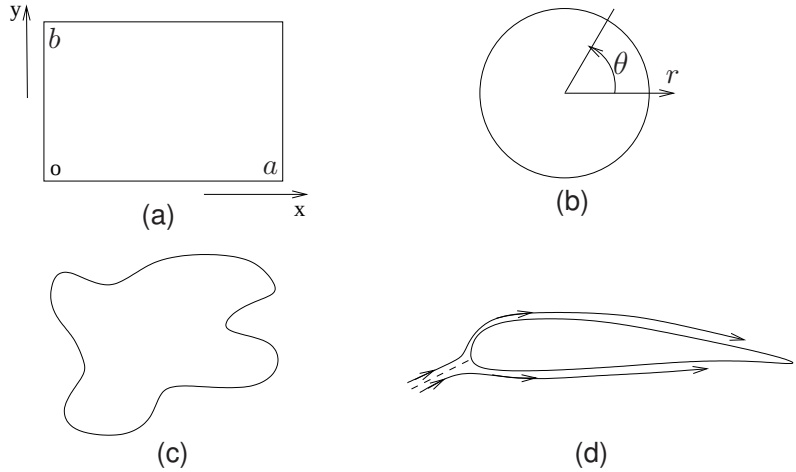

Figure 1.3: Examples of Boundary Value problems (BVPs).

of the world we live in, from engineering, finance, astronomy, physics, artificial intelligence, data analysis, weather prediction and modelling spread of infections - the list is endless. Computers have made possible solutions of scientific and engineering problems of great complexity.

This module aims to take you to the point where you will have the background knowledge to solve a range of PDEs with the aid of a computer. The course objective is to give some of the mathematical and numerical background which allows you to assess and wisely choose the most appropriate numerical technique for the PDE that you may be interested in solving. As you will see, in most cases the numerical technique devised, has to honour the mathematical and physical character of the PDE. There is no single numerical technique which will solve all PDEs that you may come across, rather the skill of a practitioner in the field is to have the necessary background in mathematical and numerical analysis to develop the technique which works best (or optimally) for the problem at hand – in terms of accuracy of the solution to the PDE which satisfies all applied boundary-conditions imposed on the PDE.

2 PDEs - Primer and Definitions

A partial differential equation (PDE) is a functional relation between an unknown function, say u, and its derivatives. We consider systems where there is some sort of interaction between independent variables, which we denote vectorially as $\mathbf{x} = (x, y, z, t)$ say, and the dependent variable u; or a set of dependent variables $\mathbf{u} = (u, v, w, \ldots)$. Here the x, y, z independent variables play the role of spatial variables in a three-dimensional (3D) world, say, and in time-dependent cases we reserve the variable t to play the role of time.

A partial derivative of u with respect to the x independent variable is denoted by

$$\frac{\partial u}{\partial x}, \quad \text{or} \quad u_x.$$

Similarly a partial derivative to second-order is denoted by

$$\frac{\partial^2 u}{\partial x^2}, \quad \text{or} \quad u_{xx}.$$

Hence a *mixed*-derivative u_{xy} implies

$$\frac{\partial^2 u}{\partial x \partial y} \equiv u_{xy}, \quad \text{and} \quad \frac{\partial^3 u}{\partial x^2 \partial y} \equiv u_{xxy} \quad \text{etc.}$$

Order of a PDE

One way to classify PDEs is according to the **order** of the equation. The order is defined to be the order of the highest derivative appearing in the equation. For example

$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = f(x, t),$$

is a **second-order** equation, while

$$\frac{\partial u}{\partial t} + \frac{\partial^4 u}{\partial x^4} = 0$$

is a **fourth-order** equation.

More formally, a PDE is a functional relation of the form

$$\mathcal{F}(\mathbf{x}, u, Du, D^2u, \ldots, D^m u) = 0, \ \mathbf{x} \in \Omega \subset \mathbb{R}^n,$$

where the unknown is the function $u : \Omega \subset \mathbb{R}^n \mapsto \mathbb{R}$ and

$$D^m u = \left| \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}} \right|, \quad \alpha \in \mathbb{N}^n, |\alpha| = m$$

is the set of all derivatives of order m.

Linearity and Nonlinearity of a PDE

PDEs are classified in two groups: **linear** PDEs and **nonlinear** PDEs. For example, the equation

$$\mathcal{F} = x^7 \frac{\partial u}{\partial x} + e^{xy} \frac{\partial u}{\partial y} + \sin^2(x^2 + y^2)u - x^3 = 0$$

is a *first-order* linear equation. The heat or diffusion equation is a *second-order* linear PDE

$$\mathcal{F} = \frac{\partial u}{\partial t} - K\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right) = 0,$$

with K a constant parameter representing either diffusion or heat conductivity. Other examples are

- **Schrodinger equation** in physics (second order, linear)

$$-\frac{h^2}{2m}\nabla^2 u + V(\mathbf{r})u = ih\frac{\partial u}{\partial t}.$$

- **Black-Scholes equation** in finance (second order, linear)

$$\frac{\partial u}{\partial t} + \frac{\sigma^2 x^2}{2}\frac{\partial^2 u}{\partial x^2} + rx\frac{\partial u}{\partial x} - ru = 0.$$

- **Fokker-Plank equation** in reaction-diffusion* problems (second order, linear) or *Fisher-Kolmogorov-Petrovsky-Piskunov's* equation

$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(b(x)\ u) + \frac{\sigma^2(x)}{2}\frac{\partial^2 u}{\partial x^2}.$$

  Fokker-Plank type equations often arise in modelling stochastic phenomenon in ecology, genetics, circuit theory, solid-state and chemical physics, finance and fusion modelling among others.

A simple example of a nonlinear PDE is

$$\mathcal{F} = \left(\frac{\partial^2 u}{\partial t^2}\right)^2 + \left(\frac{\partial^2 u}{\partial x^2}\right)^2 = 0.$$

We further classify nonlinear equations in sub-classes according to the type of nonlinearity. The nonlinearity of an equation is more pronounced when it appears on a higher derivative. We commonly call PDEs :

1. **Semi-linear** – where $\mathcal{F}$ is nonlinear only with respect to u but is linear with respect to all its derivatives, for example

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = u^3.$$

*an interesting feature here is the competing effects of diffusion or *stochastic diffusion* given by the σ^2 parameter and so-called drift or "*stochastic drift*" arising from the $b(x)$ term and their relative magnitudes – termed convection-diffusion in fluid dynamics.

Another example is the 2D *Fisher-Kolmogorov-Petrovsky-Piskunov's* second-order reaction-diffusion type equation

$$\frac{\partial u}{\partial t} - K\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = ru(M-u).$$

or the slightly more complicated *Fokker-Plank* equation

$$\frac{\partial u}{\partial t} + \frac{\sigma^2 x^2}{2}\frac{\partial^2 u}{\partial x^2} + rx\frac{\partial u}{\partial x} - ru + F(u) = 0, \tag{2.1}$$

where $F(u) = u^2$ (say).

More generally, suppose a PDE has the form

$$\sum_{|\alpha|=k} \left(a_\alpha(\mathbf{x})D^\alpha u(\mathbf{x}) + a_o(D^{k-1}u(\mathbf{x}), \ldots, u(\mathbf{x}), \mathbf{x})\right) = 0,$$

the PDE is semi-linear if the coefficient $a_\alpha(\mathbf{x})$ depends on $\mathbf{x}$ only, while a_o can comprise nonlinear combinations of derivatives of u in lower-order.

2. **Quasilinear** – where $\mathcal{F}$ is linear with respect to the highest order derivatives of u only, for example
$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = |\nabla u|^2 u;$$
or the *first-order* inviscid Burger's equation
$$\frac{\partial u}{\partial t} + cu\frac{\partial u}{\partial x} = 0 \text{ with } c \text{ a constant.}$$
More generally, the PDE
$$\sum_{|\alpha|=k} \left(a_\alpha(D^{k-1}u(\mathbf{x}), \ldots, u(\mathbf{x}), \mathbf{x})D^\alpha u(\mathbf{x}) + a_o(D^{k-1}u(\mathbf{x}), \ldots, u(\mathbf{x}), \mathbf{x})\right) = 0.$$
is quasi-linear if the coefficients a_α, a_o depend on lower-order derivatives of $u(\mathbf{x})$.

3. **Fully nonlinear** – where $\mathcal{F}$ is nonlinear with respect to the highest order derivatives of u, for example
$$\mathcal{F} = \left(\frac{\partial^2 u}{\partial x^2}\right)^2 + \left(\frac{\partial^2 u}{\partial x^2}\right)^2 = 0,$$
or the Monge-Ampere (MA) equation (in 2D)
$$\mathcal{F} = \frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} - \left(\frac{\partial^2 u}{\partial x \partial y}\right)^2 - f(x,y).$$
The MA equation is a fully nonlinear PDE with applications in physics and engineering ranging from meteorology, elasticity, geometric optics, image processing and others.

3 Finite Difference Methods (FDMs)

How might one solve a PDE numerically? We can only calculate a finite number of **discrete** values, so we begin by defining a **grid of points** on which we will seek the solution. For now, consider a one-dimensional grid, and seek to approximate the function $u(x)$ on $(a,\, b)$. We shall consider a uniform grid almost always, so we define a step-length,

$$h = (b-a)/(N-1),$$

for some large N, as shown in Fig. (3.1). We consider the points

$$x_n \equiv a + (n-1)h \;\text{ for }\; n = 1 \ldots N.$$

We shall seek an approximation U_n to the exact solution on the grid points, $u_n \equiv u(x_n)$.

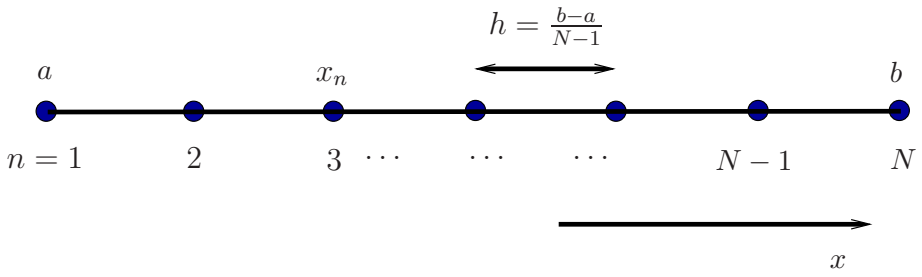

Figure 3.1: Discretised Grid.

Using suitable finite difference **(F-D)** approximations, we can represent any system of ODEs or PDEs in continuous variables as a set of algebraic equations for a finite list of unknown (*discrete*) values. We can then seek to solve this algebraic system on a computer. If $u(x)$ is a solution to a (partial) differential equation we need to represent the derivatives, $\partial u/\partial x \equiv u_x$, $\partial^2 u/\partial x^2 \equiv u_{xx}$, $\partial^2 u/\partial x \partial y \equiv u_{xy}$, ..., etc. with finite difference approximations. The F-D method replaces continuous derivatives in the governing equations by finite-difference approximations. The underlying task then is to approximate the derivative $\partial u/\partial x \equiv u_x = u'(x)$ of the continuous function $u(x)$ evaluated at a specified grid point x_n by a linear combination of **discrete** function values U_n. For the case of a first derivative this may lead to

$$\frac{\partial u}{\partial x}\Big|_n \approx \frac{u(x_n+h)-u(x_n)}{h} = \frac{U_{n+1}-U_n}{h}, \tag{3.1}$$

or we may equally well approximate the derivative as follows:

$$\frac{\partial u}{\partial x}\Big|_n \approx \frac{u(x_n)-u(x_n-h)}{h} = \frac{U_n-U_{n-1}}{h}, \tag{3.2}$$

a third choice, namely centred about x_n is given by

$$\frac{\partial u}{\partial x}\Big|_n \approx \frac{u(x_n+h)-u(x_n-h)}{2h} = \frac{U_{n+1}-U_{n-1}}{2h}. \tag{3.3}$$

Whats the difference between these 3 expressions? Can one arbitrarily pick one at random and expect the results to be correct? We will see later on in the course, with especially PDEs, that one or the other may or may not be appropriate. Can you think why?

In general we have to specify the number and location of discrete function values U_n which are used in the approximation, as well as the location at which the derivative is to be evaluated. For example, taking into account three function values $U_{n-1},\ U_n,\ U_{n+1}$ and evaluating the first derivative at the central point x_n (see Fig. 3.2), we obtain the general setup*

$$\frac{du}{dx}|_n \approx aU_{n-1} + bU_n + cU_{n+1} \tag{3.4}$$

with some yet unknown coefficients $a,\ b,\ c$. Similar setups for higher derivatives are imaginable. Once the unknown coefficients or **weights** are determined, the derivatives in the governing equations can then be replaced by linear combinations of the function values.

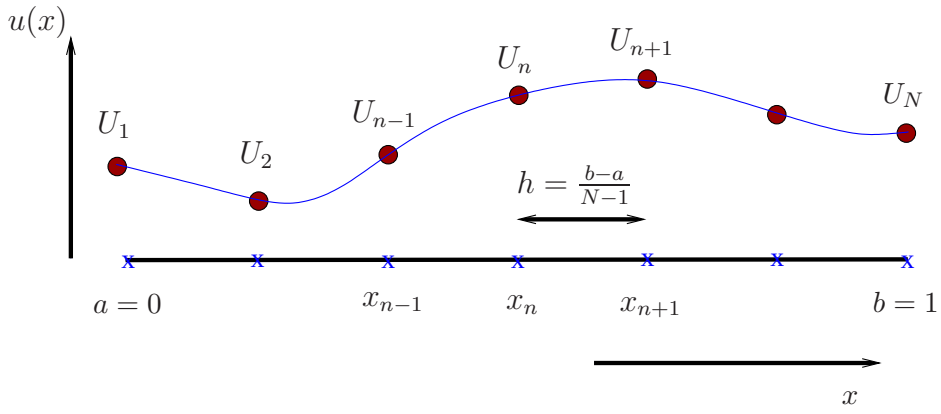

Figure 3.2: Sketch of a equi-spaced grid and a discrete representation of a continuous function $u(x)$ on this grid,

Even if our PDE is **nonlinear**, we almost always arrange things so that the system of equations we have to solve is **linear.** However there are usually a very large number of such equations, and we may have to give thought to the best way of solving them efficiently.

Depending on the method we use, they may be **explicit** or **implicit**. An explicit equation is of the form “unknown = easily calculable stuff", but an implicit system still requires further work, perhaps an iterative solution.

3.1 Finite-difference formulae via Taylor series expansions

To determine the weight coefficients $a,\ b$ and c in expression (3.4) for the first derivative, we use a Taylor series expansion of the function $u(x)$ about the point x_n where the derivative is sought. Using Taylor series we can write

$$u_{n+1} \equiv u(x_n + h) = u(x_n) + hu'(x_n) + \tfrac{1}{2}h^2u''(x_n) + \tfrac{1}{6}h^3u'''(x_n) + \tfrac{1}{24}h^4u^{iv}(x_n) + \dots \tag{3.5}$$

$$u_{n-1} \equiv u(x_n - h) = u(x_n) - hu'(x_n) + \tfrac{1}{2}h^2u''(x_n) - \tfrac{1}{6}h^3u'''(x_n) + \tfrac{1}{24}h^4u^{iv}(x_n) + \dots \tag{3.6}$$

It follows by addition and subtraction that

$$\textbf{subtraction :}\quad u_{n+1} - u_{n-1} = 2hu'(x_n) + \tfrac{1}{3}h^3u'''(x_n) + O(h^5)^\dagger, \tag{3.7}$$

*note in what follows we have changed our notation from using the ∂ symbol to d here.

†referred to as the *order*(h^5) truncated term

and

$$\textbf{addition :} \quad u_{n+1} + u_{n-1} = 2u(x_n) + h^2 u''(x_n) + \tfrac{1}{12} h^4 u^{iv}(x_n) + O(h^6). \tag{3.8}$$

We can therefore approximate the *first*-derivatives as follows

$$\frac{du}{dx}|_n = \frac{u_{n+1} - u_{n-1}}{2h} + O(h^2) \equiv \frac{(\Delta + \nabla)u_n}{2h} + O(h^2), \tag{3.9}$$

with the weights $a = -1/h, b = 0, c = 1/h$. The *second*-derivative FD form is given by

$$\frac{d^2u}{dx^2}|_n = \frac{u_{n-1} - 2u_n + u_{n+1}}{h^2} + O(h^2) \equiv \frac{\delta^2 u_n}{h^2} + O(h^2), \tag{3.10}$$

with the weights $a = 1/h^2, b = -2/h^2, c = 1/h^2$ – note usage above of *difference operator* notation too (see §3.2 below). Equations (3.10) and (3.9) are known as finite difference approximations to u_{xx} and u_x. They are **second order** as the error is $O(h^2)$. They are called **centred** which basically means that the approximation is symmetrical about n.

3.2 Difference operators

The difference operators Δ and δ^2 operate on n, just as d/dx operates on x. They may be used symbolically for non-integer n, if we want. So we define the following operators:

Forward difference : $\Delta x_n = x_{n+1} - x_n$

Backward difference : $\nabla x_n = x_n - x_{n-1}$

Shift operator : $Ex_n = x_{n+1}$

Averaging operator : $\mu x_n = \frac{1}{2}(x_{n+1/2} + x_{n-1/2})$

Central difference : 1^{st}-derivative : $\delta u_n = u_{n+1/2} - u_{n-1/2}$

Central difference : 2^{nd}-derivative : $\delta^2 u_n = u_{n-1} - 2u_n + u_{n+1}$

Differential operator : $\mathcal{D}y = dy/dx$

As an example (3.9) may also be written

$$\frac{du}{dx}|_n = \frac{u_{n+1} - u_{n-1}}{2h} = \frac{1}{h}\mu\delta u_n \equiv \triangle u_n. \tag{3.11}$$

Further note the 1^{st}-order accurate forward difference may be written

$$\frac{du}{dx}|_n = \frac{u_{n+1} - u_n}{h} = \frac{1}{h}(E - 1)u_n. \tag{3.12}$$

To derive finite difference using the above operators, $\mathcal{D}$ needs to be defined in terms of the other operators. We proceed with the Taylor series

$$f(x + h) = f(x) + h\frac{df}{dx} + \frac{h^2}{2!}\frac{d^2}{dx^2} + \frac{h^3}{3!}\frac{d^3}{dx^3} + \frac{h^4}{4!}\frac{d^4}{dx^4} + \dots ,$$

written in operator form

$$f(x+h) = Ef(x) = \left[1 + h\mathcal{D} + \frac{(h\mathcal{D})^2}{2!} + \frac{(h\mathcal{D})^3}{3!} + \ldots\right] f(x) = e^{h\mathcal{D}}\, f(x).$$

We see that

$$E = e^{h\mathcal{D}},$$

on basis that equality of the operators, whether using E or $e^{h\mathcal{D}}$ give identical results when operating on a polynomial. Hence it is easy to establish the following relationships

$$\begin{aligned} h\mathcal{D} &= \log E = \log(1+\Delta) = -\log(1-\nabla) \\ &= 2\sinh^{-1}\delta/2 = 2\log[(1+\tfrac{1}{4}\delta^2)^{1/2} + \tfrac{1}{2}\delta]. \end{aligned} \tag{3.13}$$

Hence the FD expressions follow by expansion and further manipulation of these expressions. For example

$$\log(1+\Delta)) \approx \Delta - \frac{\Delta^2}{2} + \frac{\Delta^3}{3} + \ldots,$$

truncation of this to first order then gives (3.12)[‡].

3.3 Truncation error

Above, the coefficients were obtained by requiring a match between the Taylor-expanded right-hand side and the derivative term on the left-hand side. By our design, only three constants $a,\ b,\ c$ were available to make this match. As a consequence, an error term appears when higher-order derivative terms on the right-hand side do not cancel or are neglected. The lowest of the non-cancelling terms is referred to as the **truncation error** e. In our case above (in 3.7) we have

$$e = (c-a)\frac{u'''|_n}{3!}h^3 = \frac{u'''|_n}{3}h^2 \sim O(h^2). \tag{3.14}$$

For h sufficiently small, the error will be dominated by u''' or the so-called *order-*(h^2) term, with all other even higher-order derivative contributions being negligible *relative* to this term. As the grid is refined, the truncation error decreases quadratically with the mesh width h. Representations of the first derivative to higher than second order require more $u_n\ (\equiv U_n)$ function values such that more coefficients can be used to eliminate even higher-order derivatives on the right-hand side, thus enabling the truncation error to be reduced further.

[‡]see W. F. Ames, 1977 for further details

3.4 Higher order and sided-differences

For improved accuracy, to be more precise, to reduce the truncation errors, higher-order difference formulae can be deduced using the basic formalism of Taylor series. Doing so, leads to using more discrete functional values on either side of the n^{th}-*centered* point about which derivatives are to be discretised. Note that these higher-order FDs may need modification near boundaries. When derivatives have to be evaluated on or near the boundary of the computational domain, but function values are only available on one side or a *centered-discretised* stencil is not possible, approximations to derivatives of functions there are known as **sided formulae**. For example, with reference to Fig. 3.2, at $n = 1$ we have no information or knowledge of the u-state at points $n < 1$ or $x < a$, or for that matter $n > N$ or $x > b$.

3.4.1 4^{th}-Order or $O(h^4)$ accurate finite-differences

For this order of accuracy, a 5-point scheme is required with the central differenced weights given as follows :

$$\frac{du}{dx}|_n = \frac{u_{n-2} - 8u_{n-1} + 8u_{n+1} - u_{n+2}}{12h},$$
$$\frac{d^2u}{dx^2}|_n = \frac{-u_{n-2} + 16u_{n-1} - 30u_n + 16u_{n+1} - u_{n+2}}{12h^2},$$

4^{th}-Order or $O(h^4)$ accurate sided differences, near boundaries

Left-sided non-centered boundary weights :

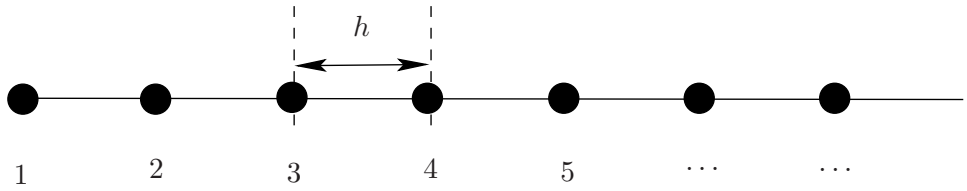

Figure 3.3: Sided differencing stencil, left-sided boundary points.

$$\frac{du}{dx}|_2 = \frac{-3u_1 - 10u_2 + 18u_3 - 6u_4 + u_5}{12h},$$

$$\frac{d^2u}{dx^2}|_2 = \frac{11u_1 - 20u_2 + 6u_3 + 4u_4 - u_5}{12h^2},$$

$$\frac{du}{dx}|_1 = \frac{-25u_1 + 48u_2 - 36u_3 + 16u_4 - 3u_5}{12h},$$

$$\frac{d^2u}{dx^2}|_1 = \frac{35u_1 - 104u_2 + 114u_3 - 56u_4 + 11u_5}{12h^2},$$

Right-sided non-centered boundary weights :

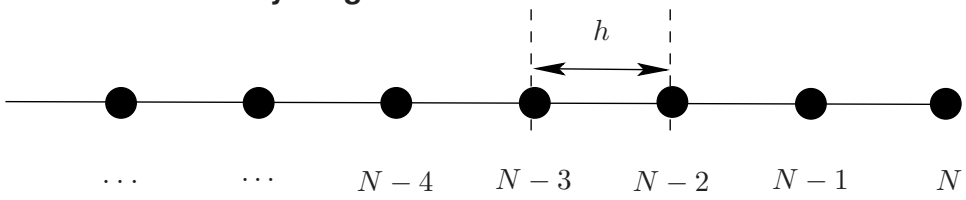

Figure 3.4: Sided differencing stencil, right-sided boundary points.

$$\frac{du}{dx}|_{N-1} = \frac{3u_N + 10u_{N-1} - 18u_{N-2} + 6u_{N-3} - u_{N-4}}{12h},$$

$$\frac{d^2u}{dx^2}|_{N-1} = \frac{11u_N - 20u_{N-1} + 6u_{N-2} + 4u_{N-3} - u_{N-4}}{12h^2},$$

$$\frac{du}{dx}|_{N} = \frac{25u_N - 48u_{N-1} + 36u_{N-2} - 16u_{N-3} + 3u_{N-4}}{12h},$$

$$\frac{d^2u}{dx^2}|_{N} = \frac{35u_N - 104u_{N-1} + 114u_{N-2} - 56u_{N-3} + 11u_{N-4}}{12h^2},$$

Further details on this and other ways of computing weights may be found in the reading list (LeVeque, 2007). To summarise the weights can be determined using:

- Taylor series with method of undetermined coefficients;
- Difference operators described in §3.2 above;
- Lagrange or Polynomials interpolation.

3.5 Finite-difference formulae via Lagrange interpolation

An alternative and more flexible method to develop finite-difference approximations to continuous derivatives is based on Lagrange interpolation. The idea is to reconstruct a local, approximate, but continuous representation of the function $u(x)$ by interpolating the discrete function values u_n using a polynomial of appropriate degree. This polynomial interpolant is then differentiated exactly and evaluated at the point of interest. The concept thus relies on a three-step procedure (see LeVeque, 2007):

- interpolation of the data points by a polynomial of appropriate degree,
- exact differentiation of the polynomial interpolant, and
- evaluation of the differentiated interpolant at the desired grid point.

A very robust and efficient procedure to evaluate weights to any order of accuracy is given by Fornberg (1996) (see Reading list).

3.6 Our first finite difference method

For example, we could seek to solve the ordinary differential equation (ODE) :

$$u' \equiv \frac{du}{dx} = -u \quad \text{in} \quad x > 0 \quad \text{with} \quad u(0) = 1. \tag{3.15}$$

Choosing a step-length and grid as shown in Fig. 3.2, we could represent the ODE fairly accurately as

$$\frac{U_{n+1} - U_{n-1}}{2h} = -U_n, \qquad \text{with} \quad U_1 = 1. \tag{3.16}$$

This is a linear recurrence relation. If we know U_1 and U_2, by varying n we can find U_n for all n. Obviously the exact solution to this problem is $u = e^{-x}$. So let's take $U_2 = e^{-h}$.

$$U_{n+1} = -2hU_n + U_{n-1} \qquad \text{for } n \geq 1, \text{ with} \quad U_1 = 1, \quad U_2 = e^{-h}. \tag{3.17}$$

More formally this is known as an **explicit** method, whereby we simply march forward in the variable x_{n+1} and values of the solution there U_{n+1} simply involves previously evaluated values of the function U at $n,\ n-1$, or all $n < n+1$.

Try it and see what happens. Does $|U_n - u_n|$ remain small as n increases? Does the approximation improve as h decreases?

It is very important to realise that even if our equations are a good approximation to the ODEs, the solutions obtained may be completely different, for various reasons.

How well does U approximate the true exact solution $u(x) = e^{-x}$? In most problems of practical interest the true exact solution is usually not known. How do we ascertain that our discretised numerical solution is a true solution of the exact governing equation, or that as h the step size is decreased the errors reduce and the solution does indeed converge to the true solution? A very good introduction on aspects of **Truncation errors**, **Global error**, **Numerical Stability**, **Consistency** and **Convergence** are given in LeVeque (2007) (see sections 2.4 – 2.10).

PDEs are trickier than ODEs, and are quite sensitive to their boundary conditions. Before we can hope to model them well, we need to have some idea of the possible underlying behaviour. So next time, we'll consider the basic types of PDE. We will then consider solving each type in turn.

Typical results of our first Finite Difference Method, Eqn. (3.16)

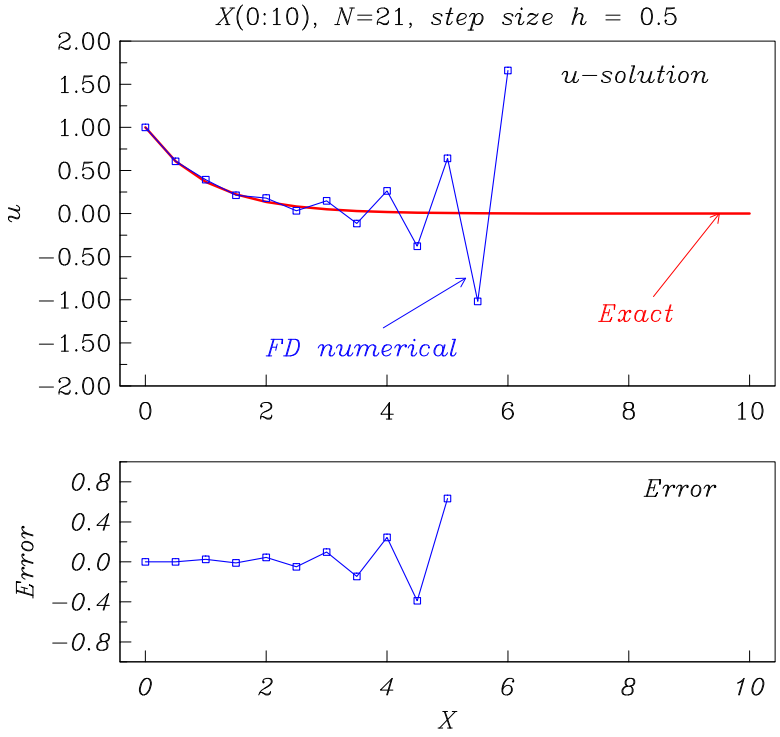

Figure 3.5: Solution of $u' = -u$ with $u(0) = 1$, N=21 grid points.

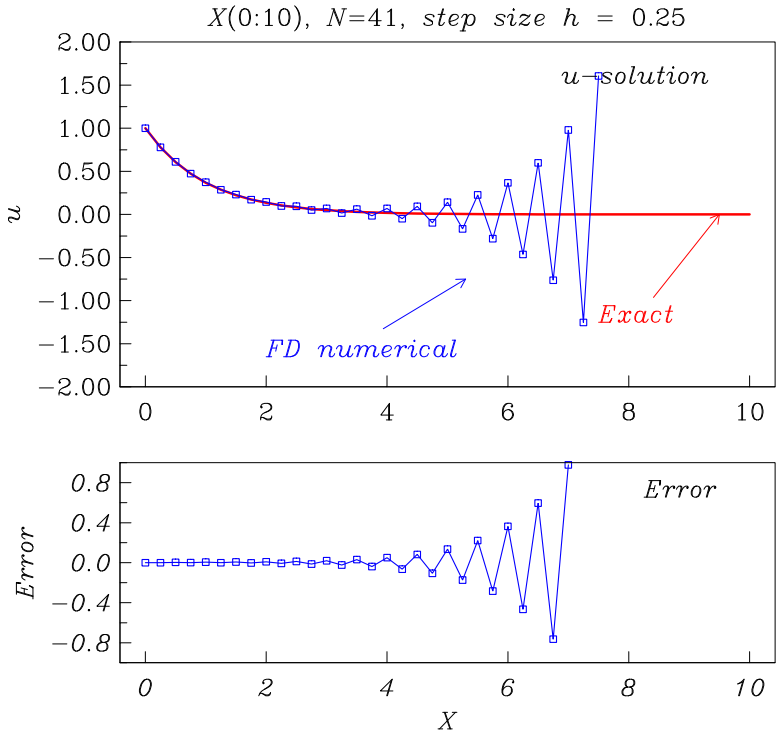

Figure 3.6: Solution of $u' = -u$ with $u(0) = 1$, N=41 grid points.

Typical results of improved Finite Difference Method

Discretisation:

$$\frac{U_{n+1} - U_{n-1}}{2h} = -\frac{U_{n+1} + U_{n-1}}{2}, \qquad \text{with} \quad U_1 = 1. \tag{3.18}$$

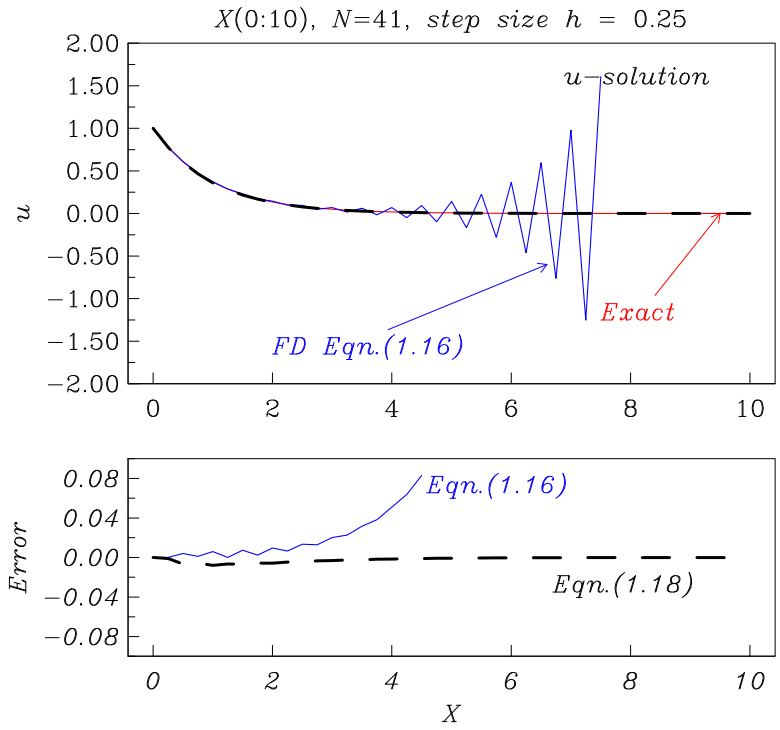

Figure 3.7: Solution of $u' = -u$ with $u(0) = 1$, N=41 grid points, using expression (3.18) given by the dashed line (– – –).

Comparing figure 11.2 with figure 3.6, why do you think the expression (3.18) gives us a much better result then using Eqn. (3.16)?

3.6.1 Numerical stability of ODEs

To conceptualise ideas, consider a general ODE of the form

$$\frac{du}{dt} = -\lambda u, \quad \text{given some initial state} \quad u(0) = \alpha, \tag{3.19}$$

and $\lambda > 0$. The **Euler**-method is the simplest example of an **explicit** method. We may discretise the above as follows

$$U_{n+1} = U_n(1 - h\,\lambda), \tag{3.20}$$

hence, given some initial state at $u(0)$, it is trivial to see we can write the solution as

$$U_n = u(0)(1 - h\,\lambda)^n. \tag{3.21}$$

Now we know that the exact solution is given by

$$u(t) = u(0)\, e^{-\lambda t},$$

which in the limit as $t \to \infty$ must go to zero. It follows for Eqn. 3.20 to give this behaviour we require $(1 - h\ \lambda)^n \to 0$ as $n \to \infty$. This will be so, provided

$$|(1 - h\ \lambda)| \leq 1. \tag{3.22}$$

This condition can also be related to the requirement that the **general error** *(see § 3.6.2) should be damped (or remain constrained) as the* t*-variable increases*. Hence it follows we require

$$h \leq \frac{2}{\lambda}$$

for stability of the numerical method – this is known as **conditional stability**. The numerical solution blows up if $h > 2/\lambda$.

Next consider discretising Eqn. 3.19 as follows

$$U_{n+1} - U_n = -h\ \lambda\ U_{n+1}, \tag{3.23}$$

i.e. known as the fully **implicit** discretisation, which upon re-arrangement gives

$$U_n = \frac{u(0)}{(1 + h\ \lambda)^n}. \tag{3.24}$$

For numerical stability, we thus require

$$\frac{1}{(1 + h\ \lambda)} \leq 1, \tag{3.25}$$

In this case we see the method is **unconditionally** stable for any h ! Though of course the larger h is then truncation errors may well be significant. Just because a method is stable does not mean it is accurate – accuracy is attained by making the step h asymptotically small.

Stability limits for the forward Euler and Implicit method are shown in Fig. 3.8. More gener-

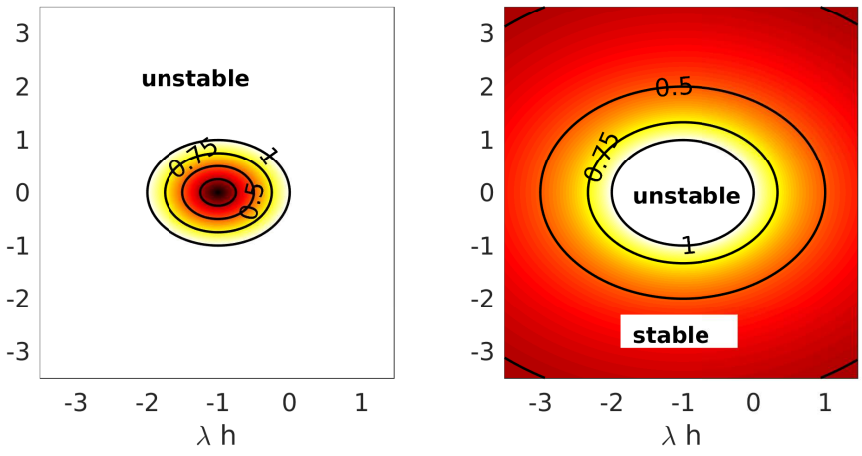

Figure 3.8: Stability region (contours ≤ 1) for the forward Euler Eqn. 3.22, and fully implicit method Eqn. 3.25.

ally we need to allow λ to be a complex value[§]. A method is then called **absolutely stable**

[§]for a comprehensive treatment see Grifiths, D. F. & Higham, D. J. (2010), *Numerical Methods for Ordinary Differential Equations – Initial Value Problems, Springer SUMS.*

if $Re(\lambda) < 0$, such that the numerical solution for, say, a scalar equation (such as Eqn. 3.19) or more general form

$$\frac{du}{dt} = \lambda f(u,t),$$

where the general errors decay to zero. We call the **region of absolute stability** the set of complex numbers

$$z = \lambda h.$$

For Euler's method, with $f(u,t) = u(t)$ the region of absolute stability is

$$|1 + z| \leq \ 1.$$

If $h\lambda$ is real, the interval of absolute stability is given by

$$-1 < 1 + h\lambda < 1.$$

which means that z must be in a unit disk in the complex plane centered at $(-1, 0)$. To see this, let $z = x + iy$, then the above leads to

$$|1 + z|^2 = \ 1 \ \text{ or } \ (x+1)^2 + y^2 = 1.$$

Undertake similar analysis for expressions 3.17 and 3.18. What are the stability restrictions with these expressions if any?

3.6.2 General Error discrete form

Now the finite difference given by Eqn. 3.20 was constructed by setting the truncation term e to zero. Retaining this in the FD-form gives

$$\hat{u}_{n+1} = \hat{u}_n(1 - h\ \lambda) + h^2 e_n, \ \text{ where } \ e_n \equiv u_n''/2. \tag{3.26}$$

Subtracting this from Eqn. 3.20, namely constructing the general **error propagation** equation by defining $\hat{z}_n = \hat{u}_n - U_n$ gives :

$$|\hat{z}_{n+1}| = \beta|\hat{z}_n| + h^2|e_n|, \ \text{ where } \ \beta = 1 - h\ \lambda. \tag{3.27}$$

There are two aspects we expect, (1) as $h \to 0$ the truncation error contribution approaches zero; and (2) the general error $\hat{z}_n$ remains constrained for all n and ideally remains "infinitesimally" small **relative to the true solution** U_n as $n \to \infty$. That is for stability of the numerical discretisation, the errors $\hat{z}$ should not be magnified by the algorithmic process.

Considering Eqn. 3.27, it follows $\hat{z}_0 = 0$, given we enforce a known (exact) initial condition to start the procedure, thus at the $n = 1$'th-step

$$|\hat{z}_1| = h^2|e_n| = Ah^2, \ \text{ where } \ A \ \text{ is some positive constant.}$$

Next considering a sequence of steps we get

$$|\hat{z}_{n+1}| = Ah^2(1 + \beta + \beta^2 + \beta^3 + \ldots + \beta^n),$$

i.e. a geometric sum, hence for numerical stability of the method, or the errors to be constrained we require

$$|\hat{z}_{n+1}| = Ah^2\,|\beta^n| \ \to 0 \ \text{ as } \ n \to \infty.$$

Numerical stability of the scheme is achieved if the norm of the error remains bounded as n increases, it follows

$$|\beta| = |1 - h\lambda| < \ 1.$$

4 Classification of 2nd-order Quasi-linear PDEs in Two Variables

Why didn't our first finite difference program work very well? This was especially the case with using Eqn. (3.16), while using Eqn. (3.18) gave a much more accurate result. There must be something wrong with (a) the mathematical problem itself, (b) our solution algorithm, or (c) our computational implementation. Usually when this happens one suspects that there must be a bug in our program – failing that, our solution method may be at fault. But sometimes, the problem we set out to solve is not well-posed. In this lecture we will introduce some theory and illustrate some ways in which a PDE problem might be insoluble.

Most physical systems are governed by second order PDEs. The majority of problems fall into three main categories: *equilibrium, eigenvalue and propagation problems*. In this course we shall not deal with eigenvalue problems*.

Equilibrium problems are generally steady state ones in which the equilibrium, state ϕ in a domain D is to be determined by solving the differential equation

$$L[\phi] = f \tag{4.1}$$

within D, subject to certain boundary conditions $B_i[\phi] = g_i$ on the boundary of D. Often the integration domain D is closed and bounded. Fig. 4.1 illustrates the general equilibrium problem - such problems are generally known as *boundary value problems (BVPs)*. Examples are steady viscous flow, steady temperature distributions and equilibrium stresses in structures among others. Generally the governing equations for equilibrium problems are *elliptic*.

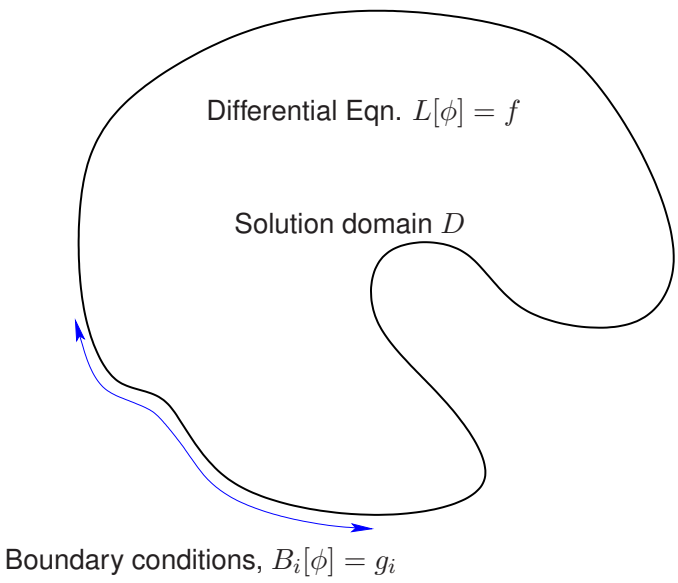

Figure 4.1: Equilibrium or bounded boundary value problem.

Propagation problems are *initial value problems (IVPs)* that have an unsteady state or a transient nature. In such problems one seeks the subsequent behaviour of the system

*The finite difference techniques we discuss for solving equilibrium problems, are generally similar to those used to discretise eigenvalue problems.

given an initial state. This is done by solving some differential equation

$$L[\phi] = f \tag{4.2}$$

within the domain D, when the initial state is prescribed as

$$I_i[\phi] = h_i \tag{4.3}$$

and subject to prescribed conditions

$$B_i[\phi] = g_i \tag{4.4}$$

on some open boundaries. Fig. 4.2 illustrates a typical propagation or *initial value problem*. Examples include propagation of pressure waves in a fluid, propagation of stresses and displacements in elastic systems, heat propagation and self excited vibrations amongst others. There are though two distinct classes of problems here, namely **parabolic** and **hyperbolic** types.

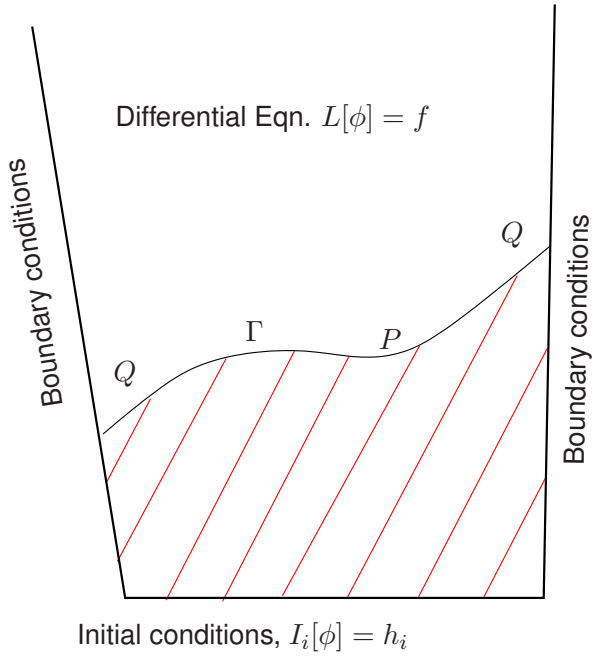

Figure 4.2: Propagation or initial value problem.

A distinction between equilibrium and propagation problems is that in the former, the entire solution is dependent upon satisfaction of all the boundary conditions and all internal requirements. In propagation problems the solution is marched out from the initial state guided during the marching process by the side boundary conditions and governing PDE.

In this course we discuss **Finite Difference Methods (FDMs)** for solving such equations. We want our algorithms to be able to reproduce the physics and so to begin with, we must understand the physical background. We consider the equation for $u(x,\, y)$

$$au_{xx} + bu_{xy} + cu_{yy} = f \ . \tag{4.5}$$

This equation is called **quasi-linear** provided the functions a, b, c and f do not depend on u_{xx}, u_{xy} or u_{yy}. They may, however depend on x, y, u, u_x and u_y, so that (4.5) is not necessarily **linear** – for example it is perfectly allowable in the discussion that follows that

$$f = du_x + eu_y + hu + g, \tag{4.6}$$

provided d, e, h, g functions retain the *quasi-linear* requirement.

Classification of equations is best accomplished by developing the concept of **characteristics**. Suppose we know u, u_x and u_y along some curve Γ in (x, y)-space. From a point P on Γ we move a small vector displacement $(dx,\ dy)$ to a new point Q not on Γ, as shown in Fig. 4.3. Under what circumstances can we determine uniquely the values of u, u_x and u_y at Q? We denote the change in these variables by du, $d(u_x)$ and $d(u_y)$. Then by the chain rule for partial derivatives, the total derivative

$$du = u_x dx + u_y dy,$$

which is known because u_x and u_y are known along Γ. Similarly we may define

$$\left.\begin{aligned} d(u_x) &= u_{xx}dx + u_{xy}dy \\ d(u_y) &= u_{xy}dx + u_{yy}dy \end{aligned}\right\}. \tag{4.7}$$

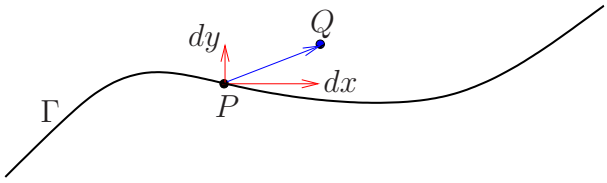

Figure 4.3: Problem description.

We combine (4.5) and (4.7) in matrix form:

$$\begin{pmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{pmatrix} \begin{pmatrix} u_{xx} \\ u_{xy} \\ u_{yy} \end{pmatrix} = \begin{pmatrix} f \\ d(u_x) \\ d(u_y) \end{pmatrix}, \tag{4.8}$$

a, b and c are known locally because u, u_x and u_y are known, and so the 3×3 matrix is known. Equation (4.8) will have a unique solution for u_{xx}, u_{xy} and u_{yy} *if* the determinant is **not** zero – then the derivatives have the same values above and below the curve Γ. If the determinant of the matrix vanishes, that is

$$\begin{vmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{vmatrix} = a(dy)^2 - b\,dxdy + c(dx)^2 = 0, \tag{4.9}$$

the equation (4.8) will have either no solution or infinitely many solutions. The condition for solutions to exist, which we will use later in the course, is that

$$a\frac{d(u_x)}{dx} + c\frac{d(u_y)}{dy} = f\ . \tag{4.10}$$

This is surprising. If we can choose a direction (dx, dy) which satisfies (4.9) we have the possibility that the second derivatives u_{xx} etc. may not be uniquely defined. In other words, the solution may have discontinuities across the line PQ, with u_{xx} taking different values on each side.

It is very important to know whether our solution can have this property. Equation (4.9) is called the **Characteristic equation** of (4.5). It is a quadratic in dy/dx with solution

$$\frac{dy}{dx} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a} \,. \tag{4.11}$$

Equation (4.5) is classified as hyperbolic, parabolic or elliptic according to whether these roots are real. Note the sign of b in the formula.

$$\text{If} \left\{ \begin{array}{lll} b^2 - 4ac > 0, & 2 \text{ real roots (4.5) is} & \textbf{hyperbolic} \\ b^2 - 4ac = 0, & 1 \text{ real roots (4.5) is} & \textbf{parabolic} \\ b^2 - 4ac < 0, & 0 \text{ real roots (4.5) is} & \textbf{elliptic} \end{array} \right\} . \tag{4.12}$$

For hyperbolic equations, (4.11) is an ODE for $y(x)$ which can be integrated to define two sets of curves (one for the $(+)ve$ sign, one for the $(-)ve$ sign), called the **characteristics** of (4.5).

4.1 Example: Significance of characteristics

Consider the one-dimensional wave equation for $u(x,\, t)$ with constant c

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} \qquad \text{for} \quad t > 0, \qquad \text{with} \quad u(x,0) = F(x) \;\text{ and }\; u_t(x,0) = G(x) \,. \tag{4.13}$$

This equation has the general solution $u = f(x - ct) + g(x + ct)$ for any functions f and g and with the above boundary conditions we have d'Alembert's solution:

$$u(x,\, t) = \tfrac{1}{2}[F(x + ct) + F(x - ct)] + \frac{1}{2c}\int_{x-ct}^{x+ct} G(\xi)d\xi \,. \tag{4.14}$$

Using (4.11) the characteristics of (4.13) are two families of straight lines

$$\frac{dt}{dx} = \pm\frac{1}{c} \qquad \text{or} \quad x \pm ct = \text{constant.} \tag{4.15}$$

From the actual solution, we see that the solution at some point P or $(x_0,\, t_0)$ with $t_0 > 0$ depends only on some of the initial data, that for which

$$x_0 - ct_0 \le x \le x_0 + ct_0.$$

Only points from which characteristics going forwards in time can reach the point P can influence the solution at P. The set of such points is called the **domain of dependence** of P. In exactly the same way not all points with $t > t_0$ can be affected by the solution at P. The collection of such points is called the **domain of influence** of P.

This behaviour is easy to understand physically, if one interprets characteristics as **curves along which information travels at a finite speed**. If something happens at P it takes a certain time before news of it reaches another point. For (4.13), the characteristics are parallel lines, but for more general hyperbolic systems they will be curved and may meet. Since the higher derivatives are indeterminate along these curves they provide paths for the propagation of discontinuities.

In such cases discontinuities may form. If neighbouring characteristics touch, conflicting information arrives at the same point, leading to the creation of **shock waves** (such as sonic booms, or pressure fronts); see Figs. 4.4–4.7. Such discontinuities then propagate into the medium along the characteristics.

It is clear from this example that whether or not characteristics exist is vital for the understanding and therefore the numerical modelling of a problem. They are associated with *"time-like"* behaviour, and have a characteristic speed associated with them, defining the rate at which information travels. In contrast elliptic problems have no "time-like" variable; x and y behave like space coordinates.

4.1.1 General solution of the wave equation D'Alembert's solution

The one-dimensional wave equation is defined

$$\frac{\partial^2 u}{\partial x^2} - \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} = 0. \tag{4.16}$$

Changing variables to $r,\ s$ where,

$$r = x - ct, \quad s = x + ct \tag{4.17}$$

Eqn. 4.16 becomes

$$\frac{\partial^2 u}{\partial r \partial s} = 0.$$

The general solution to this is

$$u = f(r) + g(s),$$

i.e.

$$u = f(x - ct) + g(x + ct); \tag{4.18}$$

Moreover note we can differentiate these expressions to give

$$\frac{\partial u}{\partial t} = -cf(x - ct) + cg(x + ct) \tag{4.19}$$

The arbitrary functions $f,\ g$ can be described completely if u and $\partial u/\partial t$ are given at $t = 0$, *i.e.* if

$$u = \xi(x), \quad \frac{\partial u}{\partial t} = \eta(x) \ \text{ at } \ t = 0,$$

where $\xi(x),\ \eta(x)$ are given functions of x, then, by substituting into Eqns. 4.18 and 4.19 a pair of equations is obtained for f and g,

$$\begin{aligned} \xi(x) &= f(x) + g(x) \\ \eta(x) &= -cf'(x) + cg'(x). \end{aligned} \tag{4.20}$$

Integrating the second expression gives

$$\int_{x_o}^{x} \eta(z)\, dz = -cf(x) + cg(x). \tag{4.21}$$

Solving Eqn. 4.20 and 4.21 gives

$$\begin{aligned} f(x) &= \frac{1}{2}\left[\xi(x) - \frac{1}{c}\int_{x_o}^{x} \eta(z)\, dz\right] \\ g(x) &= \frac{1}{2}\left[\xi(x) + \frac{1}{c}\int_{x_o}^{x} \eta(z)\, dz\right]. \end{aligned} \tag{4.22}$$

Hence

$$\begin{aligned} f(x-ct) &= \frac{1}{2}\left[\xi(x-ct) - \frac{1}{c}\int_{x_o}^{x-ct} \eta(z)\, dz\right] \\ g(x+ct) &= \frac{1}{2}\left[\xi(x+ct) + \frac{1}{c}\int_{x_o}^{x+ct} \eta(z)\, dz\right]. \end{aligned} \tag{4.23}$$

Finally substituting into Eqn. 4.18 gives

$$u = \frac{1}{2}\left[\xi(x-ct) + \xi(x+ct) + \frac{1}{c}\int_{x-ct}^{x+ct} \eta(z)\, dz\right]. \tag{4.24}$$

4.2 Boundary conditions for well-posed problems

A problem involving a PDE is said to be 'well-posed' if three conditions hold :

1. A solution exists.
2. The solution is unique.
3. The solution is continuous in the boundary conditions, *i.e.* small changes in the boundary conditions do not lead to large changes in the local solution.

The first requirement states the obvious fact that we cannot possibly think of finding a numerical approximation of the solution if it does not exist. The second states that the numerical scheme should always give one solution – if we get a numerical solution that oscillates between two or more, then clearly the solution is not unique. The last condition is the most important : a solution of the PDE is physically meaningful if small changes in the data cause small changes to the solution. In other words, a mathematical model of a physical phenomenon in general will not be useful if small errors in the measured data would lead to drastically different solutions. If this last condition fails to hold, the problem is non-physical and a disaster for numerical modelling.

Whether or not a problem is well-posed depends critically on whether its boundary conditions are appropriate. Typical boundary conditions are:

(a) boundary value problems (BVP); the PDE holds in some closed region and the solution is constrained all over the boundary;

(b) initial value problems (IVP); One or more constraints are given on some curve (usually $t = 0$) only partially bounding the region in which the PDE holds.

4.3 Hyperbolic equations

From our discussion of characteristics, it is clear that hyperbolic systems should have initial value conditions. Information spreads out from the initial values at a finite rate.

An example of an **ill-posed** hyperbolic BVP for $u(x,\, y)$ with a non-unique solution is

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial y^2} \text{ in } \left\{ \begin{array}{l} 0 < x < 1, \\ 0 < y < 1, \end{array} \right\} \text{ with } \left\{ \begin{array}{l} u(x,0) = u(x,1) = 0 \\ u(0,y) = u(1,y) = 0. \end{array} \right\}. \tag{4.25}$$

This problem has the solution $u = A \sin n\pi x \sin n\pi y$ for any constant A and integer n.

4.4 Elliptic equations

These have no characteristics; no lines along which information travels, which suggests that IVPs are inappropriate. A typical elliptic equation is **Laplace's equation**

$$\nabla^2 u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \in D, \tag{4.26}$$

where D is some region of (x, y)-space. It can be shown that this equation together with one boundary condition (say $u = f$) on the boundary ∂D can give a well-posed problem, with a smooth solution. In contrast, the IVP

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \in y > 0 \qquad \text{with} \quad u(x, 0) = u_y(x, 0) = 0 \tag{4.27}$$

is ill-posed, even though the solution $u(x, y) = 0$ is unique! Suppose we perturb the initial conditions so that $u(x,\, 0) = \varepsilon \sin nx$, and $u_y(x,\, 0) = 0$, where $0 < \varepsilon \ll 1$ and n is arbitrary but large. No matter how big n is, $|\sin nx| \leq 1$, so we are not altering the boundary condition by more than ε. The (unique) solution to this new problem is

$$u = \varepsilon \sin nx \cosh ny \simeq \varepsilon \sin nx \, \tfrac{1}{2} e^{|ny|} \qquad \text{when} \quad |ny| \gg 1 \, . \tag{4.28}$$

So a small distance away from the initial line $y = 0$, the solution is now exponentially large, whereas for the unperturbed problem it was zero. If a tiny (albeit very wiggly) perturbation to the boundary conditions can lead to a vast difference in the solution the problem is physically meaningless and impossible to model numerically. This example, due to Hadamard, shows that IVPs for elliptic equations are discontinuous in the boundary conditions.

4.5 Parabolic equations

A typical example is the **diffusion equation** for $u(x, t)$ with constant diffusivity K:

$$\frac{\partial u}{\partial t} = K \frac{\partial^2 u}{\partial x^2} \qquad \text{with} \quad u(x,\, 0) = f(x) \, . \tag{4.29}$$

As we know $u(x, 0)$, we can calculate $u_t(x, 0) = K f''(x)$ from the equation, and so we would expect to be able to step away from $t = 0$. From (4.12), the characteristics are given by the repeated root $dt/dx = 0$ or $t =$ constant. This corresponds to an infinite speed of propagation of information. Is the solution stable? Once more we consider the boundary condition $f(x) = \varepsilon \sin nx$, so that the unique solution of (4.29) is

$$u(x,\, t) = \varepsilon \sin nx \, e^{-n^2 K t} \, . \tag{4.30}$$

When $\varepsilon = 0$ the solution is $u = 0$. When $\varepsilon > 0$ the solution decays away provided $Kt > 0$, but if $Kt < 0$ and n is large, the perturbed solution blows up once more.

Thus, parabolic equations require one initial condition and it is vital that we move “forwards in time." Physically, parabolic equations describe the smoothing out of an initial configuration towards an equilibrium. Many different initial conditions give rise to almost the same final state. This is why running the process backwards in time is an ill-posed problem. You **can not** un-stir a cup of tea!

4.6 Change of type : fluid flow example

Consider the partial differential equation, the so-called Prandtl-Glauert equation for subsonic or supersonic flow

$$(1 - M_\infty^2)\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{4.31}$$

as M_∞ varies from values of $M_\infty < 1$ to $M_\infty > 1$, a change in PDE-type and behaviour of physical phenomenon occurs. A slightly more complicated form is the transonic small disturbance equation, namely

$$\left[K - (\gamma + 1)\phi_x\right]\phi_{xx} + \phi_{yy} = 0, \tag{4.32}$$

where $K = c(1 - M_\infty^2)$ with (c, γ) are constant parameters. This equation is nonlinear due to the $\phi_x\phi_{xx}$ term and on changing sign *i.e.* $K > 0$ and $K < 0$ the equation switches from elliptic to hyperbolic type.

An alternative to the above is the steady compressible potential equation :

$$\left(1 - \frac{u^2}{a^2}\right)\frac{\partial^2\phi}{\partial x^2} - \frac{2uv}{a^2}\frac{\partial^2\phi}{\partial x\partial y} + \left(1 - \frac{v^2}{a^2}\right)\frac{\partial^2\phi}{\partial y^2} = 0 \tag{4.33}$$

where

$$u = \frac{\partial\phi}{\partial x}, \qquad v = \frac{\partial\phi}{\partial y} \tag{4.34}$$

and a is the speed of sound,

$$\frac{a^2}{\gamma - 1} + \frac{u^2 + v^2}{2} = const. \tag{4.35}$$

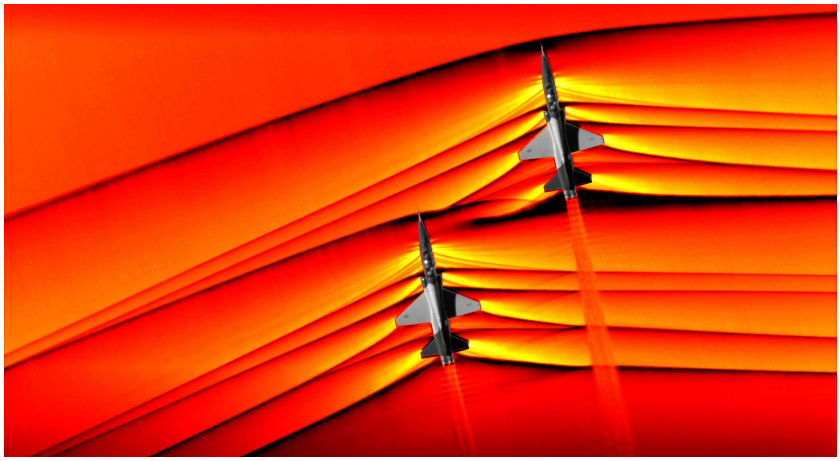

Figure 4.4: (A) Navier-Stokes Equations in action : Supersonic flow

Another example of mixed character of PDEs and complexity of solution to expect is the following steady form of the incompressible Navier-Stokes equations:

$$\left.\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} &= \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \end{aligned}\right\}. \tag{4.36}$$

Figure 4.5: (B) Navier-Stokes Equations in action : Transonic flow (1)

where ν is the kinematic viscosity, (u, v) are velocities and p is the pressure. These equations (4.38) can be shown to be elliptic. However, in a certain domain of the field of interest, it can be shown that a subset of the above operates, namely in the immediate neighbourhood of fluid flow over a solid surface, the following PDEs describe the flow behaviour:

$$\left.\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= \nu\frac{\partial^2 u}{\partial y^2} \\ \frac{\partial p}{\partial y} &= 0 \end{aligned}\right\}. \tag{4.37}$$

These equations adequately describe the so-called "viscous boundary-layer" and can be shown to be of *nonlinear* parabolic type, while far away from the surface the flow is adequately described by the inviscid Euler *elliptic* form:

$$\left.\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= 0 \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} &= 0 \end{aligned}\right\}. \tag{4.38}$$

This complexity of solutions is shown in figure 4.6.

Figure 4.7 shows the typical complexity of solutions possible in various domains in the flow over an aerofoil, varying from elliptic, hyperbolic and parabolic type behaviours, all in the same equations set, and remarkably nature is effortlessly able to slip from one type to another!

Clearly, an understanding of each type of equation is essential, prior to undertaking a "*numerical attack*". On the computer, we shall consider and use the **Finite Difference Method**

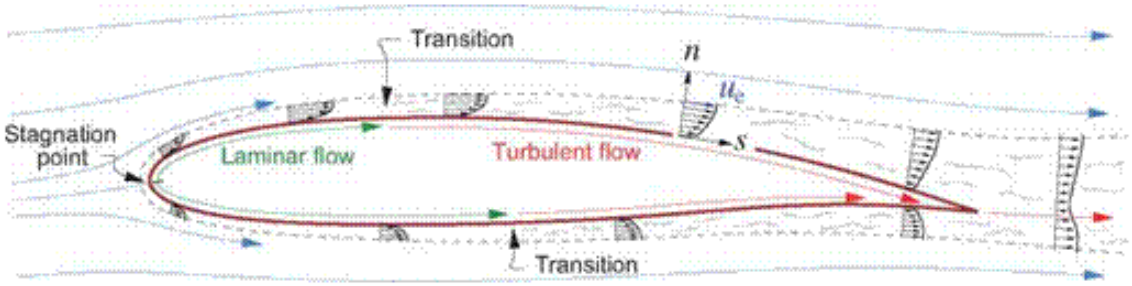

Figure 4.1: Boundary layer and wake development on a typical airfoil, shown by the u(n) velocity profiles. The layer thicknesses are shown exaggerated.

Figure 4.6: (D) Navier-Stokes Equations in action : Boundary Layers

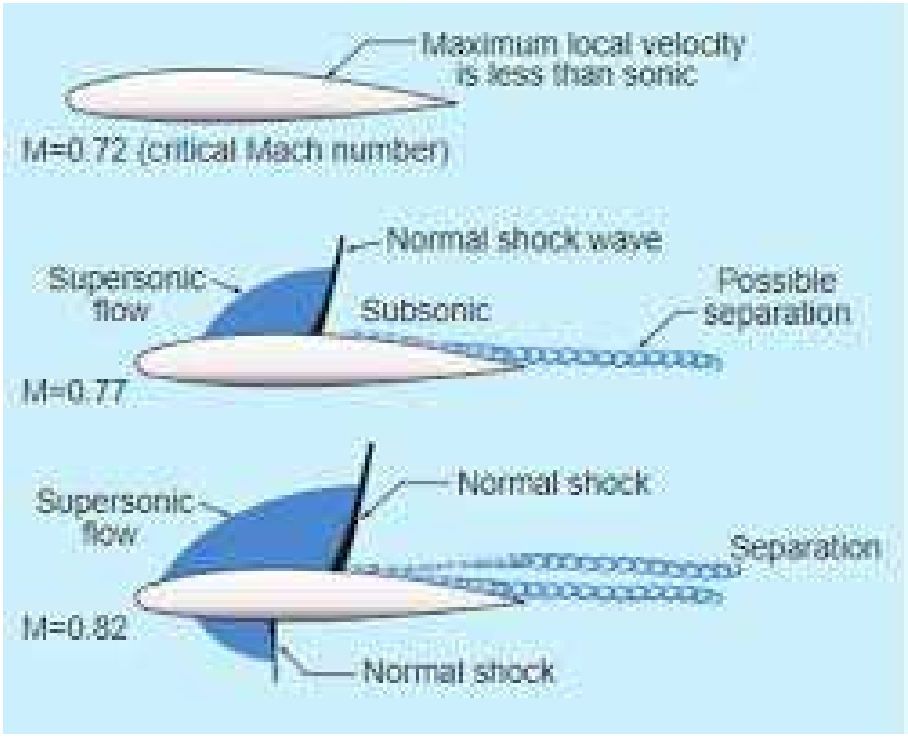

Figure 4.7: (C) Navier-Stokes Equations in action : Transonic flow (2)

(FDM) to solve the PDEs that we discuss. We will find, that the most appropriate numerical solve strategy involves an *apriori* insight into the mathematical and physical character of the problem, and the associated properties of the PDE – **together** with the correct application of boundary conditions whether on some surface or at some so-called far-field unbounded domain.

Summary

Equation type	Appropriate B.C.	Method of solution
Hyperbolic	Initial	Step in either direction from initial line
Parabolic	Initial	Step in one direction only from initial line
Elliptic	Boundary	Must solve everywhere simultaneously

Examples of various equation types

Hyperbolic	Parabolic	Elliptic
Solutions may be discontinuous (characteristic speed c)	Smooth solutions (diffusivity K)	Smooth solutions
Maxwell's equations	Heat equation	Electrostatics (Poisson)
Unsteady 1-D compressible	Unsteady incompressible N-S	Potential flow
Steady 2-D supersonic	Steady boundary layer	Steady Navier-Stokes

4.7 Classification of 1st-order quasi-linear PDEs in two variables

We next consider a coupled first-order quasi-linear systems of equations, namely

$$\left.\begin{array}{rcl} a_1 u_x + b_1 u_y + c_1 v_x + d_1 v_y & = & g_1 \\ a_2 u_x + b_2 u_y + c_2 v_x + d_2 v_y & = & g_2 \end{array}\right\}, \tag{4.39}$$

where the coefficients $a_1,\ a_2,\ b_1, \ldots,\ g_1,\ g_2$ may be functions of $x,\ y,\ u$ and v (satisfying the quasi-linear requirement). Using the chain rule, *i.e.*

$$\left.\begin{array}{rcl} du & = & u_x\, dx \ + \ u_y\, dy \\ dv & = & v_x\, dx \ + \ v_y\, dy \end{array}\right\},$$

the coupled system (4.39) may then again be written in matrix form :

$$\begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ dx & dy & 0 & 0 \\ 0 & 0 & dx & dy \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ du \\ dv \end{pmatrix}, \tag{4.40}$$

with u, v and the coefficient functions $a_1,\ a_2,\ b_1, \ldots,\ g_1,\ g_2$ known along Γ in Fig. 4.3. Again a unique solution for $u_x,\ u_y,\ v_x$ and v_y exists if the determinant of (4.40) is **not** zero – in which case the directional derivatives have the same value above and below Γ.

A zero value of the determinant implies that a multiplicity of solutions is possible, and thus the partial derivatives $u_x,\ u_y,\ v_x$ and v_y can not be determined uniquely, and thus discontinuities in these derivatives may occur on crossing Γ. Hence the characteristic equation can be shown to be

$$dy^2(a_2c_1 - a_1c_2) + dxdy(b_1c_2 - b_2c_1 - a_2d_1 + a_1d_2) + dx^2(b_2d_1 - b_1d_2) = 0, \tag{4.41}$$

which is a quadratic in dy/dx. It follows that the characteristics may be real, distinct, identical or complex according to whether the discriminant

$$(b_1c_2 - b_2c_1 - a_2d_1 + a_1d_2)^2 - 4(a_2c_1 - a_1c_2)(b_2d_1 - b_1d_2) \tag{4.42}$$

is positive, zero or negative. Hence again allowing classification of (4.39), into whether they are hyperbolic, parabolic or elliptic.

In this section we have essentially been looking at whether locally unique solutions exist for analytic quasi-linear partial differential equations associated with initial value problems. Further details of the formal proof may be found in the **Cauchy-Kovalevskaya** theorem, which concerns the existence, continuity and "*smoothness*" of solutions to a system of m differential equations in n dimensions when the coefficients are analytic functions. The theorem and its proof are valid for analytic functions of either real or complex variables.

5 The Explicit Method for the 1-D Diffusion Equation

Let us consider the problem for $u(x,\, t)$:

$$\left.\begin{array}{lll} \dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2} \quad (\text{or } \; u_t = u_{xx}\,), & \in & 0 < x < 1, \;\; t > 0 \\ \text{with} \quad u(0,\, t) = 0, & u(1,\, t) = 0\,, \quad u(x,\, 0) = f(x) & \end{array}\right\}, \qquad (5.1)$$

where the variable t denotes time and x is the spatial coordinate; note too the suffix notation used to denote partial derivatives *i.e.* $\partial u/\partial t \equiv u_t$ and $\partial^2 u/\partial x^2 \equiv u_{xx}$.

In general (more complicated PDEs), it is impossible to determine the solution of the boundary-value problem exactly in closed form. Thus we aim to describe a simple and general numerical technique to solve the problem using the finite difference method (FDM). The construction of a finite difference scheme consists of two steps :

1. The computational domain is approximated by a finite set of **grid** points, on a **mesh**.
2. The derivatives in the differential equation (and, possibly also in the boundary condition(s)) are approximated by difference weights on the mesh.

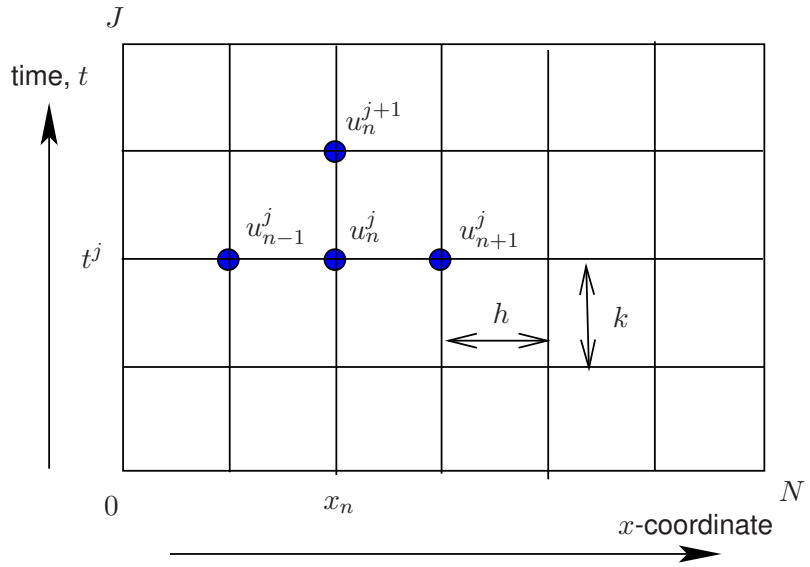

Figure 5.1: Discretised diffusion equation *mesh*.

We define a regular, rectangular grid $(x_n,\, t^j)$ for $0 \le n \le N$, $0 \le j \le J$ of lengths $(h,\, k)$, so that $Nh = 1$, $x_n = nh$, $t^j = jk$. We shall seek an approximation U_n^j to the exact solution evaluated on the grid points, $u_n^j \equiv u(nh,\, jk)$. The boundary conditions require $u_0^j = 0$, $u_N^j = 0$ and $u_n^0 = f_n \equiv f(x_n)$. Recalling the results from §3, we have

$$\frac{u_n^{j+1} - u_n^j}{k} = \frac{\partial u_n^j}{\partial t} + \tfrac{1}{2} k \frac{\partial^2 u_n^j}{\partial t^2} + O(k^2)\,,$$

and

$$\frac{u_{n+1}^j - 2u_n^j + u_{n-1}^j}{h^2} = \frac{\partial^2 u_n^j}{\partial x^2} + \tfrac{1}{12} h^2 \frac{\partial^4 u_n^j}{\partial x^4} + O(h^4)\,.$$

Using the above in the equation $u_t = u_{xx}$, we have

$$\frac{u_n^{j+1} - u_n^j}{k} = \frac{u_{n+1}^j - 2u_n^j + u_{n-1}^j}{h^2} + R_n^j \ , \tag{5.2}$$

where the **Truncation Error**, R_n^j is

$$R_n^j = \tfrac{1}{12}h^2 \frac{\partial^4 u_n^j}{\partial x^4} - \tfrac{1}{2}k\frac{\partial^2 u_n^j}{\partial t^2} + O(k^2, h^4) \ . \tag{5.3}$$

The simplest explicit method neglects terms of $O(k,\, h^2)$. If we write

$$r = k/h^2 \ ,$$

neglect R_n^j and replace u by U in (5.2), we obtain the scheme

$$U_n^{j+1} = rU_{n+1}^j + (1-2r)U_n^j + rU_{n-1}^j \ , \tag{5.4}$$

with the boundary conditions

$$U_0^j = 0 \ , \qquad U_n^j = 0 \ \text{ and } \ U_n^0 = f_n \equiv f(x_n) \ .$$

With a little thought, we see that these boundary conditions and repeated use of (5.4) enable us to calculate U_n^j everywhere. We must now consider how accurate the approximation is. We next perform a simple " *Maximum Principle Analysis*" to show that under some conditions U_n^j can be made as close as we choose to the real solution u_n^j, for all n and j.

5.1 Maximum Principle Analysis (MPA)

In general, suppose a PDE

$$Lu = f,$$

where L is some differential operator, is approximated on a suitable grid $(nh,\, jk)$, at which points u and f take the values u_n^j and f_n^j, by the FDM

$$MU_n^j = f_n^j,$$

where M is a difference operator. The **solution error**, z_n^j and the **truncation error**, R_n^j, are defined by

$$z_n^j \equiv u_n^j - U_n^j \ , \ \text{ and } \ R_n^j \equiv (Lu)_n^j - MU_n^j \ . \tag{5.5}$$

Returning to our specific equation, subtracting (5.4) from (5.2), and using (5.5), we obtain

$$z_n^{j+1} = rz_{n+1}^j + (1-2r)z_n^j + rz_{n-1}^j + kR_n^j \ \text{ and } \ z_0^j = z_N^j = z_n^0 = 0 \ . \tag{5.6}$$

Now

$$|z_n^{j+1}| \leq |r|\ |z_{n+1}^j| + |(1-2r)|\ |z_n^j| + |r|\ |z_{n-1}^j| + |kR_n^j| .$$

Since $R_n^j = O(k, h^2)$, over the finite interval $0 < t < T \equiv Jk$ we can find a positive constant A such that $|R_n^j| \leq A(|k| + h^2)$. We shall also define the norm

$$||z^j|| \equiv \max_{n=0\ldots N} |z_n^j| \ ,$$

which is the maximum error over all the points at a fixed time-level j. Then $|z_n^j| \leq ||z^j||$ for all n, and so we have

$$|z_n^{j+1}| \leq \Big(|r|+|1-2r|+|r|\Big)||z^j||+A(k^2+|k|h^2)$$

As this is true for all values of n, it is true for that value which maximises its LHS, and so

$$||z^{j+1}|| \leq \Big(|r|+|1-2r|+|r|\Big)||z^j||+A(k^2+|k|h^2)$$

We now **assume** that $0 < r \leq \frac{1}{2}$, so that the quantities inside modulus signs are positive. Then the maximum possible error at the time-level $(j+1)$ is related to the maximum at time j by

$$||z^{j+1}|| \leq ||z^j||+A(k^2+|k|h^2)$$

Now the initial error, $||z^0||$, is zero because we know the solution exactly at $t = 0$. Applying the above repeatedly therefore implies

$$||z^1|| \leq |k|A(|k|+h^2)\,, \qquad ||z^2|| \leq 2|k|A(|k|+h^2) \text{ and } ||z^j|| \leq j|k|A(|k|+h^2)\,.$$

We have therefore shown that provided $0 < r \leq \frac{1}{2}$, the maximum possible error at time $t^j \equiv jk$ is $O(k,\, h^2)t^j$, which can be made as small as we choose by choosing small enough step-lengths, k and h. Note that $r < 0$ would correspond to $k < 0$, which involves stepping *backwards in time*, which we saw in §4 leads to an ill-posed problem for this *Parabolic* equation. We have proved nothing yet about the case $r > \frac{1}{2}$, but we will find that for such values of r the FDM (5.2) is numerically unstable.

6 Explicit Method for the General Non-linear Parabolic Problem in 1-D

Last time we proved that the simple explicit method for $u_t = u_{xx}$ would work provided $r < 1/2$ where $r = k/h^2$. Experiments with the program AdvDiff.m, show that if $r > 1/2$, the method breaks down seriously as small errors get amplified out of proportion. The program also considered the effect of an advecting velocity V, solving $u_t + Vu_x = au_{xx}$. It was found that even if $r < 1/2$, the method could be unstable. We can understand that by solving the scheme exactly, using separation of variables. Suppose at time $t = 0$ there is a small error proportional to εe^{inph}, namely a *Fourier series* component and where $\varepsilon \ll 1$. Here p represents a spatial wavenumber of the Fourier wave (or mode) and h the discretisation step size. Neglecting all subsequent truncation errors R_n^j, then from (5.6), the error z_n^j obeys the equation

$$z_n^{j+1} = rz_{n+1}^j + (1-2r)z_n^j + rz_{n-1}^j, \qquad z_n^0 = \varepsilon e^{inph}. \tag{6.1}$$

We can find **separable** solutions to this equation with $z_n^j = \varepsilon \xi^j \exp(inph)$ provided

$$\xi = re^{iph} + (1-2r) + re^{-iph} = 1 - 2r(1-\cos(ph)) = 1 - 4r\sin^2(\frac{ph}{2}). \tag{6.2}$$

If this error is not to grow, then we require $|\xi| \leq 1$. Now clearly $\xi < 1$, but we need to ensure that $\xi > -1$. The worst case is when $ph = \pi$, and then we must require $1 - 4r \geq -1$ or $r \leq 1/2$. If this constraint is violated, we expect the errors to grow. Note that the worst case $ph = \pi$ corresponds to a perturbation $\exp(inph)$ which alternates between $+1$ and -1 on neighbouring grid points. This is quite a common instability of finite difference methods. This is an example of the **Fourier** or **Von Neumann stability** method. Note the Fourier method ignores boundary conditions insofar as to how they might affect the numerical stability* – nevertheless it is a powerful yet simple procedure to determine the numerical stability of the approximations using FDs.

Consider a more general nonlinear equation,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 (u^p)}{\partial x^2} \quad \text{or equivalently} \quad u_t = (u^p)_{xx}, \tag{6.3}$$

where p is some value (for example $p = 3$, say). Applying the finite difference discretisation to this gives

$$u_n^{j+1} = u_n^j + r\left[(u_{n-1}^j)^p - 2(u_n^j)^p + (u_{n+1}^j)^p\right].$$

From a stability perspective, it is easy to see that for a **linear** PDE of the form $u_t = \beta u_{xx}$ with β some constant value, the stability criterion satisfies

$$\beta r \leq \frac{1}{2} \quad \text{instead of} \quad r \leq \frac{1}{2}.$$

The **effective** diffusion coefficient of (6.3) is pu^{p-1} if we rewrite this as follows

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(pu^{p-1}\frac{\partial u}{\partial x}\right) \equiv (pu^{p-1}u_x)_x.$$

*the matrix method is a more general approach, which allows effects of boundary conditions to be also incorporated into a detailed numerical stability analysis.

This then suggests possible stability of the numerical method provided

$$pu^{p-1}r \leq \frac{1}{2}. \tag{6.4}$$

This simple intuitive analysis suggests that stability of nonlinear parabolic PDEs, not only depends of the finite difference discretisation but also generally upon the solution being obtained, which clearly will be varying as the t-variable (time say) progresses. Thus the numerical scheme may well be stable for some values of t and not for others! In practice, one monitors the stability criterion (6.4) and alters the $k = \Delta t$ size to remain below the stability constraint requirement.

Let us try to generalise the Maximum Principle Analysis (MPA) to this more general type of parabolic PDEs.

Consider the problem defined for an arbitrary function Φ,

$$u_t = \Phi(x,\ t,\ u,\ u_x,\ u_{xx}) \in 0 < x < 1 = Nh, \qquad 0 < t < T = Jk. \tag{6.5}$$

For physical sense, we will assume that the effective diffusivity is positive and bounded, so that

$$A \geq \frac{\partial \Phi}{\partial u_{xx}} \geq\ a > 0 \tag{6.6}$$

for some constants A and a. A simple, centred, explicit method replaces

$$\left.\begin{array}{lll} u_x & \text{by} & \dfrac{1}{2h}\triangle U_n^j \equiv \dfrac{U_{n+1}^j - U_{n-1}^j}{2h}, \quad (\text{note } \triangle \equiv \mu\delta) \\ u_{xx} & \text{by} & \dfrac{1}{h^2}\delta^2 U_n^j \equiv \dfrac{U_{n+1}^j + U_{n-1}^j - 2U_n^j}{h^2} \end{array}\right\} \ + \ O(h^2), \tag{6.7}$$

so that

$$U_n^{j+1} = U_n^j + k\Phi\left[nh,\ jk,\ U_n^j,\ \frac{\triangle U_n^j}{2h}, \frac{\delta^2 U_n^j}{h^2}\right] + O(k^2 + kh^2). \tag{6.8}$$

As before, we define the local error $z_n^j = u_n^j - U_n^j$. Now u obeys the same equation as U with a truncation error $R_n^j = O(k,\ h^2)$ added in. Furthermore,

$$\begin{aligned} \Phi\left[nh,\ jk,\ u_n^j,\ \frac{\triangle u_n^j}{2h},\ \frac{\delta^2 u_n^j}{h^2}\right] - \Phi\left[nh,\ jk,\ U_n^j,\ \frac{\triangle U_n^j}{2h},\ \frac{\delta^2 U_n^j}{h^2}\right], \\ = \frac{\partial \Phi}{\partial u} z_n^j + \frac{\partial \Phi}{\partial u_x}\frac{\triangle z_n^j}{2h} + \frac{\partial \Phi}{\partial u_{xx}}\frac{\delta^2 z_n^j}{h^2} + O((z)^2). \end{aligned} \tag{6.9}$$

Subtracting (6.8) from the equation involving u and using the above, gives

$$z_n^{j+1} = r\left[\frac{\partial \Phi}{\partial u_{xx}} - \frac{h}{2}\frac{\partial \Phi}{\partial u_x}\right] z_{n-1}^j + r\left[\frac{\partial \Phi}{\partial u_{xx}} + \frac{h}{2}\frac{\partial \Phi}{\partial u_x}\right] z_{n+1}^j + \left[1 + k\frac{\partial \Phi}{\partial u} - 2r\frac{\partial \Phi}{\partial u_{xx}}\right] z_n^j + kR_n^j \tag{6.10}$$

Now the Maximum Principle argument requires the three coefficients in square brackets to be positive (see the arguments of §5, Eqn. (5.6) thereafter). Suppose therefore that

$$A \geq \frac{\partial \Phi}{\partial u_{xx}} \geq\ a > 0, \qquad \left|\frac{\partial \Phi}{\partial u_x}\right| \leq\ b\,, \qquad C \geq \frac{\partial \Phi}{\partial u} \geq\ c\,, \tag{6.11}$$

where $b > 0$, c and C are constants. Then

$$\frac{\partial \Phi}{\partial u_{xx}} \pm \tfrac{1}{2} h \frac{\partial \Phi}{\partial u_x} \geq \; a - \tfrac{1}{2} hb \;\text{ and }\; 1 + k\frac{\partial \Phi}{\partial u} - 2r\frac{\partial \Phi}{\partial u_{xx}} \geq \; 1 + kc - 2rA. \tag{6.12}$$

For the MPA, we therefore require

$$a - \tfrac{1}{2} hb \geq \; 0 \;\text{ and }\; 1 + kc - 2rA \geq \; 0. \tag{6.13}$$

If (6.13) holds then we can show in the notation of §5.1 that (6.10) implies

$$\left|z_n^{j+1}\right| \leq \; \left(1 + k\frac{\partial \Phi}{\partial u}\right) \max_n \left[\, \left|z_n^j\right| \,\right] + kR_n^j \leq \; (1 + kC) \left|\left|z^j\right|\right| + D(k^2 + |k|h^2)$$

for some constant $D > 0$, and so

$$\left|\left|z^{j+1}\right|\right| \leq \; (1 + kC) \left|\left|z^j\right|\right| + D(k^2 + |k|h^2)$$

Now the initial error $||z^0|| = 0$. Consequently

$$\begin{aligned} \left|\left|z^j\right|\right| &\leq \; \left[1 + (1 + kC) + (1 + kC)^2 + \ldots + (1 + kC)^{j-1}\right] \; D(k^2 + |k|h^2) \\ &\leq e^{CT} \; jD(k^2 + |k|h^2) \leq Te^{CT} D(|k| + h^2). \end{aligned} \tag{6.14}$$

Since[†],

$$(1 + Ck)^n \leq \left(1 + \frac{CT}{n}\right)^n < e^{CT},$$

it follows that

$$\left|\left|z^j\right|\right| \leq \; \frac{e^{Cjk} - 1}{C} D(k + h^2) \qquad \text{for} \quad 1 \leq \; j \leq \; J \tag{6.15}$$

By choosing k and h small enough for fixed $T = Jk$ we can ensure that the errors are as small as we choose. Note that the exponential growth in the error term is due to the PDE itself possessing exponentially growing solutions. Compare with the equation $u_t = \nu u_{xx}$. We conclude that the explicit method will work for the nonlinear equation (6.5) provided (6.13) holds. These conditions are **sufficient** but may be overcautious.

As an example, consider Burgers' equation

$$u_t + uu_x = \nu u_{xx} \qquad \text{so that} \qquad \Phi = \nu u_{xx} - uu_x \tag{6.16}$$

where ν is constant. Then

$$A = a = \nu, \qquad b = \max_{x,t} \left[|u|\right], \qquad c = \min_{x,t} \left[-u_x\right], \; C = \max_{x,t} \left[-u_x\right] \; . \tag{6.17}$$

So the explicit method for this equation is stable if

$$\begin{gathered} \nu - \tfrac{1}{2} h \max\left[|u|\right] \geq \; 0 \qquad \text{or} \qquad |u| \leq \; 2\nu/h \\ \text{and} \quad 1 + k \min\left[-u_x\right] - 2r\nu \geq \; 0 \quad \text{or} \quad -u_x \geq \; (2r\nu - 1)/k \end{gathered} \tag{6.18}$$

The first of these conditions is a restriction on the size of the spatial step-length h for large values of the advective velocity u. In the absence of diffusion ($\nu = 0$) the centred scheme given by (6.8) is always unstable (see later). The second condition is a generalisation of the '$r < 1/2$' relation with which we are familiar. We note that if $2r\nu > 1$, no value of k will guarantee a stable scheme, and as $k \to 0$ the stability condition is violated.

[†] $k \equiv T/n$, with T a total time (say); moreover as a reminder $e^x = 1 + x + x^2/2 + x^3/3! + \ldots$

7 Implicit Scheme for the 1-D Diffusion equation

As we discussed last time, the explicit scheme (5.4) has effective characteristics with gradients $dx/dt = \pm h/k$. Arguably, this gradient should approach infinity if it is to model the physics, which might explain why the explicit scheme requires $k = O(h^2)$ for stability. Mathematically, the solution at time level $j+1$ should depend on all the values at time level j. Today we consider **implicit** methods for this problem, which do have this property. Let us choose a parameter θ and approximate $u_t = u_{xx}$ by

$$U_n^{j+1} - U_n^j = r\left[\theta\delta^2 U_n^{j+1} + (1-\theta)\delta^2 U_n^j\right]. \tag{7.1}$$

Note that the RHS now involves the unknown variables U_n^{j+1}. To find them we will have to solve a set of simultaneous linear equations (unless $\theta = 0$). Before considering how to do this numerically, we observe that for this linear problem, both the PDE and our Finite Difference approximation may be solved exactly by the method of separation of variables.

We seek solutions of the form $U_n^j = X_n T^j$. Substituting into (7.1) and separating the terms depending on j and n leads to

$$\frac{T^{j+1} - T^j}{r\left[\theta T^{j+1} + (1-\theta)T^j\right]} = \frac{X_{n+1} - 2X_n + X_{n-1}}{X_n} = \sigma, \quad \text{say.} \tag{7.2}$$

As the LHS varies with j but not n, and the RHS the other way round, σ must be a constant. X_n therefore obeys the second order difference equation

$$X_{n+1} - (\sigma+2)X_n + X_{n-1} = 0, \tag{7.3}$$

which has solutions of Fourier form $X_n = e^{\pm in\xi}$ provided $\sigma + 2 = 2\cos\xi$. In general ξ is arbitrary, but if we impose the boundary conditions $X_0 = 0 = X_N$, then we can show that

$$\xi = \frac{m\pi}{N} = m\pi h \qquad \text{for} \quad m = 1, 2\ldots \quad \text{and} \quad \sigma = -4\sin^2\left(\frac{m\pi h}{2}\right) = \sigma_m, \tag{7.4}$$

say. Then X_n is given by

$$X_n = A\sin(nm\pi h). \tag{7.5}$$

For each such permissible value $\sigma = \sigma_m$, T^j can be found from (7.2).

$$T^{j+1} = \lambda_m T^j, \qquad \text{so that} \quad T^j = C(\lambda_m)^j \qquad \text{where} \quad \lambda_m = \left[\frac{1+\sigma_m r(1-\theta)}{1-\sigma_m r\theta}\right]. \tag{7.6}$$

Now we have

$$\lambda_m = \frac{1 - 4r(1-\theta)\sin^2\frac{1}{2}\xi}{1 + 4r\theta\sin^2\frac{1}{2}\xi} = 1 - \frac{4r\sin^2\frac{1}{2}\xi}{1 + 4r\theta\sin^2\frac{1}{2}\xi}. \tag{7.7}$$

The stability requirement is that $|\lambda_m| \le 1$ for all m. Clearly $\lambda_m \le 1$, while $\lambda_m \ge -1$ if

$$1 + 4r\theta\sin^2\tfrac{1}{2}\xi \;\ge\; 2r\sin^2\tfrac{1}{2}\xi, \tag{7.8}$$

or

$$(1-2\theta)2r\sin^2\tfrac{1}{2}\xi \;\le\; 1. \tag{7.9}$$

If $\theta \ge \frac{1}{2}$, therefore, the FDM (7.1) is **unconditionally stable**.

If on the other hand zz

$$0 \leq \theta < \tfrac{1}{2},$$

stability for all m and h can be guaranteed only if

$$r \leq \frac{1}{2(1-2\theta)}. \tag{7.10}$$

Note that when $\theta = 0$ we recover the relation $r \leq \frac{1}{2}$ and (7.1) is then **conditionally stable**.

We can obtain the general solution by taking an arbitrary linear sum,

$$U_n^j = \sum_{m=1}^{\infty} B_m \sin(nm\pi h)(\lambda_m)^j, \tag{7.11}$$

where the coefficients B_m can be found from the Fourier expansion of the initial condition $u_0(x)$ of (7.1). We can now compare (7.11) with the exact solution evaluated at the grid points

$$U_n^j \equiv u(nh,\, jk) = \sum_{m=1}^{\infty} B_m \sin(nm\pi h)(e^{-m^2\pi^2 k})^j \tag{7.12}$$

The accuracy of the F.D. approximation may thus be determined by comparing λ_m and $\exp(-m^2\pi^2 k)$.

We see that it is the high modes, the large values of m for which $m \sim N$ and $\xi \sim \pi$ which are most likely to be unstable. This is typical behaviour for FDMs; instabilities tend to manifest themselves on the scale of the grid. The low modes, for which $m = O(1)$, and ξ is small are modelled well. For them we may approximate $\sin\beta \approx \beta - \frac{1}{6}\beta^3$ to give

$$\begin{aligned} \lambda_m &\approx 1 - \frac{r(\xi - \frac{1}{24}\xi^3 + O(\xi^5))^2}{1 + r\theta\xi^2 + O(r\xi^4)}, \\ &= 1 - r\xi^2(1 - \tfrac{1}{12}\xi^2)(1 - r\theta\xi^2) + O(r^3\xi^6). \\ &= 1 - m^2\pi^2 k + \left(\theta + \frac{1}{12r}\right) m^4\pi^4 k^2 + O(m^6 k^3) \\ \text{while} \quad e^{-m^2\pi^2 k} &= 1 - m^2\pi^2 k + \tfrac{1}{2}m^4\pi^4 k^2 + O(m^6 k^3). \end{aligned} \tag{7.13}$$

The agreement between λ_m and $\exp(-m^2\pi^2 k)$ is quite good, while the greatest accuracy is achieved if

$$\theta = \frac{1}{2} - \frac{1}{12r} \qquad \text{or} \qquad r(1-2\theta) = \frac{1}{6}. \tag{7.14}$$

We see from (7.9) that such a scheme would be stable. It is the generalisation of Milne's method ($\theta = 0,\ r = \frac{1}{6}$). If r is large, then the Crank-Nicolson scheme ($\theta = \frac{1}{2}$) is close to optimal.

Most importantly, using implicit methods we can produce stable schemes with $k \sim h$, and hence large values of $r = k/h^2$. Compared with the explicit schemes, we may choose relatively large time-steps.

7.1 The Crank-Nicolson method and (nearly) Tridiagonal systems

If we choose an unconditionally stable scheme with $\theta > 1/2$, then we may choose the time-step as large as we like, and our only concern is the accuracy of our approximation of the derivatives. A popular (and sensible) choice is the **Crank-Nicolson** scheme, $\theta = 1/2$. This scheme is centred about the time-level $(j + 1/2)$, and so is second-order accurate in both space and time, $R_n^j = O(k^2, h^2)$. The price we pay for an implicit scheme is that each time-step we have to solve some simultaneous linear equations. However, as the system is tri-diagonal this is not so hard. If we represent a list of all the U-values at time j by a vector U^j , then the system we need to solve is

$$AU^{j+1} = BU^j, \quad \text{where} \quad U^j = (U_1^j, U_2^j, \ldots, U_N^j)^T$$

for suitable matrices A and B. In this problem, A and B are tridiagonal, which permits efficient solution. The Crank-Nicolson ($\theta = 1/2$) method for the equation $u_t = u_{xx}$ has

$$\mathcal{A} = \begin{pmatrix} 1+r & -r/2 & 0 & \ddots & 0 \\ -r/2 & 1+r & -r/2 & \ddots & \ddots \\ 0 & -r/2 & 1+r & -r/2 & 0 \\ \ddots & \ddots & \ddots & \ddots & -r/2 \\ 0 & \ddots & 0 & -r/2 & 1+r \end{pmatrix},$$

To solve $Ax = b$, where A is an $M \times M$ matrix. usually requires $O(M^3)$ operations. Here, however, the sparseness and structure of A renders the process much more efficient. Using Gaussian elimination, subtracting $(-r/2)/(1 + r)$ times the first row from the second transforms A_{21} to zero and alters A_{22} and b_2. Then subtracting a suitable multiple of the 2nd row from the 3rd and continuing, leaves us with

$$\mathcal{A} = \begin{pmatrix} 1+r & -r/2 & 0 & \ddots & 0 \\ 0 & a_2 & -r/2 & \ddots & \ddots \\ 0 & 0 & 1+r & a_3 & 0 \\ \ddots & \ddots & \ddots & \ddots & -r/2 \\ 0 & \ddots & 0 & 0 & a_n \end{pmatrix} x = b^*,$$

where a_i and b^* are known values. The last equation is now trivial, $a_n x_n = b_n^*$, while the penultimate term $a_{n-1}x_{n-1} = b_{n-1}^* + (r/2)x_n$ which we now know, giving us x_{n-1}. Systematically back-substituting determines all the x_i unknowns.

The algorithm, also known as the **Thomas algorithm**, is a simplified form of Gaussian elimination that can be used to solve tridiagonal systems of equations. For such systems, the solution can be obtained in $O(M)$ operations instead of $O(M^3)$ required by Gaussian elimination. See the routine tridiag.m.

7.2 Aside – Separation of variables solution to Diffusion eqn.

We consider a slight modification of (7.1), namely:

$$\begin{gathered} u_t = u_{xx} \in 0 < x < 1,\ t > 0, \\ \text{with}\ \ u(0,t) = u(1,t) = 0, \quad u(x,0) = u_0(x). \end{gathered} \tag{7.15}$$

The PDE has separable solutions of the form $u(x,t) = X(x)T(t)$ provided

$$XT' = X''T \qquad \text{or} \qquad \frac{T'}{T} = \frac{X''}{X} = -\omega^2, \qquad \text{say.}$$

As T'/T is a function of t only, while X''/X is a function of x only, both functions must be a constant, which we take to be negative. Then the functions $X(x)$ and $T(t)$ take the forms

$$X = A\cos\omega x + B\sin\omega x, \ \text{ and } \ T = Ce^{-\omega^2 t}.$$

If we require X to obey the boundary conditions in (7.3), namely $X(0) = X(1) = 0$, we obtain non-zero solutions only if $A = 0$ and $\omega = m\pi$, for some integer m, so that

$$u = B_m \sin m\pi x\, e^{-m^2\pi^2 t},$$

for some constant B_m. As (7.15) is a linear problem, we may combine solutions to obtain a more general solution in the form

$$u(x,\, t) = \sum_{m=1}^{\infty} B_m \sin m\pi x\, e^{-m^2\pi^2 t}. \tag{7.16}$$

The initial condition will be satisfied if

$$u(x,\, 0) = \sum_{m=1}^{\infty} B_m \sin m\pi x = u_0(x). \tag{7.17}$$

Thus all we need do to obtain the solution of (7.15) is to expand the initial condition $u = u_0(x)$ in a Fourier series, and substitute the appropriate values of the constants B_m into (7.16).

8 Implicit Method for Nonlinearities

How should we adapt our implicit methods for nonlinear PDEs? While it would be possible to solve nonlinear equations each time-step, that is not usually necessary, or advisable. Instead, we can seek linear approximations whose errors are the same order as our original truncation. For example, consider

$$\frac{\partial u}{\partial t} = \mathcal{D}\frac{\partial^2 u}{\partial x^2} + f(u)$$

for some function $f(u)$.

As an example, the Fisher-Kolmogorov-Petrovsky-Piskunov (Fisher-KPP) equation with

$$f(u) = ru\left(1 - \frac{u}{K}\right)$$

is one of the simplest examples of a nonlinear reaction-diffusion equation. Fisher proposed this as a model of diffusion of species in a 1-D habitat; $\mathcal{D}$ is the diffusion constant, r is the growth rate of the species, and K is the carrying capacity. A dimensionless version takes the form

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + u(1-u).$$

Model forms of the equation arise in diverse physical, chemical and biological phenomena. In flame propagation

$$f(u) = \frac{\beta^2}{2}u(1-u)e^{-\beta(1-u)}, \text{ with } \beta \text{ known as the Zeldovich number.}$$

Similar forms arise in nuclear reactors and neutron population modelling, chemical reaction and Brownian-motion type problems.

The simplest thing to do would be to treat the nonlinear term $f(u)$ explicitly. Alternatively, if using a Crank-Nicolson formulation, we might expect best accuracy if we centre $f(u)$ about $j + \frac{1}{2}$, writing

$$U_n^{j+1} - U_n^j = \frac{r}{2}\left(\delta^2 U_n^j + \delta^2 U_n^{j+1}\right) + \frac{k}{2}\left(f(U_n^j) + f(U_n^{j+1})\right). \tag{8.1}$$

If $f(u)$ is differentiable, we could then approximate the nonlinear unknown term by

$$f(U_n^{j+1}) = f(U_n^j) + f'(U_n^j)\left(U_n^{j+1} - U_n^j\right) + O\left(|U_n^{j+1} - U_n^j|^2\right). \tag{8.2}$$

Combining Eqns. 8.1 and 8.2, U_n^{j+1} now appears linearly on the RHS. We can then modify the matrices $\mathcal{A}$ and $\mathcal{B}$ developed earlier:

$$\mathcal{A}\ U^{j+1} = \mathcal{B}\ U^j + c^j$$

accordingly. Note that in general, the stability of the method depends on the eigenvalues of $\mathcal{A}^{-1}\mathcal{B}$, which may change if we alter $\mathcal{A}$ and $\mathcal{B}$.

9 Matrix Representation. Neumann and Periodic Boundary Conditions

A general, implicit one–step algorithm can be represented in the form

$$\sum_{n=1}^{N-1} A_{mn} U_n^{j+1} = \sum_{n=1}^{N-1} B_{mn} U_n^j + c_m^j \qquad \text{for} \quad 1 \le m \le N-1, \quad 1 \le j \le J\,. \tag{9.1}$$

The convergence of the solution can be investigated by deriving a difference equation for the discretisation error z. Denote the exact solution of the PDE by u and the exact solution of the finite-difference by U. Then $z = u - U$ defines the error. At the mesh points

$$U_{ij} = u_{ij} - z_{ij}, \quad U_{ij+1} = u_{ij+1} - z_{ij+1}, \quad \text{etc.}$$

Thus, we introduce the solution and truncation error vectors

$$\mathbf{Z}^j = (z_1^j,\ z_2^j, \ldots,\ z_{N-1}^j)$$

and

$$\varepsilon \mathbf{d}^j = (R_1^j,\ R_2^j, \ldots,\ R_{N-1}^j)$$

where ε is small, typically $\varepsilon = O(k^2 + kh^2)$. $\mathcal{A}$ and $\mathcal{B}$ are $(N-1) \times (N-1)$ matrices and then the error–vector obeys the equation

$$\mathcal{A}\mathbf{Z}^{j+1} = \mathcal{B}\mathbf{Z}^j + \varepsilon \mathbf{d}^j. \tag{9.2}$$

Assuming the matrix inverse $\mathcal{A}^{-1}$ exists (else the entire method collapses as we will be unable to find U_n^{j+1} from U_n^j) we can write

$$\mathbf{Z}^{j+1} = \left(\mathcal{A}^{-1}\mathcal{B}\right)\mathbf{Z}^j + \varepsilon \mathcal{A}^{-1}\mathbf{d}^j = P\mathbf{Z}^j + \varepsilon \mathbf{f}^j \qquad \text{say.} \tag{9.3}$$

The initial error, $\mathbf{Z}^0 = 0$. Applying the above relation iteratively leads to

$$\mathbf{Z}^j = \varepsilon \left[(P)^{j-1}\mathbf{f}^0 + (P)^{j-2}\mathbf{f}^1 + \cdots + P\mathbf{f}^{j-2} + \mathbf{f}^{j-1}\right]\,. \tag{9.4}$$

The matrix P describes the *propagation* of errors from one time–step to the next. For the method, given by Eqn. 9.1, to be stable, we must have $\mathbf{Z}^J \to 0$ as $J \to \infty$ and $k \to 0$. We therefore want to consider the vector $(P)^j\mathbf{x}$ where $\mathbf{x}$ is any vector and j is large.

Suppose the matrix P has eigenvectors $\mathbf{e}_m$ and eigenvalues λ_m. Then

$$P\mathbf{e}_m = \lambda_m \mathbf{e}_m\,, \qquad (P)^2\mathbf{e}_m = \lambda_m^2 \mathbf{e}_m \ \text{ and } \ (P)^j\mathbf{e}_m = (\lambda_m)^j\mathbf{e}_m\,. \tag{9.5}$$

Thus, if one of the eigenvalues has modulus greater than one it is possible for $|(P)^j\mathbf{x}|$ to increase without limit. If this happens the method (*i.e.* Eqn. 9.1) will be unstable (although see the subtlety overleaf.) Conversely, suppose that $|\lambda_m| \le 1$ for all m. Then it is easy to show that when the eigenvectors of P are *complete* so that we may expand $\mathbf{x} = \sum a_m \mathbf{e}_m$, then

$$\left|(P)^j\mathbf{x}\right| = \left|\sum_{m=1}^{N-1} a_m (P)^j \mathbf{e}_m\right| = \left|\sum a_m (\lambda_m)^j \mathbf{e}_m\right| \ \le \sum |a_m||\lambda_m|^j \le \sum |a_m|\,. \tag{9.6}$$

This is bounded and so the error vector $\mathbf{Z}^j$ in Eqn. 9.6 remains small. Indeed, if all the eigenvalues are strictly less than one in modulus, then the propagated errors decrease as j increases. The solution error is then dominated by the local truncation error.

We have shown that solution by solving Eqn. 9.1 is stable provided $\max|\lambda_m| \leq 1$ and the eigenvectors are complete. In fact this result is true for incomplete sets of eigenvectors also, but is harder to prove. The maximum value of $|\lambda_m|$ is often called the **spectral radius** of P, and sometimes written as $\rho(P)$. When calculating the eigenvalues, it is usually best to consider the equation $|\mathcal{B} - \lambda\mathcal{A}|= 0$ rather than calculating P.

A subtlety: Bounded growth. Even if the spectral radius is a little greater than one, the method may still be stable, provided the errors grow in a bounded fashion. Consider a time interval $0 < t < T = Jk$. The errors over this period are bounded if for all m and some constant G,

$$|\lambda_m|^J \leq G \iff |\lambda_m| \leq G^{1/J} = e^{(\ln G)k/T} \approx 1 + k\frac{\ln G}{T} + O(k^2) \qquad \text{as} \quad k \to 0\,. \quad (9.7)$$

The stability condition allowing for bounded growth is $|\lambda_m| \leq 1 + O(k)$.

9.1 Eigenvalues of a Toeplitz matrix

The matrix method is a general, powerful and precise method of determining stability of a finite difference scheme. Unlike the Fourier method, it takes into account the boundary conditions. In general, the eigenvalues may be harder to find than in our example, but it is often possible to determine whether or not they all lie in the circle $|\lambda| \leq 1$ without calculating them explicitly.

Matrices with constant coefficients along its diagonals (running from upper left to lower right) are known as *Toeplitz* matrices. For the special case of finite tri-diagonal $N \times N$ Toeplitz matrices of the form

$$\mathsf{Tp}\,[a,\ b,\ c] = \begin{pmatrix} a & b & 0 & \ddots & 0 \\ c & a & b & \ddots & 0 \\ 0 & c & a & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & b \\ 0 & 0 & \ddots & c & a \end{pmatrix},$$

it can be shown that the eigenvalues, of a *circulant* matrix, of Tp $[a,\ b,\ c]$ form are

$$\lambda_m = a + 2\sqrt{bc}\cos(m\pi/N) \ \text{ for } \ 1 \leq m \leq N-1.$$

The corresponding eigenvector v_m has the jth component

$$v_{mj} = \left(\sqrt{c/b}\right)^j \sin(m\pi j/N), \ \text{ for } \ 1 \leq j \leq N-1,$$

which become increasingly oscillatory for large values of mj.

(Nearly) Tridiagonal systems

So far we have been assuming that u is known on the boundaries. The condition $u = g$ on a boundary is called a **Dirichlet** condition. We have shown, that discretising the diffusion equation, when imposing Dirichlet conditions, gives rise to matrix entries of $\mathcal{A}$ and $\mathcal{B}$ being tridiagonal, which permits efficient solution using the Thomas algorithm. The Crank-Nicolson ($\theta = \frac{1}{2}$) method for the equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{9.8}$$

had

$$\mathcal{A} = \begin{pmatrix} 1+r & -\mu r & 0 & \ddots & d \\ -r/2 & 1+r & -r/2 & \ddots & 0 \\ 0 & -r/2 & 1+r & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & -r/2 \\ d & 0 & \ddots & -\mu r & 1+r \end{pmatrix}, \tag{9.9}$$

where $\mu = 1/2$ and $d = 0$.

For more general problems, the diagonal entries (*i.e.* the $1 + r$ entries above) may vary with the row number, while maintaining the tridiagonal structure. In instances where non-Dirichlet conditions have to be satisfied at boundaries, the matrix may retain the tridiagonal structure but also have additional non-zero entries **off** the tridiagonal matrix entries, *i.e.* $d \neq 0$ in $\mathcal{A}$ above.

9.2 Neumann boundary conditions

An important alternative are the **Neumann** boundary conditions :

$$\frac{\partial u}{\partial x} \equiv u_x = 0 \text{ at } x = \Gamma,$$

or **Mixed-type** conditions (sometimes referred to as a **Robin** condition) :

$$\frac{\partial u}{\partial n} + \gamma u = f, \text{ applied at a boundary } \Gamma, \text{ with } n \text{ possibly normal to } \Gamma,$$

and where $f \neq 0$ and γ may also have some non-zero value. Physically, $f = 0$ corresponds to no flux across the boundary, so that if u corresponds to temperature, this would be an insulating boundary.

In this case the value of u is unknown on the boundary, and has to be computed as part of the solution. Slight differences occur to our matrix $\mathcal{A}$ system in this case. One could require that $U_0^j = U_1^j$, for example, which keeps the same number of unknowns as in the Dirichlet case. This would change the (1,1) element of $\mathcal{A}$ to $1+\frac{1}{2}r$ (and $\mu = \frac{1}{2}$). A better way of dealing with this condition is to introduce a fictitious point at $n = -1$, and then require $U_{-1} = U_1$. Then $(\delta^2 u)_0 = 2u_1 - 2u_0$, which we can use in the FD scheme evaluated on the boundary. If we have Neumann conditions at both ends, we would then have a matrix $\mathcal{A}$ given by (9.9) where $\mu = 1$ and $d = 0$. However, in that case we may have 2 more rows and columns in $\mathcal{A}$ as we have 2 more unknowns.

9.3 Periodic boundary conditions

Another important case to consider are **Periodic** boundary conditions, which need to be imposed at the boundaries (say $x = 0$ and $x = 1$). The new boundary value $U_N = U_0$, and the fictitious points U_{-1} and U_{N+1} are identified as U_{N-1} and U_1 respectively, since the derivatives at each end must also agree. When we evaluate the FDM on the boundary, we get an extra equation, and the matrix $\mathcal{A}$ takes the form (9.9) but with the top right and bottom left corners changed and having *non-zero* entries, $\mu = 1/2$ and $d = -\frac{1}{2}r$. The matrix is no longer strictly tri-diagonal, and we have to modify our solving routine.

Thomas algorithm for periodic systems:
Periodic boundary conditions are inherently more symmetric than either Neumann or Dirichlet conditions, and it perhaps surprising that the resulting system of equations is harder to deal with. Nevertheless, a relatively simple modification of the Thomas algorithm for tridiagonal matrices suffices, whenever it is only the outermost rows and columns of the matrix which are not tri-diagonal. We write the $N \times N$ system $\mathbf{A}\,\mathbf{x} = \mathbf{b}$ in the form

$$\begin{pmatrix} a_{11} & | & \rightarrow & \mathbf{v}_1^T & \rightarrow & | & a_{1N} \\ --- & + & --- & + & --- & + & --- \\ \downarrow & \vdots & & & & \vdots & \downarrow \\ \mathbf{v}_2 & + & & \mathbf{A}' & & + & \mathbf{v}_3 \\ \downarrow & \vdots & & & & \vdots & \downarrow \\ --- & + & --- & + & --- & + & --- \\ a_{N1} & | & \rightarrow & \mathbf{v}_4^T & \rightarrow & | & a_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ --- \\ \downarrow \\ \mathbf{x}' \\ \downarrow \\ --- \\ x^N \end{pmatrix} = \begin{pmatrix} b_1 \\ --- \\ \downarrow \\ \mathbf{b}' \\ \downarrow \\ --- \\ b^N \end{pmatrix}$$

where $\mathbf{A}'$ is a tridiagonal $(N-2) \times (N-2)$ matrix and $\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4, \mathbf{x}'$ and $\mathbf{b}'$ are $(N-2)$-vectors. Then in component form, the system reads

$$\begin{aligned} a_{11}x_1 + \mathbf{v}_1 \cdot \mathbf{x}' + a_{1N}x_N &= b_1, \\ x_1\mathbf{v}_2 + \mathbf{A}'\mathbf{x}' + x_N\mathbf{v}_3 &= \mathbf{b}', \\ a_{N1}x_1 + \mathbf{v}_4 \cdot \mathbf{x}' + a_{NN}x_N &= b_N. \end{aligned} \tag{9.10}$$

Now the middle $(N-2)$ equations can be written as

$$\mathbf{A}'\mathbf{x}' = \mathbf{b}' - x_1\mathbf{v}_2 - x_N\mathbf{v}_3. \tag{9.11}$$

We can now solve 3 sets of tridiagonal systems, and find the solutions $\mathbf{y}_i$ to

$$\mathbf{A}'\mathbf{y}_1 = \mathbf{b}', \quad \mathbf{A}'\mathbf{y}_2 = \mathbf{v}_2, \quad \mathbf{A}'\mathbf{y}_3 = \mathbf{v}_3. \tag{9.12}$$

Then Eqn. 9.11 has the solution

$$\mathbf{x}' = \mathbf{y}_1 - x_1\mathbf{y}_2 - x_N\mathbf{y}_3,$$

where x_1 and x_N are still unknown. But now we can substitute this expression into the first and last equations in Eqn. 9.10 to obtain two simultaneous equations for x_1 and x_N

$$\begin{aligned} (a_{11} - \mathbf{v}_1 \cdot \mathbf{y}_2)\, x_1 + (a_{1N} - \mathbf{v}_1 \cdot \mathbf{y}_3)\, x_N &= b_1 - \mathbf{v}_1 \cdot \mathbf{y}_1 \\ (a_{1N} - \mathbf{v}_4 \cdot \mathbf{y}_2)\, x_1 + (a_{NN} - \mathbf{v}_4 \cdot \mathbf{y}_3)\, x_N &= b_N - \mathbf{v}_4 \cdot \mathbf{y}_1 \end{aligned}$$

which we solve, and substitute in the expression for $\mathbf{x}'$. This algorithm therefore requires $3 \times O(N)$ operations, which is faster than the ordinary solving methods. This algorithm is implemented in periodic_tridiag.m and periodic_CN.m. Essentially, all we are doing is solving for $x_2, \ldots, x_{N-1}$ in terms of x_1 and x_N and then substituting in the other 2 equations.

The same ideas can be used for penta-diagonal systems. You could write a 5-diagonal solver and then modify it for the periodic boundary conditions on similar lines to the above. Alternatively, you can opt to use the simpler, slower $\mathbf{x} = \mathbf{A}\backslash\mathbf{b}$ function call in Matlab (say).

10 Diffusion in More Dimensions; the ADI Method

We now have the understanding to begin to tackle more dimensions. Let us consider the two-dimensional diffusion problem

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}, \tag{10.1}$$

or in suffix notation

$$u_t = \nabla^2 u \equiv u_{xx} + u_{yy} \quad \text{in } \; 0 < x < 1, \;\; 0 < y < L, \;\; t > 0, \tag{10.2}$$

with Dirichlet conditions $u = 0$ on the 4 sides of the rectangle and some initial condition $u = f(x,\, y)$ at $t = 0$.

We define a spatial grid of size $(h_x,\, h_y)$ such that $Mh_x = 1$, $Nh_y = L$. We then seek an approximation U^j_{mn} to $u(mh_x,\, nh_y,\, jk)$. We use

$$\begin{aligned} u_{xx} \simeq = \frac{\delta_x^2 u_{mn}}{h_x^2} &\equiv \frac{U_{m+1n} + U_{m-1n} - 2U_{mn}}{h_x^2}, \\ u_{yy} \simeq = \frac{\delta_y^2 u_{mn}}{h_y^2} &\equiv \frac{U_{mn+1} + U_{mn-1} - 2U_{mn}}{h_y^2}. \end{aligned} \tag{10.3}$$

For simplicity we shall assume here that $h_x = h_y = h$ and that $L \geq 1$ is an integer, but it is easy to generalise.

The explicit scheme for this problem therefore takes the form

$$U^{j+1}_{mn} = U^j_{mn} + r(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{10.4}$$

We can analyse its stability using the Fourier method. We can find solutions of the form

$$U_{mn^j} = (\lambda)^j e^{im\xi} e^{in\eta}$$

where

$$\lambda = 1 - 4r\left(\sin^2 \tfrac{1}{2}\xi + \sin^2 \tfrac{1}{2}\eta\right) \tag{10.5}$$

The worst case is $\xi = \pi$, $\eta = \pi$, and we require

$$r \leq \tfrac{1}{4}$$

for stability. As usual, the explicit method requires $k = O(h^2)$ and is slow.

As before, If we evaluate the RHS at least partially at the new time-level $(j + 1)$, we can improve stability. For example the θ-method

$$U^{j+1}_{mn} - r\theta(\delta_x^2 + \delta_y^2)U^{j+1}_{mn} = U^j_{mn} + r(1 - \theta)(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{10.6}$$

is unconditionally stable if

$$\theta \geq \tfrac{1}{2},$$

so we may choose k larger. However, there is an important difference compared with the 1-D diffusion equation. The LHS of (10.6) is **pentadiagonal** but with a gap between the main 3 diagonals and the other 2 with non-zero entries. Direct methods of solution of (10.6)

are significantly less efficient, while the size of the matrix is much larger – there are now $(M \times 1)(N \times 1)$ unknowns.

Clearly these effects are multiplied in 3-dimensions. If you want to solve on a $100 \times 100 \times 100$ grid you have 10^6 unknowns. Direct solution methods will need $O(10^{18})$ calculations. How long will this take on your computer? This would be inefficient, possibly even worse than an explicit method. We must consider ways of speeding things up.

If we are interested in the final state only, as we will be when we consider elliptic PDEs, then there are various faster methods available. Today, we consider a simple improvement for the time-dependent diffusion equation, known as the **Alternating Direction Implicit (ADI)** method.

10.1 Implicit ADI

The fully implicit method involves solving a large number, $(M-1)\times(N-1)$, of simultaneous equations each time-step. An important alternative is the ADI method, whose basic idea is to be implicit in only one of the $x,\ y$ directions at a time, and to alternate between the two. In a complete cycle, it is only necessary to solve $(N-1)$ systems of size $(M-1)$ and then $(M-1)$ systems of $(N-1)$ equations, which is considerably more efficient. Furthermore, the systems of equations are tri-diagonal.

During the first half time-step we are implicit in x, say, so that

$$\frac{U_{mn}^{j+1/2} - U_{mn}^{j}}{k/2} = \frac{1}{h^2}\left(\delta_x^2 U_{mn}^{j+1/2} + \delta_y^2 U_{mn}^{j}\right) \tag{10.7}$$

or

$$\left(1 - \tfrac{1}{2}r\delta_x^2\right) U_{mn}^{j+1/2} = (1 + \tfrac{1}{2}r\delta_y^2) U_{mn}^{j}\ . \tag{10.8}$$

Over the next half time-step we treat y implicitly, so that

$$\frac{U_{mn}^{j+1} - U_{mn}^{j+1/2}}{k/2} = \frac{1}{h^2}\left(\delta_x^2 U_{mn}^{j+1/2} + \delta_y^2 U_{mn}^{j+1}\right) \tag{10.9}$$

or

$$\left(1 - \tfrac{1}{2}r\delta_y^2\right) U_{mn}^{j+1} = (1 + \tfrac{1}{2}r\delta_x^2) U_{mn}^{j+1/2}\ . \tag{10.10}$$

If we want, we can eliminate the values at the half-time level by operating on (10.8) with $(1 + \frac{1}{2}r\delta_x^2)$ and (10.10) with $(1 - \frac{1}{2}r\delta_x^2)$ to obtain

$$\left(1 - \tfrac{1}{2}r\delta_x^2\right)\left(1 - \tfrac{1}{2}r\delta_y^2\right) U_{mn}^{j+1} = (1 + \tfrac{1}{2}r\delta_x^2)(1 + \tfrac{1}{2}r\delta_y^2) U_{mn}^{j}\ . \tag{10.11}$$

We now consider stability of the scheme to a double Fourier perturbation of the form

$$U_{mn}^{j} = \widehat{U}^{j} \exp(im\xi + in\eta).$$

In the familiar manner, the operator δ_x^2 becomes $-4\sin^2\frac{1}{2}\xi$, while δ_y^2 becomes $-4\sin^2\frac{1}{2}\eta$. Then (10.8) and (10.10) imply

$$\left.\begin{aligned} \widehat{U}^{j+1/2} &= \frac{1 - 2r\sin^2\frac{1}{2}\eta}{1 + 2r\sin^2\frac{1}{2}\xi}\,\widehat{U}^{j} = \lambda_1 \widehat{U}^{j} \\ \widehat{U}^{j+1} &= \frac{1 - 2r\sin^2\frac{1}{2}\xi}{1 + 2r\sin^2\frac{1}{2}\eta}\,\widehat{U}^{j+1/2} = \lambda_2 \widehat{U}^{j+1/2} \end{aligned}\right\} \text{ say.} \tag{10.12}$$

Then $|\lambda_1| \leq 1$ for all ξ, η only if $r \leq 1$, which is equivalent to "$r \leq \frac{1}{2}$" for a time-step $\frac{1}{2}k$ rather than k. λ_2 is similar. However, over the complete time-step,

$$\widehat{U}^{j+1} = \lambda_1 \lambda_2 \widehat{U}^j$$

and

$$|\lambda_1 \lambda_2| = \left| \frac{1 - 2r \sin^2 \frac{1}{2}\xi}{1 + 2r \sin^2 \frac{1}{2}\xi} \right| \left| \frac{1 - 2r \sin^2 \frac{1}{2}\eta}{1 + 2r \sin^2 \frac{1}{2}\eta} \right| \leq 1 \tag{10.13}$$

for every ξ and η. Thus, although each half time-step is unstable if repeated indefinitely, the alternation of one and then the other is unconditionally stable!

10.2 ADI in three dimensions

The obvious generalisation to three dimensions would be

$$\left.\begin{aligned} \left(1 - \tfrac{1}{3}r\delta_x^2\right) U_{lmn}^{j+1/3} &= \left(1 + \tfrac{1}{3}r(\delta_y^2 + \delta_z^2)\right) U_{lmn}^{j} \\ \left(1 - \tfrac{1}{3}r\delta_y^2\right) U_{lmn}^{j+2/3} &= \left(1 + \tfrac{1}{3}r(\delta_x^2 + \delta_z^2)\right) U_{lmn}^{j+1/3} \\ \left(1 - \tfrac{1}{3}r\delta_z^2\right) U_{lmn}^{j+1} &= \left(1 + \tfrac{1}{3}r(\delta_x^2 + \delta_y^2)\right) U_{lmn}^{j+2/3} \end{aligned}\right\} \tag{10.14}$$

However, such a scheme turns out **not** to be unconditionally stable, and so is little improvement on explicit schemes. If we consider

$$U_{lmn}^{j} = \widehat{U}^j \exp(il\xi + im\eta + in\tau),$$

we get

$$\widehat{U}^{j+1/3} = \lambda_1 \widehat{U}^j \quad \text{etc. where} \quad \lambda_1 = \frac{1 - \frac{4}{3}r(\sin^2 \frac{1}{2}\eta + \sin^2 \frac{1}{2}\tau)}{1 + \frac{4}{3}r \sin^2 \frac{1}{2}\xi} = \frac{1 - b - c}{1 + a}, \tag{10.15}$$

Then

$$|\lambda_1 \lambda_2 \lambda_3| = \left| \frac{(1 - b - c)(1 - a - b)(1 - a - c)}{(1 + a)(1 + b)(1 + c)} \right| . \tag{10.16}$$

Stability occurs only if a, b, c are small enough, with the worst case being $\xi = \eta = \tau = \pi$, when the necessary condition is

$$r \leq \frac{3}{4}.$$

A more successful generalisation is obtained by using a Crank-Nicolson idea ($\theta = \frac{1}{2}$) in the implicit direction. Consider the 2-D scheme

$$\left.\begin{aligned} U_{mn}^{*j+1} - U_{mn}^{j} &= \tfrac{1}{2}r\delta_x^2(U_{mn}^{*j+1} + U_{mn}^{j}) + r\delta_y^2 U_{mn}^{j} \\ U_{mn}^{j+1} - U_{mn}^{j} &= \tfrac{1}{2}r\delta_x^2(U_{mn}^{*j+1} + U_{mn}^{j}) + \tfrac{1}{2}r\delta_y^2(U_{mn}^{j+1} + U_{mn}^{j}) \end{aligned}\right\} \tag{10.17}$$

If we eliminate U^* from (10.17) we obtain the same relation (10.11) between U^{j+1} and U^j. However, whereas the intermediate value $U^{j+1/2}$ in (10.8) is an approximation to the real solution at the $j + \frac{1}{2}$ time-level, the intermediate values in (10.17) should be regarded as a first estimate of the solution at $j + 1$, which is subsequently improved upon.

In three dimensions, a working scheme to find $U(x,\, y,\, z)$ satisfying $u_t = \nabla^2 u$ involves two intermediate estimates U^* and U^*_*,

$$\left.\begin{aligned}
(1-\tfrac{1}{2}r\delta_x^2)U^{*j+1}_{lmn} &= \left[1+r(\tfrac{1}{2}\delta_x^2+\ \ \delta_y^2+\delta_z^2)\right]U^{\ j}_{lmn} \\
(1-\tfrac{1}{2}r\delta_y^2)U^{*j+1}_{*lmn} &= \left[1+r(\tfrac{1}{2}\delta_x^2+\tfrac{1}{2}\delta_y^2+\delta_z^2)\right]U^{\ j}_{lmn}+\tfrac{1}{2}r\delta_x^2U^{*j+1}_{lmn} \\
(1-\tfrac{1}{2}r\delta_z^2)U^{\ j+1}_{lmn} &= \left[1+\tfrac{1}{2}r(\delta_x^2+\ \ \delta_y^2+\delta_z^2)\right]U^{\ j}_{lmn}+\tfrac{1}{2}r\delta_x^2U^{*j+1}_{lmn}+\tfrac{1}{2}r\delta_y^2U^{*j+1}_{*lmn}
\end{aligned}\right\}. \quad (10.18)$$

This scheme is unconditionally stable and only requires solving tridiagonal systems.

11 Elliptic PDEs : Computer Solution by Iteration

Consider the elliptic PDE in Cartesian (x, y) coordinates :

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0. \tag{11.1}$$

A simple centered finite-difference discretisation, with step sizes in both $(x.y)$-directions given by $\delta x = \delta y = 1$, with boundary values specified on the outer closed domain of a "*toy-problem*" is shown in figure 11.1. A general discretisation stencil for the unknowns in the interior is given by

$$2(1 + [\frac{\delta y}{\delta x}]^2)U_{i,j} = [\frac{\delta y}{\delta x}]^2(U_{i+1,j} + U_{i-1,j}) + U_{i,j+1} + U_{i,j-1} \tag{11.2}$$

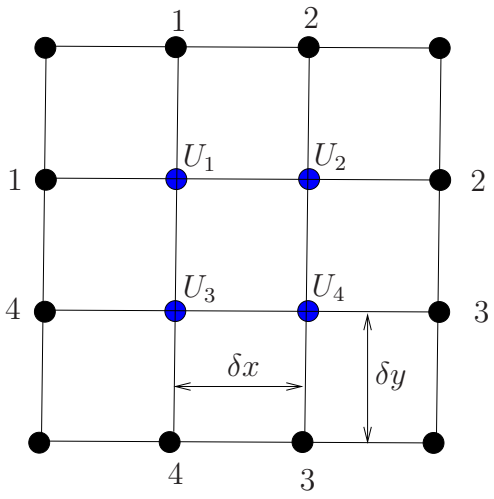

Figure 11.1: Discretised domain - Toy problem

Application of Eqn. (11.2), on each of the interior unknown points (U_1, U_2, U_3, U_4) in figure 11.1, results in a system of 4 coupled systems of equations, which may be written in matrix form as

$$[\boldsymbol{A}]\ \bar{\boldsymbol{U}} = \boldsymbol{b} \tag{11.3}$$

i.e.

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 8 \\ 6 \end{pmatrix}. \tag{11.4}$$

This may be easily solved by inverting the matrix $[\boldsymbol{A}]$, by use of Gaussian Ellimination, to give the following values :

$$U_1 = 1.833333333333333\,; \quad U_2 = 2.166666666666667\,;$$
$$U_3 = 3.166666666666667\,; \quad U_4 = 2.833333333333333\,.$$

Solution by the Gaussian elimination technique is called the **direct** approach.

11.1 Solution by Iteration

Let us try another technique, which does not require Gaussian elimination but a simple **iteration** technique.

We use the computer (or a calculator!) to solve this by an **iterative** approach, demonstrating the basic idea on a very idealised 4×4 grid with appropriate conditions enforced at the boundaries.

Four unknowns, $U_1 \;\; U_2 \;\; U_3 \;\; U_4$ which are all set, with an initial guess (or trial solution) to have values of zero. Define the next iterate by

$$\begin{aligned} 4y_1 &= U_2 + U_3 + 2 \qquad & 4y_2 &= U_1 + U_4 + 4 \\ 4y_3 &= U_1 + U_4 + 8 \qquad & 4y_4 &= U_2 + U_3 + 6 \end{aligned} \tag{11.5}$$

Then set $U_1 = y_1$ etc and repeat. The solution (corrections) proceeds as given in Table 1. Note the errors, by this so-called **Jacobi** iteration halve after each iteration.

Table 1: Solution of linear equations by Jacobi iteration

Iteration	U_1	U_2	U_3	U_4
0	0	0	0	0
1	0.5	1.0	2.0	1.5
2	1.25	1.5	2.5	2.25
3	1.5	1.875	2.875	2.5
4	1.6875	2.0	3.0	2.6875
5	1.75	2.09375	3.09375	2.75
6	1.796875	2.125	3.125	2.796875
∞	1.83333333	2.166666667	3.166666667	2.83333333

Now do the same thing but use the new iterates as soon as they become available – known as the **Gauss-Seidel (G-S)** method, *i.e.*,

$$\begin{aligned} 4U_1 &= U_2 + U_3 + 2 \qquad & 4U_2 &= U_1 + U_4 + 4 \\ 4U_3 &= U_1 + U_4 + 8 \qquad & 4U_4 &= U_2 + U_3 + 6 \end{aligned} \tag{11.6}$$

How the G-S corrections proceed is shown in Table 2.

Figure 11.2 shows the overall convergence of the solutions as the iterations proceed. Note with the G-S method, the solution has converged to an accuracy of about 7 decimal places after 14 iterations; contrast this with the Jacobi method where only an accuracy of 3 decimals is attained after about 14 iterations. Note the errors with G-S decrease by a factor of 4 after each iteration. Usually a log plot shows up convergence behaviour much better, since to graphical accuracy, looking at the first plot one would assume that both solutions were pretty similar by about 8 iterations – a log plot usually gives much clearer indications of convergence and errors.

Table 2: Solution of linear equations by Gauss-Seidel iteration

Iteration	U_1	U_2	U_3	U_4
0	0	0	0	0
1	0.5	1.125	2.125	2.3125
2	1.3125	1.90625	2.90625	2.703125
3	1.703125	2.1015625	3.1015625	2.80078125
4	1.80078125	2.150390625	3.150390625	2.825195312
5	1.825195313	2.162597656	3.162597656	2.831298828
6	1.831298828	2.165649414	3.165649414	2.832824707
∞	1.833333333	2.166666667	3.166666667	2.833333333

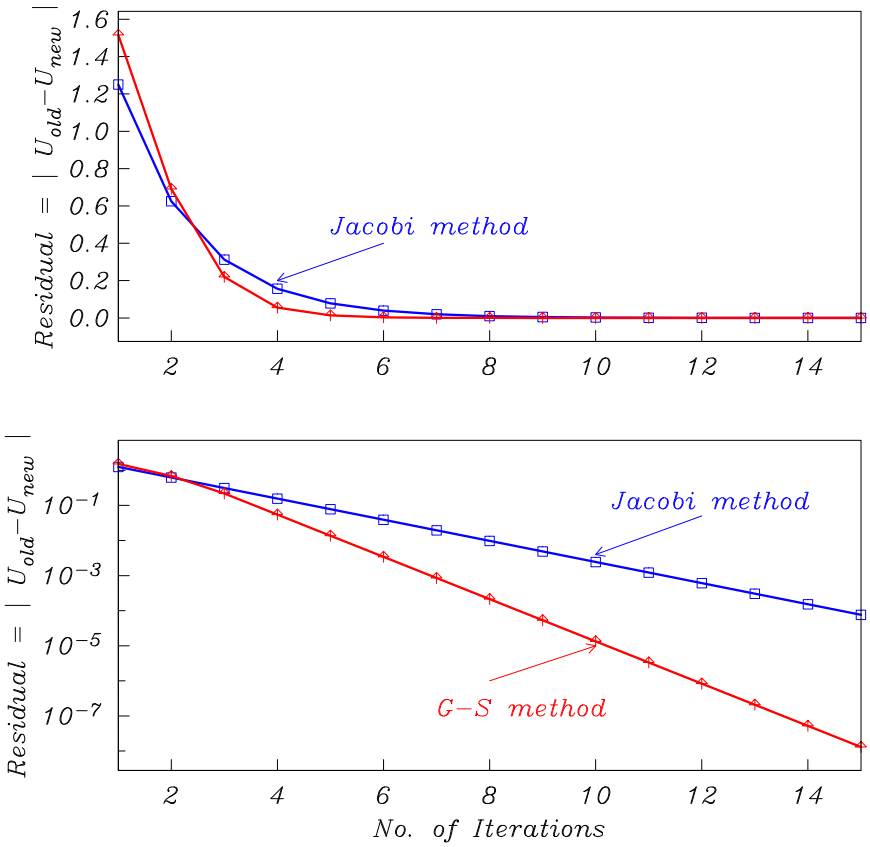

Figure 11.2: Residuals of the solution, as the iterations proceed, comparing Jacobi and Gauss-Seidel (G-S) methods.

Above we have solved a quite trivial problem and used quite a simple means to establish that our solution has converged, namely by monitoring how the value of each unknown behaves and whether they approach constant values as the iterations proceed. With just 4 unknowns this is quite adequate, however in more complex problems where we may be dealing with perhaps many hundreds, thousands or millions of unknowns, more sophisticated computer automated monitoring of the solutions and convergence criteria needs to be programmed for. Sometimes (or usually !) due to a bug in your program or for some other reason convergence to a solution does not occur, in which circumstances you may find your program to be in an endless loop – a well written computer program monitors and usually has traps programmed to terminate computations if convergence or the *residuals*

do not appear to be decreasing during the iterations.

In figure 11.2 how do you think we have defined the residuals? Are we monitoring just one of the U_i solution points, some or all of them? What would be the best means to assess a converged solution?

11.2 Direct versus iterative techniques

We have shown above that finite-difference discretisation for solving a boundary value problem (BVP) leads to a system of algebraic simultaneous equations. In most practical cases, even say when solving a nonlinear BVP, the solution discretisation procedure usually still involves solving linear systems of coupled equations. Generally their number is large possibly running into tens of thousands or even millions of unknowns, and thus their solution is a major problem – observe in our simple example above we only had 4 unknowns, see Eqn. (11.3).

We also showed that there are two different ways of solving linear systems arising from elliptic differential equations. A direct method such as Gaussian elimination produces an exact solution in a finite number of operations. Gaussian elimination solves the system of equations in a known number of arithmetic operations, and any errors in the solution arise solely from rounding errors arising from the computer.

An iterative method, starts with an initial guess for the solution and attempts to improve it through some correction procedure – the iterations are repeated and corrected values monitored to ascertain a possible convergence to an unchanging solution with increasing number of iterations. Usually the user specifies a convergence criterion to stop the iterations once a sufficiently good approximation has been obtained.

Direct inversion of the matrix $[\boldsymbol{A}]$ in Eqn. (11.3) will always work (provided $[\boldsymbol{A}]$ can be inverted), while iterative techniques are not always guaranteed to work (see later). However there are many class of problems which can be discretised where $[\boldsymbol{A}]$ has the form of a large *sparse* matrix and is *diagonally dominant*, under such conditions iterative methods are often preferable.

Finite-difference approximations often lead to sparse matrices, and, in what follows, we will concentrate on iterative techniques for solving linear systems resulting from discretisation of elliptic PDEs.

More details on the above and other aspects concerning solution of elliptic PDEs are in Chapters 3-4 of Leveque and Chapter 5 of G. D. Smith

12 Iterative Methods for Elliptic Problems

Last lecture we showed in a very simple way how the elliptic Poisson equation in the unit square, namely

$$\nabla^2 u \equiv u_{xx} + u_{yy} = f \quad \text{in } \ 0 \le x \le 1, \ \ 0 \le y \le 1, \tag{12.1}$$

with appropriate conditions on the 4-sided square boundaries (we had set $f = 0$ in our "*toy-problem*" example), could be reduced using finite-differences to requiring the solution of $M \equiv (P-2)(P-2) \equiv N \times N$ simultaneous algebraic equations (*i.e.* in the toy problem considered earlier $P = 4$, with $N = P - 2$) having the form

$$\boldsymbol{AU} = \boldsymbol{b}, \ \ \text{where } \boldsymbol{A} \text{ is an } N \times N \text{ sparse matrix}, \tag{12.2}$$

and $\boldsymbol{U}$ was an N-vector of the unknowns,

$$\boldsymbol{U} = (U_{1,1}, U_{1,2}, \ldots, U_{1,N}\,; U_{2,1}, U_{2,2}, \ldots, U_{2,N}\,; U_{N,1}, U_{N,2}, \ldots, U_{N,N}). \tag{12.3}$$

For this problem on a square grid, $\boldsymbol{A}$ can be written in block form

$$\boldsymbol{A} = \left(\begin{array}{c|c:c|c} \mathcal{T} & \mathcal{I} & \mathcal{O} & \mathcal{O} \\ \hline \mathcal{I} & \mathcal{T} & \ddots & \mathcal{O} \\ \hline \mathcal{O} & \ddots & \ddots & \mathcal{I} \\ \hline \mathcal{O} & \mathcal{O} & \mathcal{I} & \mathcal{T} \end{array}\right); \quad \mathcal{T} = \begin{pmatrix} -4 & 1 & 0 & 0 \\ 1 & -4 & \ddots & 0 \\ 0 & \ddots & \ddots & 1 \\ 0 & 0 & 1 & -4 \end{pmatrix}.$$

If you ever wish to construct such a block matrix in Matlab, the *kron* function is useful. Usually, however, we will not wish to deal with the matrices directly. The critical point is that the matrix $\boldsymbol{A}$ is **sparse** – almost all its entries are zero. Yet because of its **penta-diagonal** structure, its inverse is full. This renders direct solution methods unattractive, requiring large amounts of storage and CPU* operations. In contrast, iterative methods require very little storage or time *per iteration*. What we have to ascertain, is whether they converge, and how quickly. If we need too many iterations, there may be no gain over Gaussian elimination (or Lower-Upper (LU) decomposition) requiring $O(P^3) = O(N)^6$ operations.

An iterative scheme essentially splits the matrix $\boldsymbol{A}$ into two components, $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{C}$, where $\boldsymbol{B}$ is chosen to be easy to invert. It then solves

$$\boldsymbol{BU}^{j+1} = \boldsymbol{b} - C\boldsymbol{U}^j \ \ \text{or} \ \ \boldsymbol{U}^{j+1} = \boldsymbol{B}^{-1}\boldsymbol{b} - (\boldsymbol{B}^{-1}C)\boldsymbol{U}^j. \tag{12.4}$$

If we introduce the error vector $\boldsymbol{Z}^j = \boldsymbol{U}^j - \mathcal{U}$, where $\mathcal{U}$ is the exact solution to (12.2) then

$$\boldsymbol{Z}^{j+1} = (\boldsymbol{B}^{-1}C)\boldsymbol{Z}^j. \tag{12.5}$$

In this form it is clear that fastest convergence occurs for small **spectral** radius $\rho(\boldsymbol{B}^{-1}C)$.

*central processing unit

In the Jacobi-method we compute new values for U entirely from the previous iteration, as required from the equation. In the Gauss-Seidel-method we have already updated U_{mn-1} and U_{m-1n} before we update U_{mn} ; and these new values will be used instead of the old ones. The nice feature of both these iterative approaches is

- the matrix $\boldsymbol{A}$ is never stored, as would be the case with a direct method involving Gaussian elimination.
- Storage is optimal. Only N^2 values are stored for the Gauss-Seidel method, while $2N^2$ values are stored for the Jacobi method.
- Each iteration requires N^2 work. The total work depends on the number of iterations necessary to reach a desired level of accuracy.
- With direct methods, typically Gaussian elimination, it's running time is of $O(N^3)$, thus for very large N values iterative approaches are usually preferred – unless the form of the matrix $\boldsymbol{A}$ is such that the iterative approach is known to fail or is *ill-conditioned* – namely where the spectral radius $\rho(\boldsymbol{T}) \geq 1$ and strict diagonal dominance does not arise in the matrix.

Though we do not prove it in this course, the convergence of a linear iterative system is governed by the spectral radius or largest eigenvalue (in modulus) of the coefficient matrix. Thus, a fundamental stability criterion is that *a linear iterative system is asymptotically stable and convergent if and only if all its (complex) eigenvalues (λ) have modulus strictly less than one:* $|\lambda_j| < 1$. The spectral radius of a matrix $\boldsymbol{T}$ is defined as the maximal modulus of all of its real and complex eigenvalues: $\rho(\boldsymbol{A}) = max\{|\lambda_1|, \ldots, |\lambda_k|\}$. Unfortunately, finding accurate approximations to the eigenvalues of most matrices is a nontrivial computational task.

The spectral radius $\rho(\boldsymbol{T})$ of the coefficient matrix will govern the speed of convergence. Therefore, our main goal is to construct an iterative scheme whose coefficient matrix has as small a spectral radius as possible. At the very least, the spectral radius must be less than 1.

Completely general conditions guaranteeing convergence of the Gauss-Seidel and Jacobi methods are hard to establish. But both are guaranteed to converge when the original coefficient matrix $\boldsymbol{A}$ is strictly diagonally dominant, i.e.

$$|a_{i,1}|+|a_{i,2}|+\ldots+|a_{i,i-1}|+0+\ |a_{i,i+1}|+|a_{i,i+2}|+\ldots+|a_{i,m}| \leq\ |a_{i,i}|. \tag{12.6}$$

This states, that convergence is assured if in each row of $\boldsymbol{A}$ the modulus of the diagonal element exceeds the sum of the moduli of the off-diagonal elements.

12.1 Technical note on convergence

Consider the matrix

$$\boldsymbol{AU} = \boldsymbol{b}. \tag{12.7}$$

We next express that $\boldsymbol{A} = \boldsymbol{D} + \boldsymbol{C}$, where $\boldsymbol{D}$ represents the main diagonal elements and $\boldsymbol{C}$ the *off*-diagonal elements. We further split $\boldsymbol{C} = \boldsymbol{L} + \boldsymbol{V}$, the strictly lower triangular and strictly upper triangular elements, namely

$$\boldsymbol{D} = \begin{pmatrix} a_{11} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & a_{nn} \end{pmatrix};$$

$$\boldsymbol{V} = \begin{pmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & 0 & a_{23} & \dots & a_{2n} \\ 0 & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & 0 & a_{n-1\,n} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}; \quad \boldsymbol{L} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 & 0 \\ a_{31} & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & 0 & 0 \\ a_{n1} & a_{n2} & \dots & a_{n\,n-1} & 0 \end{pmatrix}.$$

Eqn. 12.7 can thus be written as

$$\boldsymbol{DU} = \boldsymbol{b} - \boldsymbol{CU} = \boldsymbol{b} - (\boldsymbol{L} + \boldsymbol{V})\boldsymbol{U} \tag{12.8}$$

The simplest iterative scheme essentially takes $\boldsymbol{B} = \boldsymbol{D}$ (see Eqn. 12.4), where $\boldsymbol{D}$ is the diagonal part of $\boldsymbol{A}$ and $\boldsymbol{C} = \boldsymbol{L} + \boldsymbol{V}$ has zeros on its diagonal. We then defined the iterative **Jacobi** scheme

$$\boldsymbol{DU}^{j+1} = \boldsymbol{b} - C\boldsymbol{U}^{j}$$

giving

$$\boldsymbol{U}^{j+1} = \boldsymbol{D}^{-1}(\boldsymbol{b} - C\boldsymbol{U}^{j}) \tag{12.9}$$

or

$$U_{mn}^{j+1} = \frac{1}{4}\left[U_{mn+1}^{j} + U_{mn-1}^{j} + U_{m+1n}^{j} + U_{m-1n}^{j} - h^2 f_{mn}\right].$$

This calculates all the new iterations at level $j+1$ before updating U. The matrix $\boldsymbol{D}^{-1}\boldsymbol{C}$ is called the *point Jacobi iteration matrix*.

We also considered the **Gauss-Seidel** (G-S) scheme, which uses the new calculated values as soon as they become available. Assuming m and n are scanned in an increasing direction, this scheme can be written as

$$U_{mn}^{j+1} = \frac{1}{4}\left[U_{mn+1}^{j} + U_{mn-1}^{j+1} + U_{m+1n}^{j} + U_{m-1n}^{j+1} - h^2 f_{mn}\right]. \tag{12.10}$$

This essentially involves back-substitution of the calculated values. Consider Eqn. 12.8, the G-S iteration is defined by

$$\boldsymbol{DU}^{j+1} = \boldsymbol{b} - \boldsymbol{LU}^{j+1} - \boldsymbol{VU}^{j}, \tag{12.11}$$

hence

$$(\boldsymbol{D} + \boldsymbol{L})\boldsymbol{U}^{j+1} = \boldsymbol{b} - \boldsymbol{VU}^{j}, \tag{12.12}$$

giving

$$\boldsymbol{U}^{j+1} = (\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{b} - (\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{VU}^{j}. \tag{12.13}$$

Here $(\boldsymbol{D} + \boldsymbol{L})^{-1}\boldsymbol{V}$ is called the *Gauss-Seidel iteration matrix*.

A necessary and sufficient condition for convergence

The iterative methods described generally have the form

$$\boldsymbol{x}^{j+1} = \boldsymbol{G}\boldsymbol{x}^{j} + \boldsymbol{e}, \tag{12.14}$$

where $\boldsymbol{G}$ is the *iteration matrix*. This was derived from the original discretised system by re-arranging them into the form

$$\boldsymbol{x} = \boldsymbol{G}\boldsymbol{x} + \boldsymbol{c}, \tag{12.15}$$

with $\boldsymbol{c}$ a column vector of known values. Now we know that a solution to an equation of form $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is the solution of Eqn. 12.15. The error to the exact solution is defined by

$$\boldsymbol{e}^{j+1} = \boldsymbol{x} - \boldsymbol{x}^{j}$$

in the j-th approximation, so by subtracting the above two equations, it follows that

$$\boldsymbol{e}^{j+1} = \boldsymbol{G}\boldsymbol{e}^{j}. \tag{12.16}$$

Hence, it follows

$$\boldsymbol{e}^{j} = \boldsymbol{G}\boldsymbol{e}^{j-1} = \boldsymbol{G}^{2}\boldsymbol{e}^{j-2} = \ldots = \boldsymbol{G}^{m}\boldsymbol{e}^{0},$$

and hence the sequence of iterative values $\boldsymbol{x}^{1}, \boldsymbol{x}^{2}, \ldots, \boldsymbol{x}^{j}, \ldots$ will converge to $\boldsymbol{x}$ as $j \to \infty$ if

$$\lim_{j\to\infty} \boldsymbol{e}^{j} = 0.$$

Since $\boldsymbol{x}^{0}$ and $\boldsymbol{e}^{0}$ are arbitrary, it follows that the iterations will converge if and only if

$$\lim_{j\to\infty} \boldsymbol{G}^{j} = 0.$$

It thus follows the matrix $\boldsymbol{G}$ converges for all or any initial guesses.

If a matrix $\boldsymbol{G}$ of order m has m linearly independent eigenvectors $\boldsymbol{v}_k,\ k = 1, \ldots, m$, then the arbitrary error vector $\boldsymbol{e}^{0}$ can be expressed uniquely as a linear combination, namely

$$\boldsymbol{e}^{0} = \sum_{k=1}^{m} c_k \boldsymbol{v}_k, \tag{12.17}$$

where c_k are scalars. Hence

$$\boldsymbol{e}^{1} = \boldsymbol{G}\boldsymbol{e}^{0} = \sum_{k=1}^{m} c_k \boldsymbol{G}\boldsymbol{v}_k,$$

but $\boldsymbol{G}\boldsymbol{v}_k = \lambda_k \boldsymbol{v}_k$ by definition of an eigenvalue, where λ_k is the eigenvalue associated with the eigenvectors $\boldsymbol{v}_k$. Hence

$$\boldsymbol{e}^{1} = \sum_{k=1}^{m} c_k \lambda_k \boldsymbol{v}_k,$$

and similarly

$$\boldsymbol{e}^{j} = \sum_{k=1}^{m} c_k \lambda_k^{j} \boldsymbol{v}_k.$$

Therefore $\boldsymbol{e}^{j}$ will tend to the zero vector as $j \to \infty$ for arbitrary $\boldsymbol{e}^{0}$, if

$$|\lambda_k| < 1$$

for all k, *i.e.* the iterations will converge for arbitrary $\boldsymbol{x}^{0}$ if the spectral radius $\rho(\boldsymbol{G})$ of $\boldsymbol{G}$ is less than one, or in other words "the error is forced to zero as the iteration proceeds".

Rate of convergence

Assume that the iteration matrix $\boldsymbol{G}$ has m linearly independent eigenvectors and eigenvalues λ_k and that

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \ldots > \lambda_m$$

The error vector[†] $\boldsymbol{e}^{(j)}$ can be expressed as

$$\boldsymbol{e}^{(j)} = \lambda_1^j \left(c_1 \boldsymbol{v_1} + (\frac{\lambda_2}{\lambda_1})^j c_2 \boldsymbol{v_2} + \ldots + (\frac{\lambda_m}{\lambda_1})^j c_m \boldsymbol{v_m} \right).$$

For large j this shows that

$$\boldsymbol{e}^{(j)} = \lambda_1^j c_1 \boldsymbol{v_1};$$

similarly

$$\boldsymbol{e}^{(j+1)} = \lambda_1^{j+1} c_1 \boldsymbol{v_1},$$

so

$$\boldsymbol{e}^{(j+1)} = \lambda_1 \boldsymbol{e}^{(j)}.$$

If the $i-$th component of $\boldsymbol{e}^{(j)}$ is denoted by $e_i^{(j)}$ it is seen that

$$\frac{|e_i^{(j)}|}{|e_i^{(j+1)}|} = \frac{1}{\lambda_1} = \frac{1}{\rho(\boldsymbol{G})}.$$

We next show, that the spectral radius $\rho(\boldsymbol{G})$, roughly provides the worst factor by which the error is reduced with each iteration, and also can be used to provide an estimate of number of iterations required to reduce the error by a factor of 10^{-p}.

For large j,

$$\boldsymbol{e}^{(j)} \approx \lambda_1 \boldsymbol{e}^{(j-1)},$$

therefore

$$\boldsymbol{e}^{(j+\alpha)} \approx \lambda_1 \boldsymbol{e}^{(j+\alpha-1)} \approx \ldots \approx \lambda_1^\alpha \boldsymbol{e}^{(j)}, \quad \alpha = 1, 2, \ldots$$

Suppose we define α as the smallest integer that satisfies

$$\frac{|\boldsymbol{e}^{(\alpha)}|}{|\boldsymbol{e}^{(0)}|} \leq 10^{-p}.$$

It follows, this will be approximately satisfied if

$$[\rho(\boldsymbol{G})]^\alpha \leq 10^{-p}, \text{ hence } \alpha \geq -\frac{p}{\log_{10}[\rho(\boldsymbol{G})]}.$$

The part $-\log_{10}[\rho(\boldsymbol{G})]$ is sometime called the "convergence rate", thus its inverse gives an approximation of number of iterations required to reduce the error by one decimal digit (10^{-1}). We also see that the convergence rate decreases as $\rho(\boldsymbol{G}) \to 1$, while smaller $\rho(\boldsymbol{G})$ values give the fastest convergence rate.

As an example, take $\boldsymbol{e}_i^{(j)} = 10^{-2}$ and $\boldsymbol{e}_i^{(j+1)} = 10^{-4}$. Then $\boldsymbol{e}_i^{(j)}/\boldsymbol{e}_i^{(j+1)} = 10^2$, and the logarithm in base 10 is 2. Hence, since

$$\log_{10}(1/p) = -\log p$$

[†] slight change in notation by use of the () brackets!

is an indication of number of decimal digits by which the error decreases per iteration. If we want to reduce the size of the error by 10^{-p}, say, then the number of iterations required to achieve this will be the least value of α for which

$$|\lambda_1^\alpha| = \rho^\alpha \leq 10^{-p}.$$

Taking logs and recalling that $\log \rho$ is negative for a convergent iteration leads to

$$\alpha \geq \frac{p}{-\log_{10} \rho}.$$

In terms of the matrix, $\boldsymbol{B} = \boldsymbol{D} + \boldsymbol{L}$ where $\boldsymbol{L}$ is the lower triangular part of $\boldsymbol{A}$. The spectral radii of the Jacobi and Gauss-Seidel matrices can be found. In fact, for large $M \equiv (N-1)^2$ it can be shown that

$$\rho_J = \cos\left(\frac{\pi}{N}\right) \approx 1 - \frac{\pi^2}{2N^2}, \quad \rho_G = \rho_J^2 \approx 1 - \frac{\pi^2}{N^2} \tag{12.18}$$

This agrees with the factor of 4 error reduction we found using the toy-matrix problem used. We can therefore estimate how many iterations are needed to obtain p figures of accuracy. For the error to be reduced by a factor of 10^p . If $\rho^\alpha = 10^{-p}$ then

$$\alpha = \frac{p \log 10}{-\log \rho} \approx 0.467 p N^2 \text{ or } 0.233 p N^2 \tag{12.19}$$

for Jacobi or Gauss-Seidel. This is disappointingly large for large N, but faster than direct methods. Can we do better somehow?

12.2 Successive Over-Relaxation

Consider Eqn. 12.13, we may represent the scheme in terms of the residual vector

$$\boldsymbol{r}^j = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{U}^j,$$

which is the amount by which the j-th estimate fails to satisfy the equation. We then have that

$$\boldsymbol{U}^{j+1} = \boldsymbol{U}^j + \boldsymbol{B}^{-1}\boldsymbol{r}^j \text{ or } \boldsymbol{r}^{j+1} = C\boldsymbol{B}^{-1}\boldsymbol{r}^j. \tag{12.20}$$

If we assume that we are altering the estimate in a good direction, we might try to take larger steps, and choose a parameter $\omega > 1$ and define the **over-relaxation** scheme

$$\boldsymbol{U}^{j+1} = \boldsymbol{U}^j + \omega\boldsymbol{B}^{-1}\boldsymbol{r}^j. \tag{12.21}$$

In terms of the code we modify (12.10) to read

$$U_{mn}^{j+1} = (1-\omega)U_{mn}^j + \frac{\omega}{4}\left[U_{mn+1}^j + U_{mn-1}^{j+1} + U_{m+1n}^j + U_{m-1n}^{j+1} - h^2 f_{mn}\right]. \tag{12.22}$$

We can then investigate which values of ω give best behaviour. This is a very powerful idea in general. Even if we start with an unstable scheme, it may become stable by choosing a small value of ω. For a stable scheme, choosing $\omega > 1$ probably will speed up the convergence.

We have already alluded above that the rate of convergence of the methods is related to the spectral radius – ideally we want to choose an optimum value of $\omega = \omega_o$ that minimises the spectral radius of the iteration matrix. A general formula for an arbitrary set of linear equations is difficult to find, The following result, which we do not prove (see Varga[‡]) for the Jacobi iteration matrix, can be shown to be

$$\omega_o = \frac{2}{1 + \sqrt{1 - \rho^2(\boldsymbol{B})}},$$

where $\rho(\boldsymbol{B})$ is the spectral radius of the Jacobi iteration matrix – here the matrix $\boldsymbol{A}$ is real and symmetric and has **block** tridiagonal form, with non-zero off diagonal elements of the Jacobi iteration matrix $\boldsymbol{B}$.

In general the optimum ω varies with N. But the optimal ω brings a very marked improvement to $\rho \approx 1 - 2\pi/N$ and $\alpha \approx 0.37pN$. The **Successive Over-Relaxation** (**SOR**) method (12.22) requires $O(N)$ iterations not $O(N^2)$.

Results for the toy-problem with just $N = 4$ (see Eqn. 11.3) are shown in Fig. 12.1. Observe the G-S method halves the number of iterations required, to achieve a near machine precision residual $\boldsymbol{r}^j = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{U}^j$ value. This is in almost perfect agreement with the theoretical result given by expressions (12.19). The SOR method gives still better convergence (nearly halving the iteration count again), after having found the optimal $\omega_o \approx 1.07$.

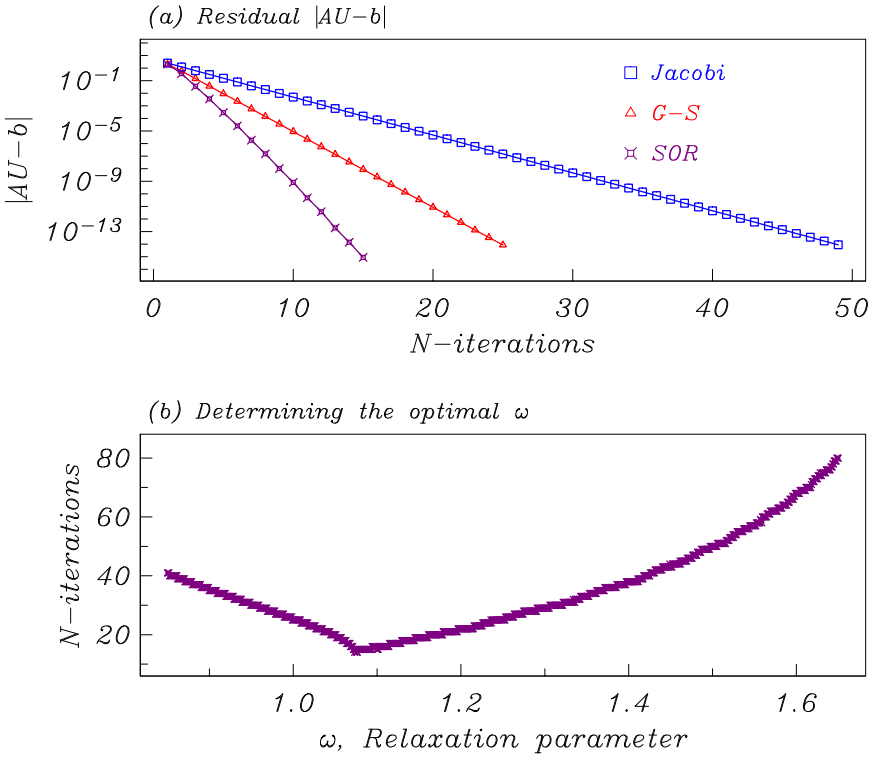

Figure 12.1: (a) Iterations required by the Jacobi, Gauss-Seidel and SOR methods. (b) Variation of the iteration count on varying the SOR relaxation parameter ω in Eqn. 12.22. The SOR computation in the upper figure is by setting the optimal $\omega_o \approx 1.07$ identified from plot (b).

[‡]Varga, R. S. (1992) *Matrix Iterative Analysis*, Prentice-Hall, New-Jersey

13 Introduction to Multigrid Methods

Last lecture we investigated the behaviour of our 3 iterative methods on different grids and with different error wavenumbers. Essentially we found:

1. Jacobi and Gauss-Seidel quickly smoothed out the high wavenumbers (oscillations on a small scale). However, they were very slow to iron out the low wavenumbers (long waves). After a few iterations the error and residuals were dominated by these low wavenumbers. Thus even if the solution to the entire problem varies on a short length-scale, the errors after a few iterations vary on a large scale.

2. These low wavenumber modes do not require a lot of points to resolve them well. Furthermore, on coarser grids they would be damped more quickly. This suggests that it might be worth swapping between more than one grid. This is the idea behind **multigrid methods.**

3. The over-relaxation methods (SOR), although their overall convergence was much faster than Jacobi/GS, did not smooth the solution at all well. At all stages, the error vector, though small in magnitude, consisted of a mixture of wavenumbers. This does not lend itself well to the idea of swapping between grids, and so surprisingly, we will use our 2nd best method (Gauss-Seidel) on the various grids.

Two grid strategy
Suppose we have a course grid C and a fine grid F containing twice as many points in each direction. We wish to solve $A_F\mathbf{u}_F = \mathbf{b}_F$ on the fine grid. The corresponding approximation on C is $A_c\mathbf{u}_c = \mathbf{b}_c$.

(a) We begin with a few Gauss-Seidel sweeps on F to get rid of the short waves, obtaining an approximation $\mathbf{u}_F$.

(b) Next we calculate the residuals $\mathbf{r}_F = A\mathbf{u}_F - \mathbf{b}$.

(c) Now we transfer to the coarser grid C using a restriction operator $\mathcal{R} : \mathrm{F} \to \mathrm{C}$, to find $\mathbf{r}_c = \mathcal{R}\mathbf{r}_F$.

(d) On the coarser grid we calculate the error $\mathbf{z_c}$ quickly, where $A_c\mathbf{z_c} = \mathbf{r_c}$.

(e) We now use an interpolation operator P to map the error $\mathbf{z_c}$ to $\mathbf{z_F} = \mathcal{P}\mathbf{z_c}$.

(f) We then modify our fine estimate by the calculated error, so that $\mathbf{u}_F + \mathbf{z_F}$ is the new solution estimate.

(g) We then take a few more sweeps using Gauss-Seidel, in case any short wave errors have crept in. This completes a cycle. If we wish we can repeat the process using our new estimate.

Before we can implement this scheme, we have to decide how to **interpolate** from a coarse mesh onto a finer one, and also, how best to **restrict** our solution on a finer mesh to a less

accurate coarse mesh. If we do this linearly, we can represent these transformations by rectangular matrices, $\mathcal{P}$ and $\mathcal{R}$. Let these have dimensions $\mu \times \nu$ and $\nu \times \mu$ respectively. As each grid point on the coarse mesh is also a point on the fine mesh, the simplest idea would be to use the same value at the new point. But it might be a good idea to use a linear combination of all the neighbouring points. We want to minimize the errors introduced by transfer between grids. Ideally, we would like $\mathcal{PR}$ and $\mathcal{RP}$ to be identity matrices but that would overconstrain the problem. It is a good idea to require the interpolation operator $\mathcal{P}$ and our restriction operator $\mathcal{R}$ to be adjoints with respect to the scalar product, $\{u, v\}$. If we denote the coarse and fine meshes by C and F, and let u_c and v_F be any vectors in the respective meshes. Then

$$\{\mathbf{u}_c, \mathcal{R}\mathbf{v}_F\} = \{\mathcal{P}\mathbf{u}_c, \mathbf{v}_F\} \quad \Longrightarrow \quad \mathcal{R} = P^T/4, \tag{13.1}$$

allowing for a factor of 4 from the halving of the steplengths (the vector $\mathbf{v}_F$ is 4 times the length of $\mathbf{u}_c$.) Note effectively we are replacing the matrix A_c by $\mathcal{R}\mathsf{A}_F\mathcal{P}$ – this is known as the Galerkin condition.

The interpolation or prolongation operator shown in Figure (13.1a) is easiest to choose, as we have less information to work with

$$\begin{aligned} u_F(2i, 2j) &= u_c(i, j) \\ u_F(2i+1, 2j) &= \tfrac{1}{2}(u_c(i, j) + u_c(i+1, j)) \\ u_F(2i, 2j+1) &= \tfrac{1}{2}(u_c(i, j) + u_c(i, j+1)) \\ u_F(2i+1, 2j+1) &= \tfrac{1}{4}(u_c(i, j) + u_c(i+1, j) + u_c(i, j+1) + u_c(i+1, j+1)) \end{aligned} \quad . \tag{13.2}$$

Using (13.1), the appropriate restriction operator, Figure (13.1b), is then $\mathbf{u}_c = \mathcal{R}\mathbf{u}_F$ where

$$\begin{aligned} u_c(i, j) = &\tfrac{1}{4}u_F(2i, 2j) + \ldots \\ &+ \tfrac{1}{8}(u_F(2i+1, 2j) + u_F(2i, 2j+1) + u_F(2i-1, 2j) + u_F(2i, 2j-1)) + \ldots \\ &+ \tfrac{1}{16}(u_F(2i+1, 2j+1) + u_F(2i-1, 2j+1) + u_F(2i-1, 2j-1) + u_F(2i+1, 2j-1)) \end{aligned} \quad . \tag{13.3}$$

This keeps the errors under control when flipping between grids.

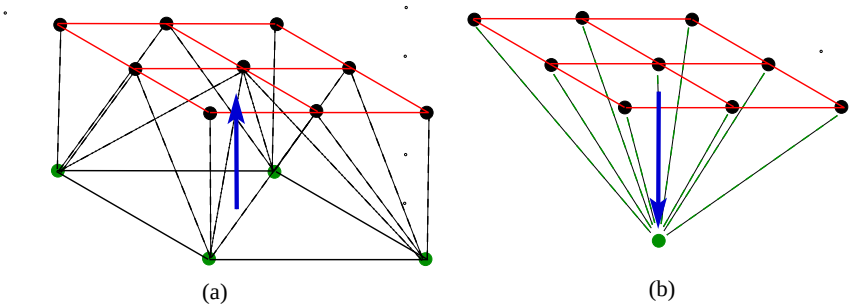

Figure 13.1: (a) Prolongation operation restoring values on a fine grid from a coarse grid. (b) Restriction operation based on full-weighting of the fine-grid values to produce a coarse grid.

13.1 More than 2 grids

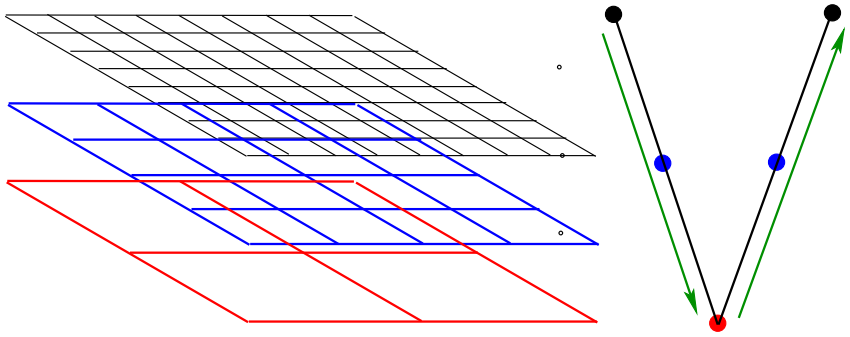

Figure 13.2: Grid hierarchy and traversal order for a V-cycle multigrid method.

The coarser grid has twice the steplength, but it may still be slow to solve for the errors as in step (d) above. But there is nothing to prevent us from performing the same algorithm again, transferring oo a still coarser grid, If our initial N is of the form $2^p + 1$, we may transfer all the way down to $p = 1$ and then move up the chain back to the finest grid.

The program MultigridV.m performs a cycle transferring down the grids to solve for the error on a coarse grid, and then interpolating the **error** back up to the finest grid. This is known as a 'V-cycle'. It is extremely efficient, but can still be improved upon!

13.2 Full multigrid

The Multigrid algorithm starts on the fine grid with an arbitrary initial guess. Instead, we could solve the problem on a coarse grid and interpolate down to give a better initial estimate, and hence reduce the iterations required. The program FullMG.m starts on the coarsest grid and interpolates to the next grid. There it solves the problem with a 2 grid algorithm, and then interpolates again, each time using a V-cycle down to the coarsest grid. Not only is it faster, but full MG also obtains the solution on every grid as opposed to the final grid only.

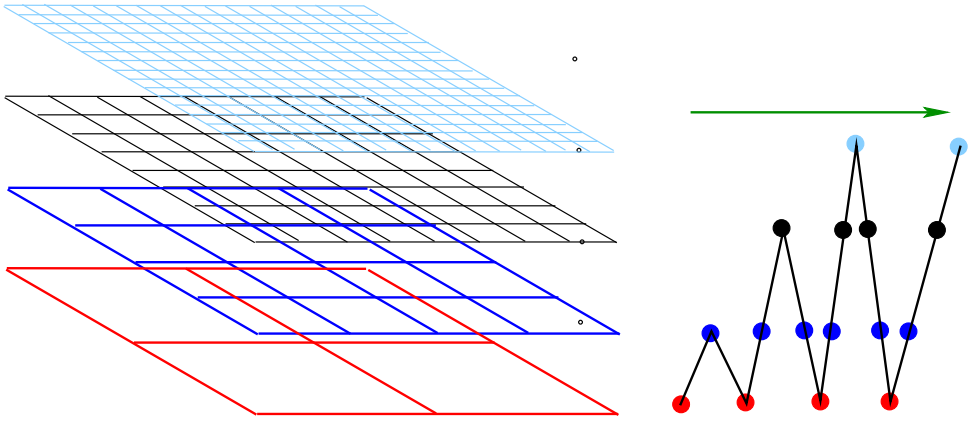

Figure 13.3: Grid hierarchy and traversal order for a full multigrid method.

14 Grids and Clustering

So far we have been discussing solving PDEs using fixed step sizes in the discretisation process. However some problems may well be on non-uniform boundaries, such as shown in left hand plots in Fig. 14.1. Defining a uniform grid in such situations clearly is not possible. Under such scenarios a coordinate mapping is usually undertaken to transform the physical domain of interest into a more tractable computational domain, on which the numerical discretisation can be performed with ease. Thus a general transformation (map-

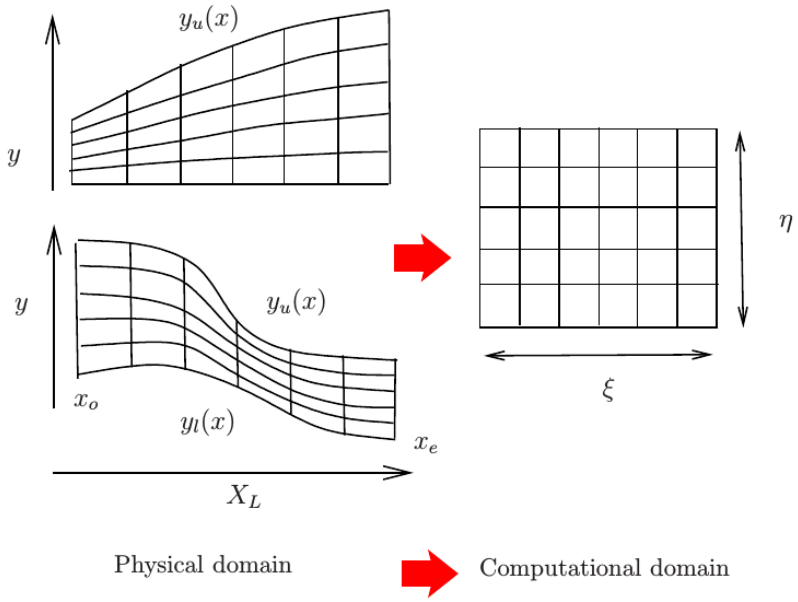

Figure 14.1: Mapping from physical plane to computational grid.

ping) of the physical coordinates (x, y) to computational plane (ξ, η), is one technique to ease the solution process. In Fig. 14.1, one possible mapping could be

$$\xi = \frac{x - x_o}{x_e - x_o}, \quad \eta = \frac{y - y_l(x)}{y_u(x) - y_l(x)} \tag{14.1}$$

thus mapping $x_o \leq x \leq x_u$ to $0 \leq \xi \leq 1$, and $y_l(x) \leq y \leq y_u(x)$ to $0 \leq \eta \leq 1$. The numerical operations and solution process is thus undertaken in the (ξ, η)-coordinates, on transforming the PDEs derivatives from the (x, y)-coordinates into the (ξ, η)-coordinates. Thus first derivatives will be transformed as follows:

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}; \quad \frac{\partial}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} \tag{14.2}$$

and the second and mixed derivatives as follows:

$$\frac{\partial^2}{\partial x^2} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right) \tag{14.3}$$

$$\frac{\partial^2}{\partial y^2} = \left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right), \tag{14.4}$$

$$\frac{\partial^2}{\partial x \partial y} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right) \tag{14.5}$$

As can be seen from Fig. 14.1, in the transformed grid, one can then make use of fixed discretisation step sizes $(\Delta\xi, \Delta\eta)$ in the transformed PDEs.

A difficulty above, in the more general case where $\xi = \xi(x, y)$ together with $\eta = \eta(x, y)$, is the evaluation of the $\xi_x,\ \eta_y$ etc. metric terms due to the **now variable** $(x,\ y)$ grid spacings. This is circumvented by observing, that if we take partial derivatives of $\xi(x, y)$ with respect to ξ and η we get

$$\begin{aligned} \xi_x\, x_\xi + \xi_y\, y_\xi &= 1 \\ \xi_x\, x_\eta + \xi_y\, y_\eta &= 0. \end{aligned} \tag{14.6}$$

If we then do the same with $\eta(x, y)$ with respect to ξ and η we get

$$\begin{aligned} \eta_x\, x_\xi + \eta_y\, y_\xi &= 0 \\ \eta_x\, x_\eta + \eta_y\, y_\eta &= 1. \end{aligned} \tag{14.7}$$

Hence the following results arise

$$\begin{aligned} \xi_x &= y_\eta/J \\ \xi_y &= -x_\eta/J \\ \eta_x &= -y_\xi/J \\ \eta_y &= x_\xi/J. \end{aligned} \tag{14.8}$$

Here J is the Jacobian given by

$$J = x_\xi y_\eta - x_\eta y_\xi. \tag{14.9}$$

The higher derivative terms such as $\xi_{xx},\ \xi_{xy},\ \eta_{yy}$ etc. follow by further differentiation and manipulation of expressions (14.6)–(14.9).

14.1 Grid Stretching, an ODE example

We illustrate the need for grid stretching by considering the ordinary differential equation:

$$\epsilon \frac{d^2u}{dx^2} - \frac{du}{dx} = f(x), \text{ satisfying } u(0) = \alpha, \quad u(1) = \beta \tag{14.10}$$

where ϵ is some parameter, and we set $\alpha = 1, \beta = 3$. Observe that as $\epsilon \to 0$, the equation reduces to $-u' = f(x)$, and thus both boundary conditions can not be satisfied - a more interesting situation develops when we consider ϵ very small but not identically equal to zero. An exact solution to this equation is given by

$$u(x) = \alpha + x + (\beta - \alpha - 1)\left(\frac{e^{x/\epsilon} - 1}{e^{1/\epsilon} - 1}\right) \tag{14.11}$$

and solutions for varying ϵ with $f(x) = -1$ are shown in Fig. 14.2. Observe that as $\epsilon \to 0, u(x)$ at the $x = 1$ boundary develops a very localised and rapidly changing solution. This region is known as a boundary-layer with thickness of $O(\epsilon)$ (see LeVeque, §2.17).

We next attempt a numerical solution with finite differences (second-order accurate), for the case $\epsilon = 0.01$, and use 11 equi-spaced grid points to discretise Eqn. 14.10. The solution is shown in Fig. 14.3 and we also compare the numerical result with the exact result (given by the blue curve). We see some significant differences between the two results, with the numerical result also displaying "wiggles". On using successively finer grids, the numerical solution eventually agrees with the exact result, as shown in Figs. 14.4 & 14.5.

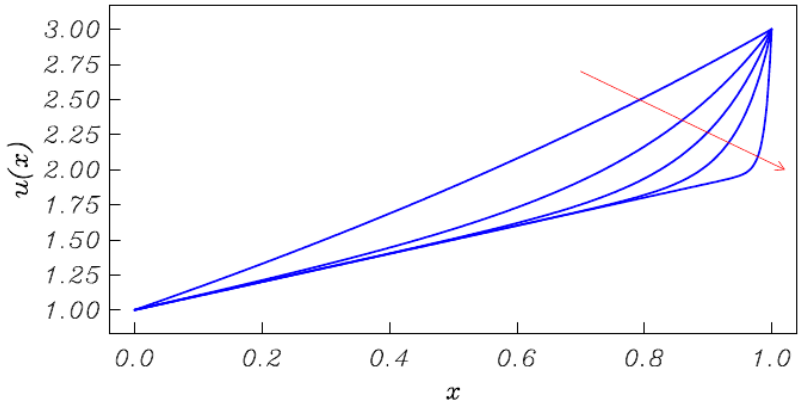

Figure 14.2: Exact solutions of Eqn. 14.10, with $\epsilon = (1.0, 0.2, 0.1, 0.05, 0.01)$, arrow in direction of decreasing ϵ

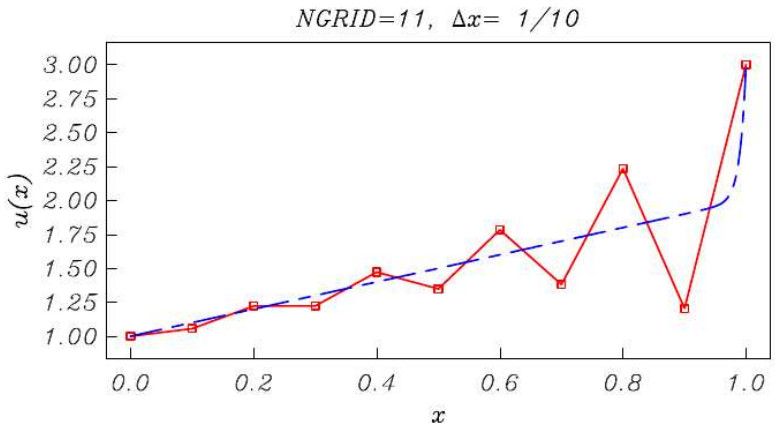

Figure 14.3: Numerical solution of Eqn. 14.10, with $\epsilon = 0.01$, with $N = 11$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

The above results thus indicate that to obtain a reasonably accurate result we need to have a number of points within the "boundary-layer" region. In the final result shown in Fig. 14.3, note that for a large part of the solution, *i.e.* $x < 0.9$ a nearly linear behaviour arises in the solution, so in this region, ideally we should require less density of grid points to resolve the solution structure there, whereas in the "boundary-layer" region $x \sim> 0.95$ it is clear many more points are needed to resolve the solution there if we were to use a uniform grid.

In more practical larger-scale problems (such as in two- and three-dimensions) exhibiting such behaviour, it would clearly be nice if one could some how cluster, or have more points only in regions where rapid variations of the solution are known to arise - thus reducing the number of grid points overall, and thus speeding up (hopefully) the solution process. This is the idea behind "grid-stretching" (or mapping function). We discuss this aspect in the context of a 1-dimensional problem, *i.e.* Eqn. 14.10, but the basic ideas will be applicable to more dimensions, albeit, using slightly more complicated expressions of the so-called mapping functions. This is best illustrated through an example.

We start with a uniform grid in our "computational" Z-frame, and then use a grid mapping function to define the associated grid points in the physical x-reference frame. There are many means of defining such functions (see LeVeque), but we choose to use the following

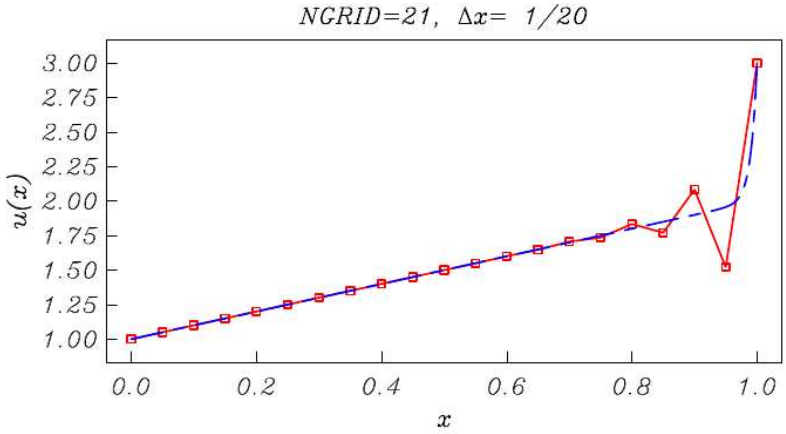

Figure 14.4: Numerical solution of Eqn. 14.10, with $\epsilon = 0.01$, with $N = 21$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

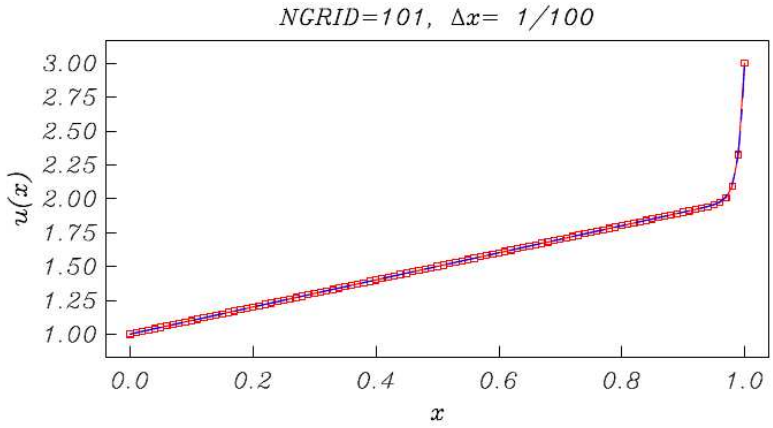

Figure 14.5: Numerical solution of Eqn. 14.10, with $\epsilon = 0.01$, with $N = 101$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

map:

$$x = c_2 + \frac{1}{c_1}\tan\left(\lambda\left[Z - \eta_o\right]\right) \tag{14.12}$$

where

$$\eta_o = \frac{z-1}{z+1}, \quad z = \frac{\tan^{-1}\left(c_1\left(1+c_2\right)\right)}{\tan^{-1}\left(c_1\left(1-c_2\right)\right)}, \quad \lambda = \frac{\tan^{-1}\left[c_1\left(1-c_2\right)\right]}{1-\eta_o}. \tag{14.13}$$

This maps $-1 \le Z \le 1$ to $0 \le x \le 1$. Some examples are shown in Figs. 14.6–14.8; here we have applied a further linear uniform mapping whereby

$$\xi = (Z+1)/2 \tag{14.14}$$

to thus map $-1 \le Z \le 1$ to $0 \le \xi \le 1$.

Hence it is easy to see, for example how such a mapping (still quite simple, and also not very flexible) could be applied to a 2D problem, as indicated in Fig. 14.9.

In all of the above, again the PDEs in the physical plane require to be transformed into the computational coordinates followed by appropriate discretisations undertaken in the computational coordinates. In this case, Eqns. 14.2-14.5 thus require to be utilised and operated upon using expressions such as (14.12) and (14.14).

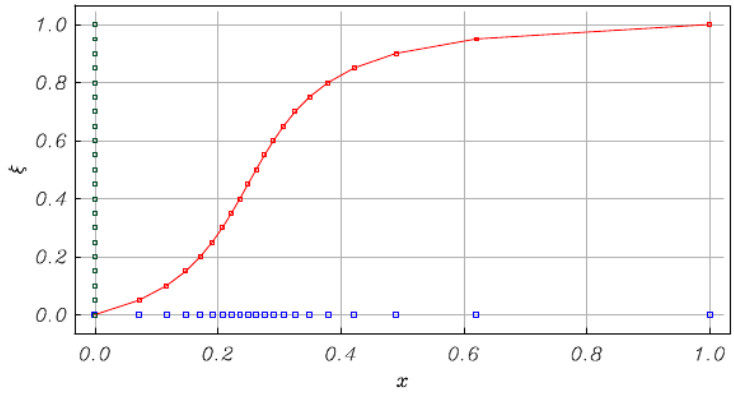

Figure 14.6: Grid mapping with $c_1 = -5$, and $c_2 = 0.25$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 0.25$

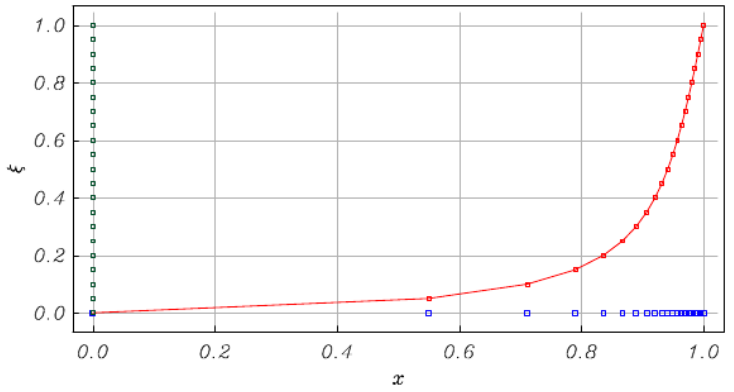

Figure 14.7: Grid mapping with $c_1 = -8$, and $c_2 = 1$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 1.0$. In this plot the mapping and clustering is quite extreme.

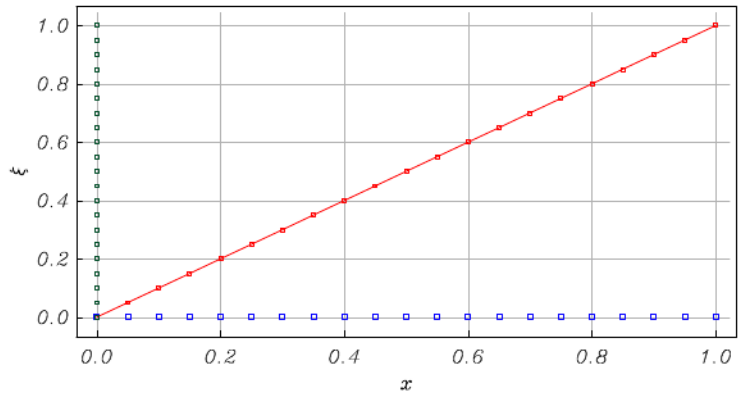

Figure 14.8: Grid mapping with $c_1 = -1 \times 10^{-22}$, and $c_2 = 0$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are uniformly distributed with a 1-to-1 correspondence.

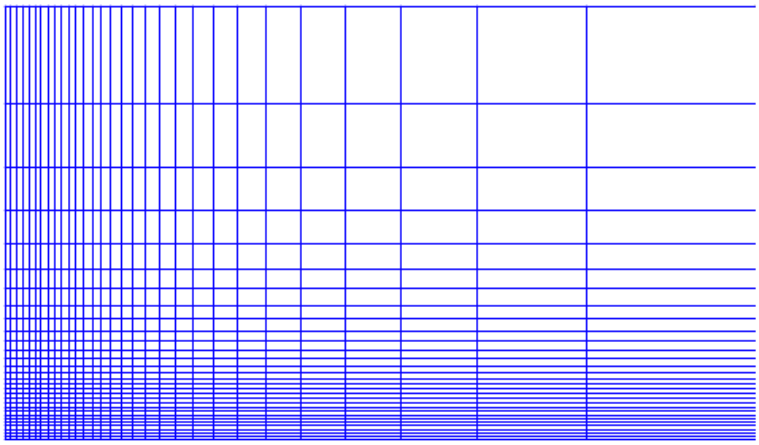

Figure 14.9: Example of 2D Grid clustering at the lower and left boundaries in the physical plane.

14.2 Adaptive Stencils

A somewhat different approach is taken by a variety of methods that adapt the stencil as the boundary is encountered (see Problem sheet 2, Question 6). One of the earliest techniques is to modify the weights of the discretisation stencil as soon as the stencil crosses the boundary.

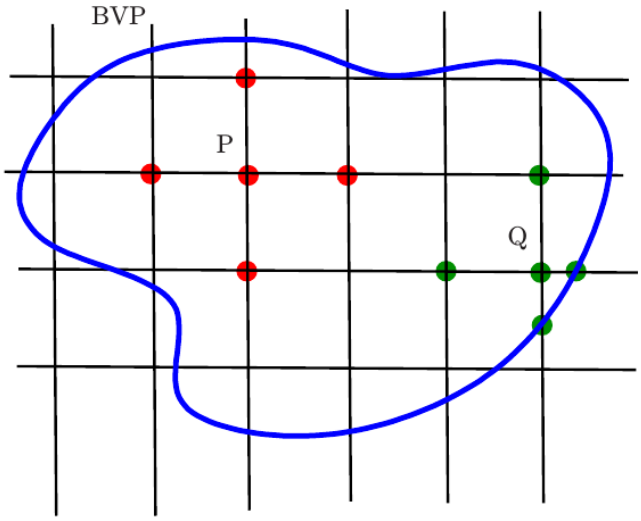

Figure 14.10: Irregular boundary without mappings

The general technique is straightforward: assume that we wish to discretise the Poisson equation using the standard five-point stencil. For each point on the grid where all five points fall within the computational domain we get

$$\frac{1}{(\Delta x)^2}\left(u_{i+1,j}+u_{i-1,j}\right)+\frac{1}{(\Delta y)^2}\left(u_{i,j+1}+u_{i,j-1}\right)-\left(\frac{2}{(\Delta x)^2}+\frac{2}{(\Delta y)^2}\right)u_{i,j}=f_{i,j}. \quad (14.15)$$

This discretisation may be used, as is, if all the grid points fall within the computational domain, as the P-data point in Fig. 14.10. As one or more points cross the boundary/interface,

such as point Q in Fig. 14.10 (also see Fig. 14.11) modifications have to made. In this case, we use the general non-equidistant grid formulae. Even though this scheme is rather straightforward to implement and thus quite popular, it has the disadvantage of disrupting the uniform stencil structure of the elliptic operator, and requires considerable care to implement, and cater for due to the many scenarios possible at the boundary.

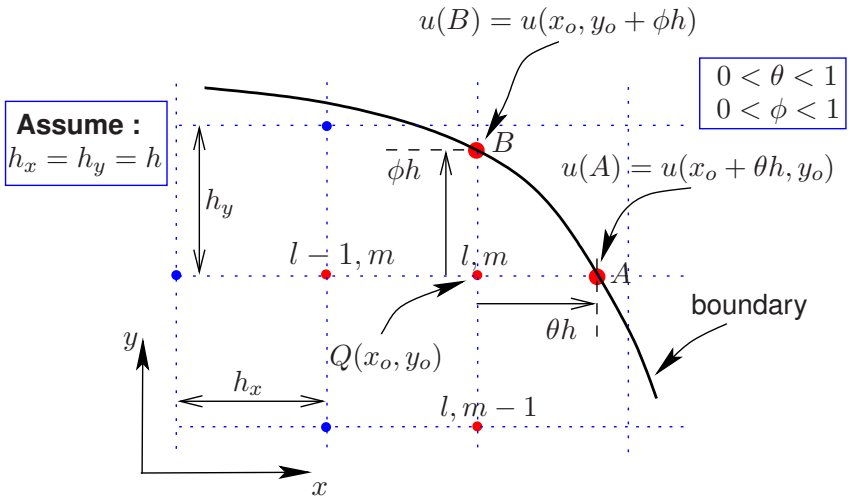

Figure 14.11: Non-uniform stencil at boundary treatment.

The nonuniform grid formulae are derived by usage of Taylor series expansions. With reference to Fig. 14.11 where for ease of discussion we assume uniform grid-spacings $h_x = h_y = h$, suppose we required to discretise Laplace's equation, namely the u_{xx}, u_{yy} terms at the point Q. Taylor expansions thus give

$$u(x_o + \theta h, y_o) = u(x_o, y_o) + \theta h \frac{\partial u}{\partial x}|_{l,m} + \frac{(\theta h)^2}{2} \frac{\partial^2 u}{\partial x^2}|_{l,m} + \frac{(\theta h)^3}{3!} \frac{\partial^3 u}{\partial x^3}|_{l,m} + \dots . \tag{14.16}$$

Similarly

$$u(x_o - h, y_o) = u(x_o, y_o) - h \frac{\partial u}{\partial x}|_{l,m} + \frac{h^2}{2} \frac{\partial^2 u}{\partial x^2}|_{l,m} - \frac{h^3}{3!} \frac{\partial^3 u}{\partial x^3}|_{l,m} + \dots . \tag{14.17}$$

Elimination of the $(\partial u/\partial x)|_{l,m}$ term gives

$$\theta u(x_o - h, y_o) + u(x_o + \theta h, y_o) = (1 + \theta) u(x_o, y_o) + \frac{\theta h^2}{2}(1 + \theta) \frac{\partial^2 u}{\partial x^2}|_{l,m} + O(h^3), \tag{14.18}$$

or (on truncation of the $O(h^3)$ term),

$$\theta U_{l-1,m} + U(A) = (1 + \theta) U_{l,m} + \frac{\theta h^2}{2}(1 + \theta) \frac{\partial^2 u}{\partial x^2}|_{l,m}. \tag{14.19}$$

Hence,

$$\frac{\partial^2 u}{\partial x^2}|_{l,m} = \frac{2\theta U_{l-1,m} + 2U(A) - 2(1 + \theta) U_{l,m}}{\theta h^2 (1 + \theta)}. \tag{14.20}$$

Similar treatment in the y-direction results in

$$\frac{\partial^2 u}{\partial y^2}|_{l,m} = \frac{2\phi U_{l,m-1} + 2U(B) - 2(1 + \phi) U_{l,m}}{\phi h^2 (1 + \phi)}. \tag{14.21}$$

Thus if one was discretising Poisson's equation $\nabla^2 u = f(x,y)$ (say) it follows, the discretised form at the boundary would be

$$\frac{2\theta U_{l-1,m} + 2U(A)}{\theta(1+\theta)} + \frac{2\phi U_{l,m-1} + 2U(B)}{\phi(1+\phi)} - 2U_{l,m}\left(\frac{1}{\theta} + \frac{1}{\phi}\right) = h^2 f_{l,m}. \tag{14.22}$$

Observe that the truncation error term indicates that this would though only be $O(h)$ accurate; $O(h^2)$ accuracy is recovered on setting both $\phi = \theta = 1$ in the grid interior!

Various other techniques, based on mappings, reflections and interpolations are common, but in all cases, both the right-hand side and the discretisation stencil have to be modified to accommodate the presence of the boundary.

14.3 Elliptic Schemes

Elliptic PDES have the property that solutions are generally very smooth. The smoothness property is an advantage, and for this reason Laplace and Poisson Equations are a very good choice. With Poisson grid generators the mapping is obtained by specifying the desired grid points (x,y) say, on the boundary of the physical domain with the interior point distribution determined through solution of the equations

$$\xi_{xx} + \xi_{yy} = \mathcal{P}(\xi,\eta), \quad \eta_{xx} + \eta_{yy} = \mathcal{Q}(\xi,\eta) \tag{14.23}$$

with (ξ,η) representing the coordinates of the computational grid, with $(\mathcal{P},\mathcal{Q})$ used to control point spacings within the grid interior. The above are then transformed to computational space by changing the roles of the independent and dependent variables, such that

$$\left.\begin{array}{ll} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} & = J^2\left(\mathcal{P}x_\xi + \mathcal{Q}x_\eta\right) \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} & = J^2\left(\mathcal{P}y_\xi + \mathcal{Q}y_\eta\right) \end{array}\right\}, \tag{14.24}$$

where

$$\left.\begin{array}{ll} \alpha & = x_\eta^2 + y_\eta^2 \\ \beta & = x_\xi x_\eta + y_\xi y_\eta \\ \gamma & = x_\xi^2 + y_\xi^2 \\ J & = x_\xi y_\eta - x_\eta y_\xi \end{array}\right\}. \tag{14.25}$$

These are solved on a uniformly spaced computational grid, and thus provides the (x,y) grid points in the physical field. Usually simple Dirichlet boundary conditions suffice. The advantage is that the resulting grid is smooth and complex boundaries are easily accommodated. Prescribing or choosing $(\mathcal{P},\mathcal{Q})$ though can prove time consuming, since grid point control within the grid interior is more difficult to achieve.

A typical example of what is though achievable is shown in Fig. 14.13. Here with reference to Fig. 14.12, a cut in the physical plane is introduced along o-a, with a requirement that the solution along o-a must be identical to the solution along b-c. Surface boundary conditions (inviscid flow tangency condition) along $\mathrm{a}-\mathrm{b}$ is thus mapped to the lower $\mathrm{a}-\mathrm{b}$ boundary in the figure on the right. The flow far-field conditions are thus imposed along the upper boundary (o-g-f-e-d-c). The flow field equations (transformed Laplace Equation, say as an example) along with Eqns. (14.24) for the grid mappings are both solved using Successive Over Relaxation (SOR) discussed in earlier lectures.

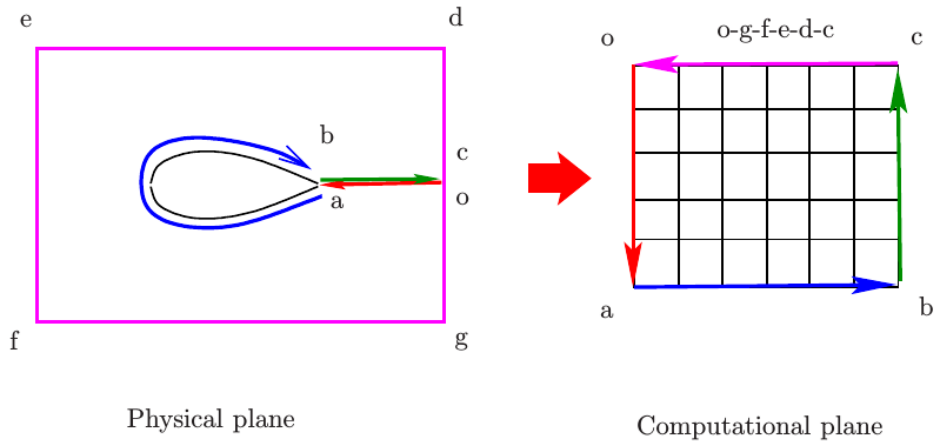

Figure 14.12: Example of complex boundary using Elliptic mapping.

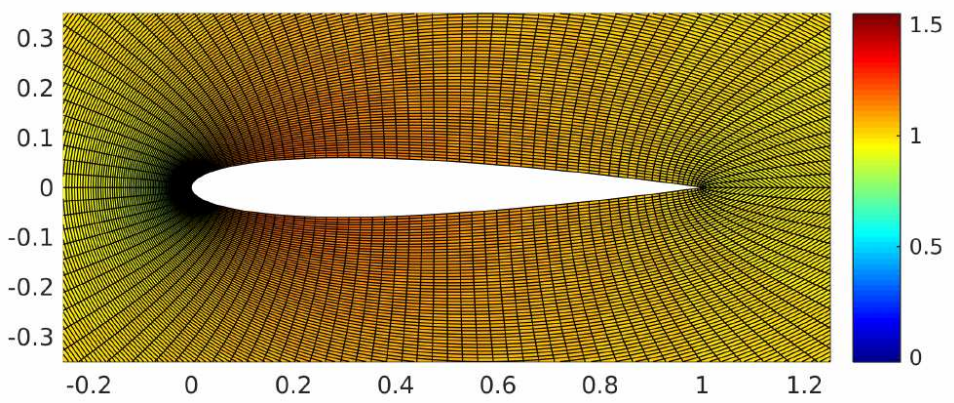

Figure 14.13: Examples of grid generated using (14.24). The strategy used is shown in Figure 14.12

Aside: Derivation of Eqn. (14.24)

This is arrived at by use of (14.8) and (14.9), noting that

$$\xi_{xx} = \frac{\partial}{\partial x}\xi_x = \left(\xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta}\right)\xi_x = \left(\frac{y_\eta}{J}\frac{\partial}{\partial \xi} - \frac{y_\xi}{J}\frac{\partial}{\partial \eta}\right)(y_\eta J^{-1}) \tag{14.26}$$

$$\xi_{yy} = \frac{\partial}{\partial y}\xi_y = \left(\xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta}\right)\xi_y = \left(\frac{x_\eta}{J}\frac{\partial}{\partial \xi} - \frac{x_\xi}{J}\frac{\partial}{\partial \eta}\right)(x_\eta J^{-1}). \tag{14.27}$$

Similarly

$$\eta_{xx} = \frac{\partial}{\partial x}\eta_x = \left(\xi_x \frac{\partial}{\partial \xi} + \eta_x \frac{\partial}{\partial \eta}\right)\eta_x = \left(\frac{y_\eta}{J}\frac{\partial}{\partial \xi} - \frac{y_\xi}{J}\frac{\partial}{\partial \eta}\right)(-y_\xi J^{-1}) \tag{14.28}$$

$$\eta_{yy} = \frac{\partial}{\partial y}\eta_y = \left(\xi_y \frac{\partial}{\partial \xi} + \eta_y \frac{\partial}{\partial \eta}\right)\eta_y = \left(-\frac{x_\eta}{J}\frac{\partial}{\partial \xi} + \frac{x_\xi}{J}\frac{\partial}{\partial \eta}\right)(x_\xi J^{-1}). \tag{14.29}$$

After some considerable manipulation of the above expressions (task made easier by use of a symbolic manipulator, eg. *Mathematica* or *Maple* software), Eqns. (14.23) can be shown to reduce to (14.24).

14.4 Complex 3D meshing

Generating meshes is a vast field, we have only discussed a few of the most basic approaches – think about meshing a three-dimensional object (or Domain) and what that would entail.

Alternative techniques to work in the physical plane are available, but we do not consider them in this course - (Google, finite elements and unstructured grids). Examples of quite complex problems are shown in Fig. 14.14, these as you will appreciate are quite advanced*.

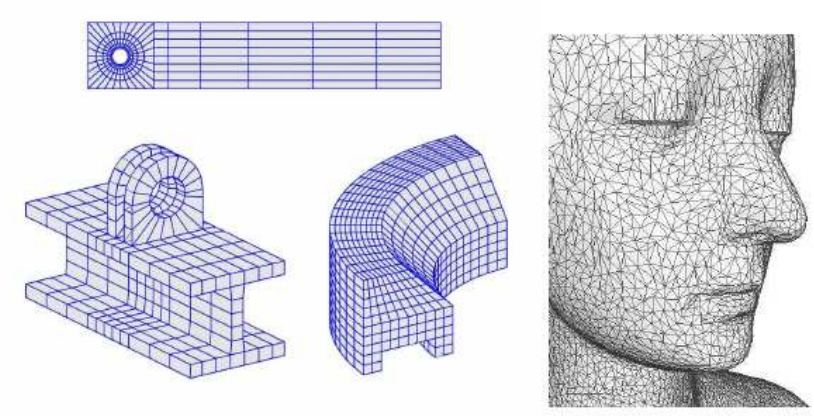

Figure 14.14: Finite element and unstructured grids in complex geometry (references below in footnote).

For further details, see Tannehill, Anderson & Pletcher, Computational Fluid Mechanics and Heat Transfer).

*see: http://www.featool.com/tutorial/2015/08/24/creatingstructured-grids-using-featool-matlab-functions. http://www.sci.utah.edu/the-institute/highlights/24research-highlights/cibc-highlights/439-cleaver.html

15 Hyperbolic PDEs

15.1 First-order PDE

Consider a quasi-linear equation of the type

$$a\,\frac{\partial z}{\partial x} + b\,\frac{\partial z}{\partial y} = c, \tag{15.1}$$

where $a,\ b,\ c$ are functions of $x,\ y$. This can be simplified by changing the variables $x,\ y$ to $s,\ t$ so that Eqn. 15.1 becomes

$$\left(\frac{\partial z}{\partial t}\right)_s = c. \tag{15.2}$$

Suitable variables s and t are identified by comparing Eqn. 15.1 with the identity

$$\frac{\partial z}{\partial t} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial t}, \tag{15.3}$$

provided s and t are chosen such that

$$\frac{\partial x}{\partial t} = a, \quad \text{and} \quad \frac{\partial y}{\partial t} = b, \tag{15.4}$$

in which case Eqn. 15.1 reduces to Eqn. 15.2.

The variable s is defined by noting that, if s is constant, Eqns. 15.4 gives a coordinate direction along which the PDE reduces to an *ordinary differential equation* (ODE)

$$\frac{dy}{dx} = \frac{\partial y/\partial t}{\partial x/\partial t} = \frac{b}{a}, \tag{15.5}$$

which can be integrated (in principle – at least). The "*constant*" of integration is an arbitrary function of s (since s has been held constant) and any convenient function can be chosen to define s. Hence

$$s = f(x, y), \tag{15.6}$$

where f is a known function which defines s explicitly. The variable t can be defined by integrating **either** of the expressions 15.4, *e.g.* the first of these gives

$$t = \int \frac{dx}{a} - g(s), \tag{15.7}$$

the integration carried out with s held constant; Eqn. 15.6 is re-arranged in the form

$$y = y(x, s)$$

and used to express $a(x, y)$ as $a(x, s)$. The arbitrary function of s in Eqn. 15.7 may be chosen for convenience; it is usually taken to be zero.

Equation 15.2 is then integrated, first of all expressing c as a function of s and t by means of Eqns. 15.6 and 15.7,

$$z = \int c\,dt + h(s). \tag{15.8}$$

The integration is carried out keeping s constant. The arbitrary function $h(s)$ is determined by inserting the given boundary condition into Eqn. 15.8.

Observe, generally we have the **characteristic** relations

$$dt = \frac{dx}{a} = \frac{dy}{b} = \frac{dz}{c}. \tag{15.9}$$

We may choose what approach and variable to use, based on ease of integration of these **characteristic** variables. As an example, or summary, note

$$\frac{dz}{dy} = \frac{c}{b}, \quad \frac{dz}{dx} = \frac{c}{a} \text{ along } \frac{dy}{dx} = \frac{b}{a}.$$

15.1.1 Example solution to first-order PDE

Consider the equation

$$y\frac{\partial U}{\partial x} + \frac{\partial U}{\partial y} = 2,$$

where U is known along an initial segment Γ along $y = 0$ and $0 \leq x \leq 1$ as indicated in Fig. 15.1. The PDE reduces to an ODE on the characteristic curve

$$\frac{dy}{dx} = \frac{1}{y}, \tag{15.10}$$

along which the solution is given by

$$\frac{dU}{dy} = 2. \tag{15.11}$$

Integration of Eqn. 15.10 gives $x = y^2/2 + A$ where A is a constant for each characteristic. So for a characteristic through $R(x_r, 0), A = x_r$. Thus the equation defining this path is

$$y^2 = 2(x - x_r)$$

and the solution on this characteristic will be given by $U = 2y + B$, with B a constant determined from the initial condition along $y = 0$ – if $U = U_r$ at $R(x_r, 0)$ then $B = U_r$, *i.e.*

$$U = 2y + U_r. \tag{15.12}$$

Since initial values of U are known along Γ or the segment P–Q, where $0 \leq x_r \leq 1$, it follows that the solution is only known in the region bounded by and including the terminating characteristics $y^2 = 2x$ and $y^2 = 2(x-1)$; outside this region (non-hatched area in Fig. 15.1) the solution will be undefined.

Along the $y^2 = 2x$ characteristic the solution U will be uniquely determined by the value of U_o at $P(0, 0)$, *i.e.* $U = 2y + U_o$. In other words the initial values for U on the initial curve $y^2 = 2x$ can not be arbitrarily prescribed, but is dependent upon the Γ initial data path defined*.

Importantly, observe that the initial data U_r prescribed along Γ need **not be smooth**, it may well be **discontinuous**, but the solution along the characteristic path $y^2 = 2(x - x_r)$ will still be uniquely defined to be $U = 2y + U_r$.

*see G.D. Smith for further details

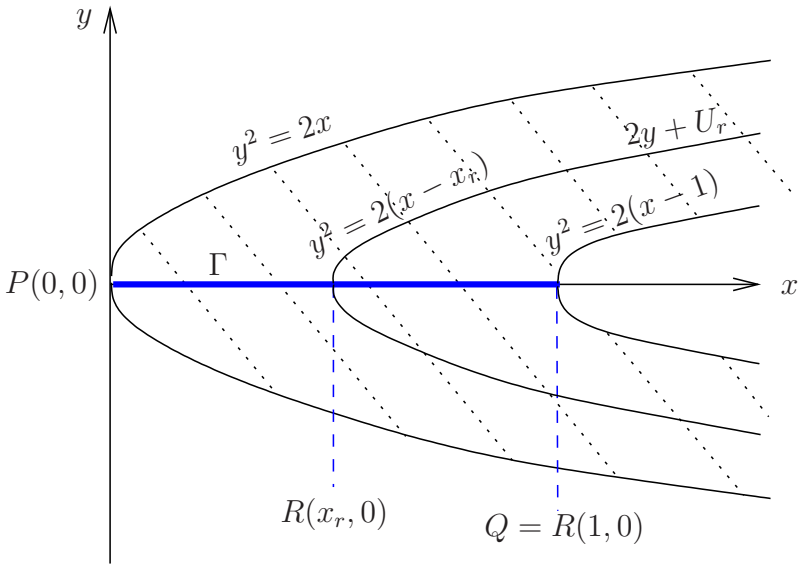

Figure 15.1: Solution domain, $\Gamma \in 0 \leq x \leq 1,\ y = 0$ (blue solid line) is the path along which the initial conditions are prescribed.

15.2 The one-dimensional linear advection equation and Upwinding

Consider the problem for $u(x,\, t)$

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \ \text{ for } \ t > 0, \ \ \text{initial condition } \ u(x,\, 0) = f(x)\ , \tag{15.13}$$

where c is a **positive** constant. The characteristics of this equation are

$$dx/dt = c \quad \text{or} \quad x - ct = \quad \text{constant.}$$

Furthermore, u is constant along each characteristic and so the solution is $u = f(x - ct)$. This represents a wave travelling in the positive x-direction with speed c, **without change of magnitude or shape** as shown in Figure 15.2 – we emphasis that the exact solution of Eqn. 15.13 says that the initial function $f(x)$ propagates unaltered in form in the positive x-direction as time progresses. The PDE simply translates as t progress, the profile $f(x)$ with velocity c to the right if $c > 0$ and to the left if $c < 0$.

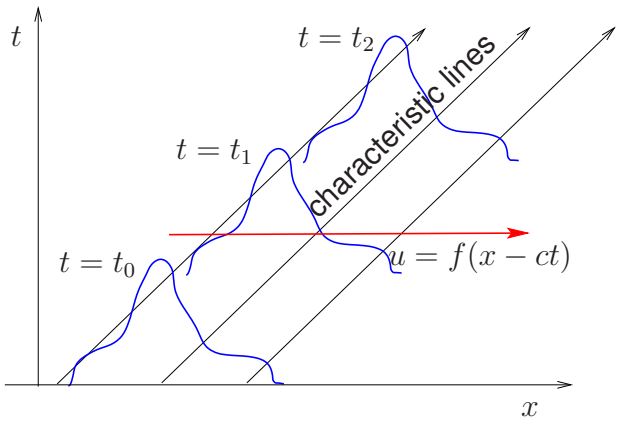

Figure 15.2: Propagation of a function $u = f(x)$ with time t.

Can we develop finite-difference methods (FDMs) with these properties? In the present context by this we mean, that our FDM reproduces precisely, or close to as possible, the expected behaviour of $f(x)$ propagating unaltered in form as time progresses.

For future reference, we observe that for a Fourier mode with $f(x) = \exp(ix/h) \equiv e^{in\xi}$. Here x/h is non-dimensional and may be viewed as a *wavenumber*; similarly (ck) has dimensions of a length, since c is a speed and k a time-scale, therefore $q = ck/h$ is non-dimensional and may too be associated as a wavenumber, in the following

$$u_n^{j+1} \equiv u(nh,\,(j+1)k) = u_n^j \exp(-i\xi q) \;\text{ where }\; q = \frac{ck}{h} > 0. \tag{15.14}$$

The value $q = ck/h$ is a very important dimensionless number called the **Courant number** (also referred to as the **Courant-Friedrichs-Lewy** (**CFL**)-condition). We shall see this plays an important role when considering the effectiveness of FD discretisations in hyperbolic problems. It may also be viewed as

$$\frac{ck}{h} \equiv \;\text{ ratio of }\; \frac{\textit{physical distance moved}}{\textit{grid-spacing}}.$$

The real solution therefore has a growth factor $\lambda \exp(-i\xi q)$. We note that for the model equation (15.13) $|\lambda| = 1$ so that there is no growth or decay. We can model our simple equation in many different ways. We shall use a forward difference for u_t and consider 6 schemes:

$$\frac{\partial u}{\partial t} \approx \frac{U_n^{j+1} - U_n^j}{-q} = \begin{cases} \textbf{A}: & \frac{1}{2}\Delta U_n^j \equiv \frac{1}{2}(U_{n+1}^j - U_{n-1}^j) & \text{Explicit, centred} \\ \textbf{B}: & U_{n+1}^j - U_n^j & \text{Explicit, forwards} \\ \textbf{C}: & U_n^j - U_{n-1}^j & \text{Explicit, backwards} \\ \textbf{D}: & \frac{1}{2}\Delta U_n^{j+1/2} & \text{Two-step, centred} \\ \textbf{E}: & \frac{1}{2}\left[\frac{1}{2}\Delta U_n^{j+1} + \frac{1}{2}\Delta U_n^j\right] & \text{Crank-Nicolson} \\ \textbf{F}: & \frac{1}{2}\Delta U_n^j - \frac{1}{2}q\delta^2 U_n^j & \text{Lax-Wendroff} \end{cases}$$

(We shall also later consider Keller's 'Box scheme'). We use the Fourier method to investigate the stability of all of these schemes, looking for a solution $U_n^j = \lambda^j \exp(in\xi)$. We find:

$$\lambda = \begin{cases} \textbf{A}: & 1 - iq\sin\xi \\ \textbf{B}: & 1 - q(e^{i\xi} - 1) \\ \textbf{C}: & 1 - q(1 - e^{-i\xi}) \\ \textbf{D}: & \frac{1}{4}\left[iq\sin\xi \pm \sqrt{4 - q^2\sin^2\xi}\right]^2 \quad \text{(two steps)} \\ \textbf{E}: & \dfrac{2 - iq\sin\xi}{2 + iq\sin\xi} \\ \textbf{F}: & 1 - iq\sin\xi - 2q^2\sin^2\tfrac{1}{2}\xi \end{cases}$$

We recall that if

1. If $|\lambda| > 1 + O(k)$, the method is unstable.
2. If $|\lambda| < 1$, it is dissipative.

3. While if $|\lambda| = 1$, it is conservative. We mean (*loosely*) that the solution propagates in time without change, and is also exact as the $t-$variable progresses.

We see therefore that

$$|\lambda|^2 = \begin{cases} \textbf{A}: & 1 + q^2 \sin^2 \xi & & \text{stable only if } q^2 = O(k) \; i.e. \; k \sim h^2 \\ \textbf{B}: & 1 + 2q(1+q)(1-\cos\xi) & \geq 1 \quad \forall\, q, \xi & \text{hopelessly unstable} \\ \textbf{C}: & 1 - 2q(1-q)(1-\cos\xi) & \leq 1 \quad \forall\, q \leq 1 & \text{stable, dissipative} \\ \textbf{D}: & 1 & & \text{provided } q^2 \sin^2 \xi \leq 4 \quad \text{or} \quad \tfrac{1}{2} q \leq 1 \\ \textbf{E}: & 1 & \forall\, q & \text{stable and conservative} \\ \textbf{F}: & 1 - 4q^2(1-q^2)\sin^4 \tfrac{1}{2}\xi & \leq 1 \text{ if } q \leq 1 & \text{dissipative, but less than } \textbf{C} \end{cases}$$

We may also be interested in $\arg\lambda$, which determines the **phase** of each mode. In case **E**, for example,

$$\arg\lambda = -2\tan^{-1}(\tfrac{1}{2}q\xi) \approx -q\xi \;\text{ when }\; \xi \;\text{ is small.}$$

All the above schemes agree well with the exact solution, for which $\arg\lambda = -q\xi$, when ξ is small. The waves with higher values of ξ are **dispersive**; their phase velocity varies with frequency. A computer demonstration will illustrate the effects of this. Although the amplitude of each wave component may be conserved, phase differences develop which alter the shape of the whole.

We can see from the above the importance of the CFL condition, $q \leq 1$. The Courant or CFL number is a non-dimensional number that plays a central role in the numerical solution of hyperbolic equations. If c can be thought of a speed, $q = ck/h \equiv c\Delta t/\Delta x$, as alluded to earlier, can be thought of a distance measured in grid points that a particle or information reaches to, in an increment of time $k \equiv \Delta t$.

Another very important conclusion we can draw is that backwards, or **Upwind** differences are a good idea. Case **C** is stable provided the CFL condition holds, whereas **B** is unconditionally unstable. If $c < 0$, the stable scheme is **B**. In general, the **Upwind** scheme can be written

$$U_n^{j+1} = U_n^j - sc\Delta U_n^j + s|c|\delta^2 U_n^j \qquad \text{where} \quad s = \tfrac{k}{2h}. \tag{15.15}$$

The need for upwind differences can be interpreted in terms of Eqn. 15.13's characteristics, $x - ct =$ *constant*, which must pass through the **Numerical Domain of Dependence**.

Above we have a very simple equation with constant coefficients (*i.e. the* c). In this very special case, setting $q = 1$ for some of the discretisation schemes above, the numerical scheme reproduces the exact solution with no error. However in more complex hyperbolic systems with varying coefficients maintaining this *exactness* is not that straightforward and should not be expected, unless one goes to considerable effort towards honouring the true physical propagation paths or characteristics of the equations set.

15.3 Upwinding for simultaneous equations

Let $\mathbf{u}(x,\, t)$ be a p-dimensional vector satisfying the equation

$$\mathbf{u}_t + A\mathbf{u}_x = \mathbf{d}$$

where A is a $p \times p$ matrix, which for simplicity we shall assume to be constant. We shall investigate how the upwinding method generalises to this problem, by **diagonalising** the matrix A, which we assume to have p distinct eigenvalues $\lambda_1, \ldots \lambda_p$ and corresponding eigenvectors. We make the linear transformation $\mathbf{u} = S\mathbf{v}$ where S is the matrix whose columns are the eigenvectors of A, so that

$$S^{-1}AS = D \equiv \mathrm{diag}(\lambda_1,\, \lambda_2, \ldots, \lambda_p) \qquad \text{and so} \quad A = SDS^{-1}\ .$$

Then v_i, the i-th component of $\mathbf{v}$, satisfies the equation

$$(v_i)_t + \lambda_i (v_i)_x = (S^{-1}\mathbf{d})_i\ .$$

We have thus separated the problem into p separate ones, each with its own characteristic family, $dx/dt = \lambda_i$, and we can use **upwinding** on each one in turn. Note that if any of the λ_i are complex, then some of the p equations are elliptic, and we cannot use a time-stepping approach for them. We assume here that all the λ_i are real. From (10.9), the upwind scheme for $\mathbf{v}$ is

$$\mathbf{V}_n^{j+1} = \mathbf{V}_n^j - sD\Delta\mathbf{V}_n^j + sD^+\delta^2\mathbf{V}_n^j + kS^{-1}\mathbf{d}_n^j\ ,$$

where $D^+ \equiv \mathrm{diag}(|\lambda_1|,\, |\lambda_2|, \ldots, |\lambda_p|)$, and $\mathbf{V}$ is the finite-difference approximation to $\mathbf{v}$. Then transforming back, defining $\mathbf{U} = S\mathbf{V}$ so that $\mathbf{V} = S^{-1}U$, we find

$$\mathbf{U}_n^{j+1} = \mathbf{U}_n^j - sA\Delta\mathbf{U}_n^j + sA^+\delta^2\mathbf{U}_n^j + k\mathbf{d}_n^j \ \text{ where } \ A^+ = SD^+S^{-1}\ . \tag{15.16}$$

This is the required generalisation of upwinding for p simultaneous equations. If all the eigenvalues of A are positive, then $D^+ = D$ and $A^+ = A$, while $A^+ = -A$ if they are all negative. Otherwise, A^+ bears no simple relation to A, and we must be very careful how we define "**up**" when upwinding. The stability condition for Eqn. 15.16 is

$$\frac{k}{h}\max_{i=1\ldots p}|\lambda_i| \le 1\ . \tag{15.17}$$

16 Dissipation and Dispersion

16.1 Summary of findings through numerical experiments

We found on solving the very simple linear advection equations the following:

1. CFL > 1 : All methods examined tend to blow up schemes (A,B,C & F), *i.e.* were found to be unstable numerically.
2. CFL $= 1$: Schemes A, B were unstable, C & F give exact result.
3. CFL < 1 : Schemes C & F display dispersive, dissipative behaviour and out of phase with exact solution result. Results exasperated at regions where sudden changes, or large gradients in the u-field arise.
4. CFL < 1 : With greater resolution in grid, a reduction in the dispersive, dissipative behaviour of schemes C & F arose. Dissipation effects of scheme F were less than C. Dispersive behaviour in C was not evident; scheme F displays relatively weaker dissipation but significant dispersive behaviour.

Thus the numerical discretisations gave rise to *advection, dissipation, dispersion* and *phase differences*. These mechanisms are central to the behaviour of PDEs and their discrete models, and together account for most linear phenomena. We expect that the advection equation will advect (of course!), but how and why do the other effects arise? The phenomena that emerge from this simple study turn out to be fundamental, in understanding and being able to control modelling accuracy, when we attempt to discretise more complicated problems.

Looked at, from the viewpoint of advection of **energy**, consider the equation

$$u_t + cu_x = 0,$$

which with sufficiently smooth initial data $u(x,0) = u_o(x)$, has the solution

$$u(x,t) = u_o(x - ct).$$

With this PDE, energy propagates at a finite speed and is **conserved** : by which we mean nothing is *lost, gained (or dissipated)*. However, not always does all the energy propagate at exactly the same finite speed. Consider the dissipation (or diffusion) given by the parabolic one-dimensional (heat) equation

$$u_t = u_{xx},$$

with sufficiently well-behaved initial data $u(x,0) = u_o(x)$, the solution using Fourier transforms, can be written as

$$u(x,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\alpha x - \alpha^2 t}\hat{u}_o(\alpha)\, d\alpha, \tag{16.1a}$$

$$= \frac{1}{4\pi t}\int_{-\infty}^{\infty} e^{-(x-s)^2/4t} u_o(s)\, ds. \tag{16.1b}$$

Here $\hat{u}_o(\alpha)$ is the Fourier transform of the initial data with α the spatial wavenumber. Physically, the above solution asserts that the oscillatory components in the initial data of

wavenumber α decay at a rate of $e^{-\alpha^2 t}$. The solution will then be composed of increasingly smooth wave components.

The simplest example of dispersion is modelled by the equation

$$u_t = i\, u_{xx},$$

the one-dimensional **Schrodinger equation**. Similar to the the heat equation, the solution to this can be expressed as

$$u(x,t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{i\alpha x - i\alpha^2 t}\hat{u}_o(\alpha)\, d\alpha, \tag{16.2a}$$

$$= \frac{1}{4i\pi t}\int_{-\infty}^{\infty} e^{i(x-s)^2/4t} u_o(s)\, ds. \tag{16.2b}$$

giving quite different behaviour. Dispersion is observed in the form of solutions that (rather than decaying as $t \to \infty$, as for the heat equation) break into oscillatory **wave packets**. Figure 16.1 illustrates these features.

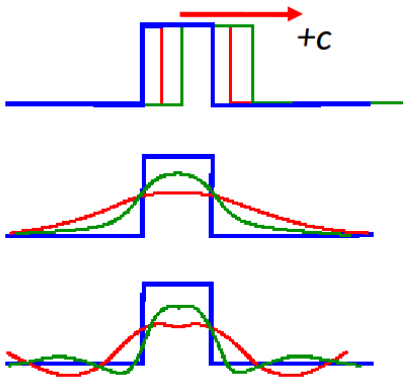

Figure 16.1: Propagation of a top hat initial condition in time under an advection (top), dissipation (middle) and dispersive (bottom) equation. Initial condition (blue) and later time instances shown.

Diffusion refers to a phenomenon where some quantity spreads out in space as time goes on. Dissipation is used to refer to loss of energy. The effects are related. If you consider the heat equation

$$T_t = \mu \nabla^2 T, \tag{16.3}$$

and start with an initial condition with high temperature in some small region, then the temperature distribution spreads out. This would be called diffusion. If you calculate

$$\int_\Omega T^2(x,t)\, dx, \tag{16.4}$$

this quantity will decrease with time. So there is a loss of energy. (This is not really energy in physical sense if T is temperature, you can substitute velocity here). Both are caused by the Laplacian term. If something spreads out, its square integral will probably decrease.

Dispersion is related to wave phenomena. For a linear wave equation

$$u_t + c u_x = 0, \quad \text{or} \quad u_{tt} = c^2 u_{xx} \tag{16.5}$$

you can resolve the solution into Fourier modes. Then each mode will travel at the same speed. We say such equations are **non-dispersive**. These two equations are also **non-dissipative**. For smooth solutions with periodic boundary conditions, they conserve all integrals of the form

$$\int_{\Omega} |u(x,t)|^P \, dx. \tag{16.6}$$

An equation like

$$u_t + cu_x = \nu u_{xxx}, \tag{16.7}$$

would have solutions whose Fourier modes travel at different speed depending on ν and wave number. We say these equations exhibit **dispersive** behaviour. We observe these phenomena at the level of PDE and we can map them to the behaviour of numerical schemes.

The effect of dispersion, therefore, is that often spurious oscillations or wiggles occur in solutions that include sharp gradients, discontinuities, or shocks. This is usually manifested by high-frequency oscillations trailing the particular effect.

If we solve a non-dissipative and non-dispersive PDE with a numerical scheme, we want the numerical solutions to be non-dissipative and non-dispersive too. As we have shown, sadly this is not possible since numerical solutions exhibit these behaviours even if the exact solution does not. Then we say that the numerical scheme is dissipative and/or dispersive and the challenge is to devise discretisation schemes which at least minimise these effects which are **not present** in the exact solution.

16.2 Modified Equations

Solutions of finite-difference schemes applied to partial differential equations are typically thought of as approximations to the exact solutions of the continuous problem. It is interesting and instructive to start with the discrete finite-difference equation and ask the question *"which partial differential equation has this discrete equation as an exact solution"*. We consider the one-dimensional diffusion equation, discretised in space by a centered second-order finite-difference scheme and in time by a forward Euler method, namely

$$\frac{U_n^{j+1} - U_n^j}{k} = \mathcal{D}\left(\frac{U_{n-1}^j - 2U_n^j + U_{n+1}^j}{h^2}\right),$$

where $\mathcal{D}$ is the diffusion, k the time step and h the spatial step interval. Employing a Taylor series about U_n^j, we obtain for the time and space derivative, respectively,

$$\frac{U_n^{j+1} - U_n^j}{k} = \frac{\partial u}{\partial t} + \frac{k}{2}\frac{\partial^2 u}{\partial t^2} + \dots$$

and

$$\frac{U_{n-1}^j - 2U_n^j + U_{n+1}^j}{h^2} = \frac{\partial^2 u}{\partial x^2} + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4},$$

which results in

$$\frac{U_n^{j+1} - U_n^j}{k} - \mathcal{D}\frac{U_{n-1}^j - 2U_n^j + U_{n+1}^j}{h^2} = \frac{\partial u}{\partial t} - \mathcal{D}\frac{\partial^2 u}{\partial x^2} - \mathcal{D}\frac{h^2}{12}\frac{\partial^4 u}{\partial x^4} + \frac{k}{2}\frac{\partial^2 u}{\partial t^2} + \dots$$

As both the time and space step tend to zero, the discrete equation approaches the continuous equation, which renders the chosen scheme **consistent** in space and time. The term on the right-hand side confirms a second-order accuracy in space and a first-order accuracy in time. Moreover, the largest error term is dissipative in nature (due to the second derivative in time and the fourth derivative in space) – using the fact that

$$\frac{\partial^2 u}{\partial t^2} = \mathcal{D}\frac{\partial}{\partial t}(\frac{\partial^2 u}{\partial x^2}) = \mathcal{D}^2\frac{\partial^4 u}{\partial x^4}.$$

Hence by a judicious choice of $\mathcal{D}, k$ and h, one may eliminate the largest error terms in the above equation for the most accurate representation of the FD to the exact PDE, but the space and time step required to meet this, is very restrictive, and so seldom used in practice, *i.e.*

$$\frac{\mathcal{D}k}{h^2} = \frac{1}{6}, \quad \text{think about Milne's method, Eqn. 7.14.}$$

Linear Advection equation, up-winded scheme :
Consider the discretisation of $u_t + cu_x = 0$,

$$U_n^{j+1} = U_n^j - q(U_n^j - U_{n-1}^j), \quad \text{with} \quad q = ck/h.$$

A Taylor series expansion around U_n^j, gives

$$u + ku_t + \frac{k^2}{2}u_{tt} + \frac{k^3}{6}u_{ttt} + \cdots - u + q\left(u - \left[u - hu_x + \frac{h^2}{2}u_{xx} - \frac{h^3}{6}u_{xxx} + \ldots\right]\right) = 0.$$

On simplification, and noting $q = ck/h$, we obtain

$$u_t + cu_x = \frac{1}{2}(chu_{xx} - ku_{tt}) - \frac{1}{6}(k^2u_{ttt} + ch^2u_{xxx}).$$

Next we use, assuming c to be a constant $u_{tt} = -cu_{xt} = -cu_{tx} = c^2u_{xx}$, while ignoring the $u_{ttt},\ u_{xxx}$ terms as being even smaller order, followed by simplification, we get

$$u_t + cu_x = \frac{ch}{2}\left(1 - \frac{ck}{h}\right)u_{xx},$$

i.e.

$$u_t + cu_x = \frac{ch}{2}(1 - q)u_{xx}. \tag{16.8}$$

A similar analysis for the Lax-Wendroff discretisation

$$U_n^{j+1} = (1 - q^2)U_n^j + \frac{q(q-1)}{2}U_{n+1}^j + \frac{q(q+1)}{2}U_{n-1}^j$$

gives

$$u_t + cu_x = -\frac{ch^2}{6}(1 - q^2)u_{xxx}. \tag{16.9}$$

Comparing Eqns.16.8 and 16.9, and recalling results from the numerical experimentation, we observe the following

1. Setting $q = 1$, namely the CFL condition, both expressions recover the exact PDE.
2. In Eqn. 16.8, the up-winded scheme gives a u_{xx} term factored by $ch(1-q)/2$ *diffusion-like* term. Thus provided $1 - q < 1$, we may expect our numerical solutions to exhibit diffusion or dissipation effects. While for $q > 1$ a negative diffusion term appears, in which case we should expect our numerical method to be unstable (ill-posed).
3. In Eqn. 16.9 the u_{xxx} term leads to dispersive behaviour, rather than dissipation.
4. We also observe, that the magnitude of the error is smaller for the Lax-Wendroff method, compared to up-winded scheme C. Since scheme F is a higher-order method (hence the $O(h^2)$ dispersive term, compared to the $O(h)$ dissipation term in scheme C.
5. Dispersive term gives rise to oscillating solution and also a shift in main peak location, *i.e.* phase error as observed in the numerical experiments.

16.3 Dispersion-dissipation analysis

When solving PDES analytically a common approach is to assume that the solution $u(x,t)$ has a wave form

$$u(x,t) = \hat{u}e^{i(\omega t+\alpha x)} \tag{16.10}$$

where ω is a frequency and α the wavenumber; the wavelength is given by

$$\lambda = \frac{2\pi}{\alpha}.$$

Consider the equation

$$u_t = \nu u_{xx}, \tag{16.11}$$

using Eqn. 16.10 gives

$$i\omega + \nu\alpha^2 = 0, \tag{16.12}$$

from which it is clear that Eqn. 16.10 can only be satisfied if

$$\omega = i\nu\alpha^2.$$

We call Eqn. 16.12 the **dispersion relationship**. The solution to Eqn. 16.11 thus becomes

$$u(x,t) = \hat{u}e^{-\nu\alpha^2 t}\, e^{i\alpha x}. \tag{16.13}$$

This indicates that the wave does not move and decays with time. Likewise for the linear advection equation

$$u_2 + cu_x = 0, \quad \text{with} \quad c \quad \text{a constant value.} \tag{16.14}$$

We get the dispersion relationship

$$\omega = -c\alpha,$$

and hence

$$u(x,t) = \hat{u}e^{i\alpha(x-ct)}. \tag{16.15}$$

In this case, assuming ω was real, the solution propagates with speed c and with no decay in amplitude. Also note that the speed of propagation is independent of the frequency ω.

The decay (or growth) and propagation of Fourier modes is an important aspect in the behaviour of solutions to PDEs. With finite difference (F-D) discretisation of the PDE an aspect to consider is upon obtaining the discretised solution, is how well do they decay and the F-D derived solution propagation characteristics match those of the original PDE. The numerical scheme will be unstable if some of the Fourier modes grow without bound. In what follows, **dissipation** in solutions of PDEs is when the Fourier modes do not grow with time and at least one mode decays. A PDE is **non-dissipative** if they neither grow nor decay, while we define **dispersion** when solutions of PDEs exhibit differing Fourier mode wavelengths propagating at different speeds. In this respect, observe that Eqn. 16.13 is dissipative for $\nu > 0$ for all wave-numbers $\alpha \neq 0$, while Eqn. 16.15 is neither dissipative nor dispersive.

Next consider

$$u_t + cu_{xxx} = 0, \tag{16.16}$$

for which the dispersion relationship is

$$\omega = \alpha^3 c, \tag{16.17}$$

and hence the Fourier mode representing a solution to Eqn. 16.16 is

$$u(x,t) = \hat{u}\ e^{i\alpha(x+\alpha^2 ct)}. \tag{16.18}$$

There are two aspects here:

1. Unlike Eqn. 16.15 the Fourier mode propagates in the opposite direction with speed $\alpha^2 c$;
2. Fourier modes with different wave-numbers α propagate with different speeds $\alpha^2 c$.

So Eqn. 16.16 is dispersive but non-dissipative: *i.e.* the amplitude neither decays nor grows (with either space or time). With some thought, one should be able to see that PDEs containing even ordered x-derivatives will be dissipative. PDEs containing only odd derivatives in x will be non-dissipative and involve propagating waves, and be dispersive when the order is greater than one.

Finally consider the equation

$$u_t + au_x - \nu u_{xx} + cu_{xxx} = 0, \tag{16.19}$$

for which the dispersion relationship is

$$\omega = -a\alpha + i\nu\alpha^2 + c\alpha^3, \tag{16.20}$$

and hence the solution for the α-mode will be

$$u(x,t) = \hat{u}e^{-\nu\alpha^2 t}\ e^{i\alpha\left[x-(a-c\alpha^2)t\right]}. \tag{16.21}$$

The dissipation term is $e^{-\nu\alpha^2 t}$, while the propagating term is

$$e^{i\alpha\left[x-(a-c\alpha^2)t\right]},$$

thus the PDE is dissipative and dispersive due to the dependence of the term $(a - c\alpha^2)$ term on α^2, and becomes larger the more larger α is.

16.4 Dispersion and dissipation in discretised equations

The discrete analogue of the Fourier mode Eqn. 16.10 is

$$u_k^n = \hat{u} e^{i(n\omega\ \Delta t + \alpha k\ \Delta x)}, \tag{16.22}$$

and thus for information about dispersion and dissipation for all wavelengths we consider $\alpha\ \Delta x$ in the range $0 \leq \alpha\ \Delta x \leq \pi$. Furthermore for what follows, the dispersion relation $\omega = \omega(\alpha)$ will in general be complex, hence we set

$$\omega = a + i\ b,$$

with (a, b) considered real only. Substituting $\omega = a + ib$ into Eqn. 16.22 and a rewrite gives

$$u_k^n = \hat{u} e^{-bn\ \Delta t}\ e^{i(na\ \Delta t + k\alpha\ \Delta x)} = \hat{u}(e^{-b\ \Delta t})^n\ e^{i\alpha(k\ \Delta x - (-a/\alpha)n\ \Delta t)}. \tag{16.23}$$

From this we see that

- If $b > 0$ for some α, the difference equation is dissipative.
- If $b < 0$ for some α, the solution grows without bound, and thus the scheme will be unstable.
- If $b = 0$ for all α, the scheme will be non-dissipative.

Furthermore

- If $a = 0$ for all α, there is no wave propagation.
- If $a \neq 0$ for some α, wave propagation with velocity $-a/\alpha$ occurs.
- If a/α is a non-trivial function of α, dispersive effects will arise.

Let us consider the linear advection Eqn. 16.5 (with c positive) discretised with an up-winded scheme, namely :

$$U_j^{n+1} = U_j^n - q(U_j^n - U_{j-1}^n), \tag{16.24}$$

where $q = ck/h$. Substitution of (16.22) into the above and simplification gives

$$e^{i\omega k} = 1 - q + q\left[\cos(\alpha h) - i\sin(\alpha h)\right]. \tag{16.25}$$

We next write $\omega = a + ib$ and hence

$$e^{iak}e^{-bk} = 1 - q + q\left[\cos(\alpha h) - i\sin(\alpha h)\right]. \tag{16.26}$$

Which thus gives the relative dissipation error $\mathcal{E}_D$

$$\mathcal{E}_D = e^{-bk} = \sqrt{(1-q)^2 + q^2 + 2q(1-q)\cos(\alpha h)}. \tag{16.27}$$

For stability we require that $b > 0$ for some αh. Setting $\alpha h = 0$ we get that

$$e^{-bk} = 1,$$

that the modes neither decay nor grow, while $\alpha \neq 0$, all modes decay for $|q| < 1$. Setting $q = 1$ thus gives $b = 0$, which implies that the scheme will be non-dissipative for all α.

Next setting $q = 1 - \epsilon$, with $\epsilon \ll 1$, it can be shown that

$$e^{-bk} = 1 - 2\epsilon \sin^2(\alpha h/2) + \frac{\epsilon^2}{2} \sin^2(\alpha h) + \dots \,, \tag{16.28}$$

which suggests the scheme will be dissipative for $q < 1$ values.

Whether the scheme is dispersive or not we consider the imaginary part of Eqn. 16.26, namely

$$e^{iak} = \cos(ak) + i \sin(ak) = \frac{1 - q + q\,[\cos(ah) - i \sin(ah)]}{|(1 - q + q\,[\cos(ah) - i \sin(ah)])|}, \tag{16.29}$$

hence

$$\tan(ak) = \frac{-q \sin(\alpha h)}{1 - q + q \cos(\alpha h)},$$

i.e. hence

$$a = -\frac{1}{k} \tan^{-1} \left(\frac{q \sin(\alpha h)}{1 - q + q \cos(\alpha h)} \right) = -\frac{1}{k} \tan^{-1} \left(\frac{q \sin(\alpha h)}{1 - 2q \sin^2(\alpha h/2)} \right). \tag{16.30}$$

This is in general dispersive since a is a nonlinear function of α. This expression then requires a careful examination to ascertain behaviour of the low frequency (αh small) and high frequency (αh near π) waves. For $\alpha h \ll 1$ (small), we obtain

$$ak = -q\alpha h + O(\alpha h)^3. \tag{16.31}$$

Thus dispersive effects should be expected for the low frequency modes. The high frequency modes require a similar but separate analysis, namely for $\alpha h \leq \pi$ limit.

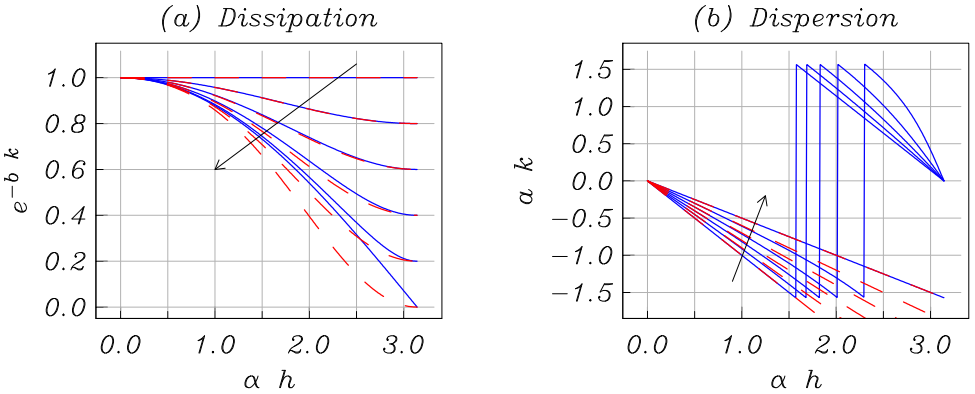

Figure 16.2: Dissipation and dispersive effects arising in Eqn. (16.24) as q and αh vary; $q = (1,\ 0.9,\ 0.8,\ 0.7,\ 0.6,\ 0.5)$ (arrow in direction of decreasing q) (a) Dissipation given by Eqn. 16.27; (b) Dispersive behaviour given by Eqn. 16.30. The approximate results Eqn. (16.28) and Eqn. (16.31) are shown by the red dashed lines.

The relative dispersion error $\mathcal{E}_p$ is thus

$$\mathcal{E}_p = \left[\frac{\tan^{-1} \left(\frac{-q \sin(\alpha h)}{1 - 2q \sin^2(\alpha h/2)} \right)}{-q\alpha h} \right],$$

Further details see, chapter 7 of Thomas, J. W. "Numerical Partial Differential Equations – Finite Difference Methods"

17 Conservation Form and Navier-Stokes Equations

It is often the case, when dealing with hyperbolic equations, that they can be formulated through conservation laws stating that a given quantity m (say, mass) is transported in space and time and is thus locally conserved. The resulting "*law of continuity*" leads to equations which are called **conservative**.

In general, an equation of the form

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{b} = Q$$

can be interpreted as a **conservation law** for a physical quantity, in above the density ρ, for which Q is a source and $\mathbf{b}$ is its **flux** or sometimes referred to as "**conserved flux**". Over any fixed volume V bounded by the surface ∂V, normal $\widehat{n}$,

$$\frac{d}{dt}\int_V \rho \, dV = \int_V Q \, dV - \oint_{\partial V} (\mathbf{b} \cdot \widehat{n}) \, dS \, ,$$

so that the rate of increase of the physical quantity (note, mass = density $\times$ volume) in V is equal to the total rate of generation within V less the amount that flows out of V across the boundary, note Fig. 17.1. In other words, a conservation law states that without **sources** or **sinks**, the rate of change of ρ in the interior volumetric space is equal to the net flux through the boundary Γ.

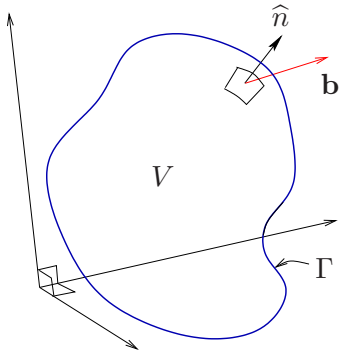

Figure 17.1: Divergence theorem.

This result, recall follows from Gauss's Divergence theorem

$$\iiint \nabla \cdot \mathbf{b} \, dV = \iint_\Gamma (\mathbf{b} \cdot \widehat{n}) \, dS. \tag{17.1}$$

In a nutshell, the conserved variable ρ (say) is transported in space and time and is thus locally conserved.

17.1 Examples of conservation forms

1. Scalar conservation form:

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x} f(u) = 0, \quad f(u) \; : \quad \text{flux function.} \tag{17.2}$$

   A simple example is the linear advection equation, where $f(u) = cu.$ with c a constant.

2. Inviscid Burger's equation has flux

$$f(u) = \frac{1}{2}u^2.$$

3. Traffic flow equation,

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x} f(\rho) = 0, \quad f(\rho) = u_m(1 - \frac{\rho}{\rho_m})\rho$$

Here the conserved variable is $\rho(x,t)$ representing density of vehicles, with $(\rho_m,\ u_m)$ parameters representing maximum density and speed.

4. In systems of conservation laws, namely

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{f}[\mathbf{u}]}{\partial x} = 0.$$

where trivially one may have $\mathbf{f}(\mathbf{u}) = \mathbf{A}\mathbf{u}$ with $\mathbf{A}$ the Jacobian matrix or in more complicated cases,

$$\frac{\partial \mathbf{f}[\mathbf{u}]}{\partial x} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \frac{\partial \mathbf{u}}{\partial x} \quad \text{with} \quad \mathbf{A} = \frac{\partial \mathbf{f}}{\partial \mathbf{u}},$$

i.e. Eqn 17.9 or Eqn. 19.6 (see later).

Example

Consider the following system governing *isothermal gas dynamics*

$$\frac{\partial \mathbf{u}}{\partial t} = \frac{\partial}{\partial t}\begin{bmatrix} \rho \\ \rho u \end{bmatrix}; \quad \frac{\partial \mathbf{f}[\mathbf{u}]}{\partial x} = \frac{\partial}{\partial x}\begin{bmatrix} \rho u \\ \rho u^2 + a^2\rho \end{bmatrix}, \tag{17.3}$$

with a a positive constant valued speed of sound. We next define the conserved variables vectorially, namely $\rho = u_1$ and $\rho u = u_2$ and the above becomes

$$\frac{\partial}{\partial t}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \frac{\partial}{\partial x}\begin{bmatrix} u_2 \\ u_2^2/u_1 + a^2 u_1 \end{bmatrix} = 0\,. \tag{17.4}$$

The corresponding Jacobian matrix is

$$\mathbf{A}(\mathbf{U}) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}} = \begin{bmatrix} 0 & 1 \\ -(u_2/u_1)^2 + a^2 & 2u_2/u_1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -u^2 + a^2 & 2u \end{bmatrix}. \tag{17.5}$$

It is a trivial matter to show that the eigenvalues of $\mathbf{A}$ are

$$\lambda_1 = u - a \quad \text{and} \quad \lambda_2 = u + a.$$

17.2 Compressible viscous Navier-Stokes equations (NSE)

In the previous section we studied in detail the behaviour and the general solution of the simplest PDE of hyperbolic type, namely the linear advection $u_t + cu_x = 0$, with constant wave propagation speed c. We next consider more complex systems of coupled PDEs. A typical example is the system of Navier-Stokes Equations (NSEs).

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0, \tag{17.6a}$$

$$\rho\left(\frac{\partial u_j}{\partial t} + u_j \frac{\partial u_i}{\partial x_i}\right) = \nabla \cdot \Pi_{ij}, \tag{17.6b}$$

$$\rho\left(\frac{\partial h}{\partial t} + u_i \frac{\partial h}{\partial x_i}\right) - \left(\frac{\partial p}{\partial t} + u_i \frac{\partial p}{\partial x_i}\right) = \nabla \cdot (\kappa \nabla T) + \Phi, \tag{17.6c}$$

Where ρ is the mass density, u_i is the velocity vector, Π_{ij} is the stress tensor, h is the specific heat enthalpy, and $q = -\kappa \nabla T$ approximated by the Fourier law is the heat flux. Indices i, j equal $1, 2, 3$, with x_i spatial coordinates (x, y, z) say in three-dimensions and t a time variable, and repeated indices are summed over. Eqn. 17.6a expresses conservation of mass, Eqn. 17.6b expresses conservation of momentum, and Eqn. 17.6c expresses conservation of energy in which Φ is known as the viscous dissipation.

The stress tensor (Π_{ij}) and the specific enthalpy (h) are

$$\Pi_{ij} = -nk_B T \delta_{ij} - \frac{2}{3}\mu \frac{\partial u_k}{\partial x_k}\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{17.7}$$

$$h = C_p T \tag{17.8}$$

Here the density $n = \rho/m$, m is the mass, T is the temperature and C_p the specific heat capacity.

17.2.1 Inviscid conservation-law (Euler) form – Ideal gas dynamics

In three space dimensions (x, y, z) neglecting viscosity and heat conduction, formulating the NSEs in conserved variables $(\rho,\ \rho u,\ \rho v,\ \rho w,\ E)^{\mathsf{T}}$, we have five inviscid conservation laws

$$\rho_t + (\rho u)_x + (\rho v)_y + (\rho w)_z = 0\ , \tag{17.9a}$$

$$(\rho u)_t + (\rho u^2 + p)_x + (\rho uv)_y + (\rho uw)_z = 0\ , \tag{17.9b}$$

$$(\rho v)_t + (\rho uv)_x + (\rho v^2 + p)_y + (\rho vw)_z = 0\ , \tag{17.9c}$$

$$(\rho w)_t + (\rho uw)_x + (\rho vw)_y + (\rho w^2 + p)_z = 0\ , \tag{17.9d}$$

$$E_t + [u(E+p)]_x + [v(E+p)]_y + [w(E+p)]_z = 0\ . \tag{17.9e}$$

Here E is the total energy per unit volume

$$E = \rho\left(\frac{1}{2}(u^2 + v^2 + w^2) + e\right), \tag{17.10}$$

and $e = C_v T$ is the specific internal energy with C_v a specific heat at constant volume. For a *gamma law gas* the pressure P is given by the equation of state

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho u^2\right), \tag{17.11}$$

with $\gamma = C_p/C_v$. The Euler equations can be shown to be hyperbolic.

The conservation laws (17.9) can be expressed in a very compact notation by defining a column vector $\mathbf{U}$ of **conserved variables** and flux vectors $\mathbf{F}(U), \mathbf{G}(U), \mathbf{H}(U)$ in the x, y and z directions, respectively. Namely:

$$\mathbf{U}_t + \mathbf{F}(U)_x + \mathbf{G}(U)_y + \mathbf{H}(U)_z = 0. \tag{17.12}$$

It is important to note that $\mathbf{F} = \mathbf{F}(U)$, $\mathbf{G} = \mathbf{G}(U)$, $\mathbf{H} = \mathbf{H}(U)$; that is, the flux vectors are functions of the conserved variable vector $\mathbf{U}$. Any set of PDEs written in this form is called a system of *conservation* laws. As partial derivatives are involved we call them a system of conservation laws in differential form. These may thus also be written in a much more compact form as follows :

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{i=1}^{3} \frac{\partial \mathbf{E}_i}{\partial x_i}, \tag{17.13}$$

Conservation Laws : To illustrate concepts, we next consider systems of 1D-PDE conservation laws of the form

$$\mathbf{U}_t + \mathbf{F}(U)_x = 0, \tag{17.14}$$

where

$$\mathbf{U} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \quad \mathbf{F}(U) = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}. \tag{17.15}$$

$\mathbf{U}$ is the vector of conserved variables, $\mathbf{F} = \mathbf{F}(U)$ is the vector of fluxes and each of its components f_i is a function of the components u_j of $\mathbf{U}$.

Jacobian Matrix : The Jacobian of the flux function $\mathbf{F}(U)$ in Eqn. 17.14 is the matrix

$$\mathbf{A}(U) = \frac{\partial F}{\partial U} = \begin{bmatrix} \partial f_1/\partial u_1 & \dots & f_1/\partial u_m \\ \partial f_2/\partial u_1 & \dots & f_2/\partial u_m \\ \vdots & \vdots & \vdots \\ \partial f_m/\partial u_1 & \dots & f_m/\partial u_m \end{bmatrix}. \tag{17.16}$$

The entries a_{ij} of $\mathbf{A}(U)$ are partial derivatives of the components f_i of the vector F with respect to the components u_j of the vector of conserved variables U, that is $a_{ij} = \partial f_i/\partial u_j$. Now since we can write

$$\frac{\partial \mathbf{F}(U)}{\partial x} = \frac{\partial F}{\partial U}\frac{\partial U}{\partial x}, \tag{17.17}$$

Eqn. 17.14 becomes

$$U_t + A(U)U_x = 0. \tag{17.18}$$

Eigenvalues : The eigenvalues λ_i of a matrix A are the solutions of the characteristic polynomial

$$\|A - \lambda I\| = \det(A - \lambda I) = 0, \tag{17.19}$$

where I is the identity matrix. The eigenvalues of the coefficient matrix A of a system of Eqn. 17.14 are called the eigenvalues of the system. Physically, eigenvalues represent

speeds of propagation of information. Speeds will be measured positive in the direction of increasing x and negative otherwise.

Hyperbolic System : A system of Eqn. 17.14 is said to be hyperbolic at a point (x,t) if A has m real eigenvalues $1,\ldots,m$ and a corresponding set of m linearly independent right eigenvectors $K(1),\ldots,K(m)$. The system is said to be strictly hyperbolic if the eigenvalues λ_i are all distinct. Note that strict hyperbolicity implies hyperbolicity, because real and distinct eigenvalues ensure the existence of a set of linearly independent eigenvectors. The Eqn. 17.14 is said to be elliptic at a point (x,t) if none of the eigenvalues λ_i of A are real.

17.3 Diagonalisation and characteristic variables

In order to analyse and solve the general IVP for Eqn. 17.14 it is found useful to transform the dependent variables $U(x,t)$ to a new set of dependent variables $W(x,t)$. To this end we recall the following :

Diagonalisable System : A matrix A is said to be diagonalisable if A can be expressed as

$$A = K\ \Lambda\ K^{-1} \ \text{ or } \ \Lambda = K^{-1}\ A\ K\ , \tag{17.20}$$

in terms of a diagonal matrix and a matrix K. The diagonal elements of Λ are the eigenvalues λ_i of A and the columns $K^{(i)}$ of K are the right eigenvectors of A corresponding to the eigenvalues λ_i, that is

$$\Lambda = \begin{bmatrix} \lambda_1 & \ldots & 0 \\ 0 & \ldots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \ldots & \lambda_m \end{bmatrix}\ ,\ \ K = \left[K^{(1)},\ldots,[K^{(m)}\right]\ ,\ \ A\ K^{(i)} = \lambda_1\ K^{(i)} \tag{17.21}$$

A system Eqn. 17.18 is said to be diagonalisable if the coefficient matrix A is diagonalisable. Based on this concept of diagonalisation one then defines Eqn. 17.18 a hyperbolic system provided A has real eigenvalues and a diagonalisable coefficient matrix.

Characteristic variables

The existence of the inverse matrix K^{-1} makes it possible to define a new set of dependent variables $W = (w_1, w_2, \ldots, w_m)^T$ via the transformation

$$W = K^{-1}\ U \ \text{ or } \ U = K\ W\ , \tag{17.22}$$

so that the linear system Eqn. 17.18, when expressed in terms of W, becomes completely **decoupled**, in a sense to be defined. The new variables W are called characteristic variables. Next we derive the governing PDEs in terms of the characteristic variables, for which we need the partial derivatives U_t and U_x in Eqn. 17.18. Provided A is constant, K is also constant and therefore these derivatives are

$$U_t = K\ W_t\ ,\ \ U_x = K\ W_x\ . \tag{17.23}$$

Direct substitution of these expressions into Eqn. 17.18 gives

$$K\ W_t + A\ K\ W_x = 0. \tag{17.24}$$

Multiplication of this equation from the left by K^{-1} and use of Eqn. 17.20 gives

$$W_t + \Lambda W_x = 0. \tag{17.25}$$

This is called the **canonical form** or **characteristic form** of Eqn. 17.18. When written in full this system becomes

$$\frac{\partial}{\partial t}\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} + \begin{bmatrix} \lambda_1 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & \lambda_m \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}_x = 0 \tag{17.26}$$

Clearly the $i-$th PDE of this system is

$$\frac{\partial w_i}{\partial t} + \lambda_i \frac{\partial w_i}{\partial x} = 0, \quad i = 1, \dots, m \tag{17.27}$$

and involves the single unknown $w_i(x,t)$; the system is therefore decoupled and is identical to the linear advection equation; the characteristic speed is λ_i and there are now m characteristic curves satisfying m ODEs

$$\frac{dx}{dt} = \lambda_i \ , \quad i = 1, \dots, m. \tag{17.28}$$

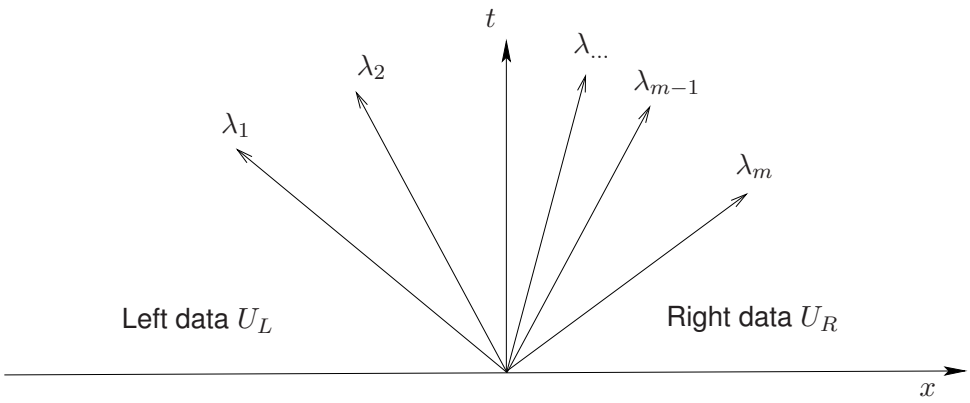

Figure 17.2: Characteristics of a linear hyperbolic system of m equations with constant coefficients.

The initial value problem

Given the initial condition,

$$\mathbf{U}^{(o)} = (u_1^{(o)}, \dots, u_m^{(o)})^{\mathsf{T}},$$

in canonical variables the initial conditions become

$$\mathbf{W}^{(o)} = K^{-1}\mathbf{U}^{(o)} \ \text{ or } \ \mathbf{U}^{(o)} = K\mathbf{W}^{(o)}.$$

Hence, each unknown $w_i(x,t)$ with corresponding initial data $w_i^{(o)}$ will be

$$w_i(x,t) = w_i^{(o)}(x - \lambda_i t), \quad i = 1, \dots, m \ . \tag{17.29}$$

Now, since

$$\mathbf{U}(x,t) = \sum_{i=1}^{m} w_i(x,t)\ \mathbf{K}^i,$$

and given Eqn. 17.29, it follows

$$\mathbf{U}(x,t) = \sum_{i=1}^{m} w_i^{(o)}(x-\lambda_i t)\ \mathbf{K}^i\ . \tag{17.30}$$

Thus the solution $\mathbf{U}(x,t)$ depends only on the initial data at the m points

$$x_o^{(i)} = x - \lambda_i t.$$

The solution 17.30 is clearly a superposition of the m waves. each of which is convected independently without change in shape; the i-th wave has shape $w_i^{(o)}(x)\ \mathbf{K}^i$ and propagates with speed λ_i.

Example

linearised equations of Gas Dynamics, see Toro (1997).

Consider the system

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + \begin{bmatrix} 0 & \rho_o \\ a^2/\rho_o & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_x = 0\ ,\quad u_1 \equiv \rho\ ,\quad u_2 \equiv u\ , \tag{17.31}$$

where ρ_o is a constant and a^2 the speed of sound (constant too). Written in matrix form this becomes

$$U_t + A\,U_x = 0. \tag{17.32}$$

The eigenvalues of A are thus given by

$$\|A-\lambda I\| = \det(A-\lambda I) = \det \begin{bmatrix} 0-\lambda & \rho_o \\ a^2/\rho_o & 0-\lambda \end{bmatrix} = 0\ . \tag{17.33}$$

There are thus two real distinct values $\lambda_1 = -a$ and $\lambda_1 = +a$. The right eigenvectors $K^{(1)}$, $K^{(2)}$ are thus

$$K^{(1)} = \begin{bmatrix} \rho_o \\ -a \end{bmatrix}\ ,\quad K^{(2)} = \begin{bmatrix} \rho_o \\ a \end{bmatrix}. \tag{17.34}$$

Thus the K matrix of right eigenvectors and its inverse K^{-1} may be stated as

$$K = \begin{bmatrix} \rho_o & \rho_o \\ -a & a \end{bmatrix}\ ,\quad K^{-1} = \frac{1}{2a\rho_o} \begin{bmatrix} a & -\rho_o \\ a & \rho_o \end{bmatrix}\ . \tag{17.35}$$

Hence in terms of characteristic variables Eqn. 17.25, we may write

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t + \begin{bmatrix} -a & 0 \\ 0 & a \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_x = 0\ , \tag{17.36}$$

or in full form

$$\frac{\partial w_1}{\partial t} - a\frac{\partial w_1}{\partial x} = 0\ ,\quad \frac{\partial w_2}{\partial t} + a\frac{\partial w_2}{\partial x} = 0\ . \tag{17.37}$$

Given some initial condition on $U(x,0)$

$$\begin{bmatrix} w_1^{(o)} \\ w_2^{(o)} \end{bmatrix} = K^{-1} \begin{bmatrix} u_1^{(o)} \\ u_2^{(o)} \end{bmatrix} \tag{17.38}$$

written in full form

$$\begin{aligned} w_1^{(o)}(x) &= \tfrac{1}{2a\rho_o}\left[a u_1^{(o)}(x) - \rho_o u_2^{(o)}(x)\right] , \\ w_2^{(o)}(x) &= \tfrac{1}{2a\rho_o}\left[a u_1^{(o)}(x) + \rho_o u_2^{(o)}(x)\right] . \end{aligned} \tag{17.39}$$

Hence, since the solution to $(w_1,\ w_2)$, on noting Eqn. 17.37, is

$$w_1 = w_1^{(o)}(x+at) , \quad w_2 = w_2^{(o)}(x-at) , \tag{17.40}$$

it follows that the solution in characteristic variables is

$$\begin{aligned} w_1(x,t) &= \tfrac{1}{2a\rho_o}\left[a u_1^{(o)}(x+at) - \rho_o u_2^{(o)}(x+at)\right] , \\ w_2(x,t) &= \tfrac{1}{2a\rho_o}\left[a u_1^{(o)}(x-at) + \rho_o u_2^{(o)}(x-at)\right] . \end{aligned} \tag{17.41}$$

We may then transform the solution back to the original problem using $U = K\ W$ (see Toro, 1997).

17.4 Integral forms of conservation laws

Conservation laws may be expressed in differential and integral form. There are two reasons for considering the integral form(s) of the conservation laws and seeking solutions of PDEs in "*conserved variable*" form :

1. The derivation of the governing equations is based on physical conservation principles expressed as integral relations on control volumes.
2. The integral formulation requires less smoothness of the solution, which allows extending the class of admissible solutions to include discontinuous solutions, *i.e.* **shocks**.

Mathematically we allow for **weak solutions** of the PDE : A "*weak solution*" is where the derivatives may not all exist but which is nonetheless deemed to satisfy the PDE in some sense.

A **non-conservative** variables based method, might give a numerical solution which appears perfectly reasonable but then upon closer inspection of results, one finds inconsistencies. These methods fail at shock-waves, giving incorrect values of the shock strength, shock speed and hence shock location.

In FD-methods examined so far, we have computed the solution at individual discrete points. Recall, FD-discretisation formulae are based on the assumption of functions that are smooth, and these will fail when applied to discontinuous functions.

Consider the conservation of mass equation, in one space dimension

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = 0. \tag{17.42}$$

This may be written, with reference to Fig. 17.3, as

$$\frac{d}{dt} \int_{x_L}^{x_R} \rho(x,t)\, dx = f(x_L,t) - f(x_R,t)\ , \tag{17.43}$$

where $f = \rho u$ is the **flux function**. We next define

$$Q(t) = \int_{x_L}^{x_R} \rho(x,t)\, dx,$$

and $Q(t)$ is the *conserved* quantity in $[x_L, x_R]$, so that $Q(t)$ only changes in time t when there is a net inflow or outflow through the **control volume** boundaries.

Thus for the complete equation we have **the first integral form I** of a conservation law :

$$\frac{d}{dt} Q(t) = F\left(U(x_L,t)\right) - F\left(U(x_R,t)\right), \tag{17.44}$$

where $F(U)$ is the flux vector.

Next integrating Eqn. 17.44 over t,

$$\int_{t_1}^{t_2} \frac{d}{dt} Q(t)\, dt = \int_{t_1}^{t_2} \left[F\left(U(x_L,t)\right) - F\left(U(x_R,t)\right)\right]\, dt, \tag{17.45}$$

the **second integral conservation law II** follows

$$Q(t_2) = Q(t_1) + \int_{t_1}^{t_2} \left[F\left(U(x_L,t)\right) - F\left(U(x_R,t)\right)\right]\, dt. \tag{17.46}$$

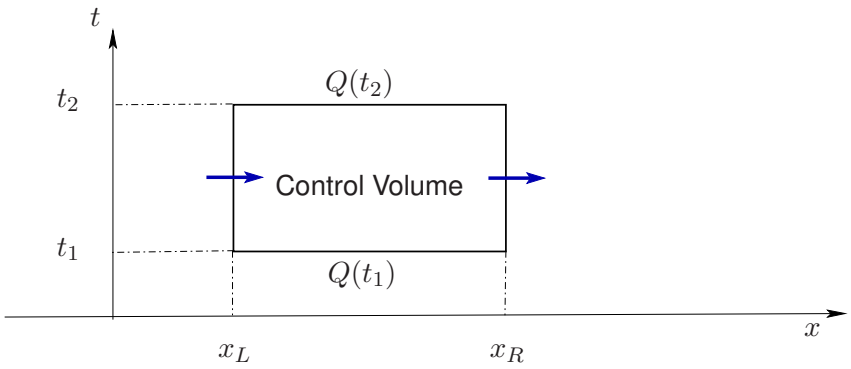

Figure 17.3: Control volume in the x, t-plane.

17.5 Finite volume conservation methods

In what follows, we divide the spatial domain into **cells of finite volume** as indicated in Fig. 17.4. We next use Eqn. 17.46 for the discrete Cell-i

$$Q_i^{n+1} - Q_i^n = \int_{t_n}^{t_{n+1}} \left[F\left(U(x_{i-1/2}, t)\right) - F\left(U(x_{i+1/2}, t)\right) \right] \, dt, \tag{17.47}$$

here we have discretised

$$Q_i^n = \int_{x_{i-1/2}}^{x_{i+1/2}} U(x, t_n) \, dx.$$

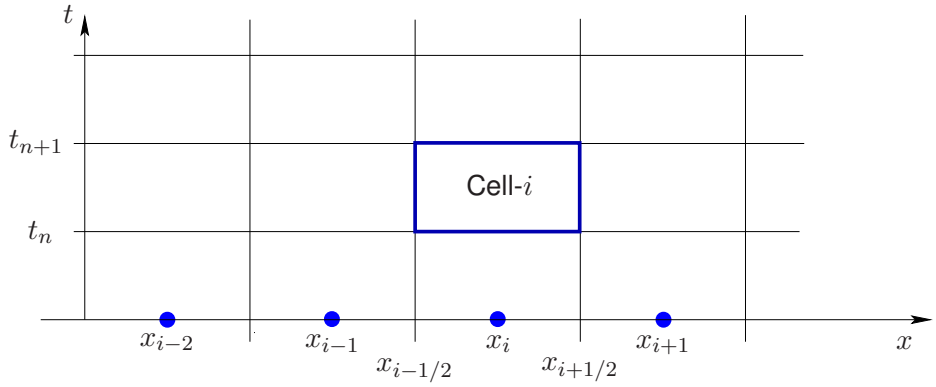

Figure 17.4: Control volume in the x, t-plane.

We can thus define

$$U_i^n = \frac{Q_i^n}{\Delta x}$$

and

$$F_{i+1/2}^{n+1/2} = \frac{1}{\Delta t} \left[\int_{t_n}^{t_{n+1}} F(U(x_{i+1/2}, t)) \, dt \right],$$

the average values of U and $F(U)$ at the interface $i + 1/2$ between t_{n+1} and t_n. Hence Eqn. 17.47 is re-written

$$\frac{U_i^{n+1} - U_i^n}{\Delta t} + \frac{F_{i+1/2}^{n+1/2} - F_{i-1/2}^{n+1/2}}{\Delta x} = 0. \tag{17.48}$$

Note, this gives an exact formula for updating the cell average of U_i^{n+1}. On removing the superscript $(n+1/2)$ notation, summarised, we have a conservative scheme for the scalar conservation law Eqn. 17.14 or Eqn. 17.42

$$U_i^{n+1} = U_i^n + \frac{\Delta t}{\Delta x}\left[F_{i-1/2} - F_{i+1/2}\right] \ , \tag{17.49}$$

where

$$F_{i+1/2} = F_{i+1/2}\left(U_{i-l_L}, \ldots, U_{i+l_R}^n\right), \tag{17.50}$$

with l_L, l_R two non-negative integers; $F_{i+1/2}$ is called the *numerical flux function*, an approximation to the physical flux $F(U)$ in Eqn. 17.14.

Observe that Eqn. 17.49 or for that matter Eqn. 17.46 is identically equivalent to the expression

$$\oint \left[U\, dx - F(U)\, dt\right] = 0, \tag{17.51}$$

which follows from integrating Eqn. 17.14 in a domain V in $x-t$ space and using Green's theorem. The line integration along the boundary of the domain is undertaken in an anticlockwise manner. This is called **integral form III** of the conservation laws, with the integral form II a special case where the control volume is the rectangle $[x_L, x_R] \times [t_1, t_2]$.

For greater generality, we rewrite

$$F_{i+1/2} = f^*(U_i, U_{i+1}) \quad \text{and} \quad F_{i-1/2} = f^*(U_i, U_{i-1}).$$

These expressions simply state that the flux through the interface $(i+1/2)$ can be calculated based on the state of the system in cells i and $i+1$; cells $i,\ i-1$ for the flux evaluation at interface $(i-1/2)$. In general, the time integrals on the right hand side cannot be evaluated exactly, but highly accurate numerical methods are possible, all invoking the general form

$$U_i^{n+1} = U_i^n + \frac{\Delta t}{\Delta x}\left[f^*(U_i, U_{i-1}) - f^*(U_i, U_{i+1})\right]. \tag{17.52}$$

The choice of how the flux functions $f^*(U_i, U_{i-1})$ and $f^*(U_i, U_{i+1})$ are constructed thus distinguishes between different FV methods that arise in the literature. For example in higher accuracy order discretisations, we may use more points to discretise the flux function, *i.e.*

$$F_{i+1/2} = f^*(U_i, U_{i+1}, U_{i+2}) \quad \text{or even} \quad F_{i+1/2} = f^*(U_{i-1}, U_i, U_{i+1}, U_{i+2}, \ldots, U_{i+p}).$$

For any particular choice of numerical flux $F_{i+1/2}$, *it can be shown* that a corresponding conservative scheme arises. A fundamental requirement on the numerical flux is the consistency condition

$$F_{i+1/2}(v, \ldots, v) = F(v)\ . \tag{17.53}$$

This means that if all arguments in Eqn. (17.50) are equal to v then the numerical flux is identical to the physical flux at $u = v$.

17.6 Shocks : Discontinuous solutions

We use the nonlinear inviscid burgers equations (Eqn. 17.2) to draw out the essential elements, re-stated here as an initial value problem

$$\left.\begin{array}{l} \dfrac{\partial u}{\partial t} + \dfrac{\partial f(u)}{\partial x} = 0, \quad f(u) = \dfrac{u^2}{2} \ : \text{ the flux function} \\ u(x,0) = u_o(x). \end{array}\right\} \ \text{IVP} \tag{17.54}$$

The characteristic speed

$$\lambda(u) = \frac{df}{du} = u,$$

and thus depends on the solution to u the conserved variable. As should be obvious, the flux function $f(u)$ plays a pivotal role in the solution, with the property of **monotonicity** of $\lambda(u)$ being important. We need to address three possibilities

1. $\lambda(u)$ is a monotone increasing function of u,

$$\frac{d\lambda(u)}{du} = f'' > 0 \quad \text{convex flux}$$

2. $\lambda(u)$ is a monotone decreasing function of u,

$$\frac{d\lambda(u)}{du} = f'' < 0 \quad \text{concave flux}$$

3. $\lambda(u)$ has extrema, for some u,

$$\frac{d\lambda(u)}{du} = f'' = 0.$$

For Burger's equation 17.54, as $\lambda'(u) = 1$, the flux is convex.

Now the characteristic curves satisfying the IVP is given by

$$\frac{dx}{dt} = \lambda(u), \quad \text{given that } x(0) = x_o,$$

while the PDE reduces to

$$\frac{du}{dt} = 0 \quad \text{along this characteristic path.}$$

It follows the value of u will be given by

$$u(x,t) = u_o(x_o) = u_o(x - \lambda[u_o(x_o)]\, t) \tag{17.55}$$

as x varies with t according to

$$x_o = x - \lambda[u_o(x_o)]\, t. \tag{17.56}$$

If u_o is constant valued, then clearly all characteristics at every x position will be oriented in an identical direction, as was the case with the simpler linear advection equation (see §15.2 and Fig. 15.2). In situations where $u_o = u_o(x)$, clearly a possibility arises of characteristics intersecting, even if smooth. Under such a scenario, one can envisage a region in the $x - t$ plane within which a family of characteristics will be intersecting – such regions are generally referred to as "*compressive*". Conversely, if the characteristic curves "*fan out*", such regions are referred to as "*expansive*". These features are illustrated in Fig. 17.5.

When $\lambda'(u) > 0$, the wave speed $\lambda(u)$ is an increasing function of u, higher values of u propagate faster than lower ones. At any compressive part of the wave, the wave speed $\lambda(u)$ is a decreasing function of x. The crest of the wave moves faster and the wave profile distorts to produce a multivalued solution for $u(x,t)$, and hence, it ultimately breaks. This progression is shown in Fig. 17.6.

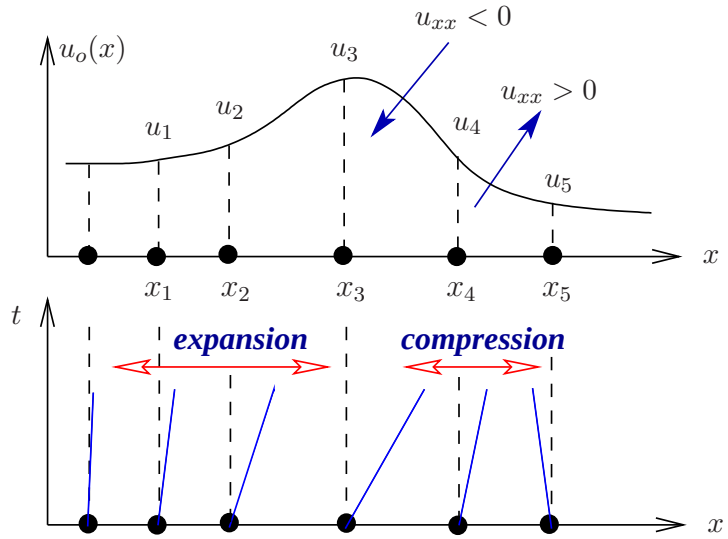

Figure 17.5: Wave steepening

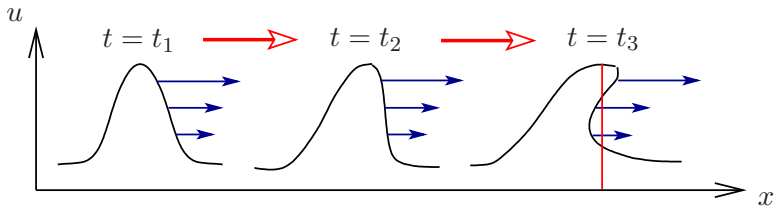

Figure 17.6: Wave profile distortion with increasing time, $t_1 < t_2 < t_3$.

17.6.1 Shock formation time

Formation of a shock arises when a family of characteristics intersect. The shock formation time is determined by evaluating the derivatives of the characteristic solutions given by Eqn. 17.55 and 17.56. Now

$$\frac{\partial u}{\partial t} = u_o' \frac{\partial x_o}{\partial t}; \quad \frac{\partial u}{\partial x} = u_o' \frac{\partial x_o}{\partial x}.$$

It follows from Eqn. 17.56

$$\frac{\partial x_o}{\partial t} = -\lambda - \lambda' u_o' \frac{\partial x_o}{\partial t}\, t, \;\; \Rightarrow \;\; \frac{\partial x_o}{\partial t} = \frac{-\lambda}{1 + \lambda' u_o' t}$$

while

$$\frac{\partial x_o}{\partial x} = 1 - \lambda' u_o' \frac{\partial x_o}{\partial x}\, t, \;\; \Rightarrow \;\; \frac{\partial x_o}{\partial x} = \frac{1}{1 + \lambda' u_o' t}.$$

These expressions thus satisfy our PDE: $u_t + \lambda u_x = 0$, **but** observe the denominator in our expressions has the possibility of being zero, *i.e.*

$$\Phi = 1 + \lambda' u_o' t.$$

For a time t for which $\Phi = 0$ we see that both $(\partial u/\partial x,\ \partial u/\partial t)$ tend to infinity – in which case a discontinuity (a shock) develops. This wave "**breaking time**" is given by

$$t_b = -\frac{1}{\lambda' u_o'}.$$

However, a smooth solution exists for all times where $\Phi \neq 0$, provided both $(\lambda',\ u_o')$ have the same signs.

17.6.2 Rankine-Hugoniot conditions and shock speed

Singularities in mathematical solutions, generally imply some weakness or shortcoming in the mathematical model of the physical phenomenon that one may be attempting to replicate. In gas dynamics, shock waves do arise, but are very narrow, *almost infinitesimally thin* regions across which very rapid **but smooth** changes occur in quantities such as pressure, temperature, density etc. over length-scales approaching the mean-free path of molecules. Thus generally, replacing the shocks waves with mathematical discontinuities is a reasonable approximation.

Consider the integral form of the conservation law (see Eqn. 17.42)

$$\frac{d}{dt}\int_{x_L}^{X_R} u(x,t)\,dx = f(u(x_L,t) - f(u(x_R,t).$$

We assume a solution $u(x,t)$ and the flux $f(u)$ and their derivatives are smooth and continuous except on a line $s = s(t)$ on the $x-t$ plane. Across $s(t)$ a jump discontinuity is imposed, and select points x_L and x_R on either sides of $s(t)$ on the x-axis. Enforcing conservation on a control volume $[x_L : x_R]$ gives

$$\frac{d}{dt}\int_{x_L}^{s(t)} u(x,t)\,dx + \frac{d}{dt}\int_{s(t)}^{X_R} u(x,t)\,dx = f(u(x_L,t) - f(u(x_R,t),$$

Recalling Leibniz's rule for differentiating under the integral, it follows

$$f(u(x_L,t) - f(u(x_R,t) = [u(s^{(-)},t) - u(s^{(+)},t)]\frac{ds}{dt} + \frac{d}{dt}\int_{x_L}^{s(t)} u_t(x,t)\,dx + \frac{d}{dt}\int_{s(t)}^{X_R} u_t(x,t)\,dx,$$

where $s^{(-)}$ is the limit as x tends to $s(t)$ from the left, and $s^{(+)}$ is the limit as x tends to $s(t)$ from the right. Next taking the limits $x_L \to s^{(-)}$ and $x_R \to s^{(+)}$ the integral terms vanish, and we obtain

$$f(u(x_L,t) - f(u(x_R,t) = [u(s^{(-)},t) - u(s^{(+)},t)]\frac{ds}{dt},$$

or in a compact form

$$[f(u)]_+^- = S[u(x,t)]_+^-. \tag{17.57}$$

Here, we have written $S = ds/dt$ the speed of the discontinuity, while expression 17.57 is called the **Rankine-Hugoniot jump condition**. Solving for the speed, it follows

$$S = \frac{\Delta f}{\Delta u},$$

where Δf and Δu are the jump values of $(f,\ u)$ on either sides of the shock. For Burger's equation, we note that

$$S = \frac{1}{2}(u_L + u_R).$$

17.6.3 On weak solutions

The notion of a solution $u(x,t)$ being analytic or at least infinitely differentiable, generally defines a smooth function. A less strict requirement, would be to ask for a solution of a PDE of order k to be at least k times differentiable. In this case, all derivatives in the equation will exist and be continuous: this is called a "**classical solution**" of the PDE.

Shock waves represent discontinuous solutions to the PDE, and integral conservation laws allow us to to obtain solutions which are not differentiable nor continuous. These solutions are called **weak** or **generalised** solutions.

To show that a weak solution of the initial value PDE problem, is acceptable (if we loosen smoothness requirements), is by way of considering a class of test functions $\phi = \phi(x,t)$ having **compact support**. An example of a compact support, would be say a rectangular region

$$\mathcal{D} = (x,t) : \ a \le x \le b, \ \text{ and } \ 0 \le t \le T$$

in the (x,t)-plane, where $\phi = 0$ outside of $\mathcal{D}$.

Let us take Eqn. 17.54 factor it with $\phi(x,t)$ and integrate over the domain $\mathcal{D}$, namely

$$\iint_{\mathcal{D}} (u_t + f_x)\, \phi \; dx \; dt = 0.$$

Integrating by parts gives

$$\int_a^b \left([u\phi]_0^T - \int_0^T u\phi_t \; dt \right) \; dx + \int_o^T \left([f\phi]_a^b - \int_a^b f\phi_x \; dx \right) \; dt = 0,$$

hence

$$\int_a^b \phi(x,0) u_o(x) \; dx + \int_a^b \int_o^T u\phi_t \; dt \; dx + \int_a^b \int_o^T f\phi_x \; dt \; dx = 0.$$

Thus, for $t > 0$

$$\iint_{\mathcal{D}} (u\phi_t + f\phi_x) \; dx \; dt + \int \phi(x,0) u_o(x) \; dx = 0, \tag{17.58}$$

and this holds for all test functions $\phi(x,t)$, and importantly does not involve any derivatives of u or f : in which case this is valid even if u and f or their derivatives are discontinuous.

This shows or proves that, if $u(x,t)$ is a *classical* solution of problem 17.54, then Eqn. 17.58 holds for all test functions $\phi(x,t)$ with compact support in the (x,t)-plane. Hence, numerical solutions $u(x,t)$ or functions (even with discontinuities) which satisfy 17.58 are called the *weak* or *generalised* solutions of 17.54.

17.7 The Riemann problem

Next, we study an idealised IVP problem, the so-called Riemann problem, where u_L (left) and u_R (right) are constant valued states with a discontinuity at $x = 0$, as shown in Fig. 17.7.

$$\begin{aligned} \text{PDE :} \qquad & u_t + f(u)_x = 0 \\ \text{Initial condition :} \quad & u(x,0) = u_o(x) = \begin{cases} u_L & \text{if } \ x < 0 \ ; \\ u_R & \text{if } \ x > 0 \ . \end{cases} \end{aligned} \tag{17.59}$$

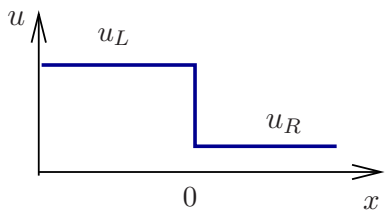

Figure 17.7: The Riemann problem

Here $u_L > u_R$, based on discussions above, it follows we expect the characteristic paths to be given by $x_L = u_L t$ and $x_R = u_R t$ on the left and right sides of the discontinuity, while where the left and right characteristics intersect a shock arises, and its path, d_S, is given by

$$d_S = \frac{1}{2}(u_L + u_R)\, t.$$

For the case, where $u_R = 0$, these key features are shown schematically in Fig. 17.8. Note the right characteristics are vertically aligned, since $dt/dx = 1/u_R$.

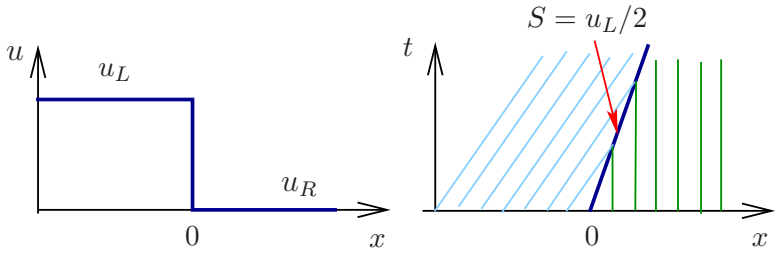

Figure 17.8: The characteristics for the shock wave

17.7.1 Entropy condition and rarefaction waves

The solution to the Riemann problem for the case, where $u_R > u_L$ and setting $u_L = 0$, namely

$$u(x,0) = \begin{cases} 0 & \text{if } x < 0 \\ u_R & \text{if } x > 0, \end{cases} \tag{17.60}$$

is shown in Fig. 17.9. In this case, the left and right characteristics do not intersect, and also a region in (x,t) space exists, into which no characteristics propagate. Amongst the many weak solutions possible, two are shown in Fig. 17.9. Both are mathematically "acceptable" weak solutions, but only the rarefaction fan solution is "physical" and thus **admissible**.

The expansion-shock solution is ruled inadmissible, through the so-called **entropy condition**[*]. The *entropy condition* is based on the requirement that characteristic curves are allowed to enter into a shock wave as in Fig. 17.8, but characteristic curves **can not emerge** from the shock. There are several variations of the entropy condition. We state only the simplest : a discontinuity propagating with speed S given by Eqn. 17.57 must satisfy the entropy condition

$$\lambda(u_L) > S > \lambda(u_R). \tag{17.61}$$

[*]to prove this is beyond the scope of this course. See Leveque *Finite Volume Methods for Hyperbolic Problems* 2002

For Burgers equation, this condition reduces to the requirement that if a discontinuity is propagating with speed S then $u_L > u_R$ – this is satisfied for the case shown in Fig. 17.8.

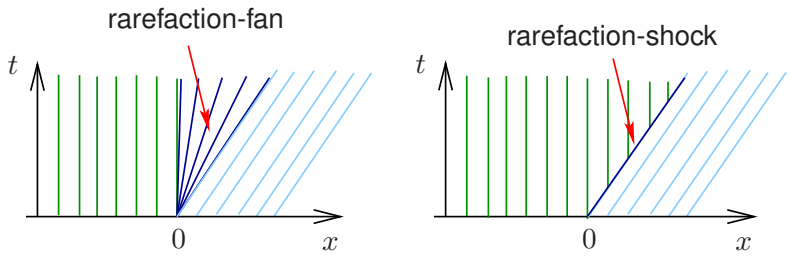

Figure 17.9: The characteristics for the rarefaction fan and the rarefaction shock (physically inadmissible).

Rarefaction shocks are called entropy-violating shocks. The connection with entropy comes from gas dynamics, and the second law of thermodynamics requiring that the entropy of a system must not decrease with time. Across a physically admissible shock the entropy of the gas increases. However across a rarefaction shock, the entropy of the gas decreases, which is inadmissible on physical grounds and therefore rejected.

For completeness, the expansion-fan solution, where $u_L = 0$, is

$$u(x,t) = \begin{cases} 0 & \text{if } x < 0, \\ x/(u_R\, t) & \text{if } 0 < x/u_R \leq t, \\ u_R & \text{if } x/u_R > t. \end{cases} \tag{17.62}$$

Expansive smooth initial data

An alternative means to see why the expansion fan is the correct solution to impose, is to consider the scenario shown in Fig. 17.10, where a linear variation in $u_o(x)$ is prescribed separating the left and right constant valued states $(u_L,\ u_R)$ by the distance $\Delta x = c - a$. In this case, we can clearly see that an expansion fan arises, which "smoothly" connects the left and right states.

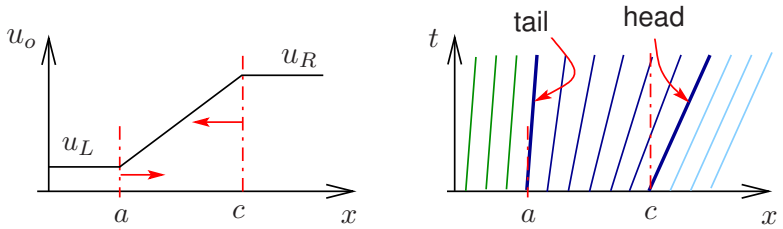

Figure 17.10: The characteristics for the rarefaction fan.

Next imagine $\Delta x \to 0$, *i.e.* points a, c coinciding. No matter how small the Δx interval is, the structure of the above expansion fan solution remains unaltered and is entirely different from the expansion-shock solution, for which small changes to the data lead to large changes in the solution – which if you recall is an undesirable property.

17.8 Riemann Solution for the inviscid Burgers equation

At an interface $x = 0$, we pose a Riemann problem :

$$u(x, t = 0) = \begin{cases} u_i^n & u_L \quad \text{if} \quad x < 0 \, ; \\ u_{i+1}^n & u_R \quad \text{if} \quad x > 0 \, . \end{cases} \tag{17.63}$$

Observe that this initial condition has no intrinsic length scale. If we rescale space by a factor and time by the same factor we get the same solution. It follows that

$$u(x, t) = u(x/t).$$

This means in particular that

$$u(x = 0, t) = u(x/t = 0) = constant,$$

hence,

$$u_{Riemann}(x = 0, t) = u(u_L, u_R), \quad \text{for} \quad t^j \leq t \leq t^{j+1}.$$

In summary, solutions of the Riemann problem for Burgers equation, are as follows

$$\begin{aligned} \text{PDE :} \qquad & u_t + uu_x = 0 \\ \text{Initial condition :} \quad & u(x, 0) = u_o(x) = \begin{cases} u_L & \text{if} \quad x < 0 \, ; \\ u_R & \text{if} \quad x > 0 \, . \end{cases} \end{aligned} \tag{17.64}$$

We have the possibility of a simple wave emanating from the origin $x = 0$, a shock wave if $u_L > u_R$, or an expansion wave when $u_L < u_R$. The solution summarised is

1. For $u_L > u_R$: the solution is a shock wave, with shock speed $S = (u_L + u_R)/2$

$$u(x, t) = \begin{cases} u_L & \text{if} \quad x - S\, t < 0 \\ u_R & \text{if} \quad x - S\, t > 0. \end{cases} \tag{17.65}$$

2. For $u_L < u_R$: the solution is a rarefaction wave

$$u(x, t) = \begin{cases} u_L & \text{if} \quad x/t \ < u_L \\ x/t & \text{if} \quad u_L < x/t < u_R \\ u_R & \text{if} \quad x/t \ > u_R. \end{cases} \tag{17.66}$$

18 Finite Volume Methods for Nonlinear Conservation Laws

In the previous section we studied in detail the behaviour and the general solution of a simple nonlinear PDE of hyperbolic type, namely Burger's equation with a flux function $f(u) = u^2/2$. The "*local*" wave propagation speed $u(x,t)$ too evolves, as the solution proceeds given some IVP. We used the PDE to illustrate the essential elements of solving nonlinear problems, and focussed on characteristics paths and were able to deduce the key solution properties to expect from any numerical method.

In this section, we summarise and put together the key steps required for a successful FV-based discretisation of Burgers equation. The approach we use is based on Godunov's* upwind method based on using **exact solutions** to the Riemann problem (RP) discussed previously. The approach was devised during Godunov's PhD and was one of the first workable FV-methods. Today it serves as the backbone for more advanced techniques that followed. This is a vast field, and we merely touch upon concepts and simplest of techniques – namely Godunov's method.

A reminder, some examples of more complex systems of coupled nonlinear PDEs would be, among others :

1. Navier-Stokes Equations Equations
2. Inviscid Euler Equations
3. Burger's Equation, traffic problem
4. Shallow water equations in geo-fluids

18.1 Godunov's first-order upwind method

Godunov's first-order upwind method is a conservative method of the form Eqn. (17.49), where the intercell numerical fluxes $f_{i+1/2}$ are computed by using solutions of local Riemann problems. A basic assumption of the method is that at a given time level n the data has a piece-wise constant distribution, where the cell average, at a fixed time $t = t^n = n\Delta t$ is defined as

$$u_i^n = \frac{1}{\Delta x}\int_{x_{i-1/2}}^{x_{i+1/2}} u(x,t^n)\,dx\,. \tag{18.1}$$

Although within cell i spatial variations of $u(x,t)$ at time $t = t^n$ arise, the integral average value u_i^n given above is constant. We shall assign that constant value at the centre of the cell, which gives rise to **cell centred conservative methods**. Computationally, we shall deal with approximations to the cell averages u_i^n, which for simplicity we shall still denote as u_i^n. The set of cell averages defines a piece-wise constant distribution of the solution at time t^n; see Fig. 18.1.

The data at time level n may be seen as pairs of constant states (u_i^n, u_{i+1}^n) separated by a discontinuity at the intercell boundary $x_{i-1/2}$. Then, locally, one can define a Riemann

*Godunov, S. K., (1959) *A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics*. Math. Sbornik, 47:271-306.

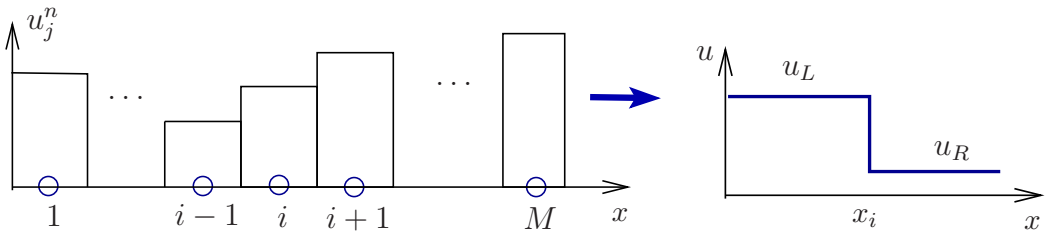

Figure 18.1: Piecewise constant reconstruction

problem

$$\begin{aligned} \text{PDE :} \qquad & u_t + f(u)_x = 0 \\ \text{Initial Condition :} \quad & u(x,0) = u_o(x) = \begin{cases} u_i^n = u_L & \text{if } x - x_i < 0\,; \\ u_{i+1}^n = u_R & \text{if } x - x_i > 0\,, \end{cases} \end{aligned} \tag{18.2}$$

where $f(u)$ is the flux function.

For consistency with the **weak form** of the PDE we must use a conservative finite volume method, namely Eqn. 17.51

$$u_i^{n+1} = u_i^n - \frac{k}{h}\left[f_{i+1/2}(u) - f_{i-1/2}(u)\right], \tag{18.3}$$

where

$$\begin{aligned} f_{i+1/2}(u) &= \tfrac{1}{k}\textstyle\int_{t^n}^{t^n+k} f(u(x_{i+1/2},t))\; dt \\ f_{i-1/2}(u) &= \tfrac{1}{k}\textstyle\int_{t^n}^{t^n+k} f(u(x_{i-1/2},t))\; dt. \end{aligned} \tag{18.4}$$

Earlier we looked at the linear advection equation and defined an **upwind** direction – we next want to define an equivalent of this to the above problem. To upwind we used a piecewise constant reconstruction, namely

$$u(x_{i-1/2} < x < x_{i+1/2}) = u_i + O(h^2).$$

At every face $x_{i+1/2}$ we need to compute a flux. If we satisfy a Courant condition, characteristics emanating from the cell faces at t^j don't reach other faces until time T^{j+1}. In this case we can focus on a single face to estimate the flux.

Suppose we could solve two separate Riemann problems

$$\text{RP}_L(u_{i-1}^n, u_i^n) \;\text{ giving }\; u_{i-1/2}(x/t)$$

solution, and

$$\text{RP}_R(u_i^n, u_{i+1}^n) \;\text{ giving }\; u_{i+1/2}(x/t).$$

It follows the overall solution would be

$$u_i^{n+1} = \frac{1}{\Delta x}\left[\int_0^{\Delta x/2} u_{i-1/2}(x/\Delta t)\; dx + \int_{-\Delta x/2}^{0} u_{i+1/2}(x/\Delta t)\; dx\right],$$

which can be re-written

$$u_i^{n+1} = \frac{1}{\Delta x}\left[\int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{u}(x,\Delta t)\; dx\right],$$

where $\tilde{u}(x, \Delta t)$ is the combined $\text{RP}_L,\ \text{RP}_R$ solutions. Now from a control volume integral viewpoint, following Eqn. 17.51, taking the line integral around the boundary, we write

$$\left.\begin{aligned}\int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{u}(x, \Delta t)\ dx = \int_{x_{i-1/2}}^{x_{i+1/2}} \tilde{u}(x, 0)\ dx + \int_0^{\Delta t} f(\tilde{u}(x_{i-1/2}, t)\ dt \\ - \int_0^{\Delta t} f(\tilde{u}(x_{i+1/2}, t)\ dt\end{aligned}\right\}. \tag{18.5}$$

Hence, through usage of Eqn. 18.1, it follows the above becomes

$$u_i^{n+1} = u_i^n + \frac{\Delta t}{\Delta x}\left[f_{i-1/2}^n - f_{i+1/2}^n\right] \tag{18.6}$$

with the intercell fluxes defined as time integral averages, namely

$$f_{i-1/2}^n = \frac{1}{\Delta t}\int_0^{\Delta t} f(\tilde{u}(x_{i-1/2}, t)\ dt, \quad f_{i+1/2}^n = \frac{1}{\Delta t}\int_0^{\Delta t} f(\tilde{u}(x_{i+1/2}, t)\ dt\ .$$

These are time integral averages of the physical flux $f(u)$ evaluated at the intercell boundaries. The integrand $f(\tilde{u}(x, t))$at each cell interface depends on the exact solution of the Riemann problem along the t-axis. In local coordinates this is given by

$$\tilde{u}(x_{i-1/2}, t) = u_{i-1/2}(0), \quad \tilde{u}(x_{i+1/2}, t) = u_{i+1/2}(0),$$

and the intercell fluxes are given by

$$f_{i-1/2} = f(u_{i-1/2}((0)), \quad f_{i+1/2} = f(u_{i+1/2}((0)).$$

So in the above, $u_{i-1/2}((0))$ denotes the exact solution $u_{i-1/2}(x/t)$ of the Riemann problem $\text{RP}(u_i^n,\ u_{i+1}^n)$ evaluated along the intercell boundary, $x = 0$, in local coordinates.

In summary :
The complete set of possible solutions to the RP for Burgers equation are as given in §17.8, and together with usage of Eqn. 18.6, this would form the basis of a computer code. These would be applied directly to all cells i in Fig. 18.1, for $i = 2,\ 3, \ldots, M-1$.

18.2 Boundary conditions

At the left and right boundary points $i = 1$ and $i = M$ boundary conditions would require to be imposed. The simplest is that where a boundary function $u_1(t)$ (at left boundary, say) is enforced. In this case one simply defines the intercell flux by setting $f_{1/2} = f(u_1(t))$.

Dependent upon the problem and type of condition enforced (transmissive, reflective etc.), ficticious cells u_0 or u_{M+1} may well prove useful, namely

$$u_0 = u_1, \quad u_{M+1} = u_M, \quad \text{transparent boundary},$$

or

$$u_0 = -u_1, \quad u_{M+1} = -u_M, \quad \text{reflective boundary (solid wall)},$$

among others. More sophisticated treatments based on characteristics of the mathematical problem, at hand, are clearly possible and used in practice.

18.3 CFL time step

In the linear advection case, we found that a crucial aspect was to satisfy the Courant number for stability of the numerical method, namely

$$\text{CFL} = s\frac{\Delta t}{\Delta x} \leq 1, \quad \text{where } s \text{ is the advection speed.}$$

In nonlinear problems, as we have seen, at each time level, there are multiple wave speeds and thus multiple associated Courant numbers. This implies, constraining the time step Δt such that the fastest wave travels at most one cell length Δx. Thus we define the maximum Courant number C_{CFL} at time level n by

$$C_{\text{CFL}} = s^n_{\text{max}}\frac{\Delta t}{\Delta x}.$$

One identifies wave speeds, such as those emerging from solutions of Riemann problems at the intercell boundaries, and characteristic speeds u. At time level n, there is the possibility of maximising over all u^n_i discrete characteristic speeds. The other possibility is to take solutions from the Riemann problem at each cell interface; this information is available as part of the flux computation process. For the shock, if present, the shock speed is added to the mix, while for the rarefaction situation, there will be two characteristic speeds, at the head and tail of the expansion fan (see Fig. 17.10). Thus for these an intercell speed is evaluated

$$s^n_{i+1/2} = \begin{cases} |(u^n_i + u^n_{i+1})/2| & \text{shock ,} \\ \max\left(|u^n_i|, |u^n_{i+1}|\right) & \text{rarefaction .} \end{cases} \tag{18.7}$$

We of course maximise over all intercell flux speeds (including the boundary points), namely

$$s^n_{\text{max}} = \max\left\{s^n_{i+1/2}\right\}, \quad \text{for } i = 0, \ldots, N.$$

Finally the marching time step Δt is then determined from

$$\Delta t = \frac{C_{\text{CFL}}}{s^n_{\text{max}}}\Delta x;$$

typically a $C_{\text{CFL}} \approx 0.9 < 1$ is imposed.

More details in :
Toro, E. F. (1997) *Riemann Solvers and Numerical Methods for Fluid Dynamics – A Practical Introduction*. Springer.

A good read, which discusses the topic with clarity is :
Roe, P. L. (1986) *Characteristic-based Schemes for the Euler Equations*, Ann. Rev. Fluid Mech. **18** : *pp.* 337-365. https://doi.org/10.1146/annurev.fl.18.010186.002005.

The mathematical approach is outlined in the seminal paper by Lax :
Lax, P. D. (1973) *Hyperbolic systems of conservation laws and the mathematical theory of shock waves*, SIAM. https://doi.org/10.1137/1.9781611970562.ch1

19 The Lax-Wendroff Method and Conservation Equations

One explicit method which attempts to get the second order terms correct is due to Lax and Wendroff. We know that

$$\frac{u_n^{j+1}-u_n^j}{k} = [u_t + \tfrac{1}{2}ku_{tt}]_n^j + O(k^2)\ . \tag{19.1}$$

So if $u_t = -cu_x$ with c constant, then $u_{tt} = -cu_{xt} = c^2u_{xx}$, and we may write

$$\frac{u_n^{j+1}-u_n^j}{k} = -c\frac{\Delta u_n^j}{2h} + \frac{kc^2}{2h^2}\delta^2 u_n^j + O(k^2,\, h^2) \tag{19.2}$$

This leads to the centred scheme (**Scheme F**) in §15.2.

$$U_n^{j+1} = U_n^j - \tfrac{1}{2}q\Delta U_n^j + \tfrac{1}{2}q^2\delta^2 U_n^j \tag{19.3}$$

Note that this formula requires c to be constant.

However, a similar result can be derived for the **conservation** equation: $u_t + [f(u,\, x)]_x = 0$.

19.1 Simultaneous conservation equations

More generally, let $\mathbf{u}$ and $\mathbf{f}[\mathbf{u}]$ be p-vectors satisfying

$$\mathbf{u}_t + (\mathbf{f}[\mathbf{u}])_x = 0 \tag{19.4}$$

Equation (19.4) represents the conservation of p physical quantities. Ideally, we would like our FDM to conserve them also. For example, we could write

$$\mathbf{U}_n^{j+1} - \mathbf{U}_n^j = -\tfrac{1}{2}s\Delta\mathbf{F}_n^j \qquad \text{where} \quad \mathbf{F}_n^j = \mathbf{f}\left[\mathbf{U}_n^j\right] \quad \text{and} \quad s = k/h. \tag{19.5}$$

This scheme is **conservative** because if we sum over all values of n the right-hand-side all cancel (apart possibly from boundary terms if we are on a finite region of x), so that $\sum_n \mathbf{U}_n^{j+1} = \sum_n \mathbf{U}_n^j$. However, we have seen that (19.5) may be unstable. Alternatively we might rewrite (19.4) in the form

$$\left[\frac{\partial(u_l)}{\partial t}\right] + \sum_{m=1}^{p} A_{lm}\left[\frac{\partial(u_m)}{\partial x}\right] = 0 \qquad \text{for} \quad l = 1\ldots p \qquad \text{where} \quad A_{lm} = \frac{\partial(f_l)}{\partial(u_m)}\ , \tag{19.6}$$

and u_m is the m-th component of $\mathbf{u}$. If $A[\mathbf{u}]$ is the matrix whose (l,m)'th element is A_{lm}, we could then approximate

$$\mathbf{U}_n^{j+1} - \mathbf{U}_n^j = -\tfrac{1}{2}sA\left(\Delta\mathbf{U}_n^j\right) \tag{19.7}$$

but this would **not** be conservative unless A is constant. The Lax-Wendroff idea applied to (19.4) gives

$$\mathbf{u}_{tt} = -(\mathbf{f}[\mathbf{u}])_{xt} = -(A\mathbf{u}_t)_x = (A(\mathbf{f})_x)_x$$

so that

$$\left.\begin{aligned}(\mathbf{u}_{tt})_n^j &= \tfrac{1}{h}\left[A_{n+1/2}^j\left(\mathbf{f}_x\right)_{n+1/2}^j - A_{n-1/2}^j\left(\mathbf{f}_x\right)_{n-1/2}^j\right] + O(h^2)\\ &= \tfrac{1}{h^2}\left[A_{n+1/2}^j\left(\mathbf{f}_{n+1}^j - \mathbf{f}_n^j\right) - A_{n-1/2}^j\left(\mathbf{f}_n^j - \mathbf{f}_{n-1}^j\right)\right] + O(h^2)\end{aligned}\right]\ . \tag{19.8}$$

In the above, to the same order of accuracy we can approximate

$$A^j_{n+1/2} \equiv A[\mathbf{u}^j_{n+1/2}] = A\left[\tfrac{1}{2}(\mathbf{u}^j_{n+1} + \mathbf{u}^j_n)\right] + O(h^2)\,.$$

Using the estimate (19.8) for $\mathbf{u}_{tt}$, we obtain

$$\mathbf{U}^{j+1}_n = \mathbf{U}^j_n - \tfrac{1}{2}s\Delta\mathbf{F}^j_n + \tfrac{1}{2}s^2\left[A^j_{n+1/2}\left(\mathbf{F}^j_{n+1} - \mathbf{F}^j_n\right) - A^j_{n-1/2}\left(\mathbf{F}^j_n - \mathbf{F}^j_{n-1}\right)\right]\,. \tag{19.9}$$

However while (19.9) is a second order scheme, it is still non-conservative if A depends on $\mathbf{u}$, and moreover it is necessary to calculate $A^j_{n+1/2}$. A superior two-step version of the Lax-Wendroff scheme is due to Richtmyer. The system

$$\left.\begin{array}{ll} \mathbf{U}^{j+1/2}_{n+1/2} & = \frac{1}{2}\left(\mathbf{U}^j_n + \mathbf{U}^j_{n+1}\right) - \frac{1}{2}s\left(\mathbf{F}^j_{n+1} - \mathbf{F}^j_n\right) \\ \mathbf{U}^{j+1}_n & = \mathbf{U}^j_n - s\left(\mathbf{F}^{j+1/2}_{n+1/2} - \mathbf{F}^{j+1/2}_{n-1/2}\right) \end{array}\right]\,. \tag{19.10}$$

has truncation error $O(k^2 + h^2)$, is **conservative** and stable if the Courant condition holds. When A is constant, so that $\mathbf{F} = A\mathbf{U}$, it reduces to (19.9). For the one-dimensional case $p = 1$, with $f = cu$, we obtain the two-step scheme (**Scheme D**) with a different step-size.

19.2 Finite volume method in two-dimensions

Suppose the flux function in 2D is $\mathbf{F}(u) = (f(u), g(u))$, then it follows the conservation law has the form

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial g(u)}{\partial y} = 0.$$

Hence

$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial g}{\partial u}\frac{\partial u}{\partial y} = 0,$$

with (f_u, g_u) representing the wave speed of the 2D system in the (x, y) directions – the advection speed vector $\mathbf{v}$ is thus given by

$$\mathbf{v} = f_u\,\mathbf{i} + g_u\,\mathbf{j}.$$

Averaging the value of u over a now 2D finite-volume cell as depicted in Fig. 19.1, we obtain the general form of an explicit FV method, namely

$$\frac{u^{j+1}_{m,n} - u^j_{m,n}}{\Delta t} + \frac{f^*(u^j_{m,n}, u^j_{m+1,n}) - f^*(u^j_{m-1,n}, u^j_{m,n})}{\Delta x} + \frac{g^*(u^j_{m,n}, u^j_{m,n+1}) - g^*(u^j_{m,n-1}, u^j_{m,n})}{\Delta y} = 0,$$

hence defining intermediate fluxes

$$f^{*j}_{m+1/2,n} = f^*(u^j_{m,n}, u^j_{m+1,n}) \;\text{ and }\; g^{*j}_{m,n+1/2} = g^*(u^j_{m,n}, u^j_{m,n+1}).$$

It follows we may write, noting expressions 17.49, 17.52 and 18.6,

$$\frac{u^{j+1}_{m,n} - u^j_{m,n}}{\Delta t} + \frac{f^{*j}_{m+1/2,n} - f^{*j}_{m-1/2,n}}{\Delta x} + \frac{g^{*j}_{m,n+1/2} - g^{*j}_{m,n-1/2}}{\Delta y} = 0.$$

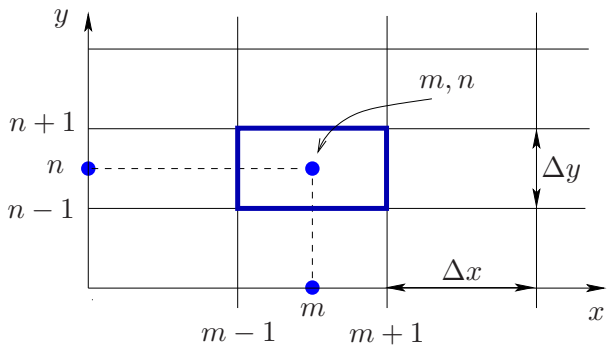

Figure 19.1: 2D flux

Lax Wendroff in two-dimensions :

The two-step Lax-Wendroff scheme (19.10) may be thus be extended to two space dimensions without difficulty. Consider the equation for $\mathbf{u}(x,\, y,\, t)$, with $\mathbf{u}$ representing a system of coupled hyperbolic PDEs in conservation form

$$\mathbf{u}_t + \mathbf{f}_x + \mathbf{g}_y = 0 \qquad \text{and take} \quad \Delta x = \Delta y = h.$$

We define $\mathbf{U}$ at the half time-levels for $(p,\, q) = (m,\, n + 1/2)$ or $(m + 1/2,\, n)$ by

$$\mathbf{U}_{pq}^{j+1/2} = \tfrac{1}{4}\left[\mathbf{U}_{p+,q}^{j} + \mathbf{U}_{p-,q}^{j} + \mathbf{U}_{p,q+}^{j} + \mathbf{U}_{p,q-}^{j}\right] - \tfrac{1}{2}s\left[\mathbf{F}_{p+,q}^{j} - \mathbf{F}_{p-,q}^{j} + \mathbf{G}_{p,q+}^{j} - \mathbf{G}_{p,q-}^{j}\right] \ ,$$

where $p\pm$ denotes $p \pm 1/2$ and similarly $q\pm$. At the integer time levels, for $(p,\, q) = (m,\, n)$ or $(m + 1/2,\, n + 1/2)$,

$$\mathbf{U}_{pq}^{j+1} = \mathbf{U}_{pq}^{j} - s\left[\mathbf{F}_{p+,q}^{j+1/2} - \mathbf{F}_{p-,q}^{j+1/2} + \mathbf{G}_{p,q+}^{j+1/2} - \mathbf{G}_{p,q-}^{j+1/2}\right] \ .$$

The resulting scheme is second order and conservative.

20 The 2D Wave Equation

We return now to pure Hyperbolic equations, such as the one-dimensional wave equation for $u(x,t)$,

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}, \quad \text{with} \quad u(x,0) \quad \text{and} \quad u_t(x,0) \quad \text{given.} \tag{20.1}$$

We know we could write this as two first order equations, but suppose we treat it directly, introducing a grid (h,k) and seek an approximation U_n^j to $u(nh,jk)$ in the familiar way. Using explicit centred differences in time and space we have the two-step second-order accurate in space and time scheme

$$U_n^{j+1} - 2U_n^j + U_n^{j-1} = q^2(U_{n+1}^j - 2U_n^j + U_{n-1}^j). \quad \text{where } q = \frac{ck}{h}. \tag{20.2}$$

The scheme is often referred to as the **leapfrog scheme**.

It being a second order equation in t, we need two initial conditions on U_n^0 and U_n^1 and then we can easily step to find U_n^{j+1} . We analyse the stability with Fourier methods, namely looking at solutions of the form

$$U_n^j = (\lambda)^j exp(in\xi)$$

and we find

$$\lambda^2 - 2\lambda + 1 = \lambda q^2(2\cos(\xi) - 2) \tag{20.3}$$

or

$$\lambda^2 - \left(2 - 4q^2 \sin^2(\frac{\xi}{2})\right)\lambda + 1 = 0. \tag{20.4}$$

We need $|\lambda| \leq 1$ for stability (note Fig. 20.1). Now without calculating λ explicitly, we can see from the constant term that the two roots have product 1. If the roots are real and distinct this means that one of the roots will have modulus greater than 1, and so the scheme will be unstable. Thus we require complex or double roots, or

$$\left(2 - 4q^2 \sin^2(\frac{\xi}{2})\right)^2 \leq 4 \quad \rightarrow \quad -2 \leq \left[2 - 4q^2 \sin^2(\frac{\xi}{2})\right] \leq 2 \tag{20.5}$$

As usual, the worst case is $\xi = \pi$ and we require the **Courant-Friedrichs-Lewy** condition $q \leq 1$ for stability, as we might have expected. The scheme (20.2) is conservative and

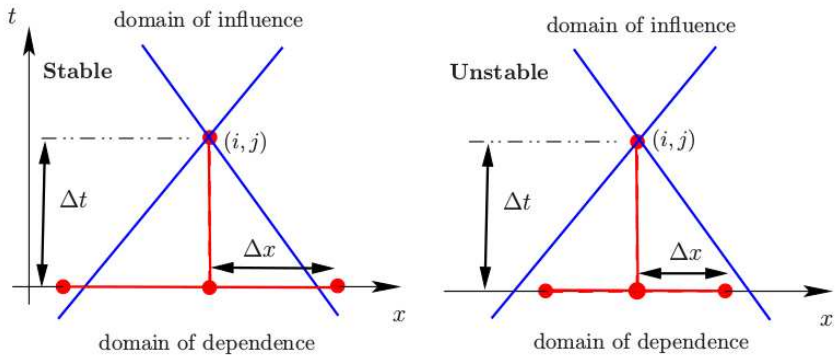

Figure 20.1: Domains of dependence and influence, CFL-condition. Stable configuration (left), unstable configuration (right).

models left and right travelling waves, $u = f(x-ct) + g(x+ct)$. If we have a boundary, say at $x = 0$, where we impose $u = 0$, then we must have $f = g$. Thus if a leftwards travelling wave encounters such a boundary, it will reflect, giving rise to a rightwards travelling wave. So if we want to have a **non-reflecting** boundary, we would need to impose the condition

$$cu_x = +u_t, \quad \text{on} \quad x = 0.$$

This is easy enough to implement. We can combine (20.2) on the boundary $n = 0$ with

$$q(U_1^j - U_{-1}^j) = U_n^{j+1} - U_n^{j-1}$$

to eliminate the ghost points at $n = -1$, keeping 2^{nd} order accuracy. Can we extend this scheme to two and three dimensions?

20.1 2D Wave equation: non reflecting boundary conditions

Suppose now $u(x, y, t)$ satisfies

$$\frac{\partial^2 u}{\partial t^2} = c^2\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = c^2\,\nabla^2 u \tag{20.6}$$

We can model this explicitly on a rectangular grid (h_x, h_y) by

$$\frac{U_{mn}^{j+1} - 2U_{mn}^j + U_{mn}^{j-1}}{c^2k^2} = \frac{1}{h_x^2}\delta_x^2 U_{mn}^j + \frac{1}{h_y^2}\delta_y^2 U_{mn}^j.$$

For simplicity, we take $h_x = h_y = h$. We analyse the stability with

$$U_{mn}^j = (\lambda)^j exp(im\xi + in\eta)$$

and as before we find stability provided λ is complex for all ξ and η, which requires

$$q \leq 1/\sqrt{2}.$$

Similarly, in 3-dimensions, the CFL condition would be $q \leq 1/\sqrt{3}$.

In 2 or 3 dimensions it is quite likely that we will want to include artificial boundaries to the computational domain which do not reflect waves, but this is harder to do. Suppose we are in $y > 0$ with a boundary at $y = 0$. Taking Fourier transforms, the general solution to (20.6) can be written as a superposition of the waves $\exp(ilx + imy + i\omega t)$, where l, m and ω are related by

$$c^2(l^2 + m^2) = \omega^2.$$

On $y = 0$, we wish to impose that only waves travelling in the negative y direction are permitted, in other words, assuming $\omega > 0$ then we must have $m > 0$, so that

$$mc = +\omega - \frac{c^2 l^2}{2\omega}.$$

This is equivalent to the unusual boundary condition

$$cu_{yt} = u_{tt} - \frac{1}{2}c^2 u_{xx}, \quad \text{on} \quad y = 0. \tag{20.7}$$

This can be implemented on the boundary and greatly reduces unwanted reflections.

This boundary condition is somewhat more difficult to implement but results in a much improved outflow boundary condition. This type of boundary condition is the most commonly used, due to its relative ease of implementation and its efficiency in substantially reducing backscatter. This type of outflow boundary conditions has been implemented, and the results are shown in Fig. 20.2.

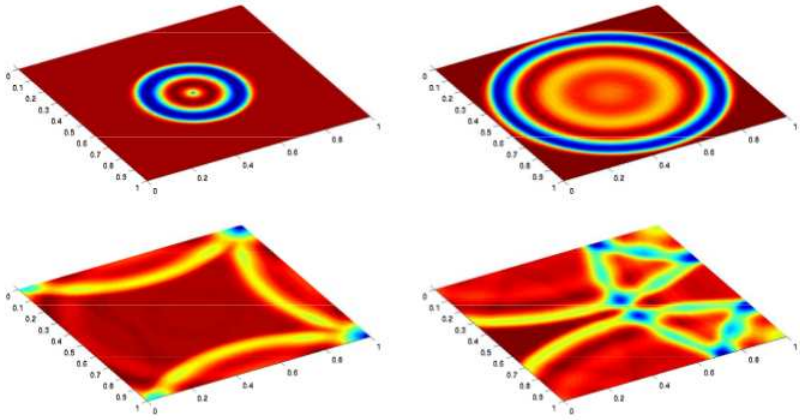

Figure 20.2: Propagation of a localised disturbance. Three reflecting and one non-reflecting boundaries based on Eqn. (20.7).

An even simpler outgoing boundary condition based on the linear advection equation for this problem is shown in Fig. 20.3.

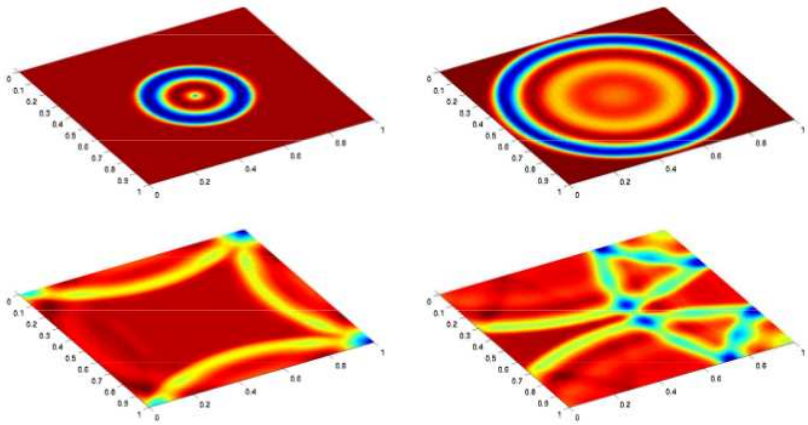

Figure 20.3: Propagation of a localised disturbance. Three reflecting and one non-reflecting boundaries based on the linear advection equation $u_t = cu_x$

But still it is imperfect. An important technique for avoiding unphysical reflections is known as **PML**, which stands for **Perfectly Matched Layer**.

21 Perfectly Matched Layers

Solutions of the wave equation can move large distances without dissipating. As a result, correct treatment of "*infinity*" is crucial – if outgoing waves are allowed to reflect back towards the region of interest, they may eventually completely invalidate the solution. Last lecture we derived an approximate boundary condition which greatly reduced spurious reflections at the edge of the computational domain, but we would like to do better.

How can we accurately impose non-reflecting boundary conditions? In real life we might clad a concert hall ceiling with absorbing material – curtains and carpets are well known to muffle echoes in houses. So imagine cladding the domain in which we are interested with a region which causes waves to decay. The difficulty is to do this in a manner which does not encourage reflections. Waves may bounce off many inhomogeneities (think of mirages, for example). Techniques to model the waves behaviour at the boundaries of a finite domain can be dealt with by attempting to solve or approximate the exact dispersion relation for waves in the far-field and impose the corresponding conditions on the boundary of the computational domain such that this boundary appears transparent to outgoing waves.

We are familiar with acoustic waves, the need also arises in quantum mechanics. Consider the 1D Schrodinger equation

$$\begin{aligned} i\frac{\partial u}{\partial t} &= -\frac{\partial^2 u}{\partial x^2} + V(x,t)u, \quad x \in \mathcal{R}, t > 0, \\ \lim_{|x|\to\infty} u(x,t) &= 0, \\ u(x,0) &= u_I(x), \end{aligned} \qquad .$$

Here, V denotes a given real potential. The equation arises in quantum semiconductors, electromagnetic wave propagation, seismic propagation, underwater acoustics among others*.

Techniques which **artificially** manipulate the waves behaviour at the boundaries of a finite domain is an alternative means to avoid issues with reflections at the boundaries. In this, the idea is to surround the computational domain by additional grid points on which a highly dissipative equation can be solved that damps the outgoing waves. A switching function is used to blend from the equations governing the flow evolution on the inside of the domain (*i.e.*, the physically relevant domain) to the equations causing the dissipation of the flow in the absorbing layer. Even though this is a simple and appealing concept, the proper implementation in a numerical simulation is far from trivial.

Specifically, on a boundary at, say $x = 0$, we want a solution which behaves like

$$e^{i\alpha(x-ct)}$$

in $x < 0$, to match with something which decays exponentially with x in $x > 0$. One way of doing that is to add an imaginary part to x. We may think of this as following the same solution along a contour in the complex x-plane other than the x-axis. Consider a variable

$$X = x + if(x),$$

*Antoine *et al.* Commun. Comput. Phys., Vol 4(4) pp729-796, 2008

where $f(x) > 0$ for $x > 0$, and $f(x) = 0$ for $x < 0$. Then the derivatives transform as

$$\frac{\partial}{\partial X} = \frac{1}{1 + if'(x)} \frac{\partial}{\partial x}$$

The idea is to solve the equation in X-space and hope that the resultant solution has the desired decaying properties in x-space.

Waves in homogeneous medium : explanation

Generally, in a homogeneous medium in infinite space, we assume that propagating waves form a superposition of linear plane-waves, namely in a simple 1-D reference frame, we may assume

$$p(x,t) = p_o e^{i(\alpha x - \omega t)},$$

where ω/α is the phase velocity. Next let us add an additional term to the above,

$$p(x,t) = p_o e^{i(\alpha(x + i\gamma x) - \omega t)},$$

with γ some *real positive* quantity. This may be re-written

$$p(x,t) = p_o e^{i(\alpha x - \omega t)}\ e^{-\gamma \alpha x},$$

that is the solution decays exponentially provided $(\gamma\alpha)$ is real and positive. Choosing where $\gamma \neq 0$, we have effectively there, introduced a *complex material* with *absorbing or dampening* properties.

The basic idea is, for unbounded domain problems, far away from the field of interest we can devise a modified set of PDEs with additional terms, which damp out the outwardly propagating waves. With hope that the wave amplitude has been effectively reduced to zero by the time it reaches the final computational boundary point, **within** the PML regions. In the non-PML regions the additional tems are zero, and so we solve the precise PDE that we **do** want to solve.

21.1 A more general wave equation

A generalised form of the wave equation for $p(\mathbf{x}, t)$ is

$$p_{tt} = b\ \nabla \cdot\ (a \nabla p), \tag{21.1}$$

where $a(\mathbf{x})$ and $b(\mathbf{x})$ are given, positive functions of position, and can represent an inhomogeneous medium through which the waves travel. We assume they do not vary with x at the boundary $x = 0$, though they may do elsewhere. If both a and b are constant, we have the standard wave equation. Now (21.1) can be written as a system of first order equations if we introduce the vector $\mathbf{v}$ such that

$$\mathbf{v}_t = a\ \nabla p, \quad \text{and} \quad p_t = b\ \nabla \cdot \mathbf{v}. \tag{21.2}$$

We will start with the 1-D wave equation in terms of X and t, so that

$$v_t = a\ p_X, \quad \text{and} \quad p_t = b\ v_X.$$

Assuming an $\exp(-i\omega t)$ behaviour and transforming from X to x, we have

$$-i\omega v \equiv v_t = \frac{a\, p_x}{1 + if'(x)} \quad \text{and} \quad -i\omega p \equiv p_t = \frac{b\, v_x}{1 + if'(x)}$$

or multiplying up

$$-i\omega p + \omega f' p = b\, v_x \quad \text{and} \quad -i\omega v + \omega f' v = a\, p_x.$$

If we introduce $\sigma(x) = \omega f'$ and re-insert the time derivatives, we have the system

$$p_t = b\, v_x - \sigma p \quad \text{and} \quad v_t = a\, p_x - \sigma v.$$

We can choose a function $\sigma(x)$ which is zero for $x < 0$ and implement this scheme easily.

Observe, in this simple case, we may eliminate the v-variable to give the following PML form of the second-order wave equation

$$p_{tt} + 2\sigma p_t + \sigma^2 p = c^2 p_{xx}, \quad \text{here, we have made} \quad ab = c^2.$$

Common choices for $\sigma(x)$ range from piece-wise constant functions to piece-wise linear functions to smooth, monotonically increasing functions. An assortment of the most commonly used is given in Fig. 21.1.

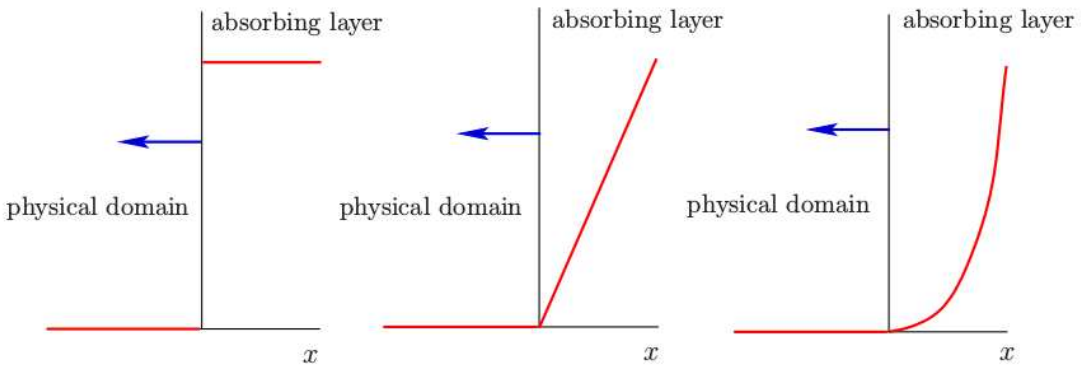

Figure 21.1: Choices for the absorption coefficient $\sigma(x)$.

At the interface between the PML and normal region, a *perfectly devised layer* should have the key feature of not causing any internal reflections back into the region of interest, to corrupt the solution. The interface is not reflection-free, but by careful choice of how the strength of $\sigma(x)$ varies (generally increasing) as one goes more and more deeper into the PML an optimal solution can usually be found, with a few iterations or fine-tuning of the $\sigma(x)$ function and observing its impact.

For the 1-D problem σ appears in a simple manner. But we saw last time there is no difficulty in imposing a non-reflecting boundary condition for the 1-D problem. In 2D we developed an approximate condition, see Eqn. (20.7), but it would clearly be useful to devise a better more general approach.

21.2 A perfectly matched layer in 2D

Now let's write $\mathbf{v} = (v, w)$. Our system takes the form

$$-i\omega p \equiv p_t = b\, \nabla \cdot \mathbf{v} \equiv b\, v_X + b\, w_y, \tag{21.3}$$

$$-i\omega v \equiv v_t = a\, p_X \quad \text{and} \quad -i\omega w \equiv w_t = a\, p_y. \tag{21.4}$$

Now the second equation in (21.4) does not involve X. Substituting for X into the first equation in (21.4) and in (21.3) and multiplying up by $(1 + i\sigma/\omega)$, we have

$$b\, v_x + b\, w_y(1 + i\sigma/\omega) = -i\omega p + \sigma p, \tag{21.5}$$

$$a\, p_x = -i\omega v + \sigma v. \tag{21.6}$$

Equation (21.6) easily translates to time derivatives, but (21.5) involves $1/(i\omega)$ which corresponds to time integration rather than differentiation. So we introduce a function $\psi(x, y, t)$, such that

$$-i\omega\psi = b\,\sigma\, w_y \quad \text{with} \quad \psi = 0 \quad \text{at} \quad t = 0.$$

Then (21.5) reads

$$b\, v_x + b\, w_y + \psi = -i\omega p + \sigma p,$$

and we can write the system as the obvious generalisation to two dimensions :

$$\left.\begin{aligned} p_t &= b\,\nabla\cdot\mathbf{v} - \sigma p + \psi \\ v_t &= a\, p_x - \sigma v \\ w_t &= a\, p_y \\ \psi_t &= b\,\sigma\, w_y \end{aligned}\right\} \tag{21.7}$$

Where $\sigma = 0$, these equations reduce to (21.2). In the layer where $\sigma > 0$, we have decay with no reflection. We can place a boundary a little distance into the layer, secure in the knowledge that any reflections from this boundary will have decayed exponentially and should not cause significant errors in the region of computational interest. Naturally, it is easy to modify these equations for other boundaries.

22 The Method of Characteristics

Recall earlier notes (repeated here) : Most physical systems are governed by second order PDEs. In this course we discuss **Finite Difference Methods (FDMs)** for solving such equations. We want our algorithms to be able to reproduce the physics and so to begin with, we must understand the physical background. We consider the equation for $u(x, y)$

$$a\, u_{xx} + b\, u_{xy} + c\, u_{yy} = f. \tag{22.1}$$

This equation is called **quasi-linear** provided the functions $a,\ b,\ c$ and f do not depend on $u_{xx},\ u_{xy}$ or u_{yy}, namely that the highest order derivatives occur linearly. They may, however depend on $x,\ y,\ u,\ u_x$ and u_y, so that (22.1) is not necessarily linear – for example it is perfectly allowable in the discussion that follows that

$$f = d\, u_x + e\, u_y + h\, u + g, \tag{22.2}$$

provided $d,\ e,\ h,\ g$ functions retain the *quasi-linear* requirement.

Suppose we know u, u_x and u_y along some curve Γ in (x, y)-space. From a point P on Γ we move a small vector displacement $(dx,\ dy)$ to a new point Q not on Γ, as shown in Fig. 22.1. Under what circumstances can we determine uniquely the values of u, u_x and u_y at Q? We denote the change in these variables by du, $d(u_x)$ and $d(u_y)$. Then by the chain rule for partial derivatives $du = u_x dx + u_y dy$, which is known because u_x and u_y are known along Γ. Similarly,

$$\left.\begin{aligned} d(u_x) &= u_{xx}dx + u_{xy}dy \\ d(u_y) &= u_{xy}dx + u_{yy}dy \end{aligned}\right\}. \tag{22.3}$$

We combine (22.1) and (22.3) in matrix form:

$$\begin{pmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{pmatrix} \begin{pmatrix} u_{xx} \\ u_{xy} \\ u_{yy} \end{pmatrix} = \begin{pmatrix} f \\ d(u_x) \\ d(u_y) \end{pmatrix}, \tag{22.4}$$

a, b and c are known locally because u, u_x and u_y are known, and so the 3×3 matrix is known. Equation (22.4) will have a unique solution for u_{xx}, u_{xy} and u_{yy} unless the determinant of that matrix vanishes, that is unless

$$\begin{vmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{vmatrix} = a(dy)^2 - b\, dxdy + c(dx)^2 = 0. \tag{22.5}$$

If (22.5) holds, the equation (22.4) will have either no solution or infinitely many solutions. The condition for solutions to exist, which we will use later in the course, is that

$$a\,\frac{d(u_x)}{dx} + c\,\frac{d(u_y)}{dy} = f\ . \tag{22.6}$$

This is surprising. If we can choose a direction (dx, dy) which satisfies (22.5) we have the possibility that the second derivatives u_{xx} etc. may not be uniquely defined. In other words, the solution may have discontinuities across the line PQ, with u_{xx} taking different values on each side.

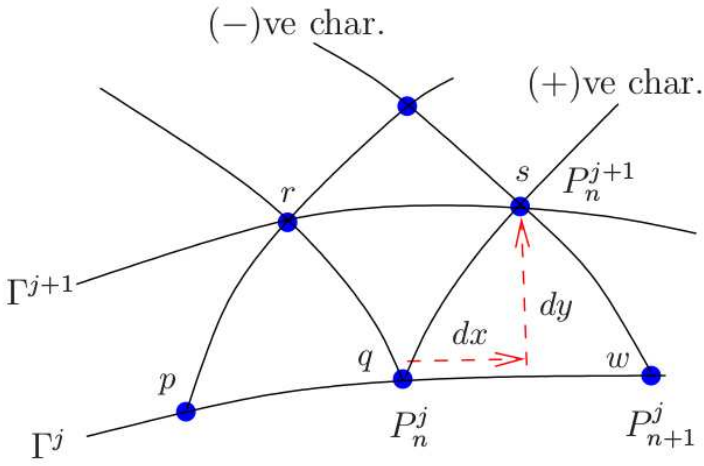

Figure 22.1: Problem description.

It is very important to know whether our solution can have this property. Equation (22.5) is called the **Characteristic equation** of (22.1). It is a quadratic in dy/dx with solution

$$\frac{dy}{dx} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a} \,. \tag{22.7}$$

We can use equations (22.7) and (22.6) to find the solution. Equation (22.7) defines two families of characteristics, one with the $+$sign and one with the $-$sign. Starting from a given point (x, y) we can step (22.7) using some ODE-solver to find a new point

$$(x + dx, y + dy)$$

which lies on the characteristic. Note that characteristics from the same family are approximately parallel – if ever they cross, then a **shock** develops and the solution becomes singular. However the two families have different gradients and so characteristics from different families meet all the time. So if we have a set of points P_n^j on some curve Γ^j at some "$time$", j, we can define points P_n^{j+1} to be the intersection of the $(+)ve$ characteristic from P_n^j and the $(-)ve$ characteristic from P_{n+1}^j as in the diagram, thus defining a new curve Γ^{j+1}.

Now assume u, u_x and u_y are known on Γ^j. This means that u is known after an infinitesimal step (dx, dy) by

$$du = u_x dx + u_y dy,$$

but u_x and u_y need to be found. The key is to use the condition (22.7) along the characteristics, relating how much u_x and u_y change j. As we have two equations in two unknowns we can from the known values at P_n^j and P_n^{j+1} easily determine u_x and u_y at the new point P_n^{j+1} and hence we can advance the region where the solution is known to a larger region, bounded by Γ^{j+1}. This defines the method of characteristics.

As an example suppose the roots of (22.7) are real and distinct, and can be denoted by

$$\frac{dy}{dx} = F, \quad \text{and} \quad \frac{dy}{dx} = G, \tag{22.8}$$

where initial values of $u,\ u_x$ and u_y are known on Γ^j. As a first approximation, with reference to Fig. 22.1, we may regard p — r and q — r as straight lines of slopes F_p and G_q. Then (22.8) can be approximated by

$$y_r - y_p = F_p\ (x_r - x_p), \quad \text{and} \quad y_r - y_q = G_q\ (x_r - x_q), \tag{22.9}$$

giving two equations for the two unknowns $(x_r,\ y_r)$. Next equation (22.1) can be used to show that the differentials $P = u_x$ and $Q = u_y$ are related by the following **compatability** relationship

$$a\ \frac{dy}{dx}\ dP + c\ dQ - f\ dy = 0, \tag{22.10}$$

hence

$$a\ F\ dP + c\ dQ - f\ dy = 0, \quad \text{and} \quad a\ G\ dP + c\ dQ - f\ dy = 0. \tag{22.11}$$

We approximate the first expression along p — r by the equation

$$a_p\ F_p\ (P_r - P_p) + c_p\ (Q_r - Q_p) - f_p\ (y_r - y_p) = 0, \tag{22.12}$$

and the second expression along q — r thus

$$a_q\ G_q\ (P_r - P_q) + c_q\ (Q_r - Q_q) - f_q\ (y_r - y_q) = 0. \tag{22.13}$$

Once (x_r, y_r) have been evaluated from (22.9), these are two equations for unknowns (P_r, Q_r). The value of u on r can then be obtained from

$$du\ =\ P\ dx\ +\ Q\ dy, \tag{22.14}$$

on replacing values of (P, Q) along p — r by

$$u_r - u_p\ = \frac{1}{2}(P_p + P_r)(x_r - x_p) + \frac{1}{2}(Q_p + Q_r)(y_r - y_p). \tag{22.15}$$

Solution of the above gives us a first approximation to u_r. We next improve upon this by replacing expressions (22.9) with new improved estimates of (x_r, y_r), using

$$y_r - y_p = \frac{1}{2}(F_p + F_r)\ (x_r - x_p), \quad \text{and} \quad y_r - y_q = \frac{1}{2}(G_q + G_r)\ (x_r - x_q), \tag{22.16}$$

and equations (22.12)–(22.13) become

$$\left.\begin{array}{rcl} \frac{1}{4}(a_p + a_r)(F_p + F_r)(P_r - P_p) + \frac{1}{2}(c_p + c_r)(Q_r - Q_p) - \frac{1}{2}(f_p + f_r)(y_r - y_p) & = & 0, \\ \frac{1}{4}(a_q + a_r)(G_q + G_r)(P_r - P_q) + \frac{1}{2}(c_q + c_r)(Q_r - Q_q) - \frac{1}{2}(f_q + f_r)(y_r - y_q) & = & 0. \end{array}\right\}, \tag{22.17}$$

An improved value for u_r then follows from (22.15). Repetition of the above steps, through iteration and correction thus provides u_r to sufficient accuracy. The number of iterations can be minimised by making q and p close to each other.

An advantage of the method is that by explicitly following characteristics we are modelling the physics well. If the solution does have discontinuities, we expect to follow them accurately. A disadvantage is that it does not generalise easily to 3D, when the characteristic curves become cones. Also, we can no longer have a regular grid – the equation decides

where to place the points P_n^j, and they may bunch together. One could argue, that grid points are being carried to areas where they are needed, but there is the danger of thinning of points in some regions.

This leads us on to a more fundamental question for this course : we have always tried to use a regular grid, which affords us reliable 2-nd order accuracy. Should we relax this in some cases? We need points in order to resolve rapid variation. If we have a solution like

$$u = e^{-10x+x^2},$$

we might want to have more points near $x = 0$ than near $x = 1$. To put it another way, if we use a small enough step-length to resolve the fast variation, we are being unnecessarily wasteful of computing power elsewhere. Suppose we introduce a new coordinate, $\xi = f(x)$. Then we can rewrite the PDE in terms of the new variable ξ and then use central differences in ξ, maintaining 2-nd order accuracy. We may clutter up the PDE with derivatives of f (which may be large in some regions), but this nevertheless gives us a methodical means of redistributing points to where we think they are needed – see notes on grid mappings.

23 Operator Splitting – Fractional Step Methods

If we have a parabolic-like equation and discretise the spatial derivatives, we have a set of differential equations of the form

$$u_t = \mathcal{L}u, \quad \text{where } \mathcal{L} \text{ is an operator, perhaps a matrix.}$$

Using a Taylor series, we have

$$\frac{u^{j+1} - u^j}{k} = u_t + \frac{1}{2}ku_{tt} + \ldots \tag{23.1}$$

Assuming the operator $\mathcal{L}$ is time-independent, then using the Lax-Wendroff idea, we can write

$$u_{tt} = \mathcal{L}u_t = \mathcal{L}(\mathcal{L}u) \equiv \mathcal{L}^2 u$$

and then

$$u^{j+1} = u^j + k\mathcal{L}u^j + \frac{1}{2}k^2\mathcal{L}^2u^j + \ldots = (\mathbf{I} + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2 + \ldots)u^j. \tag{23.2}$$

Here $\mathbf{I}$ denotes the identity operator – we can think of it as "1". Defining the exponential operator

$$\exp(kL) \equiv (\mathbf{I} + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2 + \frac{1}{6}k^3\mathcal{L}^3 + \ldots),$$

then the exact solution of (23.1) can be written

$$u^{j+1} = \exp(kL)\ u^j. \tag{23.3}$$

But we will only be concerned with the $O(k^2)$ terms below.

Often we will be dealing with equations with qualitatively different terms (e.g. advection, diffusion, nonlinear forcing) which, drawing on our previous experience, we may wish to deal with individually in different manners. For example, perhaps one part we feel should be dealt with explicitly using Lax-Wendroff, and another implicitly using multi-grid. Can we treat the two parts differently without sacrificing the overall accuracy of the scheme? This is the idea behind **Operator Splitting**. Recall the lecture on the ADI method, where for the 2D diffusion equation we alternately treated the Laplacian explicitly in the x-direction and implicitly in the y-direction and then vice versa. Can we do the same thing in general?

Suppose we have two processes which we represent by spatial discretisations $\mathcal{L}$ and $\mathcal{M}$

$$u_t = \mathcal{L}u + \mathcal{M}u \tag{23.4}$$

which has the solution

$$u^{j+1} = u^j + k(\mathcal{L} + \mathcal{M})u^j + \frac{1}{2}k^2(\mathcal{L} + \mathcal{M})^2u^j + \ldots \tag{23.5}$$

Note this is a schematic representation – the operators could involve iteration or implicit methods, but ultimately we represent the operation in this manner. Suppose we alternate solving

$$u_t = \mathcal{L}u \quad \text{and} \quad u_t = \mathcal{M}u$$

by our favourite methods, solving

$$\left.\begin{aligned} \hat{u} &= (\mathbf{I} + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2)u^j, \quad \text{and then} \\ u^{j+1} &= (\mathbf{I} + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)\hat{u} \end{aligned}\right\} \tag{23.6}$$

We then have

$$\begin{aligned} u^{j+1} &= (\mathbf{I} + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)(\mathbf{I} + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2)u^j, \\ &= u^j + k(\mathcal{L} + \mathcal{M})u^j + \frac{1}{2}k^2(\mathcal{L}^2 + \mathcal{M}^2 + 2\mathcal{M}\mathcal{L})u^j. \end{aligned} \tag{23.7}$$

Is this the same as (23.5)? Only if $\mathcal{M}\mathcal{L} = \mathcal{L}\mathcal{M}$ so that the two operators commute. This will not usually be the case, so that the simple splitting method has an error of $O(k)$, even if $\mathcal{L}$ and $\mathcal{M}$ are implemented in an $O(k^2)$ manner.

With a little more work, we can achieve 2nd order accuracy. We could, for example, now go back to u^j and do an $\mathcal{L}$ step and then an $\mathcal{M}$ step upon it. Averaging the two resulting estimates for u^{j+1} , we will have the correct $O(k^2)$ term, namely

$$\frac{1}{2}(\mathcal{L}^2 + \mathcal{M}^2 + \mathcal{L}\mathcal{M} + \mathcal{M}\mathcal{L}).$$

More subtly, and more efficiently, we could adopt the time-symmetric scheme of taking a half-step with $\mathcal{L}$, a full step with $\mathcal{M}$ and then a half-step with L, a process known as **Strang splitting**. Thus

$$\begin{aligned} \hat{u} &= (\mathbf{I} + \frac{1}{2}k\mathcal{L} + \frac{1}{2}(\frac{1}{2}k)^2\mathcal{L}^2)u^j, \\ u^* &= (\mathbf{I} + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)\hat{u} \\ u^{j+1} &= (\mathbf{I} + \frac{1}{2}k\mathcal{L} + \frac{1}{2}(\frac{1}{2}k)^2\mathcal{L}^2)u^*. \end{aligned} \tag{23.8}$$

The final estimate is $u^{j+1} = \mathcal{P}u$ where, neglecting terms of $O(k^3)$,

$$\begin{aligned} \mathcal{P} &= (\mathbf{I} + \frac{1}{2}k\mathcal{L} + \frac{1}{8}k^2\mathcal{L}^2)(\mathbf{I} + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)(\mathbf{I} + \frac{1}{2}k\mathcal{L} + \frac{1}{8}k^2\mathcal{L}^2) \\ &= \mathbf{I} + k(\frac{1}{2}\mathcal{L} + \mathcal{M} + \frac{1}{2}\mathcal{L}) + k^2(\frac{1}{8}\mathcal{L}^2 + \frac{1}{2}\mathcal{M}^2 + \frac{1}{8}\mathcal{L}^2 + \frac{1}{2}\mathcal{L}\mathcal{M} + \frac{1}{2}\mathcal{M}\mathcal{L} + \frac{1}{4}\mathcal{L}^2) \\ &= \mathbf{I} + k(\mathcal{L} + \mathcal{M}) + \frac{1}{2}k^2(\mathcal{L}^2 + \mathcal{M}^2 + \mathcal{L}\mathcal{M} + \mathcal{M}\mathcal{L}) \end{aligned} \tag{23.9}$$

which agrees with the estimate (23.5). The method (23.8) uses two calls to the "$\mathcal{L}$-routine" and one to the "$\mathcal{M}$-routine". Other things being equal, we would choose $\mathcal{M}$ to be the more time-expensive routine.

But wait! We are not just doing one step, we are taking several. Our 1st time step ends with a half-step in $\mathcal{L}$, and the second begins with a half step in L – we could combine those and take a full step with $\mathcal{L}$. As a check we note that taking two half-steps involves

$$(\mathbf{I} + \frac{1}{2}k\mathcal{L} + \frac{1}{8}k^2\mathcal{L}^2)^2 = \mathbf{I} + k\mathcal{L} + (\frac{2}{8} + \frac{1}{4})k^2\mathcal{L}^2 = \mathbf{I} + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2,$$

which is the same as one complete k-step. To put it another way, we have shown that operating alternately with $\mathcal{L}$ and $\mathcal{M}$ is the same as $(\mathcal{L} + \mathcal{M})$. Clearly the operator $\frac{1}{2}\mathcal{L}$ commutes with itself. Thus we can preserve $O(k^2)$ accuracy by beginning with a half-step with $\mathcal{L}$, then alternating full-steps with $\mathcal{M}$ and then $\mathcal{L}$, ending with a half-step in $\mathcal{L}$.

24 The Navier-Stokes equations in two-dimensions

An important application of many of the techniques of this course is to the equations of Fluid Dynamics. The motion of an incompressible Newtonian fluid is defined by its velocity $\mathbf{u}(\mathbf{x},t)$ and pressure $p(\mathbf{x},t)$ which satisfy the equations

$$\nabla\cdot\mathbf{u}=0, \tag{24.1}$$

$$\mathbf{u}_t+\mathbf{u}\cdot\nabla\mathbf{u}=-\nabla p+\nu\nabla^2\mathbf{u}+\mathbf{F}, \tag{24.2}$$

where $\mathbf{F}$ is a body force such as gravity g (say). These equations are tremendously important with very wide application. Here ν is a constant parameter, the inverse of the Reynolds number (Re) if the variables have been made non-dimensional. It measures the relative strength of diffusion and advection. The character of the equations is very different if ν is large or small because ν multiplies the highest derivative in the equation.

Here, we shall only consider two-dimensional (2D) flows, of the form

$$\mathbf{u}=(u(x,y,t),v(x,y,t),0)\,,\quad\text{and}\quad p=p(x,y,t) \tag{24.3}$$

Flows of this form can be represented by a single scalar function, $\psi(x,y,t)$. The incompressibility condition (24.1) can be satisfied by writing

$$\mathbf{u}=\nabla\times(0,0,\psi),\quad\text{and}\quad u=\frac{\partial\psi}{\partial y},\quad v=-\frac{\partial\psi}{\partial x}. \tag{24.4}$$

Then the vorticity,

$$\nabla\times\mathbf{u}=(0,0,\omega),\quad\text{where}\quad\omega=-\nabla^2\psi. \tag{24.5}$$

Now taking the curl of (24.2), we can eliminate the pressure field. The resulting vorticity equation only has a z-component, which is

$$\omega_t+\mathbf{u}\cdot\nabla\omega=\nu\nabla^2\omega+\mathbf{G}, \tag{24.6}$$

where

$$\mathbf{G}=\tilde{z}\cdot\nabla\times\mathbf{F}. \tag{24.7}$$

We could if we wish substitute for ω to obtain a single equation in ψ

We see that ν measures the relative importance of diffusion to advection. We also note that ν multiplies the highest derivative in the equation, so we require an extra boundary condition when $\nu\neq 0$.

What boundary conditions shall we impose? As we have not included an external force, we will only get motion if it is driven by a moving boundary. A typical problem is the Driven Cavity flow. Suppose we have a rectangle, $0<x<1$ and $0>y>-L$. On the three walls $x=0,x=1$ and $y=-L$ we impose that the velocity is zero, $\psi_x=\psi_y=0$. On the top wall $y=0$, we impose a constant horizontal flow, so that $\psi_y=1$ on $y=0$, but otherwise ψ,ψ_x and ψ_y are all zero on the boundary. We could envisage ourselves caught in a storm. We dig ourselves a hole to sit in to shelter from the driving wind. But does the flow penetrate into the hole?

24.1 Nearly inviscid flow, $\nu \to 0$, High Reynolds Number

If $\nu$is small, we are nearly at the inviscid Euler flow we considered last lecture, where ω was advected around, being stretched but not dissipated. If anything, adding a small amount of diffusion would probably make the flow better behaved, by preventing high gradients from developing. If ν is small, we might consider using an explicit representation of the diffusion, as the stability constraint $\nu k/h^2 < 1/2$ might not be too severe.

24.2 Stokes flow: $\nu \to \infty$, Low Reynolds Number

If the flow is very viscous, or slow, the parameter $\nu \gg 1$. In these circumstances the equations become

$$\nabla^2 \omega = 0, \quad \nabla^2 = -\omega \tag{24.8}$$

This type of flow is known as Stokes flow or creeping flow which is of importance in many applications, such as the flow in **MEMS** (micro-electromechanical systems), flow in polymers and material processing, and flow in and around micro-biological systems. Inertial effects in these systems can often be ignored leading to a simplified set of equations. How might we solve this equation numerically? Those two Laplacians look very tempting. Can we devise a multigrid code? Effectively, we are solving the **biharmonic equation**

$$\nabla^2(\nabla^2 \psi) = 0$$

We have a slight difficulty. We are missing a boundary condition on ω but instead we have two conditions on ψ. How can we nevertheless use two multigrid processes?

Suppose we knew ω. Then we could clearly calculate ψ using the Dirichlet boundary condition $\psi = 0$ only. The solution would not in general satisfy the Neumann boundary condition. However, if we could use it and the newly found ψ to define a boundary condition on ω, we would have the basis of an iterative scheme.

Suppose at $x = 0$ we have $\psi = 0$ and $\psi_x = f(y)$. Thus one step away from the wall*, we have

$$\psi(h) = h\psi_x + \frac{1}{2}h^2 \psi_{xx}(0).$$

Now,

$$-\omega = \psi_{xx} + \psi_{yy} = \psi_{xx} \text{ on } x = 0. \tag{24.9}$$

We can therefore identify

$$\omega(0, y) = \frac{2f(y)}{h} - \frac{2\psi(h, y)}{h^2} + O(h), \tag{24.10}$$

So we can envisage an iterative scheme where we use ψ to find ω and then ω to find ψ.

In place of (24.10) we could derive an $O(h^2)$ approximation using two points.

$$\left.\begin{array}{rcl} \psi(h) & = & h\psi' + \frac{h^2}{2}\psi'' + \frac{h^3}{3!}\psi''' + O(h^4) \\ \psi(2h) & = & 2h\psi' + 2h^2\psi'' + \frac{4h^3}{3}\psi''' + O(h^4) \end{array}\right\}, \tag{24.11}$$

*It follows, since $\psi = 0$ for all y, then $\psi_y = 0$ and hence $\psi_{yy} = 0$.

Eliminating ψ''' we have

$$8\psi(h) - \psi(2h) = 6h\psi' + 2h^2\psi'' + O(h^4),$$

and so

$$\omega(0,y) = \frac{3f(y)}{h} - \frac{8\psi(h,y) - \psi(2h,y)}{2h^2} + O(h^2). \tag{24.12}$$

Using such schemes we can solve Dirichlet problems for ω and ψ in turn.

24.3 The general case – intermediate Reynolds number

If both advection and diffusion are important, we would like somehow to marry our ideas for dealing with each process. Ideally, we would like to treat the advection explicitly but the diffusion implicitly, hopefully using a multigrid algorithm. One idea we might try is an **Operator Splitting** approach. In the first part of the algorithm we advect ω ignoring the diffusion, while in the 2nd part we diffuse the new ω.

24.4 Rayleigh-Benard Convection

An interesting fluid dynamical is known as **Convection**. Convection occurs because fluids expand when heated. Their density decreases and they become buoyant, in accordance with Archimedes's principle. If T denotes the excess temperature The body force

$$\mathbf{F} = -\beta T \mathbf{g}$$

where $\mathbf{g}$ is the gravitational acceleration and β is the coefficient of expansion. Gravity can then drive fluid motion, which then advects heat around. Suitably non-dimensionalised, the governing equations for the temperature T and velocity $\mathbf{u}$ are

$$\left.\begin{array}{rcl} T_t + \mathbf{u}\cdot\nabla T &=& \nabla^2 T \\ P^{-1}\left(\omega_t + \mathbf{u}\cdot\nabla\omega\right) &=& R\,T_x\,\nabla^2\omega \\ \omega &=& -\nabla^2\psi \end{array}\right\}, \tag{24.13}$$

There are two parameters in the problem: The Prandtl number, $P = \nu/\kappa$ is the ratio of the fluid kinematic viscosity to the thermal diffusivity (κ). The Rayleigh number R is a measure of the thermal driving force. In deriving (24.13), it is assumed that the density decreases linearly with temperature, and that the fluid speed is much less than the speed of sound, permitting the Boussinesq approximation. Gravity acts in the y-direction.

The problem is known as Rayleigh-Benard convection. The equations above model a variety of flow with industrial relevance such as ventilation of rooms, flow in solar energy collectors, crystal growth in liquids, cooling of electronic parts or flow in a heat exchanger. For example, a typical example would be that of applying uniform heat to the bottom of a box (or the computational domain) while keeping the top and side walls at a constant and lower temperature. For small temperature gradients, measured by the Rayleigh number R, the velocity field is zero and the thermal equilibrium is given by the conductive state. There is a purely conductive solution, with $\mathbf{u} = 0$ and $T = T(y)$. What we expect to happen is that for $R \leq R_c$ the only solution is the conductive solution, but for $R > R_c$ this solution is

unstable, and a flow (convection) develops. We want to investigate the critical value of R, and the convection patterns formed for different values of P, R. As the temperature gradient, or Rayleigh number, is increased beyond a critical value, a circular motion sets in that transports hot fluid from the lower wall in exchange for cooler fluid from the top wall. After a transient period, a steady number of rolls appear. As the Rayleigh number is increased further, the initially two rolls split and four rolls appear, with additional bifurcations observed on increasingly higher Rayleigh numbers.

Our solution plan is to use operator splitting. If T, ψ and ω are known at time t, then we :

1. Advect T
2. Diffuse T
3. Advect ω
4. Diffuse ω
5. Find new ψ and hence new velocity field
6. Derive new boundary condition on ω from new ψ
7. Repeat.

The equations (24.13) written out explicitly, for a slightly more generalised form are :

$$\left.\begin{array}{rcl} T_t + u\,T_x + v\,T_y &=& T_{xx} + T_{yy} \\ \omega_t + u\,\omega_x + v\,\omega_y &=& P(\omega_{xx} + \omega_{yy}) + R\,P\,\left[T_x\cos(\phi) + T_y\sin(\phi)\right] \\ \psi_{xx} + \psi_{yy} &=& -\omega \end{array}\right\}, \tag{24.14}$$

with ϕ representing an inclination angle, with (R, P) defined as follows :

$$R = \frac{g\beta D^3(T_h - T_c)}{\nu\kappa}, \quad P = \frac{\nu}{\kappa} \tag{24.15}$$

where D is a characteristic box size, β the thermal expansion coefficient, (T_c, T_h) temperatures of the cold and hot wall respectively, κ is the thermal diffusivity and g the gravity.

24.5 Molten cores of planets and moons

The final topic, consists of the solution of the Convection Equations in an annulus. This is intended as a simplified (two-dimensional) model of thermally-driven convection between two spheres. This problem is important in the centre of the earth, whose outer core consists of a spherical shell of liquid iron. It also occurs in the recently discovered underground oceans in the moons of the outer planets. We hope our 2-D annular model will capture many of the important features of the problem.

Convection occurs because fluids expand when heated. Their density decreases and they become buoyant. Gravity can then drive fluid motion, which then advects heat around.

Suitably non-dimensionalised, the governing equations for the temperature T, pressure p and velocity $\mathbf{u}$ are

$$\left.\begin{array}{rcl} T_t + \mathbf{u}\cdot\nabla T &=& \nabla^2 T \\ P^{-1}(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) &=& -\nabla p - R\,\hat{g}\,T + \nabla^2\mathbf{u} \\ \nabla\cdot\mathbf{u} &=& 0 \end{array}\right\}, \tag{24.16}$$

There are two parameters in the problem: The Prandtl number, P is the ratio of the fluid kinematic viscosity to the thermal diffusivity. The Rayleigh number R is a measure of the thermal driving force, as defined earlier. In deriving (24.16), it is assumed that the density decreases linearly with temperature, and that the fluid speed is much less than the speed of sound, permitting the Boussinesq approximation. Gravity acts in the $\hat{g}$-direction. In two-dimensions, we may once again eliminate the pressure by taking the curl. The vorticity $\nabla\times\mathbf{u}$ then has a single component ω, related to a stream-function ψ by

$$-\omega = \nabla^2\psi. \tag{24.17}$$

In our annular model, we will consider flow in the domain $1 < r < b$ in terms of polar coordinates (r,θ). The temperature boundary conditions are $T=1$ on $r=1$ and $T=0$ on $r=b$. Gravity acts in the negative $\widehat{r}$-direction. The r- and θ-components of the velocity are respectively ψ_θ/r and $-\psi_r$. The equations for $\psi(r,\theta,t)$ and $T(r,\theta,t)$ reduce to

$$\left.\begin{array}{rcl} rT_t + \psi_\theta T_r - \psi_r T_\theta &=& (rT_r)_r + \frac{1}{r}T_{\theta\theta} \\ P^{-1}(r\omega_t + \psi_\theta\omega_r - \psi_r\omega_\theta) &=& -RT_\theta + (r\omega_r)_r + \frac{1}{r}\omega_{\theta\theta} \end{array}\right\}, \tag{24.18}$$

The problem obviously is very similar to the Rayleigh-Bénard convection considered above. Here too, there is a purely conductive solution, with $\mathbf{u}=0$ and $T=T(r)$. What we expect to happen is that for $R\le R_c$ the only solution is the conductive solution, but for $R>R_c$ this solution is unstable, and a flow (*convection*) develops. We want to investigate the critical value of R, and the convection patterns formed for different values of P, R and maybe b.

25 The Keller-Box method for nonlinear parabolic PDEs

As a case study we consider the incompressible laminar boundary layer equations arising in fluid dynamics. The equations are as follows :

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{25.1a}$$

$$u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{1}{\rho}\frac{\partial P}{\partial x} = \frac{\mu}{\rho}\frac{\partial^2 u}{\partial y^2}, \tag{25.1b}$$

where the variable $(u,\ v)$ denote fluid velocities in x and y directions, P is a known surface pressure field, ρ is the constant density field and μ is the dynamic viscosity. Since the flow is assumed to be incompressible for what follows we take the density ρ and viscosity μ to be fixed given constants (note the kinematic viscosity ν is defined $\nu = \mu/\rho$). These equations to leading order (or zeroth order) can be used to describe the development of the viscous laminar boundary layer over an aerofoil, assuming the x coordinate is along the curved aerofoil body surface (*i.e.* $x = s$)* as shown in Fig. 25.1. In order to account for the

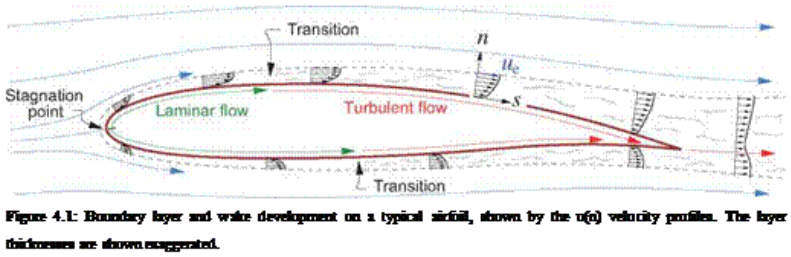

Figure 4.1: Boundary layer and wake development on a typical airfoil, shown by the u(n) velocity profiles. The layer thicknesses are shown exaggerated.

Figure 25.1: Navier-Stokes equations in action: boundary-layers.

growth of the boundary-layer from a zero thickness† at the so-called stagnation point $s = 0$ to some finite thickness, which can be shown to grow as $s^{1/2}$ as we move along the body surface a change in variables and scalings are applied to the above equations, namely :

$$\eta = y\sqrt{\frac{\rho U_e(s)}{\mu s}}\ ;\quad \psi = \sqrt{s\rho\mu U_e(s)}\ f(\eta, s)\ ;\quad x = s\ . \tag{25.2}$$

Observe here that

$$\sqrt{\frac{\mu s}{\rho U_e}} = \delta,$$

is a measure of the viscous layer thickness normal to the surface, hence y is scaled with δ in the definition of η in (25.2). The main advantage of adopting this change of variables is to obtain a coordinate frame such that in computations the boundary layer coordinate (so-called thickness) remains as constant as possible and more importantly it removes the singularity in the equations at $s = 0$ (the stagnation point).

*This approximation suffices provided the local change of curvature of the surface (radius of surface curvature) is $\gg$ the local viscous boundary layer thickness $\delta \approx \sqrt{\nu s/U_e}$.

†The boundary layer is only of zero-thickness for a flat plate, whereas for a curved surface (as shown in Fig. 25.1) the viscous layer has a finite non-zero-thickness at $s = 0$ too.

In the above the function $U_e(s)$ represents the so-called inviscid slip velocity and is usually computed by a separate solution of the inviscid Euler equations, which thus allows one to link the surface pressure variations to $U_e(s)$ variation through the relationship

$$\frac{dP}{ds} = \rho U_e \frac{dU_e}{ds}, \tag{25.3}$$

while ψ represents the so called stream-function which automatically satisfies the first equation in Eqn. 25.1, through the relationships

$$u = \frac{\partial \psi}{\partial y}, \quad v = -\frac{\partial \psi}{\partial x}. \tag{25.4}$$

Hence by a change in variables and coordinates namely (x, y) to (s, η) and usage of Eqn. 25.2, it can be shown that the second equation in Eqn. 25.1 reduces to the following:

$$\frac{\partial^3 f}{\partial \eta^3} + m(1 - \left(\frac{\partial f}{\partial \eta}\right)^2) + cf\frac{\partial^2 f}{\partial \eta^2} = s\left(\frac{\partial f}{\partial \eta}\frac{\partial^2 f}{\partial \eta \partial s} - \frac{\partial^2 f}{\partial \eta^2}\frac{\partial f}{\partial s}\right), \tag{25.5}$$

where

$$m = \frac{s}{U_e}\frac{dU_e}{ds}, \quad c = (1+m)/2, \tag{25.6}$$

and in general $m = m(s)$ due to the dependence of U_e on s.

Observe setting $m = 0$, $c = 1/2$ and $s = 0$ in the above, Eqn. 25.5 then simplifies considerably to the following famous equation known as Blasius's Equation :

$$\frac{d^3 f}{d\eta^3} + \frac{f}{2}\frac{d^2 f}{d\eta^2} = 0. \tag{25.7}$$

The most general boundary conditions on $f(\eta, s)$ that may arise under various scenarios are that at

$$\eta = 0, f(0, s) = 0 \text{ or } f(0, s) = f_w(s) \tag{25.8}$$

with the latter $f_w(s)$ dependence representing a surface transpiration; moreover

$$\frac{\partial f}{\partial \eta}(0, s) = 0\ , \text{ known as the no-slip condition, while as } \eta \to \infty, \frac{\partial f}{\partial \eta}(\eta \to \infty, s) = 1. \tag{25.9}$$

In the above it should be noted that

$$\frac{\partial f}{\partial \eta} = u/U_e, \tag{25.10}$$

the streamwise velocity field.

Equation 25.5 is third-order nonlinear parabolic PDE with s serving as the forward marching coordinate, while η coordinate represents the boundary-value nature of the problem, through having to match boundary conditions at $\eta = 0$ and as $\eta \to \infty$ as we progress forwards in the s-coordinate. The technique we describe is the so-called Box method (Keller & Cebeci, 1972)[‡], and requires rewriting of Eqn. 25.5 as a system of first order PDEs, through

[‡]Keller, H. B. & Cebeci T. 1972, Accurate Numerical Methods for Boundary Layer Flows. II: Two-Dimensional Turbulent Flows, AIAA Journal, vol 10, pp. 1193-1199.

the following:

$$\left.\begin{array}{rcll} F^{(1)} & = & f & (a) \\ F^{(2)} & = & \frac{\partial F^{(1)}}{\partial \eta} & (b) \\ F^{(3)} & = & \frac{\partial F^{(2)}}{\partial \eta} & (c) \end{array}\right\}, \tag{25.11}$$

Hence we may rewrite Eqn. 25.5

$$\frac{\partial F^{(3)}}{\partial \eta} + m(1 - \left(F^{(2)}\right)^2) + cF^{(1)}F^{(3)} = s\left(F^{(2)}\frac{\partial F^{(2)}}{\partial s} - F^{(3)}\frac{\partial F^{(1)}}{\partial s}\right). \tag{25.12}$$

Thus the PDE is written as a system of three first order coupled PDEs, ultimately giving rise to a block tridiagonal system comprising 3×3 blocks.

Observe that setting $s = 0$, (25.12) reduces to requiring the solution of a coupled systems of nonlinear ODEs – whether the viscous boundary layer is of zero, or finite, thickness!

25.1 Discretisation and Newton Linearisation

We next describe the steps to set about discretising Eqn. 25.12. We shall break up the process by considering the treatment of the following terms:

$$\left.\begin{array}{lll} F^{(\alpha)}\,, & \frac{\partial F^{(\alpha)}}{\partial \eta}\,, & \frac{\partial F^{(\alpha)}}{\partial s} \\ F^{(\alpha)}F^{(\beta)}\,, & F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial \eta}\,, & F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial s} \end{array}\right\}, \tag{25.13}$$

where the superscripts (α, β) may have values $1,\ 2$ or 3, with reference to Eqn. 25.5 and we assume that solutions have already been computed at the previous $i-1$ data point for all j points (see Fig. 25.2). We define the spatial step size in s as $\Delta s = h$ and that

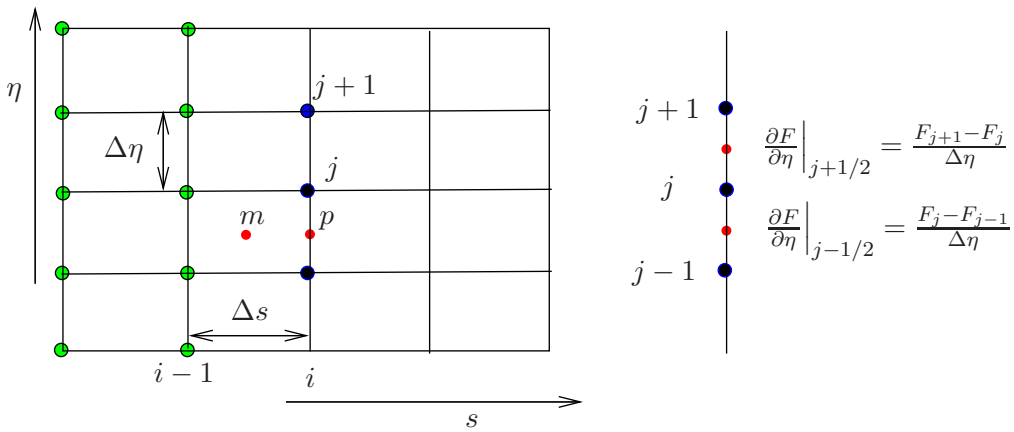

Figure 25.2: Discretised grid notation. We assume solutions at earlier data planes $i-1$ are known and we seek to compute a solution at the i'th data plane for all j grid nodes.

in η as $\Delta\eta = k$. There are two possibilities to consider, that of developing a semi-implicit discretisation or a fully-implicit one – by this we mean that in the semi-implicit discretisation the η-differencing is carried out at the mid-point "m" of the box shown in Fig. 25.2 while for a fully implicit discretisation the η-differencing is carried out at the mid-point "p" in Fig. 25.2. We demonstrate the semi-implicit form in what follows, hence each of the variables in expression 25.13 are developed as follows:

1. $F^{(\alpha)}$ discretisation :

$$F^{(\alpha)} = \frac{1}{4}\left(F^{(\alpha)}_{i,j-1} + F^{(\alpha)}_{i,j} + F^{(\alpha)}_{i-1,j-1} + F^{(\alpha)}_{i-1,j}\right). \tag{25.14}$$

Here we may combine the $i-1$ known data points in the form

$$F^{(\alpha)}_{i-1,j-1/2} \equiv F^{(\alpha)}_{i-1,j-1} + F^{(\alpha)}_{i-1,j}. \tag{25.15}$$

Next since we wish to employ a Newton linearisation procedure for the nonlinear terms, we assume each $F^{(\alpha)}$ term can be written in the form

$$F^{(\alpha)}_{i,j} = \bar{F}^{(\alpha)}_{i,j} + \hat{F}^{(\alpha)}_{i,j},$$

where $\bar{F}^{(\alpha)}_{i,j}$ is a known value computed from some earlier computation while $\hat{F}^{(\alpha)}_{i,j}$ represents the residual which on having obtained a converged solution must tend to zero. Thus we may write Eqn. 25.14, as follows:

$$F^{(\alpha)} = \frac{1}{4}\left(\bar{F}^{(\alpha)}_{i,j-1/2} + \bar{F}^{(\alpha)}_{i-1,j-1/2} + \hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right). \tag{25.16}$$

Which on setting $\bar{F}^{(\alpha)}_{i-1/2,j-1/2} \equiv \bar{F}^{(\alpha)}_{i,j-1/2} + \bar{F}^{(\alpha)}_{i-1,j-1/2}$, followed by a Newton linearisation step, Eqn. 25.14 may thus be summarised to

$$F^{(\alpha)} = \frac{1}{4}\left(\bar{F}^{(\alpha)}_{i-1/2,j-1/2} + \hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right), \tag{25.17}$$

2. $\frac{\partial F^{(\alpha)}}{\partial \eta}$ discretisation :
This follows the same logic as used for the $F^{(\alpha)}$ term, namely

$$\frac{\partial F^{(\alpha)}}{\partial \eta} = \frac{1}{2k}\left(F^{(\alpha)}_{i,j} - F^{(\alpha)}_{i,j-1} + \bar{F}^{(\alpha)}_{i-1,j} - \bar{F}^{(\alpha)}_{i-1,j-1}\right), \tag{25.18}$$

noting the factor of $1/2$ as we average the derivative over the box at point i and $i-1$. Next the Newton linearisation procedure leads to

$$\frac{\partial F^{(\alpha)}}{\partial \eta} = \frac{1}{2k}\left(\bar{F}^{(\alpha)}_{i,j} - \bar{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j} - \hat{F}^{(\alpha)}_{i,j-1} + \bar{F}^{(\alpha)}_{i-1,j} - \bar{F}^{(\alpha)}_{i-1,j-1}\right), \tag{25.19}$$

which on collection of terms then gives

$$\frac{\partial F^{(\alpha)}}{\partial \eta} = \frac{1}{2k}\left(\bar{F}^{(\alpha)}_{i-1/2,j} - \bar{F}^{(\alpha)}_{i-1/2,j-1} + \hat{F}^{(\alpha)}_{i,j} - \hat{F}^{(\alpha)}_{i,j-1}\right), \tag{25.20}$$

3. $\frac{\partial F^{(\alpha)}}{\partial s}$ discretisation :
This follows the same logic as used above, namely

$$\frac{\partial F^{(\alpha)}}{\partial s} = \frac{1}{2h}\left(F^{(\alpha)}_{i,j} + F^{(\alpha)}_{i,j-1} - \bar{F}^{(\alpha)}_{i-1,j} - \bar{F}^{(\alpha)}_{i-1,j-1}\right). \tag{25.21}$$

Next Newton linearisation step gives:

$$\frac{\partial F^{(\alpha)}}{\partial s} = \frac{1}{2h}\left(\bar{F}^{(\alpha)}_{i,j-1/2} - \bar{F}^{(\alpha)}_{i-1,j-1/2} + \hat{F}^{(\alpha)}_{i,j} + \hat{F}^{(\alpha)}_{i,j-1}\right). \tag{25.22}$$

4. $F^{(\alpha)}F^{(\beta)}$ discretisation :
This follows from Eqn. 25.17, hence

$$F^{(\alpha)}F^{(\beta)} = \frac{1}{16}\left(\bar{F}^{(\alpha)}_{i-1/2,j-1/2} + \hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right)\left(\bar{F}^{(\beta)}_{i-1/2,j-1/2} + \hat{F}^{(\beta)}_{i,j-1} + \hat{F}^{(\beta)}_{i,j}\right), \tag{25.23}$$

which on a bit of manipulation may be written

$$\begin{aligned} F^{(\alpha)}F^{(\beta)} = \tfrac{1}{16}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\bar{F}^{(\beta)}_{i-1/2,j-1/2} &+ \tfrac{1}{16}\bar{F}^{(\beta)}_{i-1/2,j-1/2}\left(\hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right) \\ &+ \tfrac{1}{16}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\left(\hat{F}^{(\beta)}_{i,j-1} + \hat{F}^{(\beta)}_{i,j}\right), \end{aligned} \tag{25.24}$$

where we set the quadratic terms involving the residuals to zero.

5. $F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial s}$ discretisation :
Noting Eqns. 25.17 and 25.22 gives

$$F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial s} = \frac{1}{8h}\left(\bar{F}^{(\alpha)}_{i-1/2,j-1/2} + \hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right)\left(\bar{F}^{(\beta)}_{i,j-1/2} - \bar{F}^{(\beta)}_{i-1,j-1/2} + \hat{F}^{(\beta)}_{i,j} + \hat{F}^{(\beta)}_{i,j-1}\right). \tag{25.25}$$

Hence by a bit of manipulation may be reduced to the following

$$\begin{aligned} \tfrac{1}{8h}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\left(\bar{F}^{(\beta)}_{i,j-1/2} - \bar{F}^{(\beta)}_{i-1,j-1/2}\right) &+ \tfrac{1}{8h}\left(\bar{F}^{(\beta)}_{i,j-1/2} - \bar{F}^{(\beta)}_{i-1,j-1/2}\right)\left(\hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right) \\ &+ \tfrac{1}{8h}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\left(\hat{F}^{(\beta)}_{i,j} + \hat{F}^{(\beta)}_{i,j-1}\right). \end{aligned} \tag{25.26}$$

6. $F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial \eta}$ discretisation :
Noting Eqns. 25.17 and 25.20 gives

$$F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial \eta} = \frac{1}{8k}\left(\bar{F}^{(\alpha)}_{i-1/2,j-1/2} + \hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right)\left(\bar{F}^{(\beta)}_{i-1/2,j} - \bar{F}^{(\beta)}_{i-1/2,j-1} + \hat{F}^{(\beta)}_{i,j} - \hat{F}^{(\beta)}_{i,j-1}\right), \tag{25.27}$$

Hence by a bit of manipulation may be reduced to the following

$$\begin{aligned} \tfrac{1}{8k}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\left(\bar{F}^{(\beta)}_{i-1/2,j} - \bar{F}^{(\beta)}_{i-1/2,j-1}\right) &+ \tfrac{1}{8k}\left(\bar{F}^{(\beta)}_{i-1/2,j} - \bar{F}^{(\beta)}_{i-1/2,j-1}\right)\left(\hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right) \\ &+ \tfrac{1}{8k}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\left(\hat{F}^{(\beta)}_{i,j} - \hat{F}^{(\beta)}_{i,j-1}\right). \end{aligned} \tag{25.28}$$

As an example let us consider discretising Eqn. 25.11c, namely

$$\frac{\partial F^{(2)}}{\partial \eta} - F^{(3)} = 0. \tag{25.29}$$

Using Eqns. 25.17 and 25.20 we get the following:

$$\frac{1}{2k}\left(\bar{F}^{(2)}_{i-1/2,j} - \bar{F}^{(2)}_{i-1/2,j-1} + \hat{F}^{(2)}_{i,j} - \hat{F}^{(2)}_{i,j-1}\right) - \frac{1}{4}\left(\bar{F}^{(3)}_{i-1/2,j-1/2} + \hat{F}^{(3)}_{i,j-1} + \hat{F}^{(3)}_{i,j}\right) = 0. \tag{25.30}$$

A bit of simplification, and using the same procedure on Eqn. 25.11b thus gives

$$\hat{F}^{(2)}_{i,j} - \hat{F}^{(2)}_{i,j-1} - \frac{k}{2}\left(\hat{F}^{(3)}_{i,j-1} + \hat{F}^{(3)}_{i,j}\right) = \bar{F}^{(2)}_{i-1/2,j} - \bar{F}^{(2)}_{i-1/2,j-1} - \frac{k}{2}\bar{F}^{(3)}_{i-1/2,j-1/2}\,, \tag{25.31}$$

$$\hat{F}^{(1)}_{i,j} - \hat{F}^{(1)}_{i,j-1} - \frac{k}{2}\left(\hat{F}^{(2)}_{i,j-1} + \hat{F}^{(2)}_{i,j}\right) = \bar{F}^{(1)}_{i-1/2,j} - \bar{F}^{(1)}_{i-1/2,j-1} - \frac{k}{2}\bar{F}^{(2)}_{i-1/2,j-1/2} \,. \tag{25.32}$$

Observe that the right hand side expressions are assumed given, or been computed from a previous computation, while the solution procedure is to solve for the left hand side variables $(\hat{F}^{(1)},\ \hat{F}^{(2)},\ \hat{F}^{(3)})$. A more complicated expression arises for Eqn. 25.12, but the basic elements are similar and the logic can be easily automated on a computer to allow setting up of the matrix of unknowns in a block tridiagonal form. Hence, a matrix problem $[A]\mathcal{F} = \mathcal{R}$ for a given right hand side of vectors $\mathcal{R}$ (e.g.. terms on right hand side of equals sign in Eqn. 25.31, 25.32, etc.) is thus set up for all the (i, j) variable unknown fields $\mathcal{F}$ by using the discretising stencils above, and supplemented by enforcement of boundary conditions at $\eta = 0$ and at some determined $\eta = \eta_{max}$ upper domain to mimic the condition $\eta \to \infty$. Since the matrix can be arranged in a 3×3 block tri-diagonal form, the solution process for the residual vectors $\mathcal{F}$ can be accomplished very efficiently using a block tri-diagonal form of Thomas's algorithm. The method of solution is thus as follows:

1. Guess an initial trial solution for the $\bar{F}$ vectors, call this $\bar{F}_{OLD}$ (say).
2. Setup the matrix and solve for the $\mathcal{F} \equiv \hat{F}$ residuals.
3. Update the initial $\bar{F}$ vectors by addition of the newly computed residual fields, i.e. $\bar{F}_{NEW} = \bar{F}_{OLD} + \hat{F}$, and recompute a corrected vector $\mathcal{R}$.
4. Repeat stages 2 and 3 until the residuals in th entire domain meet some convergence criterion.
5. Advance solution the next $i + 1$'th position on your grid, and use the previous i'th-solution for a trial $\hat{F}$ field.
6. Repeat steps 2–4, etc.

The hardest and most time-consuming aspect is essentially is correctly setting up the matrix $[A]$, and determining an initial trial solution in part (1) of the above steps; since the 3 equations are coupled. However once a converged solution is obtained, it then becomes much easier, as one can always use the earlier converged solution as the initial estimate for consequent computations at the next $i + 1$'th data plane.

The technique described above is the most robust, in terms of being able to deal with non-linear terms effectively, since the method involves a coupled Newton linearisation process. It is also second-order accurate in both the (s, η) directions. A fully implicit scheme may also be constructed involving taking averages at the midpoint "p" of the box shown in Fig. 25.2; doing so makes the scheme only first-order accurate in s, but there may well be situations where the semi-implicit scheme shows a weak numerical instability, but treating the equations fully implicitly overcomes such numerical problems. Of course a way to build back in a second order accurate scheme would by way of using two previous s-positions, namely the $i,\ i - 1$ and $i - 2$ data points.

Alternate approaches of course do exist – one particular simpler form is the so-called lagged coefficient approach, whereby coefficients involving nonlinear terms are based on the previous $(i - 1)$'th position, thus effectively solving a linear system of equations. The solution field comprising the coefficients is then subsequently updated and computations repeated iteratively until no further changes arise in the full solution. The accuracy of the lagging

approach is though no-longer second-order accurate, and it is also known to fail in certain situations, whereas the coupled newton linearisation approach outlined in a number of test cases has been found to be superior.

For more details on the lagging and alternative approaches, see chapter 7 of *Tannehill. Anderson & Pletcher, Computational Fluid Mechanics and Heat Transfer*.

26 Multistage Methods

Linear multistep formulas represent one extreme – one function evaluation per time step. Multistage methods represent another extreme: the function $f(u,t)$ is evaluated at any number of intermediate stages in the process of getting from v^n to v^{n+1}, but those stages are never used again. As a result, multistage methods are often more stable than linear multistep formulas, but at the price of more work per time step. They tend to be easier to implement than linear multistep formulas, since starting values are not required, but harder to analyse. Multistage methods were first developed by Runge, Kutta and Heun, and thus bear their names. Runge-Kutta formulas tend to be "not unique" in a sense that if we define the number of stages s and the order of accuracy p, then for most values of s there are infinitely many Runge-Kutta formulas with the maximal order of accuracy p. As an example, a two-stage explicit Runge-Kutta method is given by

$$\begin{aligned} v^* &= v^n + \frac{1}{2}\Delta t\ f(v^n) \\ v^{n+1} &= v^n + \Delta t\ f(v^*). \end{aligned} \tag{26.1}$$

In the first stage an intermediate value is generated which approximates $u(t_{n+1/2})$ via Euler's method. In the second step the function f is evaluated at this midpoint to estimate the slope over the full time step. Since this represents a centered approximation to the derivative we might hope for second order accuracy. Combining the two steps above, we can write

$$v^{n+1} = v^n + \Delta t\ f\left(v^n + \frac{1}{2}\Delta t\ f(v^n)\right). \tag{26.2}$$

Viewed as a one-step explicit method that can be shown to be second-order accurate.

A very popular higher-order Runge-Kutta scheme is the four-stage fourth-order Runge-Kutta method, generally referred to as the **RK4** or the *classic Runge-Kutta* method. This is of *order*-(h^4) accuracy and requires 4 inter-mediate steps. The method for a step size h is given as follows :

$$v_{n+1} = v_n\ +\ \frac{h}{6}\ (k_1 + 2k_2 + 2k_3 + k_4), \tag{26.3}$$

where for $n = 0,\ 1,\ 2,\ 3,\ \ldots$, the $k_{()}$-terms in Eqn. 26.3 above are evaluated as follows*

$$\begin{aligned} k_1 &= f(t_n, v_n)\ , \\ k_2 &= f(t_n + \frac{h}{2}, v_n + h\frac{k_1}{2})\ , \\ k_3 &= f(t_n + \frac{h}{2}, v_n + h\frac{k_2}{2})\ , \\ k_4 &= f(t_n + h, v_n + hk_3)\ . \end{aligned} \tag{26.4}$$

To explore stability of the RK methods, consider the model equation

$$u_t = f(u,t) = au, \quad \text{say.}$$

*see Wikipedia or Griffiths & Higham (chapter 9.1) for details.

For the second-order Runge-Kutta scheme we obtain

$$\begin{aligned} F_o &= a\ \Delta t\ v^n\ , \\ F_1 &= \Delta t(av^n + \frac{1}{2}a^2\Delta t\ v^n) \end{aligned} \tag{26.5}$$

Hence

$$\begin{aligned} v^{n+1} &= v^n + a\Delta t\ (1 + \frac{1}{2}a\Delta t)v^n\ , \\ &= v^n(1 + a\ \Delta t + \frac{a^2\ \Delta t^2}{2}) \end{aligned} \tag{26.6}$$

For stability, we require

$$\left|1 + a\ \Delta t + \frac{a^2\ \Delta t^2}{2}\right| \leq 1.$$

A convenient way to obtain the stability boundary of the second-order Runge-Kutta method is to set

$$1 + a\ \Delta t + \frac{a^2\ \Delta t^2}{2} = e^{i\theta},$$

and find the complex roots $a\Delta t$ of this polynomial for different values of θ in the interval $0 \leq \theta \leq 2\pi$.

An analogous analysis for the fourth-order Runge-Kutta method yields the equivalent polynomial condition

$$\left|1 + a\ \Delta t + \frac{a^2\ \Delta t^2}{2} + \frac{a^3\ \Delta t^3}{6} + \frac{a^4\ \Delta t^4}{24}\right| \leq 1.$$

Note the fourth-order accuracy of the method. The stability diagram is obtained by finding the roots of the fourth-order polynomial with complex coefficients

$$1 + a\ \Delta t + \frac{a^2\ \Delta t^2}{2} + \frac{a^3\ \Delta t^3}{6} + \frac{a^4\ \Delta t^4}{24} - e^{i\theta} = 0,$$

in the interval $0 \leq \theta \leq 2\pi$. The stability regions for the second-order and fourth-order Runge-Kutta methods are shown in Fig. 26.1.

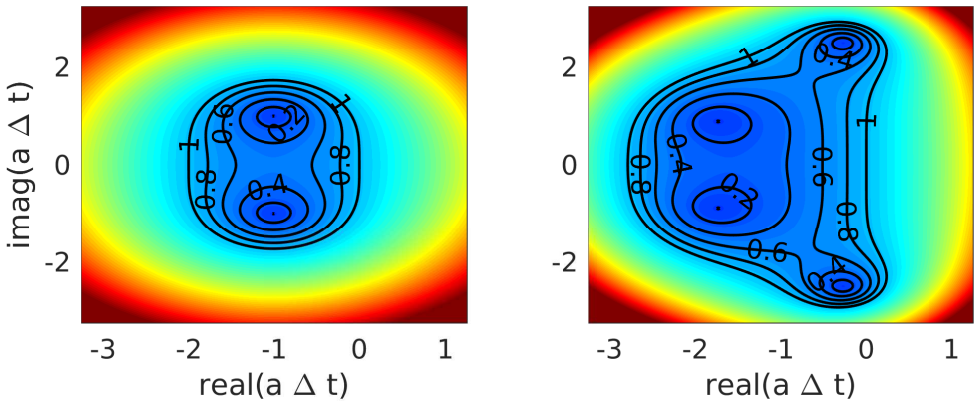

Figure 26.1: Stability region (contours ≤ 1) for the second- and fourth-order Runge-Kutta method.

The most expensive part of numerical solution is the function evaluation, with the **RK4** requiring 4 function evaluations per time step. The number of steps required to reach the

final integration time T is directly related to the cost of the computation. Hence, both the stability characteristics and the accuracy come into play in establishing cost-effectiveness of a numerical method. Clearly the **RK4** has superior stability as well as improved accuracy properties compared to the second-order scheme (26.1). These characteristics, together with its ease of programming, have made the fourth-order Runge-Kutta method one of the most popular schemes for the solution of ordinary and partial differential equations.

Example :

The Euler equation in Cartesian coordinates can be cast into a vectorial form:

$$\frac{\partial \mathbf{U}}{\partial t} + \sum_{i=1}^{3} \frac{\partial \mathbf{E}_i}{\partial x_i}, \tag{26.7}$$

where $\mathbf{U}$ is the unknown vector and $\mathbf{E}_i$ are the inviscid fluxes corresponding to the x_i directions, *i.e.* (x, y, z) or see Eqn. 17.9 and (17.13). The Runge-Kutta algorithm would be applied to the temporal derivative, while the right hand side terms be treated using an appropriate finite difference discretisation (second, fourth and higher-order).

26.1 Compact differences

When higher-order accurate schemes are necessary, the stencil width in general becomes increasingly large, as shown in §3.4 for both central and one-sided schemes. Wide stencils, however, are problematic for various reasons. Near boundaries they have to be modified resulting in non-central schemes, even though throughout the computational domain they are central[†]. An alternative class of FDs yielding higher-order accurate results while maintaining a narrower stencil are known as **compact difference** or **CD**-schemes.

There are various ways of constructing CD schemes. We illustrate one possible way via an example; see the paper by Lele (1992)[‡] for a more general approach. Suppose we wish to solve the ODE

$$u'' + bu = f(x). \tag{26.8}$$

The FD for the second derivative is

$$U''_n = \frac{U_{n-1} - 2U_n + U_{n+1}}{h^2} - \frac{1}{12} h^2 U^{iv}_n + O(h^4). \tag{26.9}$$

Next, make use of (26.8) to get the U^{iv}_n derivative, namely

$$u^{iv} = f'' - bu'', \quad \text{assume for simplicity constant } \; b. \tag{26.10}$$

Hence, it follows (26.9) becomes

$$U''_n = \frac{U_{n-1} - 2U_n + U_{n+1}}{h^2} - \frac{1}{12} h^2 (f'' - bu'') + O(h^4). \tag{26.11}$$

[†]Recall fourth-order accurate FD has a 5-point stencil, while the sixth-order FD has a 7-point stencil. The stencil size thus grows as $p + 1$ for the p-th order FD method

[‡]Lele, S. K. (1992) *Compact finite difference schemes with spectral-like resolution*. J. Comp. Physics, **133**,(1) pp. 16-42.

Hence, the original ODE (26.8) becomes

$$u_n'' + bu_n - f \equiv \frac{U_{n-1} - 2U_n + U_{n+1}}{h^2} + bU_n - \frac{1}{12}h^2(f_n'' - bu_n'') - f_n + O(h^4). \tag{26.12}$$

Discretising the $u_n'',\ f_n''$ terms then gives

$$\frac{U_{n-1} - 2U_n + U_{n+1}}{h^2} + bU_n - \frac{1}{12}(f_{n-1} - 2f_n + f_{n+1}) + \frac{1}{12}b(U_{n-1} - 2U_n + U_{n+1}) - f_n = 0. \tag{26.13}$$

Finally collecting terms we obtain

$$\left(\frac{1}{h^2} + \frac{b}{12}\right)(U_{n-1} + U_{n+1}) + U_n\left(\frac{5b}{6} - \frac{2}{h^2}\right) = \frac{1}{12}(f_{n-1} + 10f_n + f_{n+1}), \tag{26.14}$$

with a truncation term $O(h^4)$.

We note the following points :

1. Unlike the straight-forward FD schemes we have employed until now, observe the fourth-order method is **tailored** to the equation which is being solved. Since we effectively used the original ODE (26.8) to develop the FD form (26.14).
2. The discretising stencil requires only 3 nodal values (*i.e.* the $n-1,\ n,\ n+1$ points) and is fourth-order accurate.
3. It is possible to take the approach further and develop sixth-order accurate formulae which still involve only the three-point stencil.
4. The computational time, if we were solving a tridiagonal type matrix using Thomas's algorithm would not incur any additional overhead as the accuracy of the approach is increased. Similar nice feature arises in explicit approaches, with a fixed number of terms to be manipulated.
5. Can be directly applied to nonlinear equations too.

26.1.1 Elliptic BVPs and 2D spatial discretisation

Consider a 2D Poisson equation

$$u_{xx} + u_{yy} = f(x, y). \tag{26.15}$$

As earlier, use Taylor series, assuming for simplicity fixed h step size in both (x, y) directions, and obtain

$$u_{xx} + u_{yy} \ = \ \frac{1}{h^2}(\delta_x^2 + \delta_y^2)U^{(n)} - \frac{h^2}{12}(u_{xxxx} + u_{yyyy}) + O(h^4). \tag{26.16}$$

The $O(h^2)$ terms are accounted for by differentiating (26.15), namely

$$u_{xxxx} = f_{xx} - u_{xxyy}; \quad u_{yyyy} = f_{yy} - u_{yyxx}. \tag{26.17}$$

Hence,

$$u_{xx}+u_{yy}-f \equiv \frac{1}{h^2}(\delta_x^2+\delta_y^2)U^{(n)}-\frac{h^2}{12}(f_{xx}+f_{yy}-2u_{xxyy})+O(h^4), \tag{26.18}$$

$$u_{xx}+u_{yy}-f \equiv \frac{1}{h^2}(\delta_x^2+\delta_y^2)U^{(n)}-\left(1+\frac{h^2}{12}(\delta_x^2+\delta_y^2)\right)f^{(n)}+\frac{2h^2}{12}u_{xxyy}. \tag{26.19}$$

The mixed derivative u_{xxyy} has the stencil

$$U^{(n)}_{xxyy}=\frac{1}{h^4}\begin{pmatrix}1&-2&1\\-2&4&-2\\1&-2&1\end{pmatrix}U_{i,j}+O(h^2), \tag{26.20}$$

which on using and manipulation of (26.19), we obtain the compact fourth-order accurate formula for (26.15)

$$\frac{1}{6h^2}\begin{pmatrix}1&4&1\\4&-20&4\\1&4&1\end{pmatrix}U_{i,j}=\frac{1}{12}\begin{pmatrix}0&1&0\\1&8&1\\0&1&0\end{pmatrix}f_{i,j}. \tag{26.21}$$

As a way of comparison with the standard forms, we would have had the following

- Second order :
$$\frac{1}{h^2}\begin{pmatrix}0&1&0\\1&-2&1\\0&1&0\end{pmatrix}U_{i,j}=f_{i,j}.$$
- Fourth-order :
$$\frac{1}{h^2}\begin{pmatrix}0&0&-1/12&0&0\\0&0&4/3&0&0\\-1/12&4/3&5&4/3&-1/12\\0&0&4/3&0&0\\0&0&-1/12&0&0\end{pmatrix}U_{i,j}=f_{i,j}.$$

26.2 Semi-discretisation and the method of lines (MoL)

Time-dependent partial differential equations can readily be converted to a system of ordinary differential equations by using finite difference approximations for the spatial derivatives. The process, known as **semi-discretisation**, leads to a system of weakly coupled ordinary differential equations. It does this using the intermediate step where we discretise in space *only* but keep time continuous. This system can then be solved, *i.e.* with Runge-Kutta methods above say, and analysed using the techniques for systems of ordinary differential equations.

Consider, for example, the one-dimensional diffusion equation $u_t=u_{xx}$ on a finite domain $[-L,L]$ and homogeneous boundary conditions $u(\pm L)=0$, and some given initial state at $t=t_o$, say $u(x,t_o)=f(x)$. We will discretise the spatial derivative using a second-order central difference approximation to arrive at a system of coupled ordinary differential

equations governing the temporal evolution of the solution value $U_j(t)$ at each spatial grid point x_j. We get (assuming an equi-spaced spatial grid)

$$\frac{d}{dt}U_j = \frac{1}{h^2}(U_{j+1} - 2U_j + U_{j-1})$$

or in matrix form

$$\frac{d}{dt}\mathbf{U} = \mathbf{AU} = \frac{1}{h^2}\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}\begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{N-1} \\ U_N \end{pmatrix} \tag{26.22}$$

Since (26.22) involves the simultaneous solution of N coupled ODEs, it follows N initial conditions are required for their solution – which arises from the discretised version of the initial condition at $t = t_o$ say $f(x_j)$.

In the case, where the PDE is nonlinear, we have to discretise a system of nonlinear algebraic equations. This would typically be solved by an iteration procedure using Newton's method and usage of the Jacobian matrix of the nonlinear algebraic system.

As we have seen, the spectrum of the matrix $\mathbf{A}$ is instrumental in the analysis of numerical stability of the scheme. In the present case the spectrum of $\mathbf{A}$ can be determined analytically, taking advantage of the special Toeplitz structure of its coefficients, giving the eigenvalues for a $N \times N$ Toeplitz matrix

$$\lambda_j = b + 2\sqrt{ca}\cos(\frac{j\pi}{N+1}), \quad j = 1, 2, \ldots, N$$

where $b = -2,\ a = c = 1$ in (26.22).

Observe complex valued eigenvalues if arising, then the location of these spectra in the complex plane helps in identifying dissipative and dispersive effects in the discretised representation of the PDE. For example think what differences arise in the eigenvalues for the following :

$$\begin{aligned} u_t &= u_x \quad \text{advection equation;} \\ u_t &= u_{xx} \quad \text{diffusion equation above;} \\ u_t &= i\, u_{xx} \quad \text{Schrodinger equation.} \end{aligned} \tag{26.23}$$

27 Further Reading

Google computational PDES! you will come across many resources on the subject.

Programming Language Critique : Python/Matlab ok for small code testing/ideas construction unless know that **NumPy** or underlying libraries-called are pre-compiled binary/C++/etc. for fast computation and know precisely, limitations of usage of pre-written software.
Pure Python script not recommended :: Heavier computational resource than C++, C, Fortran *i.e.* compiled languages. Also make usage of Parallel programming (MPI, OpenMP).

Alternative Numerical Discretisations :

1. Spectral Methods: Powerful if domain is simple and the solution smooth.
2. Finite Element Method (FEM) : Popular due to flexibility to handle complex surfaces/geometries.
3. Spectral Element Method (SEM) : The SEM is a formulation of the FEM that uses high-degree piecewise polynomials for very high accuracy and resolution.
4. Finite Volume: Popular in computational fluid dynamics (CFD) and complex surfaces/geometries.
5. Runge-Kutta Methods : (Explicit/Implicit).
6. Method of Lines (MoL).
7. Compact Differences.

Good Reading :

1. Compact Differences : An Introduction to Compact Finite Differences (pptx) by Andrew Rees, University of Bath.
2. Riemann Problem: Eleuterio F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics : A Practical Introduction*. Fluids: Navier-Stokes Eqns. Conservation Laws, Numerical Techniques (shock waves etc.)
3. Riemann Problem: LeVeque R. J., *Finite Volume Methods for Hyperbolic Problems*.
4. *High-resolution high-order upwind compact scheme-based numerical computation of natural convection flows in a square cavity*, by Zhao et al. (2016) International Journal of Heat and Mass Transfer 98 (2016) 313-328.
5. *An optimized dispersion-relation-preserving combined compact difference scheme to solve advection equations*, by Yu et al. Journal of Computational Physics 300 (2015) 92-115.
6. *Numerical Methods in Finance :* wwwf.imperial.ac.uk/ ajacquie
7. Option Pricing under a Heston Volatility model using ADI schemes
8. Practical Problems in the Numerical Solution of PDE's in Finance, by Gianluca Fusai et al. (2002).
9. *Astrophysical Hydrodynamics* : Yosuke Mizuno, Shanghai Jiao Tong University. Graduate lecture, SJTU, 2021. https://web.tdli.sjtu.edu.cn/mizuno/astrophysical-hydrodynamics