Chapter 1

Elementary Acquaintance With Multigrid

The following few pages would acquaint you with the conceptual basis of all multigrid solvers, with elementary mode-analysis and with an example of a simple algorithm along with its MATLAB implementation and output.

For a more detailed introduction to multigrid techniques, through a comprehensive treatment of some *model* problems by a variety of multigrid algorithms, mode analyses and numerical experiments, see [ST82]. The latter appears in a book [HT82] which also includes a previous version of the current Guide [Bra82b], a complete (as of 1982) multigrid bibliography, and many other multigrid papers. Additional material, and in fact summaries of all new multigrid papers, appear quarterly in the *MULTIGRID NEWSLETTER*, obtainable in North and South America from its Editor (Steve McCormick, Department of Mathematics, Colorado State University, Fort Collins, CO 80523, U.S.A.) and in other countries from the Managing Editor (Kurt Brand, GMD/FIT, Postfach 1240, D-5205 St. Augustin 1, Federal Republic of Germany). Periodically, it publishes a complete list of all past papers. Also available is a set of videoed multigrid lectures [Bra83].

1.1 Properties of slowly converging errors

The origin of multi-level (multigrid) fast solvers is a certain insight concerning the nature of the algebraic errors that become dominant when conventional iterative schemes are slow to converge. Let us first present this insight in its most general algebraic setting, where a matrix equation $Ax = b$ is being solved for any (possibly rectangular) matrix A.

For any approximate solution $\tilde{x}$, denote by $e = x - \tilde{x}$ the error vector and by $r = Ae = b - A\tilde{x}$ the vector of residuals. The common feature of all iterative schemes is that at each step some corrections to $\tilde{x}$ are calculated based on the magnitude of certain residuals. As a result, convergence must

be slow if the individual residuals do not show the true magnitude of the error, i.e., if r is in some sense small compared with e. The converse is also true: If convergence of a *suitable* relaxation scheme is slow, residuals must in some sense be small compared with e.

To see this more concretely, consider for example Kaczmarz relaxation applied to the above matrix equation and converging to a solution x. Denoting by a_i the ith row of A, the Kaczmarz step corresponding to that row is to replace $\tilde{x}$ by $\tilde{x} + (r_i/a_i a_i^\top)a_i^\top$, thereby forcing r_i to zero. A full Kaczmarz sweep is the employment of such a step for each row of A, in the natural ordering . (We take this scheme as our example because it applies to the most general matrix: It converges whenever a solution exists [Tan71].) Let

$$E = e^\top e = \sum_i e_i^2 \quad \text{and} \quad R = \sum_i \frac{r_i^2}{a_i a_i^\top}$$

be square norms for errors and residuals, respectively, evidently scaled so that they are comparable. One can then prove the following result [Bra86, Theorem 3.4]:

Theorem 1.1. *A Kaczmarz sweep reduces E at least by* $\max\{\gamma_0 R_0, \gamma_1 R_1\}$, *where R_0 and R_1 are the values of R before and after the sweep, respectively, and*

$$\gamma_0 = ((1+\gamma_+)(1+\gamma_-))^{-1}, \qquad \gamma_1 = (\gamma_-\gamma_+)^{-1}$$
$$\gamma_- = \max_i \sum_{j<i} \frac{|a_i a_j^\top|}{a_i a_i^\top}, \quad \gamma_+ = \max_i \sum_{j>i} \frac{|a_i a_j^\top|}{a_i a_i^\top}.$$

The theorem in essence says that *slow convergence can occur only when R is small compared with E* (observe that in case A arises from the discretization of differential equations, γ_i are completely local quantities, independent of the size of A. The above values of γ_0 and γ_1 are close to the best possible ones). Similar theorems (with suitably modified E, R and γ_i) hold for all familiar relaxation schemes, such as Gauss-Seidel, Jacobi, SOR, and block schemes (line relaxation, etc. See [Bra86]).

Because E and R were scaled to be comparable, it is generally only for special types of error components that the slow convergence condition

$$R \ll E \tag{1.1}$$

is satisfied. The deeper (1.1) is satisfied, the more special must the error be, and hence the fewer the number of parameters needed to approximate it. Thus, broadly speaking, *relaxation efficiently reduces the information content of the error, and quickly makes it approximable by far fewer variables.* This can be shown to be true even for nonlinear systems.

When the matrix equation $Ax = b$ is a discretization of a differential system $Lu = f$ on some grid, condition (1.1), rewritten in the form $\|Ae\| \ll \|A\|\|e\|$, can be interpreted as saying that the error e approximates a continuous function v satisfying $\|Lv\| \ll \|L\|\|v\|$, in some corresponding norms. If L is a uniformly elliptic operator, this implies that e is a smooth function (in case L is not elliptic, a certain smoothness along characteristics is at least implied). Hence, *relaxation efficiently reduces non-smooth error components, thus making the error approximable on a coarser grid* (where solution is much less expensive). A precise measure for this efficiency is discussed next.

1.2 Error smoothing and its analysis: Example

For clarity, consider a simple example. Suppose the partial differential equation

$$Lu(x,y) \equiv a\frac{\partial^2 u(x,y)}{\partial x^2} + c\frac{\partial^2 u(x,y)}{\partial y^2} = f(x,y), \qquad (a,c>0) \tag{1.2}$$

is to be solved with some suitable boundary conditions. Denoting by u^h and f^h approximations to u and f, respectively, on a grid with meshsize h, the usual second-order discretization of (1.2) is

$$L^h u^h_{\alpha,\beta} \equiv a\frac{u^h_{\alpha-1,\beta} - 2u^h_{\alpha,\beta} + u^h_{\alpha+1,\beta}}{h^2} + c\frac{u^h_{\alpha,\beta-1} - 2u^h_{\alpha,\beta} + u^h_{\alpha,\beta+1}}{h^2} = f^h_{\alpha,\beta}, \tag{1.3}$$

where

$$u^h_{\alpha,\beta} = u^h(\alpha h, \beta h), \qquad f^h_{\alpha,\beta} = f(\alpha h, \beta h); \qquad \alpha,\beta \text{ integers}$$

(in the multigrid context it is important to define the difference equations in this divided form, without, for example, multiplying throughout by h^2, in order to get the proper relative scale at different grids). Given an approximation $\tilde{u}$ to u^h, a simple example of a relaxation scheme to improve it is the following.

Gauss-Seidel Relaxation. The points (α,β) of the grid are scanned one by one in some prescribed order; e.g., lexicographic order. At each point, the value $\tilde{u}_{\alpha,\beta}$ is replaced by a new value, $\overline{u}_{\alpha,\beta}$, such that (1.3) at that point is satisfied. That is, $\overline{u}_{\alpha,\beta}$ satisfies

$$a\frac{\overline{u}^h_{\alpha-1,\beta} - 2\overline{u}^h_{\alpha,\beta} + \tilde{u}^h_{\alpha+1,\beta}}{h^2} + c\frac{\overline{u}^h_{\alpha,\beta-1} - 2\overline{u}^h_{\alpha,\beta} + \tilde{u}^h_{\alpha,\beta+1}}{h^2} = f^h_{\alpha,\beta}, \tag{1.4}$$

where the new values $\overline{u}^h_{\alpha-1,\beta}, \overline{u}^h_{\alpha,\beta-1}$ are used because, in the lexicographic order, by the time (α,β) is scanned, new values have already replaced old ones at $(\alpha-1,\beta)$ and $(\alpha,\beta-1)$.

A complete pass, scanning in this manner all gridpoints, is called a (Gauss-Seidel lexicographic) *relaxation sweep.* The new approximation $\overline{u}$ does not satisfy (1.3), and further relaxation sweeps may be required to improve it. An important quantity therefore is the *convergence factor* μ, which may be defined by

$$\mu := \frac{\|\overline{v}\|}{\|v\|}, \qquad \text{where} \quad v := u^h - \tilde{u}, \quad \overline{v} := u^h - \overline{u}, \tag{1.5}$$

$\|\cdot\|$ being any suitable discrete norm. For the Gauss-Seidel scheme, with the possible exception of its first few sweeps, $\mu = 1 - O(h^2)$. This means that $O(h^{-2})$ relaxation sweeps are needed to reduce the error by an order of magnitude.

In multigrid methods, however, the role of relaxation is not to reduce the error, but to smooth it out so that it becomes well approximable on a coarser grid. This relaxation can do very effectively. Indeed, subtracting (1.3) from (1.4), the relation

$$a\left(\overline{v}^h_{\alpha-1,\beta} - 2\overline{v}^h_{\alpha,\beta} + v^h_{\alpha+1,\beta}\right) + c\left(\overline{v}^h_{\alpha,\beta-1} - 2\overline{v}^h_{\alpha,\beta} + v^h_{\alpha,\beta+1}\right) = 0 \tag{1.6}$$

shows that $\overline{v}^h_{\alpha,\beta}$ is a weighted average of neighboring values (of both v and $\overline{v}$), so that, if the old error v is not smooth, the new error $\overline{v}$ must be much smoother.

To analyze the smoothing effect of a relaxation sweep quantitatively, we take advantage of its *local* nature (points several meshsizes apart affecting each other exponentially little). It allows us, for the purpose of studying the smoothing well in the interior, to regard the grid as embedded in a rectangular domain. We can then expand both v and $\overline{v}$ in Fourier series

$$v_{\alpha,\beta} = \sum A_{\underline{\theta}} e^{i(\theta_1\alpha+\theta_2\beta)}, \qquad \overline{v}_{\alpha,\beta} = \sum \overline{A}_{\underline{\theta}} e^{i(\theta_1\alpha+\theta_2\beta)}, \tag{1.7}$$

where $\underline{\theta} := (\theta_1, \theta_2)$ and the summations are over a subset of the square $|\underline{\theta}| := \max(|\theta_1|, |\theta_2|) \le \pi$. Substituting (1.7) into (1.6) yields

$$\left(ae^{-i\theta_1} + ce^{-i\theta_2} - 2a - 2c\right) A_{\underline{\theta}} + \left(ae^{i\theta_1} + ce^{i\theta_2}\right) \overline{A}_{\underline{\theta}} = 0. \tag{1.8}$$

Hence, the *amplification factor* of the $\underline{\theta}$ component due to one relaxation sweep is

$$\mu(\underline{\theta}) = \left|\frac{\overline{A}_{\underline{\theta}}}{A_{\underline{\theta}}}\right| = \left|\frac{ae^{i\theta_1} + ce^{i\theta_2}}{2a + 2c - ae^{-i\theta_1} - ce^{-i\theta_2}}\right|. \tag{1.9}$$

Observe that $\mu(\underline{\theta}) \to 1$ as $\underline{\theta} \to (0,0)$. In domains of diameter $O(1)$, the lowest non-trivial Fourier components have $|\underline{\theta}| = O(h)$, for which $\mu(\underline{\theta}) = 1 - O(h^2)$, showing why convergence factors are that bad. Here, however, we are only interested in the smoothing effect, i.e., in the amplification factors of those components not approximable on a coarser grid. These are the components for which $(h/H)\pi \le |\underline{\theta}| \le \pi$, where H is the meshsize of

the next coarser grid. We usually assume $H/h = 2$, because it is the most convenient, and as effective as any other meshsize ratio (cf. §4.2). The *smoothing factor* is thus defined to be

$$\overline{\mu} := \max_{\frac{\pi}{2} \le |\underline{\theta}| \le \pi} \mu(\underline{\theta}). \tag{1.10}$$

It essentially gives the relaxation convergence factor for those components which need to converge only through relaxation; others will also converge through their approximation on the coarser grid.

Consider first the case $a = c$ (Poisson equation). A simple calculation shows that $\overline{\mu} = \mu(\pi/2, \arccos(4/5)) = .5$. This is a very satisfactory rate; it implies that *three relaxation sweeps reduce the high-frequency error-components by almost an order of magnitude.* Similar rates are obtained for general a and c, provided a/c is of moderate size.

The rate of smoothing is less remarkable in the degenerate case $a \ll c$ (or $c \ll a$). For instance,

$$\mu\left(\frac{\pi}{2}, 0\right) = \left(\frac{a^2 + c^2}{a^2 + (c + 2a)^2}\right)^{\frac{1}{2}}$$

which approaches 1 as $a \to 0$. Thus, for problems with such a degeneracy, Gauss-Seidel relaxation is not a suitable smoothing scheme. But better schemes exist, such as the following example.

Line Relaxation. Instead of treating each grid point (α, β) separately, one simultaneously takes a vertical line of points at a time, i.e., the set of all points (α, β) with the same α. All values $\tilde{u}_{\alpha,\beta}$ on such a line are simultaneously replaced by new values $\overline{u}_{\alpha,\beta}$ that simultaneously satisfy eqs. (1.3) on that line. (This is easy and inexpensive to do, because the system of equations to be solved for each line is a tridiagonal, diagonally dominant system.) As a result, we get the same relation as (1.4) above, except that $\tilde{u}_{\alpha,\beta+1}$ is replaced by $\overline{u}_{\alpha,\beta+1}$. Hence, instead of (1.9) we now obtain

$$\mu(\underline{\theta}) = \left|\frac{\overline{A}_{\underline{\theta}}}{A_{\underline{\theta}}}\right| = \left|\frac{a}{2(a + c - c\cos(\theta_2)) - ae^{-i\theta_1}}\right|. \tag{1.11}$$

from which one can derive the smoothing factor

$$\overline{\mu} = \max\left\{5^{-\frac{1}{2}}, \frac{a}{a + 2c}\right\}, \tag{1.12}$$

which is very satisfactory, even in the degenerate case $a \ll c$.

This situation is very general. Namely, for any stable discretization of a well-posed differential boundary-value problem, there exists a relaxation scheme which very efficiently reduces non-smooth error components (see §3 and §5.3). Moreover, the smoothing factor (1.10) for any candidate relaxation scheme is usually easy to calculate (e.g. a computer program

that numerically evaluates $\mu(\underline{\theta})$ on a sufficiently fine $\underline{\theta}$ grid; see examples in [Wei01]), even for nonlinear equations or equations with non-constant coefficients, by local linearization and coefficients freeze (see §3.1). This gives us a general tool for optimizing the relaxation scheme and predict its efficiency. It is the first example of *local mode analysis*, extensively used in multigrid analysis (see §§3.1, 4.1, 7.4 and 7.5).

1.3 Coarse grid correction

We have seen that relaxation sweeps very quickly reduce all high-frequency components of the error. Its smoother part should then be reduced by being approximated on a coarser grid, a grid with meshsize $H = 2h$, say. Generally, for any linear fine-grid equation $L^h u^h = f^h$ (for the nonlinear case, see §8.1), and any approximate solution $\tilde{u}^h$, the error $v^h = u^h - \tilde{u}^h$ satisfies

$$L^h v^h = r^h, \qquad \text{where} \qquad r^h := f^h - L^h \tilde{u}^h . \tag{1.13}$$

It can therefore be approximated by the coarse-grid function v^H that satisfies

$$L^H v^H = I_h^H r^h , \tag{1.14}$$

where L^H is some coarse-grid approximation to L^h (e.g., a finite-difference approximation on grid H to the same differential operator approximated by L^h), and I_h^H is a fine-to-coarse transfer operator, called *residual weighting* or *restriction*. That is, $I_h^H r^h$ is a coarse-grid function whose value at each point is a certain weighted average of the values of r^h at neighboring fine-grid points (see much more on this process of "coarsening" in §4 and 11, and on the treatment of boundary conditions and global conditions in §5.4, 5.5, and 5.6).

Having obtained an approximate solution $\tilde{v}^H$ to (1.14) (in a way to be discussed below), we use it as a correction to the fine-grid solution. Namely, we replace

$$\tilde{u}^h \leftarrow \tilde{u}^h + I_H^h \tilde{v}^H , \tag{1.15}$$

where I_H^h is a coarse-to-fine *interpolation* (also called *prolongation*). That is, at each fine-grid point, the value of $I_H^h \tilde{v}^H$ (designed to approximate the error v^h) is interpolated from values of $\tilde{v}^H$ at neighboring coarse-grid points. Linear interpolation can be use in most cases (further discussion of interpolation orders appears in §4.3). The whole process of calculating $I_h^H r^h$, solving (1.14) and interpolating the correction (1.15) is called a *coarse-grid correction*.

1.4 Multigrid cycle

To efficiently get an approximate solution to the coarse-grid equation (1.14), we employ the above solution process recursively; i.e., (1.14) is itself solved

by relaxation sweeps, combined with a still-coarser-grid corrections. We thus have a sequence of grids with meshsizes $h_1 > h_2 > \cdots > h_M$, where usually $h_k = 2h_{k-1}$. The grid-h_k equation is generally written as

$$L^k u^k = f^k , \tag{1.16}$$

where all the operators L^k approximate each other (e.g., they are all finite-difference approximations to the same differential operator), and unless k is the finest level ($k = M$), equation (1.16) is of the form (1.14), i.e., u^{k-1} is always designed to be the coarse correction to $\tilde{u}^k$ (the current approximation on the next finer grid), and hence

$$f^{k-1} = I_k^{k-1}\left(f^k - L^k \tilde{u}^k\right) \tag{1.17}$$

(superscripts and subscripts l are now used instead of h_l in the notation of §1.3. Also, u^{k-1} is used instead of v^{2h} for the purpose of uniform expressions at all levels).

The exact algorithm for improving a given approximate solution $\tilde{u}^k$ to (1.16) is usually the *multigrid cycle* (MGC)

$$\tilde{u}^k \leftarrow \mathrm{MGC}\left(k, \tilde{u}^k, f^k\right) , \tag{1.18}$$

defined recursively as follows:

> If $k = 1$, solve (1.16) by Gaussian elimination or by several relaxation sweeps (either is usually inexpensive, because the grid is extremely coarse. For additional remarks concerning the coarsest-grid solution, see §6.3). Otherwise, do the following four steps:
>
> (A) Perform ν_1 relaxation sweeps on (1.16), resulting in a new approximation $\overline{u}^k$.
>
> (B) Starting with $\tilde{u}^{k-1} = 0$, make γ successive cycles of the type
>
> $$\tilde{u}^{k-1} \leftarrow \mathrm{MGC}\left(k-1, \tilde{u}^{k-1}, I_k^{k-1}\left(f^k - L^k \overline{u}^k\right)\right) .$$
>
> (C) Calculate
>
> $$\overline{\overline{u}}^k = \overline{u}^k + I_{k-1}^k \tilde{u}^{k-1} . \tag{1.19}$$
>
> (D) Finally, perform ν_2 additional relaxation sweeps on (1.16), starting with $\overline{\overline{u}}^k$ and yielding the final $\tilde{u}^k$ of (1.18).

The sweep count ν_1 and ν_2 are usually either 0, 1 or 2, with $\nu = \nu_1 + \nu_2$ usually being 2 or 3 (see more about this in §4.1 and 6.1). The cycle count γ is usually either 1 or 2. The cycle with $\gamma = 1$ is called a *V cycle*, or $\mathrm{V}(\nu_1, \nu_2)$, in view of the shape of its flowchart (see Fig. 1.1). For a similar

reason, the cycle with $\gamma = 2$ is called a *W cycle*, or $\mathrm{W}(\nu_1, \nu_2)$ (see more about different cycles and alternative switching criteria in §6.2).

The ν sweeps performed in each V cycle on any grid h_k are expected to reduce error components with wave-length between $2h_k$ and $4h_k$ at least by the factor $\overline{\mu}^{\nu}$, where $\overline{\mu}$ is the smoothing factor (1.10). Because all grids are so traversed, *the cycle should reduce all error components by at least the factor* $\overline{\mu}^{\nu}$. Experience and more advanced theory show that for regular elliptic problems, this is indeed the case, provided the boundary conditions are properly relaxed, and correct inter-grid transfers are used. Thus, $\overline{\mu}$ can serve as an excellent predictor of the multigrid performance one should *be able to* obtain.

1.5 Model program and output

A simple unoptimized MATLAB program called `TestCycle` is included in Appendix A to illustrate multigrid algorithmic ideas and programming techniques. The software design easily carries over to any object-oriented language such as `C++` or `Java`(TM).

The main Class `TestCycle` solves the Poisson equation $\Delta u = F(x, y)$ with Dirichlet boundary conditions $u = G(x, y)$ on a rectangle by applying `numCycles` $V(\nu_1, \nu_2)$ cycles to a random initial guess.

The input options are centralized in Class `Options`. A sequence of `numLevels` grids is defined over the domain $[0, \texttt{domainSize}(1)] \times [0, \texttt{domainSize}(2)]$. The coarsest grid (level `1`) has $\texttt{nCoarsest}(1) \times \texttt{nCoarsest}(2)$ intervals of of length $\mathtt{h}_1$ each. Subsequent grids at levels $\mathtt{k} = 2, \ldots, \texttt{numLevels}$ are defined as uniform refinements with meshsizes $\mathtt{h}_\mathtt{k} := 2^{1-\mathtt{k}}\mathtt{h}_1$. Therefore, the problem is solved at the finest grid with meshsize $\mathtt{h}_{\texttt{numLevels}}$; coarse levels are only used to accelerate convergence.

The V-cycle flow depicted in Fig. 1.1 is implemented by Class `Cycle`, which manages an internal list of `Level` instances. Each `Level` instance encapsulates the discrete operator (the same 5-point approximation to the Laplace operator (1.3) with $a = c = 1$ is used at all levels), relaxation scheme (Gauss-Seidel), residual transfer $I_{\mathtt{k}}^{\mathtt{k}-1}$ (full weighting) and interpolation of corrections $I_{\mathtt{k}-1}^{\mathtt{k}}$ (bilinear). Note that the `Level` class is generic, and conveniently allows swapping in and out different multigrid ingredients without affecting the rest of the code.

The program output (generated with MATLAB 7.10.0.499 (R2010a)) is shown on the next page. At each step of the algorithm, the l_2 norm of the ("dynamic") residuals is printed, as well as a count of the cumulative relaxation work, where a finest level sweep serves as the work unit. Cycles exhibit an asymptotic convergence factor of about .11 per cycle, slightly better than $\overline{\mu} = .5^3 = .125$ predicted by the smoothing factor (the two-level mode analysis described in §4.1 is able to precisely predict the observed factor). This is equivalent to a convergence factor of $.11^{\frac{1}{3}} = .48$ per relaxation sweep.

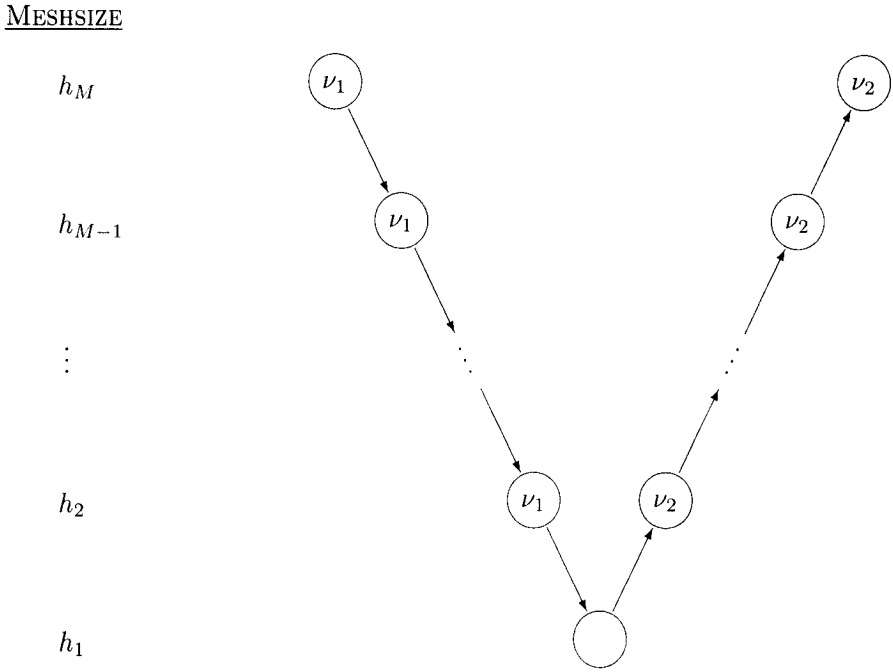

Figure 1.1. ***Multigrid cycle*** $V(\nu_1, \nu_2)$.

ν_i *stands for* ν_i *relaxation sweeps at the meshsize shown to the left. (At the coarsest grid :* $\nu_1 + \nu_2$ *relaxation sweeps are usually performed, or the equations are solved directly.)*

↘ *is the fine-to-coarse (*$k+1$ *to* k*) transfer.* f^k *is defined by (1.17) and* u^k *is trivially initialized (*$u^k \leftarrow 0$*).*

↗ *is the coarse-to-fine (*$k-1$ *to* k*) interpolation of correction (1.19).*

The same algorithm attains the exact same efficiency on general non-rectangu-lar domains: `TestCycle`'s flexible design is extensible, although the implementation of its components naturally becomes more complex. Collections of multigrid programs with varying degrees of simplicity vs. generality are available, e.g. [Hym77, Dou05].

We strongly recommend the reader to experiment with the program and tweak it in various ways (vary the input options, modify the equations, etc.) to acquire a deep understanding of the conceptual and technical aspects of multigrid.

```
              TestCycle Output
              ----------------
>> TestCycle.run;
##################### CYCLE #1 #####################
LEVEL ACTION                    ERROR NORM    WORK
5     Initial                   1.211e+003     0.00
5     Relaxation sweep 1        3.512e+002     1.00
5     Relaxation sweep 2        1.312e+002     2.00
4     Initial                   5.661e+001     2.00
4     Relaxation sweep 1        2.673e+001     2.25
4     Relaxation sweep 2        1.752e+001     2.50
3     Initial                   1.353e+001     2.50
3     Relaxation sweep 1        7.609e+000     2.56
3     Relaxation sweep 2        5.408e+000     2.63
2     Initial                   4.126e+000     2.63
2     Relaxation sweep 1        2.296e+000     2.64
2     Relaxation sweep 2        1.362e+000     2.66
1     Initial                   7.402e-001     2.66
1     Relaxation sweep 400      0.000e+000     4.22
2     Coarse-grid correction    9.729e-001     4.22
2     Relaxation sweep 1        3.406e-001     4.23
3     Coarse-grid correction    3.632e+000     4.23
3     Relaxation sweep 1        1.007e+000     4.30
4     Coarse-grid correction    1.313e+001     4.30
4     Relaxation sweep 1        3.423e+000     4.55
5     Coarse-grid correction    1.179e+002     4.55
5     Relaxation sweep 1        3.971e+001     5.55
CYCLE  1   CONVERGENCE FACTOR = 0.033
##################### CYCLE #2 #####################
LEVEL ACTION                    ERROR NORM    WORK
5     Initial                   3.971e+001     5.55
5     Relaxation sweep 1        1.584e+001     6.55
5     Relaxation sweep 2        7.647e+000     7.55
4     Initial                   4.535e+000     7.55
4     Relaxation sweep 1        2.360e+000     7.80
4     Relaxation sweep 2        1.635e+000     8.05
3     Initial                   1.291e+000     8.05
3     Relaxation sweep 1        7.766e-001     8.11
3     Relaxation sweep 2        5.736e-001     8.17
2     Initial                   4.439e-001     8.17
2     Relaxation sweep 1        2.515e-001     8.19
2     Relaxation sweep 2        1.479e-001     8.20
1     Initial                   7.935e-002     8.20
1     Relaxation sweep 400      0.000e+000     9.77
```

```
2     Coarse-grid correction    1.079e-001     9.77
2     Relaxation sweep 1        3.743e-002     9.78
3     Coarse-grid correction    3.899e-001     9.78
3     Relaxation sweep 1        1.060e-001     9.84
4     Coarse-grid correction    1.228e+000     9.84
4     Relaxation sweep 1        3.180e-001    10.09
5     Coarse-grid correction    6.290e+000    10.09
5     Relaxation sweep 1        2.175e+000    11.09
CYCLE  2   CONVERGENCE FACTOR = 0.055
##################### CYCLE #3 #####################
LEVEL ACTION                    ERROR NORM    WORK
5     Initial                   2.175e+000    11.09
5     Relaxation sweep 1        1.014e+000    12.09
5     Relaxation sweep 2        5.740e-001    13.09
4     Initial                   3.827e-001    13.09
4     Relaxation sweep 1        2.189e-001    13.34
4     Relaxation sweep 2        1.583e-001    13.59
3     Initial                   1.268e-001    13.59
3     Relaxation sweep 1        8.092e-002    13.66
3     Relaxation sweep 2        6.139e-002    13.72
2     Initial                   4.785e-002    13.72
2     Relaxation sweep 1        2.725e-002    13.73
2     Relaxation sweep 2        1.583e-002    13.75
1     Initial                   8.372e-003    13.75
1     Relaxation sweep 400      0.000e+000    15.31
2     Coarse-grid correction    1.180e-002    15.31
2     Relaxation sweep 1        4.049e-003    15.33
3     Coarse-grid correction    4.234e-002    15.33
3     Relaxation sweep 1        1.135e-002    15.39
4     Coarse-grid correction    1.201e-001    15.39
4     Relaxation sweep 1        3.090e-002    15.64
5     Coarse-grid correction    4.576e-001    15.64
5     Relaxation sweep 1        1.572e-001    16.64
CYCLE  3   CONVERGENCE FACTOR = 0.072
CYCLE  4   CONVERGENCE FACTOR = 0.084
CYCLE  5   CONVERGENCE FACTOR = 0.093
CYCLE  6   CONVERGENCE FACTOR = 0.098
CYCLE  7   CONVERGENCE FACTOR = 0.103
CYCLE  8   CONVERGENCE FACTOR = 0.105
CYCLE  9   CONVERGENCE FACTOR = 0.103
CYCLE 10   CONVERGENCE FACTOR = 0.109
CYCLE 11   CONVERGENCE FACTOR = 0.111
CYCLE 12   CONVERGENCE FACTOR = 0.106
```

1.6 Full Multigrid (FMG) algorithm

The multigrid cycles described above can be applied to any first approximation given at the finest grid. In a full multigrid (FMG) algorithm, the

first approximation is obtained by interpolation from a solution at the next coarser grid, which has previously been calculated by a similar FMG algorithm. With such a first approximation, and provided that the interpolation correctly corresponds to the error norm used, one multigrid cycle should suffice to solve the fine-grid equations to the level of truncation errors (see §7). A typical FMG algorithm, with one V cycle per refinement, is shown in Fig. 1.2. FMG algorithms are less sensitive than the multigrid cycles. That is, in many irregular cases the asymptotic convergence factor of the cycles is not good, but an FMG algorithm with one cycle per refinement still guarantees solution to the level of truncation errors. In fact, even this guarantee is not necessary: from differences between the final solutions at different meshsizes (e.g., differences between the solutions at the doubly-circled stages in Fig. 1.2), one can directly calculate the rate of convergence to the *differential* solution, which is all that really matters.

1.7 General warnings. Boundary conditions. Nonlinearity

Attempts to extend existing multigrid software often fail. For example, almost everyone who tries to extend the program of §1.5 from Dirichlet to Neumann boundary conditions, first obtains a much slower solver. Some would then hastily announce that the method is inherently slower for general non-Dirichlet boundary conditions. Others, realizing the conceptual generality of the basic multigrid approach, would ask themselves what caused the slowness and how to correct it. They will eventually discover that various additional processes should be done in case of non-Dirichlet conditions, such as relaxation sweeps over these conditions and transfers of their residuals to coarser grids, concurrently with the corresponding interior processes. It will take some more effort to realize that the best relaxation of boundary conditions in multigrid solvers is often quite markedly different from the scheme one would use in relaxation solvers. Eventually, when everything is done correctly, the algorithm will regain the efficiency it showed in the Dirichlet case. Indeed, with proper treatment, the multigrid efficiency should never depend on boundary conditions, only on the interior equations (and in fact, only on the interior smoothing rates, hence only on the *factors* of the subprincipal *determinant* of the interior operator – see §3.7).

Imagine now someone trying to extend the simplest program and write a multigrid solver for the steady-state compressible Navier-Stokes or Euler equations. Here he has a *multitude* of new features: non-Dirichlet boundary conditions is just one of them, and by no means the most difficult one. Others are: non-symmetry; non-linearity; anisotropy; non-ellipticity, in fact a challenging mix of PDE types; boundary singularities (e.g., trailing edges); discontinuities (boundary layers, shocks); global conditions (Kutta

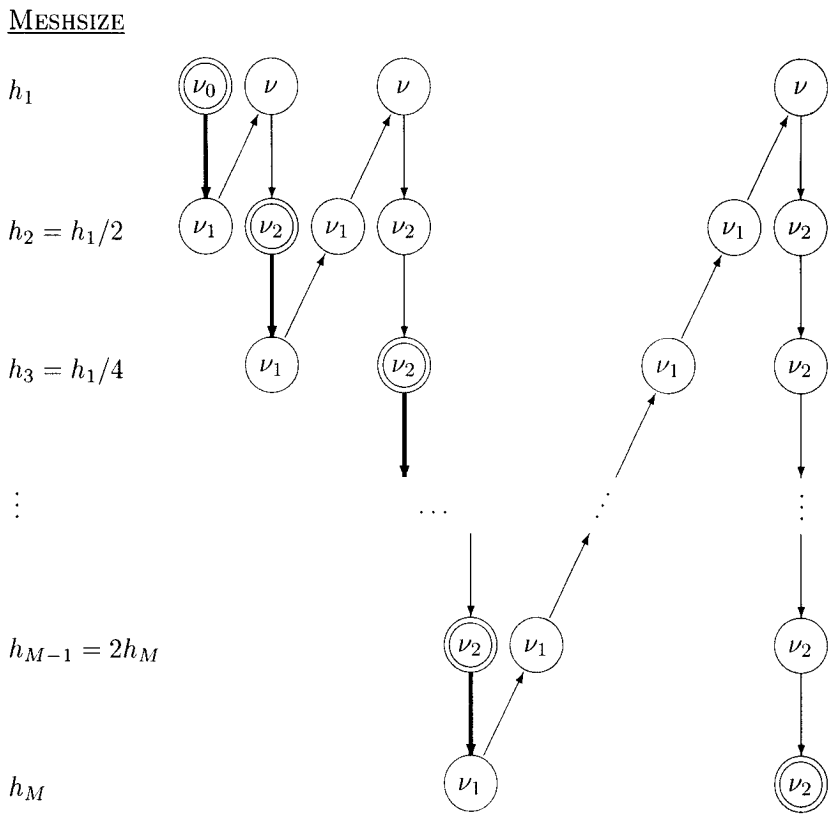

Figure 1.2. ***FMG Algorithm with one*** $\mathrm{V}(\nu_1, \nu_2)$ ***cycle per level.*** *The grids are pictured upside down relative to Fig. 1.1 (both ways are common in the multigrid literature).*

↓ *is the solution interpolation to a new grid.*

↓ *is the coarse-to-fine ($k-1$ to k) interpolation of correction (1.19).*

↗ *is the fine-to-coarse ($k+1$ to k) transfer. f^k is defined by (1.17) and u^k is set to 0.*

(ν_i) *stands for ν_i relaxation sweeps. At the coarsest grid $\nu = \nu_1 + \nu_2$ or somewhat larger ν_0 are usually used, or the equations are solved directly.*

◎ *shows the stage in the algorithm where the final solution is obtained for the corresponding meshsize.*

condition, total mass, etc.); unbounded domains; complex geometries; three dimensions; not to mention physical instabilities and turbulence.

Each and every one of these features requires a thorough understanding as to what it implies in terms of each multigrid process. Mistreating just one of them may cause the solution time to increase very significantly, sometimes by orders of magnitude. Because convergence will often still be obtained, due to the corrective nature of the fine-grid relaxation, one may be misled to believe that nothing is wrong. Even if he suspects errors, he is unlikely to find them all, because they confusingly interact with each other, an impossible network of conceptual mistakes with programming bugs.

As stated in the Introduction, we believe that every problem should be solvable in just few work units. But only a systematic development is likely to produce this top performance. Each feature should first be separately studied in the framework of as simple a problem as possible. The solver should be then constructed step by step, adding one feature at a time, making sure that full efficiency is always maintained. This Guide may help in the process.

It may be useful to add here a preliminary remark about nonlinearity. The multigrid processes are *not* inherently linear. The basic idea described in §1.1, namely, the relations between slow convergence, smallness of residuals, and error smoothness, has nothing to do with linearity. Multigrid can thus be applied *directly to nonlinear problems*, as efficiently as to the corresponding linearized problems (see §8). Hence, repeated linearizations are neither required nor advised; they are especially wasteful in case the nonlinear problem is autonomous (as is usual in fluid dynamics), because the linearized problems are not autonomous (see more detailed arguments in §8.3). Moreover, the multigrid version developed for nonlinear cases, called FAS, is useful in many other ways. In particular, it gives a convenient way to create non-uniform adaptable discretization patterns, based on the interaction between the levels and therefore very flexible, allowing fast local refinements and local coordinate transformations, with equations being still solved in the usual multigrid efficiency (see §9). Using FAS, one can integrate into a single FMG process various other processes, such as continuation, design and optimization, solution to eigenvalue problems and to inverse problems, grid adaptation, etc.