M345N10: Computational Partial Differential Equations

M. S. Mughal, 2020-2021

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

Lecture 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,

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{in} \quad 0 < x < a, \qquad 0 < y < b, \tag{1.1}$$

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

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

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. 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. 1.1a,b, however if the boundaries or boundary conditions become complex (Fig. 1.1c,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.

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 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

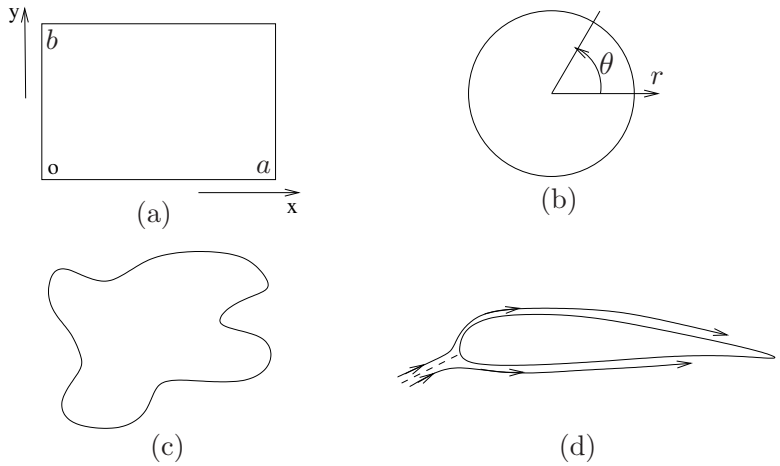

Figure 1.1: Examples of Boundary Value problems (BVPs).

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 practicionar 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.

Lecture 2: Finite Difference Methods (FDMs)

How might one solve a PDE numerically? We can only calculate a finite number of values, so we begin by defining a *grid of points* on which we will seek the solution. For now, let's a 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. (1.2). We consider the points $x_n \equiv a+(n-1)h$ for $n = 1 \ldots N$. We shall seek an approximation U_n to the exact solution on the grid points, $u_n \equiv u(x_n)$.

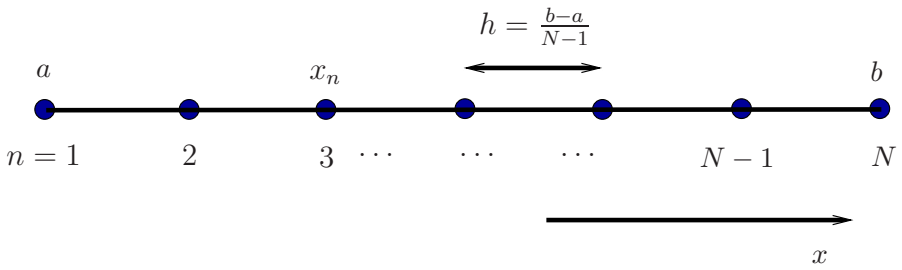

Figure 1.2: Discretised Grid.

Using suitable finite difference **(F-D)** approximations, we can represent any system of PDEs in continuous variables as a set of algebraic equations for a finite list of unknown 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, $\partial u/\partial x \equiv u_x$, $\partial^2 u/\partial x^2 \equiv u_{xx}$, $\partial^2 u/\partial x \partial y \equiv u_{xy}$, $\ldots$, 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

$$\frac{\partial u}{\partial x}|_n \approx \frac{u(x_n+h)-u(x_n)}{h} = \frac{U_{n+1}-U_n}{h}, \tag{1.4}$$

or we may equally well approximate the derivative as follows:

$$\frac{\partial u}{\partial x}|_n \approx \frac{u(x_n)-u(x_n-h)}{h} = \frac{U_n-U_{n-1}}{h}, \tag{1.5}$$

a third choice, namely centred about x_n is given by

$$\frac{\partial u}{\partial x}|_n \approx \frac{u(x_n+h)-u(x_n-h)}{2h} = \frac{U_{n+1}-U_{n-1}}{2h}. \tag{1.6}$$

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. 1.3), we obtain the general

setup*

$$\frac{du}{dx}|_n \approx aU_{n-1} + bU_n + cU_{n+1} \tag{1.7}$$

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.

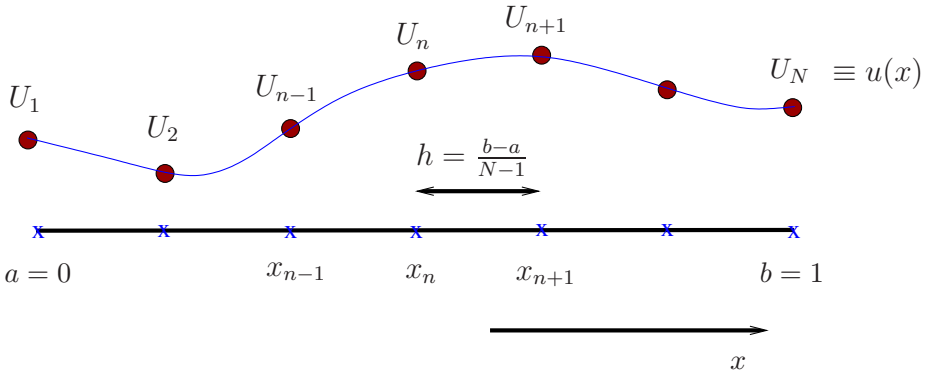

Figure 1.3: Sketch of a equi-spaced grid and a discrete representation of a continuous function $u(x)$ on this grid,

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.

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

Our first Finite Difference Method

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

$$u' \equiv \frac{du}{dx} = -u \ \text{ in } \ x > 0 \ \text{ with } \ u(0) = 1. \tag{1.8}$$

Choosing a step-length and grid as shown in Fig. 1.3, we could represent the ODE fairly accurately as

$$\frac{U_{n+1} - U_{n-1}}{2h} = -U_n, \qquad \text{with} \quad U_1 = 1. \tag{1.9}$$

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 $U_2 = e^{-h}$.

$$U_{n+1} = -2hU_n + U_{n-1} \qquad \text{for } n \geqslant 1, \text{ with} \quad U_1 = 1, \quad U_2 = e^{-h}. \tag{1.10}$$

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. (1.16)

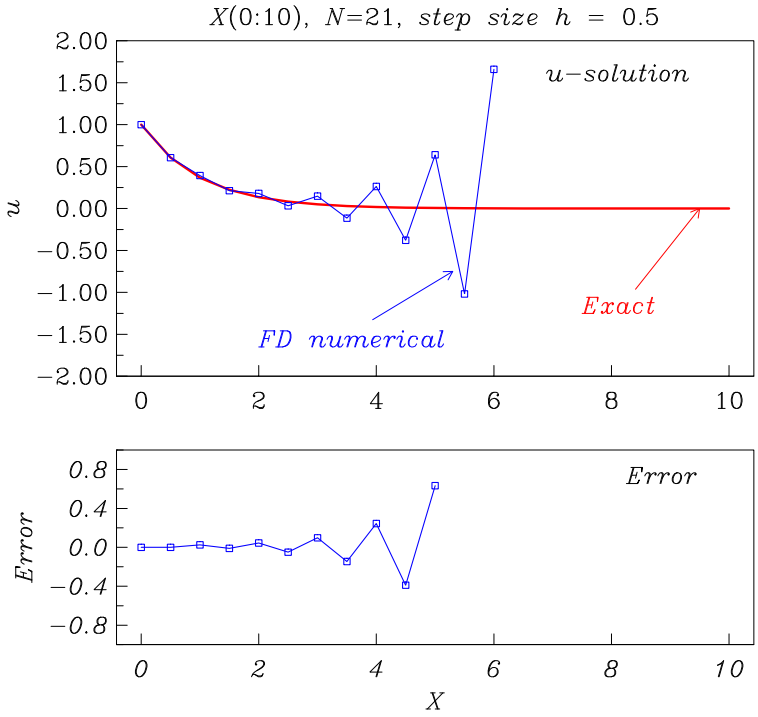

Figure 1.4: Solution of $u' = -u$ with $u(0) = 1$, N=21 grid points.

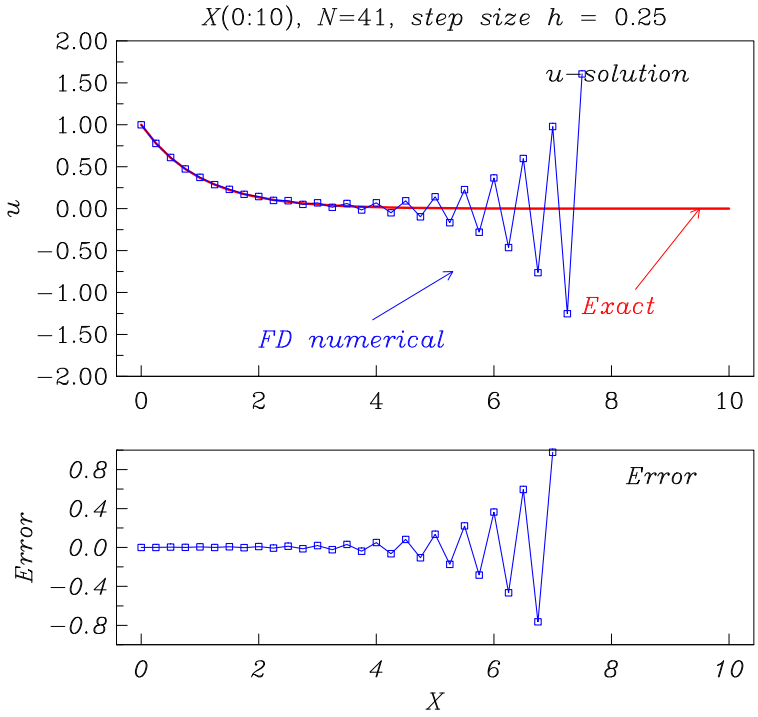

Figure 1.5: Solution of $u' = -u$ with $u(0) = 1$, N=41 grid points.

Typical results of improved Finite Difference Method

Discretisation:

$$\frac{U_{n+1} - U_{n-1}}{2h} = -\frac{U_{n+1} + U_{n-1}}{2}, \qquad \text{with} \quad U_1 = 1. \tag{1.11}$$

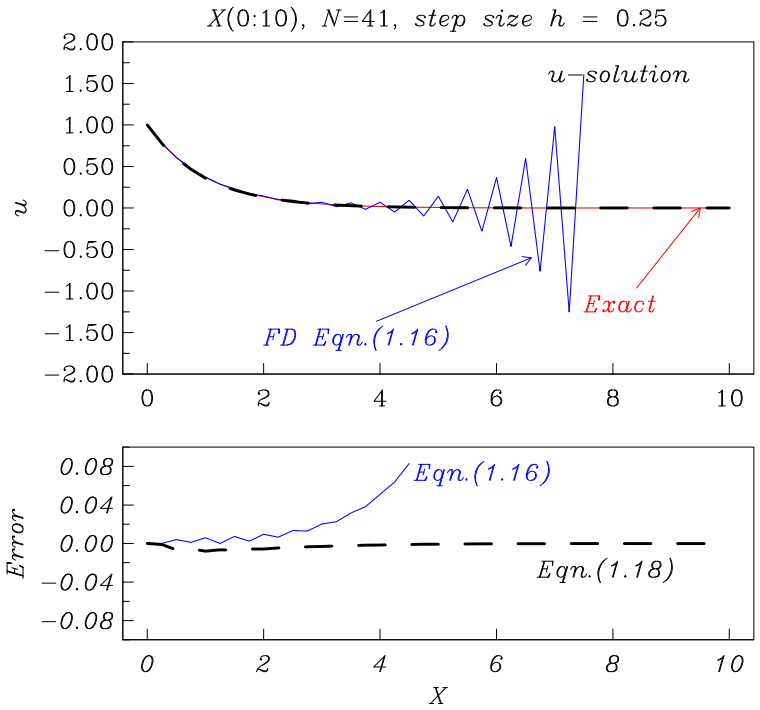

Figure 1.6: Solution of $u' = -u$ with $u(0) = 1$, N=41 grid points, using expression (1.11) given by the dashed line (– – –).

Comparing figure 1.6 with figure 1.5, why do you think the expression (1.11) gives us a much better result then using Eqn. (1.9)?

Finite-difference formula via Taylor series expansions

To determine the weight coefficients a, b, and c in expression (1.7) 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

$$u_{n+1} \equiv u(x_n + h) = u(x_n) + hu'(x_n) + \tfrac{1}{2}h^2u''(x_n) + \tfrac{1}{6}h^3u'''(x_n) + \tfrac{1}{24}h^4u''''(x_n) + \dots \quad (1.12)$$

$$u_{n-1} \equiv u(x_n - h) = u(x_n) - hu'(x_n) + \tfrac{1}{2}h^2u''(x_n) - \tfrac{1}{6}h^3u'''(x_n) + \tfrac{1}{24}h^4u''''(x_n) + \dots \quad (1.13)$$

It follows by addition and subtraction that

$$u_{n+1} - u_{n-1} = 2hu'(x_n) + \tfrac{1}{3}h^3u'''(x_n) + O(h^5) \quad (1.14)$$

and

$$u_{n+1} + u_{n-1} = 2u(x_n) + h^2u''(x_n) + \tfrac{1}{12}h^4u''''(x_n) + O(h^6). \quad (1.15)$$

We can therefore approximate the derivatives by the difference operators

$$\frac{du}{dx}|_n = \frac{u_{n+1} - u_{n-1}}{2h} + O(h^2) \equiv \frac{\mu\delta}{h}u_n + O(h^2), \quad (1.16)$$

with the weights $a = -1/(2h), b = 0, c = 1/(2h)$; the second derivative FD form is given by

$$\frac{d^2u}{dx^2}|_n = \frac{u_{n-1} - 2u_n + u_{n+1}}{h^2} + O(h^2) \equiv \frac{\delta^2}{h^2}u_n + O(h^2), \quad (1.17)$$

with the weights $a = 1/h^2, b = -2/h^2, c = 1/h^2$. Equations (1.17) and (1.16) 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.

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

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

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

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

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 1.14) we have

$$e = (c-a)\frac{u'''|_n}{3!}h^3 = \frac{u'''|_n}{3}h^2 \sim O(h^2) \tag{1.18}$$

For h sufficiently small, the error will be dominated by u''' with all other even higher-order derivative contributions being negligible compared 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 function values U_n 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.

Higher order and sided-differences

There are many difference formulae which can be used in this way. Note that they may need modification near boundaries, for example at $n = 1$, as we have not defined u_{-1}. When derivatives have to be evaluated on or near the boundary of the computational domain, where function values are only available on one side – approximations to derivatives of functions there are known as sided formulae.

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:

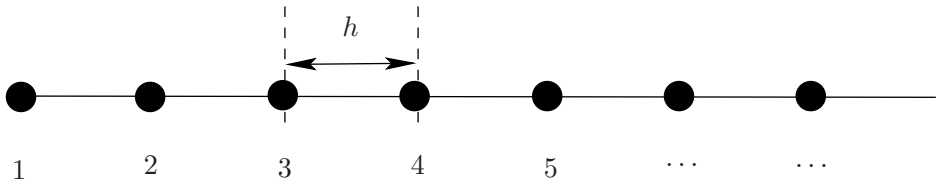

Figure 1.7: Sided differencing stencil, left-sided boundary points.

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

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

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

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

Right-sided non-centered boundary weights:

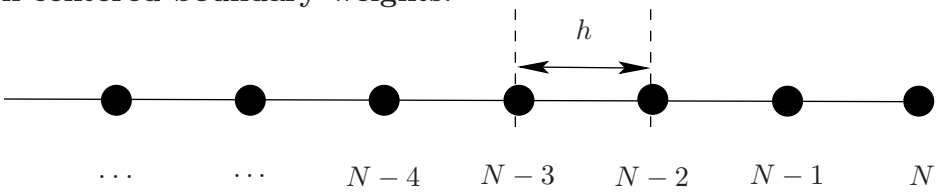

Figure 1.8: Sided differencing stencil, right-sided boundary points.

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

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

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

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

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;

- Lagrange or Polynomials interpolation.

Finite-difference formula 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).

2. 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. (1.16), while using Eqn. (1.18) 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

$$L[\phi] = f \tag{2.1}$$

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. 2.1 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*.

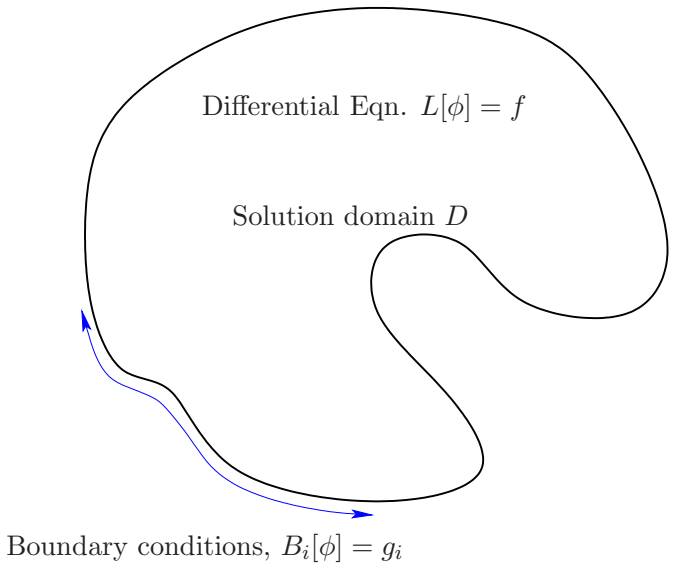

Figure 2.1: Equilibrium or bounded boundary value problem.

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

$$L[\phi] = f \tag{2.2}$$

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

$$I_i[\phi] = h_i \tag{2.3}$$

and subject to prescribed conditions

$$B_i[\phi] = g_i \tag{2.4}$$

on some open boundaries. Fig. 2.2 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.

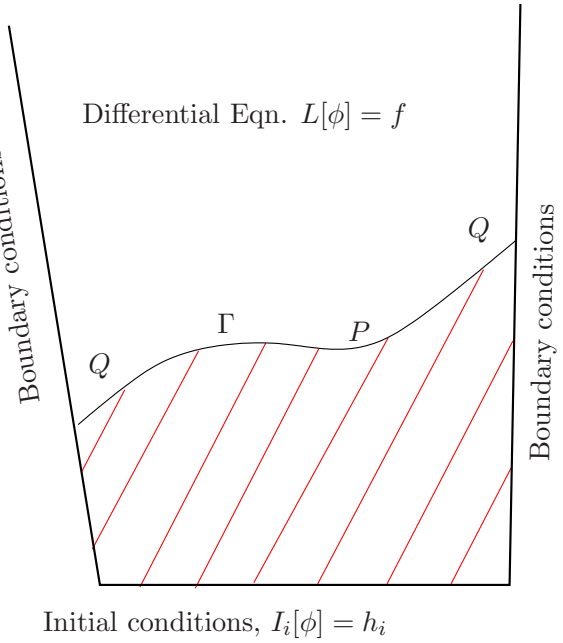

Figure 2.2: Propagation or initial value problem.

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 $u(x, y)$

$$au_{xx} + bu_{xy} + cu_{yy} = f \ . \tag{2.5}$$

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 (2.5) is not necessarily **linear** – for example it is perfectly allowable in the discussion that follows that

$$f = du_x + eu_y + hu + g, \tag{2.6}$$

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. 2.3. 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 $du = u_x dx + u_y dy$, which is known because u_x and u_y are known along Γ. Similarly,

$$\left.\begin{aligned} d(u_x) &= u_{xx}dx + u_{xy}dy \\ d(u_y) &= u_{xy}dx + u_{yy}dy \end{aligned}\right\}. \tag{2.7}$$

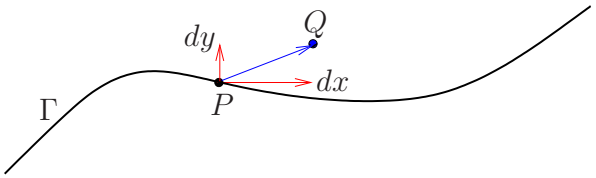

Figure 2.3: Problem description.

We combine (2.5) and (2.7) in matrix form:

$$\begin{pmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{pmatrix} \begin{pmatrix} u_{xx} \\ u_{xy} \\ u_{yy} \end{pmatrix} = \begin{pmatrix} f \\ d(u_x) \\ d(u_y) \end{pmatrix}, \tag{2.8}$$

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 (2.8) will have a unique solution for u_{xx}, u_{xy} and u_{yy} unless the determinant of that matrix vanishes, that is unless

$$\begin{vmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{vmatrix} = a(dy)^2 - b\,dxdy + c(dx)^2 = 0. \tag{2.9}$$

If (2.9) holds, the equation (2.8) 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

$$a\,\frac{d(u_x)}{dx} + c\,\frac{d(u_y)}{dy} = f\ . \tag{2.10}$$

This is surprising. If we can choose a direction (dx, dy) which satisfies (2.9) 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 (2.9) is called the **Characteristic equation** of (2.5). It is a quadratic in dy/dx with solution

$$\frac{dy}{dx} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a}\ . \tag{2.11}$$

Equation (2.5) is classified as hyperbolic, parabolic or elliptic according to whether these roots are real. Note the sign of b in the formula.

$$\text{If} \left\{ \begin{array}{lll} b^2-4ac>0, & 2 \text{ real roots (2.5) is} & \textbf{hyperbolic} \\ b^2-4ac=0, & 1 \text{ real roots (2.5) is} & \textbf{parabolic} \\ b^2-4ac<0, & 0 \text{ real roots (2.5) is} & \textbf{elliptic} \end{array} \right\}. \tag{2.12}$$

For hyperbolic equations, (2.11) is an ODE for $y(x)$ which can be integrated to define two sets of curves (one for the + sign, one for the − sign), called the **characteristics** of (2.5).

Example: Significance of Characteristics.

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

$$\frac{\partial^2 u}{\partial x^2} = \frac{1}{c^2}\frac{\partial^2 u}{\partial t^2} \qquad \text{for} \quad t>0 \qquad \text{with} \quad u(x,0)=F(x) \qquad \text{and} \quad u_t(x,0)=G(x)\ . \tag{2.13}$$

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:

$$u(x,\, t) = \tfrac{1}{2}\big[F(x+ct)+F(x-ct)\big] + \frac{1}{2c}\int_{x-ct}^{x+ct} G(\xi)d\xi\ . \tag{2.14}$$

Using (2.11) the characteristics of (2.13) are two families of straight lines

$$\frac{dt}{dx} = \pm\frac{1}{c} \qquad \text{or} \quad x \pm ct = \text{constant}. \tag{2.15}$$

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 $x_0-ct_0 \leqslant x \leqslant x_0+ct_0$. 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 (2.13), the characteristics are parallel lines, but for more general hyperbolic systems they will be curved and may meet. 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.) Such discontinuities then propagate 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.

2.b 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. 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.

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

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial^2 u}{\partial y^2} \ \text{ in } \ \left\{ \begin{array}{l} 0 < x < 1, \\ 0 < y < 1, \end{array} \right\} \ \text{ with } \ \left\{ \begin{array}{l} u(x,0) = u(x,1) = 0 \\ u(0,y) = u(1,y) = 0. \end{array} \right\}. \tag{2.16}$$

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

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**

$$\nabla^2 u \equiv \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{in} \quad D, \tag{2.17}$$

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

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \qquad \text{in} \quad y > 0 \qquad \text{with} \quad u(x,0) = u_y(x,0) = 0 \tag{2.18}$$

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, $|\sin nx| \leqslant 1$, so we are not altering the boundary condition by more than ε. The (unique) solution to this new problem is

$$u = \varepsilon \sin nx \cosh ny \simeq \varepsilon \sin nx \, \tfrac{1}{2} e^{|ny|} \qquad \text{when} \quad |ny| \gg 1 \ . \tag{2.19}$$

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.

Parabolic Equations

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

$$\frac{\partial u}{\partial t} = K\frac{\partial^2 u}{\partial x^2} \qquad \text{with} \quad u(x,\,0) = f(x)\ . \tag{2.20}$$

As we know $u(x,0)$, we can calculate $u_t(x,0) = Kf''(x)$ from the equation, and so we would expect to be able to step away from $t = 0$. From (2.12), 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 (2.20) is

$$u(x,\,t) = \varepsilon \sin nx\, e^{-n^2 Kt}\ . \tag{2.21}$$

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't un-stir a cup of tea!

Change of Type, Fluid flow example:

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

$$(1 - M_\infty^2)\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \tag{2.22}$$

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

$$\left[K - (\gamma + 1)\phi_x\right]\phi_{xx} + \phi_{yy} = 0, \tag{2.23}$$

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 :

$$\left(1 - \frac{u^2}{a^2}\right)\frac{\partial^2 \phi}{\partial x^2} - \frac{2uv}{a^2}\frac{\partial^2 \phi}{\partial x \partial y} + \left(1 - \frac{v^2}{a^2}\right)\frac{\partial^2 \phi}{\partial y^2} = 0 \tag{2.24}$$

where

$$u = \frac{\partial \phi}{\partial x}, \qquad v = \frac{\partial \phi}{\partial y} \tag{2.25}$$

and a is the speed of sound,

$$\frac{a^2}{\gamma - 1} + \frac{u^2 + v^2}{2} = const. \tag{2.26}$$

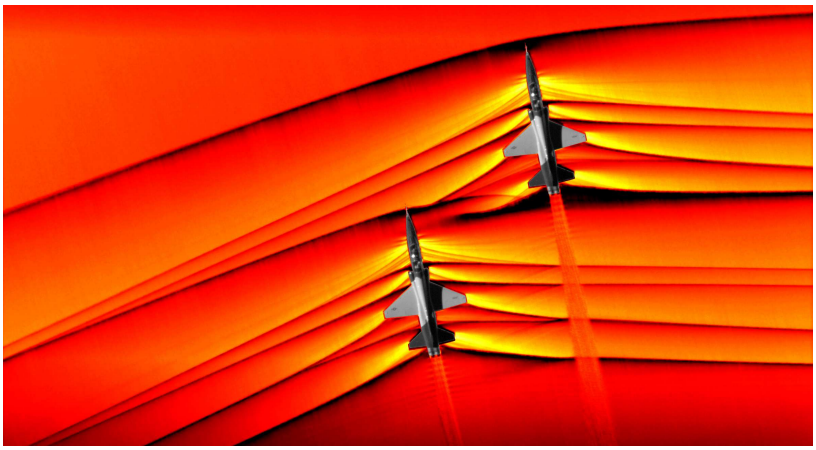

Figure 2.4: (A) Navier-Stokes Equations in action : Supersonic flow

Figure 2.5: (B) Navier-Stokes Equations in action : Transonic flow (1)

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

$$\left.\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} &= \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right) \end{aligned}\right\}. \tag{2.27}$$

where ν is the kinematic viscosity, (u, v) are velocities and p is the pressure. These equations (2.29) 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:

$$\left.\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= \nu\frac{\partial^2 u}{\partial y^2} \\ \frac{\partial p}{\partial y} &= 0 \end{aligned}\right\}. \tag{2.28}$$

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:

$$\left.\begin{aligned} \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \\ u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} &= 0 \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} &= 0 \end{aligned}\right\}. \tag{2.29}$$

This complexity of solutions is shown in figure 2.6.

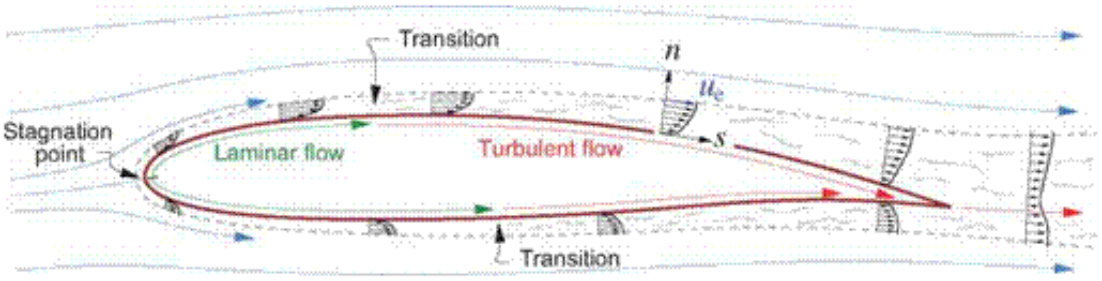

Figure 4.1: Boundary layer and wake development on a typical airfoil, shown by the u(n) velocity profiles. The layer thicknesses are shown exaggerated.

Figure 2.6: (D) Navier-Stokes Equations in action : Boundary Layers

Figure 2.7 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. We shall consider Finite Difference schemes appropriate to each in turn, reflecting the physics, where possible.

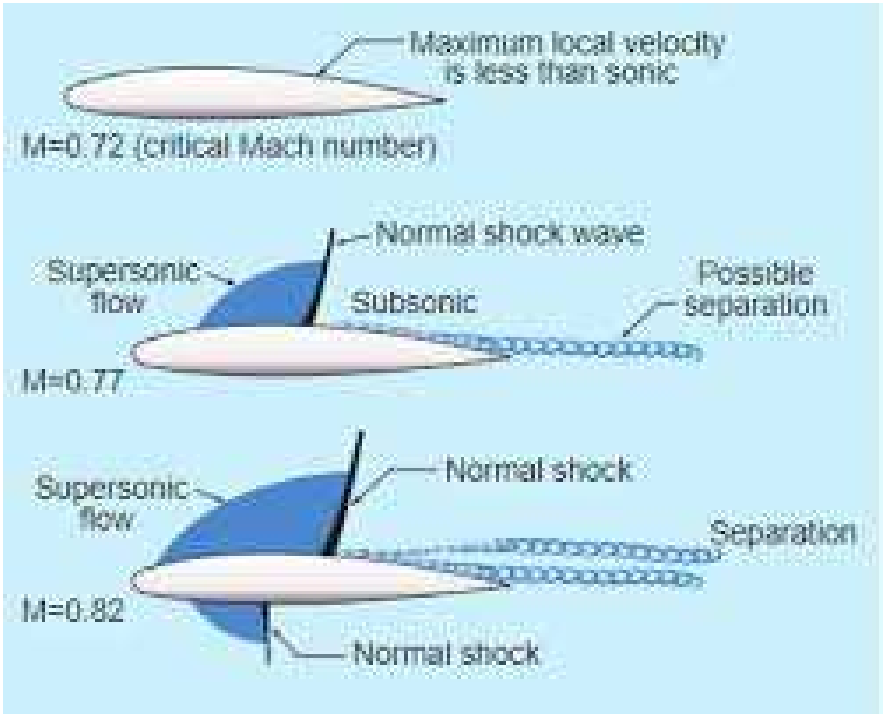

Figure 2.7: (C) Navier-Stokes Equations in action : Transonic flow (2)

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	Smooth solutions	Smooth solutions
(characteristic speed c)	(diffusivity K)	
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

2c. Classification of 1st-order Quasi-linear PDEs in Two Variables

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

$$\left.\begin{array}{rcl} a_1 u_x + b_1 u_y + c_1 v_x + d_1 v_y &=& g_1 \\ a_2 u_x + b_2 u_y + c_2 v_x + d_2 v_y &=& g_2 \end{array}\right\}, \tag{2.30}$$

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.*

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

the coupled system (2.30) may then again be written in matrix form :

$$\begin{pmatrix} a_1 & b_1 & c_1 & d_1 \\ a_2 & b_2 & c_2 & d_2 \\ dx & dy & 0 & 0 \\ 0 & 0 & dx & dy \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ v_x \\ v_y \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ du \\ dv \end{pmatrix}, \tag{2.31}$$

with u, v and the coefficient functions $a_1,\ a_2,\ b_1, \ldots,\ g_1,\ g_2$ known along Γ in Fig. 2.3. Again a unique solution for $u_x,\ u_y,\ v_x$ and v_y exists if the determinant of (2.31) 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

$$dy^2(a_2c_1 - a_1c_2) + dxdy(b_1c_2 - b_2c_1 - a_2d_1 + a_1d_2) + dx^2(b_2d_1 - b_1d_2) = 0, \tag{2.32}$$

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

$$(b_1c_2 - b_2c_1 - a_2d_1 + a_1d_2)^2 - 4(a_2c_1 - a_1c_2)(b_2d_1 - b_1d_2) \tag{2.33}$$

is positive, zero or negative. Hence again allowing classification of (2.30), 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 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.

3. The Explicit Method for the 1-D Diffusion Equation

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

$$\left.\begin{array}{cccc} & u_t = u_{xx} & \text{in} & 0 < x < 1, \;\; t > 0 \\ \text{with} & u(0,\, t) = 0, & u(1,\, t) = 0\,, & u(x,\, 0) = f(x) \end{array}\right\}, \tag{3.1}$$

where the variable t denotes time and x is the spatial coordinate.
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.

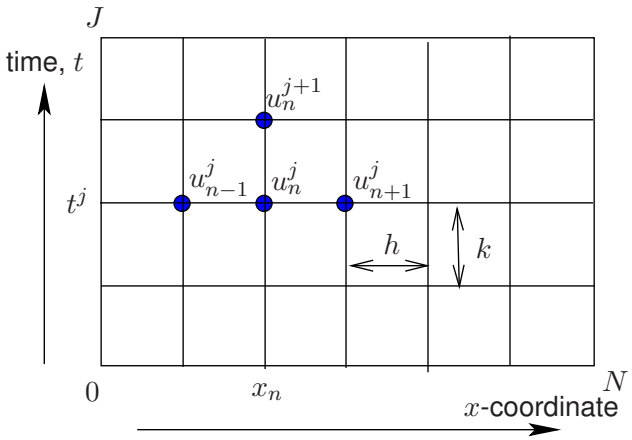

Figure 3.1: Discretised Diffusion equation *mesh*.

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 §1, 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

$$\frac{u_n^{j+1} - u_n^j}{k} = \frac{u_{n+1}^j - 2u_n^j + u_{n-1}^j}{h^2} + R_n^j\,, \tag{3.2}$$

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

$$R_n^j = \tfrac{1}{12}h^2\frac{\partial^4 u_n^j}{\partial x^4} - \tfrac{1}{2}k\frac{\partial^2 u_n^j}{\partial t^2} + O(k^2, h^4)\,. \tag{3.3}$$

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 (3.2), we obtain the scheme

$$U_n^{j+1} = rU_{n+1}^j + (1-2r)U_n^j + rU_{n-1}^j \,, \tag{3.4}$$

with the boundary conditions

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

With a little thought, we see that these boundary conditions and repeated use of (3.4) enable us to calculate U_n^j everywhere. We must now consider how accurate the approximation is. We shall 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.

Truncation Error and Solution Error

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

$$z_n^j \equiv u_n^j - U_n^j \,, \qquad \text{and} \quad R_n^j \equiv (Lu)_n^j - Mu_n^j \,. \tag{3.5}$$

Returning to our specific equation, subtracting (3.4) from (3.2), and using (3.5), we obtain

$$z_n^{j+1} = rz_{n+1}^j + (1-2r)z_n^j + rz_{n-1}^j + kR_n^j \qquad \text{and} \quad z_0^j = z_N^j = z_n^0 = 0 \,. \tag{3.6}$$

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

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

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 steplengths, k and h. Note that $r < 0$ would correspond to $k < 0$, which involves stepping **backwards in time**, which we saw in §2 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 (3.2) is numerically unstable.

4. 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 $r = k/h^2$. Importantly we also showed that the technique only works by marching in positive time t – *i.e.* for $r < 0$ our method fails. Numerical experiments further show that if $r > 1/2$, the method also breaks down seriously as small errors get amplified out of proportion. We can understand that by solving the scheme exactly, using separation of variables.

We know, the general solution to the heat equation can be based on Fourier series. Thus, we can concentrate on understanding what the numerical scheme does to an individual complex exponential, bearing in mind that we can then reconstruct its effect on more general initial data by taking suitable linear combinations of exponentials. To this end, suppose that, at time $t = 0$ there is a small error proportional to εe^{inph} and we neglect all subsequent truncation errors R_n^j; here p represents a spatial wavenumber and h the discretisation step size. Then from (3.6), the error z_n^j obeys the equation

$$z_n^{j+1} = rz_{n+1}^j + (1-2r)z_n^j + rz_{n-1}^j, \qquad z_n^0 = \varepsilon e^{inph}. \tag{4.1}$$

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

$$\xi = re^{iph} + (1-2r) + re^{-iph} = 1 - 2r(1-\cos(ph)) = 1 - 4r\sin^2(\frac{ph}{2}). \tag{4.2}$$

If this error is not to grow, then we require

$$|\xi| \leqslant 1.$$

Now clearly $\xi < 1$, but we need to ensure that $\xi > -1$. The worst case is when $ph = \pi$, and then we must require $1 - 4r \geqslant -1$ or $r \leqslant 1/2$. 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. The stability criterion $|\xi| \leqslant 1$ effectively distinguishes the stable, and hence valid, numerical algorithms from the unstable – hence can be used as a guiding principle to distinguish which discretisations are unsuited to producing a numerically converged solution.

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. Let us try to generalise the Maximum Principle Analysis from last lecture to more general parabolic PDEs.

Consider the problem defined for an arbitrary function Φ,

$$u_t = \Phi(x,\ t,\ u,\ u_x,\ u_{xx}) \qquad \text{in} \quad 0 < x < 1 = Nh, \qquad 0 < t < T = Jk. \tag{4.3}$$

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

$$A \geqslant \frac{\partial \Phi}{\partial u_{xx}} \geqslant a > 0 \tag{4.4}$$

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

$$\left.\begin{array}{lll} u_x & \text{by} & \dfrac{1}{2h}\triangle U_n^j \equiv \dfrac{U_{n+1}^j - U_{n-1}^j}{2h}, \\ u_{xx} & \text{by} & \dfrac{1}{h^2}\delta^2 U_n^j \equiv \dfrac{U_{n+1}^j + U_{n-1}^j - 2U_n^j}{h^2} \end{array}\right\} \quad + O(h^2), \tag{4.5}$$

so that

$$U_n^{j+1} = U_n^j + k\Phi\left[nh,\ jk,\ U_n^j,\ \frac{\triangle U_n^j}{2h}, \frac{\delta^2 U_n^j}{h^2}\right]. \tag{4.6}$$

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,

$$\begin{aligned} \Phi\left[nh,\ jk,\ u_n^j,\ \frac{\triangle u_n^j}{2h},\ \frac{\delta^2 u_n^j}{h^2}\right] - \Phi\left[nh,\ jk,\ U_n^j,\ \frac{\triangle U_n^j}{2h},\ \frac{\delta^2 U_n^j}{h^2}\right], \\ = \frac{\partial\Phi}{\partial u} z_n^j + \frac{\partial\Phi}{\partial u_x}\frac{\triangle z_n^j}{2h} + \frac{\partial\Phi}{\partial u_{xx}}\frac{\delta^2 z_n^j}{h^2} + O((z)^2). \end{aligned} \tag{4.7}$$

Subtracting (4.6) from the equation involving u and using the above, gives

$$z_n^{j+1} = r\left[\frac{\partial\Phi}{\partial u_{xx}} - \frac{h}{2}\frac{\partial\Phi}{\partial u_x}\right] z_{n-1}^j + r\left[\frac{\partial\Phi}{\partial u_{xx}} + \frac{h}{2}\frac{\partial\Phi}{\partial u_x}\right] z_{n+1}^j + \left[1 + k\frac{\partial\Phi}{\partial u} - 2r\frac{\partial\Phi}{\partial u_{xx}}\right] z_n^j + kR_n^j \tag{4.8}$$

Now the Maximum Principle argument requires the three coefficients in square brackets to be positive (see the arguments of §3, equation (3.4) etc.). Suppose therefore that

$$A \geqslant \frac{\partial\Phi}{\partial u_{xx}} \geqslant a > 0, \qquad \left|\frac{\partial\Phi}{\partial u_x}\right| \leqslant b\ , \qquad C \geqslant \frac{\partial\Phi}{\partial u} \geqslant c\ , \tag{4.9}$$

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

$$\frac{\partial\Phi}{\partial u_{xx}} \pm \tfrac{1}{2}h\frac{\partial\Phi}{\partial u_x} \geqslant a - \tfrac{1}{2}hb \qquad \text{and} \quad 1 + k\frac{\partial\Phi}{\partial u} - 2r\frac{\partial\Phi}{\partial u_{xx}} \geqslant 1 + kc - 2rA. \tag{4.10}$$

For the MPA, we therefore require

$$a - \tfrac{1}{2}hb \geqslant 0 \qquad \text{and} \quad 1 + kc - 2rA \geqslant 0. \tag{4.11}$$

If (4.11) holds then we can show in the notation of §3 that

$$\left|\left|z^j\right|\right| \leqslant \frac{e^{Cjk} - 1}{C} D(k + h^2) \qquad \text{for} \quad 1 \leqslant j \leqslant J \tag{4.12}$$

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 the equation $u_t = cu$.) We conclude that the explicit method will work for the nonlinear equation (4.3) provided (4.11) holds. These conditions are **sufficient** but may be overcautious.

As an example, consider Burgers' equation

$$u_t + uu_x = \nu u_{xx} \qquad \text{so that} \qquad \Phi = \nu u_{xx} - uu_x \tag{4.13}$$

where ν is constant. Then

$$A = a = \nu, \qquad b = \max_{x,t}\left[|u|\right], \qquad c = \min_{x,t}\left[-u_x\right],\ C = \max_{x,t}\left[-u_x\right]\ . \tag{4.14}$$

So the explicit method for this equation is stable if

$$\begin{gathered} \nu - \tfrac{1}{2}h\max\left[|u|\right] \geqslant 0 \qquad \text{or} \qquad |u| \leqslant 2\nu/h \\ \text{and} \quad 1 + k\min\left[-u_x\right] - 2r\nu \geqslant 0 \quad \text{or} \quad -u_x \geqslant (2r\nu - 1)/k \end{gathered} \tag{4.15}$$

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 (4.6) 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.

Nonlinear BVP equation example (see, Le-Veque)

We next consider a nonlinear BVP to illustrate how to treat such problems – not a PDE as such but the same technqiue can be used with nonlinear PDEs.

We consider the motion of a pendulum with mass m at the end of a rigid (but massless) bar of length L, and let $\theta(t)$ be the angle of the pendulum from vertical at time t. Ignoring the mass of the bar and forces of friction and air resistance, the differential equation for the pendulum motion can be well approximated by

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta(t), \tag{4.16}$$

where g is the gravitational constant. For simplicity we take $g/L = 1$, hence

$$\frac{d^2\theta}{dt^2} = -\sin\theta(t). \tag{4.17}$$

This is a nonlinear ODE due to the $\sin\theta(t)$ term. For small amplitudes of the angle θ it is possible to approximate $sin\theta \sim \theta$ and obtain the approximate linear differential equation $\theta'' = -\theta(t)$, which has a simple general solution of the form $A\cos(t) + B\sin(t)$. Let us define the more general problem of Eqn (4.17), such that

$$\frac{d^2\theta}{dt^2} = -\sin\theta(t) \quad \text{for } \ 0 < t < T, \tag{4.18}$$

with conditions $\theta(0) = \alpha$ and $\theta(T) = \beta$, with (α, β) specified as two enforced locations.

We can discretise the nonlinear problem Eqn. (4.18) following the approach we undertook for linear problems, namely

$$\frac{1}{h^2}\left(\theta_{i-1} - 2\theta_i + \theta_{i+1}\right) + \sin(\theta_i) = 0, \tag{4.19}$$

for $i = 1, 2, \ldots, m$, where $h = T/(m+1)$ and set $\theta_0 = \alpha$ and $\theta_{m+1} = \beta$. Thus we have a system of m equations for m unknowns –but the $\sin(\theta_i)$ nonlinear term causes difficulties. This can not be solved directly, but instead we must generally use some iterative method, such as Newton's method. This proceeds by way of writing the nonlinear system of equations in the form

$$\mathcal{G}(\theta) = 0, \tag{4.20}$$

now if $\theta^{[k]}$ is an approximation to θ in step k, the Newton's method is derived using a Taylor series expansion

$$\mathcal{G}(\theta^{[k+1]}) = \mathcal{G}(\theta^{[k]} + \mathcal{G}'(\theta^{[k]}) \left(\theta^{[k+1]} - \theta^{[k]}\right) + \ldots. \tag{4.21}$$

Setting $\mathcal{G}(\theta^{[k+1]}) = 0$ as desired and dropping higher order terms gives

$$0 = \mathcal{G}(\theta^{[k+1]}) = \mathcal{G}(\theta^{[k]} + \mathcal{G}'(\theta^{[k]}) \left(\theta^{[k+1]} - \theta^{[k]}\right), \tag{4.22}$$

which gives the Newton update

$$\theta^{[k+1]} = \theta^{[k]} + \delta^{[k]}, \tag{4.23}$$

where $\delta^{[k]}$ solves the linear system

$$\mathcal{J}(\theta^{[k]})\delta^{[k]} = -\mathcal{G}(\theta^{[k]}). \tag{4.24}$$

Here $\mathcal{J}(\theta) \equiv \mathcal{G}'(\theta)$ is the Jacobian matrix with elements

$$\mathcal{J}_{i,j}(\theta) = \frac{\partial}{\partial \theta_j} \mathcal{G}_i(\theta). \tag{4.25}$$

5. Implicit Scheme for the 1-D Diffusion equation

As we discussed last time, the explicit scheme (3.5) 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

$$U_n^{j+1} - U_n^j = r\left[\theta\delta^2 U_n^{j+1} + (1-\theta)\delta^2 U_n^j\right]. \tag{5.1}$$

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 (5.1) and separating the terms depending on j and n leads to

$$\frac{T^{j+1} - T^j}{r\left[\theta T^{j+1} + (1-\theta)T^j\right]} = \frac{X_{n+1} - 2X_n + X_{n-1}}{X_n} = \sigma, \quad \text{say.} \tag{5.2}$$

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

$$X_{n+1} - (\sigma+2)X_n + X_{n-1} = 0, \tag{5.3}$$

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

$$\xi = \frac{m\pi}{N} = m\pi h \qquad \text{for} \quad m = 1, 2\ldots \qquad \text{and} \quad \sigma = -4\sin^2\left(\frac{m\pi h}{2}\right) = \sigma_m, \tag{5.4}$$

say. Then X_n is given by

$$X_n = A\sin(nm\pi h). \tag{5.5}$$

For each such permissible value $\sigma = \sigma_m$, T^j can be found from (5.2).

$$T^{j+1} = \lambda_m T^j, \qquad \text{so that} \quad T^j = C(\lambda_m)^j \qquad \text{where} \quad \lambda_m = \left[\frac{1 + \sigma_m r(1-\theta)}{1 - \sigma_m r\theta}\right]. \tag{5.6}$$

Now we have

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

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

$$1 + 4r\theta\sin^2\tfrac{1}{2}\xi \geqslant 2r\sin^2\tfrac{1}{2}\xi,$$

or

$$(1-2\theta)2r\sin^2\tfrac{1}{2}\xi \leqslant 1. \tag{5.7}$$

If $\theta \geqslant \frac{1}{2}$, therefore, the FDM (5.1) is **unconditionally stable**. If on the other hand $0 \leqslant \theta < \frac{1}{2}$, stability for all m and h can be guaranteed only if

$$r \leqslant \frac{1}{2(1-2\theta)}. \tag{5.8}$$

Note that when $\theta = 0$ we recover the relation $r \leqslant \frac{1}{2}$ and (5.1) is then **conditionally stable**.

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

$$U_n^j = \sum_{m=1}^{\infty} B_m \sin(nm\pi h)(\lambda_m)^j, \tag{5.9}$$

where the coefficients B_m can be found from the Fourier expansion of $u_0(x)$ in (3.1). We can now compare (5.9) with the exact solution (4.4) evaluated at the grid points

$$u_n^j \equiv u(nh,\, jk) = \sum_{m=1}^{\infty} B_m \sin(nm\pi h)(e^{-m^2\pi^2 k})^j \tag{5.10}$$

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

$$\begin{aligned}
\lambda_m &\approx 1 - \frac{r(\xi - \frac{1}{24}\xi^3 + O(\xi^5))^2}{1 + r\theta\xi^2 + O(r\xi^4)}, \\
&= 1 - r\xi^2(1 - \tfrac{1}{12}\xi^2)(1 - r\theta\xi^2) + O(r^3\xi^6). \\
&= 1 - m^2\pi^2 k + \left(\theta + \frac{1}{12r}\right) m^4\pi^4 k^2 + O(m^6 k^3) \\
\text{while} \quad e^{-m^2\pi^2 k} &= 1 - m^2\pi^2 k + \tfrac{1}{2}m^4\pi^4 k^2 + O(m^6 k^3).
\end{aligned} \tag{5.11}$$

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

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

We see from (5.7) 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.

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, $R_n^j = O(k2, h2)$. 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 $a_{n-1}x_{n-1} = b_{n-1}^* + (r/2)x_n$ which we now know, giving us x_{n-1}. Systematically back-substituting determines all the unknowns x_i.

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.

Aside – Separation of variables solution to Diffusion Eqn.

We consider a slight modification of (3.1), namely:

$$\begin{aligned} u_t = u_{xx} \qquad &\text{in} \quad 0 < x < 1,\ t > 0, \\ \text{with} \ \ u(0,t) = u(1,t) = 0, \quad &u(x,0) = u_0(x). \end{aligned} \tag{5.12}$$

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

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

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

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

If we require X to obey the boundary conditions in (4.3), 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 (5.12) is a linear problem, we may combine solutions to obtain a more general solution in the form

$$u(x,\, t) = \sum_{m=1}^{\infty} B_m \sin m\pi x\, e^{-m^2\pi^2 t}. \tag{5.13}$$

The initial condition will be satisfied if

$$u(x,\, 0) = \sum_{m=1}^{\infty} B_m \sin m\pi x = u_0(x). \tag{5.14}$$

Thus all we need do to obtain the solution of (5.12) 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 (5.13).

8. Matrix representation. Neumann and Periodic Boundary Conditions

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

$$\sum_{n=1}^{N-1} A_{mn} U_n^{j+1} = \sum_{n=1}^{N-1} B_{mn} U_n^j + c_m^j \qquad \text{for} \quad 1 \leqslant m \leqslant N-1, \quad 1 \leqslant j \leqslant J \; . \tag{8.1}$$

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 $\varepsilon = O(k^2 + kh^2)$. A and B are $(N-1) \times (N-1)$ matrices and then the error–vector obeys the equation

$$A\mathbf{Z}^{j+1} = B\mathbf{Z}^j + \varepsilon \mathbf{d}^j. \tag{8.2}$$

Assuming the matrix inverse 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

$$\mathbf{Z}^{j+1} = \left(A^{-1}B\right) \mathbf{Z}^j + \varepsilon A^{-1}\mathbf{d}^j = P\mathbf{Z}^j + \varepsilon \mathbf{f}^j \qquad \text{say.} \tag{8.3}$$

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

$$\mathbf{Z}^j = \varepsilon \left[(P)^{j-1}\mathbf{f}^0 + (P)^{j-2}\mathbf{f}^1 + \cdots + P\mathbf{f}^{j-2} + \mathbf{f}^{j-1} \right] \; . \tag{8.4}$$

The matrix P describes the *propagation* of errors from one time–step to the next. For the method (6.1) 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

$$P\mathbf{e}_m = \lambda_m \mathbf{e}_m \; , \qquad (P)^2\mathbf{e}_m = \lambda_m^2 \mathbf{e}_m \qquad \text{and} \quad (P)^j\mathbf{e}_m = (\lambda_m)^j \mathbf{e}_m \, . \tag{8.5}$$

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 (6.1) will be unstable (although see the subtlety overleaf.) Conversely, suppose that $|\lambda_m| \leqslant 1$ 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

$$\left|(P)^j\mathbf{x}\right| = \left| \sum_{m=1}^{N-1} a_m (P)^j \mathbf{e}_m \right| = \left| \sum a_m (\lambda_m)^j \mathbf{e}_m \right| \; \leqslant \; \sum |a_m||\lambda_m|^j \; \leqslant \; \sum |a_m| \, . \tag{8.6}$$

This is bounded and so the error vector $\mathbf{Z}^j$ in (8.6) 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 the method (8.1) is stable provided $\max |\lambda_m| \leqslant 1$ 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 $|B - \lambda A| = 0$ 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,

$$|\lambda_m|^J \leqslant G \iff |\lambda_m| \leqslant G^{1/J} = e^{(\ln G)k/T} \approx 1 + k\frac{\ln G}{T} + O(k^2) \qquad \text{as} \quad k \to 0 \; . \tag{8.7}$$

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

(Nearly) Tridiagonal systems

Last time we showed that an important case has A and B tridiagonal, which permits efficient solution. The Crank-Nicolson ($\theta = \frac{1}{2}$) method for the equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{8.1}$$

had

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

where $\mu = 1/2$ and $d = 0$. For more general problems, the diagonal entries vary with the row number, while keeping the tridiagonal structure.

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 $|\lambda| \leqslant 1$ without calculating them explicitly.

$$\text{Tp } [1+r,\ -\tfrac{1}{2}r,\ -\tfrac{1}{2}r] \qquad \text{and} \quad B = \text{Tp } [1-r,\ \tfrac{1}{2}r,\ \tfrac{1}{2}r] \ ,$$

where we define the tri–diagonal *Toeplitz* matrix

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

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 tridiagonal $N \times N$ Toeplitz matrices of the above form, last time we essentially found the eigenvalues of A and B exactly, showing that the eigenvalues of Tp $[a,\ b,\ c]$ are $a + 2\sqrt{bc}\cos(m\pi/N)$ for $1 \leqslant m \leqslant N-1$.

Neumann Boundary Conditions:

Slight differences occur if we change the boundary conditions on $x = 0, 1$. The condition $u = 0$ on a boundary is called a **Dirichlet** condition. An important alternative are the **Neumann** boundary conditions, $u_x = 0$ at $x = 0$, say. Physically, this 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 wall. 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 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 A given by (8.7) where $\mu = 1$ and $d = 0$. However, in that case we may have 2 more rows and columns in A as we have 2 more unknowns.

Periodic Boundary Conditions:
Another important case to consider are **Periodic** boundary conditions, where we identify $x = 0$ 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. When we evaluate the FDM on the boundary, we get an extra equation, and the matrix A takes the form (6.3) with $\mu = 1/2$ and $d = -\frac{1}{2}r$. The matrix is no longer strictly tri-diagonal, and we have to modify our solving routine.

Lecture 9: Diffusion in more dimensions; the ADI method.

We now have the understanding to begin to tackle more dimensions. Let us consider the problem

$$u_t = \nabla^2 u \equiv u_{xx} + u_{yy} \qquad \text{in } 0 < x < 1,\, 0 < y < L,\, t > 0 \tag{9.1}$$

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

$$u_{xx} \simeq \frac{\delta_x^2 u_{mn}}{h_x^2} \equiv \frac{U_{m+1n} + U_{m-1n} - 2U_{mn}}{h_x^2}, \quad u_{yy} \simeq \frac{\delta_y^2 u_{mn}}{h_y^2} \equiv \frac{U_{mn+1} + U_{mn-1} - 2U_{mn}}{h_y^2}. \tag{9.2}$$

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

The explicit scheme for this problem therefore takes the form

$$U^{j+1}_{mn} = U^j_{mn} + r(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{9.3}$$

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

$$\lambda = 1 - 4r\left(\sin^2 \tfrac{1}{2}\xi + \sin^2 \tfrac{1}{2}\eta\right) \tag{9.4}$$

The worst case is $\xi = \pi$, $\eta = \pi$, and we require $r \leqslant \frac{1}{4}$ 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

$$U^{j+1}_{mn} - r\theta(\delta_x^2 + \delta_y^2)U^{j+1}_{mn} = U^j_{mn} + r(1 - \theta)(\delta_x^2 + \delta_y^2)U^j_{mn}. \tag{9.5}$$

is unconditionally stable if $\theta \geqslant \frac{1}{2}$, so we may choose k larger. However, there is an important difference compared with the 1-D diffusion equation. The LHS of (9.5) is **pentadiagonal** but with a gap between the main 3 diagonals and the other 2 with non-zero entries. Direct methods of solution of (9.5) 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 $100\times100\times100$ grid you have 10^6 unknowns. Direct solution methods will need $O(10^{18})$ calculations. How long will this take 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.

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

$$\frac{U_{mn}^{j+1/2}-U_{mn}^{j}}{k/2}=\frac{1}{h^2}\left(\delta_x^2U_{mn}^{j+1/2}+\delta_y^2U_{mn}^{j}\right) \tag{9.6}$$

or

$$\left(1-\tfrac{1}{2}r\delta_x^2\right)U_{mn}^{j+1/2}=(1+\tfrac{1}{2}r\delta_y^2)U_{mn}^{j}\ . \tag{9.7}$$

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

$$\frac{U_{mn}^{j+1}-U_{mn}^{j+1/2}}{k/2}=\frac{1}{h^2}\left(\delta_x^2U_{mn}^{j+1/2}+\delta_y^2U_{mn}^{j+1}\right) \tag{9.8}$$

or

$$\left(1-\tfrac{1}{2}r\delta_y^2\right)U_{mn}^{j+1}=(1+\tfrac{1}{2}r\delta_x^2)U_{mn}^{j+1/2}\ . \tag{9.9}$$

If we want, we can eliminate the values at the half-time level by operating on (9.7) with $(1+\frac{1}{2}r\delta_x^2)$ and (9.9) with $(1-\frac{1}{2}r\delta_x^2)$ to obtain

$$\left(1-\tfrac{1}{2}r\delta_x^2\right)\left(1-\tfrac{1}{2}r\delta_y^2\right)U_{mn}^{j+1}=(1+\tfrac{1}{2}r\delta_x^2)(1+\tfrac{1}{2}r\delta_y^2)U_{mn}^{j}\ . \tag{9.10}$$

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 (9.7) and (9.9) imply

$$\left.\begin{aligned}\widehat{U}^{j+1/2}&=\frac{1-2r\sin^2\frac{1}{2}\eta}{1+2r\sin^2\frac{1}{2}\xi}\,\widehat{U}^{j}=\lambda_1\widehat{U}^{j}\\ \widehat{U}^{j+1}&=\frac{1-2r\sin^2\frac{1}{2}\xi}{1+2r\sin^2\frac{1}{2}\eta}\,\widehat{U}^{j+1/2}=\lambda_2\widehat{U}^{j+1/2}\end{aligned}\right\}\ \text{say.} \tag{9.11}$$

Then $|\lambda_1|\leqslant 1$ for all ξ, η only if $r\leqslant 1$, which is equivalent to "$r\leqslant\frac{1}{2}$" 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

$$|\lambda_1\lambda_2|=\left|\frac{1-2r\sin^2\frac{1}{2}\xi}{1+2r\sin^2\frac{1}{2}\xi}\right|\left|\frac{1-2r\sin^2\frac{1}{2}\eta}{1+2r\sin^2\frac{1}{2}\eta}\right|\ \leqslant 1 \tag{9.12}$$

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!

ADI in Three Dimensions :

The obvious generalisation to three dimensions would be

$$\left.\begin{aligned}\left(1-\tfrac{1}{3}r\delta_x^2\right)U_{lmn}^{j+1/3} &= \left(1+\tfrac{1}{3}r(\delta_y^2+\delta_z^2)\right)U_{lmn}^{j}\\ \left(1-\tfrac{1}{3}r\delta_y^2\right)U_{lmn}^{j+2/3} &= \left(1+\tfrac{1}{3}r(\delta_x^2+\delta_z^2)\right)U_{lmn}^{j+1/3}\\ \left(1-\tfrac{1}{3}r\delta_z^2\right)U_{lmn}^{j+1} &= \left(1+\tfrac{1}{3}r(\delta_x^2+\delta_y^2)\right)U_{lmn}^{j+2/3}\end{aligned}\right\} \tag{9.13}$$

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

$$\widehat{U}^{j+1/3}=\lambda_1\widehat{U}^{j}\quad\text{etc. where}\quad\lambda_1=\frac{1-\frac{4}{3}r(\sin^2\frac{1}{2}\eta+\sin^2\frac{1}{2}\tau)}{1+\frac{4}{3}r\sin^2\frac{1}{2}\xi}=\frac{1-b-c}{1+a}, \tag{9.14}$$

Then

$$|\lambda_1\lambda_2\lambda_3|=\left|\frac{(1-b-c)(1-a-b)(1-a-c)}{(1+a)(1+b)(1+c)}\right|\ . \tag{9.15}$$

Stability occurs only if a,b,c are small enough, with the worst case being $\xi=\eta=\tau=\pi$, when the necessary condition is $r\leqslant 3/4$.

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

$$\left.\begin{aligned}U_{mn}^{*j+1}-U_{mn}^{j} &= \tfrac{1}{2}r\delta_x^2(U_{mn}^{*j+1}+U_{mn}^{j})+r\delta_y^2U_{mn}^{j}\\ U_{mn}^{j+1}-U_{mn}^{j} &= \tfrac{1}{2}r\delta_x^2(U_{mn}^{*j+1}+U_{mn}^{j})+\tfrac{1}{2}r\delta_y^2(U_{mn}^{j+1}+U_{mn}^{j})\end{aligned}\right\} \tag{9.16}$$

If we eliminate U^* from (9.16) we obtain the same relation (9.10) between U^{j+1} and U^j. However, whereas the intermediate value $U^{j+1/2}$ in (9.7) is an approximation to the real solution at the $j+\frac{1}{2}$ time-level, the intermediate values in (9.16) 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 $u_t=\nabla^2u$ involves two intermediate estimates U^* and U_*^*,

$$\left.\begin{aligned}(1-\tfrac{1}{2}r\delta_x^2)U^{*j+1} &= \left[1+r(\tfrac{1}{2}\delta_x^2+\ \ \delta_y^2+\delta_z^2)\right]U^{\ j}_{lmn}\\ (1-\tfrac{1}{2}r\delta_y^2)U_{*lmn}^{*j+1} &= \left[1+r(\tfrac{1}{2}\delta_x^2+\tfrac{1}{2}\delta_y^2+\delta_z^2)\right]U^{\ j}_{lmn}+\tfrac{1}{2}r\delta_x^2U_{lmn}^{*j+1}\\ (1-\tfrac{1}{2}r\delta_z^2)U^{\ j+1}_{lmn} &= \left[1+\tfrac{1}{2}r(\delta_x^2+\ \ \delta_y^2+\delta_z^2)\right]U^{\ j}_{lmn}+\tfrac{1}{2}r\delta_x^2U_{lmn}^{*j+1}+\tfrac{1}{2}r\delta_y^2U_{*lmn}^{*j+1}\end{aligned}\right\} \tag{9.17}$$

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

10. Elliptic PDEs : Computer Solution by Iteration

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

$$u_{xx} + u_{yy} = 0. \tag{10.1}$$

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 (denoted in figure 10.1). A general discretisation stencil for the unknowns in the interior is given by

$$2(1 + [\frac{\delta y}{\delta x}]^2)U_{i,j} = [\frac{\delta y}{\delta x}]^2(U_{i+1,j} + U_{i-1,j}) + U_{i,j+1} + U_{i,j-1} \tag{10.2}$$

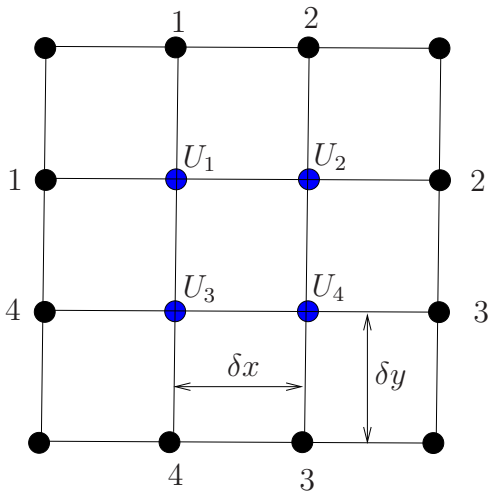

Figure 10.1: Discretised domain

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. Application of Eqn. (10.2), on each of the interior unknown points (U_1, U_2, U_3, U_4) in figure 10.1, results in a system of 4 coupled systems of equations, which may be written in matrix form as

$$[\boldsymbol{A}]\ \bar{\boldsymbol{U}} = \boldsymbol{b} \tag{10.3}$$

i.e.

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 8 \\ 6 \end{pmatrix}. \tag{10.4}$$

This may be easily solved using Gaussian elimination 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 above technique is called the *direct* approach.

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

$$\begin{aligned} 4y_1 &= U_2 + U_3 + 2 \qquad & 4y_2 &= U_1 + U_4 + 4 \\ 4y_3 &= U_1 + U_4 + 8 \qquad & 4y_4 &= U_2 + U_3 + 6 \end{aligned} \tag{10.5}$$

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.*,

$$\begin{aligned} 4U_1 &= U_2 + U_3 + 2 \qquad & 4U_2 &= U_1 + U_4 + 4 \\ 4U_3 &= U_1 + U_4 + 8 \qquad & 4U_4 &= U_2 + U_3 + 6 \end{aligned} \tag{10.6}$$

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

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 10.2 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.

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

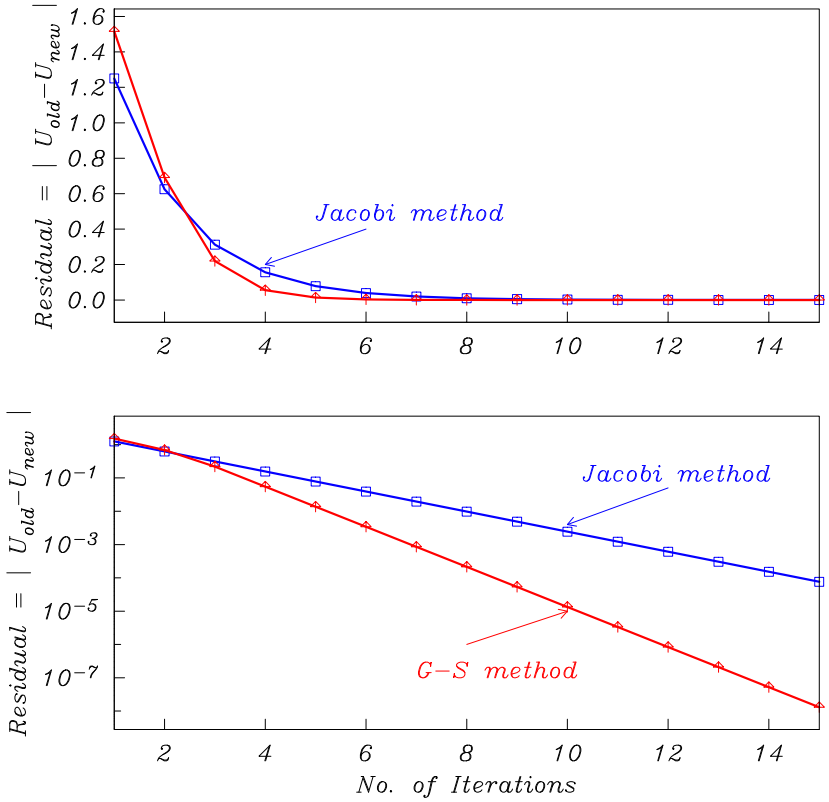

Figure 10.2: Residuals of the solution, as the iterations proceed, comparing Jacobi and Gauss-Seidel (G-S) methods.

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 10.2 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?

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. (10.3).

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. (10.3) 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

11. Iterative methods for Elliptic problems

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

$$\nabla^2 u \equiv u_{xx} + u_{yy} = f \quad \text{in } \ 0 \le x \le 1, \ \ 0 \le y \le 1, \tag{11.1}$$

with appropriate conditions on the 4-sided square boundaries (we had set $f = 0$ in our human computer 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.* $P = 4$, with $N = P - 2$) having the form

$$\boldsymbol{AU} = \boldsymbol{b}, \ \ \text{where } \boldsymbol{A} \text{ is an } N \times N \text{ sparse matrix}, \tag{11.2}$$

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

$$\boldsymbol{U} = (U_{1,1}, U_{1,2}, \ldots, U_{1,N}\,; U_{2,1}, U_{2,2}, \ldots, U_{2,N}\,; U_{N,1}, U_{N,2}, \ldots, U_{N,N}). \tag{11.3}$$

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. 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 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

$$\boldsymbol{B}\boldsymbol{U}^{j+1} = \boldsymbol{b} - C\boldsymbol{U}^j \ \ \text{or} \ \ \boldsymbol{U}^{j+1} = \boldsymbol{B}^{-1}\boldsymbol{b} - (\boldsymbol{B}^{-1}C)\boldsymbol{U}^j. \tag{11.4}$$

If we introduce the error vector $\boldsymbol{Z}^j = \boldsymbol{U}^j - \mathcal{U}$, where $\mathcal{U}$ is the exact solution to (11.2) then

$$\boldsymbol{Z}^{j+1} = (\boldsymbol{B}^{-1}C)\boldsymbol{Z}^j. \tag{11.5}$$

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

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: $\rho(\boldsymbol{T}) = max\{|\lambda_1|, \ldots, |\lambda_k|\}$. 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.

The simplest iterative scheme essentially takes $\boldsymbol{B} = \boldsymbol{D}$, where $\boldsymbol{D}$ is a diagonal part of $\boldsymbol{A}$ and C has zeros on its diagonal. We then defined the iterative **Jacobi** scheme

$$\boldsymbol{U}^{j+1} = \boldsymbol{D}^{-1}(\boldsymbol{b} - C\boldsymbol{U}^{j}) \quad \text{or} \quad U_{mn}^{j+1} = \frac{1}{4}\left[U_{mn+1}^{j} + U_{mn-1}^{j} + U_{m+1n}^{j} + U_{m-1n}^{j} - h^2 f_{mn}\right]. \tag{11.6}$$

This calculates all the new iterations at level $j+1$ before updating U . We also considered the **Gauss-Seidel** 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

$$U_{mn}^{j+1} = \frac{1}{4}\left[U_{mn+1}^{j} + U_{mn-1}^{j+1} + U_{m+1n}^{j} + U_{m-1n}^{j+1} - h^2 f_{mn}\right]. \tag{11.7}$$

This essentially involves back-substitution of the calculated values.

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.

$$|a_{i,1}| + |a_{i,2}| + \ldots + |a_{i,i-1}| + 0 + |a_{i,i+1}| + |a_{i,m}| < |a_{i,i}|. \tag{11.8}$$

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.

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.

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

$$\rho_J = \cos\left(\frac{\pi}{N}\right) \approx 1 - \frac{\pi^2}{2N^2}, \quad \rho_G = \rho_J^2 \approx 1 - \frac{\pi^2}{N^2} \tag{11.9}$$

This agrees with the factor of 4 error reduction we found using the human computer last time. 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

$$\alpha = \frac{p \log 10}{-\log \rho} \approx 0.467 p N^2 \text{ or } 0.233 p N^2 \tag{11.10}$$

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

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

$$\boldsymbol{U}^{j+1} = \boldsymbol{U}^j + \boldsymbol{B}^{-1}\boldsymbol{r}^j \text{ or } \boldsymbol{r}^{j+1} = C\boldsymbol{B}^{-1}\boldsymbol{r}^j. \tag{11.11}$$

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 **overrelaxation** scheme

$$\boldsymbol{U}^{j+1} = \boldsymbol{U}^j + \omega\boldsymbol{B}^{-1}\boldsymbol{r}^j. \tag{11.12}$$

In terms of the code we modify (11.6) to read

$$U_{mn}^{j+1} = (1-\omega)U_{mn}^j + \frac{\omega}{4}\left[U_{mn+1}^j + U_{mn-1}^{j+1} + U_{m+1n}^j + U_{m-1n}^{j+1} - h^2 f_{mn}\right]. \tag{11.13}$$

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.

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 $\alpha \approx 0.37 pN$. The **Successive Over-Relaxation** (**SOR**) method (11.13) requires $O(N)$ iterations not $O(N^2)$. But can we do better still?

Lecture 12: 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 PR and 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

$$\{\mathbf{u}_c, \mathcal{R}\mathbf{v}_F\} = \{\mathcal{P}\mathbf{u}_c, \mathbf{v}_F\} \quad \Longrightarrow \quad \mathcal{R} = P^T/4, \tag{12.1}$$

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 (12.1a) is easiest to choose, as we have less information to work with

$$\begin{aligned} u_F(2i, 2j) &= u_c(i, j) \\ u_F(2i+1, 2j) &= \tfrac{1}{2}(u_c(i, j) + u_c(i+1, j)) \\ u_F(2i, 2j+1) &= \tfrac{1}{2}(u_c(i, j) + u_c(i, j+1)) \\ u_F(2i+1, 2j+1) &= \tfrac{1}{4}(u_c(i, j) + u_c(i+1, j) + u_c(i, j+1) + u_c(i+1, j+1)) \end{aligned} \quad . \tag{12.2}$$

Using (12.1), the appropriate restriction operator, Figure (12.1b), is then $\mathbf{u}_c = \mathcal{R}\mathbf{u}_F$ where

$$\begin{aligned} u_c(i, j) = &\tfrac{1}{4}u_F(2i, 2j) + \ldots \\ &+ \tfrac{1}{8}(u_F(2i+1, 2j) + u_F(2i, 2j+1) + u_F(2i-1, 2j) + u_F(2i, 2j-1)) + \ldots \\ &+ \tfrac{1}{16}(u_F(2i+1, 2j+1) + u_F(2i-1, 2j+1) + u_F(2i-1, 2j-1) + u_F(2i+1, 2j-1)) \end{aligned} \quad . \tag{12.3}$$

This keeps the errors under control when flipping between grids.

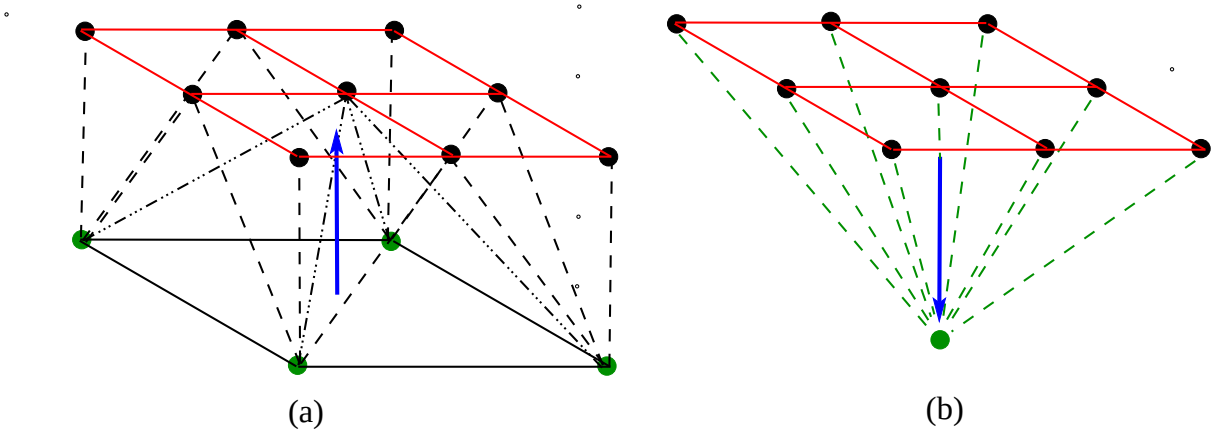

Figure 12.1: (a) Prolongation operation restoring values on a fine grid from a coarse grid. (b) Restriction operation based on full-weighting of the fine-grid values to produce a coarse grid.

More than 2 grids

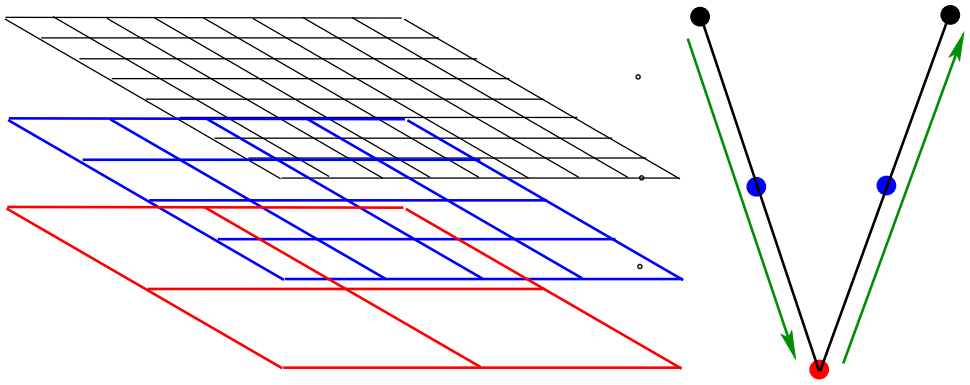

Figure 12.2: Grid hierarchy and traversal order for a V-cycle multigrid method.

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!

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.

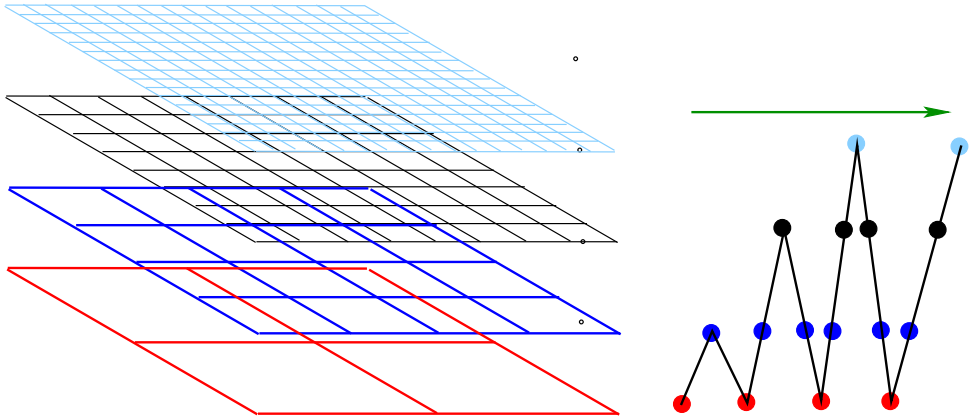

Figure 12.3: Grid hierarchy and traversal order for a full multigrid method.

Lecture 14-15 : Hyperbolic PDEs
The One-dimensional Advection Qquation and Upwinding

Consider the problem for $u(x,\, t)$

$$u_t + c\, u_x = 0 \qquad \text{for} \quad t > 0, \qquad \text{in} \quad u(x,\, 0) = f(x), \tag{14.1}$$

where c is a **positive** constant. The characteristics of this equation are $dx/dt = c$ or $x - ct =$ 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 size or shape** as shown in Figure 14.1 – we emphasis that the exact solution of (14.1) says that the initial function $f(x)$ propagates unaltered in form in the positive x-direction as time progresses.

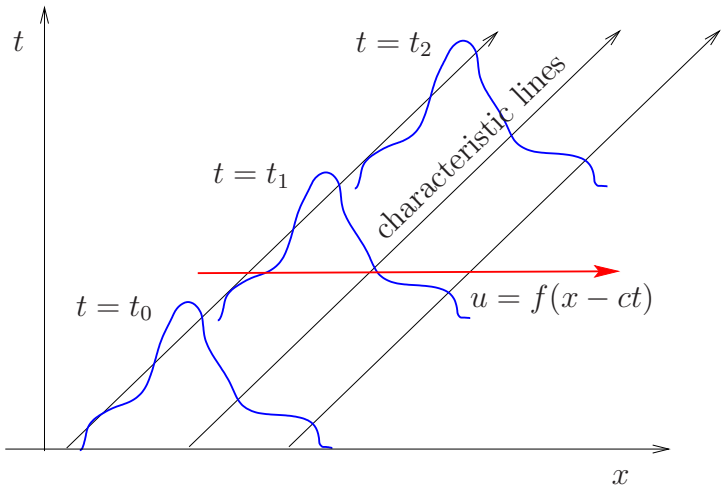

Figure 14.1: Propagation of a function $u = f(x)$ with time t.

Can we develop finite-difference methods (FDMs) with these conservation properties? For future reference, we observe that for a Fourier mode with $f(x) = \exp(ix\xi/h) \equiv e^{in\xi}$

$$u_n^{j+1} \equiv u(nh,\, (j+1)k) = u_n^j \exp(-i\xi q) \qquad \text{where} \quad q = ck/h > 0 \tag{14.2}$$

is the **Courant number** (also referred to as the **Courant-Friedrichs-Lewy** (**CFL**)-condition).

The real solution therefore has a growth factor $\lambda = \exp(-i\xi q)$. We note that $|\lambda| = 1$ so that there is no growth or decay. We can model (14.1) in many different ways. We shall use a forward difference for u_t and consider 6 schemes (**Schemes-6**) :

$$\frac{U_n^{j+1} - U_n^j}{-q} = \begin{cases} \mathbf{A}: & \frac{1}{2}\Delta U_n^j \equiv \frac{1}{2}(U_{n+1}^j - U_{n-1}^j) & \text{Explicit, centred} \\ \mathbf{B}: & U_{n+1}^j - U_n^j & \text{Explicit, forwards} \\ \mathbf{C}: & U_n^j - U_{n-1}^j & \text{Explicit, backwards} \\ \mathbf{D}: & \frac{1}{2}\Delta U_n^{j+1/2} & \text{Two-step, centred} \\ \mathbf{E}: & \frac{1}{2}\left[\frac{1}{2}\Delta U_n^{j+1} + \frac{1}{2}\Delta U_n^j\right] & \text{Crank-Nicolson} \\ \mathbf{F}: & \frac{1}{2}\Delta U_n^j - \frac{1}{2}q\delta^2 U_n^j & \text{Lax-Wendroff} \end{cases}$$

(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 (**Table I**) :

$$\lambda = \begin{cases} \mathbf{A}: & 1 - iq\sin\xi \\ \mathbf{B}: & 1 - q(e^{i\xi} - 1) \\ \mathbf{C}: & 1 - q(1 - e^{-i\xi}) \\ \mathbf{D}: & \frac{1}{4}\left[iq\sin\xi \pm \sqrt{4 - q^2\sin^2\xi}\right]^2 \qquad \text{(two steps)} \\ \mathbf{E}: & \dfrac{2 - iq\sin\xi}{2 + iq\sin\xi} \\ \mathbf{F}: & 1 - iq\sin\xi - 2q^2\sin^2\tfrac{1}{2}\xi \end{cases}$$

We recall that if $|\lambda| > 1 + O(k)$ the method is unstable; if $|\lambda| < 1$ it is dissipative, while if $|\lambda| = 1$ it is conservative. We see therefore that (**Table II**)

$$|\lambda|^2 = \begin{cases} \mathbf{A}: & 1 + q^2\sin^2\xi & & \text{stable only if } q^2 = O(k) \textit{ i.e. } k \sim h^2 \\ \mathbf{B}: & 1 + 2q(1+q)(1-\cos\xi) & \geqslant 1 \quad \forall\, q, \xi & \text{hopelessly unstable} \\ \mathbf{C}: & 1 - 2q(1-q)(1-\cos\xi) & \leqslant 1 \quad \forall\, q \leqslant 1 & \text{stable, dissipative} \\ \mathbf{D}: & 1 & & \text{provided } q^2\sin^2\xi \leqslant 4 \quad \text{or} \quad \tfrac{1}{2}q \leqslant 1 \\ \mathbf{E}: & 1 & \forall\, q & \text{stable and conservative} \\ \mathbf{F}: & 1 - 4q^2(1-q^2)\sin^4\tfrac{1}{2}\xi & \leqslant 1 \text{ if } q \leqslant 1 & \text{dissipative, but less than } \mathbf{C} \end{cases}$$

We may also be interested in $\arg\lambda$, which determines the *phase* of each mode. In case **E**, for example,

$$\arg\lambda = -2\tan^{-1}(\tfrac{1}{2}q\xi) \approx -q\xi \qquad \text{when } \xi \text{ is small.}$$

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, $q \leqslant 1$. 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, $q = ck/h \equiv c\Delta t/\Delta x$ 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

$$U_n^{j+1} = U_n^j - sc\Delta U_n^j + s|c|\delta^2 U_n^j \qquad \text{where} \quad s = k/2h. \tag{14.3}$$

The need for upwind differences can be interpreted in terms of the characteristics of (14.1), $x - ct =$ 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 discretsation 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.

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 (**Schemes-6**), 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 FDA to $\mathbf{v}$. Then transforming back, defining $\mathbf{U} = S\mathbf{V}$ so that $\mathbf{V} = S^{-1}U$, we find

$$\mathbf{U}_n^{j+1} = \mathbf{U}_n^j - sA\Delta\mathbf{U}_n^j + sA^+\delta^2\mathbf{U}_n^j + k\mathbf{d}_n^j \qquad \text{where} \quad A^+ = SD^+S^{-1} \ . \tag{14.4}$$

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 (14.4) is

$$\frac{k}{h} \max_{i=1\ldots p} |\lambda_i| \ \leqslant 1 \ . \tag{14.5}$$

Lecture 16. 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

$$\frac{u_n^{j+1} - u_n^j}{k} = \left[u_t + \tfrac{1}{2}k u_{tt}\right]_n^j + O(k^2) \ .$$

So if $u_t = -cu_x$ with c constant, then $u_{tt} = -cu_{xt} = c^2 u_{xx}$, and we may write

$$\frac{u_n^{j+1} - u_n^j}{k} = -c\frac{\Delta u_n^j}{2h} + \frac{kc^2}{2h^2}\delta^2 u_n^j + O(k^2,\, h^2)$$

This leads to the centred scheme (**Scheme F)** on lectures sheets 14-15.

$$U_n^{j+1} = U_n^j - \tfrac{1}{2}q\Delta U_n^j + \tfrac{1}{2}q^2\delta^2 U_n^j \tag{16.1}$$

Note that this formula requires c to be constant. However, a similar result can be derived for the **conservation** equation $u_t + [f(u,\, x)]_x = 0$. 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 of density ρ, for which Q is a source and $\mathbf{b}$ is its *flux*. Over any fixed volume V bounded by the surface ∂V, normal $\widehat{\mathbf{n}}$,

$$\frac{d}{dt}\int_V \rho \, dV = \int_V Q \, dV - \int_{\partial V} \mathbf{b}\cdot\widehat{\mathbf{n}} \, dS \ ,$$

so that the rate of increase of the physical quantity in V is equal to the total rate of generation within V less the amount that flows out of V across the boundary.

Lax-Wendroff schemes for simultaneous conservation equations.
More generally, let $\mathbf{u}$ and $\mathbf{f}[\mathbf{u}]$ be p-vectors satisfying

$$\mathbf{u}_t + (\mathbf{f}[\mathbf{u}])_x = 0. \tag{16.2}$$

Equation (16.2) represents the conservation of p physical quantities. Ideally, we would like our FDM to conserve them also. For example, we could write

$$\mathbf{U}_n^{j+1} - \mathbf{U}_n^j = -\tfrac{1}{2}s\Delta\mathbf{F}_n^j \qquad \text{where} \quad \mathbf{F}_n^j = \mathbf{f}\left[\mathbf{U}_n^j\right] \qquad \text{and} \quad s = k/h \ . \tag{16.3}$$

This scheme is **conservative** because if we sum over all values of n the right-hand-side all cancels (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 (16.3) may be unstable. Alternatively we might rewrite (16.2) in the form

$$(u_l)_t + \sum_{m=1}^{p} A_{lm}(u_m)_x = 0 \qquad \text{for} \quad l = 1 \ldots p \qquad \text{where} \quad A_{lm} = \frac{\partial(f_l)}{\partial(u_m)} \ , \tag{16.4}$$

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

$$\mathbf{U}_n^{j+1} - \mathbf{U}_n^j = -\tfrac{1}{2}sA\left(\Delta\mathbf{U}_n^j\right) \tag{16.5}$$

but this would **not** be conservative unless A is constant. The Lax-Wendroff idea applied to (16.2) gives

$$\mathbf{u}_{tt} = -(\mathbf{f}[\mathbf{u}])_{xt} = -(A\mathbf{u}_t)_x = \big(A(\mathbf{f})_x\big)_x$$

so that

$$\left.\begin{aligned}(\mathbf{u}_{tt})^j_n &= \frac{1}{h}\left[A^j_{n+1/2}\,(\mathbf{f}_x)^j_{n+1/2} - A^j_{n-1/2}\,(\mathbf{f}_x)^j_{n-1/2}\right] + O(h^2)\\ &= \frac{1}{h^2}\left[A^j_{n+1/2}\left(\mathbf{f}^j_{n+1} - \mathbf{f}^j_n\right) - A^j_{n-1/2}\left(\mathbf{f}^j_n - \mathbf{f}^j_{n-1}\right)\right] + O(h^2).\end{aligned}\right\} \tag{16.6}$$

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 (16.6) for $\mathbf{u}_{tt}$, we obtain

$$\mathbf{U}^{j+1}_n = \mathbf{U}^j_n - \tfrac{1}{2}s\Delta\mathbf{F}^j_n + \tfrac{1}{2}s^2\Big[A^j_{n+1/2}\left(\mathbf{F}^j_{n+1} - \mathbf{F}^j_n\right) - A^j_{n-1/2}\left(\mathbf{F}^j_n - \mathbf{F}^j_{n-1}\right)\Big]\ . \tag{16.7}$$

However while (16.7) 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

$$\left.\begin{aligned}\mathbf{U}^{j+1/2}_{n+1/2} &= \tfrac{1}{2}\left(\mathbf{U}^j_n + \mathbf{U}^j_{n+1}\right) - \tfrac{1}{2}s\left(\mathbf{F}^j_{n+1} - \mathbf{F}^j_n\right)\\ \mathbf{U}^{j+1}_n &= \mathbf{U}^j_n - s\left(\mathbf{F}^{j+1/2}_{n+1/2} - \mathbf{F}^{j+1/2}_{n-1/2}\right)\end{aligned}\right\} \tag{16.8}$$

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 (16.7). For the one-dimensional case $p = 1$, with $f = cu$, we obtain the two-step scheme (**Scheme D**) with a different step-size.

The two-step Lax-Wendroff scheme (16.8) may be extended to two space dimensions without difficulty. Consider the equation for $\mathbf{u}(x,\, y,\, t)$

$$\mathbf{u}_t + \mathbf{f}_x + \mathbf{g}_y = 0 \qquad \text{and take} \quad h_x = h_y = h\ .$$

We define $\mathbf{U}$ at the half time-levels for $(p,\, q) = (m,\, n+1/2)$ or $(m+1/2,\, n)$ by

$$\mathbf{U}^{j+1/2}_{pq} = \tfrac{1}{4}\big[\mathbf{U}^j_{p+,q} + \mathbf{U}^j_{p-,q} + \mathbf{U}^j_{p,q+} + \mathbf{U}^j_{p,q-}\big] - \tfrac{1}{2}s\big[\mathbf{F}^j_{p+,q} - \mathbf{F}^j_{p-,q} + \mathbf{G}^j_{p,q+} - \mathbf{G}^j_{p,q-}\big]\ ,$$

where $p\pm$ denotes $p \pm 1/2$ and similarly $q\pm$. At the integer time levels, for $(p,\, q) = (m,\, n)$ or $(m+1/2,\, n+1/2)$,

$$\mathbf{U}^{j+1}_{pq} = \mathbf{U}^j_{pq} - s\left[\mathbf{F}^{j+1/2}_{p+,q} - \mathbf{F}^{j+1/2}_{p-,q} + \mathbf{G}^{j+1/2}_{p,q+} - \mathbf{G}^{j+1/2}_{p,q-}\right]\ . \tag{16.9}$$

The resulting scheme is second order and conservative.

Lecture 17. 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

$$u_t = \mathcal{L}u, \quad \text{where } \mathcal{L} \text{ is an operator, perhaps a matrix.} \tag{17.1}$$

Using a Taylor series, we have

$$\frac{u^{j+1} - u^j}{k} = u_t + \frac{1}{2}ku_{tt} + \dots \tag{17.2}$$

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

$$u^{j+1} = u^j + k\mathcal{L}u^j + \frac{1}{2}k^2\mathcal{L}^2u^j + \dots = (I + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2 + \dots)u^j. \tag{17.3}$$

Here I denotes the identity operator – we can think of it as "1". Defining the exponential operator $\exp(kL) \equiv (I + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2 + \frac{1}{6}k^3\mathcal{L}^3 + \dots)$, then the exact solution of (17.2) can be written

$$u^{j+1} = \exp(k\mathcal{L})u^j. \tag{17.4}$$

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 multigrid. Can we treat the two parts differently without sacrificing the overall accuracy of the scheme? This is the idea behind **Operator Splitting**. Recall lecture 9 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 L and M

$$u_t = \mathcal{L}u + \mathcal{M}u \tag{17.5}$$

which has the solution

$$u^{j+1} = u^j + k(\mathcal{L} + \mathcal{M})u^j + \frac{1}{2}k^2(\mathcal{L} + \mathcal{M})^2u^j + \dots \tag{17.6}$$

Note this is a schematic representation ˘2013 the operators could involve iteration or implicit methods, but ultimately we represent the operation in this manner. Suppose we alternate solving $u_t = \mathcal{L}u$ and $u_t = \mathcal{M}u$ by our favourite methods solving

$$\begin{aligned} \hat{u} &= (I + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2)u^j, \quad \text{and then} \\ u^{j+1} &= (I + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)\hat{u} \end{aligned} \tag{17.7}$$

We then have

$$\begin{aligned} u^{j+1} &= (I + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)(I + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2)u^j \\ &= u^j + k(\mathcal{L} + \mathcal{M})u^j + \frac{1}{2}k^2(\mathcal{L}^2 + \mathcal{M}^2 + 2\mathcal{M}\mathcal{L})u^j. \end{aligned} \tag{17.8}$$

Is this the same as (17.6)? 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, $\frac{1}{2}(\mathcal{L}^2+\mathcal{M}^2+\mathcal{L}\mathcal{M}+\mathcal{M}\mathcal{L})$. 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 $\mathcal{L}$, a process known as Strang splitting. Thus

$$\begin{aligned} \hat{u} &= (I + \frac{1}{2}k\mathcal{L} + \frac{1}{2}(\frac{1}{2}k)^2\mathcal{L}^2)u^j \\ u^* &= (I + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)\hat{u} \\ u^{j+1} &= (I + \frac{1}{2}k\mathcal{L} + \frac{1}{2}(\frac{1}{2}k)^2\mathcal{L}^2)u^* \end{aligned} \tag{17.9}$$

The final estimate is $u^{j+1} = Pu^j$ where, neglecting terms of $O(k^3)$,

$$\begin{aligned} P &= (I + \frac{1}{2}k\mathcal{L} + \frac{1}{8}k^2\mathcal{L}^2)(I + k\mathcal{M} + \frac{1}{2}k^2\mathcal{M}^2)(I + \frac{1}{2}k\mathcal{L} + \frac{1}{8}k^2\mathcal{L}^2) \\ &= I + k(\frac{1}{2}\mathcal{L} + \mathcal{M} + \frac{1}{2}\mathcal{L}) + k^2(\frac{1}{8}\mathcal{L}^2 + \frac{1}{2}\mathcal{M}^2 + \frac{1}{8}\mathcal{L}^2 + \frac{1}{2}\mathcal{L}\mathcal{M} + \frac{1}{2}\mathcal{M}\mathcal{L} + \frac{1}{4}\mathcal{L}^2) \\ &= I + k(\mathcal{L} + \mathcal{M}) + \frac{1}{2}k^2(\mathcal{L}^2 + \mathcal{M}^2 + \mathcal{L}\mathcal{M} + \mathcal{M}\mathcal{L}), \end{aligned} \tag{17.10}$$

which agrees with the estimate (17.6). The method (17.9) 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're taking several. Our 1st time step ends with a half-step in $\mathcal{L}$, and the second begins with a half step in $\mathcal{L}$ – we could combine those and take a full step with $\mathcal{L}$. As a check we note that taking two half-steps involves

$$(I + \frac{1}{2}k\mathcal{L} + \frac{1}{8}k^2\mathcal{L}^2)^2 = I + k\mathcal{L} + (\frac{2}{8} + \frac{1}{4})k^2\mathcal{L}^2 = I + k\mathcal{L} + \frac{1}{2}k^2\mathcal{L}^2, \tag{17.11}$$

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 $\mathcal{L}$.

Lecture 18. Dissipation and Dispersion

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

$$T_t = \mu \nabla^2 T \tag{18.1}$$

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

$$\int_\Omega T^2(x,t)dx, \tag{18.2}$$

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

$$u_t + cu_x = 0, \qquad \text{or} \qquad u_{tt} = c^2 u_{xx} \tag{18.3}$$

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

$$\int_\Omega |u(x,t)|^p dx. \tag{18.4}$$

An equation like

$$u_t + cu_x = \nu u_{xxx} \tag{18.5}$$

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.

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. Sadly this is not possible since numerical solutions will exhibit these behaviours even if the exact solution does not. Then we say that the numerical scheme is dissipative and/or dispersive and the interesting thing is to design discretisation schemes which at least minimise these effects which are not present in the exact solution.

When solving PDES analytically a common approach is to assume that the solution $u(x,t)$ has a wave form

$$u(x,t) = \hat{u} e^{i(\omega t + \alpha x)}, \tag{18.6}$$

where ω is a frequency and α the wavenumber with wavelength λ given by $\lambda = 2\pi/\alpha$. Consider the equation

$$u_t = \nu u_{xx}, \tag{18.7}$$

using (18.6) gives

$$i\omega + \nu\alpha^2 = 0, \tag{18.8}$$

from which it is clear that (18.6) can only be satisfied if $\omega = i\nu\alpha^2$, we call (18.8) the **dispersion relationship**. The solution to (18.7) thus becomes

$$u(x,t) = \hat{u}e^{-\nu\alpha^2 t}e^{i\alpha x}. \tag{18.9}$$

This indicates that the wave does not move and decays with time. Likewise for the linear advection equation

$$u_t + cu_x = 0, \quad \text{with } c \text{ a constant value.} \tag{18.10}$$

We get the dispersion relationship $\omega = -c\alpha$, and hence

$$u(x,t) = \hat{u}e^{i\alpha(x-ct)}. \tag{18.11}$$

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 (18.9) is dissipative for $\nu > 0$ for all wave-numbers $\alpha \neq 0$, while (18.11) is neither dissipative nor dispersive.
Next consider

$$u_t + cu_{xxx} = 0, \tag{18.12}$$

for which the dispersion relationship is

$$\omega = \alpha^3 c, \tag{18.13}$$

and hence the Fourier mode representing a solution to (18.12) is

$$u(x,t) = \hat{u}e^{i\alpha(x+\alpha^2 ct)}. \tag{18.14}$$

There are two aspects here, (1) unlike (18.11) the Fourier mode propagates in the opposite direction with speed $\alpha^2 c$, and (2) Fourier modes with different wave-numbers (α) propagate with different speeds ($\alpha^2 c$). So (18.12) is dispersive but non-dissipative (*i.e.* the amplitude neither decays nor grows). 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

$$u_t + au_x - \nu u_{xx} + cu_{xxx} = 0, \tag{18.15}$$

for which the dispersion relationship is

$$\omega = -a\alpha + i\nu\alpha^2 + c\alpha^3 \tag{18.16}$$

and hence the solution for the α-mode will be

$$u(x,t) = \hat{u}e^{-\nu\alpha^2 t}e^{i\alpha\left[x-(a-c\alpha^2)t\right]}. \tag{18.17}$$

The dissipation term is $e^{-\nu\alpha^2 t}$, while the propagating term is

$$e^{i\alpha\left[x-(a-c\alpha^2)t\right]}, \tag{18.18}$$

thus the PDE is dissipative and dispersive due to the dependence of the term $(a - c\alpha^2)$ term above on α^2, and becomes larger the more larger α is.

Dispersion and Dissipation in Discretised Equations

The discrete analogue of the Fourier mode (18.6) is

$$u_k^n = \hat{u}e^{i(\omega n\Delta t+\alpha k\Delta x)}, \tag{18.19}$$

and thus 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 $\omega = a + ib$, with (a, b) considered real only. Substituting $\omega = a + ib$ in (18.19) and a rewrite gives

$$u_k^n = \hat{u}e^{-bn\Delta t}e^{i(na\Delta t+k\alpha\Delta x)} = \hat{u}\left(e^{-b\Delta t}\right)^n e^{i\alpha(k\Delta x-(-a/\alpha)n\Delta t)}. \tag{18.20}$$

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 (18.3) (with c positive) discretised with an up-winded scheme, namely :

$$U_j^{n+1} = U_j^j + q\left(U_j^n - U_{j-1}^n\right), \tag{18.21}$$

where $q = ck/h$. Substitution of (18.19) into the above and simplification gives

$$e^{i\omega k} = 1 - q + q\left[\cos(\alpha h) + i\sin(\alpha h)\right]. \tag{18.22}$$

We next write $\omega = a + ib$ and hence

$$e^{iak}e^{-bk} = 1 - q + q\left[\cos(\alpha h) + i\sin(\alpha h)\right]. \tag{18.23}$$

Which thus gives

$$e^{-bk} = \sqrt{(1-q)^2 + q^2 + 2q(1-q)\cos(\alpha h)}. \tag{18.24}$$

Setting $q = 1$ thus gives $b = 0$, which implies that the scheme will be non-dissipative. Next setting $q = 1 - \varepsilon$, with $\varepsilon << 1$, it can be shown that

$$e^{-bk} = \sqrt{(1-\varepsilon)^2 + \varepsilon^2 + 2(\varepsilon - \varepsilon^2)\cos(\alpha h)}, \tag{18.25}$$

which suggests the scheme will be dissipative for q values just less than 1. Whether the scheme is dispersive or not we consider the imaginary part of (18.23), namely

$$e^{iak} = \cos(ak) + i\sin(ak) = \frac{1 - q + q\left[\cos(\alpha h) + i\sin(\alpha h)\right]}{(1-q)^2 + q^2 + 2q(1-q)\cos(\alpha h)}, \tag{18.26}$$

hence

$$\tan(ak) = \frac{q\sin(\alpha h)}{1 - q + q\cos(\alpha h)}, \tag{18.27}$$

i.e. hence

$$a = \frac{1}{k}\tan^{-1}\left(\frac{q\sin(\alpha h)}{1 - q + q\cos(\alpha h)}\right) = \frac{1}{k}\tan^{-1}\left(\frac{q\sin(\alpha h)}{1 - 2q\sin^2(\alpha h/2)}\right). \tag{18.28}$$

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.

Further details see, chapter 7 of Thomas, J. W. "Numerical Partial Differential Equations – Finite Difference Methods"

Lecture 19: NSE and Conservation Form

Compressible Viscous Navier-Stokes Equations

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.

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u_i}{\partial x_i} = 0, \tag{19.1a}$$

$$\rho\left(\frac{\partial u_j}{\partial t} + u_j \frac{\partial u_i}{\partial x_i}\right) = \nabla \cdot \Pi_{ij}, \tag{19.1b}$$

$$\rho\left(\frac{\partial h}{\partial t} + u_i \frac{\partial h}{\partial x_i}\right) - \left(\frac{\partial P}{\partial t} + u_i \frac{\partial P}{\partial x_i}\right) = \nabla \cdot (\kappa \nabla T) + \Phi, \tag{19.1c}$$

(19.1d)

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 repeated indices are summed over. Eq. (19.1a) expresses conservation of mass, Eq. (19.1b) expresses conservation of momentum, and Eq. (19.1c) expresses conservation of energy in which Φ is known as the viscous dissipation.
The stress tensor (Π_{ij}) and the specific enthalpy (h) are

$$\Pi_{ij} = -nk_B T\delta_{ij} - \frac{2}{3}\mu \frac{\partial u_k}{\partial x_k}\delta_{ij} + \mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{19.2}$$

$$h = C_p T \tag{19.3}$$

Here the density $n = \rho/m$, m is the mass, T is the temperature and C_p the specific heat capacity.

Inviscid Conservation-Law (Euler) Form – Ideal Gas Dynamics

The five governing inviscid conservation laws which neglect viscosity and heat conduction are

$$\rho_t + (\rho u)_x + (\rho v)_y + (\rho w)_z = 0\ , \tag{19.4a}$$

$$(\rho u)_t + (\rho u^2 + p)_x + (\rho uv)_y + (\rho uw)_z = 0\ , \tag{19.4b}$$

$$(\rho v)_t + (\rho uv)_x + (\rho v^2 + p)_y + (\rho vw)_z = 0\ , \tag{19.4c}$$

$$(\rho w)_t + (\rho uw)_x + (\rho vw)_y + (\rho w^2 + p)_z = 0\ , \tag{19.4d}$$

$$E_t + [u(E+p)]_x + [v(E+p)]_y + [w(E+p)]_z = 0\ . \tag{19.4e}$$

Here E is the total energy per unit volume

$$E = \rho\left(\frac{1}{2}(u^2 + v^2 + w^2) + e\right), \tag{19.5}$$

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

$$p = (\gamma - 1)\left(E - \frac{1}{2}\rho u^2\right), \tag{19.6}$$

with $\gamma = C_p/C_v$. The Euler equations can be shown to be hyperbolic.
The conservation laws (19.4) can be expressed in a very compact notation by defining a column vector U of conserved variables and flux vectors $F(U), G(U), H(U)$ in the x, y and z directions, respectively. Namely:

$$U_t + F(U)_x + G(U)_y + H(U)_z = 0. \tag{19.7}$$

It is important to note that $F = F(U)$,$G = G(U)$, $H = H(U)$; that is, the flux vectors are functions of the conserved variable vector 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.

Conservation Laws : Conservation laws are systems of partial differential equations that can be written in the form (1-dimensional as an example)

$$U_t + F(U)_x = 0, \tag{19.8}$$

where

$$U = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}, \quad F(U) = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}. \tag{19.9}$$

U is called the vector of conserved variables, $F = F(U)$ is the vector of fluxes and each of its components f_i is a function of the components u_j of U.

Jacobian Matrix : The Jacobian of the flux function $F(U)$ in Eq. (19.8) is the matrix

$$A(U) = \frac{\partial F}{\partial U} = \begin{bmatrix} \partial f_1/\partial u_1 & \dots & f_1/\partial u_m \\ \partial f_2/\partial u_1 & \dots & f_2/\partial u_m \\ \vdots & \vdots & \vdots \\ \partial f_m/\partial u_1 & \dots & f_m/\partial u_m \end{bmatrix}. \tag{19.10}$$

The entries a_{ij} of $A(U)$ 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

$$\frac{\partial F(U)}{\partial x} = \frac{\partial F}{\partial U}\frac{\partial U}{\partial x}, \tag{19.11}$$

Eq. 19.8 becomes

$$U_t + A(U)U_x = 0. \tag{19.12}$$

Eigenvalues : The eigenvalues λ_i of a matrix A are the solutions of the characteristic polynomial

$$\|A - \lambda I\| = \det(A - \lambda I) = 0, \tag{19.13}$$

where I is the identity matrix. The eigenvalues of the coefficient matrix A of a system of Eq. (19.8) 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 Eq. (19.8) 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 Eq. (19.8) is said to be elliptic at a point (x, t) if none of the eigenvalues λ_i of A are real.

Diagonalisation and Characteristic Variables

In order to analyse and solve the general IVP for Eq. (19.8) 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

$$A = K\ \Lambda\ K^{-1} \ \text{ or } \ \Lambda = K^{-1}\ A\ K\ , \tag{19.14}$$

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

$$\Lambda = \begin{bmatrix} \lambda_1 & \ldots & 0 \\ 0 & \ldots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \ldots & \lambda_m \end{bmatrix} \ , \ \ K = \left[K^{(1)}, \ldots, [K^{(m)}\right] \ , \ \ A\ K^{(i)} = \lambda_1\ K^{(i)} \tag{19.15}$$

A system Eq. 19.12 is said to be diagonalisable if the coefficient matrix A is diagonalisable. Based on this concept of diagonalisation one then defines Eq. 19.12 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

$$W = K^{-1}\ U \ \text{ or } \ U = K\ W\ , \tag{19.16}$$

so that the linear system Eq. (19.12), 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 equations Eq. (19.12). Provided A is constant, K is also constant and therefore these derivatives are

$$U_t = K\ W_t\ , \ \ U_x = K\ W_x\ . \tag{19.17}$$

Direct substitution of these expressions into Eq. (19.12) gives

$$K\ W_t + A\ K\ W_x = 0. \tag{19.18}$$

Multiplication of this equation from the left by K^{-1} and use of Eq. (19.14) gives

$$W_t + \Lambda W_x = 0. \tag{19.19}$$

This is called the *canonical* form or characteristic form of Eq. (19.12). When written in full this system becomes

$$\frac{\partial}{\partial t}\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix} + \begin{bmatrix} \lambda_1 & \dots & 0 \\ 0 & \dots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \dots & \lambda_m \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{bmatrix}_x = 0 \tag{19.20}$$

Clearly the $i-$th PDE of this system is

$$\frac{\partial w_i}{\partial t} + \lambda_i \frac{\partial w_i}{\partial x} = 0, \quad i = 1, \dots, m \tag{19.21}$$

and involves the single unknown $w_i(x,t)$; the system is therefore decoupled and is identical to the linear advection equation; now the characteristic speed is λ_i and there are m characteristic curves satisfying m ODEs

$$\frac{dx}{dt} = \lambda_i \ , \quad i = 1, \dots, m. \tag{19.22}$$

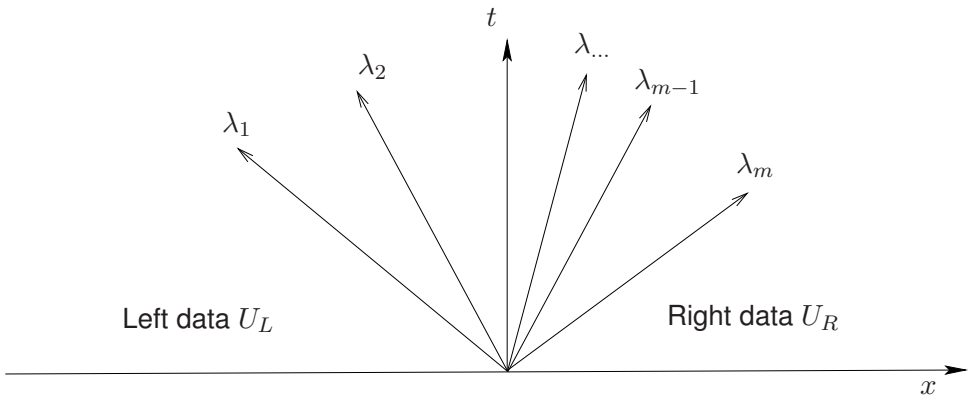

Figure 19.1: Characteristics of a linear hyperbolic system of m equations with constant coefficients.

Example

linearised equations of Gas Dynamics See, Toro (1997).
Consider the system

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_t + \begin{bmatrix} 0 & \rho_o \\ a^2/\rho_o & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}_x = 0 \ , \quad u_1 \equiv \rho \ , \quad u_2 \equiv u \ , \tag{19.23}$$

where ρ_o is a constant and a^2 the speed of sound (constant too). Written in matrix form this becomes

$$U_t + A\, U_x = 0. \tag{19.24}$$

The eigenvalues of A are thus given by

$$\|A-\lambda I\| = \det(A-\lambda I) = \det\begin{bmatrix} 0-\lambda & \rho_o \\ a^2/\rho_o & 0-\lambda \end{bmatrix} = 0\,. \tag{19.25}$$

There are thus two real distinct values $\lambda_1 = -a$ and $\lambda_1 = +a$. The right eigenvectors $K^{(1)}$, $K^{(2)}$ are thus

$$K^{(1)} = \begin{bmatrix} \rho_o \\ -a \end{bmatrix}\,, \quad K^{(2)} = \begin{bmatrix} \rho_o \\ a \end{bmatrix}. \tag{19.26}$$

Thus the K matrix of right eigenvectors and its inverse K^{-1} may be stated as

$$K = \begin{bmatrix} \rho_o & \rho_o \\ -a & a \end{bmatrix}\,, \quad K^{-1} = \frac{1}{2a\rho_o}\begin{bmatrix} a & -\rho_o \\ a & \rho_o \end{bmatrix}\,. \tag{19.27}$$

Hence in terms of characteristic variables Eq. (19.19), we may write

$$\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_t + \begin{bmatrix} -a & 0 \\ 0 & a \end{bmatrix}\begin{bmatrix} w_1 \\ w_2 \end{bmatrix}_x = 0\,, \tag{19.28}$$

or in full form

$$\frac{\partial w_1}{\partial t} - a\frac{\partial w_1}{\partial x} = 0\,, \quad \frac{\partial w_2}{\partial t} + a\frac{\partial w_2}{\partial x} = 0\,. \tag{19.29}$$

Given some initial condition on $U(x,0)$

$$\begin{bmatrix} w_1^{(o)} \\ w_2^{(o)} \end{bmatrix} = K^{-1}\begin{bmatrix} u_1^{(o)} \\ u_2^{(o)} \end{bmatrix}\,, \tag{19.30}$$

written in full form

$$\begin{aligned} w_1^{(o)}(x) &= \tfrac{1}{2a\rho_o}\left[a u_1^{(o)}(x) - \rho_o u_2^{(o)}(x)\right]\,, \\ w_2^{(o)}(x) &= \tfrac{1}{2a\rho_o}\left[a u_1^{(o)}(x) + \rho_o u_2^{(o)}(x)\right]\,. \end{aligned} \tag{19.31}$$

Hence, since the solution to $(w_1,\ w_2)$, on noting Eq. (19.29), is

$$w_1 = w_1^{(o)}(x+at)\,, \quad w_2 = w_2^{(o)}(x-at)\,, \tag{19.32}$$

it follows that the solution in characteristic variables is

$$\begin{aligned} w_1(x,t) &= \tfrac{1}{2\rho_o}\left[a u_1^{(o)}(x+at) - \rho_o u_2^{(o)}(x+at)\right]\,, \\ w_2(x,t) &= \tfrac{1}{2\rho_o}\left[a u_1^{(o)}(x-at) + \rho_o u_2^{(o)}(x-at)\right]\,. \end{aligned} \tag{19.33}$$

We may then transform the solution back to the original problem using $U = K\,W$ (see Toro, 1997).

Example

Isothermal Gas Dynamics

The isothermal equations of Gas Dynamics are one example of a non-linear system of conservation laws. (see Toro, 1997)

Integral Forms of Conservation Laws

Conservation laws may be expressed in differential and integral form. There are two good reasons for considering the integral form(s) of the conservation laws: (i) the derivation of the governing equations is based on physical conservation principles expressed as integral relations on control volumes, (ii) the integral formulation requires less smoothness of the solution, which paves the way to extending the class of admissible solutions to include discontinuous solutions. Consider the conservation of mass, equation in one space dimension

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} = 0. \tag{19.34}$$

This may be written, with reference to Fig. 19.2, as

$$\frac{d}{dt} \int_{x_L}^{x_R} \rho(x,t)\, dx = f(x_L,t) - f(x_R,t)\ , \tag{19.35}$$

where $f = \rho u$ is the flux. Thus for the complete equation we have

$$\frac{d}{dt} \int_{x_L}^{x_R} U(x,t)\, dx = F\left(U(x_L,t)\right) - F\left(U(x_R,t)\right)\ , \tag{19.36}$$

where $F(U)$ is the flux vector. This is one version of the integral form of the conservation laws.

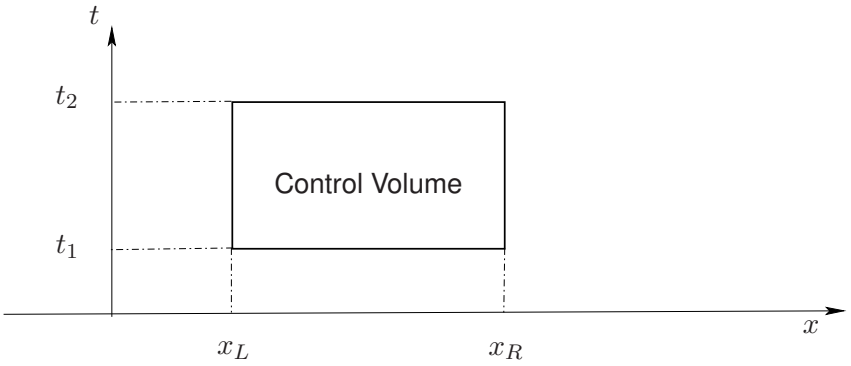

Figure 19.2: Control volume in the x, t-plane.

Acknowledgments: More details in
Riemann Solvers and Numerical Methods for Fluid Dynamics – A Practical Introduction. Toro, E. F., Springer 1997.

Lecture 20. 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

$$\nabla \cdot \mathbf{u} = 0, \tag{20.1}$$

$$\mathbf{u}_t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{F}, \tag{20.2}$$

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 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 flows, of the form

$$\mathbf{u} = (u(x, y, t),\, v(x, y, t), 0) \qquad p = p(x, y, t). \tag{20.3}$$

Flows of this form can be represented by a single scalar function, $\psi(x, y, t)$. The incompressibility condition (20.1) can be satisfied by writing

$$\mathbf{u} = \nabla \times (0, 0, \psi) \qquad u = \psi_y, \quad v = -\psi_x. \tag{20.4}$$

Then the vorticity, $\nabla \times \mathbf{u} = (0, 0, \omega)$ where

$$\omega = -\nabla^2 \psi. \tag{20.5}$$

Now taking the curl of (20.2), we can eliminate the pressure field. The resulting vorticity equation only has a z-component, which is

$$\omega_t + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega + G, \tag{20.6}$$

where $G = \hat{z} \cdot \nabla \times \mathbf{F}$. 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?

Nearly inviscid flow, $\nu \to 0$, High Reynolds Number

If ν 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.

Stokes flow: $\nu \to \infty$, Low Reynolds Number

If the flow is very viscous, or slow, the parameter $\nu >> 1$. In these circumstances the equations become

$$\nabla^2 \omega = 0, \qquad \nabla^2 \psi = -\omega. \tag{20.7}$$

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 $-\omega = \psi_{xx} + \psi_{yy} = \psi_{xx}$ on $x = 0$. We can therefore identify

$$\omega(0, y) = \frac{2f(y)}{h} - \frac{2\psi(h, y)}{h^2} + O(h), \tag{20.8}$$

So we can envisage an iterative scheme where we use ψ to find ω and then ω to find ψ. In place of (20.8) we could derive an $O(h^2)$ approximation using two points.

$$\begin{aligned} \psi(h) &= h\psi' + \tfrac{1}{2}h^2\psi'' + \tfrac{1}{6}h^3\psi''' + O(h^4) \\ \psi(2h) &= 2h\psi' + 2h^2\psi'' + \tfrac{4}{3}h^3\psi''' + O(h^4) \end{aligned} \tag{20.9}$$

Eliminating h''' we have $8\psi(h) - \psi(2h) = 6h\psi' + 2h^2\psi'' + O(h^4)$ and so

$$\omega(0, y) = \frac{3f(y)}{h} - \frac{8\psi(h, y) - \psi(2h, y)}{2h^2} + O(h^2), \tag{20.10}$$

Using such schemes we can solve Dirichlet problems for ω and ψ in turn.

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.

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' 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

$$\left.\begin{aligned} T_t + \mathbf{u}\cdot\nabla T &= \nabla^2 T \\ P^{-1}\left(\omega_t + \mathbf{u}\cdot\nabla\omega\right) &= RT_x + \nabla^2\omega \\ \omega &= -\nabla^2\psi. \end{aligned}\right\} \tag{20.11}$$

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. The Rayleigh number R is a measure of the thermal driving force. In deriving (20.11), 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;

6. Derive new boundary condition on ω from new ψ;

7. Repeat.

The equations (20.11) written out explicitly, for a slightly more generalised form is :

$$\left.\begin{aligned} T_t + uT_x + vT_y &= T_{xx} + T_{yy} \\ \omega_t + u\omega_x + v\omega_y &= P\left(\omega_{xx} + \omega_{yy}\right) + RP\left(T_x \cos\phi + T_y \sin\phi\right) \\ \psi_{xx} + \psi_{yy} &= -\omega, \end{aligned}\right\} \tag{20.12}$$

with ϕ representing an inclination angle, with (R, P) defined as follows:

$$R = \frac{g\beta D^3 (T_h - T_c)}{\nu\alpha}, \qquad P = \frac{\nu}{\alpha},$$

where D is a characteristic box size, β the thermal expansion coefficient, (T_c, T_h) temperatures of the cold and hot wall respectively, α is the thermal diffusivity and g the gravity.

Lecture 21. The 2D Wave Equation

We return now to pure Hyperbolic equations, such as the one-dimensional wave equation for $u(x,t)$,

$$u_{tt} = c^2 u_{xx} \qquad \text{with } u(x,0) \text{ and } u_t(x,0) \text{ known.} \tag{21.1}$$

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

$$U_n^{j+1} - 2U_n^j + U_n^{j-1} = q^2\left(U_{n+1}^j - 2U_n^j + U_{n-1}^j\right), \qquad \text{where } q = \frac{ck}{h}. \tag{21.2}$$

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 . We analyse the stability with three Fourier methods, looking at solutions $U_n^j = (\lambda)^j \exp(in\xi)$ and we find

$$\lambda^2 - 2\lambda + 1 = \lambda q^2(2\cos\xi - 2) \tag{21.3}$$

or

$$\lambda^2 - \left(2 - 4q^2\sin^2(\frac{1}{2}\xi)\right)\lambda + 1 = 0. \tag{21.4}$$

We need $|\lambda| \le 1$ for stability. 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

$$\left(2 - 4q^2\sin^2(\frac{1}{2}\xi)\right)^2 \le 4 \quad \rightarrow \quad -2 \le 2 - 4q^2\sin^2(\frac{1}{2}\xi) \le 2. \tag{21.5}$$

As usual, the worst case is $\xi = \pi$ and we require the Courant-Friedrichs-Lewy condition $q \le 1$ for stability, as we might have expected.

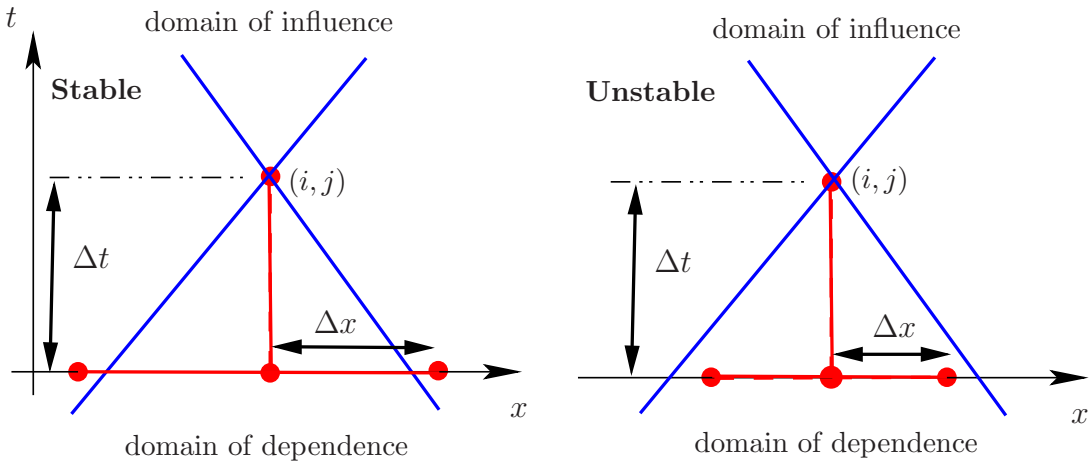

Figure 21.1: Domains of dependence and influnce, CFL-condition. Stable configuration (left), unstable configuration (right).

The scheme (21.2) is conservative and 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

$$cu_x = +u_t \qquad \text{on } \ x = 0. \tag{21.6}$$

This is easy enough to implement. We can combine (21.2) on the boundary $n = 0$ with

$$q(U_1^j - U_{-1}^j) = U_n^{j+1} - U_n^{j-1} \tag{21.7}$$

to eliminate the ghost points at $n = -1$, keeping 2nd order accuracy. Can we extend this scheme to two and three dimensions?

2D Wave equation: non reflecting boundary conditions

Suppose now $u(x, y, t)$ satisfies

$$u_{tt} = c^2\nabla^2 u = c^2(u_{xx} + u_{yy}). \tag{21.8}$$

We can model this explicitly on a rectangular grid (h_x, h_y) by

$$\frac{U_{mn}^{j+1} - 2U_{mn}^j + U_{mn}^{j-1}}{c^2k^2} = \frac{1}{h_x^2}\delta_x^2 U_{mn}^j + \frac{1}{h_y^2}\delta_y^2 U_{mn}^j. \tag{21.9}$$

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 $q \le 1/\sqrt{2}$. Similarly, in 3D, the CFL condition would be $q \le 1/\sqrt{3}$.

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 (21.1) can be written as a superposition of the waves $\exp(ilx + imy + i\omega t)$, where l, m and ω are related by

$$c^2(l^2 + m^2) = \omega^2. \tag{21.10}$$

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

$$mc = +\sqrt{\omega^2 - c^2l^2}. \tag{21.11}$$

The 1-D condition (21.6), is obtained by putting $l = 0$ in this formula, but this is not physically correct for waves which hit the boundary obliquely. The exact condition is difficult to implement because of the square root. A better approximation for small cl/ω is obtainable by expanding (21.11) to next order:

$$mc = \omega - \frac{c^2l^2}{2\omega}. \tag{21.12}$$

This is equivalent to the unusual boundary condition

$$cu_{yt} = u_{tt} - \frac{1}{2}c^2u_{xx} \qquad \text{on } \ y = 0. \tag{21.13}$$

This can be implemented on the boundary and greatly reduces unwanted reflections. But still it is imperfect. An important technique for avoiding unphysical reflections is known as **PML**, which stands for **Perfectly Matched Layer**.

Lecture 22. 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.

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 $\exp(ik(x - ct))$ 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. Let's consider a variable $X = x + if(x)$ where $f > 0$ and $f = 0$ for $x < 0$. Then the derivatives transform as

$$\frac{\partial}{\partial X} = \frac{1}{1 + if'(x)} \frac{\partial}{\partial x}, \tag{22.1}$$

The idea is to solve the equation in X-space and hope that the resultant solution has the desired decaying properties in x-space.

A more general wave equation

A generalised form of the wave equation for $p(\mathbf{x}, t)$ is

$$p_{tt} = b\nabla \cdot (\nabla p), \tag{22.2}$$

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 (22.2) can be written as a system of first order equations if we introduce the vector $\mathbf{v}$ such that

$$\mathbf{v}_t = a\nabla p, \quad \text{and} \quad p_t = b\nabla \cdot \mathbf{v}. \tag{22.3}$$

We will start with the 1-D wave equation in terms of X and t, so that

$$v_t = ap_X, \quad \text{and} \quad p_t = bv_X. \tag{22.4}$$

Assuming an $\exp(-i\omega t)$ behaviour and transforming from X to x, we have

$$-i\omega v \equiv v_t = \frac{ap_x}{1+if'} \quad \text{and} \quad -i\omega p \equiv p_t = \frac{bv_x}{1+if'}, \tag{22.5}$$

or multiplying up

$$-i\omega p + \omega f' p = bv_x \quad \text{and} \quad -i\omega v + \omega f' v = ap_x. \tag{22.6}$$

If we introduce $\sigma(x) = \omega f'$ and re-insert the time derivatives, we have the system

$$p_t = bv_x - \sigma p \quad \text{and} \quad v_t = ap_x - \sigma v. \tag{22.7}$$

We can choose a function $\sigma(x)$ which is zero for $x < 0$ and implement this scheme easily. 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 Figure 22.1.

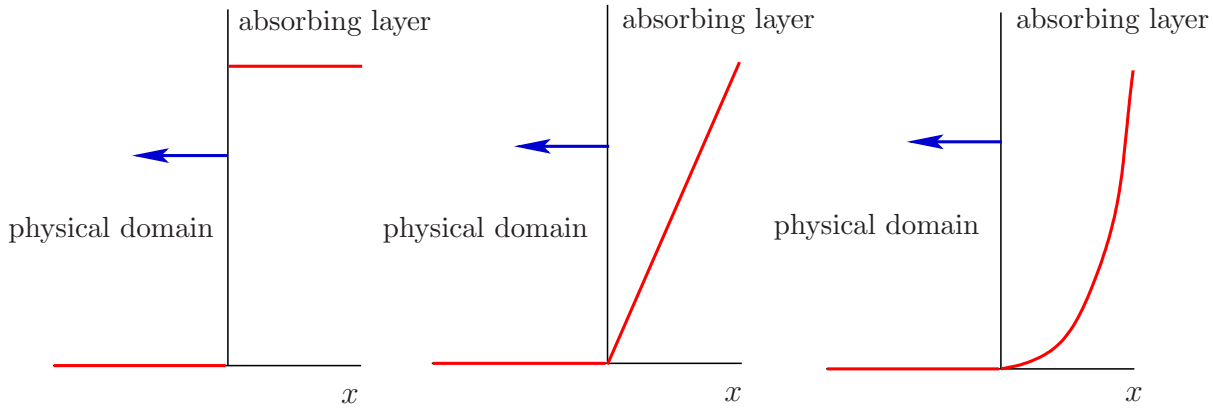

Figure 22.1: Choices for the absorption coefficient $\sigma(x)$.

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. What happens in 2D?

A perfectly matched layer in 2D

Now let's write $\mathbf{v} = (v, w)$. Our system takes the form

$$-i\omega p \equiv p_t = b\nabla \cdot \mathbf{v} \equiv bv_X = bw_y, \tag{22.8}$$

$$-i\omega v \equiv v_t = ap_X \quad \text{and} \quad -i\omega w \equiv w_t = ap_y. \tag{22.9}$$

Now (22.9b) does not involve X. Substituting for X in (22.9a) and (22.8) and multiplying up by $(1 + i\sigma/\omega)$, we have

$$bv_x + bw_y(1 + i\sigma/\omega) = -i\omega p + \sigma p, \tag{22.10}$$

$$ap_x = -i\omega v + \sigma v. \tag{22.11}$$

Equation (22.11) easily translates to time derivatives, but (22.10) involves $1/(i\omega)$ which corresponds to time integration rather than differentiation. So we introduce a function $\psi(x, y, t)$, such that

$$-i\omega\psi = b\sigma w_y \text{ with } \psi = 0 \text{ at } t = 0. \tag{22.12}$$

Then (22.10) reads $bv_x + bw_y + \psi = -i\omega p + \sigma p$, and we can write the system as

$$\left.\begin{aligned} p_t &= b\nabla \cdot \mathbf{v} - \sigma p + \psi \\ v_t &= ap_x - \sigma v \\ w_t &= ap_y \\ \psi_t &= b\sigma w_y. \end{aligned}\right\} \tag{22.13}$$

Where $\sigma = 0$, these equations reduce to (22.5). 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.

Lecture 23. 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 Figure 23.1. 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

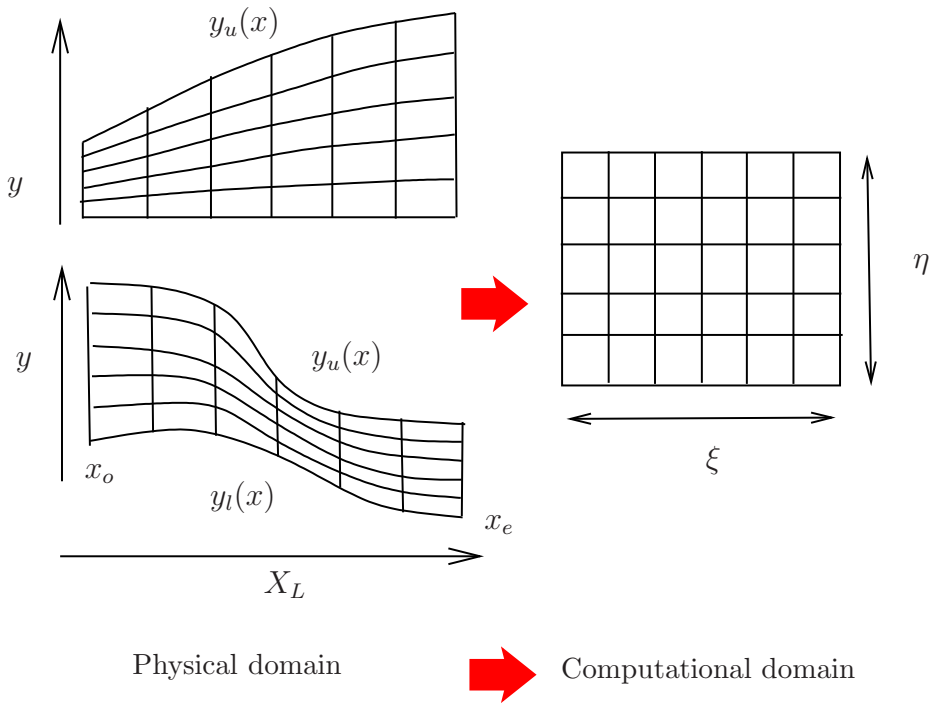

Figure 23.1: Mapping from physical plane to computational grid.

(mapping) of the physical coordinates (x, y) to computational plane (ξ, η), is one technique to ease the solution process. In Figure 23.1, one possible mapping could be

$$\xi = \frac{x - x_o}{x_e - x_o}, \qquad \eta = \frac{y - y_l(x)}{y_u(x) - y_l(x)}; \tag{23.1}$$

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:

$$\frac{\partial}{\partial x} = \frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\,; \qquad \frac{\partial}{\partial y} = \frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}; \tag{23.2}$$

and the second and mixed derivatives as follows:

$$\frac{\partial^2}{\partial x^2} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta}\right), \tag{23.3}$$

$$\frac{\partial^2}{\partial y^2} = \left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta}\right), \tag{23.4}$$

$$\frac{\partial^2}{\partial x \partial y} = \left(\frac{\partial \xi}{\partial x}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial x}\frac{\partial}{\partial \eta} \right) \left(\frac{\partial \xi}{\partial y}\frac{\partial}{\partial \xi} + \frac{\partial \eta}{\partial y}\frac{\partial}{\partial \eta} \right). \tag{23.5}$$

As can be seen from Figure 23.1, in the transformed grid, one can then make use of fixed discretisation step sizes $(\Delta\xi, \Delta\eta)$ in the transformed PDEs.

Alternative techniques to work in the physical plane are available, but we do not consider these in this course – (Google, finite elements and unstructured grids). Examples of quite complex problems are shown in Figure 23.2, these as you will appreciate are quite advanced.

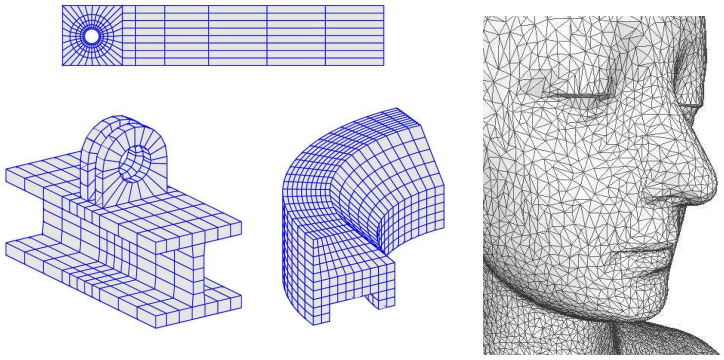

Figure 23.2: a: Finite element and unstructured grids in complex geometry (see: http://www.featool.com/tutorial/2015/08/24/creating-structured-grids-using-featool-matlab-functions. http://www.sci.utah.edu/the-institute/highlights/24-research-highlights/cibc-highlights/439-cleaver.html)

Grid Stretching

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

$$\epsilon \frac{d^2u}{dx^2} - \frac{du}{dx} = f(x), \quad \text{satisfying} \quad u(0) = \alpha, \quad u(1) = \beta, \tag{23.6}$$

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

$$u(x) = \alpha + x + (\beta - \alpha - 1)\left(\frac{e^{x/\epsilon} - 1}{e^{1/\epsilon} - 1} \right), \tag{23.7}$$

and solutions for varying ϵ with $f(x) = -1$ are shown in Figure 23.3. 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, chapter 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 (23.6). The solution is shown in Figure 23.4 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,

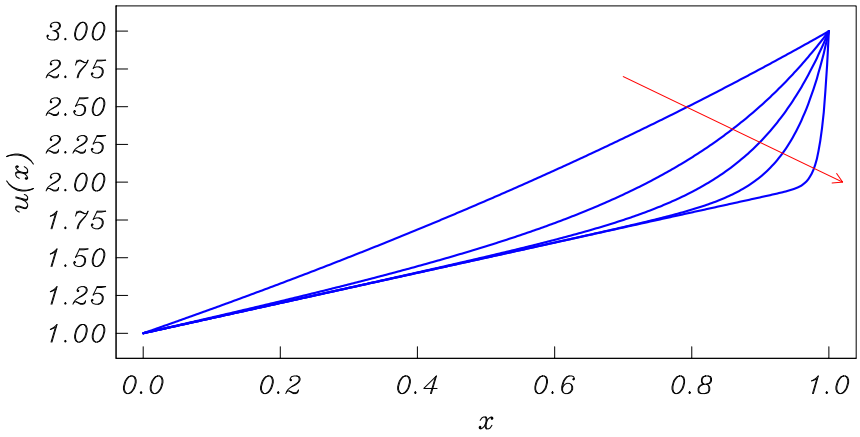

Figure 23.3: Exact solutions of (23.6), with $\epsilon = (1.0,\ 0.2,\ 0.1,\ 0.05,\ 0.01)$, arrow in direction of decreasing ϵ.

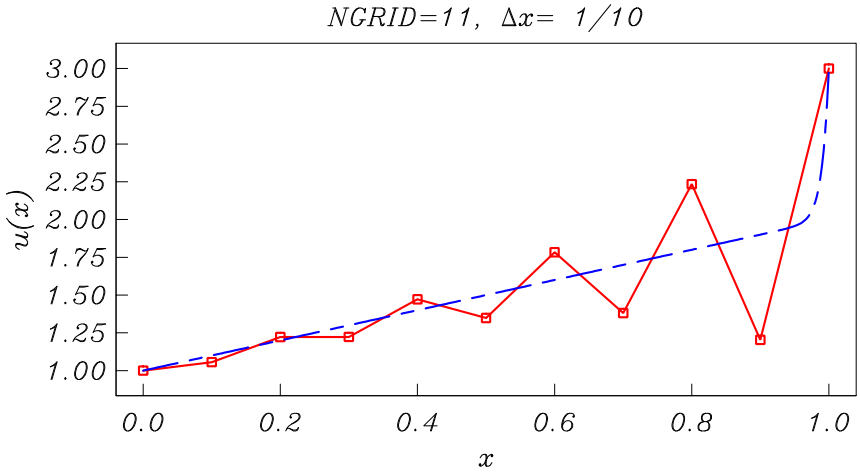

Figure 23.4: Numerical solution of (23.6), with $\epsilon = 0.01$), with $N = 11$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

with the numerical result also displaying "wiggles". On using successively finer grids, the numerical solution eventually agrees with the exact result, as shown in Figure's 23.5 & 23.6.

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 Figure 23.6, 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. (23.6), 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.

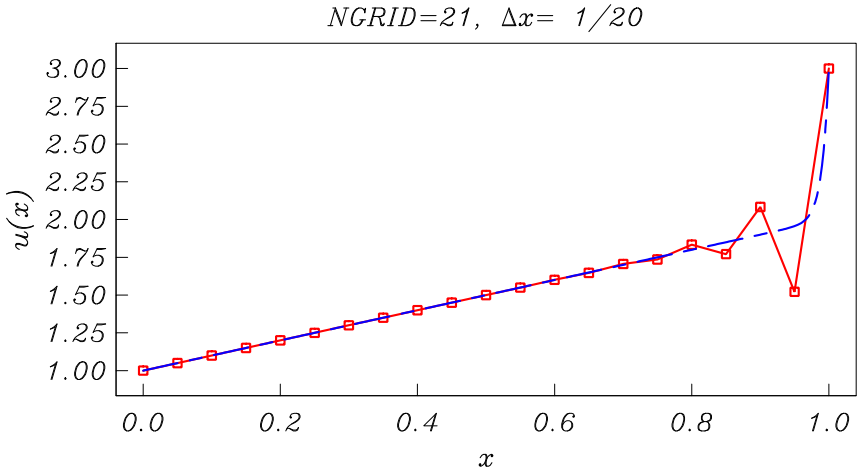

Figure 23.5: Numerical solution of (23.6), with $\epsilon = 0.01$), with $N = 21$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

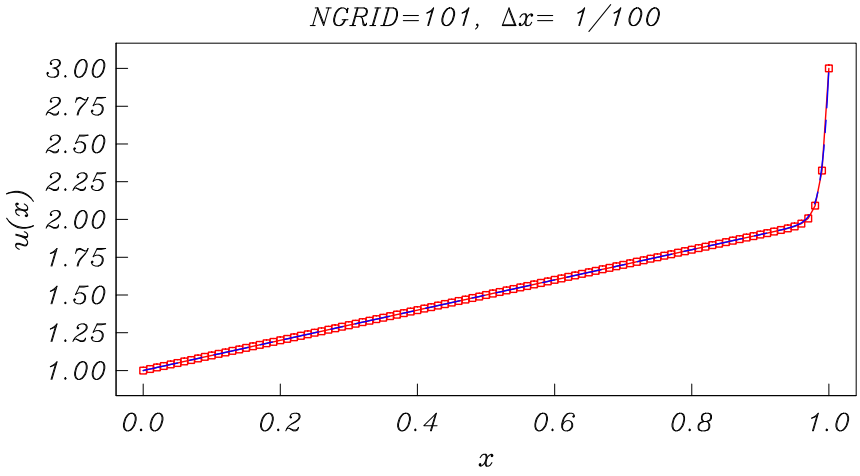

Figure 23.6: Numerical solution of (23.6), with $\epsilon = 0.01$), with $N = 101$ grid points, $\Delta x = 1/(N-1)$. Exact solution is given by the blue dashed line.

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 map:

$$x = c_2 + \frac{1}{c_1}\tan(\lambda\,[Z - \eta_o]), \tag{23.8}$$

where

$$\eta_o = \frac{z-1}{z+1}, \quad z = \frac{\tan^{-1}(c_1(1+c_2))}{\tan^{-1}(c_1(1-c_2))}, \quad \lambda = \frac{\tan^{-1}\left[c_1(1-c_2)\right]}{1-\eta_o}. \tag{23.9}$$

This maps $-1 \le Z \le 1$ to $0 \le x \le 1$. Some examples are shown in Figures 23.7- 23.9 – here we have applied a further linear uniform mapping whereby

$$\xi = (Z+1)/2 \tag{23.10}$$

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

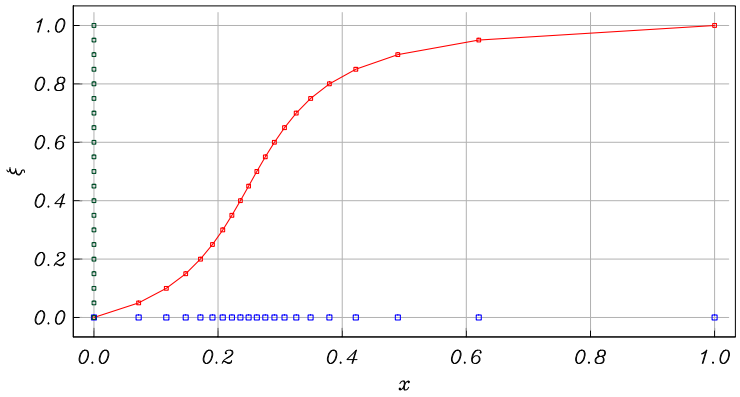

Figure 23.7: Grid mapping with $c_1 = -5$, and $c_2 = 0.25$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 0.25$

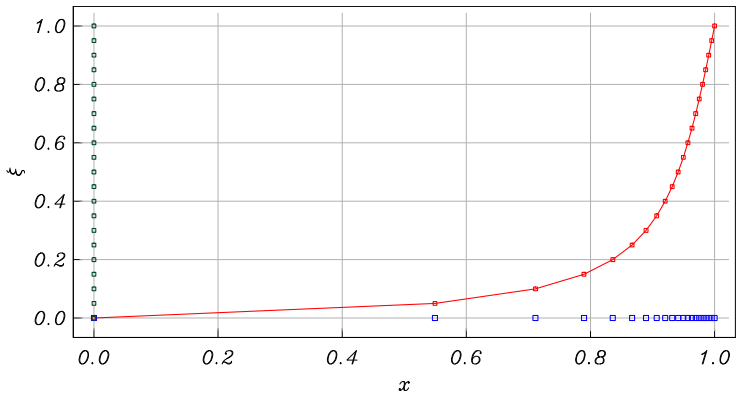

Figure 23.8: Grid mapping with $c_1 = -8$, and $c_2 = 1$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are most clustered at $x = 1.0$. In this plot the mapping and clustering is quite extreme.

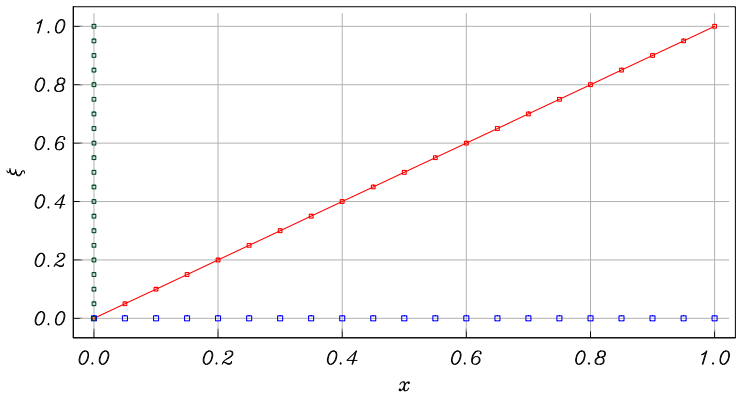

Figure 23.9: Grid mapping with $c_1 = -1 \times 10^{-22}$, and $c_2 = 0$. The Δx step variations are given by the blue symbols, while note the $\Delta\xi$ spacing is uniform and given in green symbols along the vertical ξ axis. In this mapping the points are uniformly distributed with a 1-to-1 correspondence.

Hence it therefore 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 Figure 23.10.

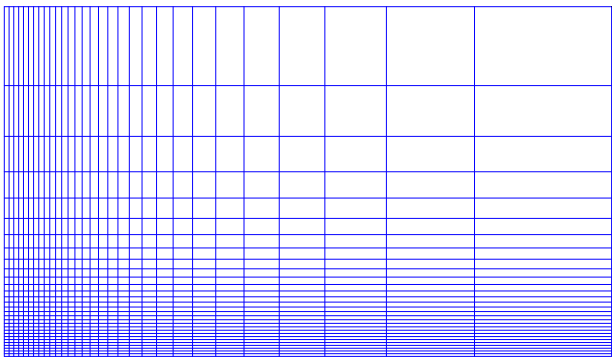

Figure 23.10: Example of 2D Grid clustering at the lower and left boundaries in the physical plane.

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, Expressions (23.2)–(23.5) thus require to be utilised and operated upon using expressions such as (23.8) and (23.10).

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.

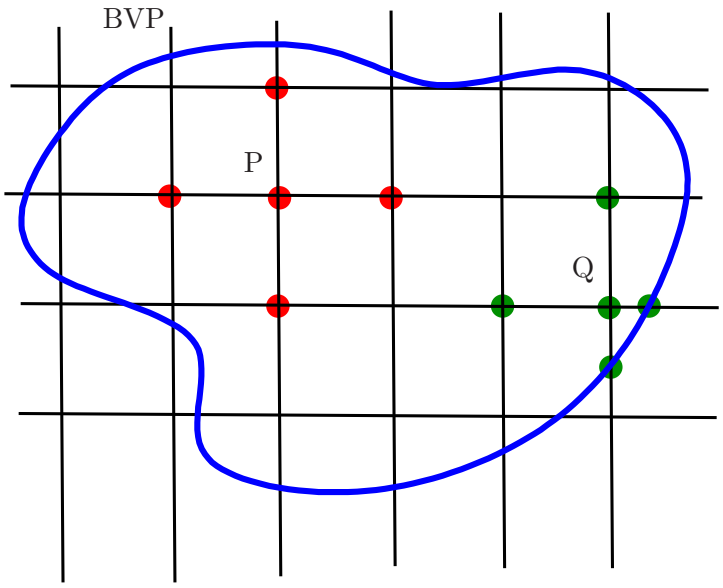

Figure 23.11: Irregular boundary without mappings.

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

$$\frac{1}{(\Delta x)^2}(u_{i+1,j} + u_{i-1,j}) + \frac{1}{(\Delta y)^2}(u_{i,j+1} + u_{i,j-1}) - (\frac{2}{(\Delta x)^2} + \frac{2}{(\Delta x)^2})u_{i,j} = f_{i,j}. \quad (23.11)$$

This discretisation may be used, as is, if all the grid points fall within the computational domain, as the P-data point in Figure 23.11. As one or more points cross the boundary/interface, such as point Q in Figure 23.11 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.

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.

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

$$\xi_{xx} + \xi_{yy} = \mathcal{P}(\xi, \eta), \qquad \eta_{xx} + \eta_{yy} = \mathcal{Q}(\xi, \eta), \quad (23.12)$$

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

$$\left.\begin{aligned} \alpha x_{\xi\xi} - 2\beta x_{\xi\eta} + \gamma x_{\eta\eta} &= J^2(\mathcal{P}x_\xi + \mathcal{Q}x_\eta) \\ \alpha y_{\xi\xi} - 2\beta y_{\xi\eta} + \gamma y_{\eta\eta} &= J^2(\mathcal{P}y_\xi + \mathcal{Q}y_\eta) \end{aligned}\right\}, \quad (23.13)$$

where

$$\left.\begin{aligned} \alpha &= x_\eta^2 + y_\eta^2 \\ \beta &= x_\xi x_\eta + y_\xi y_\eta \\ \gamma &= x_\xi^2 + y_\xi^2 \\ J &= x_\xi y_\eta - x_\eta y_\xi \end{aligned}\right\}, \quad (23.14)$$

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 control.

A typical example of what is though achievable is shown in Figure 23.13. Here with reference to Figure 23.12, 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 a–b is thus mapped to the lower a–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 (Laplace Equation) along with the equations (23.12) for the grid mappings are both solved using Successive Over Relaxation (SOR) discussed in earlier lectures.
(See Tannehill, Anderson & Pletcher, *Computational Fluid Mechanics and Heat Transfer*).

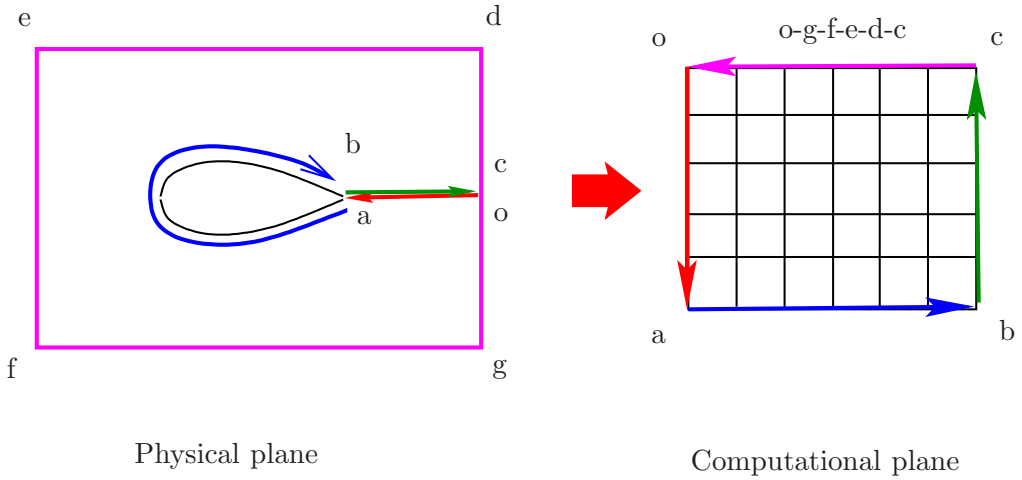

Figure 23.12: Example of complex boundary using Elliptic mapping.

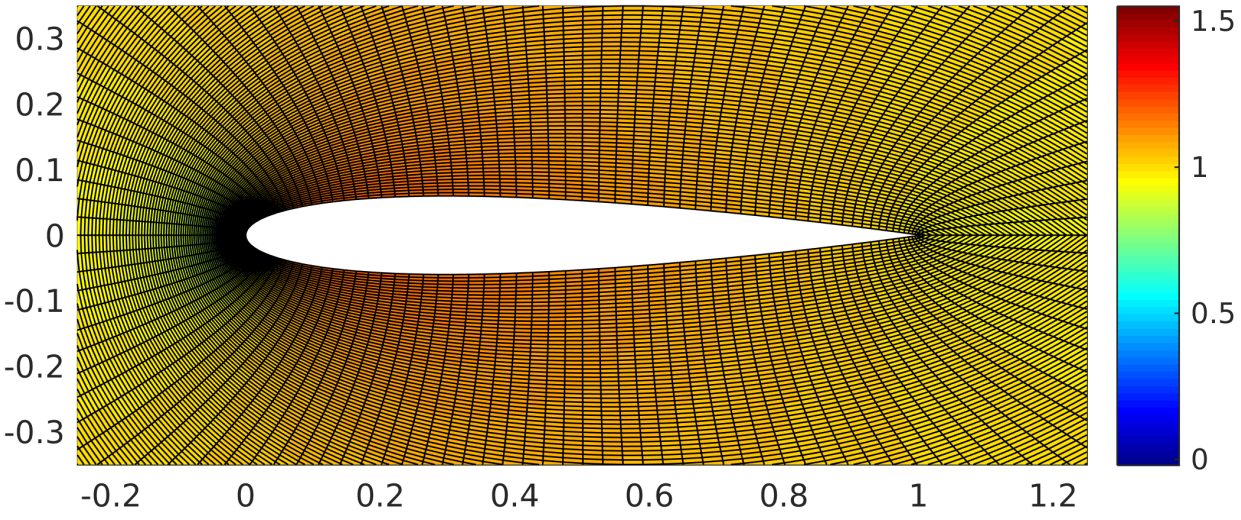

Figure 23.13: Examples of grid generated using (23.13). The strategy used is shown in Figure 23.12.

24. The Method of Characteristics

Recall lecture 2, which began:

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 $u(x,\, y)$

$$au_{xx} + bu_{xy} + cu_{yy} = f \ . \tag{24.1}$$

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 (24.1) is not necessarily **linear** – for example it is perfectly allowable in the discussion that follows that

$$f = du_x + eu_y + hu + g, \tag{24.2}$$

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. 24.1. 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,

$$\left.\begin{aligned} d(u_x) &= u_{xx}\, dx + u_{xy}\, dy \\ d(u_y) &= u_{xy}\, dx + u_{yy}\, dy \end{aligned}\right\}. \tag{24.3}$$

We combine (24.1) and (24.3) in matrix form:

$$\begin{pmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{pmatrix} \begin{pmatrix} u_{xx} \\ u_{xy} \\ u_{yy} \end{pmatrix} = \begin{pmatrix} f \\ d(u_x) \\ d(u_y) \end{pmatrix}, \tag{24.4}$$

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 (24.4) will have a unique solution for u_{xx}, u_{xy} and u_{yy} unless the determinant of that matrix vanishes, that is unless

$$\begin{vmatrix} a & b & c \\ dx & dy & 0 \\ 0 & dx & dy \end{vmatrix} = a(dy)^2 - b\, dxdy + c(dx)^2 = 0. \tag{24.5}$$

If (24.5) holds, the equation (24.4) 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

$$a\, \frac{d(u_x)}{dx} + c\, \frac{d(u_y)}{dy} = f \ . \tag{24.6}$$

*We use the shorthand notation u_x, u_y to represent $\partial u/\partial x$ and $\partial u/\partial y$ respectively. Similarly u_{xx}, u_{xy}, u_{yy} to represent $\partial^2 u/\partial x^2$, $\partial^2 u/\partial x \partial y$ and $\partial^2 u/\partial y^2$ respectively

This is surprising. If we can choose a direction (dx, dy) which satisfies (24.5) 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.

It is very important to know whether our solution can have this property. Equation (24.5) is called the **Characteristic equation** of (24.1). It is a quadratic in dy/dx with solution

$$\frac{dy}{dx} = \frac{b \pm \sqrt{b^2 - 4ac}}{2a} \ . \tag{24.7}$$

We can use equations (24.7) and (24.6) to find the solution. Equation (24.7) defines two families of characteristics, one with the +sign and one with the -sign. Starting from a given point (x, y) we can step (24.7) 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 +char from P_n^j and the -char from P_{n+1}^j as in the diagram, thus defining a new curve Γ^{j+1}.

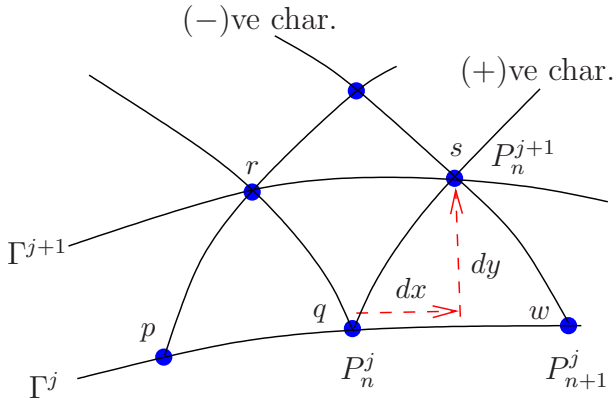

Figure 24.1: Problem description.

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 (24.7) 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+1}^j 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 (24.7) are real and distinct, and can be denoted by

$$\frac{dy}{dx} = F, \ \text{ and } \ \frac{dy}{dx} = G, \tag{24.8}$$

where initial values of u, u_x and u_y are known on Γ^j. As a first approximation, with reference to Figure 24.1, we may regard $p - r$ and $q - r$ as straight lines of slopes F_p and G_q. Then (24.8) can be approximated by

$$y_r - y_p = F_p(x_r - x_p), \ \text{ and } \ y_r - y_q = G_q(x_r - x_q), \tag{24.9}$$

giving two equations for the two unknowns (x_r, y_r). Next equation (24.1) can be used to show that the differentials $P = u_x$ and $Q = u_y$ are related by the equation

$$a\,\frac{dy}{dx}\,dP + c\,dQ - f\,dy = 0, \tag{24.10}$$

hence

$$a\,F\,dP + c\,dQ - f\,dy = 0, \;\text{ and }\; a\,G\,dP + c\,dQ - f\,dy = 0. \tag{24.11}$$

We approximate the first expression along $p - r$ by the equation

$$a_p\,F_p\,(P_r - P_p) + c_p\,(Q_r - Q_p) - f_p\,(y_r - y_p) = 0, \tag{24.12}$$

and the second expression along $q - r$ thus

$$a_q\,G_q\,(P_r - P_q) + c_q\,(Q_r - Q_q) - f_q\,(y_r - y_q) = 0. \tag{24.13}$$

Once (x_r, y_r) have been evaluated from (24.9), these are two equations for unknowns (P_r, Q_r). The value of u on r can then be obtained from

$$du = Pdx + Qdy, \tag{24.14}$$

on replacing values of (P, Q) along $p - r$ by

$$u_r - u_p = \frac{1}{2}(P_p + P_r)(x_r - x_p) + \frac{1}{2}(Q_p + Q_r)(y_r - y_p). \tag{24.15}$$

Solution of the above gives us a first approximation to u_r. We next improve upon this by replacing expressions (24.9) with new improved estimates of (x_r, y_r), using

$$y_r - y_p = \frac{1}{2}(F_p + F_r)(x_r - x_p), \;\text{ and }\; y_r - y_q = \frac{1}{2}(G_q + G_r)(x_r - x_q), \tag{24.16}$$

and equations (24.12-24.13) become

$$\left.\begin{aligned} \frac{1}{4}(a_p + a_r)\,(F_p + F_r)\,(P_r - P_p) + \frac{1}{2}(c_p + c_r)\,(Q_r - Q_p) - \frac{1}{2}(f_p + f_r)\,(y_r - y_p) = 0, \\ \frac{1}{4}(a_q + a_r)\,(G_q + G_r)\,(P_r - P_q) + \frac{1}{2}(c_q + c_r)\,(Q_r - Q_q) - \frac{1}{2}(f_q + f_r)\,(y_r - y_q) = 0. \end{aligned}\right\}. \tag{24.17}$$

An improved value for u_r then follows from (24.15). 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 – this is discussed in Lecture 23.

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 :

$$\left.\begin{array}{rcl} \frac{\partial u}{\partial x}+\frac{\partial v}{\partial y} & = & 0 \\ u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+\frac{1}{\rho}\frac{\partial P}{\partial x} & = & \frac{\mu}{\rho}\frac{\partial^2 u}{\partial y^2} \end{array}\right\}, \tag{25.1}$$

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. These equations to leading 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

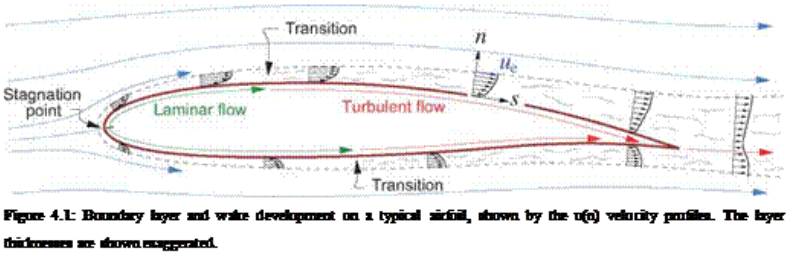

Figure 4.1: Boundary layer and wake development on a typical airfoil, shown by the u(n) velocity profiles. The layer thicknesses are shown exaggerated.

Figure 25.1: Navier-Stokes equations in action: boundary-layers.

order to account for the 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:

$$\eta = y\sqrt{\frac{\rho U_e(s)}{\mu s}}\ ;\quad \psi = \sqrt{s\rho\mu U_e(s)}\ f(\eta, s)\ ;\quad x = s\ . \tag{25.2}$$

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).
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 $m = 0$ and $c = 1/2$ 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 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.

*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.

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:

$$\left.\begin{array}{ccc} F^{(\alpha)}\,, & \frac{\partial F^{(\alpha)}}{\partial \eta}\,, & \frac{\partial F^{(\alpha)}}{\partial s} \\ F^{(\alpha)}F^{(\beta)}\,, & F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial \eta}\,, & F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial s} \end{array}\right\}, \tag{25.13}$$

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

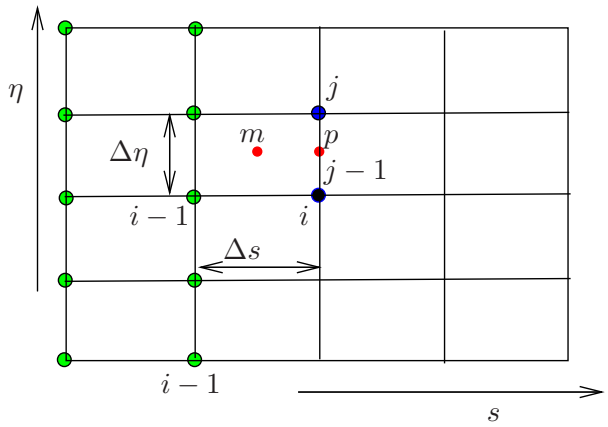

Figure 25.2: Discretised grid notation. We assume solutions at earlier data planes $i-1$ are known and we seek to compute a solution at the i'th data plane for all j grid nodes.

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:

- $F^{(\alpha)}$ 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

$$F^{(\alpha)} = \frac{1}{4}\left(\bar{F}^{(\alpha)}_{i-1/2,j-1/2} + \hat{F}^{(\alpha)}_{i,j-1} + \hat{F}^{(\alpha)}_{i,j}\right), \tag{25.17}$$

- $\frac{\partial F^{(\alpha)}}{\partial \eta}$ 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

$$\frac{\partial F^{(\alpha)}}{\partial \eta} = \frac{1}{2k}\left(\bar{F}^{(\alpha)}_{i-1/2,j} - \bar{F}^{(\alpha)}_{i-1/2,j-1} + \hat{F}^{(\alpha)}_{i,j} - \hat{F}^{(\alpha)}_{i,j-1}\right), \tag{25.20}$$

- $\frac{\partial F^{(\alpha)}}{\partial s}$ 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:

$$\frac{\partial F^{(\alpha)}}{\partial s} = \frac{1}{2h}\left(\bar{F}^{(\alpha)}_{i,j-1/2} - \bar{F}^{(\alpha)}_{i-1,j-1/2} + \hat{F}^{(\alpha)}_{i,j} + \hat{F}^{(\alpha)}_{i,j-1}\right). \tag{25.22}$$

- $F^{(\alpha)}F^{(\beta)}$ 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.

- $F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial s}$ 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

$$\begin{aligned}\tfrac{1}{8h}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\left(\bar{F}^{(\beta)}_{i,j-1/2}-\bar{F}^{(\beta)}_{i-1,j-1/2}\right) &+ \tfrac{1}{8h}\left(\bar{F}^{(\beta)}_{i,j-1/2}-\bar{F}^{(\beta)}_{i-1,j-1/2}\right)\left(\hat{F}^{(\alpha)}_{i,j-1}+\hat{F}^{(\alpha)}_{i,j}\right)\\ &+ \tfrac{1}{8h}\bar{F}^{(\alpha)}_{i-1/2,j-1/2}\left(\hat{F}^{(\beta)}_{i,j}+\hat{F}^{(\beta)}_{i,j-1}\right).\end{aligned} \tag{25.26}$$

- $F^{(\alpha)}\frac{\partial F^{(\beta)}}{\partial \eta}$ 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 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 nonlinear 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.