Numerical Computation Notes

I. J. Sobey
Computing Laboratory, Oxford University
Fellow in Engineering, St John's College

November 21, 2000

CONTENTS

1

DISCRETISATION OF PDE'S

Much impetus for the development of supercomputing facilities has been to enable calculation of approximate solutions of PDE's, particularly those occuring for large physical systems such as weather, flight or ship hydrodynamics. It is important to realise that there are different levels of approximation which lead to a physical prediction such as drag. The real, physical situation has to be modelled mathematically and that will introduce approximations such as two-dimensionality, semi-infinite domains and so on. Next the resulting differential equations from the model need to be discretised intoducing another set of approximations. Finally the discrete system needs to be solved and again this is often only done approximately, either because physical constants may be known only approximately or because the discrete system has only an approximate numerical solution. You need to be aware that 'numerical algorithm' can be used in a number of different contexts and is often used to refer to discretisation of a continuous differential model and to refer to how you obtain an approximate solution of a discrete system.

In describing the variation of physical quantities we use functions: displacement as a function of time and so on. Thus it is important to think about how functions come about. If we consider a simple example, $y = \sin t$, then $y(t)$ could equally well be defined by

(a) geometrically by using a unit circle,

(b) algebraically by using a power series,

(c) as the solution of a differential equation

$$y'' + y = 0, \ y(0) = 0, \ y'(0) = 1. \tag{1.1}$$

Most of the functions we use in physics and engineering are defined only by the last method: they are defined by a differential equation. Our experience is that most such equations have to be solved numerically.

Another question which should occur to you is why do we have definitions of functions in terms of differential equations in the first place? The main reason is because in engineering and physics we are usually looking at functions which are solutions of conservations laws; laws which relate rates of change of quantities in space and time. Such laws usually fall into only a few simple classes.

Conservation laws

(a) steady problem without sources: what comes into a region must leave.

So if $\mathbf{q}$ is the flux of some quantity ϕ, in two dimensions,

$$\nabla.\mathbf{q} = 0. \tag{1.2}$$

If then there is a relation such as $\mathbf{q} = -\kappa\nabla\phi$ for a potential (or temperature or ...) then

$$\nabla(\kappa\nabla\phi) = 0 \tag{1.3}$$

or if κ is constant, $\nabla^2\phi = 0$, a simple elliptic equation.

(b) unsteady problem: what comes into a volume either stays (and changes the amount in a volume) or leaves.

$$\frac{\partial\phi}{\partial t} + \nabla.\mathbf{q} = 0. \tag{1.4}$$

So depending on the relation between the flux and the quantity ϕ we get most of the simple PDE's which you have studied in your mathematics courses. As an example, if the flux is determined by diffusion, $\mathbf{q} = -\kappa\nabla\phi$, then

$$\frac{\partial\phi}{\partial t} = \kappa\nabla^2\phi. \tag{1.5}$$

Hence systems of partial differential equations can be just the mathematical expression of conservation laws, and it is very common to end up with two space derivatives (∇^2 is the divergence of a gradient).

Our plan is to examine how simple differential systems can be discretised (approximate solution of the discretised system will come in section 2) in a number of ways, some of which reflect the physical basis of the equations. We do this via a simple model problem and a series of observations which extend the model problem to cover many aspects of numerical solution of PDE's.

1.1 Model Problem

A simple model problem needs to have a second derivative and a source term, both common in conservation laws. The simplest such model is as follows.

Suppose $u(x)$ is twice differentiable and satisfies

$$-u'' + \kappa^2 u = f(x),\ \ 0 \le x \le 1, \tag{1.6}$$

with $u(0) = a$ and $u(1) = b$.

Although the differential equation can be solved analytically it needs to be treated as if a computed solution has to be found. There are a number of ways of discretising this model equation.

1.1.1 *Finite Differences*

The definition of derivative can be used to obtain a discrete approximation, since the definition can be expressed in a number of different ways, so to can the discrete approximation. The definition in terms of limits can be any of

$$u'(x) = \lim_{\Delta x \to 0} \frac{u(x+\Delta x) - u(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{u(x) - u(x-\Delta x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{u(x+\Delta x) - u(x-\Delta x)}{2\Delta x}. \quad (1.7)$$

Define a set of points $\Omega_J = \{x_0 = 0 < x_1 \ldots < x_m = 1\}$ and a set of nodes $J_\Omega = \{0, 1, \ldots, m\}$ and approximate $u_j = u(x_j)$ by U_j, $\forall j \in J_\Omega$. Then discrete approximations for a derivative are:

$$u'(x_j) \sim \frac{U_{j+1} - U_j}{x_{j+1} - x_j} = \frac{U_{j+1} - U_j}{h} \text{ if } x_j = jh, \quad (1.8)$$

$$u'(x_j) \sim \frac{U_j - U_{j-1}}{x_j - x_{j-1}} = \frac{U_j - U_{j-1}}{h} \text{ if } x_j = jh, \quad (1.9)$$

$$u'(x_j) \sim \frac{U_{j+1} - U_{j-1}}{x_{j+1} - x_{j-1}} = \frac{U_{j+1} - U_{j-1}}{2h} \text{ if } x_j = jh, \quad (1.10)$$

For simplicity take $x_j = jh$ with $h = 1/m$ unless it is indicated otherwise.

Higher derivatives can be approximated in a similar way:

$$u''(x) \sim \frac{u'(x_j + \frac{1}{2}h) - u'(x_j - \frac{1}{2}h)}{h}, \quad (1.11)$$

so that

$$u'' \sim \frac{\frac{U_{j+1} - U_j}{h} - \frac{U_j - U_{j-1}}{h}}{h} = \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2}. \quad (1.12)$$

Now if $f_j = f(x_j)$, $j = 1, \ldots, m-1$, $f_0 = a$ and $f_m = b$,

$$U_0 = f_0, \quad (1.13)$$

$$-\frac{U_{j+1} - 2U_j + U_{j-1}}{h^2} + \kappa^2 U_j = f_j, \quad j = 1, \ldots, m-1, \quad (1.14)$$

$$U_m = f_m. \quad (1.15)$$

If we let $\mathbf{U}^{\mathrm{T}} = (U_0, U_1, \ldots, U_m)$ be a vector of approximate values, define matrices $\mathbf{K}$, $\mathbf{I}^*$,

$$\mathbf{K} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ -1 & 2 & -1 & 0 \cdots 0 \\ 0 & -1 & 2 & -1 \cdots 0 \\ 0 & 0 & -1 & 2 \cdots 0 \\ . & . & . & \cdots . \\ 0 & 0 & 0 & 0 \cdots 1 \end{pmatrix}, \quad (1.16)$$

$$\mathbf{I}^* = \begin{pmatrix} 0\,0\,0\,0 \cdots 0\,0 \\ 0\,1\,0\,0 \cdots 0\,0 \\ 0\,0\,1\,0 \cdots 0\,0 \\ .\ .\ .\ .\ .\,.\,.\,.\ . \\ 0\,0\,0\,0 \cdots 1\,0 \\ 0\,0\,0\,0 \cdots 0\,0 \end{pmatrix}. \tag{1.17}$$

Then we can write the finite difference scheme as

$$(\frac{1}{h^2}\mathbf{K} + \kappa^2 \mathbf{I}^*)\mathbf{U} = \mathbf{f}. \tag{1.18}$$

This is a discrete version of the original differential equation (1.6).

It is possible to reduce the matrices to $m-1 \times m-1$ by eliminating the explicitly given values of U_0, U_m. On the other hand, if one of the boundary conditions involved a derivative (corresponding to a physical situation where as an example, heat transfer rate is specified) then keeping these values as "unkowns" is needed; overall keeping them in the matrix is a very small overhead.

1.1.1.1 *Note 1 Derivative Boundary Conditions* If the boundary condition at $x = 0$ is for instance $u'(0) = 0$ we proceed by introducing a fictitious point U_{-1}, so that we have

$$\text{(a) differential equation : } -\frac{U_1 - 2U_0 + U_{-1}}{h^2} + \kappa^2 U_0 = f_0, \tag{1.19}$$

$$\text{(b) boundary condition : } U_1 - U_{-1} = 0, \tag{1.20}$$

and eliminating U_{-1},

$$-\frac{2U_1 - 2U_0}{h^2} + \kappa^2 U_0 = f_0. \tag{1.21}$$

$\mathbf{K}$, $\mathbf{I}^*$ become

$$\mathbf{K} = \begin{pmatrix} 2 & -2 & 0 & 0 \cdots 0 \\ -1 & 2 & -1 & 0 \cdots 0 \\ 0 & -1 & 2 & -1 \cdots 0 \\ 0 & 0 & -1 & 2 \cdots 0 \\ . & . & . & . \cdots . \\ 0 & 0 & 0 & 0 \cdots 1 \end{pmatrix}, \mathbf{I}^* = \begin{pmatrix} 1\,0\,0\,0 \cdots 0\ \ 0 \\ 0\,1\,0\,0 \cdots 0\ \ 0 \\ 0\,0\,1\,0 \cdots 0\ \ 0 \\ .\ .\ .\ .\ .\,.\,.\,.\ .\ \ . \\ 0\,0\,0\,0 \cdots 1\ \ 0 \\ 0\,0\,0\,0 \cdots 0\,0 \end{pmatrix}, \tag{1.22}$$

1.1.1.2 *Note 2 Unsteady PDE's* If the problem is unsteady, for instance

$$\frac{\partial u}{\partial t} = \sigma \frac{\partial^2 u}{\partial x^2} - \kappa^2 u + f(x,t), \tag{1.23}$$

then introduce a set of time points $t_0, t_1, \ldots$ (usually assuming $t_n = n\Delta t$) and let U_j^n approximate $u(x_j, t_n)$. Than approximate

$$\frac{\partial u}{\partial t} \sim \frac{U_{n+1} - U_n}{t_{n+1} - t_n} = \frac{U_j^{n+1} - U_j^n}{\Delta t} \text{ if } t_n = n\Delta t. \tag{1.24}$$

It is now possible to use a variety of schemes.

(a) **Explicit Scheme**

If we approximate the RHS of the differential equation at time t_n, and let $\mu = \sigma\Delta t/h^2$ and define a vector $\mathbf{U}^n$ at each time step,

$$\mathbf{U}^{n+1} = (\mathbf{I} - \kappa^2\Delta t\mathbf{I}^* + \mu\mathbf{K})\mathbf{U}^n + \Delta t\mathbf{f}, \tag{1.25}$$

the vector U^{n+1} is given explicitly in terms of known quantities so there is no matrix problem to solve.

(b) **Implicit Scheme**

Approximate the RHS of the differential equation at time t_{n+1},

$$(\mathbf{I} + \kappa^2\Delta t\mathbf{I}^* - \mu\mathbf{K})\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\mathbf{f}^{n+1}. \tag{1.26}$$

Now a matrix problem has to be solved at each time step.

(c) **θ-method**

We approximate the right hand side as a weighted sum of values at t_n and t_{n+1}, using $\theta\mathrm{RHS}^{n+1} + (1-\theta)\mathrm{RHS}^n$,

$$[\mathbf{I} + \theta\kappa^2\Delta t\mathbf{I}^* - \theta\mu\mathbf{K}]\mathbf{U}^{n+1} = [\mathbf{I} - (1-\theta)\kappa^2\Delta t\mathbf{I}^* + (1-\theta)\mu\mathbf{K}]\mathbf{U}^n + \theta\mathbf{f}^{n+1} + (1-\theta)\mathbf{f}^n. \tag{1.27}$$

The case $\theta = 1/2$ is called Crank–Nicholson and has a higher order of accuracy in approximating the time derivative than the other cases (see note 4 below).

1.1.1.3 *Note 3 Two Space Dimensions* If steady but two-dimensional, for instance

$$-\nabla^2 u + \kappa^2 u = f \quad \text{on } \Omega = [0,1] \times [0,1], \tag{1.28}$$

let $U_{r,s}$ approximate $u(r\Delta x, s\Delta y)$, $r = 0, 1, \ldots, m_x$, $s = 0, 1, \ldots, m_y$ where $\Delta x = 1/m_x$, $\Delta y = 1/m_y$, and approximate

$$\frac{\partial^2 u}{\partial x^2} \sim \frac{U_{r+1,s} - 2U_{r,s} + U_{r-1,s}}{\Delta x^2}, \tag{1.29}$$

$$\frac{\partial^2 u}{\partial y^2} \sim \frac{U_{r,s+1} - 2U_{r,s} + U_{r,s-1}}{\Delta y^2}. \tag{1.30}$$

The discrete form of the differential equation is

$$\frac{U_{r+1,s} - 2U_{r,s} + U_{r-1,s}}{\Delta x^2} + \frac{U_{r,s+1} - 2U_{r,s} + U_{r,s-1}}{\Delta y^2} + \kappa^2 U_{r,s} = f_{r,s}, \tag{1.31}$$

Now we need to define the **vector U** to include all the elements of the **array** $U_{r,s}$. One way is just to enumerate the elements of $\mathbf{U}$,

$$\mathbf{U}^{\mathrm{T}} = \{U_{0,0}, U_{1,0}, \cdots, U_{m_x,0}, U_{0,1}, \cdots, U_{m_x,m_y}\}, \tag{1.32}$$

another is to suppose there is a map $\mathcal{M}(r,s)$ which maps the pairs (r,s) to the integers $0, 1, 2, \cdots, (m_x+1)(m_y+1) - 1$. Either way, $\mathbf{U}$ is a vector of dimension

$(m_x+1)(m_y+1)$ - a partial explanation of why supercomputers are needed for anything like realistic problems. Even using only 101×101 points means we have to determine a vector of length $\sim 10^4$ by inverting a matrix system where the coefficient matrix is roughly of size $10^4 \times 10^4$. If we now define one row of a matrix $\mathbf{K}$ by

$$\mathrm{row}(\mathbf{K}) = \{0, \cdots, 0, -\frac{1}{\Delta x^2}, 0, \cdots, 0, -\frac{1}{\Delta y^2}, \frac{2}{\Delta x^2} + \frac{2}{\Delta y^2} + \kappa^2, -\frac{1}{\Delta x^2}, 0, \cdots, 0, -\frac{1}{\Delta y^2}, 0, \cdots, 0\}, \tag{1.33}$$

with $\mathbf{I}^*$ suitably defined, we still have a matrix system

$$(\mathbf{K} + k^2\mathbf{I}^*)\mathbf{U} = \mathbf{f}, \tag{1.34}$$

to invert in order to calculate $\mathbf{U}$. In a later part we shall consider algorithms which can invert matrix systems such as this where it is impossible to store the whole coefficent matrix on a computer; all we are able to store is the vector obtained when the matrix multiplies another vector.

1.1.1.4 *Note 4 Error Analysis* If we write the continuous system as

$$\mathcal{L}u = f, \tag{1.35}$$

and the discrete system as

$$\mathrm{L}_h U_j = f_j, \tag{1.36}$$

where $\mathcal{L}$, L_h are **linear** operators, then define a truncation error

$$T_j = (\mathrm{L}_h u - f)|_{x=jh}, \tag{1.37}$$

and an error

$$e_j = u_j - U_j. \tag{1.38}$$

We then have

$$T_j = (\mathrm{L}_h u - f)|_{x=jh} = (\mathrm{L}_h u - \mathrm{L}_h U)|_{x=jh} = \mathrm{L}_h e_j. \tag{1.39}$$

This is a central result in analysis of errors associated with discretisation. In principle, to determine the magnitude of errors, we have only to know the 'magnitude' (that is some form of norm) of the inverse of L_h and the 'magnitude' of the truncation error.

In our model problem

$$\mathrm{L}_h U_j = -\frac{U_{j+1} - 2U_j + U_{j-1}}{h^2} + \kappa^2 U_j, \tag{1.40}$$

so that

$$\mathrm{L}_h u_j = -\frac{u_{j+1} - 2u_j + u_{j-1}}{h^2} + \kappa^2 u_j, \tag{1.41}$$

and using Taylor expansions,

$$u_{j+1} = u(x_j + h) = u_j + hu'_j + \frac{h^2}{2}u''_j + \dots, \tag{1.42}$$

$$u_{j-1} = u(x_j - h) = u_j - hu'_j + \frac{h^2}{2}u''_j + \dots, \tag{1.43}$$

and hence

$$T_j = -u''_j + k^2 u_j - f_j - \frac{1}{12}h^2 u''''_j + \dots, \tag{1.44}$$

the first three terms are just the differential equation evaluated at x_j and so add to zero; with a bit more work, we can show

$$T_j = -\frac{1}{12}h^2 u''''(\xi_j), \quad (j-1)h < \xi_j < (j+1)h, \tag{1.45}$$

so that if $M = \max|u''''|$, then

$$|T_j| \leq \frac{1}{12}h^2 M. \tag{1.46}$$

If $T \to 0$ as $h \to 0$ the scheme is called consistent.

If $T \sim h^p$ as $h \to 0$ the scheme is called order-p accurate. In the model problem, the scheme is second order accurate. With PDE's, it is possible for the accuracy to be of different order for different independent variables. In the problem considered in note 2 above, the scheme is second order accurate in h but only first order accurate in Δt, unless using crank–Nicholson when it is second order accurate in Δt.

If we define $\mathbf{e}^{\mathrm{T}} = \{0, e_1, \dots, e_{m-1}, 0\}$, $\mathbf{T}^{\mathrm{T}} = \{0, T_1, \dots, T_{m-1}, 0\}$, then in matrix notation

$$(\frac{1}{h^2}\mathbf{K} + \kappa^2 \mathbf{I}^*)\mathbf{e} = \mathbf{T}, \tag{1.47}$$

and formally

$$\mathbf{e} = h^2(\mathbf{K} + \kappa^2 h^2 \mathbf{I}^*)^{-1}\mathbf{T}, \tag{1.48}$$

so we need an estimate of the norm of $(\mathbf{K} + \kappa^2 h^2 \mathbf{I}^*)^{-1}$. This turns out to be $O(m^2) = O(h^{-2})$ in our model problem so that $||\mathbf{e}|| = O(h^2)$ as $h \to 0$.

1.1.2 *Finite Volume Methods*

It has already been observed thay many differential equations which we owuld like to solve come from conservation laws which are integrals over volumes. This idea has been carried onto the discretisation of such equations by instead of interpreting U_j as an approximation to a point value, $u(x_j)$, rather

$$U_j \sim \frac{1}{h}\int_{(j-1/2)h}^{(j+1/2)h} u(x)\mathrm{d}x. \tag{1.49}$$

In one sense we now reverse the process by which we arrived at the differential equation from a conservation law: take the differential equation and integrate

$$\int_{(j-1/2)h}^{(j+1/2)h} (-u'' + \kappa^2 u - f)\mathrm{d}x = 0. \tag{1.50}$$

If we integrate the first term and define

$$\bar{f}_j = \frac{1}{h}\int_{(j-1/2)h}^{(j+1/2)h} f(x)\mathrm{d}x, \tag{1.51}$$

then

$$-u'|_{(j-1/2)h}^{(j+1/2)h} + \kappa^2 h U_j - h\bar{f}_j = 0. \tag{1.52}$$

Alternately we can think of applying a conservation argument to the 'volume' $((j-\frac{1}{2})h, (j+\frac{1}{2})h)$. The difficulty now is to represent the 'fluxes' across the faces of the volume at $(j-\frac{1}{2})h$, $(j+\frac{1}{2})h$ in terms of the integral quantities, U_j. In this simple case we approximate

$$u'((j+\frac{1}{2})h) \sim \frac{U_{j+1} - U_j}{h}, \tag{1.53}$$

so that we end up with a coefficient matrix which is identical to the finite difference one we derived above, but the right hand side is now a vector of integrals of f,

$$(\frac{1}{h^2}\mathbf{K} + \kappa^2\mathbf{I}^*)\mathbf{U} = \bar{\mathbf{f}}. \tag{1.54}$$

While this appears very similar to an ordinary finite difference method, if using unstructured meshes in two and three space dimensions (so that the 'volumes' are either arbitrary triangles or quadrilaterals) the finite volume method is much easier to apply than conventional finite differences.

1.1.3 *Finite Element Method*

The finite element method has now quite a long and interesting history. It was originally developed by engineers dealing with structural analysis. They realised that if say a loaded plate was divided into small triangles or quadrilaterals ('elements') and the displacement within an element was assumed to be linear in space variables, then the stress could be calculated throughout the element and the sum of all the stresses would have to agree with the loading. Hence they obtained a linear combination of displacements equal to the loading, and inverting this system gave the displacements and then the stresses in the structure. It was quickly realised that this very practical method could be put in a sound mathematical framework, and once that framework was developed, the method could be applied to problems other than structural analysis. We shall look only at the modern understanding of finite elements. It is important to distinguish our understanding of how finite element analysis approximates the solution of a differential system from the details of how the method is implemented.

We seek a solution of $\mathcal{L}u = f$ on $x \in \Omega$ and its associated boundary conditions. In finite differences we tried to find a set of values $\{U_j\}$ which approximated u at a set of points in the domain. Now ask the question. Suppose we have a set of simple **functions**, can we approximate u using these functions? In abstract terms, if $u \in \mathcal{A}$ and we have a set of (possibly) simple functions $\mathcal{B} \subset \mathcal{A}$ can we find a function $U \in \mathcal{B}$ which is a good approximation to u? Obviously this immediately points us towards needing some way of measuring how good one function represents another: that is we need some way of measuring distance in the set of functions $\mathcal{A}$. One way to achieve this is to use an inner product, for instance if $\omega > 0$ we can define

$$(u, v) = \int_\Omega \omega(x)u(x)v(x)\mathrm{d}x, \tag{1.55}$$

to be an inner product and then we can measure the size of a function by

$$||u||_2 = \sqrt{(u, u)}, \;\; u \in \mathcal{A} \tag{1.56}$$

and 'distance' between functions by

$$d(u, v) = ||u - v||_2, \quad u, v \in \mathcal{A}. \tag{1.57}$$

The definition of 'distance' is not unique, for the inner product example, the weight function ω will affect the calculation of 'distance'.

The approximation problem can now be posed very precisely: Given $u \in \mathcal{A}$ and a subset $\mathcal{B} \subset \mathcal{A}$ can we find $U \in \mathcal{B}$ such that

$$||u - U||_2 \leq ||u - V||_2, \quad \forall V \in \mathcal{B}? \tag{1.58}$$

In the case of a norm based on an inner product we can go someway to answering this question. Since

$$||u - V||_2^2 = ||u - U + U - V||_2^2 = (u - U + U - V, u - U + U - V), \tag{1.59}$$

we have

$$||u-V||_2^2 = ||u-U||_2^2 + ||U-V||_2^2 + 2(u-U, U-V). \quad (1.60)$$

Since the last term could be positive or negative, whereas the second last term is alsways positive, in order for

$$||u-U||_2 \leq ||u-V||_2 \ \ \forall V \in \mathcal{B}, \quad (1.61)$$

we need only

$$(u-U, W) = 0 \ \ \forall W \in \mathcal{B}. \quad (1.62)$$

This is now called Galerkin orthogonality. It is just a generalisation of our physical intuition that in coordinate geometry, if you wanted the point on a straight line closest to a curve, you would make the line from the approximating point on the straight line to the curve perpendicular to the straight line.

Before we finally deal with the finite element method we need to re-look at the differential equation. We have already seen that many systems come from conservation laws which deal with integrals over volumes of quantities. We have also seen that in the finite volume method, we reversed this by looking at an integral of the equation over a small 'volume'. Now we generalise this further by saying that instead of requiring $\mathcal{L}u - f = 0$, for some set of functions $\mathcal{A}_0 = \{w \in \mathcal{A} | w(0) = w(1) = 0\}$ we require the inner product of the equation with all functions in this space to be zero:

$$(\mathcal{L}u - f, w) = 0, \ \ \forall w \in \mathcal{A}_0. \quad (1.63)$$

This is usually called a weak form of the equation. One reason for doing this is that we can try to use integration by parts to reduce the highest derivative of u which occurs. Another reason is that many conservation laws have solutions which give an extremum of some functional and these very natuarlly fall into this framework.

In the case of our model problem, a solution of the differential equation needs to be twice differentiable. If we can integrate by parts we need only consider functions which have integrable derivatives (so the derivative does not need to be continuous). There are special symbols for this space of functions and some of its subspaces.

$$H^1[0,1] = \{\text{functions } u \text{ on } [0,1] \text{ with } \int_0^1 ({u'}^2 + u^2) \mathrm{d}x \text{ bounded}, \quad (1.64)$$

We will usually just write H^1 and leave the domain ambiguous.

$$H_E^1 = \{u \in H^1 | \ u(0) = a, \ u(1) = b\}, \quad (1.65)$$

$$H_0^1 = \{u \in H^1 | \ u(0) = 0, \ u(1) = 0\}. \quad (1.66)$$

The weak form of our equation is

$$(-u'' + \kappa^2 u - f, w) = 0, \ \forall w \in H_0^1, \tag{1.67}$$

and if we integrate by parts

$$(u', w') + \kappa^2(u, w) - (f, w) = 0. \tag{1.68}$$

The variational form of the equation is: find $u \in H_E^1$ such that

$$J[u] = \int_0^1 [\frac{1}{2}({u'}^2 + \kappa^2 u^2) - fu]\mathrm{d}x, \tag{1.69}$$

is an extremum. We shall not take this further here.

If we define a new inner product:

$$a(u, v) = (u', v') + \kappa^2(u, v), \tag{1.70}$$

this can be used to define a new norm, sometimes called an energy norm,

$$||u||_a = \sqrt{a(u, u)}, \tag{1.71}$$

and the weak form of the equation is

$$a(u, w) = (f, w) \ \ \forall w \in H_0^1. \tag{1.72}$$

This has now established the framework within which we can apply finite elements. Instead of looking for the solution of a differential equation, we are seeking a function $u \in H_E^1$ (as it is in H_E^1 it automatically satisfies the boundary conditions) which satisfies the weak form, an integral equation, for all functions in H_0^1.

We now wish to approximate the function u throughout the whole interval, $[0, 1]$, on which it is defined. It is logical to look at functions which are already in H_E^1, for instance we might choose functions which are piecewise linear. In the background, but not needed to be specified at this point, is that there is a mesh (or set of elements) which cover the interval and they have some characteristic size, say h. For instance the simplest set of elements would come from dividing the interval by the set of points $x_j = jh, \ j = 0, 1, \ldots, m$ where $h = 1/m$ so that the elements are the intervals $[jh, (j+1)h], \ j = 0, \ldots, m-1$. Suppose we have chosen a set of approximating functions in H^1 and we let

$$S_E^h = \{U \in H^1 \mid U(0) = a, \ U(1) = b, \ \text{U a linear combination of approximating functions}\}, \tag{1.73}$$

$$S_0^h = \{W \in H^1 \mid W(0) = 0, \ W(1) = 0, \ \text{W a linear combination of approximating functions}\}. \tag{1.74}$$

We see that $S_E^h \subset H_E^1$, and $S_0^h \subset H_0^1$.

Now $U \in S_E^h$ will be a best approximation to $u \in H_E^1$ measured in the a-norm if the error is orthogonal (orthogonality measured using the a-inner product) to all functions in S_0^h,

$$a(u-U, W) = 0,\ \forall W \in S_0^h. \tag{1.75}$$

Now $a(.,.)$ is a linear operator and we know that the weak form which u satisfies gives that as $S_0^h \subset H_0^1$ $a(u, W)$ then $a(u, W) = (f, W)$, so that the orthogonality condition is

$$a(U, W) = (f, W)\ \forall W \in S_0^h. \tag{1.76}$$

This is the essence of the finite element method: if we can find U then it is a best approximation to u measured in a particular norm (in this case the a-norm).

To make this useful we have to carry on a bit to understand how we can calculate U. We suppose that we use a set of linearly independent **basis** functions $\phi_0, \ldots, \phi_m$ (linear independence measured using the $a(.,.)$ inner product) such that

$$S_E^h = \{U \in H_E^1 \mid U = \sum_{j=0}^{m} U_j \text{ with } U(0) = a,\ U(1) = b)\}, \tag{1.77}$$

$$S_0^h = \{W \in H_0^1 \mid W = \sum_{j=0}^{m} W_j \phi_j \text{ with } U(0) = U(1) = 0\}, \tag{1.78}$$

then since S_0^h is a linear subspace the condition which determines U can also be written

$$a(U, \phi_r) = (f, \phi_r)\ r = 1, \ldots, m-1, \tag{1.79}$$

and if we substitute the form of U in terms of the basis funcitons,

$$U = \sum_{j=0}^{m} U_j \phi_j, \tag{1.80}$$

then

$$\sum_{j=0}^{m} U_j a(\phi_j, \phi_r) = (f, \phi_r),\ r = 1, \ldots, m-1. \tag{1.81}$$

This is just a matrix equation for the vector $\mathbf{U}^{\mathrm{T}} = (U_0, U_1, \ldots, U_m)$ (it is incomplete, it needs the addition of the two boundary conditions to produce a full matrix). This is still a very general framework so we now focus on one example implementation of this finite element framework.

Suppose the functions ϕ_r are piecewise linear functions,

$$\phi_r(x) = \begin{cases} 0 & x \leq (r-1)h, \\ \frac{x-(r-1)h}{h} & (r-1)h < x \leq rh, \\ \frac{(r+1)h-x}{h} & rh < x \leq (r+1)h, \\ 0 & x > (r+1)h. \end{cases} \tag{1.82}$$

We next need to evaluate the numbers $a(\phi_r, \phi_s) = (\phi_r' \phi_s') + \kappa^2(\phi_r, \phi_s)$.

(a) $A_{r,s} = (\phi_r', \phi_s')$

If $s < r-1$ or $s > r+1$ then $A_{r,s} = 0$, if $s = r-1$ or $s = r+1$, $A_{r,s} = -1/h$, if $s = r$, $A_{s,r} = 2/h$.

(b) $B_{r,s} = (\phi_r, \phi_s)$

If $s < r-1$ or $s > r+1$ then $B_{r,s} = 0$, if $s = r-1$ or $s = r+1$, $B_{r,s} = h/6$, if $s = r$, $A_{s,r} = 2h/3$.

if we define

$$\tilde{f}_j = \frac{1}{h}\int_{(j-1)h}^{(j+1)h} \phi_j(x) f(x) \mathrm{d}x, \tag{1.83}$$

then we must have

$$-\frac{1}{h}U_{r+1} + \frac{2}{h}U_r - \frac{1}{h}U_{r-1} + \frac{h}{6}U_{r+1} + \frac{2h}{3}U_r + \frac{h}{6}U_{r-1} = h\tilde{f}_r, \;\; r = 1, \ldots, m-1. \tag{1.84}$$

if we define a 'mass' matrix $\mathbf{M}$ by

$$\mathbf{M} = \frac{1}{6}\begin{pmatrix} 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 \\ . & . & . & . & \cdots & . \\ 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}, \tag{1.85}$$

and use the same matrix $\mathbf{K}$ as occured in finite difference and finite volume formulations, then in our model problem the vector $\mathbf{U}$ is determined by

$$(\frac{1}{h^2}\mathbf{K} + \kappa^2\mathbf{M})\mathbf{U} = \tilde{\mathbf{f}}. \tag{1.86}$$

1.1.3.1 *Note 1 Error Analysis* We are not able to go into error analysis of finite element methods in great detail. There are very readable sections in Eriksson et al which you are advised to read through. One key idea is as follows: since we have a best approximation (measured in the energy norm) the error in that norm must be bounded by the error which occurs for any other approximating function, so we choose an interpolating approximation since Lagrange's theorem tells us the error in interpolating functions. Mathematically, since

$$||u - U||_a \leq ||u - V||_a, \ \forall V \in S_E^h, \tag{1.87}$$

using the same piecewise linear functions ϕ_r, choose

$$V = \sum_{r=0}^{m} u_r \phi_r(x), \tag{1.88}$$

and Lagrange's theorem gives

$$\text{on } (x_i < x \leq x_{i+1}) \ : \ u - V = \frac{1}{2}(x - x_i)(x_{i+1} - x)u''(\xi_i), \tag{1.89}$$

where $x_i < \xi_i < x_{i+1}$ so that

$$|u - V| \leq \frac{1}{8}(x_{i+1} - x_i)^2 ||u''||_\infty, \tag{1.90}$$

and similarly

$$|u' - V'| \leq (x_{i+1} - x_i)||u''||_\infty, \tag{1.91}$$

so we have a *a priori* error estimate

$$||u - U||_a^2 \leq [h^2 + \frac{1}{64}\kappa^2 h^4]||u''||_\infty. \tag{1.92}$$

This may appear a little weak, since it indicates an error in the energy norm which is only first order in h, but with more analysis better error estimates can be found.

1.1.3.2 *Note 2 A posteriori error analysis and mesh adaptation* Once we have an approximate solution $U(x)$, one sensible question is 'how well does U satisfy the original differential equation?' Of course this needs to be carefully thought out because there will be a set of points (the boundaries of elements) where U cannot be differentiated sufficiently to be substituted into the equation. As this is ony a finite set of points, we can still calculate **integrals** across the domain.

We start by defining a residual function

$$R(U) = f - \mathcal{L}U, \tag{1.93}$$

on each interval (x_i, x_{i+1}) and a mesh **function** $h(x)$ by

$$h(x) = x_{i+1} - x_i, \quad x_i \leq x < x_{i+1}, \tag{1.94}$$

so with a uniform mesh, h is just a constant but with a variable mesh, gives the local mesh size.

Intuitively, a measure of how well the approximation represents the solution is given by an integral of the residual across the domain, so if the residual is relatively large we should think of changing the mesh to reduce the residual, if the residual is relatively small, the mesh could be expanded. We cannot go into the mathematics too deeply, but it is possible to derive results for our model problem of the form

$$||u' - U'||_2 \leq C||h(x)R(U)||_2, \tag{1.95}$$

for some positive constant C. This means we can produce an **algorithm** to adapt the mesh to a particular problem and to achieve a given tolerance for some measure of the error. In this case we can seek to make $M = ||hR(U)||_2^2$ less than some tolerance by firstly moving mesh points, and then increasing the number of mesh points.

As an example, consider the case $k = 0$ and approximation using piecewise linear functions, when the residual is just $R(U) = f$, and we take $f = 1,\ 0 \leq x < \frac{2}{3}$, $f = 100, \frac{2}{3} \leq x \leq 1$. If we take four points uniformly across the interval, then $M = 1111$. If however we move the mesh so the four points are at $x = 0, \frac{2}{3}, \frac{5}{6}, 1$, then $M = 93$ and if we add one more point, so the points are at $x = 0, \frac{2}{3}, \frac{7}{9}, \frac{8}{9}, 1$ then $M = 14$. If five points were to be uniformly distributed across the interval $[0, 1]$ them $M = 208$. Since such large gains can be made using an algorithmic approach, mesh adaptation is becoming more important in the numerical solution of PDE's.

1.2 Stability

1.2.1 *Introduction*

We have seen that time dependent problems are approximated by a vector of values, $\mathbf{U}^n$ at each time step and further the form of the iteration for these values was

$$\mathbf{B}_1\mathbf{U}^{n+1} = \mathbf{B}_0\mathbf{U}^n + \mathbf{f}^n, \tag{1.96}$$

where $\mathbf{B}_0, \mathbf{B}_1$ are matrices. If $\mathbf{B}_1 = \mathbf{I}$ we called the method explicit, otherwise we referred to an implicit method. If the PDE involved two or more space dimensions, it can still be dealt with under this framework, elements of vector $\mathbf{U}$ have just a more complicated mapping onto the physical mesh. Thus in principle we have iterations of the form

$$\mathbf{U}^{n+1} = \mathbf{B}_1^{-1}\mathbf{B}_0\mathbf{U}^n + \mathbf{B}_1^{-1}\mathbf{f}^n, \tag{1.97}$$

to calculate the approximation time step by time step. Thus we need first to understand some things about iterated sequences.

Suppose we have the sequence

$$2y_{n+2} - 5y_{n+1} + 2y_n = 0, \tag{1.98}$$

with $y_0 = a,\ y_1 = b$. We look for a solution $y_n \sim \lambda^n$ so that $2\lambda^2 - 5\lambda + 2 = 0,$. The roots are $\lambda = 2, \frac{1}{2}$ and the general solution is $y_n = A2^{-n} + B2^n$. Using the initial conditions

$$y_n = \frac{4a - 2b}{3}2^{-n} + \frac{2b - a}{3}2^n. \tag{1.99}$$

Now suppose we choose the inital values a, b such that $2b - a = 0$, then $y_n = a2^{-n} \to 0, \text{ as } n \to \infty,$ and it might appear that we have a sensible computation to determine y_n. However suppose, because of finite machine accuracy, that $y_0 = a \pm \epsilon,\ y_1 = b \pm \epsilon,$ then

$$y_n = (a \pm \frac{2}{3}\epsilon)2^{-n} \pm \epsilon 2^n, \tag{1.100}$$

and for example, if $\epsilon = 10^{-6}$ (typical single precison accuracy) then $\epsilon 2^n \sim 1$ after only $n \sim 20$ iterations! Hence we could not carry out any practical computations using this iteration without the calulations being swamped by the growing solution even though the 'solution' should have no component of this mode.

Consider now a matrix example,

$$\mathbf{U}^{n+1} = \mathbf{A}\mathbf{U}^n, \tag{1.101}$$

where

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix}. \tag{1.102}$$

We look for a solution where $U^n \sim \lambda \mathbf{z}$ so that $\mathbf{A}\mathbf{z} = \lambda \mathbf{z}$, and λ is an eigenvalue of the iteration matrix $\mathbf{A}$. In this case the eigenvalues are $\lambda_1 = (2 - \sqrt{5})/2 < 1$, $\lambda_2 = (3 + \sqrt{5})/2 > 1$, and if we denote the eigenvectors by $\mathbf{z}_1, \mathbf{z}_2$, the general solution is

$$\mathbf{U}^n = A\lambda_1^n \mathbf{z}_1 + B\lambda_2^n \mathbf{z}_2, \tag{1.103}$$

where A and B are constants. Again, if we choose the inital data so that B is zero, machine rounding errors will still produce a growing component which will swamp the calculation very quickly.

Thus one difficulty with calculating iterated sequences is that growing modes lead to rapid breakdown of a calculation and this 'instability' may not be a part of the original physical system, but introduced as an artefact of the discretisation of the continuous system. Often stability is a more challenging practical problem than accuracy.

Definition 1.1 *If λ_i are the eigenvalues of a matrix G then $\rho(G) = \max_i |\lambda_i|$, is called the spectral radius of $\mathbf{G}$.*

Lemma 1.2 *If $\mathbf{U}^{n+1} = \mathbf{G}\mathbf{U}^n$ then the iteration is stable provided $\rho(G) \le 1$.*

1.2.2 *Fourier Analysis of Stability*

It can be very difficult to calculate all of the eigenvalues of the iteration matrix. One approach which has been very successful in provinding practical stability criteria is to extend the domain of the PDE to the whole real line (or the whole plane ...) and then to look at stability of Fourier modes. On the real line, if U_j^n is assumed to behave like

$$U_j^n \sim [\lambda(k)]^n \mathrm{e}^{ikjh}, \tag{1.104}$$

(a discrete analogue of $u(x,t) \sim \mathrm{e}^{\mu t}\mathrm{e}^{ikx}$) then we can examine $\max_k |\lambda(k)|$ to decide on stability of the numerical scheme. It is important to realise that although practical problems do not have infinite domains, they do not have constant coefficients; nevertheless criteria which have come from Fourier analysis of model problems are usually very good predictors for the behaviour of a numerical scheme.

Example 1: Heat Equation

If we have

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \tag{1.105}$$

and we discretise explicitly,

$$U_j^{n+1} = U_j^n + \mu(U_{j+1}^n - 2U_j^n + U_{j-1}^n), \tag{1.106}$$

where $\mu = D\Delta t/h^2$, then substituting the Fourier mode gives

$$\lambda = 1 - 4\mu \sin^2 \frac{1}{2}kh, \tag{1.107}$$

so that λ is real and $-1 \leq \lambda \leq 1$ provided $0 \leq \mu \leq \frac{1}{2}$. This is a very stringent criterion, equivalent to $\Delta t \leq \frac{h^2}{2D}$. If the mesh is refined then the time step has to be reduced by a factor equal to the square of the refinement.

Example 2: Lax–Wendroff applied to a hyperbolic equation

One important method to generate a finite difference scheme is Lax-Wendroff whereby the time derivatives in a Taylor expansion of $u(.,t+\Delta t)$ about t are replaced by space derivatives using the differential equation, and those space derivatives discretised using finite differences. We consider the model equation

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{1.108}$$

where a is constant. Since

$$u(x,t+\Delta t) = u(x,t) + \Delta t\frac{\partial u}{\partial t}(x,t) + \frac{1}{2}\Delta t^2\frac{\partial^2 u}{\partial t^2}(x,t) + \cdots, \tag{1.109}$$

and using $u_t = -au_x$, $u_{tt} = a^2 u_{xx}$,

$$u(x, t+\Delta t) = u(x,t) - a\Delta t \frac{\partial u}{\partial x}(x,t) + \frac{1}{2}a^2 \Delta t^2 \frac{\partial^2 u}{\partial x^2}(x,t) + \cdots . \tag{1.110}$$

This series is truncated after the third term to give a second order accurate scheme,

$$U_j^{n+1} = U_j^n - \nu(U_{j+1}^n - U_{j-1}^n) + \frac{1}{2}\nu^2(U_{j+1}^n - 2U_j^n + U_{j-1}^n), \tag{1.111}$$

where $\nu = a\Delta t/h$. Substituting a Fourier mode gives,

$$\lambda = 1 - i\nu \sin kh + \frac{1}{2}\nu^2(2\cos kh - 2), \tag{1.112}$$

so that λ is complex. If $p = \sin \frac{1}{2}kh$,

$$|\lambda|^2 = \lambda\lambda^* = 1 - 4\nu^2(1-\nu^2)p^2, \tag{1.113}$$

and we will have $|\lambda| \leq 1$ provided $-1 \leq \nu \leq 1$.

Example 3: θ-method for Diffusion Equation

If we apply a θ-method to a diffusion equation, then

$$U_j^{n+1} - \mu\theta(U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}) = U_j^n + \mu(1-\theta)(U_{j+1}^n - 2U_j^n + U_{j-1}^n), \tag{1.114}$$

where again $\mu = D\Delta t/h^2$. Substituting in a Fourier mode,

$$\lambda = \frac{1 - 4\mu(1-\theta)p^2}{1 + 4\mu\theta p^2} \tag{1.115}$$

with $p = \sin \frac{1}{2}kh$. We deduce that $|\lambda| \leq 1$ provided

$$\theta \geq \frac{1}{2} - \frac{1}{4\mu}. \tag{1.116}$$

1.2.3 *Practical Stability*

We have assumed that the largest growth rate modulus must be bounded by 1 for stability; it is possible to conclude that there will be stability in the sense that solutions remain bounded, provided the largest eigenvalue of $\mathbf{B}_1^{-1}\mathbf{B}_0$ satisfies

$$|\lambda| \leq 1 + C\Delta t, \text{ as } \Delta t \to 0, \tag{1.117}$$

for some constant $C > 0$, since then

$$|\lambda^n| \leq (1 + C\Delta t)^n \leq \mathrm{e}^{Cn\Delta t}, \tag{1.118}$$

and so

$$|\lambda^n| \leq \mathrm{e}^{Ct} \text{ as } n\Delta t \to t \text{ with } \Delta t \to 0,\ n \to \infty. \tag{1.119}$$

This is called Lax-Richtmyer stability.

However for practical purposes, if the modulus of λ exceeds 1 for finite Δt then the calculations invariably become too large to handle. Hence it is safer to require **practical or strict** stability, $|\lambda| \leq 1$.

Example: Convection Diffusion

Consider an explicit discretisation of a convection-diffusion equation,

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = D\frac{\partial^2 u}{\partial x^2}, \tag{1.120}$$

$$U_j^{n+1} = U_j^n - \frac{1}{2}\nu(U_{j+1}^n - U_{j-1}^n) + \mu(U_{j+1}^n - 2U_j^n + U_{j-1}^n), \tag{1.121}$$

where $\nu = a\Delta t/h$, $\mu = D\Delta t/h^2$. Using a Fourier mode

$$\lambda = 1 - i\nu \sin kh - 4\mu \sin\frac{1}{2}kh, \tag{1.122}$$

so if $p = \sin^2 \frac{1}{2}kh$,

$$|\lambda(p)|^2 = (1 - 4\mu p)^2 + 4\nu^2 p(1-p). \tag{1.123}$$

(a) Lax-Richtmyer Stability

If we write $\nu^2 = \frac{a^2\mu}{D}\Delta t$, then

$$|\lambda|^2 = (1 - 4\mu p)^2 + \frac{4a^2\mu p(1-p)}{D}\Delta t, \tag{1.124}$$

and provided $0 \leq \mu \leq \frac{1}{2}$, we will have

$$|\lambda|^2 \leq 1 + \frac{a^2\mu}{D}\Delta t, \tag{1.125}$$

so that the scheme will be L-R stable for $\Delta t \to 0$ with $\mu = D\Delta t/h^2$ fixed. However in practice this is not sufficiently strong, for instance with $\mu = 0.25$, $\nu = 0.75$, $|\lambda|^2 = 1 + \frac{17}{4}p - \frac{13}{4}p^2$ so that growth factors of greater than 1 occur for a range of small values of p leading to breakdown of a calculation.

(b) Practical Stability

If we write

$$|\lambda|^2 = 1 - 4(2\mu - \nu^2)p + 4(4\mu^2 - \nu^2)p^2, \tag{1.126}$$

then $|\lambda(0)|^2 = 1$ and $|\lambda(1)|^2 = (1 - 4\mu) \leq 1$ provided $0 \leq \mu \leq \frac{1}{2}$. Since we are just dealing with a quadratic all we need have is the slope (with respect to p) to negative at $p = 0$ in order for $|\lambda|$ to be less than 1 over the range $0 \leq p \leq 1$. Hence we also need $2\mu - \nu^2 \geq 0$. This can be rearranged as

$$\frac{a^{\Delta}t^2}{2h^2} \leq \frac{D\Delta t}{h^2} \leq \frac{1}{2}, \tag{1.127}$$

or

$$\frac{a\Delta t}{h} \leq \min(\frac{2}{Pe}, \frac{Pe}{2}), \tag{1.128}$$

where Pe is the mesh Peclet number, $Pe = ah/D$.

2

ITERATIVE METHODS FOR SOLVING LINEAR ALGEBRAIC SYSTEMS

2.1 Introduction

We have seen that discretisation of PDE's leads naturally to a matrix equation for a vector of unknowns which might represent nodal values of a function, local averages of a function or coefficients in an expanion in terms of a set of basis functions which generate an approximation space. In most real problems the dimension of the coefficient matrix is so large that Gauss elimination is not a feasible option, so that an interative method must be used to approximate the solution vector. We take as our model problem

$$\mathbf{A}\mathbf{x} = \mathbf{f}, \tag{2.1}$$

where $\mathbf{A}$ is a 'non-singular $m \times m$ matrix. For any $m \times m$ matrix $\mathbf{B}$ we can write

$$\mathbf{B}\mathbf{x} + (\mathbf{A} - \mathbf{B})\mathbf{x} = \mathbf{f}, \tag{2.2}$$

so that

$$\mathbf{x} = (\mathbf{I} - \mathbf{B}^{-1}\mathbf{A})\mathbf{x} + \mathbf{B}^{-1}\mathbf{f}, \tag{2.3}$$

and we can seek an iterative solution by the algorithm $\mathbf{x}^{(0)} = \mathbf{0}$,

$$\mathbf{x}^{(k+1)} = (\mathbf{I} - \mathbf{B}^{-1}\mathbf{A})\mathbf{x}^{(k)} + \mathbf{B}^{-1}\mathbf{f}. \tag{2.4}$$

We will use the notation

$$\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F} = \mathbf{D}(\mathbf{I} - \mathbf{L} - \mathbf{U}), \tag{2.5}$$

where $\mathbf{E}$, $\mathbf{L}$ are strict lower triangular matrices and $\mathbf{F}$, $\mathbf{U}$ are strict upper triangular matrices and $\mathbf{D}$ is a diagonal matrix.

2.1.1 *Jacobi iteration*

Let $\mathbf{B} = \mathbf{D}$ and

$$\mathbf{x}^{(k+1)} = \mathbf{J}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{f}, \tag{2.6}$$

where

$$\mathbf{J} = \mathbf{L} + \mathbf{U} \quad \text{Jacobi Iteration Matrix} \tag{2.7}$$

2.1.2 *Gauss-Seidel iteration*

If the vector $\mathbf{x} = \{x_1, \ldots, x_m\}$ we used Jacobi type iteration then when we come to evaluate $x_s^{(k+1)}$ we have already evaluated $x_1^{(k+1)}, \ldots, x_{s-1}^{(k+1)}$, so we could evaluate the s^{th} row of $\mathbf{L}\mathbf{x}^{(k+1)}$. In matrix motation

$$\mathbf{x}^{(k+1)} = \mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{f}, \tag{2.8}$$

$$\mathbf{x}^{(k+1)} = (\mathbf{I} - \mathbf{L})^{-1}(\mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{f}), \tag{2.9}$$

so that the iteration matrix is

$$\mathbf{H} = (\mathbf{I} - \mathbf{L})^{-1}\mathbf{U}, \quad \text{Gauss} - \text{Seidel iteration matrix.} \tag{2.10}$$

2.1.3 *Relaxation*

Suppose a scalar s is determined by $s = G(s)$ and we iterated $x^{(k+1)} = G(x^{(k)})$, then this could be written $x^{(k+1)} = x^{(k)} + (G(x^{(k)}) - x^{(k)})$, or $x^{(k+1)} = x^{(k)} + \delta$ where $\delta = G(x^{(k)}) - x^{(k)}$. In many situations, over a number of iterations, δ is consistently either too large or too small. So in some situations, it is possible to accelerate convergence by taking a multiple of δ so that

$$x^{(k+1)} = x^{(k)} + \omega\delta = (1 - \omega)x^{(k)} + \omega G(x^{(k)}). \tag{2.11}$$

This is the basis of relaxation; if $\omega < 1$ we speak of under-relaxation, if $\omega > 1$ it is called over-relaxation. If we apply this idea to Jacobi we would have

$$\mathbf{x}^{(k+1)} = (1 - \omega)\mathbf{x}^{(k)} + \omega\mathbf{J}\mathbf{x}^{(k)} + \omega\mathbf{D}^{-1}\mathbf{f}, \tag{2.12}$$

so the iteration matrix would be

$$\mathcal{J}(\omega) = (1 - \omega)\mathbf{I} + \omega\mathbf{J} = \mathbf{I} - \omega\mathbf{D}^{-1}\mathbf{A}. \tag{2.13}$$

It is more common to combine relaxation with a form of Gauss–Seidel. If we use

$$\mathbf{x}^{(k+1)} = \mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{f}, \tag{2.14}$$

to generate an increment δ as in the scalar case, we will obtain

$$\mathbf{x}^{(k+1)} = (1 - \omega)\mathbf{x}^{(k)} + \omega(\mathbf{L}\mathbf{x}^{(k+1)} + \mathbf{U}\mathbf{x}^{(k)} + \mathbf{D}^{-1}\mathbf{f} - \mathbf{x}^{(k)}), \tag{2.15}$$

so that

$$\mathbf{x}^{(k+1)} = (I - \omega\mathbf{L})^{-1}[(1 - \omega)\mathbf{I} + \omega\mathbf{U}]\mathbf{x}^{(k)}, \tag{2.16}$$

and the iteration matrix is

$$\mathcal{H}(\omega) = (I - \omega\mathbf{L})^{-1}[(1 - \omega)\mathbf{I} + \omega\mathbf{U}]. \tag{2.17}$$

2.1.4 *Some General Theory*

The reason we have taken so much effort to determine the iteration matrix is that if

$$\mathbf{x}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} + \mathbf{b}, \tag{2.18}$$

and as the true solution $\mathbf{s}$ satisfies

$$\mathbf{s} = \mathbf{G}\mathbf{s} + \mathbf{b}, \tag{2.19}$$

so defining an error $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{s}$, we must have

$$\mathbf{e}^{(k+1)} = \mathbf{G}\mathbf{x}^{(k)} = \mathbf{G}^{k+1}\mathbf{e}^{(0)}. \tag{2.20}$$

Thus the iteration will converge provided $\mathbf{G}^k$ tends to the zero matrix as k becomes large.

Theorem If $\mathbf{G}$ has eigenvalues λ_r, the iteration converges as $k \to \infty$ provided $\rho(\mathbf{G}) = \max_r |\lambda_r| < 1$. We call ρ the spectral radius of the matrix $\mathbf{G}$.

We consider only a special case, where there is a full set of eigenvalues, and if $\mathbf{G} = \mathbf{S}\Lambda\mathbf{S}^{-1}$ where Λ is a diagonal matrix of the eigenvalues, then $\mathbf{G}^n = \mathbf{S}\Lambda^n\mathbf{S}^{-1}$ and this will become the zero vector provided the moduli of all the eignevalues are less than one.

Theorem (Gershgorin) The eigenvalues of $\mathbf{G} = \{g_{rs}\}$ lie within the union of the discs

$$D_r \;:\; |\lambda - g_{rr}| \le \sum_{s \ne r} |g_{rs}|. \tag{2.21}$$

This can be seen by supposing eigenvalue λ has eigenvector $\mathbf{z}$ so that

$$\mathbf{G}\mathbf{z} = \lambda\mathbf{z} \;\; => \;\; (\lambda - g_{rr})z_r = \sum_{s \ne r} g_{rs}z_s, \;\; r = 1, \ldots, m. \tag{2.22}$$

Choose r such that $|z_r| = ||\mathbf{z}||_\infty$, and then

$$|\lambda - g_{rr}|.||\mathbf{z}||_\infty \le \sum_{s \ne r} |g_{rs}|.|z_s| \le \sum_{s \ne r} |g_{rs}|.||\mathbf{z}||_\infty, \tag{2.23}$$

whence the result follows.

Lemma Jacobi iteration will converge if $\mathbf{A}$ is strictly diagonally dominant,

$$|a_{rr}| > \sum_{s \ne r} |a_r s|, \quad \forall r = 1, \ldots, m. \tag{2.24}$$

In this case, using Jacobi iteration

$$J_{rs} = \begin{cases} 0, & r = s, \\ \dfrac{a_{rs}}{a_{rr}}, & s \ne r, \end{cases} \tag{2.25}$$

so that for all eigenvalues of the iteration matrix,

$$|\lambda| \leq \sum_{s \neq r} |J_{rs}| = \sum_{s \neq r} \frac{|a_{rs}|}{|a_{rr}|} < 1, \tag{2.26}$$

so that all the eigenvalues of the iteration matrix must have moduli less than one.

Lemma Relaxation can converge only if $0 < \omega < 2$.

To show this, consider the eigenvalues of the iteration matrix, $\mathcal{H}(\omega)$,

$$[(1-\omega)\mathbf{I} + \omega\mathbf{U}]\mathbf{z} = \lambda(\mathbf{I} - \omega\mathbf{L})\mathbf{z}, \tag{2.27}$$

so that

$$[(1-\omega-\lambda)\mathbf{I} + \omega\mathbf{U} + \lambda\omega\mathbf{L}]\mathbf{z} = 0. \tag{2.28}$$

If we take the determinant of the coefficient matrix and look at the term which is independent of λ (and so is the product of all the eigenvalues), it is $(1-\omega)^m$, so that if $|1-\omega| \geq 1$ then at least one eigenvalue must be greater than or equal to 1. Hence it is necessary that $|1-\omega| < 1$, or $0 < \omega < 2$.

We do not have time to go into the mathematics deeply, but there will be a critical value ω_0 for which $\rho(\mathcal{H}(\omega))$ is a minimum. For most practical matrices this optimum value of the relaxation parameter can be found from numerical experiments.

2.1.5 *Application to a Poisson Equation*

Consider now the discretisation of

$$\nabla^2 w = f, \quad (x, y) \in [0,1] \times [0,1] \tag{2.29}$$

with condition $w = 0$ on the boundary. If we use central differences with mesh size $h = 1/m$ in both x and y directions,

$$W_{r,s} - \frac{1}{4}(W_{r-1,s} + W_{r,s-1} + W_{r,s+1} + W_{r+1,s}) = h^2 f_{r,s}, \quad r, s = 1, \ldots, m-1. \tag{2.30}$$

and boundary conditions $W_{0,s} = W_{m,s} = W_{r,0} = W_{r,m} = 0$, $s = 1, \ldots, m-1$ and $r = 1, \ldots, m-1$. For an appropriately defined vector $\mathbf{x}$ we have

$$\mathbf{Lx} = \frac{1}{4}(x_{r-1,s} + x_{r,s-1}), \tag{2.31}$$

$$\mathbf{Ux} = \frac{1}{4}(x_{r+1,s} + x_{r,s+1}). \tag{2.32}$$

Suppose that μ is an eigenvalue of the Jacobi iteration matrix for this problem, then

$$(\mathbf{L}+\mathbf{U})\mathbf{x} = \mu\mathbf{x}. \tag{2.33}$$

Consider now a vector $\mathbf{d}$ whose elements are defined by $d_{r,s} = \alpha^{r+s}x_{r,s}$, then the $\mathcal{M}(r,s)$ element of $(\alpha\mathbf{L}+\alpha^{-1}\mathbf{U})\mathbf{d}$ is

$$\text{element } \mathcal{M}(r,s): \quad (\alpha\mathbf{L}+\alpha^{-1}\mathbf{U})\mathbf{d} = \frac{1}{4}\alpha^{r+s}(x_{r-1,s}+x_{r,s-1}+x_{r,s+1}+x_{r+1,s}), \tag{2.34}$$

and the RHS is $\mu\alpha^{r+s}\mathbf{x}_{r,s} = \mu d_{r,s}$ so that μ is also an eigenvalue of the matrix $\alpha\mathbf{L}+\alpha^{-1}\mathbf{U}$ for any α.

Lemma 2.1

$$\rho(\mathbf{H}) = \rho(\mathbf{J})^2 \tag{2.35}$$

If we define the **asymptotic convergence rate** as $\log\rho$ the Lemma implies that Gauss-Seidel converges twice as fast as Jacobi. To show this result in this case (it is a more general result valid for a wider class of matrices) we use the fact that $\det(\mathbf{I}-\mathbf{L}) = 1$ so that

$$\det(\lambda\mathbf{I}-\mathbf{H}) = \det[(\mathbf{I}-\mathbf{L})(\lambda\mathbf{I}-\mathbf{H})] = \det[\lambda\mathbf{I}-\lambda\mathbf{L}-\mathbf{U}], \tag{2.36}$$

since $\mathbf{H} = (\mathbf{I}-\mathbf{L})^{-1}\mathbf{U}$, so that

$$\det(\lambda\mathbf{I}-\mathbf{H}) = \det[\sqrt{\lambda}(\sqrt{\lambda}\mathbf{I}-\sqrt{\lambda}\mathbf{L}-\frac{1}{\sqrt{\lambda}}\mathbf{U})] = 0,, \tag{2.37}$$

and using our result above, $\sqrt{\lambda}$ must be an eigenvalue of $\mathbf{L}+\mathbf{U} = \mathbf{J}$

In addition to this result we can find the eigenvalues and eigenvectors of $\mathbf{J}$ explicitly. If we let

$$\mathbf{x}_{r,s}^{(p,q)} = \sin\pi prh \sin\pi qsh, \quad p,q = 1,\ldots,m-1, \tag{2.38}$$

then

$$\mathbf{J}\mathbf{x}^{(p,q)} = \frac{1}{2}(\cos\pi ph + \cos\pi qh)\mathbf{x}^{(p,q)}, \tag{2.39}$$

and the eigenvalue associated with eigenvector $\mathbf{x}^{(p,q)}$ is

$$\lambda_{(p,q)} = \frac{1}{2}(\cos\pi ph + \cos\pi qh), \tag{2.40}$$

so that the largest eigenvalue occurs when $p = q = 1$ or $p = q = m-1$ and

$$\rho(\mathbf{J}) = \cos\pi h, \tag{2.41}$$

$$\rho(\mathbf{H}) = \cos^2\pi h. \tag{2.42}$$

2.2 Conjugate Gradient Iteration

2.2.1 *Minimisation Problem & Steepest Descents*

Suppose $\mathbf{A}$ is a real symmetric positive definite matrix (so that $\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} > 0$ for all non-zero $\mathbf{x}$) and consider the function

$$\phi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{b}, \tag{2.43}$$

then

$$\nabla\phi = \mathbf{A}\mathbf{x} - \mathbf{b}, \tag{2.44}$$

so that ϕ will be a minimum when $\mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{0}$. So to solve the matrix equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ we can try to find a minimum of the function ϕ. We can use this idea to generate an iterative method to determine an approximate solution of the original matrix equation.

If we have an approximation $\mathbf{x}_0$ then the residual,

$$\mathbf{r}_o = \mathbf{b} - A\mathbf{x}_0 = -\nabla\phi(\mathbf{x}_0), \tag{2.45}$$

will give the direction of fastest descent from the curve $\phi(\mathbf{x}) = \phi(\mathbf{x}_0)$ at the point $\mathbf{x}_0$. So we can find a value $\alpha = \alpha_1$ for which $\phi(\mathbf{x}_0 + \alpha\mathbf{r}_0)$ is smallest. If we solve $\frac{\mathrm{d}\phi}{\mathrm{d}\alpha} = 0$, we obtain

$$\alpha_1 = \frac{\mathbf{r}_0^{\mathrm{T}}\mathbf{r}_0}{\mathbf{r}_0^{\mathrm{T}}\mathbf{A}\mathbf{r}_0}, \tag{2.46}$$

so that an iteration algorithm is:

Steepest descents

Choose an initial guess $\mathbf{x}_0$ (e.g. $\mathbf{x}_0 = \mathbf{b}$), tolerance ϵ and calculate $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$,

loop for $s = 1, 2, \ldots$

$$\alpha_s = \frac{\mathbf{r}_{s-1}^{\mathrm{T}}\mathbf{r}_{s-1}}{\mathbf{r}_{s-1}^{\mathrm{T}}\mathbf{A}\mathbf{r}_{s-1}},$$
$$\mathbf{x}_s = \mathbf{x}_{s-1} + \alpha_s\mathbf{r}_{s-1},$$
$$\mathbf{r}_s = \mathbf{b} - \mathbf{A}\mathbf{x}_s,$$

until $||\mathbf{r}_s||_2 < \epsilon$.

This algorithm can converge slowly because it may be repetitively calculating the same few search directions with only small changes of components in other directions.

Convergence

If the true solution is $\mathbf{s}$ and the error is $\mathbf{e}_r = \mathbf{s} - \mathbf{x}_s$, the iteration formula gives

$$\mathbf{e}_s = \mathbf{e}_{s-1} + \alpha_s\mathbf{A}\mathbf{e}_{s-1}. \tag{2.47}$$

If the eigenvalues and eigenvectors of $\mathbf{A}$ are λ_j, $\mathbf{z}_j$ and the error is written in terms of $\mathbf{e}_{s-1} = \sum_j \xi_j \mathbf{x}_j$ (for some numbers ξ_j, leaving out the $s-1$ index), algebra gives

$$\alpha_s = \frac{\sum \lambda_j^2 \xi_j^2}{\sum \lambda_j^3 \xi_j^2}, \tag{2.48}$$

and we can find the error at the next step. Using a norm $||\mathbf{e}||_A = \mathbf{e}^T \mathbf{A}\mathbf{e}$,

$$||\mathbf{e}_s||_A = \beta^2 ||\mathbf{e}_{s-1}||_A, \tag{2.49}$$

where

$$\beta^2 = 1 - \frac{(\sum \lambda_j^2 \xi_j^2)^2}{(\sum \lambda_j^3 \xi_j^2)(\sum \lambda_j \xi_j^2)}. \tag{2.50}$$

This formula is not much use without knowing the values of ξ_j but we can ask what is the largest value it can have over all possible combinations of ξ_j. The method to answer this question is beyond this course, but if the condition number of the matrix $\mathbf{A}$ is κ then

$$\beta \leq \frac{\kappa - 1}{\kappa + 1}. \tag{2.51}$$

So convergence becomes slower as the condition number increases but is good if κ is less than around 10.

2.2.2 *Conjugate Gradient Algorithm*

We will call two vectors $\mathbf{x}$ and $\mathbf{y}$ A-conjugate if $\mathbf{x}^T \mathbf{A}\mathbf{y} = 0$. We can also see that if we denote the characteristic polynomial of $\mathbf{A}$ as

$$p(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I}), \tag{2.52}$$

then Cayley-Hamilton's theorem tells us that $p(\mathbf{A}) = 0$ too. If we suppose $p(\lambda) = \sum_j p_j \lambda^j$, then

$$p_n \mathbf{A}^n + p_{n-1} \mathbf{A}^{n-1} + \cdots + p_0 \mathbf{I} = 0, \tag{2.53}$$

and so, as the matrix is non-singular ($p_0 \neq 0$),

$$\mathbf{A}^{-1} = -\frac{1}{p_0}[p_1 \mathbf{I} + p_2 \mathbf{A} + \cdots + p_n \mathbf{A}^{n-1}] \tag{2.54}$$

and the solution $\mathbf{x}$ must be

$$\mathbf{x} = \sum_{j=0}^{n-1} x_j \mathbf{A}^j \mathbf{b}, \tag{2.55}$$

where $x_j = -p_{j+1}/p_0$. Thus the solution is able to be expressed as a linear combination of vectors of the form $\mathbf{A}^j \mathbf{b}$, $j = 0, \ldots, n-1$. Let $K_r = \operatorname{span}\{\mathbf{b}, \ldots, \mathbf{A}^r \mathbf{b}\}$

then one approximation strategy is to try to find approximations from the subspaces $K_0, K_1, \cdots$ which successively minimise ϕ. After n steps we must have the solution but of course we hope that when n is very large, we might obtain a good approximation after only a small number of steps. The obvious way to do this would be at step r, approximating from K_r to try to find coefficients q_j such that if $\mathbf{x}_r = \sum_{j=0}^{j=r} q_j \mathbf{A}^j \mathbf{b}$ then $\phi(\mathbf{x}_r) \equiv \phi(q_0, q_1, \ldots, q_r)$ is minimised. In this case

$$\phi(q_0, \ldots, q_r) = \frac{1}{2} \sum_{j=0}^{j=r} \sum_{k=0}^{k=r} q_j q_k \mathbf{b}^T \mathbf{A}^{j+k+1} \mathbf{b} - \sum_{j=0}^{j=r} q_j \mathbf{b}^T \mathbf{A}^j \mathbf{b}, \tag{2.56}$$

and so setting the derivatives of ϕ with respect to $q_0, \cdots, q_r$ to be zero would mean solving another matrix system of growing size at each iteration. The idea of conjugate gradient is to find coefficients only once in an expansion of the solution in terms of a growing set of basis vectors. To do this we need a different basis for K_r. Suppose we call this basis $\{\mathbf{d}_0, \ldots, \mathbf{d}_r\}$ and that the basis vectors are A-conjugate. Now if we let $\mathbf{x}_r = \sum_{j=0}^{j=r} q_j \mathbf{d}_j$ then

$$\phi(q_0, \ldots, q_r) = \sum_{j=0}^{j=r} [\frac{1}{2} q_j^2 \mathbf{d}_j^T \mathbf{A} \mathbf{d}_j - q_j \mathbf{d}_j^T \mathbf{b}], \tag{2.57}$$

so now the minimisation problem is almost trivial, each component q_j is independent of the others **and** at each step, the existing components do not change. So we only calculate the components in each direction **once**. Of course we do have to construct the basis $\{\mathbf{d}_0, \ldots, \mathbf{d}_r\}$ but since it has to span K_r and at the r^{th} step we already have an A-conjugate basis for K_{r-1}, all we have to do is compute one additional basis vector at each step.

Theorem 2.2 *If $\mathbf{x}_{r-1}$ minimises ϕ over all vectors in K_{r-1} and gives residual $\mathbf{r}_{r-1} = \mathbf{b} - A\mathbf{x}_{r-1}$, and if for a vector $\mathbf{d}_r$ we have $\mathbf{d}_r^T \mathbf{A} \mathbf{y} = 0$, for all $\mathbf{y} \in K_{r-1}$, then*

$$\mathbf{x}_r = \mathbf{x}_{r-1} + \alpha_r \mathbf{d}_r, \tag{2.58}$$

minimises ϕ over all vectors in K_r when

$$\alpha_r = \frac{\mathbf{d}_r^T \mathbf{r}_{k-1}}{\mathbf{d}_r^T \mathbf{A} \mathbf{d}_r}. \tag{2.59}$$

The proof is straight forward. It is almost trivial from our discussion above that

$$\alpha_r = \frac{\mathbf{d}_r^T \mathbf{b}}{\mathbf{d}_r^T \mathbf{A} \mathbf{d}_r}. \tag{2.60}$$

Now $\mathbf{r}_{r-1} = \mathbf{b} - \mathbf{A}\mathbf{x}_{r-1}$, so that $\mathbf{b} = \mathbf{r}_{r-1} + \mathbf{A}\mathbf{x}_{r-1}$ and as $\mathbf{d}_r$ is A-conjugate with all vectors in K_{r-1}, including $\mathbf{x}_{r-1}$, we must have ${\mathbf{d}_r}^T \mathbf{b} = \mathbf{d}_r^T \mathbf{r}_{r-1}$.

This enables us to carry out one iteration of the **Conjugate Gradient algorithm**.

Suppose $\mathbf{x}_r \in K_{r-1} = span\{\mathbf{b}, \mathbf{Ab}, \ldots, \mathbf{A}^{r-1}\mathbf{b}\} \equiv span\{\mathbf{d}_0, \ldots, \mathbf{d}_{r-1}\}$, minimises ϕ over K_{r-1} and the basis vectors $\mathbf{d}_0, \ldots, \mathbf{d}_{r-1}$ are A-conjugate. Calculate $\mathbf{r}_{r-1} = \mathbf{b} - \mathbf{Ax}_{r-1} \in K_r$ and suppose

$$\mathbf{d}_r = \mathbf{r}_{r-1} + \beta_r \mathbf{d}_{r-1}, \tag{2.61}$$

where β_r is chosen to make $\mathbf{d}_r$ A-conjugate to $\mathbf{d}_{r-1}$,

$$\beta_r = -\frac{\mathbf{r}_{r-1}^T \mathbf{A}\mathbf{d}_{r-1}}{\mathbf{d}_{r-1}^T \mathbf{A}\mathbf{d}_{r-1}}. \tag{2.62}$$

If

$$\alpha_r = \frac{\mathbf{d}_r^T \mathbf{r}_{r-1}}{\mathbf{d}_r^T \mathbf{A}\mathbf{d}_{r-1}}, \tag{2.63}$$

is a search length, then

$$\mathbf{x}_r = \mathbf{x}_{r-1} + \alpha_r \mathbf{d}_r, \tag{2.64}$$

minimises ϕ over K_r and

$$\mathbf{r}_{r-1}^T \mathbf{d}_s = \mathbf{r}_{r-1}^T \mathbf{r}_s = \mathbf{d}_r^T \mathbf{A}\mathbf{d}_s = 0, \quad s = 0, \ldots, r-1. \tag{2.65}$$

Note that we can also show the

$$\alpha_r = \frac{\mathbf{d}_r^T \mathbf{r}_{r-1}}{\mathbf{d}_r^T \mathbf{A}\mathbf{d}_r} = \frac{\mathbf{r}_{r-1}^T \mathbf{r}_{r-1}}{\mathbf{d}_r^T \mathbf{A}\mathbf{d}_r}, \tag{2.66}$$

and

$$\beta_r = -\frac{\mathbf{r}_{r-1}^T \mathbf{A}\mathbf{d}_{r-1}}{\mathbf{d}_{r-1}^T \mathbf{A}\mathbf{d}_{r-1}} = -\frac{\mathbf{r}_{r-1}^T \mathbf{r}_{r-1}}{\mathbf{d}_{r-1}^T \mathbf{A}\mathbf{d}_{r-1}}. \tag{2.67}$$

It is worth observing that we only ever need to store four vectors, $\mathbf{x}_{r-1}$, $\mathbf{d}_{r-1}$, $\mathbf{r}_{r-1}$, $\mathbf{Ad}_{r-1}$ each of which can be replaced as the iteration proceeds and other than the vector, $\mathbf{Ad}_{r-1}$ there are only inner products to evaluate. The algorthm is very compact,

Algorithm

$$
\begin{aligned}
&\mathbf{x} = 0\\
&\mathbf{r} = \mathbf{b}\\
&\mathbf{d} = \mathbf{r}\\
&\delta_0 = \mathbf{r}^T\mathbf{r}\\
&\delta_1 = \delta_0\\
&\text{while}(\delta_1 > \text{tolerance})\\
&\qquad \mathbf{q} = \mathbf{A}\mathbf{d}\\
&\qquad \alpha = \delta_1/\mathbf{d}^T\mathbf{q}\\
&\qquad \mathbf{x} = \mathbf{x} + \alpha\mathbf{d}\\
&\qquad \mathbf{r} = \mathbf{r} - \alpha\mathbf{q}\\
&\qquad \delta_0 = \delta_1\\
&\qquad \delta_1 = \mathbf{r}^T\mathbf{r}\\
&\qquad \beta = -\delta_1/\delta_0\\
&\qquad \mathbf{d} = \mathbf{r} + \beta\mathbf{d}\\
&\text{end while}
\end{aligned}
\tag{2.68}
$$

We do not have time to develop an error analysis but we can note that if the condition number of the matrix $\mathbf{A}$ is κ, then the error changes by a factor smaller or equal to

$$\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}, \tag{2.69}$$

each iteration. This is considerable better than for steepest descents. Even further advantage can be gained by preconditioning the matrix $\mathbf{A}$ by pre-multiplying the equation $\mathbf{A}\mathbf{x} = \mathbf{b}$ by an additional matrix. We shall not consider that aspect in this course.

2.3 Multigrid Iteration

Central to a great deal of fluid mechanics is the Poisson equation $\nabla^2 w = f$ and in two or more dimensions it is common to obtain approximate solutions by iterative methods. If we just return to a 1-D model problem,

$$w'' = f, \tag{2.70}$$

then for Jacobi iteration

$$W_r^{(k+1)} = \frac{1}{2}(W_{r+1}^{(k)} + W_{r-1}^{(k)}) - \frac{1}{2}h^2 f_r, \tag{2.71}$$

so that the iteration matrix $\mathbf{J}$ satisfies

$$(\mathbf{J}\mathbf{W})_r = \frac{1}{2}(W_{r+1} + W_{r-1}), \tag{2.72}$$

and if $W_r = \sin \pi r k h,\ k = 1, \ldots, m-1$, then we see that the eigenvaues of $\mathbf{J}$ are just $\cos \pi k h$ and the spectral radius is $\cos \pi h \sim 1 - \pi^2 h^2/2$ so that as the mesh becomes finer, not only are there more calculations per iteration, the convergence rate becomes poorer. Since the poor convergence is associated mainly with low frequency errors one way which has been developed for accelerating convergence is to carry out some iterations on a coarser mesh where the convergence rate is better and then to transfer the solution back to the fine mesh.

Suppose two vectors are defined, $\mathbf{W}_h$ which approximates the values $(\mathbf{W}_h)_r \sim W(rh),\ r = 0, \ldots, m,$ and $\mathbf{W}_{2h}$ with $(\mathbf{W}_{2h})_s \sim w(2sh),\ s = 0, \ldots, m/2$. Obviously assume m is even. Next define two matrix operators, one which **restricts** values on the fine mesh to the coarse mesh, symbolically

$$\mathbf{W}_{2h} = \mathbf{R}_h^{2h}\mathbf{W}_h, \tag{2.73}$$

and an interpolation operator which takes values from the coarse mesh to the fine mesh, again symbolically

$$\mathbf{W}_h = \mathbf{I}_{2h}^h\mathbf{W}_{2h}. \tag{2.74}$$

Multigrid then carries out a number of iterations on the fine mesh, damping out high frequency errors, transfers values to the coarse mesh and carries out further iterations on that mesh to damp out low frequency errors before interpolating values back to the original fine mesh. The overall sequence of iteration on fine mesh, restriction, iteration on the coarse mesh, interpolation and then further iteration on the fine mesh represents one cycle of multigrid iteration. In practice a number of meshes are used, not just two. Symbolically in the case of two meshes, if we carry out p Jacobi iterations on the fine mesh and q iterations on the coarse mesh,

$$\mathbf{W}_h^{(k+1)} = \mathbf{I}_{2h}^h(\mathbf{J}_{2h})^q\mathbf{R}_h^{2h}(\mathbf{J}_h)^p\mathbf{W}_h^{(k)}. \tag{2.75}$$

A key result is that the spectral radius of the iteration matrix can be **independent** of h. In practice the finest mesh is chosen with $m = 2^k$ and then it is possible to use up to k meshes, each half the resolution of the previous mesh and a multigrid cycle is a sequence of (either Jacobi or Gauss–Seidel) iterations on each of the meshes in succession, both after restricting from fine to coarse, and after interpolating from coarse to fine. It is also usual to work with residuals, so that if $\mathbf{W}_h$ is our current approximation and $\mathbf{W}_h + \delta\mathbf{W}_h$ is a better approximation, then as

$$L_h[\mathbf{W}_h + \delta\mathbf{W}_h] = \mathbf{f}, \tag{2.76}$$

we have

$$L_h[\delta\mathbf{W}_h] = \mathbf{f} - L_h\mathbf{W}_h = \mathbf{r}_h, \tag{2.77}$$

where $\mathbf{r}_h$ is the residual so that we iterate on the coarse mesh to solve

$$\mathbf{L}_{2h}[\delta\mathbf{W}_{2h}] = \mathbf{R}_h^{2h}\mathbf{r}_h, \tag{2.78}$$

and then

$$\mathbf{W}_h \leftarrow \mathbf{W}_h + \mathbf{I}_{2h}^h[\delta\mathbf{W}_{2h}]. \tag{2.79}$$

ACKNOWEDGEMENT

These notes have been extracted from lecture notes I used to teach a second year course on Numerical Computation for Engineering and Computer Science students. I am very grateful to Karina Terekhova for commenting on a version of the lecture notes.