Lectures for VII Mexican School on Gravitation and Mathematical Physics
Relativistic Astrophysics and Numerical Relativity
Numerical Analysis for Numerical Relativists

Matthew W. Choptuik
Dept. of Physics & Astronomy
University of British Columbia
choptuik@physics.ubc.ca

November 2006

Contents

1 Basic Finite Difference Techniques for Time Dependent PDEs

There are several good reference texts for the material in Section 1. Among my personal favorites are [1]-[4].

1.1 Preliminaries

We can divide time-dependent PDEs into two broad classes:

1. **Initial-value Problems (Cauchy Problems)**, spatial domain has no boundaries (either infinite or "closed"—e.g. "periodic boundary conditions"
2. **Initial-Boundary-Value Problems**, spatial domain *finite*, need to specify boundary conditions

Note: Even if *physical* problem is really of Type 1, finite computational resources $\longrightarrow$ finite spatial domain $\longrightarrow$ approximate as Type 2; will hereafter loosely refer to either type as an IVP.

Working Definition: **Initial Value Problem**

- State of physical system arbitrarily (usually) specified at some initial time $t = t_0$.
- Solution exists for $t \geq t_0$; uniquely determined by equations of motion (EOM) and boundary conditions (BCs).

Issues in Finite Difference (FD) Approximation of IVPs

- Discretization (Derivation of FDA's)
- Solution of algebraic systems resulting from discretization
- Consistency
- Accuracy
- Stability
- Converegence
- Dispersion / Dissipation
- Treatment of Non-linearities
- Computational cost—expect $O(N)$ work ($N \equiv$ number of "grid points" (discrete events at which approximate solution is computed)

1.2 Types of IVPs (by example)

In the following three examples, u is always a function of one space and one time variable, i.e. $u \equiv u(x,t)$. Such a problem is often referred to as "1-d" by numericists, the time dimension being implicit in this nomenclature. I will also use the subscript notation for partial differentiation, e.g. $u_t \equiv \partial_t u$.

1.2.1 Wave and "Wave-Like" ("Hyperbolic"): The 1-d Wave Equation

$$\begin{aligned} u_{tt} &= c^2 u_{xx} \qquad c \in \mathbf{R}, \\ u(x,0) &= u_0(x) \\ u_t(x,0) &= v_0(x) \end{aligned} \tag{1}$$

1.2.2 Diffusion ("Parabolic"): The 1-d Diffusion Equation

$$\begin{aligned} u_t &= \sigma u_{xx} \qquad \sigma \in \mathbf{R}, \quad \sigma > 0. \\ u(x,0) &= u_0(x) \end{aligned} \tag{2}$$

1.2.3 Schrödinger: The 1-d Schrödinger Equation

$$\begin{aligned} i\psi_t &= -\frac{\hbar}{2m}\psi_{xx} + V(x,t)\psi \qquad \psi \in \mathbf{C} \\ \psi(x,0) &= \psi_0(x) \end{aligned} \tag{3}$$

Note: Although $\psi(x,t)$ is *complex* in this case, we can rewrite (3) as a *system* of 2 coupled scalar, real-valued equations.

1.3 Some Basic Concepts, Definitions and Techniques

We will be considering the finite-difference approximation (FDA) of PDEs, and as such, will generally be interested in the continuum limit, where the *mesh spacing*, or *grid spacing*, usually denoted h, tends to 0. Because any specific calculation must necessarily be performed at some specific, *finite* value of h, we will also be (extremely!) interested in the way that our discrete solution varies as a function of h. In fact, we will *always* view h as the basic "control" parameter of a typical FDA. Fundamentally, for sensibly constructed FDAs, we expect the error in the approximation to go to 0, as h goes to 0.

Let

$$Lu = f \tag{4}$$

denote a general *differential* system. For simplicity and concreteness, you can think of $u = u(x,t)$ as a single function of one space variable and time, but the discussion in this section applies to cases in more independent variables ($u(x,y,t)$, $u(x,y,z,t) \cdots$ etc.), as well as multiple *dependent* variables ($u = \mathbf{u} = [u_1, u_2, \cdots, u_n]$). In (4), L is some differential operator (such as $\partial_{tt} - \partial_{xx}$) in our wave equation example), u is the unknown, and f is some specified function (frequently called a *source* function) of the independent variables.

Here and in section 1.9 it will be convenient to adopt a notation where a superscript h on a symbol indicates that it is discrete, or associated with the FDA, rather than the continuum. (Note, however, that for simplicity of presentation, we will *not* adopt this notation in much of the development below). With this notation, we will generically denote an FDA of (4) by

$$L^h u^h = f^h \tag{5}$$

where u^h is the discrete solution, f^h is the specified function evaluated on the finite-difference mesh, and L^h is the finite-difference approximation of L.

1.3.1 Residual

Note that another way of writing our FDA is

$$L^h u^h - f^h = 0 \tag{6}$$

It is often useful to view FDAs in this form for the following reason. First, we have a canonical view of what it means to solve the FDA—"drive the left-hand side to 0". Furthermore, for iterative approaches to the solution of the FDA (which are common, since it may be too expensive to solve the algebraic equations directly), we are naturally lead to the concept of a *residual*. The residual is simply the level of "non-satisfaction" of our FDA (and, indeed, of any algebraic expression). Specifically, if $\tilde{u}^h$ is some approximation to the true solution of the FDA, u^h, then the residual, r^h, associated with $\tilde{u}^h$ is just

$$r^h \equiv L^h \tilde{u}^h - f^h \tag{7}$$

This leads to the view of a convergent, iterative process as being one which "drives the residual to 0".

1.3.2 Truncation Error

The *truncation error*, τ^h, of an FDA is defined by

$$\tau^h \equiv L^h u - f^h \tag{8}$$

where u satisfies the continuum PDE (4). We note that the *form* of the truncation error can always be computed (typically using Taylor series) from the finite difference approximation and the differential equations.

1.3.3 Convergence

Assuming that our FDA is characterized by a *single* discretization scale, h, we say that the approximation *converges* iff

$$u^h \to u \quad \text{as} \quad h \to 0. \tag{9}$$

Operationally (i.e. in practice), convergence is clearly our chief concern as numerical analysts, particularly if there is reason to suspect that the solutions of our PDEs are good models for real phenomena. We note that this is believed to be the case for many interesting problems in general relativistic astrophysics—the two black hole problem being an excellent example.

1.3.4 Consistency

Assuming that the FDA with truncation error τ^h is characterized by a single discretization scale, h, we say that the FDA is *consistent* if

$$\tau^h \to 0 \quad \text{as} \quad h \to 0. \tag{10}$$

Consistency is obviously a necessary condition for convergence.

1.3.5 Order of an FDA

Assuming that the FDA is characterized by a single discretization scale, h, we say that the FDA is *p-th order accurate* or simply *p-th order* if

$$\lim_{h\to 0} \tau^h = O(h^p) \qquad \text{for some integer } p \tag{11}$$

1.3.6 Solution Error

The solution error, e^h, associated with an FDA is defined by

$$e^h \equiv u - u^h \tag{12}$$

1.3.7 Relation Between Truncation Error and Solution Error

Is is common to tacitly assume that

$$\tau^h = O(h^p) \qquad \longrightarrow \qquad e^h = O(h^p)$$

This assumption is often warranted, but it is extremely instructive to consider *why* it is warranted and to investigate (following Richardson 1910 (!) [5]) in some detail the *nature* of the solution error. We will return to this issue in more detail in section 1.9.

1.3.8 Deriving Finite Difference Formulae

The essence of finite-difference approximation of a PDE is the replacement of the continuum by a discrete lattice of grid points, and the replacement of derivatives/differential operators by finite-difference expressions. These finite-difference expressions (finite-difference quotients) approximate the derivatives of functions at grid points, using the grid values themselves. All of the operators and expressions we need can easily be worked out using Taylor series techniques. For example, let us consider the task of approximating the first derivative $u_x(x)$ of a function $u(x)$, given a discrete set of values $u_j \equiv u(jh)$ as shown in Figure 1. As it turns out,

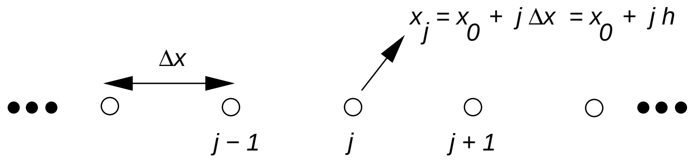

Figure 1: A one-dimensional, uniform finite difference mesh. Note that the spacing, $\triangle x = h$, between adjacent mesh points is *constant*. In the text we tacitly assume that the origin, x_0, of our coordinate system is $x_0 = 0$.

given the three values $u(x_j - h), u(x_j)$ and $u(x_j + h)$, which we will denote u_{j-1}, u_j, and u_{j+1} respectively, we can compute an $O(h^2)$ approximation to $u_x(x_j) \equiv (u_x)_j$ as follows. Taylor expanding, we have

$$\begin{aligned}
u_{j-1} &= u_j - h(u_x)_j + \frac{1}{2}h^2(u_{xx})_j - \frac{1}{6}h^3(u_{xxx})_j + \frac{1}{24}h^4(u_{xxxx})_j + O(h^5) \\
u_j &= u_j \\
u_{j+1} &= u_j + h(u_x)_j + \frac{1}{2}h^2(u_{xx})_j + \frac{1}{6}h^3(u_{xxx})_j + \frac{1}{24}h^4(u_{xxxx})_j + O(h^5)
\end{aligned}$$

We now seek a linear combination of u_{j-1}, u_j, and u_{j+1} which yields $(u_x)_j$ to $O(h^2)$ accuracy, i.e. we seek c_-, c_0 and c_+ such that

$$c_-\, u_{j-1} + c_0\, u_j + c_+\, u_{j+1} = (u_x)_j + O(h^2)$$

This results in a system of three linear equations for u_{j-1}, u_j, and u_{j+1}:

$$\begin{aligned}
c_- \,+\, c_0 \,+\, c_+ &= 0 \\
-hc_- \,+\, hc_+ &= 1 \\
\frac{1}{2}h^2c_- \,+\, \frac{1}{2}h^2c_+ &= 0
\end{aligned}$$

which has the solution

$$\begin{aligned}
c_- &= -\frac{1}{2h} \\
c_0 &= 0 \\
c_+ &= +\frac{1}{2h}
\end{aligned}$$

Thus our $O(h^2)$ finite difference approximation for the first derivative is

$$\frac{u(x+h) - u(x-h)}{2h} = u_x(x) + O(h^2) \tag{13}$$

Note that it may not be obvious to you *a priori*, that the truncation error of this approximation *is* $O(h^2)$, since a naive consideration of the number of terms in the Taylor series expansion which can be eliminated using 2 values (namely $u(x+h)$ and $u(x-h)$) suggests that the error might be $O(h)$. The fact that the $O(h)$ term "drops out" is a consequence of the *symmetry*, or *centering* of the stencil, and is a common theme in such FDAs (which, naturally enough, are called centred difference approximations).

Using the same technique, we can easily generate the $O(h^2)$ expression for the *second* derivative, which uses the same difference stencil as the above approximation for the first derivative.

$$\frac{u(x+h) - 2u(x) + u(x-h)}{h^2} = u_{xx}(x) + O(h^2) \tag{14}$$

Exercise: Compute the precise form of the $O(h^2)$ terms in expressions (13) and (14).

1.4 Sample Discretizations / FDAs

1.4.1 1-d Wave equation with fixed (Dirichlet) boundary conditions

$$\begin{aligned} u_{tt} &= u_{xx} \qquad (c=1) \qquad 0 \le x \le 1; \quad t \ge 0 \\ u(x,0) &= u_0(x) \\ u_t(x,0) &= v_0(x) \\ u(0,t) &= u(1,t) = 0 \end{aligned} \tag{15}$$

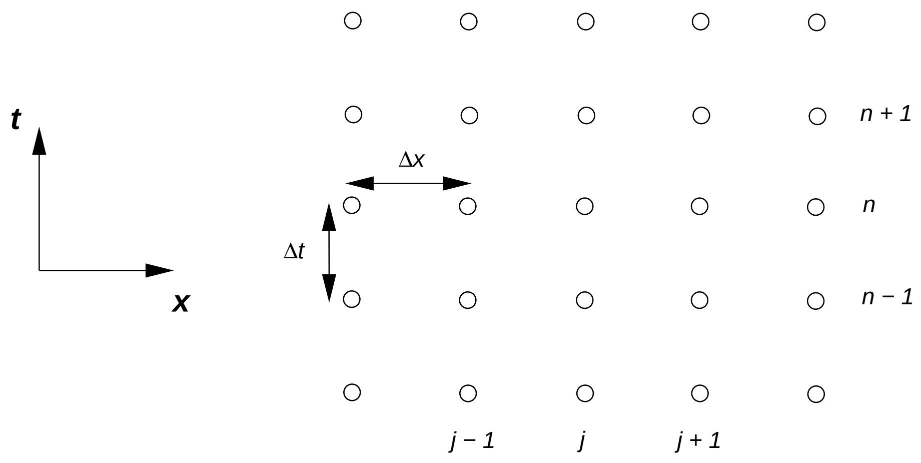

Figure 2: Portion of uniform finite-difference mesh (grid) for 1-d time-dependent problem. Note that the spacings in both the spatial *and* temporal directions are constant

We now introduce a discrete domain (uniform grid) (x_j, t^n)—part of which is shown in Figure 2.

$$\begin{aligned} t^n &\equiv n\,\triangle t\,, \qquad n = 0, 1, 2, \cdots \\ x_j &\equiv (j-1)\,\triangle x\,, \qquad j = 1, 2, \cdots J \\ u^n_j &\equiv u(n\,\triangle t\,, (j-1)\,\triangle x\,) \\ \triangle x &= (J-1)^{-1} \\ \triangle t &= \lambda\,\triangle x \qquad \lambda \equiv \text{"Courant number"} \end{aligned}$$

Note: When solving wave equations using FDAs, we will typically keep λ constant when we vary $\triangle x$. Thus, our FDA will always be characterized by a *single* discretization scale, h.

$$\triangle x \equiv h$$
$$\triangle t \equiv \lambda h$$

(Also note the `Fortran`-style indexing of the spatial grid index ($j = 1, 2, \cdots$) and the `C`-style indexing of the temporal one ($n = 0, 1, \cdots$). This is a particular convention which I, as a predominantly `Fortran` programmer, find convenient.)

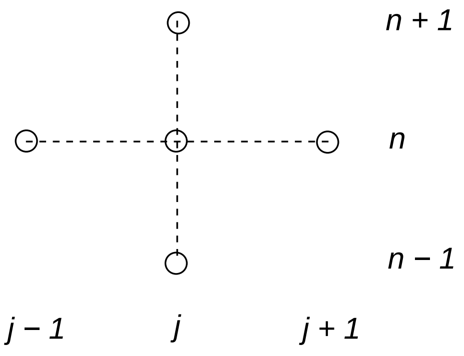

Figure 3: Stencil (molecule/star) for "standard" $O(h^2)$ approximation of (15).

FDA: "standard $O(h^2)$"

Discretized Interior equation:

$$\begin{aligned}
(\triangle t)^{-2}\left(u_j^{n+1} - 2u_j^n + u_j^{n-1}\right) &= (u_{tt})_j^n + \frac{1}{12}\triangle t^2 (u_{tttt})_j^n + O(\triangle t^4) \\
&= (u_{tt})_j^n + O(h^2) \\
(\triangle x)^{-2}\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right) &= (u_{xx})_j^n + \frac{1}{12}\triangle x^2 (u_{xxxx})_j^n + O(\triangle x^4) \\
&= (u_{xx})_j^n + O(h^2)
\end{aligned}$$

Putting these two together, we get the $O(h^2)$ approximation

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\triangle t^2} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\triangle x^2} \qquad j = 2, 3, \cdots, J-1 \tag{16}$$

Note that a scheme such as (16) is often called a *three level scheme* since it couples *three "time levels"* of data (i.e. unknowns at three distinct, disrete times t^{n-1}, t^n, t^{n+1}.

Discretized Boundary conditions:

$$u_1^{n+1} = u_J^{n+1} = 0$$

Discretized Initial conditions:
We need to specify *two* "time levels" of data (effectively $u(x,0)$ and $u_t(x,0)$), i.e. we must specify

$$\begin{aligned}
u_j^0 \quad &, \quad j = 1, 2, \cdots, J \\
u_j^1 \quad &, \quad j = 1, 2, \cdots, J
\end{aligned}$$

ensuring that the initial values are compatible with the boundary conditions.

Note that we can solve (16) *explicitly* for u_j^{n+1}:

$$u_j^{n+1} = 2u_j^n - u_j^{n-1} + \lambda^2 \left(u_{j+1}^n - 2u_j^n + u_j^{n-1} \right) \tag{17}$$

Also note that (17) is actually a *linear system* for the unknowns u_j^{n+1}, $j = 1, 2, \cdots, J$; in combination with the discrete boundary conditions we can write

$$\mathbf{A}\ \mathbf{u}^{n+1} = \mathbf{b} \tag{18}$$

where $\mathbf{A}$ is a *diagonal* $J \times J$ matrix and $\mathbf{u}^{n+1}$ and $\mathbf{b}$ are vectors of length J. Such a difference scheme for an IVP is called an *explicit* scheme.

1.4.2 1-d Diffusion equation with Dirichlet boundary conditions

$$\begin{aligned} u_t &= u_{xx} \qquad (\sigma = 1) \qquad 0 \le x \le 1; \quad t \ge 0 \\ u(x,0) &= u_0(x) \\ u(0,t) &= u_(1,t) = 0 \end{aligned} \tag{19}$$

We will use same discrete domain (grid) as for the 1-d wave equation.

FDA: Crank-Nicholson
This scheme illustrates a useful "rule of thumb": *Keep the difference scheme "centred"*

- centred in time, centred in space
- minimizes truncation error for given h
- tends to minimize instabilities

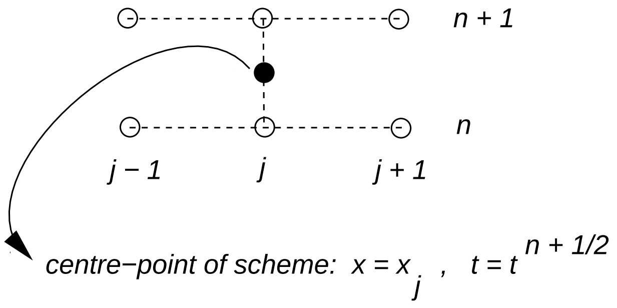

Figure 4: Stencil (molecule/star) for $O(h^2)$ Crank-Nicholson approximation of (19).

Discretization of time derivative:

$$\begin{aligned} \triangle t^{-1} \left(u_j^{n+1} - u_j^n \right) &= (u_t)_j^{n+\frac{1}{2}} + \frac{1}{24} \triangle t^2 (u_{ttt})_j^{n+\frac{1}{2}} + O(\triangle t^4) \\ &= (u_t)_j^{n+\frac{1}{2}} + O(\triangle t^2) \end{aligned} \tag{20}$$

$O(h^2)$ *second-derivative operator:*

$$D_{xx}\, u_j^n \equiv \triangle x^{-2} \left(u_{j+1}^n - 2u_j^n + u_{j+1}^n \right) \tag{21}$$

$$D_{xx} = \partial_{xx} + \frac{1}{12} \triangle x^2\, \partial_{xxxx} + O(\triangle x^4) \tag{22}$$

(Forward) Time-averaging operator, μ_t:

$$\mu_t u_j^n \equiv \frac{1}{2}\left(u_j^{n+1} + u_j^{n-1}\right) = u_j^{n+\frac{1}{2}} + \frac{1}{8}\triangle t^2 (u_{tt})_j^{n+\frac{1}{2}} + O(\triangle t^4) \tag{23}$$

$$\mu_t = \left[I + \frac{1}{8}\triangle t^2 \partial_{tt} + O(\triangle t^4)\right]_{t=t^{n+1/2}} \tag{24}$$

where I is the identity operator. Assuming that $\triangle t = O(\triangle x) = O(h)$, is is easy to show (*exercise*) that

$$\mu_t \left[D_{xx} u_j^n\right] = (u_{xx})_j^{n+\frac{1}{2}} + O(h^2)$$

Putting the above results together, we are lead to the ($O(h^2)$) Crank-Nicholson approximation of (19):

$$\frac{u_j^{n+1} - u_j^n}{\triangle t} = \mu_t \left[D_{xx} u_j^n\right] \tag{25}$$

Written out in full, this is

$$\frac{u_j^{n+1} - u_j^n}{\triangle t} = \frac{1}{2}\left[\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\triangle x^2} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\triangle x^2}\right] \qquad j = 2, 3, \cdots, J-1 \tag{26}$$

We can rewrite (26) in the form

$$a_+ u_{j+1}^{n+1} + a_0 u_j^{n+1} + a_- u_{j-1}^{n+1} = b_j \qquad j = 2, 3, \cdots, J-1 \tag{27}$$

where

$$\begin{aligned}
a_+ &\equiv -\frac{1}{2}\triangle x^{-2} \\
a_0 &\equiv \triangle t^{-1} + \triangle x^{-2} \\
a_- &\equiv -\frac{1}{2}\triangle x^{-2} \\
b_j &\equiv \left(\triangle t^{-1} - \triangle x^{-2}\right) u_j^n + \frac{1}{2}\triangle x^{-2}\left(u_{j+1}^n + u_{j-1}^n\right)
\end{aligned}$$

which, along with the BCs ($u_1^{n+1} = u_J^{n+1} = 0$), is again a linear system of the form

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

for the "unknown vector" $\mathbf{u}^{n+1}$. This time, however, the matrix $\mathbf{A}$, is *not* diagonal, and the scheme is called *implicit*—i.e. the scheme *couples* unknowns at the *advanced* time level, $t = t^{n+1}$.

Note that $\mathbf{A}$ is a *tridiagonal* matrix: all elements A_{ij} for which $j \neq i+1, i$ or $i-1$ vanish. The solution of tridiagonal systems can be performed very efficiently using special purpose routines (such as `DGTSV` in `LAPACK` [6]): specifically, the operation count for solution of (26) is $O(J)$.

Also note that we can immediately write down the analogous scheme for the Schrödinger equation (3):

$$i\,\frac{\psi_j^{n+1} - \psi_j^n}{\triangle t} = -\frac{\hbar}{2m}\mu_t\left[D_{xx}\psi_j^n\right] + V(x_j)\,\mu_t \psi_j^n \tag{28}$$

In this case we get a *complex* tridiagonal system, which can also be solved in $O(J)$ time, using, for example, the `LAPACK` routine `ZGTSV`.

1.5 The 1-D Wave Equation in More Detail

Recall our "standard" $O(h^2)$ discretization:

$$\begin{aligned} u_j^{n+1} &= 2u_j^n - u_j^{n-1} + \lambda^2 \left(u_{j+1}^n - 2u_j^n + u_{j-1}^n \right), \qquad j = 2, 3, \cdots, J-1 \\ u_1^{n+1} &= u_J^{n+1} = 0 \end{aligned}$$

As we have discussed, to initialize the scheme, we need to specify u_j^0 and u_j^1, which is equivalent (in the limit $h \to 0$) to specifying $u(x,0)$ and $u_t(x,0)$.

Before proceeding to a discussion of a "proper initialization", let us briefly digress and consider the continuum case, and, for the sake of presentation, assume that we are considering a true IVP on an unbounded domain; i.e. we wish to solve

$$u_{tt} = u_{xx} \qquad -\infty < x < \infty \quad , \quad t \geq 0 \tag{29}$$

As is well known, the general solution of (29) is the superposition of an arbitrary *left-moving* profile ($v = -c = -1$), and an arbitrary *right-moving* profile ($v = +c = +1$); i.e.

$$u(x,t) = \ell(x+t) + r(x-t) \tag{30}$$

where (see Figure 5)

ℓ	:	constant along "left-directed" characteristics
r	:	constant along "right-directed" characteristics

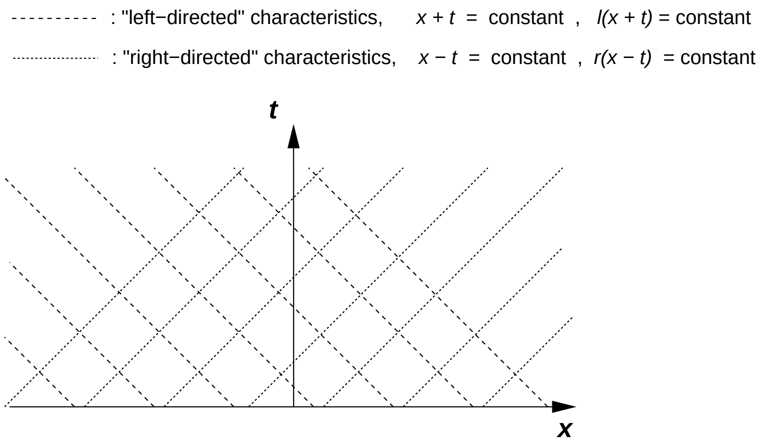
: "left−directed" characteristics, x + t = constant , l(x + t) = constant
: "right−directed" characteristics, x − t = constant , r(x − t) = constant
t
x

Figure 5: Characteristics of the wave equation: $u_{xx} = u_{tt}$. Signals (disturbances) travel along the characteristics (dashed and dotted lines.)

This observation provides us with an alternative way of specifying initial values, which is often quite convenient in practice. Rather than specifying $u(x,0)$ and $u_t(x,0)$ directly, we can specify *initial* left-moving and right-moving parts of the solution, $\ell(x)$ and $r(x)$. Specifically, we set

$$u(x,0) = \ell(x) + r(x) \tag{31}$$

$$u_t(x,0) = \ell'(x) - r'(x) \equiv \frac{d\ell}{dx}(x) - \frac{dr}{dx}(x) \tag{32}$$

Returning now to the solution of the finite-differenced version of the wave equation, it is clear that given the initial data (31–32), we can trivially initialize u_j^0 with *exact* values, but that we can only approximately initialize u^1_j. The question then arises: *How accurately must we initialize the advanced values so as to ensure second order ($O(h^2)$) accuracy of the difference scheme?*

A brief, heuristic answer to this question (which can be more rigorously justified) is as follows. We have $\triangle x = O(h)$, $\triangle t = O(h)$ and the FDA is $O(h^2)$. Since the scheme is $O(h^2)$, we expect that

$$u_{\text{exact}}(x,t) - u_{\text{FD}}(x,t) = O(h^2)$$

for arbitrary, *fixed, FINITE t*. However, the number of time steps required to integrate to time t is $O(\triangle t^{-1}) = O(h^{-1})$. Thus, the per-time-step error must be $O(h^2)/O(h^{-1}) = O(h^3)$, and, therefore, we require

$$(u_{\text{FD}})^{1}_{j} = (u_{\text{exact}})^{1}_{j} + O(h^3)$$

We can readily accomplish this using (1) Taylor series and (2) the equation of motion to rewrite higher time derivatives in terms of spatial derivatives:

$$\begin{aligned} u^{1}_{j} &= u^{0}_{j} + \triangle t\ (u_t)^{0}_{j} + \frac{1}{2}\triangle t^{2}\,(u_{tt})^{0}_{j} + O(\triangle t^{3}) && (33)\\ &= u^{0}_{j} + \triangle t\ (u_t) + \frac{1}{2}\triangle t^{2}\,(u_{xx})^{0}_{j} + O(\triangle t^{3}) && (34) \end{aligned}$$

which, using results from above, can be written as

$$u^{1}_{j} = (\ell + r)_{j} + \triangle t\ (\ell' - r')_{j} + \frac{1}{2}\triangle t^{2}\,(\ell'' + r'')_{j} \tag{35}$$

1.6 Stability Analysis

One of the most frustrating—yet fascinating—features of FD solutions of time dependent problems, is that the discrete solutions often "blow up"—e.g. floating-point overflows are generated at some point in the evolution. Although "blow-ups" can sometimes be caused by legitimate (!) "bugs"—i.e. an incorrect implementation—at other times it is simply the *nature of the FD scheme* which causes problems. We are thus lead to consider the *stability* of solutions of difference equations (as well as their differential-equation progenitors).

Let us again consider the 1-d wave equation (15) and let us now remark that this is a *linear, non-dispersive* wave equation, a consequence of which is the fact that the "size" of the solution does *not* change with time:

$$\|u(x,t)\| \sim \|u(x,0)\|\,, \tag{36}$$

where $\|\cdot\|$ is an suitable norm, such as the L_2 norm:

$$\|u(x,t)\| \equiv \left(\int_0^1 u(x,t)^2\,dx\right)^{1/2}. \tag{37}$$

We will use the property captured by (36) as our working definition of stability. In particular, if you believe (36) is true for the wave equation, then you believe the wave equation is stable.

Fundamentally, if our FDA approximation *converges*, then we expect the same behaviour for the difference solution:

$$\|u^{n}_{j}\| \sim \|u^{0}_{j}\|\,. \tag{38}$$

Now, we construct our FD solution by *iterating in time*, generating

$$u^{0}_{j},\ u^{1}_{j},\ u^{2}_{j},\ u^{3}_{j},\ u^{4}_{j},\ \cdots$$

in succession, using the FD equation

$$u^{n+1}_{j} = 2u^{n}_{j} - u^{n-1}_{j} + \lambda^2\left(u^{n}_{j+1} - 2u^{n}_{j} + u^{n}_{j-1}\right).$$

As it turns out, we are *not* guaranteed that (38) holds for all values of $\lambda \equiv \triangle t \,/\, \triangle x$. In fact, for certain λ (all $\lambda > 1$, as we shall see), we have

$$\|u_j^n\| \gg \|u_j^0\| \,,$$

and for those λ, $\|u^n\|$ *diverges* from u, even (especially!) as $h \to 0$—that is, the difference scheme is *unstable.*

In fact, for many wave problems (including all linear problems), given that a FD scheme is *consistent* (i.e. so that $\hat{\tau} \to 0$ as $h \to 0$), stability is the necessary and sufficient condition for convergence (Lax's theorem).

1.6.1 Heuristic Stability Analysis

Let us write a general time-dependent FDA in the form

$$\mathbf{u}^{n+1} = \mathbf{G}[\mathbf{u}^n] \,, \tag{39}$$

where $\mathbf{G}$ is some *update operator* (linear in our example problem), and $\mathbf{u}$ is a column vector containing sufficient unknowns to write the problem in first-order-in-time form. For example, if we introduce a new, auxiliary set of unknowns, v_j^n, defined by

$$v_j^n = u_j^{n-1} \,,$$

then we can rewrite the differenced-wave-equation (16) as

$$\begin{aligned} u_j^{n+1} &= 2u_j^n - v_j^n + \lambda^2 \left(u_{j+1}^n - 2u_j^n + u_{j-1}^n \right) \,, & (40) \\ v_j^{n+1} &= u_j^n \,, & (41) \end{aligned}$$

so with

$$\mathbf{u}^n = [u_1^n, v_1^n, \; u_2^n, v_2^n, \; \cdots \; u_J^n, v_J^n] \,,$$

(for example), (40-41) is clearly of the form (39). Equation (39) provides us with a compact way of describing the solution of the FDA. Given initial data, $\mathbf{u}^0$, the solution after n time-steps is

$$\mathbf{u}^n = \mathbf{G}^n \mathbf{u}^0 \,, \tag{42}$$

where $\mathbf{G}^n$ is the n-th power of the matrix $\mathbf{G}$. Now, assume that $\mathbf{G}$ has a complete set of orthonormal eigenvectors

$$\mathbf{e}_k, \quad k = 1, 2, \cdots J \,,$$

and corresponding eigenvalues

$$\mu_k, \quad k = 1, 2, \cdots J \,,$$

so that

$$\mathbf{G}\, \mathbf{e}_k = \mu_k \, \mathbf{e}_k, \quad k = 1, 2, \cdots J \,.$$

We can then write the initial data as (spectral decomposition):

$$\mathbf{u}^0 = \sum_{k=1}^{J} c_k^0 \, \mathbf{e}_k \,,$$

where the c_k^0 are coefficients. Using (42), the solution at time-step n is then

$$\begin{aligned} \mathbf{u}^n &= \mathbf{G}^n \left(\sum_{k=1}^{J} c_k^0 \, \mathbf{e}_k \right) & (43) \\ &= \sum_{k=1}^{J} c_k^0 \, (\mu_k)^n \, \mathbf{e}_k \,. & (44) \end{aligned}$$

Clearly, if the difference scheme is to be stable, we must have

$$|\mu_k| \le 1 \qquad k = 1, 2, \cdots J \tag{45}$$

(Note: μ_k will be complex in general, so $|\mu|$ denotes the complex modulus, $|\mu| \equiv \sqrt{\mu \mu^\star}$).

Geometrically, then, the eigenvalues of the update matrix must lie on or within the unit circle (see Figure 6).

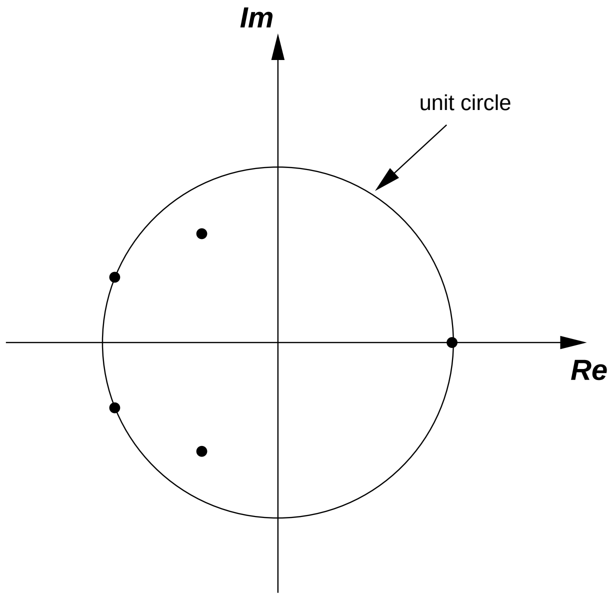

Figure 6: Schematic illustration of location in complex plane of eigenvalues of update matrix $\mathbf{G}$. In this case, all eigenvalues (dots) lie on or within the unit circle, indicating that the corresponding finite difference scheme is stable.

1.6.2 Von-Neumann (Fourier) Stability Analysis

Von-Neumann stability analysis is based on the ideas sketched above, but additionally assumes that the difference equation is linear with constant coefficients, and that the boundary conditions are periodic. We can then use Fourier analysis, which has the same benefits in the discrete domain—difference operators in real-space variable $x \longrightarrow$ algebraic operations in Fourier-space variable k—as it does in the continuum Schematically, instead of writing

$$\mathbf{u}^{n+1}(x) = \mathbf{G}[\mathbf{u}^{n}(x)]\,,$$

we consider the Fourier-domain equivalent:

$$\tilde{\mathbf{u}}^{n+1}(k) = \tilde{\mathbf{G}}[\tilde{\mathbf{u}}^{n}(k)]\,,$$

where k is the wave-number (Fourier-space variable) and $\tilde{\mathbf{u}}$ and $\tilde{\mathbf{G}}$ are the Fourier-transforms of $\mathbf{u}$ and $\mathbf{G}$, respectively. Specifically, we define the Fourier-transformed grid function via

$$\tilde{\mathbf{u}}^{n}(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{-ikx}\, \mathbf{u}^{n}(x)\, dx\,. \tag{46}$$

For a general difference scheme, we will find that

$$\tilde{\mathbf{u}}^{n+1}(k) = \tilde{\mathbf{G}}(\xi)\, \tilde{\mathbf{u}}^{n}(k)\,,$$

where $\xi \equiv kh$, and we will have to show that $\tilde{\mathbf{G}}(\xi)$'s eigenvalues lie within or on the unit circle for all conceivable ξ. The appropriate range for ξ is

$$-\pi \le \xi \le \pi\,,$$

since the shortest wavelength representable on a uniform mesh with spacing h is $\lambda = 2h$ (Nyquist limit), corresponding to a maximum wave number $k = (2\pi)/\lambda = \pm\pi/h$.

Let us consider the application of the Von-Neumann stability analysis to our current model problem. We first define a (non-divided) difference operator D^2 as follows:

$$D^2\, u(x) = u(x+h) - 2u(x) + u(x-h)\,.$$

Then, suppressing the spatial grid index, we can write the first-order form of the difference equation (40-41) as

$$\begin{aligned} u^{n+1} &= 2u^{n} - v^{n} + \lambda^2 D^2 u^{n}, \\ v^{n+1} &= u^{n}, \end{aligned}$$

or

$$\begin{bmatrix} u \\ v \end{bmatrix}^{n+1} = \begin{bmatrix} 2+\lambda^2 D^2 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}^{n}. \qquad (47)$$

In order to perform the Fourier transform, we need to know the action of D^2 in Fourier-space. Using the transform inverse to (46) we have

$$u(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} e^{ikx}\,\tilde{u}(k)\,dk\,,$$

so

$$\begin{aligned} D^2\,u(x) = u(x+h) - 2u(x) + u(x-h) &= \int_{-\infty}^{+\infty} \left(e^{ikh} - 2 + e^{-ikh}\right) e^{ikx}\,\tilde{u}(k)\,dk \\ &= \int_{-\infty}^{+\infty} \left(e^{i\xi} - 2 + e^{-i\xi}\right) e^{ikx}\,\tilde{u}(k)\,dk\,. \end{aligned}$$

Now consider the quantity $-4\sin^2(\xi/2)$:

$$\begin{aligned} -4\sin^2\frac{\xi}{2} &= -4\left(\frac{e^{i\xi/2} - e^{-i\xi/2}}{2i}\right)^2 \\ &= \left(e^{i\xi/2} - e^{-i\xi/2}\right)^2 = e^{i\xi} - 2 + e^{-i\xi}\,, \end{aligned}$$

so

$$D^2\,u(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \left(-4\sin^2\frac{\xi}{2}\right) e^{ikx}\,\tilde{u}(k)\,dk\,.$$

In summary, under Fourier transformation, we have

$$\begin{aligned} \mathbf{u}(x) &\longrightarrow \tilde{\mathbf{u}}(k)\,, \\ D^2\mathbf{u}(x) &\longrightarrow -4\sin^2\frac{\xi}{2}\tilde{\mathbf{u}}(k)\,. \end{aligned}$$

Using this result in the Fourier transform of (47), we see that we need to compute the eigenvalues of

$$\begin{bmatrix} 2 - 4\lambda^2\sin^2(\xi/2) & -1 \\ 1 & 0 \end{bmatrix},$$

and determine the conditions under which the eigenvalues lie on or within the unit circle. The characteristic equation (whose roots are the eigenvalues) is

$$\begin{vmatrix} 2 - 4\lambda^2\sin^2(\xi/2) - \mu & -1 \\ 1 & -\mu\,, \end{vmatrix} = 0$$

or

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

This equation has roots

$$\mu(\xi) = \left(1 - 2\lambda^2\sin^2\frac{\xi}{2}\right) \pm \left(\left(1 - 2\lambda^2\sin^2\frac{\xi}{2}\right) - 1\right)^{1/2}.$$

We now need to find sufficient conditions for

$$|\mu(\xi)| \le 1,$$

or equivalently

$$|\mu(\xi)|^2 \le 1.$$

To this end, we note that we can write

$$\mu(\xi) = (1-Q) \pm ((1-Q)^2 - 1)^{1/2},$$

where the quantity, Q

$$Q \equiv 2\lambda \sin^2 \frac{\xi}{2},$$

is *real* and *non-negative* ($Q \ge 0$). There are now two cases to consider:

1. $(1-Q)^2 - 1 \le 0$,
2. $(1-Q)^2 - 1 > 0$.

In the first case, $((1-Q)^2-1)^{1/2}$ is purely imaginary, so we have

$$|\mu(\xi)|^2 = (1-Q)^2 + (1-(1-Q)^2) = 1.$$

In the second case, $(1-Q)^2 - 1 > 0 \longrightarrow (1-Q)^2 > 1 \longrightarrow Q > 2$, and then we have

$$1 - Q - ((1-Q^2)-1)^{1/2} < -1,$$

so, in this case, our stability criterion will *always* be violated. We thus conclude that a necessary condition for Von-Neumann stability is

$$(1-Q)^2 - 1 \le 0 \longrightarrow (1-Q)^2 \le 1 \longrightarrow Q \le 2.$$

Since $Q \equiv 2\lambda \sin^2(\xi/2)$ and $\sin^2(\xi/2) \le 1$, we must therefore have

$$\lambda \equiv \frac{\triangle t}{\triangle x} \le 1,$$

for stability of our scheme (16). This condition is often called the CFL condition—after Courant, Friedrichs and Lewy who derived it in 1928 (the ratio $\lambda = \triangle x \,/\, \triangle t$ is also frequently called the *Courant number*). In practical terms, we must limit the time-discretization scale , $\triangle t$, to values no larger than the space-discretization scale, $\triangle x$. Furthermore, this type of instability has a "physical" interpretation, often summarized by the statement *the numerical domain of dependence of an explicit difference scheme must contain the physical domain of dependence.*

1.7 Dispersion and Dissipation

Let us now consider an even simpler model "wave equation" than (1), the so-called *advection*, or *color* equation:

$$\begin{aligned} u_t &= a\,u_x \quad (a > 0) \qquad -\infty < x < \infty \quad , \quad t \ge 0 \\ u(x,0) &= u_0(x) \end{aligned} \tag{48}$$

which has the exact solution

$$u(x,t) = u_0(x + at) \tag{49}$$

Equation (48) is another example of a non-disspative, non-dispersive partial differential equation.

To remind ourselves what "non-dispersive" means, let us note that (48) admits "normal mode" solutions:

$$u(x,t) \sim e^{ik(x+at)} \equiv e^{i(kx+\omega t)}$$

where $\omega \equiv ka$; in general, of course, $\omega \equiv \omega(k)$ is known as the *dispersion relation*, and

$$\frac{d\omega}{dk} \equiv \text{ speed of propagation of mode with wave number } k$$

In the current case, we have

$$\frac{d\omega}{dk} = a = \text{constant}$$

which means that all modes propagate at the same speed, which is precisely what is meant by "non-dispersive". Further, if we consider resolving the general initial profile, $u_0(x)$, into "normal-mode" (Fourier) components, we find that the magnitudes of the components are preserved in time, i.e. equation (48) is also *non-dissipative*.

Ostensibly, we would like our finite-difference solutions to have the same properties—i.e. to be dissipationless and dispersionless, but, in general, this will not be (completely) possible. We will return to the issue of dissipation and dispersion in FDAs of wave problems below.

1.8 The Leap-Frog Scheme

First note that (48) is a good prototype for the general hyperbolic *system*:

$$\mathbf{u}_t = \mathbf{A}\mathbf{u}_x \tag{50}$$

where $\mathbf{u}$(x,t) is the n-component *solution vector*:

$$\mathbf{u}(x,t) = [u_1(x,t),\, u_2(x,t),\, \cdots\, u_n(x,t)] \tag{51}$$

and the $n \times n$ matrix $\mathbf{A}$ has distinct real eigenvalues

$$\lambda_1,\, \lambda_2,\, \cdots\, \lambda_n$$

so that, for example, there exists a similarity transformation $\mathbf{S}$ such that

$$\mathbf{S}\mathbf{A}\mathbf{S}^{-1} = \text{diag}(\lambda_1,\, \lambda_2,\, \cdots\, \lambda_n)$$

The leap-frog scheme is a commonly used finite-difference approximation for hyperbolic systems. In the context of our simple scalar ($n = 1$) advection problem (48):

$$u_t = a\, u_x$$

an appropriate stencil is shown in Figure 7. Applying the usual $O(h^2)$ approximations to ∂_x and ∂_t, our leap-frog (LF) scheme is

$$\frac{u^{n+1}_j - u^{n-1}_j}{2\,\triangle t} = a\,\frac{u^n_{j+1} - u^n_{j-1}}{2\,\triangle x} \tag{52}$$

or explicitly

$$u^{n+1}_j = u^{n-1}_j + a\lambda\left(u^n_{j+1} - u^n_{j-1}\right) \tag{53}$$

where

$$\lambda \equiv \frac{\triangle t}{\triangle x}$$

is the *Courant number* as previously.

Exercise: Perform a von Neumann stability analysis of (52) thus showing that $a\lambda \le 1$ (which, you should note, is just the CFL condition) is necessary for stability.

Observe that the LF scheme (52) is a *three-level* scheme. As in our treatment of the wave equation, $u_{tt} = u_{xx}$ using the "standard scheme", we need to specify

$$u^0_j \quad , \quad u^1_j \qquad j = 1, 2, \cdots J$$

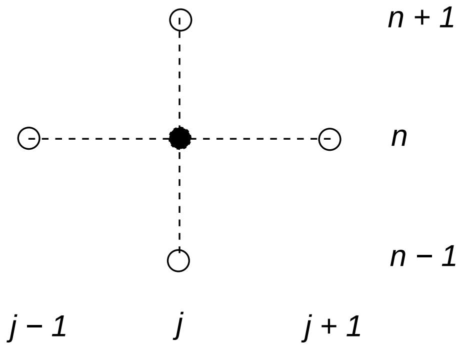

Figure 7: Stencil (molecule/star) for leap-frog scheme as applied to (48). Note that the central grid point has been filled in this figure to emphasize that the corresponding unknown, u_j^n, does not appear in the local discrete equation at that grid point (hence the term "leap-frog")

to "get the scheme going"—that is, we need to specify *two* numbers per spatial grid point. This should be contrasted to the continuum where we need to specify only *one* number per x_j, namely $u_0(x_j)$. Again, the initialization of the u_j^0 is trivial, given the (continuum) initial data $u_0(x)$, and, again, we need u_j^1 to $O(\triangle t^3) = O(h^3)$ accuracy for $O(h^2)$ global accuracy. Two possible approaches are as follows.

Taylor Series: The development here is parallel to that for the wave equation. We have

$$u_j^1 = u_j^0 + \triangle t\,(u_t)_j^0 + \frac{1}{2}\triangle t^2\,(u_{tt})_j^0 + O(\triangle t^2)$$

also, from the equation of motion $u_t = a u_x$, we get

$$u_{tt} = (u_t)_t = (a u_x)_t = a\,(u_t)_x = a^2 u_{xx}.$$

so we have our desired initialization formula:

$$u_j^1 = u_j^0 + \triangle t\,(u_0')_j^0 + \frac{1}{2}\triangle t^2\left(a^2 u_0''\right)_j^0 + O(\triangle t^3) \tag{54}$$

Self-Consistent Iterative Approach: The idea here is to initialize the u_j^1 from the u_j^0 and a version of the discrete equations of motion which introduces a "ficticious" half-time-level—see Figure 8.

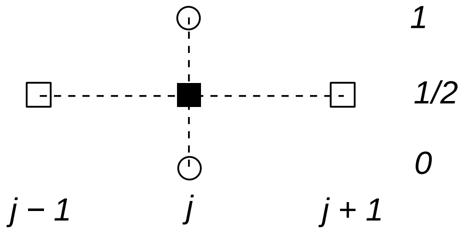

Figure 8: Stencil for initialization of leap-frog scheme for to (48). Note the introduction of the "fictitious" half-time level $t = t^{1/2}$ (squares).

Applying the leap-frog scheme on the stencil in the Figure, we have

$$\frac{u_j^1 - u_j^0}{\triangle t} = a\,\frac{u_{j+1}^{\frac{1}{2}} - u_{j-1}^{\frac{1}{2}}}{2\,\triangle x}$$

or, explicitly solving for u_j^1:

$$u_j^1 = u_j^0 + \frac{1}{2}\lambda\left(u_{j+1}^{\frac{1}{2}} - u_{j-1}^{\frac{1}{2}}\right)$$

It is a straightforward exercise to show that in order to retain $O(h^2)$ accuracy of the difference scheme, we need "fictitious-time" values, $u_j^{1/2}$ which are accurate to $O(h^2)$ (i.e. we can neglect terms which are of $O(h^2)$). In particular, if we *define* $u_j^{1/2}$, via

$$u_j^{\frac{1}{2}} = \frac{u_j^1 + u_j^0}{2}$$

which amounts to defining the half-time values via linear interpolation in the advanced and retarded unknowns, we will retain second-order accuracy.

We are thus led to the following initialization algorithm which is perhaps best expressed in pseudo-code (note, all loops over j are implicit:)

```
u[0,j] := u_0(x_j)
u[1,j] := u_0(x_j)
DO
   usave[j] := u[1,j]
   u[1/2,j] := (u[1,j] + u[0,j]) / 2

   u[1,j]   := u[0,j] + (lambda / 2) * (u[1/2,j+1] - u[1/2,j-1])

UNTIL  norm(usave[j] - u[1,j]) < epsilon
```

1.9 Error Analysis and Convergence Tests

As a side remark, we note that the discussion in this section applies to essentially *any* continuum problem which is solved using FDAs on a *uniform* mesh structure. In particular, the discussion applies to the treatment of ODEs and elliptic problems, where, in fact, convergence is often easier to achieve due to the fact that the FDAs are typically intrinsically stable (i.e. we have an easier time constructing stable FDAs for these types of problems). We also note that departures from non-uniformity in the mesh do not, in general, complete destroy the picture, but, rather, tend to distort it in ways which are beyond the scope of these notes. In any case, my colleagues have been known to paraphrase my whole approach to this subject as

Convergence!, Convergence!, Convergence!

1.9.1 Sample Analysis: The Advection Equation

We again consider the solution of the advection equation, but this time we impose periodic boundary conditions on our spatial domain, which we take to be $0 \le x \le 1$ with $x = 0$ and $x = 1$ identified (i.e. we solve the wave equation on $\mathbf{S}^1 \times \mathbf{R}$):

$$\begin{aligned} u_t &= a\,u_x \quad (a > 0) \qquad 0 \le x \le 1, \quad t \ge 0 \\ u(x,0) &= u_0(x) \end{aligned} \tag{55}$$

Note that the initial conditions $u_0(x)$ must be compatible with periodicity, i.e. we must specify periodic initial data.

Again, given the initial data, $u_0(x)$, we can immediately write down the full solution

$$u(x,t) = u_0(x + a\,t \bmod 1) \tag{56}$$

where mod is the usual modulus function which "wraps" $x + a\ t$, $t > 0$ onto the unit interval. As we shall see, because of the simplicity and solubility of this problem, one can perform a rather complete closed-form ("analytic") treatment of the convergence of simple FDAs of (55). The point of the exercise, however, is *not* to advocate parallel closed-form treatments for more complicated problems. Rather, the key idea to be extracted from the following is that, in principle (always), and in practice (almost always, i.e. I've never seen a case where it *didn't* work, but then there's a lot of computations I haven't seen):

The error, e^h, of an FDA is no less computable than the solution, u^h itself.

This has widespread ramifications, one of which is that there's really no excuse for publishing solutions of FDAs without error bars, or their equivalents!

Proceeding with our sample error analysis of the leap-frog scheme applied to the advection equation, we first introduce some difference operators for the usual $O(h^2)$ centred approximations of ∂_x and ∂_t:

$$D_x u^n_j \equiv \frac{u^n_{j+1} - u^n_{j-1}}{2\,\triangle x} \qquad (57)$$

$$D_t u^n_j \equiv \frac{u^{n+1}_j - u^{n-1}_j}{2\,\triangle t} \qquad (58)$$

We again take

$$\triangle x \equiv h \qquad \triangle t \equiv \lambda\,\triangle x = \lambda\,h$$

and will hold λ fixed as h varies, so that, as usual, our FDA is characterized by the single scale parameter, h.

The idea behind our error analysis is that we want to view the solution of the FDA as a *continuum* problem, and hence we will express both the difference operators and the FDA solution as asymptotic series (in h) of differential operators, and continuum functions, respectively. We have the following expansions for D_x and D_t:

$$D_x = \partial_x + \frac{1}{6}h^2\,\partial_{xxx} + O(h^4) \qquad (59)$$

$$D_t = \partial_t + \frac{1}{6}\lambda^2 h^2\,\partial_{ttt} + O(h^4) \qquad (60)$$

Now, in terms of the general, abstract formulation of (1.3), we have:

$$L\,u - f = 0 \quad \Longleftrightarrow \quad (\partial_t - a\,\partial_x)\,u = 0 \qquad (61)$$

$$L^h u^h - f^h = 0 \quad \Longleftrightarrow \quad (D_t - a\,D_x)\,u^h = 0 \qquad (62)$$

$$L^h u - f^h \equiv \tau^h \quad \Longleftrightarrow \quad (D_t - a\,D_x)\,u \equiv \tau^h = \frac{1}{6}h^2\left(\lambda^2\partial_{ttt} - a\,\partial_{xxx}\right)u + O(h^4) = O(h^2) \qquad (63)$$

The Richardson ansatz: The key to our analysis is L.F. Richardson's old observation (*ansatz*) [5], that the solution, u^h, of *any* FDA which (1) uses a uniform mesh structure with scale parameter h, and (2) is completely centred, should have the following expansion in the limit $h \to 0$:

$$u^h(x,t) = u(x,t) + h^2 e_2(x,t) + h^4 e_4(x,t) + \cdots \qquad (64)$$

Here u is the continuum solution, while e_2, e_4, $\cdots$ are (continuum) *error functions* which *do not depend on* h. In a very real sense (64), is *the* key expression from which all error analysis of FDAs derives. We note that in the case that the FDA is *not* completely centred, we will have to modify the *ansatz*. In particular, for first order schemes (which are more common in numerical relativity and relativistic astrophysics than one might expect!), we will have

$$u^h(x,t) = u(x,t) + he_1(x,t) + h^2 e_x(x,t) + h^3 e_3(x,t) + \cdots \qquad (65)$$

Note that the Richardson *ansatz* (64) is completely compatible with the assertion discussed in (1.3.7), namely that

$$\tau^h = O(h^2) \quad \longrightarrow \quad e^h \equiv u - u^h = O(h^2) \qquad (66)$$

However, the Richardson form (64) contains much more information than "second-order truncation error should imply second-order solution error", telling us the precise form of the h dependence of u^h.

Given the Richardson expansion, we can now proceed with our error analysis. We start from the FDA, $L^h u^h - f^h = 0$, and replace both L^h and u^h with continuum expansions:

$$\begin{aligned} L^h u^h = 0 \qquad &\longrightarrow \qquad (D_t - a\,D_x)\left(u + h^2 e_2 + \cdots\right) = 0 \\ &\longrightarrow \qquad \left(\partial_t + \frac{1}{6}\lambda^2 h^2 \partial_{ttt} - a\,\partial_x - \frac{1}{6} a h^2\,\partial_{xxx} + \cdots\right)\left(u + h^2 e_2 + \cdots\right) = 0 \end{aligned} \tag{67}$$

We now demand that terms in (67) vanish order-by-order in h. At $O(1)$ (zeroth-order), we have

$$(\partial_t - a\,\partial_x)\,u = 0 \tag{68}$$

which is simply a statement of the *consistency* of the difference approximation. More interestingly, at $O(h^2)$ (second-order), we find

$$(\partial_t - a\,\partial_x)\,e_2 = \frac{1}{6}\left(a\partial_{xxx} - \lambda^2 \partial_{ttt}\right) u \tag{69}$$

which, assuming that we view u as a "known" function, is simply a PDE for the leading order error function, e_2. Moreover, the PDE governing e_2 is of *precisely* the same nature as the original PDE (48).

In fact, we can *solve* (69) for e_2. Given the "natural" initial conditions

$$e_2(x, 0) = 0$$

(i.e. we initialize the FDA with the exact solution so that $u^h = u$ at $t = 0$), and defining $q(x + at)$:

$$q(x + at) \equiv \frac{1}{6} a\left(1 - \lambda^2 a^2\right)\partial_{xxx} u(x, t)$$

we have

$$e_2(x, t) = t\,q(x + at \bmod 1) \tag{70}$$

We note that, as is typical for leap-frog, we have *linear* growth of the finite difference error with time (to leading order in h). We also note that we can obviously push this analysis to higher order in h—what results, then, is an entire *hierarchy* of differential equations for u and the error functions e_2, e_4, e_6, $\cdots$. Indeed, it is useful to keep this view in mind:

> When one solves an FDA of a PDE, one is *not* solving some system which is "simplified" relative to the PDE, rather, one is solving a much *richer* system consisting of an (infinite) hierarchy of PDEs, one for each function appearing in the Richardson expansion (64).

In the general case, of course, we will not be able to solve the PDE governing u, let alone that governing e_2—otherwise we wouldn't be considering the FDA in the first place! But it is precisely in this instance where the true power of Richardson's observation is evident. The key observation is that starting from (64), and computing FD solutions using the same initial data, but with differing values of h, we can learn a great deal about the error in our FD approximations. The whole game of investigating the manner in which a particular FDA converges or doesn't (i.e. looking at what happens as one varies h) is known as *convergence testing*. It is important to realize that there are no hard and fast rules for convergence testing; rather, one tends to tailor the tests to the specifics of the problem at hand, and, being largely an empirical approach, one gains experience and intuition as one works through more and more problems. That said, I emphasize again that the Richardson expansion, in some form or other, *always* underlies convergence analysis of FDAs.

A simple example of a convergence test, and the one I use most often in practice is the following. We compute three distinct FD solutions u^h, u^{2h}, u^{4h} at resolutions h, $2h$ and $4h$ respectively, but using the same initial data (as naturally expressed on the 3 distinct FD meshes). We also assume that the finite difference meshes "line up", i.e. that the $4h$ grid points are a subset of the $2h$ points which are a subset of the h points, so that, in particular, the $4h$ points constitute a common set of events (x_j, t^n) at which specific grid function values can be directly (i.e. no interpolation required) and meaningfully compared to one another. From the Richardson *ansatz* (64), we expect:

$$\begin{aligned} u^h &= u + h^2 e_2 + h^4 e_4 + \cdots \\ u^{2h} &= u + (2h)^2 e_2 + (2h)^4 e_4 + \cdots \\ u^{4h} &= u + (4h)^2 e_2 + (4h)^4 e_4 + \cdots \end{aligned}$$

We then compute a quantity $Q(t)$, which I will call a *convergence factor*, as follows:

$$Q(t) \equiv \frac{\|u^{4h} - u^{2h}\|_x}{\|u^{2h} - u^h\|_x} \tag{71}$$

where $\|\cdot\|_x$ is any suitable discrete spatial norm, such as the ℓ_2 norm, $\|\cdot\|_2$:

$$\|u^h\|_2 = \left(J^{-1} \sum_{j=1}^{J} \left(u_j^h\right)^2 \right)^{1/2} \tag{72}$$

and, for concreteness, the subtractions in (71) can be taken to involve the sets of mesh points which are common between u^{4h} and u^{2h}, and between u^{2h} and u^h. It is a simple exercise to show that, if our finite difference scheme is converging, then we should find:

$$\lim_{h\to 0} Q(t) = 4. \tag{73}$$

In practice, one can use additional levels of discretization, $8h$, $16h$, etc. to extend this test to look for "trends" in $Q(t)$ and, in short, to convince oneself (and, with luck, others), that the FDA really *is* converging. Moreover, once convergence of an FDA has been established, then a point-wise subtraction of any two solutions computed at different resolutions, will immediately provide an estimate of the level of error in both. For example, if we have u^h and u^{2h}, then, again by the Richardson *ansatz* we have

$$u^{2h} - u^h = \left(\left(u + (2h)^2 e_2 + \cdots\right) - \left(u + h^2 e_2 + \cdots\right)\right) = 3h^2 e_2 + O(h^4) \sim 3e^h \sim \frac{3}{4} e^{2h} \tag{74}$$

Richardson extrapolation: Richardson's observation (64) also provides the basis for all the techniques of *Richardson extrapolation*, where solutions computed at different resolutions are linearly combined so as to *eliminate* leading order error terms, and hence provide more accurate solutions. As an example, given u^h and u^{2h} which satisfy (64), we can take the linear combination, $\bar{u}^h$:

$$\bar{u}^h \equiv \frac{4u^h - u^{2h}}{3} \tag{75}$$

which, by (64), is easily seen to be $O(h^4)$, i.e. *fourth*-order accurate!

$$\begin{aligned} \bar{u}^h &\equiv \frac{4u^h - u^{2h}}{3} = \frac{4\left(u + h^2 e_2 + h^4 e_4 + \cdots\right) - \left(u + 4h^2 e_2 + 16h^4 e_4 + \cdots\right)}{3} \\ &= -4h^4 e_4 + O(h^6) = O(h^4) \end{aligned} \tag{76}$$

When it works, Richardson extrapolation has an almost magical quality about it, but one generally has to start with fairly accurate (on the order of a few %) solutions in order to see the dramatic improvement in accuracy suggested by (76). Partly because it is still a struggle to achieve that sort of accuracy (i.e. a few %) for *any* computation in many areas of numerical relativity/astrophysics, techniques based on Richardson extrapolation have not had a major impact in this context.

Independent Residual Evaluation A question which often arises in discussions of convergence testing is the following:

> "OK, you've established that u^h is converging as $h \to 0$, but how do you know you're converging to u, the solution of the continuum problem?"

Here, the notion of an independent residual evaluation is very useful. The idea is as follows: we have our continuum PDE

$$Lu - f = 0 \tag{77}$$

and our FDA

$$L^h u^h - f^h = 0 \tag{78}$$

We have demonstrated that u^h is apparently converging by, for example, computing the convergence factor (71) and verifying that it tends to 4 as h tends to 0. However, we do not know if we have derived and/or implemented our discrete operator L^h correctly. Note that implicit in the "implementation" is the fact that, particularly for multi-dimensional and/or implicit and/or multi-component FDAs, considerable "work" (i.e. analysis and coding) may be involved in setting up and solving the algebraic equations for u^h. As a check that we *are* converging to u, we consider a *distinct* (i.e. independent) discretization of the PDE:

$$\tilde{L}^h \tilde{u}^h - f^h = 0 \tag{79}$$

The only thing we need from this FDA for the purposes of the independent residual test is the new FD operator $\tilde{L}^h$. As with L^h, we can expand $\tilde{L}^h$ in powers of the mesh spacing:

$$\tilde{L}^h = L + h^2 E_2 + h^4 E_4 + \cdots \tag{80}$$

where E_2, E_4, $\cdots$ are higher order (involve higher order derivatives than L) differential operators. We then simply apply the new operator $\tilde{L}^h$ to our FDA u^h and investigate what happens as $h \to 0$. If u^h *is* converging to the continuum solution, u, we will have

$$u^h = u + h^2 e_2 + O(h^4) \tag{81}$$

and we will compute

$$\tilde{L}^h u^h = \left(L + h^2 E_2 + O(h^4)\right)\left(u + h^2 e_2 + O(h^4)\right) = Lu + h^2 (E_2\, u + L\, e_2) = O(h^2) \tag{82}$$

i.e., $\tilde{L}^h u^h$ will be a residual-like quantity which converges quadratically as $h \to 0$. Conversely, if we have goofed in our derivation and/or implementation of $L^h u^h = f^h = 0$, but we still see convergence; i.e. we have, for example, $u^{2h} - u^h \to 0$ as $h \to 0$, then we must have something like

$$u^h = u + e_0 + h e_1 + h^2 e_2 + \cdots \tag{83}$$

where the crucial fact is that the error must have an $O(1)$ component, e_0. In this case, we will compute

$$\tilde{L}^h u^h = \left(L + h^2 E_2 + O(h^4)\right)\left(u + e_0 + h e_1 + h^2 e_2 + O(h^4)\right) = Lu + Le_0 + hLe_1 + O(h^2) = Le_0 + O(h) \tag{84}$$

and, unless we are *extraordinarily* lucky, and $L\, e_0$ vanishes, we will *not* observe the expected convergence, rather, we will see $\tilde{L}^h u^h - f^h$ tending to a *finite* ($O(1)$) value—a sure sign that something is wrong.

There is of course, the problem that we might have slipped up in our implementation of the "independent residual evaluator", $\tilde{L}^h$, in which case the results from our test will be ambigous at best! However, a key point here is that because $\tilde{L}^h$ is only used *a posteriori* on a computed solution (we never use it to compute $\tilde{u}^h$, for example) it is a relatively easy matter to ensure that $\tilde{L}^h$ has been implemented in an error-free fashion (perhaps using symbolic manipulation facilities). Furthermore, many of the restrictions commonly placed on the "real" discretization (such as stability and the ease of solution of the resulting algebraic equations) do not apply to $\tilde{L}^h$.

Finally, we note that although we have assumed in the above that L, L^h and $\tilde{L}^h$ are *linear*, the technique of independent residual evaluation works equally well for non-linear problems.

1.10 Dispersion and Dissipation in FDAs

We again consider the advection model problem, $u_t = a\,u_x$, but now discretize only in space (semi-discretization) using the usual $O(h^2)$ centred difference approximation:

$$u_t = a\,D_x\,u \equiv a\,\frac{u_{j+1} - u_{j-1}}{2\,\triangle x} \tag{85}$$

We now look for normal-mode solutions to (85) of the form

$$u = e^{ik\,(x+a'\,t)}$$

where the "discrete phase speed", a', is to be determined. Substitution of this *ansatz* in (85) yields

$$ika'u = \frac{a\,(2i\sin(k\triangle x\,))}{2\triangle x}u$$

or, solving for the discrete phase speed, a'

$$a' = a\frac{\sin(k\triangle x\,)}{k\triangle x} = a\frac{\sin\xi}{\xi}$$

where we have defined the dimensionless wave number, ξ:

$$\xi \equiv k\triangle x$$

In the *low frequency* limit, $\xi \to 0$, we have the expected result:

$$a' = a\frac{\sin\xi}{\xi} \to a$$

so that low frequency components propagate with the correct phase speed, a. However, in the *high frequency* limit, $\xi \to \pi$, we have

$$a' = a\frac{\sin\xi}{\xi} \to 0 \quad !!$$

i.e. the highest frequency components of the solution don't propagate at all! This is typical of FDAs of wave equations, particularly for relatively low-order schemes. The propagation of high frequency components of the difference solution is essentially completely wrong. Arguably then, there can be little harm in attenuating (dissipating) these components, and, in fact, since high frequency components are potentially troublesome (particularly *vis a vis* non-linearities and the treatment of boundaries), it is often *advantageous* to use a dissipative difference scheme.

Some FDAs are naturally dissipative (the Lax-Wendroff scheme, for example), while others, such as leap-frog, are not. In the case of a leap-frog-based scheme, the idea is to add dissipative terms to the method, but in such a way as to retain $O(h^2)$ accuracy of the scheme. Consider, for example, the leap-frog scheme as applied to the advection model problem:

$$u_j^{n+1} = u_j^{n-1} + a\lambda\left(u_{j+1}^{n} - u_{j-1}^{n}\right)$$

We add dissipation to the scheme by modifying it as follows:

$$u_j^{n+1} = u_j^{n-1} + a\lambda\left(u_{j+1}^{n} - u_{j-1}^{n}\right) - \frac{\epsilon}{16}\left(u_{j+2}^{n-1} - 4u_{j+1}^{n-1} + 6u_j^{n-1} - 4u_{j-1}^{n-1} + u_{j-2}^{n-1}\right) \tag{86}$$

where ϵ is an adjustable, non-negative parameter. Note that

$$\begin{aligned} u_{j+2}^{n-1} - 4u_{j+1}^{n-1} + 6u_j^{n-1} - 4u_{j-1}^{n-1} + u_{j-2}^{n-1} &= \triangle x^4 (u_{xxxx})_j^{n-1} + O(h^6) \\ &= \triangle x^4 (u_{xxxx})_j^{n} + O(h^5) = O(h^4) \end{aligned}$$

so that the term which is added, does not change the leading order truncation error, which in the form we have written the equation, is $O(\triangle t^3) = O(h^3)$ (local/one-step truncation error).

A Von Neumann analysis of the modified scheme shows that, in addtion to the CFL condition $\lambda \le 1$, we must have $\epsilon < 1$ for stability, and, further, that the per-step amplification factor for a mode with wave number ξ is, to leading order

$$1 - \epsilon \sin^4 \frac{\xi}{2}$$

Thus the addition of the dissipation term is analagous to the use of an explicit "high frequency filter" (low-pass filter), which has a fairly sharp rollover as $\xi \to \pi$.

We note that an advantage to the use of explicit dissipation techniques (versus, for example, the use of an intrinsically dissipative scheme) is that the amount of dissipation can be controlled by tuning the dissipation parameter.

2 Introduction to Finite Differerence Solution of Time-Independent (Elliptic) Problems

2.1 Outline

- Overview
- Typical Setup of an Elliptic Problem
- Model Problem
- Discretization of Model Problem
- Traditional Solution of Discrete Equations
  - Direct Methods
  - Relaxation Methods (Jacobi, Gauss-Seidel, SOR)
  - Convergence of Relaxation Methods
  - Relaxation as Smoother (Prelude to Multi-Grid)
- The Multi-grid Method
  - Motivation
  - Multi-level Algorithms
  - Multi-grid for Linear Problems (LCS Scheme)
  - Computational Cost of Multi-Grid
  - Multi-grid for Non-Linear Problems (FAS Scheme)

2.2 Overview

In addition to time-dependent PDEs primarily of hyperbolic type, time-independent, or *elliptic* PDEs also play an important role in numerical relativity. Specifically, elliptic PDEs in general *must* be solved in order to determine initial data for the Einstein equations which are consistent with the constraint equations (the Hamiltonian and momentum constraint). At the continuum level, the Bianchi identities guarantee that initial data which is consistent with the constraints will remain so under subsequent evolution via the Einstein field equations. Computationally, however, so called *free-evolution* schemes, in which the constraints are solved *only* at the initial time, will in general *not* preserve discrete versions of the constraints under evolution. Historically, the non-preservation of the constraints has often been associated with instability (non-convergence) of the overall numerical scheme, and researchers have thus often used so called *constrained* schemes, in which the discretized constraint equations are re-solved periodically during an evolution—most often at every time step.

In addition, in many instances, coordinate choices, such as the well known *maximal slicing* condition will lead to additional elliptic equations that need to be solved at every time step during an evolution.

A key property of elliptic equations is their *global* nature. In contrast to hyperbolic equations, there are generally *no* characteristics in the solution of such equations, and thus no "marching" techniques by which a solution can be constructed step by step, with each step yielding a solution on a $d-1$ dimensional sub-manifold for a d dimensional solution. Rather, given data on some $d-1$ dimensional boundary of some d dimensional domain, the solution must be determined throughout the domain in "one go", and, further, change in *any* of the boundary data generally changes the solution everywhere in the domain.

Largely due to this global property, "traditional" methods for solving discrete (finite-differenced) versions of elliptic equations were sub-optimal in the sense that the algorithms were slower (sometimes considerably slower) than $O(N)$, where N is the total number of grid points in the discretization. This lead to a view that still persists in the numerical relativity community that the treatment of elliptics was *necessarily* slow (computer resource intensive), and that therefore, the solution of elliptic PDEs was to be avoided if at all possible.
One of the key points of what follows below is to emphasize that with "non-traditional" methods, this view is largely unfounded. In particular, the multi-grid method, due to Achi Brandt and others, provides the basis for the $O(N)$ solution of general elliptic systems, and should be part of the arsenal of techniques of any practicing numerical relativist.

2.3 Typical Setup of an Elliptic Problem

For our purposes, an "elliptic equation" will be taken to be synonymous with "boundary-value problem" (BVP), and although elliptic problems in three dimensions, $d = 3$, have been routinely solved in numerical relativity and other fields for many years now, the following development will focus (for convenience) on 2D model equations. Figure 9 schematically illustrates the basic setup of a typical two-dimensional boundary

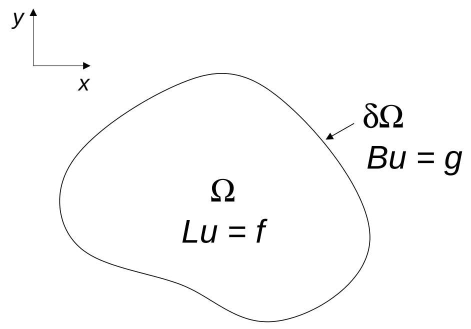

Figure 9: Typical setup of a two-dimensional boundary value problem. The solution domain is Ω with boundary $\partial\Omega$. Appropriate boundary conditions, $Bu = g$, must be applied everywhere on the boundary in order to uniquely determine the solution, $u \equiv u(x, y)$, throughout the solution domain, subject to the interior elliptic equation, $Lu = g$

value problem. Here, we seek a solution $u(x, y)$ that satisfies

$$Lu(x, y) = f(x, y) \qquad \text{on} \qquad \Omega \tag{87}$$

where L is an elliptic differential operator, and $f(x, y)$ is some specified "source" or "forcing" function, and

$$Bu(x, y) = g(x, y) \qquad \text{on} \qquad \partial\Omega \tag{88}$$

where B is another operator that encodes the boundary conditions, and $g(x, y)$ is another source function.

In general, there are three types of boundary conditions:

1. *Dirichlet* conditions, in which the value of u on $\partial\Omega$ is specified.
2. *Neumann* conditions, in which the value of the normal derivative $\partial u/\partial n$ on $\partial\Omega$ is given.
3. *Mixed* or *Robin* conditions, in which some linear combination $\alpha(x,y)u + \beta(x,y)\partial u/\partial n$ is specified.

2.4 Model Problem

A convenient model problem for the following discussion is the Poisson equation on the unit square with (homogeneous) Dirichlet boundary conditions. That is, we consider

$$\nabla u(x,y) \equiv u_{xx} + u_{yy} = f(x,y) \tag{89}$$

on the domain,

$$\Omega : 0 \le x \le 1\,,\ 0 \le y \le 1 \tag{90}$$

subject to the boundary conditions

$$u(0,y) = u(1,y) = u(x,0) = u(x,1) = 0\,. \tag{91}$$

2.5 Discretization of Model Problem

In order to discretize our model problem using finite difference methods, we adopt the same basic approach that we applied to time-dependent problems. In this case, this first entails replacing the continuum domain, Ω, with a discrete set of grid points constituting the discrete domain, Ω^h. Again, for simplicity, as well as to extremize truncation error accuracy for a given width of finite difference stencil, we adopt a *uniform* discretization, with a single, constant mesh spacing, h, in each of the coordinate directions. This results in a finite difference grid, Ω^h, having n grid points in each direction, with $h = 1/(n-1)$. The total number of points in the discretization is thus thus $N = n^2$. Figure 10 illustrates the discrete domain for the case $h = 1/4$, $n = 5$. Note that the finite difference mesh points are defined by

$$\{(x_i, y_j) \equiv ((i-1)\,h, (j-1)_h)\,,\ i,j = 1,2,\cdots n\} \tag{92}$$

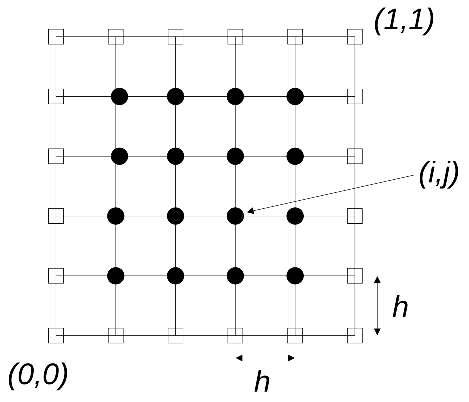

Figure 10: Discrete domain for the model problem for the case $h = 1/4$, $n = 5$. Solid circles denote discrete *interior* points, while the open squares are the discrete *boundary* points. A typical interior grid point having coordinates, $((i-1)h, (j-1)h)$ is labelled (i,j)

Having set up the finite difference mesh, we now proceed to the discretization of the model problem, replacing the continuum system (89) with a discrete version

$$\begin{aligned} L^h u^h &= f^h &(93)\\ B^h u^h &= g^h &(94)\end{aligned}$$

Here u^h is the discrete solution, with individual values denoted $u_{i,j}$, L^h and B^h are discrete versions of the differential operators L and B, and f^h and g^h are the values of the source function *restricted* to the finite difference mesh (i.e. evaluated at the the discrete mesh values).

In order to discretize the Laplacian operator, we need finite difference approximations to the second-order derivatives u_{xx} and u_{yy}. Here we use the standard second-order, centred approximations:

$$u_{xx} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2) \tag{95}$$

$$u_{yy} = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2) \tag{96}$$

Substitution of the above in (89) yields the desired discretization of the Poisson equation:

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = f_{i,j} \qquad 2 \le i,j \le n-1 \tag{97}$$

Note that (97) may be applied at all interior points (see Figure 10). The Dirichlet boundary conditions (91) provide (trivial) equations for the boundary values on the discrete domain:

$$u_{1,j} = u_{n,j} = u_{i,1} = u_{i,n} \; 1 \le i,j \le n \tag{98}$$

(Observe that the four “corner” values, $u(1,1), u(1,n), u(n,1)$ and $u(n,n)$ actually completely decouple for the rest of the scheme, but, algorithmically, it may prove useful to retain them.)

We note that the discretization of (89-91) using (97-98) generally results in a large sparse linear system of equations. Specifically, consider some enumeration of the unknowns $u_{i,j}$ in terms of a single index K:

$$u_{i,j} \longrightarrow u_K \tag{99}$$

so that we can view the discrete grid function as a single vector $\mathbf{u}$. One specific example is provided by a so called *lexicographic* ordering such as

$$K \equiv (j-1)n + i \tag{100}$$

so that the various index pairs appearing in (97) map as follows

$$\begin{aligned}
(i,j) &\longrightarrow K && (101)\\
(i+1,j) &\longrightarrow K+1 && (102)\\
(i-1,j) &\longrightarrow K-1 && (103)\\
(i,j+1) &\longrightarrow K+n && (104)\\
(i,j-1) &\longrightarrow K-n && (105)\\
& && (106)
\end{aligned}$$

Then (97) becomes

$$\frac{u_{K+1} + u_{K-1} + u_{K+n} + u_{K-n} - 4u_K}{h^2} = f_K \tag{107}$$

When supplemented with the appropriately remapped discrete boundary conditions (98), we see that we must solve a linear system

$$\mathbf{L}\mathbf{u} = \mathbf{f} \tag{108}$$

where $\mathbf{L}$ is an $n^2 \times n^2$ sparse, banded matrix, with bandwidth $2n+1$, and $\mathbf{u}$ and $\mathbf{f}$ are n-component vectors.

Standard results from linear analysis tell us that the computational cost to *directly* solve a linear system of size N and bandwidth w using LU decomposition (Gaussian elimination) is $O(w^2 N)$. Thus, at least naively, the cost to directly solve our discrete boundary value problem (108) is $O(N^2)$. This is, of course, very bad news, and, although the cost can be reduced somewhat through clever reordering of the equations in the linear system, direct methods are thus seldom used in the solution of boundary value problems in more than one dimension.

Consequently, *iterative* methods, such as *relaxation*, and more recently, *multi-grid*, have come to dominate the solution of linear systems arising from the discretization of elliptic PDEs, and we will thus focus on these. (Another class of iterative method that will not be discussed here, but which has found use in numerical relativity, is the Krylov subspace method—GMRES (generalized minimum residual) and BiCGSTAB (stabilized biconjugate gradient) are specific members of this class).

2.6 Relaxation Methods

Relaxation methods are among the oldest and best known techniques for the iterative solution of large sparse linear systems, such as those arising from the discretization of elliptic PDEs. As with almost any technique, relaxation has both advantages and disadvantages. On the positive side, the method is generally very easy (almost trivial!) to code, it works for many discrete BVPs, and it is extremely storage efficient. On the other hand, relaxation methods are characterized by slow convergence, with the time for solution *per grid point* tending to increase as the mesh spacing, h, tends to 0. This disadvantage is severe enough that relaxation should almost *never* be used in practice, except if one can get away with relatively small meshes, or if one really is only interested in just a few solutions.

This then leads to the question: Why study such methods at all? In the context of these notes, the answer is that relaxation forms the core of arguably the best method for the solution of discretized elliptic equations—the multi-grid method.

2.6.1 Jacobi and Gauss-Seidel Relaxation

The key idea underlying relaxation techniques is quite intuitive. We associate a single equation and a corresponding single unknown, $u_{i,j}$, with each mesh point in Ω^h. We then repeatedly "sweep" through the mesh, visiting each mesh point in some prescribed order. Each time a point is visited, we adjust the value of the unknown at that grid point so that the corresponding equation is ("instantaneously") satisifed.

To give a concrete illustration of this process, we adopt a "residual based" approach to locally satisfying the discrete equations. We also proceed in a manner that generalizes immediately to the solution of *non-linear* elliptic PDEs, which, naturally, lead to systems of *non-linear* algebraic equations upon discretization. In addition, for the sake of presentation, we will temporarily ignore the treatment of boundary conditions, bearing in mind that if those conditions are *differential* in nature (i.e. Neumann / Dirichlet), then their discretizations will have to be relaxed as well.

As just mentioned, we will adopt a "residual-based" approach to the problem of locally satisfying equations via relaxation. To that end we consider our general form of a discretized BVP

$$L^h u^h = f^h \tag{109}$$

and recast this in the canonical form

$$F^h\left[u^h\right] = 0\,. \tag{110}$$

We call this form "canonical" since we have 0 on the right hand side of the equation. Clearly any system of the form (109) can be trivially rewritten in the form (110). We also stress that the quantity u^h which appears above is the *exact* solution of the difference equations.

In solving the system (110) iteratively, we will generally only compute the exact discrete solution, u^h, in the limit of infinite iteration. We thus introduce the quantity $\tilde{u}^h$, to denote the "current" or "working" approximation to u^h, so that labelling the iteration number by n (not to be confused with the number of grid points per edge of the finite difference grid), and assuming that our iterative technique *does* converge, we have

$$\lim_{n\to\infty} \tilde{u}^h = u^h \tag{111}$$

Associated with the current approximation, $\tilde{u}^h$ is the residual, $\tilde{r}^h$, defined by

$$\tilde{r}^h \equiv L^h \tilde{u}^h - f^h \tag{112}$$

or in terms of the canonical form (110),

$$\tilde{r}^h \equiv F^h \left[\tilde{u}^h\right] . \tag{113}$$

When referring to a specific *component* (grid value) of the residual, $\tilde{r}^h_{i,j}$, we can without danger of confusion, drop the h superscript, and write

$$\tilde{r}_{i,j} = \left[L^h \tilde{u}^h - f^h\right]_{i,j} \equiv \left[F^h \left[u^h\right]\right]_{i,j} \tag{114}$$

For the specific case of our model problem we have

$$\tilde{r}_{i,j} = h^{-2} \left(\tilde{u}_{i+1,j} + \tilde{u}_{i-1,j} + \tilde{u}_{i,j+1} + \tilde{u}_{i,j-1} - 4\tilde{u}_{i,j}\right) - f_{i,j} \tag{115}$$

The task of the relaxation method is to adjust the value of $\tilde{u}_{i,j}$ so that the corresponding residual is "instantaneously" zeroed. To do this, it is useful to appeal to Newton's method for the solution of a single non-linear equation in a single unknown. In the current case, because the difference equation is *linear* in $\tilde{u}_{i,j}$, we can solve the equation with a single Newton step. Again, however, we can also apply relaxation to *non-linear* difference equations, in which case we would generally only zero the residual in the limit of an infinite number of Newton steps.

We thus write the relaxation in terms of an update, $\delta\tilde{u}^{(n)}_{i,j}$ of the unknown

$$\tilde{u}^{(n)}_{i,j} \longrightarrow \tilde{u}^{(n+1)}_{i,j} = \tilde{u}^{(n)}_{i,j} + \delta\tilde{u}^{(n)}_{i,j} \tag{116}$$

where, again, (n) labels the iteration number. Appealing to Newton's method, we have

$$\begin{aligned}
\tilde{u}^{(n+1)}_{i,j} &= \tilde{u}^{(n)}_{i,j} - \tilde{r}_{i,j} \left[\frac{\partial F^h_{i,j}}{\partial u_{i,j}} \bigg|_{u_{i,j}=\tilde{u}^{(n)}_{i,j}} \right]^{-1} && (117)\\
&= \tilde{u}^{(n)}_{i,j} - \frac{\tilde{r}_{i,j}}{-4h^{-2}} && (118)\\
&= \tilde{u}^{(n)}_{i,j} + \frac{1}{4} h^2 \tilde{r}_{i,j} && (119)
\end{aligned}$$

The precise computation of the residual requires some clarification. We are considering an iterative method which ultimately takes an entire vector of unknowns $\mathbf{u}^{(n)}$ to an new estimate $\mathbf{u}^{(n+1)}$, but which works on a component by component basis. In the computation of the individual residuals, we could either choose only "old" values; i.e. values from iteration n, or, wherever available, we could use "new" values from iteration $n+1$, with the rest from iteration n.

The first approach is known as *Jacobi relaxation*, and for our current set of difference equations would mean the residual would be computed as

$$\tilde{r}_{i,j} = h^{-2} \left(\tilde{u}^{(n)}_{i+1,j} + \tilde{u}^{(n)}_{i-1,j} + \tilde{u}^{(n)}_{i,j+1} + \tilde{u}^{(n)}_{i,j-1} - 4\tilde{u}^{(n)}_{i,j}\right) - f_{i,j} \tag{120}$$

The second approach is known as *Gauss-Seidel relaxation*, and, assuming a lexicographic ordering of unknowns, $i = 1, 2, \cdots n$, $j = 1, 2, \cdots n$, with the i index varying most rapidly, our residual is computed via

$$\tilde{r}_{i,j} = h^{-2} \left(\tilde{u}^{(n)}_{i+1,j} + \tilde{u}^{(n+1)}_{i-1,j} + \tilde{u}^{(n)}_{i,j+1} + \tilde{u}^{(n+1)}_{i,j-1} - 4\tilde{u}^{(n)}_{i,j}\right) - f_{i,j} \tag{121}$$

We now make a few observations and comments about these two relaxation methods. First, tacit in the description of the above methods is the fact that at each iteration we "visit" each and every unknown *exactly once* and modify its value so that the local equation is instantaneously satisfied. Thus, a complete pass through the mesh of unknowns (i.e. a complete iteration) is often known as a *relaxation sweep.*

For the Jacobi method, the order in which we visit unknowns is clearly irrelevant to what values are obtained at the end of each iteration, and hence Jacobi relaxation is often known as *simultaneous relaxation.* The structure of the iteration is advantageous from the point of view of parallelization, but storage is required for *both* the new and old values of $\tilde{u}_{i,j}$.

For Gauss-Seidel (GS), on the other hand, one only needs storage for the current estimate of $\tilde{u}_{i,j}$, and the sweeping order *does* impact the details of the solution process. Thus, lexicographic ordering does not lend itself to parallelization, whereas for difference equations such as those for our model problem, which have nearest-neigbour couplings, so called *red-black* ordering *can* be parallelized, and turns out to have other advantages. Red-black ordering can be understood by appealing to the red and black squares on a chess board, and is composed of two subiterations (which can be performed in arbitrary order)

1. Visit and modify all "red" points, i.e. all (i, j) such that $\mathrm{mod}(i + 1, 2) = 0$

2. Visit and modify all "black" points, i.e. all (i, j) such that $\mathrm{mod}(i + 1, 2) = 1$

Within each sub-iteration the order in which individiual red or black values is modified is irrelevant, as is the case for Jacobi.

2.6.2 Convergence of Relaxation Methods

A key issue (if not *the* key issue) with any iterative technique is whether or not the iteration will converge. For methods such as Gauss-Seidel, this was an issue that was examined comprehensively in the 1960's and 1970's, and it is beyond the scope of these notes to delve into any level of detail concerning this matter.

As a very rough rule-of-thumb, GS will converge if the linear system being solved is *diagonally dominant.* That is, casting our (linear) difference equations

$$L^h u^h = f^h \tag{122}$$

in the matrix form

$$\mathbf{A}\mathbf{u} = \mathbf{b} \tag{123}$$

where $\mathbf{A}$ is an $N \times N$ matrix (N being the total number of unknowns), and $\mathbf{u}$ and $\mathbf{b}$ are N-component vectors, the matrix $\mathbf{A}$ is said to be diagonally dominant if

$$|a_{ij}| \le \sum_{j=1, j\ne i}^{N} |a_{ij}|\,, i = 1, 2, \cdots N \tag{124}$$

with strict inequality holding for at least on value of i.

Another practical issue that arises with any iterative method is *how* to monitor convergence. With relaxation methods, there are two natural quantities (scalars) that one can compute in this regard: the *residual norm* $\|\tilde{r}^h\|$ and the *solution update norm* $\|\tilde{u}^{(n+1)} - \tilde{u}^{(n)}\|$, both of which tend to 0 in the limit of infinite iteration, should the method be convergent. In practice, monitoring the residual norm is straightforward and often sufficient. One should note that with Gauss-Seidel, the norm of the true residual $\tilde{r}^h$ (i.e. the N-component vector) is constantly changing, and that is therefore useful to define the concept of the *running residual*, which is the collection of individual residuals computed during any single relaxation sweep.

Let us consider the issue of convergence in a little more detail. For a general iterative process leading to a solution vector $\mathbf{u}$, and starting from some initial estimate (guess) of the solution, $\mathbf{u}^{(0)}$, we have

$$\mathbf{u}^{(0)} \to \mathbf{u}^{(1)} \to \cdots \to \mathbf{u}^{(n)} \to \mathbf{u}^{(n+1)} \to \cdots \to \mathbf{u} \tag{125}$$

Similarly, for the residual vector, we have

$$\mathbf{r}^{(0)} \to \mathbf{r}^{(1)} \to \cdots \to \mathbf{r}^{(n)} \to \mathbf{r}^{(n+1)} \to \cdots \to \mathbf{0} \tag{126}$$

and, finally, for the solution error, $\mathbf{e}^{(n)} \equiv \mathbf{u} - \mathbf{u}^{(n)}$, we have

$$\mathbf{e}^{(0)} \to \mathbf{e}^{(1)} \to \cdots \to \mathbf{e}^{(n)} \to \mathbf{e}^{(n+1)} \to \cdots \to \mathbf{0} \tag{127}$$

For linear relaxation (and the basic idea generalizes to the non-linear case) we can view the transformation of the error, $\mathbf{e}^{(n)}$, at each iteration in terms of a linear matrix (operator), $\mathbf{G}$, known as the *error amplification matrix.* We then have

$$\mathbf{e}^{(n+1)} = \mathbf{G}\mathbf{e}^{(n)} = \mathbf{G}^2\mathbf{e}^{(n-1)} = \cdots = \mathbf{G}^n\mathbf{e}^{(0)} \tag{128}$$

Asymptotically then, the convergence of the iteration is determined by the *spectral radius*, $\rho(\mathbf{G})$, of the error amplification matrix, $\mathbf{G}$, where $\rho(\mathbf{G})$ is given by

$$\rho(\mathbf{G}) \equiv \max_i |\lambda_i(\mathbf{G})| \tag{129}$$

and the λ_i are the eigenvalues of $\mathbf{G}$. That is, in general we have

$$\lim_{n\to\infty} \frac{\|\mathbf{e}^{(n+1)}\|}{\|\mathbf{e}^{n}\|} = \rho(\mathbf{G}) \tag{130}$$

and we can then define the *asymptotic convergence rate*, R, via

$$R \equiv \log_{10}\left(\rho^{-1}\right) \tag{131}$$

A useful interpretation of R is that its reciprocal, R^{-1} yields the number of iterations necessary to asymptotically decrease $\|\mathbf{e}^{(n+1)}\|$ by an order of magnitude.

We can now consider the computational cost of solving (general) discrete BVPs using relaxation. Here "cost" is to be roughly identified with CPU time, and we approach the issue from the point of view of complexity analysis; i.e. we are interested in how the cost scales with the key numerical parameters in the problem. In thise instance we will assume that there is *one* key parameter, which is the total number of grid points, N, in our discrete domain, Ω^h. For concreteness, and to simplify the analysis, we make the following assumptions

1. The domain Ω is d-dimensional ($d = 1, 2, 3$ being most common)

2. There are n grid points per edge of the (uniform) discrete domain, Ω^h

The total number of grid points, N, is thus given by

$$N = n^d \tag{132}$$

We will further define the computational cost or computational work, $W \equiv W(N)$ to be the work required to reduce the error, $\|\mathbf{e}\|$ in the computed solution, by an order of magnitude. Ideally, perhaps, one would like to be able to compute the work needed to reduce the error to some specified level, but for the purposes of comparison of methods, and from the point of view of complexity analysis, this definition will suffice. We observe that, clearly, the best case situation is that $W(N) = O(N)$.

Turning now to the case of Gauss-Seidel relaxation, we state without proof that for our model problem (and for many other elliptic systems in $d = 1, 2, 3$), we have

$$\rho\left(\mathbf{G}_{\mathrm{GS}}\right) = 1 - O(h^2) \tag{133}$$

This in turn implies that the relaxation work, W_{GS}, required to reduce the solution error by an order of magnitude is given by

$$W_{\mathrm{GS}} = O\left(h^{-2}\,\mathrm{sweeps}\right) = O\left(n^2\,\mathrm{sweeps}\right) \tag{134}$$

Now, each relaxation sweep costs $O(n^d) = O(N)$, so we have

$$W_{\mathrm{GS}} = O\left(n^2 N\right) = O\left(N^{2/d} N\right) = O\left(N^{(d+2)/d}\right) \tag{135}$$

Tabulating the values for $d = 1, 2$ and 3 we find

d	$W_{\rm GS}$
1	$O(N^3)$
2	$O(N^2)$
3	$O(N^{5/3})$

Although the scaling improves as the dimensionality, d, increases, $O(N^2)$ and $O(N^{5/3})$ for the cases $d = 2$ and $d = 3$ are pretty bad, and, again, it is for this reason that Gauss-Seidel is rarely used in practice, particularly for large problems (large values of n, small values of h).

2.6.3 Successive Over Relaxation (SOR)

Historically, researchers studying Gauss-Seidel relaxation found that the convergence of the method could often be substantially improved by systematically "over correcting" the solution, relative to what the usual GS computation would dictate. Algorithmically, the change from GS to SOR is very simple, and is given by the following formula:

$$\tilde{u}^{(n+1)}_{i,j} = \omega\, \hat{u}^{(n+1)}_{i,j} + (1-\omega)\, \tilde{u}^{(n)}_{i,j} \tag{136}$$

Here, ω is the so-called *overrelaxation parameter*, typically chosen in the range $1 \le \omega < 2$, and $\hat{u}^{(n+1)}_{i,j}$ is the value computed from the normal Gauss-Seidel iteration. Note that when $\omega = 1$, we recover the GS iteration itself, and that for large enough ω, e.g. $\omega \ge 2$, the iteration will tend to become unstable.

Under *ideal* conditions, the use of SOR reduces the number of relaxation sweeps required to drop the error norm by an order of magnitude to $O(n)$. That is, we have

$$W_{\rm SOR} = O\,(nN) = O\left(N^{1/d}N\right) = O\left(N^{(d+1)/d}\right) \tag{137}$$

Again tabulating the values for $d = 1, 2$ and 3 we find

d	$W_{\rm SOR}$
1	$O(N^2)$
2	$O(N^{3/2})$
3	$O(N^{4/3})$

We thus note that *optimal* SOR is not entirely unreasonable for at least moderately-sized $d = 3$ problems.

The issue, of course, is how to choose $\omega = \omega_{\rm OPT}$ in order to get optimal performance. Except for those cases where $\rho_{\rm GS} \equiv \rho(G_{\rm GS})$ can be computed explicitly, $\omega_{\rm OPT}$ will generally need to be determined *empirically* on a case-by-case *and* resolution-by-resolution basis. If $\rho_{\rm GS}$ *is* known, then one has

$$\omega_{\rm OPT} = \frac{2}{1+\sqrt{1-\rho_{\rm GS}}} \tag{138}$$

2.6.4 Relaxation as Smoother (Prelude to Multi-Grid)

As seen in the previous section, the slow convergence rate of relaxation methods such as Gauss-Seidel means that they do not provide a very good way of solving discretized elliptic problem. We now consider an even simper model problem than previously to illustrate what relaxation *does* tend to do well, which in turn will elucidate why relaxation is so essential to the multi-grid method.

Let us thus consider a one dimensional ($d = 1$) "ellliptic" model problem, specifically an *ordinary* differential equation that is to be solved as a two-point boundary value problem. The equation to be solved is

$$Lu(x) \equiv \frac{d^2u}{d^2x} = f(x) \tag{139}$$

where $f(x)$ is specified source function. We solve (139) on a domain, Ω, given by

$$\Omega\,:\, 0 \le x \le 1 \tag{140}$$

subject to the (homogeneous) boundary conditions

$$u(0) = u(1) = 0 \tag{141}$$

To discretize this problem, we introduce a uniform mesh, Ω^h

$$\Omega^h = \{(i-1)h,\ i = 1, 2, \cdots n\} \tag{142}$$

where $n = h^{-1} + 1$ and h, as usual, is the mesh spacing. We now discretize (139) to $O(h^2)$ using the usual centred difference approximation for the second derivative, yielding

$$\frac{u_{i+1} - 2u_i + u_{i-1}}{h^2} = f_i\ ,\ i = 2, 3, \cdots n-1 \tag{143}$$

The above equations, combined with the boundary conditions

$$u_1 = u_n = 0 \tag{144}$$

constitute a set of n linear equations in the n unknowns, $u_i, i = 1, 2, \cdots n$. For the analysis it proves useful to eliminate u_1 and u_n from the discretization (which is possible since their values are known), so that in terms of our generic form

$$L^h u^h = f^h \tag{145}$$

u^h and f^h are to be viewed as $n-2$-component vectors, while L^h is an $(n-2)\times(n-2)$ matrix.

We now consider a specific relaxation method, known as *damped Jacobi*, which is chosen for ease of analysis, but, in terms of the lessons that we wish to glean from the analysis, is quite representative of other relaxation techniques, including Gauss-Seidel. Using vector notation, the damped Jacobi iteration is given by

$$\tilde{\mathbf{u}}^{(n+1)} = \tilde{\mathbf{u}}^{(n)} - \omega \mathbf{D}^{-1} \tilde{\mathbf{r}}^{(n)} \tag{146}$$

where $\tilde{\mathbf{u}}$ is the approximate solution of the difference equations, $\tilde{\mathbf{r}}$ is the residual vector, ω is an (underrelaxation) parameter, and $\mathbf{D}$ is the main diagonal of L^h. For the case under consideration we have

$$\mathrm{D} = -2h^{-2}\mathbf{1} \tag{147}$$

Note that when $\omega = 1$, this is just the usual Jacobi relaxation method discussed previously.

We now examine the effect of (146) on the residual vector, $\tilde{\mathbf{r}}$. We have

$$\begin{aligned}
\tilde{\mathbf{r}}^{(n+1)} &\equiv L^h \tilde{\mathbf{u}}^{(n+1)} - f^h &(148)\\
&= L^h \left(\tilde{\mathbf{u}}^{(n)} - \omega \mathbf{D}^{-1}\tilde{\mathbf{r}}^{(n)}\right) - f^h &(149)\\
&= \left(\mathbf{1} - \omega L^h \mathbf{D}^{-1}\right)\tilde{\mathbf{r}}^{(n)} &(150)\\
&\equiv \mathbf{G}_{\mathrm{R}} \tilde{\mathbf{r}}^{(n)} &(151)
\end{aligned}$$

where, in analogy to the previously considered error amplification matrix, the *residual amplification matrix*, $\mathbf{G}_{\mathrm{R}}$, is given by

$$\mathbf{G}_{\mathrm{R}} \equiv \mathbf{1} - \omega L^h \mathbf{D}^{-1} \tag{152}$$

Clearly we have the following relation for the residual at the n-th iteration, $\tilde{\mathbf{r}}^{(n)}$, in terms of the initial residual, $\tilde{\mathbf{r}}^{(0)}$:

$$\tilde{\mathbf{r}}^{(n)} = (\mathbf{G})^n_{\mathrm{R}}\, \tilde{\mathbf{r}}^{(0)} \tag{153}$$

Now, it transpires that the $(n-2)\times(n-2)$ matrix, $\mathbf{G}_{\mathrm{R}}$, has a complete set of orthogonal eigenvectors, $\phi_m\ ,\ m = 1, 2, \cdots n-2$ with corresponding eigenvalues μ_m. Thus, in particular, we can expand the initial residual, $\tilde{\mathbf{r}}^{(0)}$, in terms of the ϕ_m:

$$\tilde{\mathbf{r}}^{(0)} = \sum_{m=1}^{n-2} c_m \phi_m \tag{154}$$

where the c_m are coefficients. This immediately yields the following expansion for the residual at the n-th iteration:

$$\tilde{\mathbf{r}}^{(n)} = \sum_{m=1}^{n-2} c_m \left(\mu_m\right)^n \phi_m \, . \tag{155}$$

Thus, after n sweeps, the m-th "Fourier" component of the initial residual vector, $\tilde{r}^{(0)}$ is reduced by a factor of $(\mu_m)^n$.

Again, for illustrative purposes, we now consider the specific value of the underrelaxation parameter, $\omega = 1/2$. Then it is left to the reader to verify that the eigenvectors, ϕ_m are given by

$$\phi_m = \left[\, \sin\left(\pi m h\right), \sin\left(2\pi m h\right), \cdots, \sin\left((n-2)\pi m h\right) \right]^T \ , \ m = 1, 2, \cdots, n-2 \tag{156}$$

while the corresponding eigenvalues, μ_m are

$$\mu_m = \cos^2\left(\frac{1}{2}\pi m h\right) \ , \ m = 1, 2, \cdots, n-2 \tag{157}$$

We note that each of the eigenvectors, ϕ_m, has an associated "wavelength", λ_m, which can be identified by the relation

$$\sin\left(\pi m x\right) = \sin\left(2\pi x/\lambda_m\right) \ . \tag{158}$$

Thus we have

$$\lambda_m = \frac{2}{m} \tag{159}$$

so as m increases (and λ_m decreases), the "frequency" of ϕ_m *increases*, and, from (157), the eigenvalues, μ_m, *decrease* in magnitude.

As discussed above, the asymptotic convergence rate of the relaxation scheme is determined by the *largest* of the μ_m, which is μ_1

$$\mu_1 = \cos^2\left(\frac{1}{2}\pi h\right) = 1 - \frac{1}{4}\pi^2 h^2 + \cdots = 1 - O(h^2) \, . \tag{160}$$

This is the same basic result previously quoted for the Gauss-Seidel iteration and implies that $O(n^2)$ sweeps are required to reduce the norm of the residual by an order of magnitude.

We thus see that the asymptotic convergence rate of this relaxation scheme is dominated by the (slow) convergence of *smooth* (low frequency, long wavelength) components of the residual, $\tilde{\mathbf{r}}^{(n)}$, or, equivalently, the solution error, $\tilde{\mathbf{e}}^{(n)}$. Again, this comment applies to many other relaxation schemes applied to typical elliptic problems, including Gauss-Seidel.

It is now highly instructive to consider what happens to *high* frequency components of the residual (or solution error) as the damped Jacobi iteration is applied. In particular, we consider those components of $\tilde{r}^h$ which cannot be represented on a coarser grid, Ω^{2h}. As can be seen from Figure 11, we are thus concerned with the eigenvectors, ϕ_m, such that

$$\lambda_m \le 4h \longrightarrow mh \ge \frac{1}{2} \tag{161}$$

In this case, we have

$$\mu_m = \cos^2\left(\frac{1}{2}\pi m h\right) \le \cos^2\left(\frac{\pi}{4}\right) \le \frac{1}{2} \tag{162}$$

Thus, *all* components of the residual (or solution error) that cannot be represented on Ω^{2h} get supressed by a factor of at least $1/2$ *per sweep*. Furthermore, the rate at which these high-frequency components are liquidated is *independent* of the mesh spacing, h.

To summarize, relaxation tends to be a *dismal solver* of systems $L^h u^h = f^h$, arising from FDAs of elliptic PDEs. However, it tends to be a *very good smoother* of such systems, which, as we will see is crucial for the efficacy of the multi-grid method.

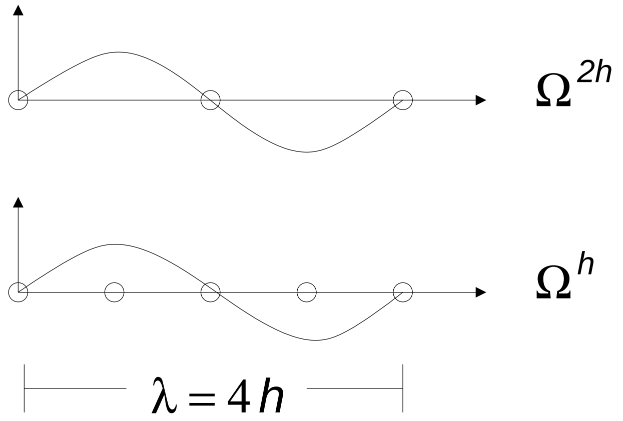

Figure 11: Illustration of the definition of "high frequency" components of the residual or solution error, on a fine grid, Ω^h, relative to a coarse grid, Ω^{2h}. As illustrated in the figure, any wave with $\lambda < 4h$ *cannot* be represented on the coarse grid (i.e. the Nyquist limit of the coarse grid is $4h$.)

2.7 The Multi-grid Method

2.7.1 Motivation and Introduction

Consider again our model problem

$$u(x,y)_{xx} + u_{yy} = f(x,y) \tag{163}$$

solved on the unit square subject to homogeneous boundary conditions. We again discretize the system using the standard 5-point $O(h^2)$ FDA on a uniform $n \times n$ grid with mesh spacing h, resulting in the algebraic system $L^h u^h = f^h$. We also assume that we will apply an iterative solution technique, so that we will start with some initial estimate $\tilde{\mathbf{u}}^{(0)}$, then iterate until $\|\tilde{\mathbf{r}}^{(n)}\| \le \epsilon$.

Several questions concerning the solution of the discrete problem then arise:

1. How do we choose n (equivalently, h)?
2. How do we choose ϵ?
3. How do we choose $\tilde{\mathbf{u}}^{(0)}$?
4. How fast can we "solve" $L^h u^h = f^h$?

We will now provide at least partial answers to all of these questions.

1) Choosing n (equivalently h)

Ideally, we would choose the mesh spacing h such that the error in the discrete solution, $\|\tilde{\mathbf{u}} - \mathbf{u}\|$, satisfies $\|\tilde{\mathbf{u}} - \mathbf{u}\| < \epsilon_u$, where ϵ_u is some user-prescribed error tolerance.

Now, from our discussions of FDAs applied to time-dependent problems, we already know how to estimate the solution error for essentially *any* finite difference solution. For concreteness, we assume that our FDA has been constructed from centred, $O(h^2)$ difference approximations. We then appeal to Richardson, and note that, for solutions computed on grids with mesh spacings h and $2h$, we expect

$$u^h \sim u + h^2 e_2 + h^4 e_4 + \cdots \tag{164}$$
$$u^{2h} \sim u + 4h^2 e_2 + 16h^4 e_4 + \cdots \tag{165}$$

so that, to leading order in h, the error in the solution, $u^h - u$, is given by

$$e \sim \frac{u^{2h} - u^h}{3h^2} \tag{166}$$

That is, the basic strategy is to perform *convergence tests*—comparing the finite difference solutions at different resolutions, and increasing (or decreasing!) h until the level of error is deemed satisfactory.

2) Choosing ϵ (ϵ_r)

Consider the following 3 expressions:

$$\begin{aligned} L^h u^h - f^h &= 0 && (167)\\ L^h \tilde{u}^h - f^h &= \tilde{r}^h && (168)\\ L^h u - f^h &= \tau^h && (169) \end{aligned}$$

where u^h is the (exact) solution of the finite difference equations, $\tilde{u}^h$ is the approximate solution of the FDA at any stage of the iteration, and u is the (exact) solution of the continuum problem, $Lu - f = 0$. Now, (167) is simply our FDA written in a canonical form, (168) defines the residual, while (169) defines the truncation error.
Comparing (168) and (169), we see that is natural to stop an iterative process for the solution of (167) when

$$\|\tilde{r}^h\| \sim \|\tau^h\| \tag{170}$$

This leaves us with the problem of estimating the size of the truncation error (not to be confused with the solution error!), and we will see below how estimates of τ^h arise naturally within the context of multi-grid algorithms.

3) Choosing $\tilde{\mathbf{u}}^{(0)}$

The key idea here is to use the solution of a coarse-grid problem as an initial estimate for the fine-grid problem. That is, assume that we have determined that we wish to compute a solution at resolution h; i.e. that we wish to solve

$$L^h u^h = f^h \tag{171}$$

Then we first pose and solve (perhaps approximately) the corresponding problem (i.e. same domain, boundary conditions and source function, f) on a mesh with spacing $2h$. That is, we solve

$$L^{2h} u^{2h} = f^{2h} \tag{172}$$

and then set the initial estimate, $(u^h)^{(0)}$ via

$$\left(u^h\right)^{(0)} = \bar{I}^{h}_{2h}\, u^{2h} \tag{173}$$

where $\bar{I}^h_{2h}$ is known as a *prolongation operator* and transfers a coarse-grid function to a fine-grid. Typically, $\bar{I}^h_{2h}$ (which is a discrete representation of an identity operator) will perform d-dimensional polynomial interpolation of the coarse-grid unknown, u^{2h}, to a suitable order in the mesh spacing, h.

The chief advantage of this approach is that the solution of the coarse-grid problem should be inexpensive to solve relative to the fine-grid problem; in particular, the cost of solving on the coarse-grid should be no more than 2^{-d} of the cost of soving the fine-grid equations.

Furthermore, we can apply this basic approach *recursively*, initializing u^{2h} with the prolonged solution of a problem on a grid with spacing $4h$, which itself can be initialized via the solution of a problem with spacing $8h$ etc.

We are thus lead to consider a general *multi-level* technique for the treatment of discretized elliptic PDEs, in which the solution of a fine-grid problem is preceded by the solution of a series of coarse-grid problems. We label each distinct mesh spacing with an integer ℓ

$$\ell = 1, 2, \cdots \ell_{\max} \tag{174}$$

where $\ell = 1$ corresponds to the coarsest mesh spacing h_1, while $\ell = \ell_{\max}$ labels the finest mesh spacing $h_{\ell_{\max}}$. Operationally, it almost always most convenient (and usually most efficient) to use mesh spacings, h_ℓ, that satisfy

$$h_{\ell+1} = \frac{1}{2} h_\ell \tag{175}$$

which then implies

$$n_{\ell+1} \sim 2^d n_\ell \,. \tag{176}$$

In what follows, we will always assume that our multi-level algorithms use a hierarchy of meshes that are related by 2 : 1 refinement factors. It will also be convenient to use ℓ itself to denote the resolution associated with a particular finite difference approximation, or a transfer operator. That is we define u^ℓ via

$$u^\ell = u^{h_\ell} \tag{177}$$

and could then rewrite (173) as

$$u^{\ell+1} = \bar{I}_\ell^{\ell+1} u^\ell \tag{178}$$

Having made these caveats and definitions, we can now write down pseudo-code for our general multi-level technique:

```
for $\ell = 1 \,,\, \ell_{\max}$
  if $\ell = 1$ then
    $u^\ell := 0$
  else
    $u^\ell := \bar{I}_{\ell-1}^{\ell} u^{\ell-1}$
  end if
  solve_iteratively($u^\ell$)
end for
```

Note that we (arbitrarily) use 0 for the initial estimate of the coarsest-grid problem ($\ell = 0$). Presumably, the solution of this problem is so cheap relative to the finest-grid problem ($\ell = \ell_{\max}$), that this specific choice is essentially irrelevant.

This is a very simple and intuitive algorithm, and when novices hear the term multi-grid, they sometimes think that this sort of approach is all that is involved. However, we shall shortly see that the grid hierarchy that appears in a multi-grid algorithm is also used for an entirely different, and even more essential purpose!

4) How fast can we solve $L^h u^h = f^h$?

The answer to this question is short and sweet. A properly constructed multi-grid algorithm can generally solve a discretized elliptic equation in $O(N)$ operations! This is, of course, computationally optimal, and is the principal reason that multi-grid methods have become such an important tool for numerical analysts—numerical relativists included.

Although the workings of a multi-grid algorithm are relatively complex, particularly when compared to a relaxation technique, the basic idea behind the operation of the algorithm can be stated rather succinctly. As emphasized above, although relaxation is usually quite a bad technique for *solving* discretized elliptic systems, it is generally a very good technique for *smoothing* such systems. Multi-grid combines this fact with the use of a hierarchy of meshes—*using the coarse meshes in the hierarchy to effectively and cheaply annihilate smooth components of the residual and error.* These are precisely the components on the fine mesh which relaxation is slow to converge.

2.7.2 Multi-grid for Linear Problems (LCS Scheme)

We will first discuss a multi-grid algorithm for linear problems—the so called *Linear Correction Scheme* or LCS. For the most part, the development will apply to essentially any elliptic system, but for specificity, the reader may find it useful to think in terms of the two dimensional model problem discussed above.

Before proceeding to a description of the LCS algorithm, we note that multi-grid algorithms generally involve the solutions of sets of equations of the form

$$L^\ell u^\ell = s^\ell \tag{179}$$

where the "right-hand-side" or "source" function, s^ℓ, may *not* necessarily coincide with the source function $f^{\ell_{\max}}$ that appears in the specification of the fine-grid problem

$$L^{\ell_{\max}} u^{\ell_{\max}} = f^{\ell_{\max}} \tag{180}$$

We also note that in *all* of the discussion that follows, we will essentially ignore the role of (discretized) boundary conditions in the operation of multi-grid techniques. However, the reader should be aware that, especially for non-Dirichlet conditions, the treatment of boundary conditions in the context of multi-grid is non-trivial. Fortunately, most of the general references for multi-grid supplied in the bibliography address this issue in considerable detail.

We begin our discussion of the LCS scheme assuming that our grid hierarchy has precisely two grids: a fine grid with mesh spacing h, on which we wish to solve the problem at hand

$$L^h u^h = f^h\,, \tag{181}$$

and a coarse grid with mesh spacing $2h$ which will serve as the "accelerator" of the fine grid solution process. As usual, we denote by $\tilde{u}^h$ the current approximation to the exact solution, u^h, of the system (181). Given some initial estimate of $\tilde{u}^h$, we start by applying relaxation to (181), and for concreteness (as well as the fact that it is commonly used in practice), we will assume that we are using Gauss-Seidel relaxation with red-black ordering (RBGS). Then, after a few sweeps (perhaps as few as 1 or 2!), we will find that the residual, $\tilde{r}^h$

$$\tilde{r}^h := L^h \tilde{u}^h - f^h \tag{182}$$

as well as the error

$$\tilde{e}^h := u^h - \tilde{u}^h \tag{183}$$

will be *smooth* on the scale of the mesh.

At this point, we now think of the task of completing the solution of (181), or, equivalently, of driving the residual, $\tilde{r}^h$, to 0, in terms of computing a *correction*, v^h, such that

$$u^h = \tilde{u}^h + v^h \tag{184}$$

Clearly, due to the linearity of L^h, the correction satisfies

$$L^h v^h = -\tilde{r}^h \tag{185}$$

Now, by assumption, the residual, $\tilde{r}^h$ is smooth and, therefore so must be the sought correction, v^h (this reasonably assumes that L and L^h are "well-behaved").

The first key observation underlying the LCS is that this smoothness in the residual and correction means that we can *sensibly* pose a coarse-grid version of (185). That is, we now consider the coarse grid problem

$$L^{2h} v^{2h} = -I_h^{2h} \tilde{r}^h \equiv s^{2h}\,, \tag{186}$$

where L^{2h} is the coarse-grid difference operator (which employs the same FDA as L^h) and I_h^{2h} is a fine-grid-to-coarse-grid transfer operator known as a *restriction operator* (we will discuss details of this operator later on).

We now assume that we have in some fashion computed an (approximate) solution, $\tilde{v}^{2h}$, of (186). We then update $\tilde{u}^h$ via

$$\tilde{u}^h := \tilde{u}^h + I_{2h}^h \tilde{v}^{2h} \tag{187}$$

where I_{2h}^h is a coarse-to-fine transfer operator, i.e. a prolongation operator, which is not necessarily the same as the prolongation operator $\bar{I}_{2h}^h$ discussed previously. Again, however, I_{2h}^h will typically involve polynomial interpolation, and the interpolation of the computed correction, $\tilde{v}^{2h}$, to the fine grid will generically introduce new high-frequency components in both the residual and the solution error. However, these high-frequency components can be effectively annihilated with a few more relaxation sweeps, at which point, *all* components of the residual will have been substantially reduced. The sequence of posing (186), solving it, then updating the fine grid unknown using (187) is known as a coarse-grid correction (CGC).

The next key observation is that we can apply the smooth/CGC/smooth process to solve the coarse grid problem (186) itself, and then keep recursing, solving problems with mesh spacings, $4h$, $8h$, etc. until on the *coarsest* level, we have a problem which is so small (perhaps 3×3 unknowns, including boundary values!) that it is computationally *trivial* to *solve* the problem (not just smooth it) using relaxation. The solution of the coarse grid problem is then followed by a succession of coarse-to-fine transfers of the various coarse grid corrections that have been computed, with post-CGC relaxation sweeps applied after each restriction operation to ensure that high-frequency components (re)-introduced by the restriction are effectively liquidated. The entire process of working from the finest to coarsest grid then back to the finest grid is known

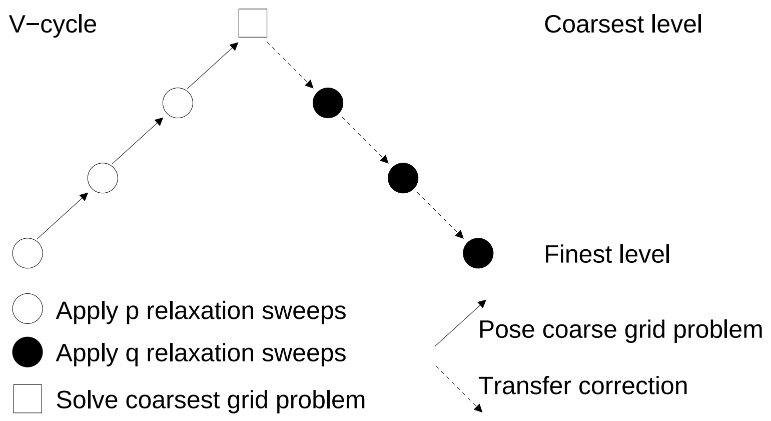
V−cycle
Coarsest level
Finest level
Apply p relaxation sweeps
Apply q relaxation sweeps
Solve coarsest grid problem
Pose coarse grid problem
Transfer correction

Figure 12: Schematic representation of a LCS multi-grid V-cycle, for the case of 4-level mesh hierarchy. (Since the figure has the coarsest level on top, one might more properly refer to it as an "inverted V-cycle".) Open and filled circles represent pre- and post-CGC relaxation sweeps, respectively while the open square denotes the *solution* of the coarse-grid problem. Intergrid transfers are represented by the two different types of arrows.

as a *multi-grid V-cycle* and is schematically illustrated in Figure 12. An important question that arises is: How many relaxation sweeps should be applied before and after each coarse-grid correction? The answer to this generally depends on the specifics of the problem, but in many situations one can parametrize the number of sweeps, with parameters p and q:

$$p \equiv \text{number of pre-CGC sweeps} \tag{188}$$

$$q \equiv \text{number of post-CGC sweeps} \tag{189}$$

and then set p and q to specific values. This results in what is known as a *fixed V-cycle* algorithm, and pseudo-code for this method will be presented shortly.

Turning now to the question of convergence of the multi-grid process, experience shows that, assuming the initial estimate of $\tilde{u}^h$ was good (as should be the case if we are using a multi-level strategy), then we will find that

$$\|\tilde{r}^h\| \sim \|\tau^h\| \tag{190}$$

after a single V-cycle, and our fine grid problem will be effectively solved.

Should we wish for some reason to reduce the norm of the residual further, then we can apply additional V-cycles. Indeed, we can associate a residual amplification matrix, G_V, with the entire V-cycle process. For a well-constructed LCS scheme we will typically find

$$\rho\left(G_V\right) \leq 0.1 \tag{191}$$

and thus we can reasonably expect the residual norm to decrease by an order of magnitude or more per V-cycle. Another crucial point is that, in stark contrast to the case of relaxation, the spectral radius of the

V-cycle, $\rho(G_V)$ is *independent* of the mesh spacing h. This fact is essential to the computational optimality of the multi-grid process. Finally, we observe that in the case of multiple V-cycles, there is no need to perform pre-CGC smoothing sweeps on the finest level for any but the *first* of the V-cycles.

We now present a pseudo-code version of a fixed V-cycle multi-grid algorithm, using two pseudo-procedures: one which implements the V-cycle *per se*, and the other which uses a multi-level approach to "drive" the V-cycle procedure. We will then discuss some details of appropriate coarse-to-fine and fine-to-coarse transfer operators for the specific case where red-black Gauss-Seidel is used as a smoother, and the problem being treated is similar to (or identical to) our model problem.

```
procedure lcs_vcycle ( $\ell$, cycle, $p$, $q$ )
  Cycle from fine to coarse levels
  do $m = \ell, 2, -1$
    if cycle $= 1$ or $m \neq \ell$ then
      Apply pre-CGC smoothing sweeps
      do $p$ times
        $u^m :=$ relax_rb ( $u^m$, $s^m$, $h^m$ )
      end do
      Set up coarse grid problem
      $u^{m-1} := 0$
      $s^{m-1} := -I_m^{m-1}\,(L^m u^m - s^m)$
    end if
  end do
  Solve coarsest-level problem
  $u^1 :=$ relax_rb ( $u^1$, $s^1$, $h^1$ ) until $\|\tilde{r}^1\| \leq \epsilon_1$
  Cycle from coarse to fine levels
  do $m = 2, \ell, +1$
    Apply coarse-grid correction
      $u^m := u^m + I_{m-1}^m u^{m-1}$
    Apply post-CGC smoothing sweeps
    do $q$ times
      $u^m :=$ relax_rb ( $u^m$, $s^m$, $h^m$ )
    end do
  end do
end procedure
```

Note that in the above pseudo-code we have introduced an additional parameter, ϵ_1, which is the convergence criterion for the coarsest-grid problem, which is to be *solved* (rather than simply smoothed) by relaxation. Again, because the cost of relaxation on the coarsest grid is typically so small compared to the work expended on fine grids, ϵ_1 can typically be set quite conservatively (i.e. several orders of magnitude below the typical level of truncation error on such a grid) with no significant loss in overall performance of the algorithm.

```
procedure lcs_mg ( $\ell_{\max}$, ncycle, $p$, $q$ )
  do $\ell = 1, \ell_{\max}, +1$
    if $\ell = 1$ then
      On coarsest level, arbitrarily initialize solution to 0
      $u^1 := 0$
    else
      Initialize solution via prolongation from coarse grid solution
      $u^\ell := \bar{I}_{\ell-1}^{\ell} u^{\ell-1}$
    end if
    Initialize source function
    $s^\ell := f^\ell$
    Perform ncycle V-cycles
    do cycle $= 1$ , ncycle
```

```
            lcs_vcycle( $\ell$, cycle, $p$, $q$ )
        end do
    end do
end procedure
```

Thus far we have not discussed the nature of the grid-to-grid transfer operators (i.e. the prolongation and restriction operators) in any detail. The proper construction of $\bar{I}^{\ell}_{\ell+1}$, $I^{\ell}_{\ell+1}$, and $I^{\ell+1}_{\ell}$ is an extremely important aspect of any multi-grid algorithm, and again, it is beyond the scope of these lectures to treat this topic in any depth. We note, however, that the multi-grid technique (i.e. the V-cycle) induces non-trivial interactions between these operators and the relaxation (smoothing) scheme, so that for example, prolongation and restriction operators that work well with red-black Gauss-Seidel relaxation do not necessarily work well when lexicographic Gauss-Seidel is used.

Here we will describe specific implementations for these operators that work well for problems such as our model equation—as well as similar systems in two and three dimensions—in conjunction with red-black Gauss-Seidel smoothing.

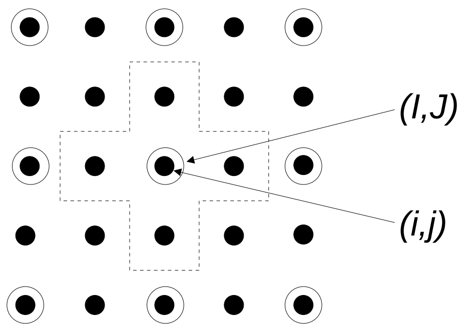

Figure 13: Schematic representation of the stencil used for the operation of half-weighted restriction. Small filled and large open circles represent fine and coarse grid locations respectively, and the task of the restriction operator is to define all of the coarse grid values from the fine grid unknowns. The dotted line indicates the 5-pt stencil that is used in the half-weighted stencil; see the text for full details.

We begin with the restriction operator, $I^{\ell-1}_{\ell}$ (I^{2h}_{h}), which is used in the LCS to transfer the right-hand-side of the equation for the correction from the fine to the coarse grid. Experience (which can be backed up with rigorous analysis) has shown that *half-weighted restriction* works well in this case. To define this operator, we appeal to Figure 13, which depicts a portion of a both the fine grid (small filled circles) and coarse grid (large open circles), overlaid on one another. Indices (i,j) and (I,J) label the fine and coarse grids respectively, and the task of the restriction operator is to define *all* of the coarse grid values from the fine grid quantities. Let as assume, as shown in the picture, that the *specific* indices (i,j) and (I,J) correspond to the same physical location. Then the following formula defines half-weighted restriction in two dimensions:

$$u^{\ell}_{I,J} = \frac{1}{2}u^{\ell-1}_{i,j} + \frac{1}{8}\left(u^{\ell-1}_{i-1,j} + u^{\ell-1}_{i+1,j} + u^{\ell-1}_{i,j+1} + u^{\ell-1}_{i,j-1}\right) \qquad (192)$$

The term "half-weighting" comes from the fact that the fine-grid unknown, $u^{\ell-1}_{i,j}$, located at the same physical location as the target coarse-grid unknown, $u^{\ell}_{I,J}$ has a weight of 1/2 in the transfer equation (192). (Note that the sum of all of the "weights" associated with the fine grid unknowns is 1, as befits a discrete representation of an identity operator). The half-weighted restriction operator may be analogously defined in 1 and 3 dimensions, where the factor of 1/8 is replaced by 1/4 and 1/12, respectively, and the sum is over the central fine-grid unknown, and its 2 and 6 nearest neighbors respectively.

We note in passing that the "obvious" restriction formula

$$u^{\ell}_{I,J} = u^{\ell-1}_{i,j} \qquad (193)$$

which is known as *injection* fails miserably when used with red-black Gauss-Seidel relaxation, although it tends to work well when lexicographically-ordered GS is employed as the smoother.

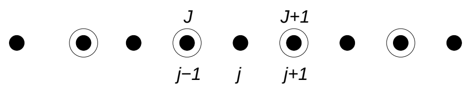

Figure 14: Illustration of coarse and fine meshes in one dimension for the purpose of linear interpolation (prolongation), of coarse grid values to the fine grid. Refer to the text for details.

Turning now to the prolongation operator, $I^{\ell}_{\ell-1}$ (I^h_{2h}), which the LCS uses to transfer the coarse grid correction back to the fine grid, we note that we have already mentioned that such operators typically employ polynomial interpolation of some sort. Again, in the context of the algorithm pseudo-coded above, and as applied to our model problem and related, *bilinear interpolation* is found to work well. Before proceeding to the details of bilinear interpolation, we consider the analogous problem in one dimension, namely linear interpolation from a mesh Ω^h to Ω^{2h}. Figure 14 illustrates a portion of both the coarse-grid (large open circles) with grid index, J, and the fine-grid, with grid index, j. The fine-grid values u^{ℓ}_{j-1} and u^{ℓ}_{j+1} are trivially interpolated as "copies" of the corresponding coarse-grid unknowns

$$u^{\ell}_{j-1} = u^{\ell-1}_{J} \tag{194}$$

$$u^{\ell}_{j+1} = u^{\ell-1}_{J+1} \tag{195}$$

and, intuitively, the other half of the fine-grid quantities can be linearly interpolated by "averaging" the neighboring values: that is, assuming that u^{ℓ}_{j-1} and u^{ℓ}_{j+1} have been defined as above, then u^{ℓ}_{j} is given by

$$u^{\ell}_{j} = \frac{1}{2}\left(u^{\ell}_{j-1} + u^{\ell}_{j+1}\right) \tag{196}$$

We can establish this result more formally by considering Taylor series expansions in the mesh spacing, h, about the grid point, $x = x_j$. We have

$$u_{j+1} = u_j + hu'_j + \frac{1}{2}h^2u''_j + O(h^2) \tag{197}$$

$$u_{j-1} = u_j - hu'_j + \frac{1}{2}h^2u''_j + O(h^2) \tag{198}$$

so that

$$\frac{1}{2}\left(u_{j+1} + u_{j-1}\right) = u_j + \frac{1}{2}h^2u''_j + O(h^4) \tag{199}$$

$$= u_j + O(h^2) \tag{200}$$

Thus the averaging formula *does* yield an $O(h^2)$ approximation to the values that actually need to be interpolated (rather than simply "copied") and this is precisely what is meant by linear interpolation in this context. Moreover, we note that (196) is the *unique* formula for linear interpolation of u^{ℓ}_{j} from its nearest neighbors u^{ℓ}_{j-1} and u^{ℓ}_{j+1}.

We now return to the two dimensional case and refer to Figure 15, which again shows a portion of both the coarse grid (large open circles) and fine grid (small circles). We see that there are generally three types of values that must be defined on the fine grid via interpolation process: values which may be copied from the coarse grid, values that can be determined via one-dimensional linear interpolation in either the x or y direction, and, finally, values that genuinely require two-dimensional interpolation. Such points have precisely 4 nearest neighbors in the coarse grid, but, as can again be verified via Taylor series (as well as elementary geometry—how many distinct, non-colinear points are needed to define a plane?), only 3 of these are required to produce a linear interpolant. Thus, in the two dimensional case (as well as in higher dimensions) there is no *unique* formula for linear interpolation from Ω^{2h} to Ω^h. Figure 16 schematically illustrates two of the possible formulae that we have at our disposal and it is easy to convince oneself that there are actually an infinite number of possible formulae.

One would hope that the precise details of the interpolation—i.e. which specific formula is used—would not matter, and indeed, experience shows that this is the case. To summarize then, we are free to implement the

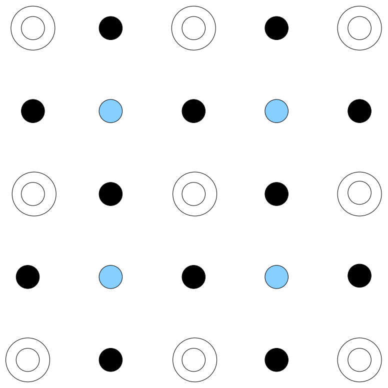

Figure 15: Illustration of a portion of a coarse mesh (large open circles) and a fine mesh (small circles), in two dimensions, and in the context of bi-linear interpolation. There are three distinct types of fine grid points: (1) those whose values are copies of coarse grid unknowns (small open circles), (2) those whose values may be computed via one-dimensional interpolation in either the x or y direction (dark filled circles), and (3) those whose values require genuine two-dimensional interpolation (light filled circles).

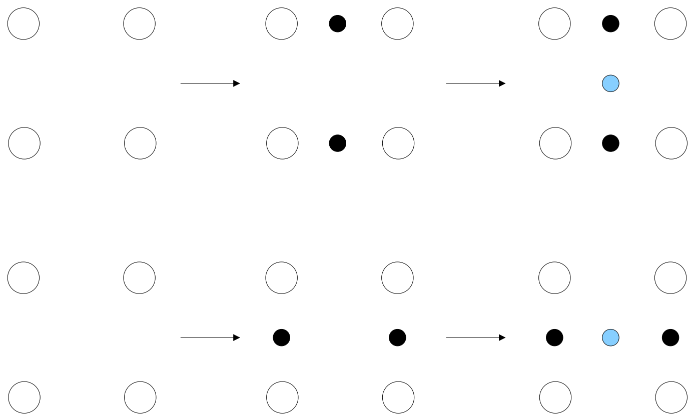

Figure 16: Schematic illustration of two of the infinite number of ways of computing the "central" fine grid value (light filled circle) from the coarse grid using bi-linear interpolation.

interpolation operator more or less as we please, and Figure 17 illustrates an approach with is particularly convenient to implement, and which extends to more dimensions, as well as to higher order interpolation. The last transfer operator that we must consider is the prolongation operator, $\tilde{I}^{\ell}_{\ell-1}$ which is used in the multi-level solution process to initialize a fine-grid unknown from the solution of a coarse grid problem. Again, there has been considerable research on this subject over the years, and, in particular, an old rule of thumb due to Brandt would ostensibly suggest that we use *bi-cubic* interpolation, in the coarse grid mesh, so that the leading order error term would be $O(h^4)$. However, empirically one finds that *bi-linear* interpolation actually provides better overall performance (again, in the context of our model problem being smoothed with red-black Gauss-Seidel, and with the other transfer operators defined as above), so that in the current case, we have

$$\tilde{I}^{\ell}_{\ell-1} \equiv I^{\ell}_{\ell-1} \tag{201}$$

2.7.3 Computational Cost of Multi-Grid

We now consider the computational cost of a V-cycle-based multi-grid algorithm, again illustrated for the case of a two dimensional problem. We first define the following quantities

$$w_\ell \quad \equiv \quad \text{work required to perform one level } \ell \text{ relaxation sweep} \tag{202}$$

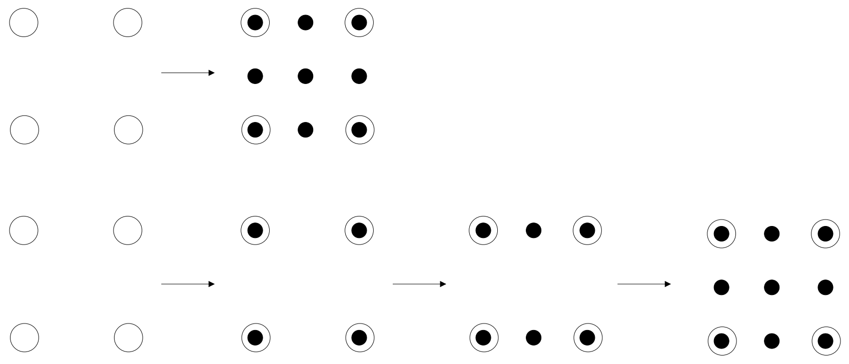

Figure 17: Schematic illustration of a particularly convenient strategy for implementing 2:1 bi-linear interpolation, which generalizes to higher dimensions as well as higher order interpolation.

$$W_\ell \equiv \text{work required to } \textit{solve} \text{ the level } \ell \text{ problem} \tag{203}$$
$$p \equiv \text{number of pre-CGC sweeps} \tag{204}$$
$$q \equiv \text{number of post-CGC sweeps} \tag{205}$$
$$\sigma \equiv \text{number of CGCs needed per solve} \tag{206}$$

Then we have

$$W_\ell \sim (p+\sigma q)\, w_\ell + \sigma W_{\ell-1} \tag{207}$$
$$= (p+\sigma q) + \sigma\left((p+\sigma q)\, w_{\ell-1} + \sigma W_{\ell-2}\right) \tag{208}$$

But (again, in two dimensions)

$$w_{\ell-1} \sim \frac{1}{4} w_\ell \tag{209}$$

so

$$W_\ell \sim (p+\sigma q)\, w_\ell \left(1 + \frac{1}{4}\sigma + \frac{1}{16}\sigma^2 + \cdots + \left(\frac{\sigma}{4}\right)^{\ell-2}\right) + \sigma^{\ell-1} W_1 \tag{210}$$

where W_1 is the work required to *solve* the coarsest-grid problem $L^1 u^1 = s^1$. Now, so long as the number of CGCs, σ, required to solve any fine grid problem, satisfies $\sigma < 4$, then we have

$$1 + \frac{1}{4}\sigma + \frac{1}{16}\sigma^2 + \cdots + \left(\frac{\sigma}{4}\right)^{\ell-2} < \left(1 - \frac{\sigma}{4}\right)^{-1} \tag{211}$$

and thus

$$W_\ell \le w_\ell \left(\frac{p+\sigma q}{1-\sigma/4}\right) + \sigma^{\ell-1} W_1 \tag{212}$$

But, the work, w_ℓ, required to perform a level-ℓ relaxation sweep is proportional to the number of grid points, N_ℓ, in the level-ℓ mesh:

$$w_\ell \sim c N_\ell \tag{213}$$

where c is some constant, and by our assumption that $\sigma < 4$, we have

$$\sigma^{\ell-1} < 4^{\ell-1} \sim \frac{N_{\ell-1}}{N_1} \tag{214}$$

where N_1 is the number of points on the coarsest grid (another constant). Putting these results together we have

$$W_\ell \le N_\ell \left(\frac{c\,(p+\sigma q)}{1-\sigma/4} + \frac{W_1}{N_1}\right) = O(N_\ell) \tag{215}$$

Thus, as previously claimed, multi-grid can solve the N algebraic equations resulting from the discretization of elliptic PDEs in $O(N)$ time. Heuristically, this result can be understood from the following basic observations

1. The cost of a multi-grid algorithm is dominated by the cost of relaxation work on the *finest grid*; because of the geometrically decreasing number of grid points on coarse grids, the cost of the CGCs is largely negligible (and increasingly so for higher dimensional problems)
2. The role of fine grid relaxation is simply to *smooth* the fine grid problem, *not* to solve it
3. A properly choosen smoother (relaxation technique) is h-independent and costs $O(N)$
4. Thus, the *work per fine grid point* expended for the entire multi-grid solution is also h-independent and $O(N)$

2.7.4 Multi-grid for Non-Linear Problems (FAS Scheme)

We now turn our attention to the application of the multi-grid scheme to non-linear problems, using the so-called *Full Approximation Storage Scheme.* Let us first quickly re-cap the Linear Correction Scheme. There, we wished to solve

$$L^h u^h - f^h = 0\,, \tag{216}$$

and, at any stage of the iterative solution scheme, with the current approximation to u^h being denoted by $\tilde{u}^h$, attention focused on the residual, $\tilde{r}^h$ defined by

$$L^h \tilde{u}^h - f^h = \tilde{r}^h \tag{217}$$

Assuming that $\tilde{r}^h$ was smooth, we sought a correction, v^h satisfying

$$u^h = \tilde{u}^h + v^h\,, \tag{218}$$

and computed an approximation to that correction by posing and solving the coarse-grid problem

$$L^{2h} u^{2h} = -I_h^{2h} \tilde{r}^h\,, \tag{219}$$

and then updating the fine grid unknown using

$$\tilde{u}^h := \tilde{u}^h + I_{2h}^h v^{2h}\,. \tag{220}$$

However, in order to derive this scheme, we had to take advantage of the *linearity* of L^h by using

$$L^h\left(\tilde{u}^h + v^h\right) = L^h \tilde{u}^h + L^h v^h \tag{221}$$

Clearly, then, the LCS will not work if L^h is non-linear. However, suppose that given a non-linear problem (i.e. a non-linear L^h), that we can still *smooth* the residual, $\tilde{r}^h$, using a few (non-linear) relaxation sweeps. Then, we can again view the solution of the discrete system in terms of computing a *smooth* correction, v^h, such that

$$L^h\left(\tilde{u}^h + v^h\right) = f^h \tag{222}$$

In particular, consider the following expression obtained by subtracting the ostensibly smooth quantity, $L^h\tilde{u}^h$, from both sides of the above equation (each of which is also smooth, by assumption):

$$L^h\left(\tilde{u}^h + v^h\right) - L^h\tilde{u}^h = f^h - L^h\tilde{u}^h = -\tilde{r}^h \tag{223}$$

Provided $\tilde{r}^h$ *is* smooth, then we can sensibly pose a coarse grid version of (223), namely

$$L^{2h}u^{2h} - L^{2h}I_h^{2h}\tilde{u}^h = -I_h^{2h}\tilde{r}^h \tag{224}$$

where I_h^{2h} is a suitably constructed restriction operator. We can rewrite (224) as an equation for the coarse grid unknown, u^{2h}:

$$L^{2h}u^{2h} = L^{2h}I_h^{2h}\tilde{u}^h - I_h^{2h}\tilde{r}^h \tag{225}$$

Note that, here, u^{2h} is the coarse-grid representation of $\tilde{u}^h + v^h$; that is, on the coarse grid, we manipulate a "full approximation" to the desired solution, u^h, rather than the correction, v^h, itself, hence the terminology "Full Approximation Storage".

Now, as was the case with the LCS, we proceed to solve the coarse-grid problem (225). Having done so, we then update the fine grid unknown using

$$\tilde{u}^h := \tilde{u}^h + I^h_{2h}\left(u^{2h} - I^{2h}_h \tilde{u}^h\right) \tag{226}$$

where I^h_{2h} is an appropriate prolongation operator, which again, typically performs some sort of polynomial interpolation in the coarse grid.

We observe that the fine grid update formula (226) is *not* equivalent to the more obvious

$$\tilde{u}^h := I^h_{2h} u^{2h}\,, \tag{227}$$

since, in general

$$I^h_{2h} I^{2h}_h \neq \mathbf{1} \tag{228}$$

(rank deficiency). The former expression is to be preferred since, quoting/paraphrsasing Brandt, it "retains (useful) high frequency information contained in $\tilde{u}^h$ prior to the coarse grid correction. As with the LCS, the interpolation of the coarse-grid correction will tend to introduce new high frequency components in the fine-grid residual and error. However, these can again be effectively annihilated provided that a non-linear relaxation process with good smoothing properties is in hand.

Equations (225) and (226) are the basic equations for the FAS Coarse Grid Correction, and although we could proceed to a pseudo-code description of a typical FAS algorithm at this point, it is instructive to consider an alternate derivation of (225) that reveals another way of viewing that FAS multi-grid algorithm.

Recall once more our basic trio of equations for the continuum solution, u, the corresponding discrete solution, u^h, and the associated truncation error, τ^h, respectively:

$$Lu = f \tag{229}$$

$$L^h u^h = f^h \tag{230}$$

$$\tau^h \equiv L^h u - f^h \tag{231}$$

Observe that we can re-write (231) as

$$L^h u = f + \tau^h \tag{232}$$

so that, as a moment's reflection makes clear, the truncation error is the quantity that we need to add to the right hand side of (230), so that the solution of the corresponding *discrete* equations actually coincide with the values of the *continuum* solution evaluated on the mesh.

In general, of course, we *cannot* compute τ^h exactly. But suppose we had an approximation, $\tilde{\tau}^h$ of τ, that we had calculated in some fashion. Then, by solving

$$L^h u^h_\star = f^h + \tilde{\tau}^h \tag{233}$$

we would get an *improved* approximation, $u^h_\star$, relative to u^h.

Now, equation (231) can also be rewritten as

$$\tau^h = L^h u - L^h u^h \tag{234}$$

We now re-introduce the coarse ($2h$) mesh and consider by analogy to (234) a quantity τ^{2h}_h defined by

$$\tau^{2h}_h \equiv L^{2h} I^{2h}_h u^h - I^{2h}_h L^h u^h \tag{235}$$

We call τ^{2h}_h the *relative (local) truncation error*, defined on the coarse grid (mesh spacing $2h$) relative to the fine grid (mesh spacing h). The appropriateness of this nomenclature becomes clear when we re-write (235) as

$$L^{2h} I^{2h}_h u^h = I^{2h}_h L^h u^h + \tau^{2h}_h \tag{236}$$

Identifying $I_h^{2h} u^h$ with a coarse-grid unknown u^{2h}, and using $L^h u^h = f^h$, this last result becomes

$$L^{2h} u^{2h} = f^{2h} + \tau_h^{2h} \tag{237}$$

so, in direct analogy to the interpretation of τ^h made above, we can think of τ_h^{2h} as the correction to the source (or "forcing") terms of the coarse grid equations that makes the solution of those coarse grid equations *coincide with the solution of the fine grid equations.*

Again, in general we cannot compute τ_h^{2h} exactly. However, given some approximate solution, $\tilde{u}^h$, of the fine grid equations, we *can* define an approximation, $\tilde{\tau}_h^{2h}$, to it using the following formula

$$\tilde{\tau}_h^{2h} = L^{2h} I_h^{2h} \tilde{u}^h - I_h^{2h} L^h \tilde{u}^h \tag{238}$$

Substituting the right hand side of (238) for τ_h^{2h} in (236), we find

$$\begin{aligned} L^{2h} u^{2h} &= f^{2h} + \tilde{\tau}_h^{2h} = f^{2h} + L^{2h} I_h^{2h} \tilde{u}^h - I_h^{2h} L^h \tilde{u}^h & (239) \\ &= L^{2h} I_h^{2h} \tilde{u}^h - I^{2h} \left(L^h \tilde{u}^h - f^h \right) & (240) \\ &= L^{2h} I_h^{2h} \tilde{u}^h - I^{2h} \tilde{r}^h & (241) \end{aligned}$$

This last expression is precisely the first of our two basic equations for the FAS coarse-grid correction process, equation (225).

In view of the above development, and as emphasized by Brandt, we are thus lead to the following *dual* views of the multi-grid process:

1. Coarse grids are used to accelerate the convergence of low-frequency components of fine grid residuals (and solution errors)

2. Fine grids are used to compute correction terms to coarse grid equations (through modification of the source/forcing terms), yielding fine-grid accuracy on the coarse grids

Equations (237) and (238) (with $\tilde{tau}_h^{2h}$ replacing τ_h^{2h} in (237) are equivalent to (225), and it is this form that I typically use when implementing an FAS multi-grid algorithm.

One significant rationale for this approach is that it provides an easy route to the estimation of the true truncation error in the difference scheme. As discussed above, the norm of the truncation error, $\|\tau^h\|$ provides a natural stopping criterion for the overall iteration; that is, it is natural to stop the iterative solution of the difference equations when

$$\|\tilde{r}^h\| \sim \|\tau^h\| \tag{242}$$

One can show that for the case that τ^h is $O(h^2)$, then

$$\tau^h \sim \frac{1}{3} \tau_h^{2h} \sim \tilde{\tau}_h^{2h} \tag{243}$$

Thus, by computing and monitoring the relative truncation error, $\tilde{\tau}_h^{2h}$, we can quite easily decide when it is time to terminate the multi-grid process.

Finally, in this context, we note that the computation of $\tilde{\tau}_h^{2h}$, together with relations such as (243), provide the basis for so-called *tau-extrapolation* in which one attempts to increase the accuracy of a discrete solution, u^h, by solving

$$L^h u^h = f^h + \tilde{\tau}^h \tag{244}$$

Before proceeding to a pseudo-code version of an FAS V-cycle we need to discuss the topic of smoothing of residuals via non-linear relaxation. Fortunately, for many "typical" non-linear problems (posed in cartesian coordinates, and having smoothly varying, well-scaled coefficients), 1-step-Newton red-black Gauss-Seidel is a good smoother.

We illustrate this technique using the following non-linear elliptic equation:

$$\nabla u(x, y) + \sigma u^2 = f(x, y) \tag{245}$$

where σ is a constant, $f(x, y)$ is a specified function, and we eschew discussion of the domain and boundary conditions since they are not relevant to the description of the relaxation technique. We discretize (245) in the usual fashion, and write the resulting non-linear equations in the canonical form:

$$F\left[u_{i,j}\right] = 0\,. \tag{246}$$

Specifically, we have

$$F\left[u_{i,j}\right] \equiv \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} + \sigma u_{i,j} - f_{i,j} = 0 \tag{247}$$

We note that by defining our equations in this canonical form, the application of F to the instantaneous approximation, $\tilde{u}^h$ of the difference equations is simply the residual, $\tilde{r}^h$

$$\tilde{r}^h \equiv F\left[\tilde{u}^h\right] \tag{248}$$

The Newton Gauss-Seidel iteration can then be defined via

$$\tilde{u}_{i,j} := \tilde{u}_{i,j} - F\left[\tilde{u}_{i,j}\right] \left(\frac{\partial F\left[u_{i,j}\right]}{\partial u_{i,j}} \right)^{-1} \Bigg|_{u_{i,j} = \tilde{u}_{i,j}} \tag{249}$$

In the current example we have

$$\frac{\partial F\left[u_{i,j}\right]}{\partial u_{i,j}} = -4h^{-2} + 2\sigma u_{i,j} \tag{250}$$

Because the role of the relaxation is again to *smooth* the discrete system of equations, rather than to solve it (except on the coarsest level), one generally takes only a single Newton step per visit of a grid point; i.e. one does *not* attempt to instantaneously solve the local equation as is the case for linear relaxation.

Again, for many "well behaved" non-linear elliptic systems, many other details of the FAS algorithm, including the basic structure of the V-cycle, as well as appropriate choices for the prolongation and restriction operators carry over from the LCS. We thus conclude this section with a pseudo code version of an FAS V-cycle; the pseudo-code for the corresponding driver routine (i.e. `fas_mg`) is essentially identical to that for `lcs_mg`

```
procedure fas_vcycle ( $\ell$, cycle, $p$, $q$ )
   Cycle from fine to coarse levels
   do $m = \ell, 2, -1$
      if cycle = 1 or $m \neq \ell$ then
         Apply pre-CGC smoothing sweeps
         do $p$ times
            $u^m :=$ relax_rb ( $u^m$, $s^m$, $h^m$ )
         end do
         Set up coarse grid problem
         $\tau^{m-1}_m := L^{m-1} I^{m-1}_m u^m - I^{m-1}_m L^m u^m$
         $s^{m-1} := \tau^{m-1}_m + I^{m-1}_m s^m$
         $u^{m-1} := I^{m-1}_m u^m$
      end if
   end do
   Solve coarsest-level problem
   $u^1 :=$ relax_rb ( $u^1$, $s^1$, $h^1$ ) until $\|\tilde{r}^1\| \leq \epsilon_1$
   Cycle from coarse to fine levels
   do $m = 2, \ell, +1$
      Apply coarse-grid correction
         $u^m := u^m + I^m_{m-1}\left(u^{m-1} - I^{m-1}_m u^m\right)$
      Apply post-CGC smoothing sweeps
      do $q$ times
```

```
        $u^m$ := relax_rb( $u^m$, $s^m$, $h^m$ )
      end do
    end do
  end procedure
```

References

[1] Mitchell, A. R., and D. F. Griffiths, **The Finite Difference Method in Partial Differential Equations**, New York: Wiley (1980)

[2] Richtmeyer, R. D., and Morton, K. W., **Difference Methods for Initial-Value Problems**, New York: Interscience (1967)

[3] H.-O. Kreiss and J. Oliger, **Methods for the Approximate Solution of Time Dependent Problems**, GARP Publications Series No. 10, (1973)

[4] Gustatsson, B., H. Kreiss and J. Oliger, **Time-Dependent Problems and Difference Methods**, New York: Wiley (1995)

[5] Richardson, L. F., "The Approximate Arithmetical Solution by Finite Differences of Physical Problems involving Differential Equations, with an Application to the Stresses in a Masonry Dam", *Phil. Trans. Roy. Soc.*, **210**, (1910) 307–357.

[6] Anderson, E. em et al "Lapack Users' Guide - Release 2.0", (1994)
`http://www.netlib.org/lapack/lug/lapack_lug.html`

[7] Brandt, A. 1977. *Math. Comput.* **31**, 333; Brandt, A. 1977: In Rice, J. (Ed), Numerical Software III. Academic Press, New York, pp 277-318; Brandt, A. 1979: In Hemker, PW & Miller, J. J., (Eds), Numerical Analysis of Singular Peturbation Problems. Academic Press, New York, pp 52-146; Brandt, A. 1982: In Hackbusch, W. & Trottenberg, U., (Eds), Lecture Notes in Mathematics: Multi Grid Methods, v 960

[8] Varga, R. S., **Matrix Iterative Analysis**, Springer (Second Ed.), 2000