Chapter 7

Full Multi-Grid (FMG) Algorithms

The cycling algorithms developed in the previous stages are easily converted into full multigrid (FMG) programs. The main difference is that instead of starting with an arbitrary approximation (e.g., $u_0^h = 0$) on the finest grid, the first approximation u_0^h is obtained by an interpolation $\mathbb{I}_H^h$ from a coarse-grid (approximate) solution u^H. Namely, $u_0^h = \mathbb{I}_H^h u^H$, where $H = 2h$ and where u^H has been calculated by a similar FMG process with H as its finest level. The full algorithm can be either "fixed" (as for example in Fig. 1.2 above), or "accommodative" (as in [Bra77b, §1.3], [Bra80b, Fig. 1], [Bra79a, §3.6 and Fig. 1], [BD79, §2.2]). Both versions are available in the model program FMG1 [MUG84].

FMG algorithms are in a sense *easier* to program than cycling algorithms. Their main deriving routine is some lines longer, they include an additional interpolation routine ($\mathbb{I}_H^h$), and they involve several more algorithmic questions (dealt with in the following subsections) - but on the other hand they are much more *forgiving.* Their basic performance, which is to solve all problems to the level of truncation errors in just one or two cycles (see §7.3), is undisturbed by various little mistakes (conceptual mistakes or even programming bugs, especially in treating boundaries) which may degrade very much the *asymptotic* convergence of cycling algorithms. These mistakes may still be important in other situations, hence it is safer to detect them by perfecting the multigrid cycling (as in §4, 5 and 6) before turning FMG on. But it is important to understand those cases in which, for good reasons, the FMG results are absolutely satisfactory despite the necessarily bad asymptotic convergence factors. Examples are numerous; some are emphasized in §3.3, 5.7, 7.4, 7.5, 10.2 and 18.6. In many of these cases the analyses of FMG described in §7.4 and 7.5 can be useful.

More important than these apriori analyses, though, one should *always calculate in the FMG solver the rate of convergence to the differential* *solution*, which is the real purpose of the solver; see the end of §7.2 and §12.

It is also worth mentioning at this point that the FMG algorithm can incorporate into it continuation processes (see §8.3.2), grid adaptation processes (see §9.6), and, generally speaking, any process aimed at solving original "outer" problems (see §13).

7.1 Order of the FMG interpolation

The FMG full-solution interpolation operator $\mathbb{I}_H^h$ is not necessarily the same as the correction interpolation operator I_H^h used in the multigrid correction cycles. Often the order of $\mathbb{I}_H^h$ should be higher than the order of I_H^h since the first approximation is smoother than the corrections: In the right-hand side of the latter (i.e., in the residuals) the amplitude of high-frequency components is usually comparable to that of low-frequency components. The optimal order of $\mathbb{I}_H^h$ depends on the purpose of calculations. If one desires ultimately to get the algebraic error (i.e., the errors in solving the *difference* equations) to be very small (far below truncation errors), then $\mathbb{I}_H^h$ should exploit all the smoothness of u^h in order not to produce unnecessary high-frequency algebraic errors. (High frequency errors are the most expensive to liquidate in the multigrid cycling, since they are processed on the finest grid.) In fact, in such a case the first few cycles should also employ a correction interpolation I_H^h of suitably high orders. The precise rules for scalar elliptic equations are given in [Bra81a, App. A.2]. Note that these rules assume that the order of smoothness is known in advance.

Usually, however, the smoothness order is not apriori known. More importantly, we are not interested in solving to arbitrarily small algebraic errors; we like them only to be smaller than the truncation errors. The optimal order depends then on the norm by which we measure errors. Suppose we solve a $q \times q$ system of differential equations, and assume our error norm includes difference-quotients up to order l_j in the *j-th* unknown function, $1 \le j \le q$. Then the order $\hat{m}^j$ of the first interpolation of that function should not be less than $p + l_j$, where p is the approximation order. Otherwise, the $O(h^{\hat{m}^j - l_j})$ high-frequency errors produced by interpolation would be much larger than the $O(h^p)$ (low-frequency) truncation errors.

In the case of equations with strongly discontinuous coefficients, the higher order interpolation $\mathbb{I}_H^h$ should be of a different form, taking into account the different sense of smoothness in the solutions (cf. §4.6. A higher-order interpolation of this sort is presented in [ABDP81, Eq. (5.12)]). The remarks of §5.2 apply here as well.

A general approach for equations with discontinuous coefficients or right-hand side is to use one-sided interpolation stencils that avoid straddling the discontinuity.

Whatever FMG interpolation $\mathbb{I}_H^h$ is used near a right-hand side discontinuity, it should be followed by special local relaxation steps around the discontinuity. (This automatically follows from the general Local Relaxation Rule of §5.7).

In some programs, especially general-domain programs, the higher order interpolation $\mathbb{I}_H^h$ turned out to cost more CPU time than the rest of the algorithm [Oph78]. An interpolation of an order smaller than indicated above may then be more practical. In case of rotatable differential operators, simpler higher-order interpolations can be used, based on the equations themselves [Hym77], [FW81, §3].

7.2 Optimal switching to a new grid

In designing the FMG algorithm one should decide how well the equations on level $H = 2h$ should be solved before the solution is interpolated for the first time to grid h and the h-cycles start. The optimal point to switch is when the work of h-cycles becomes as efficient as the work of H-cycles in reducing the *differential* error (the difference between the differential solution u and our current computed solutions, $\tilde{u}^H$ or $u_0^h = \mathbb{I}_H^h \tilde{u}^H$). This happens when the *algebraic* error on grid H, namely $e_*^H = \|u^H - \tilde{u}^H\|$, is about 2^{-d} times the algebraic error on grid h, $e_0^h = \|u^h - u_0^h\|$, where d is the dimension, u^H is the exact solution of the H-equations and u^h is the exact solution of the h-equations. This is because h-cycles are about 2^d times as expensive as H-cycles. The switching point $e_*^H \approx 2^{-d} e_0^h$ is roughly equivalent to

$$e_*^H \approx \beta E^H, \qquad \beta := \frac{1-2^{-p}}{2^d - 1}, \tag{7.1}$$

where $E^H = \|u^H - u\|$ is the truncation error on grid H and p is the order of approximation. This is because $e_*^H + E^H \approx e_0^h + 2^{-p} E^H$, both sides being estimates for $\|\tilde{u}^H - u\|$. In practice the values of e_*^H and E^H are of course not known, but we can derive from (7.1) the algebraic reduction needed on level H before switching. Namely, denoting by e_0^H the value of e^H when the H cycles are started and by e_*^H its value at the switching point (7.1), and assuming that the switching from the $2H$-cycles to the H-cycles has been made when a relation similar to (7.1) was reached on level $2H$, we find that $e_0^H \approx 2^d e_*^{2H} \approx 2^d \beta E^{2H}$ while $e_*^H = \beta E^H = \beta 2^{-p} E^{2H}$, consequently the algebraic reduction on grid H is roughly

$$\frac{e_*^H}{e_0^H} \approx 2^{-p-d}. \tag{7.2}$$

This can be obtained by about

$$\frac{p+d}{\log_2(1/\bar{\lambda})} \tag{7.3}$$

H-cycles, where $\bar{\lambda}$ is the convergence factor per cycle (§4.1), which can of course be measured. This number of required H-cycles usually turns out to be 1 or 2. A more precise strategy for simultaneously optimizing the number of cycles at all levels is described in [TTKR10].

7.3 Total computational work. Termination criteria

Suppose that on the finest grid h we wish to obtain an algebraic error smaller than a specified factor α times the truncation error: $e^h \leq \alpha E^h$. Suppose also that the switch from level $H = 2h$ is made roughly when (7.1) is met; i.e., when $2^{-d}e_0^h \approx e_*^H \approx 2^p\beta E^h$. Then the algebraic error reduction required on grid h is roughly $\alpha_1 = e^h/e_0^h \approx \alpha(1-2^{-d})/(2^p-1)$. The number of work units to obtain such a reduction is about $\log(1/\alpha_1)/\log(1/\mathring{\mu})$, where $\mathring{\mu}$ is the interior convergence factor per work unit (see §4.1) and is usually just modestly larger than the interior smoothing factor $\overline{\mu}$. Counting also the work for the reduction (7.2) on coarser grids, we find that the total number of work units theoretically required by the Full MultiGrid (FMG) algorithm is about

$$\left\{\log\left(\frac{2^p-1}{\alpha(1-2^{-d})}\right)+\frac{p+d}{2^d-1}\log 2\right\}\Big/\log\left(\frac{1}{\mathring{\mu}}\right). \tag{7.4}$$

The actual total number of work units is usually slightly larger than (7.4), because of the need to perform *integral* numbers of relaxation sweeps and coarse grid corrections. *In practice, one V or W cycle with $\nu = 2$ or 3 at each FMG level, yields an e^h that is significantly smaller E^h.*

The observation that only one cycle on each level is needed in FMG algorithms, and is also basically enough to reduce the algebraic errors to the level of truncation errors (even though sometimes two shorter cycles, each including less relaxation work, may be more efficient), can heuristically be understood as follows. The first approximation on grid h, obtained by interpolating the grid-$2h$ solution, necessarily contains two types of errors:

(A) High-frequency errors, i.e., errors having oscillations invisible and hence unapproximable on the coarser grid.

(B) Aliasing errors, i.e., smooth errors introduced by high-frequency data because on the coarse grid high-frequency data is mistaken for smooth data.

Relaxation on grid h can be efficient in removing the high-frequency errors (because of their local nature: At each point they are essentially determined by the neighboring residuals). Having removed the high-frequency errors we have also removed the high-frequency data from the *residual* problem, hence we can then go back to grid 2h with the residual problem to remove the aliasing errors (which are smooth errors, hence not determined by neighboring residuals, hence inefficiently treated by relaxation).

The algorithm may indeed be terminated after a fixed number of cycles on the finest grid h. This number is roughly $\log(1/\alpha_1)/\log(1/\overline{\lambda})$, and in practice it is one or two. Or else, especially if an estimate for $\overline{\lambda}$ is not known, termination can be done when a certain norm of the residuals on grid h becomes smaller than a corresponding norm of $\alpha\tau^h \approx \alpha(2^p-1)^{-1}\tau_h^{2h}$ (see §8.4).

One should of course check numerically, using a problem with a known solution or a solution computed on a finer grid, that with these termination procedures $e^h \leq \alpha E^h$ is indeed obtained. Better still (from the more advanced point of view of §13), one can observe the behavior of e^h as function of h. This can approximately be done in (the real, production) runs where the solution is *not* known (see §1.6 and the further discussion in §13). Quite often these checks also reveal mistakes in the *discretization* schemes, not just in the multigrid solver.

7.4 Two-level FMG mode analysis

Instead of developing full multigrid (FMG) programs from the cycling programs, including boundary conditions, one can first develop the FMG algorithm still within the framework of two-level mode analysis (immediately following the stage of §4). This may again serve to separate away questions related to boundary conditions (questions discussed in §5), and questions related to many levels and to very coarse grids (§6) from the particular questions of the FMG algorithm (§7.1–7.3). The latter can then be examined in the interior, without boundary interference (or also with ideal boundaries – see §7.5), and the performance figures so calculated can serve as ideals against which the actual program can be developed and debugged. Such an analysis is particularly useful in cases the usual two-level analysis (that of §4.1 above) is too pessimistic because of the existence of different components with markedly different convergence properties. For example, in case of nearly non-elliptic or nearly semi-elliptic problems there are smooth characteristic components which converge slower than others, since for such components L^H is not a very good approximation to L^h. But exactly for the same components and for the same reason, L^h itself is not a good approximation to L, hence these components do not need much algebraic convergence, and the fact that they have slow asymptotic rates does not matter [Bra81a, §5.1, 5.2]. What we need then is an analysis which does not tell us the worst *asymptotic* rate, but tells us, separately in each mode, how well we solve the problem by an FMG algorithm with a given number of prescribed cycles.

To analyze the FMG solution of $Lu = f$, where L is a $q \times q$ system and has constant coefficients (or frozen local values, in case the original L was variable), we first analyze a single component $u(\underline{\theta}) = exp(i\underline{\theta} \cdot \underline{x}/\underline{h})$. We calculate the corresponding f, and hence also the solution $u^H = u^H(\underline{\theta})$ to the coarse-grid equation $L^H u^H = f^H = I^H f$, where I^H is the local averaging used in our discretization for transferring a continuum function to grid H. The interpolation of u to the fine grid gives an approximation $u_0^h = \mathbb{I}_H^h u^H$ made up of 2^d Fourier components (the harmonics of $\underline{\theta}$, i.e. all the components $\underline{\theta}'$ such that $\underline{\theta}' = \underline{\theta} + (\nu_1, \dots, \nu_d)\pi$, ν_j integer and $-\pi < \theta_j' \le \pi$). To the set of 2^d amplitudes we then apply the usual (cycling) two-level mode analysis (§4.1); i.e., using (4.1) we calculate the $2^d q \times 2^d q$ matrix $M(\underline{\theta})$ describing the transformation of these 2^d amplitudes by one cycle. The result of applying k such cycles on grid h we denote by $u_k^h(\underline{\theta}) = M(\underline{\theta})^k u_0^h(\underline{\theta})$. Having calculated $u_k^h(\underline{\theta})$ we can then examine its qualities by several measures.

One measure is **how well below truncation errors** u_k^h is. This is measured for example by

$$\max_{|\underline{\theta}| \le \pi} \frac{\|u_k^h(\underline{\theta}) - u(\underline{\theta})\|}{\|u(\underline{\theta}) - u^h(\underline{\theta})\|}, \tag{7.5}$$

where $u^h(\underline{\theta})$ is the exact solution of the grid h equations, and $\|\cdot\|$ is any norm under which we want to guarantee convergence. Note that u_k^h is made of 2^d components, while u^h and u are made of only one of those; the norms can be taken anyway.

Another, perhaps more direct and important measure, is **how well we have solved the differential equations**. That is, we directly measure $\|u_k^h - u\|$ thus evaluating not only the performance of our fast solver, but also the quality of our discretization scheme itself: We evaluate the total quality of our procedures in solving the differential equations at a given amount of work. In measuring the error $\|u_k^h - u\|$ we should of course give smaller weights to high-frequency $\underline{\theta}$'s; we cannot and need not solve for them as accurately as for low frequencies. Thus, if we aim at an approximation order p, good measures of performance are

$$\max_{|\underline{\theta}| \le \pi} \frac{\|u_k^h(\underline{\theta}) - u(\underline{\theta})\|}{|\underline{\theta}|^p}, \tag{7.6}$$

or

$$\left\{ \int_{|\underline{\theta}| \le \pi} |\underline{\theta}|^{-2p} \|u_k^h(\underline{\theta}) - u(\underline{\theta})\|^2 d\underline{\theta} \right\}^{\frac{1}{2}}, \tag{7.7}$$

etc. Several such measures can easily be produced, approximately, by the program that calculates $u_k^h(\underline{\theta})$. The program is an easy extension of the usual (cycling) two-level mode-analysis program.

All the issues examined by the two-level cycling analysis (relaxation, the number $\nu = \nu_1 + \nu_2$ of sweeps per cycle, and the interior operators I_h^H, L^H and I_H^h – see §4) can further (more accurately) be optimized by the FMG mode analysis. In addition we can examine by this analysis the effect (in the interior) of various $\mathbb{I}_H^h$ interpolation procedures, various values of ν_1, ν_2 and k, and, most importantly, various interior discretization procedures.

An important advantage is that this analysis can be used even in cases where no *algebraic* convergence is desired or obtained (cf. §10.2). Moreover, *the predictions of the FMG-mode analysis are more robustly held than those of the cycling mode analysis when real boundaries are added.* For example, take a problem with singularities in the boundary, such as reentrant corners. The effect of such singularities (similar to the effect of singular perturbations mentioned above) is to make the asymptotic convergence factors per cycle worse than predicted by the interior cycling mode analysis. But this occurs for particular components with large truncation errors (see §5.7), so that predictions like (7.5) are likely to be roughly held up.

A very simple example of two-level FMG mode analysis for a singular perturbation equation is given in [Bra81a, §5.2]. A similar analysis holds for semi-elliptic cases.

7.5 Half-space FMG mode analysis. First differential approximations

Another advantage of the two-level FMG mode analysis is the possibility to make it also near a piece of boundary, modeled by a grid hyperplane, so that the entire domain is modeled by a half space. This is particularly important in non-elliptic or singular perturbation problems, where the high-frequencies far in the interior can still strongly be affected by the boundary.

A simple example is given in [Bra81a, §5.3] for a singular perturbation equation. Its upshot is that both the algebraic error (after a one-cycle two-level FMG algorithm) and the truncation error increase as functions of the distance from the boundary, both eventually becoming as large as the solution itself; but at points where the truncation error is still small (compared with the solution), the algebraic error is smaller yet: the latter is at most a quarter the size of the former. (Again, a similar analysis can be made, with very similar results, for semi-elliptic equations.) *Interior* analysis could not of course describe this situation, and its relevance in such cases is therefore questionable.

Those examples in [Bra81a] illustrate another technique which can be used whenever one wants to focus one's analysis on *smooth* components. One can then simplify the analysis very much by using the **first-differential approximation** (the first terms in a Taylor expansion) of the difference operators, instead of the difference operators themselves. For example, the first differential approximation to the 5-point Laplacian is the differential operator $\partial_{xx} + \partial_{yy} + \frac{1}{12}h^2(\partial_{xxxx} + \partial_{yyyy})$. The analysis proceeds in the continuous domain, without mentioning grids except through the quantity h appearing in the operators.