Chapter 5

Boundary Conditions and Two-Level Cycling

The theoretical two-level mode analysis described above (§4.1), and/or the numerical experiments with periodic boundary conditions (§4), give us the ideal convergence factor per cycle ($\bar{\lambda}$), or per work-unit ($\mathring{\mu}$) . These are the *interior convergence factors*, obtained in the absence of boundary interference. The next stage is to construct an actual multigrid program for an actual, bounded domain, and in particular to decide on the special treatment the various processes should take at points near or on boundaries. The goal is to attain or approach the interior convergence factors. For elliptic problems this is generally possible, since smoothing away from the boundary is decoupled from the boundary and since the boundary neighborhood itself is a lower-dimensional set of grid points, hence we can allow there more work (per point) than in the interior, without changing the total work by much [Bra94]. The comparison to the interior factors is a very important tool in debugging the program or finding conceptual mistakes, especially mistakes in treating boundary conditions or interior equations at points adjacent to boundaries. On the other hand, approaching the interior factors is not all-important; in fact, optimal performance of the full multigrid (FMG) algorithm may well be obtained without it, especially in non-elliptic or small-ellipticity problems (see end of §4.1).

In §5.2–5.5 below we mention some rules related to the multigrid processes near or on boundaries. The general remarks of §11 and the curved-boundary treatment in §9.3 are also relevant here.

In addition to boundary conditions, some problems have global conditions. These should also be incorporated at this stage. Their multigrid implementation is discussed in §5.6.

5.1 Simplifications and debugging

It is advisable to start with a program for **rectangular domains** whose boundaries coincide with grid lines at all levels. This will make the programming much easier (the program in §1.5 can serve as a model), and will separate away various difficulties related to more general domains. In fact, the first stage in constructing such a program could be the case of periodic boundary conditions (separating away boundary considerations altogether) discussed in §4.

Having made rectangular models work satisfactorily, one can then proceed to other domains. At this point one has to decide whether to write a general-domain or a specific domain program. Experience shows **general-domain** multigrid programs to be considerably less efficient (typically requiring twice the CPU time). One can model one's general domain program after MG01 [ST82, §10], or after MUGPACK, or actually use the MUGPACK or GRIDPACK software [MUG84]. But the efficiency of this software, too, is still considerably below the efficiency of specific domain programs (where the efficiency of rectangular domains can be approached). The reason is the many checks that should be made to distinguish between various possible positions of gridpoints with regard to the boundary, especially in interpolation and residual-transfer routines, where two grids are simultaneously involved.

It is advisable to start programming **cycling algorithms**, before proceeding to the additional questions related to the full multigrid (FMG) algorithm (taken up in §7). Cycling algorithms start with some arbitrary approximation on the finest grid and reduce its error by cycling between that grid and coarser grids. At this stage, one can avoid the question of what cycle to use: For debugging purposes it is best to start with comparing the theoretical two-level asymptotic convergence factor ($\overline{\lambda}$ - see §4.1) with the experimental one by an algorithm which **simulates a two-level algorithm**. This is done by returning from the next coarser grid H back to the finest grid h only when the H equations have been solved to a very good accuracy (e.g., by taking large γ or very small δ in the cycles of §6.2). In this way we still separate away questions particular to too-coarse grids or related to three or more levels (delaying them to §6).

Another major simplification is to experiment first with particularly **convenient known solutions**. Even for complicated nonlinear systems, one can engineer the tested problem so that it has a *known* solution u. This is done simply by planting suitable right-hand sides, both for the interior differential equations and for the boundary conditions. (Even if the original problem is homogeneous, the program should anyway be written for *general* right-hand sides, because non-vanishing right-hand sides on *coarser* grids are obtained from finer grids residuals.)

For many nonlinear problems it is especially useful to experiment with solutions of the form $u = u_0 + \eta u_1$, where u_0 is a constant (or a constant vector, if u is a vector of functions), and $\eta \ll 1$ is employed in the first experiments. Taking u_0 as the first approximation for a cycling algorithm, the behavior of the solution process should almost identically (identically in the limit $\eta \to 0$) be the same as for a linear problem with constant coefficients. For such problems precise comparison can be made with mode analysis. The comparison can be pushed to be even more precise by choosing u_1 to be a particular Fourier component.

Afterwards, η can gradually be increased to study the effect of nonlinearity. This effect is in this way easily and precisely separated away from other effects, including effects which are often confused with nonlinearity because they appear in nonlinear terms; e.g., convection, whose relative magnitude in fluid problems depend on the solution itself. (See §8 for the algorithm used in nonlinear cases.)

Debugging of multigrid programs can generally benefit from relations between the levels. Most bugs and conceptual errors immediately show as irregular behavior in the standard multigrid output (listing the history of the dynamic residual norms for every relaxation sweep on every level, as in §1.5). A preliminary error-detection table, based on that output, is provided in [B$^+$78, Lecture 18]. *Troubles related to the treatment of boundaries often show in the following way*: The first couple of cycles exhibit the expected (interior) convergence factor, since the relative weight of errors near the boundaries is small. Later, however, the errors near the boundaries start to dominate and the convergence factor degrades. The coarser is the basic (finest) grid, the sooner this degradation appears.

5.2 Interpolation near boundaries and singularities

The coarse-to-fine interpolation of corrections $I_H^h v^H$ should use the boundary conditions on v^H even when they are not explicitly shown on the grid (sometimes they are only implicit in the program). Otherwise extrapolation formulae would be needed for I_H^h, giving slower asymptotic convergence factors [Oph78]. Exception is the case of discontinuity on the boundary, such as a boundary layer thinner than the meshsize, in which case boundary data should *not* be used (see §4.6).

Near boundary singularities, such as reentrant corners, the interpolation can be improved by using the asymptotic behavior, whenever known. That is, if the correction v^h to be interpolated from the coarser grid is expected to be of the form $v^h = w^h \psi$, where ψ is a known singular function and w^h is smooth, then polynomial interpolation should be used to interpolate w^h, not v^h. But such improvements are hardly needed (see §5.7).

5.3 Relaxation on and near boundaries

Except for some simple Dirichlet problems, discrete boundary conditions should generally be relaxed and transferred to the coarser grid in the same way interior difference equations do. It is important to notice that the boundary relaxation may spoil very much the' smoothness of interior residuals near the boundary. Indeed, for a smooth error function, the interior residuals formed near the boundary by relaxing the boundary conditions are $O(h^{l-m})$ times the typical magnitude of other interior residuals, where m is the order of the interior differential equation and l is the order of the boundary condition (usually $l < m$).

One way around this difficulty is immediately realized by looking at the one-dimensional case. It is clear in that case that boundary conditions need not be relaxed at all. Their errors are not functions that can be smoothed out in any way; they are just isolated values, which can always very well be represented on the coarser grid.

Analogously in higher dimensional cases, the role of relaxation should not be to impose the boundary conditions, but only to smooth their error *along* the boundary. Instead of Gauss-Seidel-type relaxation for the boundary condition $Bu = g$, say, one can make a Gauss-Seidel relaxation of the $\Delta_s Bu = \Delta_s g$, where Δ_s is an approximation to the Laplace operator along the boundary; e.g., in two-dimensional problems, $\Delta_s = \partial/\partial s^2$, where s is the boundary arclength.

In practice this means that, instead of satisfying the given condition at each boundary point, we only change its error to be equal to an average of the errors at neighboring boundary points. This increases the above l by 2, making the perturbation to the interior smoothness negligible. In case the boundary smoothing factor is not as good as the interior one, a couple of boundary sweeps may be performed per each interior one.

Another way around the above difficulty is to ignore it and rely on more precise residual transfers (§5.4). This however is cumbersome, hence not used often. A general practical way to obtain sufficient smoothing on and near the boundary is to apply there a robust block relaxation scheme (box relaxation – §3.4) and to employ more sweeps near the boundary, according to the Local Relaxation Rule (§5.7).

5.4 Residual transfers near boundaries

Relaxation seldom leaves smooth residuals near the boundaries, where the normal sequence of relaxation steps breaks off (in lexicographic schemes, for example). This is especially the case when relaxation of boundary conditions is not done in the manner explained in §5.3. Thus, in many cases it is important that each individual fine grid residual is correctly represented on the coarse grid. This is what we called *full* residual weighting. The full weighting near boundaries, and also near interfaces, is considerably more complicated than the interior full weighting (described in §4.4). This is because the influence of the residual on the solution depends on its distance from the boundary; e.g., in Dirichlet problems for m-order elliptic equations, the influence is proportional to the $(m/2)$-*th* power of the distance.

Thus the weight used in transferring a residual from a fine-grid point to a coarse-grid point depends on the distance of both points from the boundary. Near boundary corners the dependence is even more involved. Hence, near boundaries the interior full-weighting rule (4.5) is modified to the requirement that

$$\sum_{x^H} \left(I_h^H r^h\right)(x^H) W^H(x^H) G(x^H) = \sum_{x^h} r^h(x^h) W^h(r^h) G(x^h) \qquad (5.1)$$

is satisfied for any given $r^h(x)$, where $\sum f(x^H)W^H(x^H)$ and $\sum f(x^h)W^h(x^h)$ are discrete approximations, on grids H and h respectively, to the integral $\int f dx$ for any function f, and where $G(\xi)$ has the behavior of the Green function near the boundary. That is, for two neighboring ξ_1 and ξ_2, the ratio $G(\xi_1)/G(\xi_2)$ roughly gives the ratio between the solutions of $Lu(x) = \delta_{\xi_1}(x)$ and $Lu(x) = \delta_{\xi_2}(x)$, with homogeneous boundary conditions. Usually one can take $G(\xi) = d_\xi^\alpha$, where d_ξ is the distance of the point ξ from the boundary, and $\alpha = m - l - 1$, where l is the order of the highest normal derivative in the neighboring boundary condition.

Relation (5.1) need not of course be kept very precisely. Residual weighting I_h^H that deviate from it by 20% may still easily exhibit the same convergence rates. Another way of deriving residual weighting near boundaries is by variational rules, like (4.10) in essentially-symmetric cases. Still other ways exist. It may all seem complicated, but, as explained in §11, it is in principle no more complicated than discretizing the original differential equations near the boundaries.

5.5 Transfer of boundary residuals

Residuals are defined and are transferred (with some averaging) to the coarser grid H, not only with respect to the interior equations, but also with respect to the boundary conditions. Boundary residuals of grid h are averaged to form the right-hand side of the corresponding boundary conditions of grid H. In order to do it in the right scale, the *divided* form of the boundary conditions (the form analogous to the differential conditions, without multiplying through by a power of h) should be used to calculate residuals, average them and transfer. For this purpose a clear conceptual separation should be made between boundary conditions and neighboring interior equations. Incorporating the former into the latter is often convenient to do, but it may easily lead to wrong transfers. (To do it right, one should assume the given boundary condition is incorporated on the finest grid, while the corresponding homogeneous condition is incorporated on all coarser grids. Even when correctly done, however, this is equivalent to *imposing* the boundary condition at relaxation, which, as explained in §5.3, will sometimes result in large neighboring residuals and hence slower convergence, unless more precise residual weighting is used.) In symmetric problems one can consistently use the variational relation (4.10) without ever distinguishing between interior equations and boundary conditions, provided good interpolation I_H^h is defined. For some classes of problems this interpolation may be based on the difference equations, interior and boundary alike [Den82a].

5.6 Treatment and use of global constraints

In addition to boundary conditions many problems also specify some global conditions, such as integral relations, etc. For example, the pressure $p(x)$ in incompressible Navier-Stokes equations is determined only up to an additive constant; for its unique determination one should add an integral condition like

$$\int p(\underline{x})d\underline{x} = 0 \tag{5.2}$$

(integrating over the entire flow field), or a pointwise condition such as

$$p(\underline{x}_0) = 0\,. \tag{5.3}$$

Both conditions are in fact "global" in the sense we like to consider here, even though (5.3) does not look so global: One should generally consider as global any single discrete condition which has a large global effect on the solution. Boundary conditions in one-dimensional problems are also of this type (cf. §5.3). The normalization condition $(u,u) = 1$ in eigenproblems is a nonlinear condition of this type. In that case, unlike (5.2) or (5.3), together with the global condition there also comes a global unknown, the eigenvalue. This is often the case. The one-dimensional boundary conditions are associated with unknown boundary values, which should be considered as global unknowns.

A very useful device is indeed to *add global constraints to a problem to make it better posed, freeing as many global parameters.* For example, a physical continuation parameter γ for solving a nonlinear problem (cf. §8.3.2) may be a bad parameter near some "limit points"; i.e., with fixed the problem is ill-posed (or nearly ill-posed, hence not approximable on coarse grids). By converting γ into an unknown and adding instead another global characterization of solutions (e.g., the arclength along the continuation path, as in [Kel77]), the problem is made well posed (and, in particular, the coarse grid can well approximate fine-grid errors). Or one can free some accuracy parameters (e.g., allow a small constant error to be added to all equations) and add some known global information (e.g., total mass, or energy, or vorticity, when solving a time-step problem as part of a long-range evolution where such quantities must be strictly conserved). A particularly useful device is to free such accuracy parameters only on coarser grids, using global constraints only for coordinating solutions on those grids with the current fine-grid solution. In this way for example the above bad physical parameter γ can still be fixed on the finest grid. Still another example of an added constraint with an added parameter (α) appears in §5.7.

An important advantage of the multigrid processing is the easy and natural way with which such global conditions and global unknowns can be handled. To be sure, it is often done in a wrong way: For example, misguided by the practice in relaxation solvers, one would tend to treat (5.3) at the relaxation phase. Imposing such a pointwise global condition just by changing p at x_0 is really harmful to the multigrid solution, since it frustrates the error-smoothing processes near x_0.

Global conditions need not be treated at all on the fine grid. There can be no error-smoothing related to such single conditions. All one has to do is to transfer the residual of the condition to serve as the right-hand side for a similar condition on the next coarser grid. In case of a nonlinear condition, FAS should be used (§8.3). A condition like $(u^h, u^h) = b^h$, for example, will be transferred by FAS to the condition $(u^H, u^H) = b^H$, where

$$b^H = b^h + (I_h^H u^h, I_h^H u^h) - (u^h, u^h), \tag{5.4}$$

which is a special case of (8.5). The global nature of a condition like (5.3) becomes increasingly transparent as it is transferred to coarser grids by proper approximations.

The global condition must of course finally be enforced while solving the coarsest-grid problem (cf. §6.3). Likewise, global unknowns should usually be changed only on the coarsest grid (thereby matching the number of unknowns to the number of equations therein). Sometimes the global equation should be operated *on several* of the coarsest grids. For example, approximations to a condition like $(w, u) = b$, where w is a given weight function which changes signs in the domain, must perhaps be operated on a grid fine enough to resolve these sign changes (or at least crudely simulate them). Similarly, the condition $(u^H, u^H) = b^H$ should be operated on a grid fine enough to crudely resolve the sign changes in the solution u.

When a global condition *is* treated in relaxation (on a coarse but not the coarsest grid, for example) this should be done in a global way. For example, the condition should be satisfied (at the end of each sweep, say) by adding a constant (or a smooth function) to the entire solution, or by multiplying the entire solution by a constant (or a smooth function), so that the error-smoothing process is not frustrated.

There are sometimes conditions which are neither completely global nor quite local, but have some intermediate scale. For example, sometimes, when one global control would not do, several constraints are added, each controlling the solution on one subdomain. Any such condition should not be treated in relaxation wherever the meshsize is small compared with the scale of the condition.

In some relaxation schemes, the global condition seems to be needed in the local relaxation. For example, in the BGS schemes (§3.4) one solves in small boxes little problems similar to the given boundary-value problem. For the solution in the box to be uniquely determined, a condition like the global condition is needed there. In solving discrete incompressible Navier-Stokes equations in a small local box, for example, a pressure condition similar to (5.2) or (5.3) is needed. The best then is to use in each box a "no. change" kind of condition. That is, to require, for example, that some discrete approximation to $\int p(x)dx$ (integration being over the small box) retains its value from before the relaxation step.

5.7 Structural singularities. Reentrant corners. Local relaxation

Structural singularities are singularities in the problem other than those caused simply by singular forcing terms (singular right-hand sides in either the differential equations or the boundary condition. These are discussed in §7.1.). Boundary reentrant corners and singularities in the (left-hand side) differential operators are structural singularities. Such singularities cause the asymptotic multigrid convergence factor to degrade (although it is still bounded away from 1 independently of the meshsize). The reason is that the error components slow to converge in relaxation, which are approximate solution to the homogeneous differential equations (cf. §4.6), have a singular behavior around the structural singularity, therefore they are not well approximated by a coarse-grid correction, unless this singular behavior is taken into account in formulating I^h_H, L^H and I^H_h (see §5.2 and §4.6).

It is easy to overcome this difficulty in several other, perhaps simpler, ways. First, the degradation of the asymptotic factors may not be a two-level feature at all, appearing only because errors of the above type accumulate by cascading through many levels. Indeed, the degradation almost disappeared (in cases of reentrant corners studied in [BB87]) when W cycles replaced V cycles. Secondly, even with V cycles, the degraded asymptotic factors little affect the ability of the FMG algorithm to solve the problem to within truncation errors by just one cycle per level. This is explained by the fact that the same singularity causing the degraded convergence also causes large truncation errors, in exactly the same components which are slow to converge. Thus, exactly these components need not converge much in order to be approximated below truncation error. Numerical experiment with reentrant corners [Oph78], [BB87] show this to be true, unless the number of levels is very large indeed; this usually occurs when local refinements are used.

Finally, a general way to deal with structural singularities, so that even the many-levels-V-cycle convergence factors are not degraded at all, is by *local relaxation sweeps*, i.e., special relaxation passes over a small number of points near the singularity. Experiments with reentrant corners [BB87] show that just one such pass over a small number (typically less than 1%) of points before each full sweep is enough to completely restore the interior (i.e., regular) convergence factors. In fact, there is no need to explicitly identify singularities if the following more general rule is adopted.

Local Relaxation Rule: *add local relaxation steps wherever, and as long as, exceptionally large normalized residuals are detected at a small number of points.* The normalized residual is the error in satisfying a discrete equation, divided by the l_1 norm of the equation coefficients. (This general rule is justified by the theory of §1.1, and by the unified exchange-rate approach for controlling multigrid processes discussed in §9.6.)

Reducing Truncation Errors. Another question of course is how to obtain small truncation errors (i.e., a good approximation to the differential solution) near structural (and other) singularities. Two general ways are local refinements and subtraction of the singularity. The former is a very general approach, discussed in §9; the latter can be used when the singular behavior of the solution u is known and simple. For example, near a reentrant corner $u = \alpha\psi + w$, where ψ is a known singular function (e.g., $\psi = r^{\gamma}\sin(\gamma\theta)$ in case of Poisson equation and a corner of π/γ radians), α is an unknown constant, and w is an unknown *non*-singular function. The procedure then is to rewrite the problem in terms of w. In the new problem, α serves as a global unknown, and a new constraint should be added to express the requirement that w is smooth at the singularity. (In the this example, an inward derivative near the reentrant corner may be required to vanish.) This constraint is "global", since it controls the size of α, so its treatment, and the determination of α, should follow the rules in §5.6. When the singularity is subtracted off like this, the degraded convergence factor discussed above should disappear.