Chapter 3

Interior Relaxation and Smoothing Factors

The crucial step in developing multigrid solvers is the design of interior relaxation schemes with high error-smoothing rates. Namely, the crucial question is how to reduce non-smooth error components for as little computational work as possible, neglecting interactions with boundaries. This is the crucial question, from the point of view of solution efficiency, because reducing *smooth* error components will require less computational work (being done at coarser grids), and because reducing non-smooth error components *near boundaries* will require much less work as it involves local work only near the boundary (which is a lower-dimensional manifold). Also, relaxation is the most problem-dependent part of the algorithm – other parts are usually quite standard (the relaxation of boundary conditions is discussed in §5.3).

3.1 Local analysis of smoothing

To reduce non-smooth error components is basically a *local* task; it can be done in a certain neighborhood independently of other parts of the domain. This is why it can be efficiently performed through relaxation, which is basically a local process (the information propagates just few meshsizes per sweep). Hence also, the efficiency of this process can accurately be measured by local mode analysis.

That is, one can assume the problem to be in an unbounded domain, with constant (frozen) coefficients, in which case the algebraic error $u^h - \tilde{u}^h$ (where u^h is the exact solution to the discrete equations and $\tilde{u}^h$ is the computed approximation) is a combination of Fourier components $e^{i\underline{\theta}\cdot\underline{x}/\underline{h}}$. For each such Fourier component and any proposed relaxation scheme, one can easily calculate the amplification factor $\mu(\underline{\theta})$, defined as the factor by which the amplitude of that component is multiplied as a result of a

relaxation sweep (see simple examples in §1.2 above and in [ST82, §3.2]). The **smoothing factor** $\overline{\mu}$ of the relaxation scheme, defined by

$$\overline{\mu} = \max_{\frac{\pi}{2} \le |\underline{\theta}| \le \pi} |\mu(\underline{\theta})| \tag{3.1}$$

can then be easily computed, usually by a standard computer program (see for example [Wei01]). This is indeed the measure we need: $\overline{\mu}$ is the worst-case (largest) factor by which all high-frequency error components are reduced per sweep, where we define the frequency to be high if the component is not visible (aliases with a lower component) at the next coarser grid (grid $2h$).

In case of a **system** of q grid equations in q unknown grid functions (i.e., q unknowns and q algebraic equations are defined per mesh cell), each Fourier amplitude is a q-vector, hence $\mu(\underline{\theta})$ is a $q \times q$ amplification *matrix*. $\overline{\mu}$ is still defined as in (3.1), except that $|\mu(\underline{\theta})|$ is replaced by $\rho(\mu(\underline{\theta}))$, where $\rho(\mu)$ is the spectral radius of μ.

For L^h with **non-constant coefficients**, $\overline{\mu}$ defined by (3.1) depends on the location. In case of a **nonlinear** L^h, the analysis is made for the linearized operator, hence $\overline{\mu}$ also depends on the solution around which linearization is made. The quality of relaxation is then determined by the worst $\overline{\mu}$, i.e., the maximum $\overline{\mu}$ over all possible coefficients of L^h for any solution which may evolve in the calculations (one may disregard $\overline{\mu}$ over small regions: see §3.3).

A major simplification in calculating $\overline{\mu}$ for complicated systems is to look at **subprincipal terms only** (see §2.1 and 3.4, and examples in §19.2 and 20.2).

Some relaxation schemes do not transform each Fourier component of the error to a multiple of itself. Instead, they couple several (l, say) Fourier components at a time (even for an infinite domain). For example, if relaxation is performed in red-black (checker-board) ordering (cf. §3.6), the $\underline{\theta}$ component is coupled to the $\underline{\theta} + (\pi, \ldots, \pi)$ component. Instead of the $q \times q$ amplification matrix $\mu(\underline{\theta})$, we then have the $(ql) \times (ql)$ matrix $\mu(\underline{\theta}^1, \ldots, \underline{\theta}^l)$, describing the transformation of the l q-vector amplitudes corresponding to the coupled components $(\underline{\theta}^1, \ldots, \underline{\theta}^l)$. Definition (3.1) is extended to such cases by *defining*

$$\overline{\mu}_\nu := \max \left[\rho \left(C \left(\underline{\theta}^1, \ldots, \underline{\theta}^l \right) \mu \left(\underline{\theta}^1, \ldots, \underline{\theta}^l \right)^\nu \right) \right]^{\frac{1}{\nu}}, \tag{3.2}$$

where the max is taken over all coupled l-tuples $(\underline{\theta}^1, \ldots, \underline{\theta}^l)$, C is an $l \times l$ matrix of $q \times q$ blocks C_{ij}, such that $C_{ij} = 0$ for $i \neq j$, $C_{ii} = I_q$ (the $q \times q$ identity matrix) if $|\underline{\theta}^i| \geq \frac{\pi}{2}$, and $C_{ii} = 0$ otherwise. ν is the number of sweeps performed at the finest grid per multigrid cycle; only in the simple case ($l = 1$), $\overline{\mu}$ does not depend on ν. For examples with $l > 1$, see [Bra81b, §3.3], [Lin81, §2.3.1].

The smoothing factor is the first and simplest quantitative predictor of the *obtainable* multigrid efficiency: $\overline{\mu}^\nu$ (or $\overline{\mu}_\nu^\nu$) is an approximation to the asymptotic convergence factor obtainable per multigrid cycle. Usually this prediction is more accurate than needed. There are still more accurate predictors (see §4.1). But *the main importance of $\overline{\mu}$ is that it separates the design of the interior relaxation from all other algorithmic questions. Moreover, it sets an ideal figure against which the performance of the full algorithm can later be assessed* (see §§4, 5).

The analysis of relaxation within multigrid is thus much easier than its analysis as an independent iterative solver. The latter is not a local process, and its speed depends on smooth components badly approximated by mode analysis due to boundaries and variable coefficients. For multigrid purposes, however, wherever the equations (or their linearized version) do not change too much within few meshsizes, the smoothing factor can be used as a standard measure of performance. A general computer program for calculating $\overline{\mu}$ is described in [Wei01].

3.2 Work, robustness and other considerations

In comparing several candidate relaxation schemes we should of course take into account not only their smoothing factors, but also the amount of work per sweep. The aim is generally to have the best high-frequency convergence rate per operation, i.e., the largest $w_0^{-1}\log(1/\overline{\mu})$, where w_0 is the number of operations per gridpoint per sweep. But other considerations should enter as well: The rate should be robust, that is, $\overline{\mu}$ should not depend too sensitively on problem parameters or on a precise choice of various relaxation parameters. Also, between two schemes with similar values of $w_0^{-1}\log(1/\overline{\mu})$ but with very different w_0, the simpler scheme (where w_0 is smaller) should be preferred, because very small factors $\overline{\mu}$ cannot fully be obtained in practice (owing to the inability of the coarse-grid correction to obtain such small factors for the smooth components, and owing to interactions with boundaries). Moreover, large values of w_0 leave us with less flexibility as to the amount of relaxation work to be performed per cycle. Very small $\overline{\mu}$ may in fact be below what we need in the Full Multigrid (FMG) algorithm (see §7).

An important consideration, sometimes overlooked, is that each relaxation sweep should of course be **stable**. The most familiar schemes *are* stable, but distributive schemes (§3.4) for example, can be unstable exactly in cases showing the best $\overline{\mu}$. A trivial example: satisfy each difference equation in its turn by changing its *latest* unknown (in the sweeping ordering) instead of its usual corresponding unknown. $\overline{\mu}$ will then vanish, but the process will be unstable. Stability analysis can in each case be performed as Von-Neumann analysis for time dependent problems, taking the main relaxation marching direction as the timelike direction.

Also, let us not forget that relaxation has a certain **effect on smooth**

(low frequency) components, too. Usually this effect is slow: $\mu(\underline{\theta})$ is close to 1 for small $|\underline{\theta}|$. But sometimes schemes which show spectacularly small values of $\overline{\mu}$ also show either bad divergence ($|\mu(\underline{\theta})| \gg 1$) or fast convergence ($|\mu(\underline{\theta})| \ll 1$) for *low* frequencies. This for example may happen in relaxing hyperbolic (relative to some time-like direction) equations using upstream differencing and marching with the stream (the time-like) direction. Schemes with bad divergence should clearly be rejected (see an example in §20.3.4, the super-fast smoothing case). Those with fast convergence may also have some disadvantage (in case high-order corrections, as in §10.2, are desired; see for example [Bra81a, §2.2]).

It is therefore advisable to add to the program of calculating $\overline{\mu}$ also a routine for checking the stability of the scheme examined, and to calculate, together with (3.1), also the value of $\max_{|\underline{\theta}|\leq\pi} |\mu(\underline{\theta})|$. It is also useful to calculate weighted mean squares of $\mu(\underline{\theta})$ for high-frequency $\underline{\theta}$'s. Such quantities predict the error decrease in a given number of multigridded relaxation sweeps for a given initial error [BD79, §4.5]. Some of these measures are listed in [Wei01].

The value of local mode analysis becomes dubious at places of strong discontinuities, e.g., where the coefficients of the differential equation change their order of magnitude discontinuously (or within few meshsizes). This usually happens along manifolds of lower dimensionality, therefore more computational work per gridpoint can there be afforded, hence an accurate measure of efficiency is not so needed. But some basic rules, outlined below, must still be followed. One can also employ local R–E analysis (cf. §1.1 and more details in [Bra86]), which yields good quantitative information even in strongly discontinuous cases.

3.3 Block relaxation rule. Semi smoothing

The most basic rule in devising relaxation schemes is that *a locally strongly coupled block of unknowns which is locally decoupled from (or weakly coupled with) the coarser-grid variables, should be simultaneously relaxed.* The reason is that a point-by-point relaxation smoothes only along the strongest couplings, whereas block relaxation also smoothes along second-strongest couplings (provided the strongest ones are included in the blocks. See for example the case $a \ll c$ in §1.2, and a more general analysis in [Bra86, §3.5, 4.6].

This rule is of course important whether or not the equations are continuous. In the case of **persistent S_1-h-ellipticity** (i.e., a difference operator with good S_1-h-ellipticity throughout a substantial subdomain, but without uniformly good S_2-h-ellipticity measure for any $S_2 \not\subset S_1$) , the rule implies either the use of block relaxation in suitable directions (line relaxation, plane relaxation, etc.), or the use of suitable "semi coarsening" (see below), or both.

Generally, for any set S of grid directions (usually $S \subset \{1, 2, \ldots, d\}$

, but sometimes including a special bisecting direction could be advantageous), a relaxation scheme is called an **S-block relaxation** if it relaxes simultaneously all (or many contiguous) equations defined on the same S-subspace. (Two points $(x_1, \ldots, x_d)$ and $(y_1, \ldots, y_d)$ are in the same S-subspace if $x_j = y_j$ for all $j \notin S$.) For example, the line relaxation in §1.2 is a y-block (vertical line) relaxation.

Semi coarsening, or more specifically **S-coarsening**, means that $H_j = 2h_j$ for $j \in S$, and $H_j = h_j$ otherwise, where H_j and h_j are the mesh-sizes of the coarse grid and the fine grid, respectively, in the x_j direction $(j = 1, \ldots, d)$. In such a coarsening, we need to smooth the error only in directions S. The definition of the smoothing factor should accordingly be modified. We generalize (3.1) to any coarsening situation by defining

$$\overline{\mu} := \max \left\{ \rho(\mu(\underline{\theta})) : \ |\underline{\theta}| \le \pi, \max_{1 \le j \le d} \frac{|\theta_j| H_j}{h_j} \ge \pi \right\}. \tag{3.3}$$

Similarly we generalize (3.2) by defining $C_{ii} = I_q$ if $\max |\theta_j^i| H_j / h_j \ge \pi$, and $C_{ii} = 0$ otherwise. The **S-smoothing factor** is defined as (3.3), or the generalized (3.2), for S-coarsening. (An improved $\overline{\mu}$ definition that allows the constant π in (3.3) to be replaced by a smaller positive number, is explained in §12.)

If *point* (not block) relaxation is to be used, then S-h-ellipticity defined in §2.1 is a necessary and sufficient condition for the existence of relaxation schemes with good (i.e., bounded away from 1) S-smoothing factors. This is an easy generalization of a theorem proved in [Bra80b, §4.2]. The more general situation, with *block* relaxation, is summarized by the following theorem:

Theorem 3.1. *Let S and S' be two sets of directions: $S, S' \subset \{1, \ldots, d\}$. A necessary and sufficient condition for the existence of an S-block relaxation scheme with good S'-smoothing rates is that the discrete operator L^h is uniformly coupled in S' modulo S; that is,*

$$E^h_{S',S}(L^h) := \min_{\substack{\frac{\pi}{2} \le |\underline{\theta}|_{S'} \le \pi \\ \theta'_j = \theta_j \text{ for } j \in S}} \frac{\tilde{L}^h(\underline{\theta})}{\tilde{L}^h(\underline{\theta}')} = O(1). \tag{3.4}$$

$E^h_{S',S}(L^h)$ is called the measure of uniform coupling in S' modulo S. The theorem states, in other words, that the S'-smoothing factors, produced from L^h by any *suitable* S-block relaxation, are bounded away from 1 by a quantity which depends only on $E^h_{S',S}(L^h)$.

In this context, the role of block relaxation can sometimes be played by simple point relaxation. This is when the relaxation marching direction conforms with the downstream time-like direction of a hyperbolic-like system, so that a relaxation sweep nearly *solves* the equations (especially

when upstream differencing is used). The role of block relaxation can also be played, more automatically and in more situations, by ILU smoothers (see §3.8).

Variable coefficients. When the coefficients of L^h (or of its linearization, in case it is nonlinear) are not constant, a *perfect smoother* is a relaxation scheme whose formal $\overline{\mu}$ (calculated at each point by assuming the coefficients there to extend as constant coefficients throughout) is good at all points. Such a perfect smoother can sometimes require quite costly block relaxation (e.g., plane relaxation) in several directions, because of varying semi-h-ellipticity directions. It is therefore important to realize that *such perfect smoothers are not really needed* because *accidental* semi h-ellipticity need not be taken into account. Explanation:

First, the above block-relaxation rule itself suggests that $\overline{\mu}$ may be allowed to be bad (close to 1) at some isolated points.

The multigrid convergence rates will still be good. More importantly, however, consider the common case of a (nearly) non-elliptic differential operator whose (sub)characteristic directions continuously vary over the domain, so that in some particular small region they happen to approach some grid directions. As a result, in that particular region the discretization may be semi h-elliptic, smoothing there will be bad, hence the asymptotic multigrid convergence will slow down. Notice, however, that the components slow to converge are very special ones: They are necessarily high-frequency characteristic components in that particular region; i.e., components smooth in the (sub)characteristic directions but not smooth in all directions. Such components exist on the grid *only* when (sub)characteristic directions approach grid directions. Elsewhere, such components are not represented at all by the finite-difference solution; they are truncated. Hence, if one is interested in solving the discrete equations only to the level of truncation errors (and hence using an FMG algorithm – cf. §7), such components can be ignored: Their slow algebraic convergence in regions of *accidental* semi h-ellipticity does not matter, because similar components are not approximated at all in other regions.

Only in cases of **strong alignment** (see §2.1) the corresponding block relaxation must be used. But it may be confined to the region of strong alignment. If, for example, the strong alignment is due to grid alignment of boundary layers, it is enough to perform line (or plane) relaxation only at the very lines adjacent to such boundaries (and sometimes not even there – see [Bra81a, §3.3]), with just point relaxation elsewhere. If the alignment is strong because it occurs throughout a major subdomain, line (or plane) relaxation of only that special direction is needed there. Alternating-direction block schemes may be needed only if errors far below truncation errors are for some reason desired.

A general way to avoid any need for block relaxation, even when solving far below truncation errors, is to use semi coarsening, as mentioned above. In the case of variable coefficients, causing variable coarsening di-

rections, this leads to AMG processes (see §13.1).

3.4 Distributive, weighted, collective and box Gauss-Seidel. Principal linearization

To obtain efficient smoothing, a selection should be made from an abundance of available relaxation schemes. The choice depends on experience and on some physical insight, with $\overline{\mu}$ calculations serving for final quantitative judgement. We list here some important types of schemes. Each of those can be operated pointwise or blockwise (see §3.3) and in different orderings (§3.6). Some simple schemes are described in more detail in [ST82]. We first describe successive displacement schemes, then we mention their simultaneous-displacement counterparts (§3.5). In the case of equations with many or complicated linear or nonlinear lower-order terms, it may pay to apply any of these schemes with the scaled principal terms only (see §10.3). In the case of complicated *systems*, see the general approach in §3.7.

The most basic scheme is the **Gauss-Seidel** (GS) scheme , in which all the discrete equations are scanned one by one in some prescribed order. Each of them in its turn is satisfied by changing the value of one corresponding discrete unknown. This is easy to do if the problem is linear and if there is a natural one-to-one correspondence between equations and unknowns (i.e., if the matrix of coefficients is definite or is approximately definite; see §6.3). If the problem is **nonlinear**, each discrete equation may be a nonlinear equation in terms of the corresponding unknown. It is then usually best to make just one Newton step toward solving each equation in its turn. This **Gauss-Seidel-Newton** (GSN) scheme is not related to any global linearization of the system of equations, it just linearizes one discrete equation in terms of one discrete unknown, yielding usually a very simple scheme that does not require any storage other than the storage of the (approximate) solution.

Principal linearization. Moreover, it is actually enough to relax an equation through an approximate linearization of its h-principal terms, corresponding to the (sub)principal terms of the differential operator (§2.1). Thus, for example, if the equation is $\mu\Delta u + uu_x + \cdots$ and the current approximation just before relaxing at some point is $\tilde{u}$, then relaxation at that point can simply be the same as if relaxing the equation $\mu\Delta u + \tilde{u}u_x + \cdots$. A full linearization would in addition include the term $(u - \tilde{u})\tilde{u}_x$, but on the scale of the grid, hence in relaxation, that term is negligible. This "principal linearization" is *as good as full linearization for purposes of relaxation*, at least as long as differences of $\tilde{u}$ at adjacent gridpoints are small compared with $\tilde{u}$ itself. Note that for quasi-linear equations, which include almost all practical equations, the principal linearization involves *no linearization at all, just trivially retarding the lower-order derivatives in each term*, as in the example.

When relaxation is used as the prime solver, much may be gained by Successive Over Relaxation (SOR), in which the GS correction calculated for each unknown is multiplied by a **relaxation parameter** ω. The situation is different when relaxation is used only as a smoother in multigrid solvers. The best smoothing (lowest $\overline{\mu}$) is usually obtained for the natural value $\omega = 1$, so that GS is not only cheaper (per sweep), but also at least as effective (per sweep) as SOR. Lower $\overline{\mu}$ may be obtained by other parametrizations (e.g., the distributive GS described below), but for regular second-order elliptic equations this gain hardly justifies the extra work involved: Simple GS is probably the best known smoother (especially with red-black ordering – see §3.6).

If block relaxation is required (cf. §3.3), **block GS** can be used. This means that blocks are scanned one-by-one; the equations of each block are simultaneously satisfied by updating the corresponding block of unknowns. In the two-dimensional plane (x, y), if the blocks are lines parallel to x (constant-y lines), the relaxation is called x-Line GS (xLGS). yLGS is similarly defined (see example in §1.2).

When there is no natural one-to-one correspondence between discrete equations and unknowns (the matrix is not approximately definite; e.g., non-elliptic and singular perturbation equations, or elliptic *systems* which are not strongly elliptic [BD79, §3.6]), simple GS should be replaced either by **Distributive Gauss-Seidel** (DGS) or by Weighted Gauss-Seidel schemes. In DGS, with each discrete equation we associate a "ghost" unknown, with some prescription being selected for the dependence of regular unknowns on ghost unknowns. Usually, each regular unknown is written as a prescribed linear combination of neighboring ghost unknowns. Then, as in GS, the equations are scanned one by one, each being satisfied by changing the corresponding ghost unknown. This means in practice that a certain pattern of changes is distributed to *several* neighboring regular unknowns (hence the denomination "distributive" GS); the ghost unknowns do not explicitly appear, nor stored in any way, they just serve for the description of DGS. (In fact their values are never known – only changes in their values are calculated to induce changes in the regular unknowns.) In the case of block (e.g., line) DGS relaxation, a block of ghost unknowns is simultaneously changed to simultaneously satisfy the corresponding block of equations. In two dimensions we thus have xLDGS and yLDGS schemes. A special case of DGS is the Kaczmarz relaxation (see §1.1). Other examples are described in detail in §17.3, 18.3, 19.3 and 20.3. The smoothing analysis of DGS schemes is best executed in terms of the ghost unknowns; see §3.7.

In **Weighted GS** (WGS) schemes, with each discrete unknown we associate a ghost equation, which is a preassigned linear combination of neighboring equations, and we perform GS in terms of the ghost equations. Taking work into account, WGS is usually inferior to DGS, since each equation is calculated several times per sweep, unless the ghost equations

explicitly replace the original equations – which is just a linear transformation of the discrete system of equations. It pays to transform the system (as against performing DGS) only if the resulting system is not more complicated than the original, which is seldom the case: A transformation that yields a simpler system could usually be done in terms of the *differential* equations, giving a simpler differential system. (Exceptions are cases where the simplifying transformation gives a worse system for discretization; e.g., a system not in conservation form as in [Bra82c, §2.1]. To relax in conservation form in that case, a combination of WGS and the DGS scheme of [Bra82c, §4.1] is indeed needed.)

For systems of equations ($q > 1$), simple GS is appropriate only in case the system is strongly elliptic [BD79, §3.6]. Otherwise collective GS or DGS schemes should be employed. **Collective Gauss-Seidel** (CGS) is performed when the grid is not staggered, i.e., all the q grid equations and q unknown functions are defined on the same gridpoints: The grid points are scanned one by one, at each point we change simultaneously ("collectively") its q unknowns so as to simultaneously satisfy its q equations. In case of a staggered grid, one can divide the domain into (usually overlapping) small boxes. The boxes are scanned, for each one we change simultaneously all unknowns interior to it so as to simultaneously satisfy all equations interior to it. This is called **Box GS** (BGS). DGS schemes are generally more efficient for staggered grids than BGS (except sometimes in very coarse grids; cf. §6.3), because they do not couple the equations (see §3.7).

For *nonlinear* equations, all these methods can be used, but instead of fully satisfying an equation (or a collective of q equations, or a box of equations), only one Newton step (or just principal linearization) is made in terms of the corresponding (regular or ghost) unknown (or collective of q unknowns, or box of unknowns). For semi h-elliptic cases, block CGS (e.g., line CGS, meaning simultaneous solution of all equations on a line through changing all that line's unknowns), or block DGS, or block BGS, may be performed (after principal linearization, if needed) .

Higher-order equations are sometimes most efficiently relaxed by writing them as systems of lower order equations. For example, the biharmonic can be written as a pair of Poisson equations. Relaxing this system involves less work (per complete sweep) and yields better smoothing (per sweep) than relaxing the biharmonic. But special care should be taken in relaxing the boundary conditions for this system (see §5.3).

3.5 Simultaneous displacement (Jacobi) schemes

The GS schemes described above are *successive-displacement* schemes: The new value of an unknown (or block of unknowns) replaces the old one as soon as it is calculated, and is immediately used in relaxing the next

equations. In *simultaneous displacement* schemes new values replace old ones only at the end of the sweep, after all of them have been calculated; hence each of them is calculated explicitly in terms of old values only. Corresponding to each of the schemes above we have a simultaneous-displacement scheme, called: Jacobi-relaxation, Jacobi-Newton, distributive Jacobi, weighted Jacobi, collective Jacobi, box Jacobi, line Jacobi, line distributive Jacobi, etc. – corresponding to GS, GSN, DGS, WGS, CGS, BGS, line GS, line DGS, etc., respectively.

Unlike GS, Jacobi schemes often require **under-relaxation** ($\omega < 1$) in order to provide good smoothing. But with relaxation as a smoother (not an independent solver), good and optimal values of w are independent of the domain, and can easily be calculated by local mode analysis.

Distributive and weighted Jacobi (under-)relaxation amounts actually to the same thing. An example of an optimized weighted Jacobi scheme is analyzed in [Bra77a, §3.3].

Experience so far shows that Jacobi schemes are inferior to the corresponding GS schemes. They not only require more work (for operating the relaxation parameter) and more storage (for storing the new values separately), but their smoothing factors are in fact worse. For the 5-point Poisson equation, for example, Jacobi under-relaxation ($\omega_{\text{optimal}} = .8$) yields $\overline{\mu} = .6$, while GS gives $\overline{\mu} = .5$ and .25 for lexicographic and red-black orderings, respectively. The situation is similar in all cases so far examined. The advantage of simultaneous displacement schemes is in their being more amenable to certain rigorous analyses (but there usually seems to be little practical value to this – see §14) and their vectorizability and parallelizability (but red-black GS and similar schemes are also fully parallelizable – see §3.6).

3.6 Relaxation ordering. Vector and parallel processing

For successive-displacement schemes, the order in which the equations (or blocks of equations) are relaxed has an important effect on the smoothing factors. The main orderings used are the usual **lexicographic** (LEX) order (in which the equation at grid point $(i_1, \ldots, i_d)$ is relaxed before $(j_1, \ldots, j_d)$ if $i_k = j_k$ for $1 \leq k < l$ and $i_l < j_l$), and related orders (LEX order for some permutation of the coordinates, some of them possibly reversed); **symmetric** relaxation (lexicographic sweep followed by a sweep in the reversed order); **Red-Black** (RB) ordering (in which all "red" gridpoints are relaxed before all "black" ones, where the coloring is similar to that of a checkerboard, namely a point $(i_1, \ldots, i_d)$ is red if $i_1 + \cdots + i_d$ is odd, and black if it is even); and more general **pattern relaxation** (similar to RB, but with different coloring and possibly more colors). For difference equations involving more than nearest neighbors, RB schemes still depend

on the ordering of points within each color. If such points are displaced simultaneously, the scheme is called Jacobi-RB; similarly LEX-RB, etc. The performance (i.e., smoothing per operation) of Jacobi-RB is more like GS schemes than like Jacobi, and like them it does not generally require underrelaxation.

Each of these orderings has its block-relaxation versions. xLGS (or xLDGS) can be done lexicographically forward (increasing y) or backward (decreasing y), or symmetrically (forward alternating with backward). Or, corresponding to RB, we can first relax the even lines, then the odd lines. This is called **zebra** xLGS (or x-zebra) relaxation. Similarly, yLGS (or yLDGS) can be done upward, downward, symmetrically or zebra. Particularly robust schemes are the Alternating-Direction Zebra (ADZ = x-zebra alternating with y-zebra) and Alternating-Direction Symmetric LGS (ADS = symmetric xLGS alternating with symmetric yLGS). Many more block GS schemes are similarly defined in higher dimensions. The choice of blocks is governed by the rule in §3.3. Concerning the choice of ordering we have the following remarks.

It has been found that GS with RB ordering is the best for the 5-point Poisson equation [FST81]. Similarly, DGS with RB ordering within each of its passes (called briefly Distributive RB, or DRB) is the best for many *systems*, such as Cauchy-Riemann and compressible and incompressible Navier-Stokes equations (see §17.3, 18.3, 19.3 and 20.3). For 5-point Poisson, RB-GS provides $\overline{\mu}_1 = .25, \overline{\mu}_2 = .25$ and $\overline{\mu}_3 = .32$ (cf. (3.2)), as opposed to $\overline{\mu} = .5$ for LEX-GS. Moreover, RB-GS can be executed with only four operations per grid point, whereas lexicographic GS requires five. Similar comparisons hold for the more complicated elliptic systems.

In addition, the mentioned RB schemes (more precisely Jacobi-RB) and similar pattern relaxation schemes are fully **vectorizable and parallelizable**: All the equations of the same color can be relaxed in parallel, thus taking full advantage of computers having vector or parallel processing capabilities. The zebra schemes are similarly parallelizable. (See more about parallelization of all multigrid processes in [Bra81b].)

For non-elliptic equations or for elliptic equations with large non-isotropic lower-order terms (singular perturbation problems, in particular), the first approach [Bra76], [SB77], [Bra77a], [Bra79b] was to employ **"downstream" ordering**, in which the equation at a point A is relaxed before (or simultaneously with) that at point B if the solution at B depends more heavily on the solution at A than vice-versa (e.g., if the fluid flows, or the convection transports, from A to B). This provides very good smoothing factors (better than those for regular elliptic problems). If different "downstream" directions exist at different parts of the domain, this may require a sequence of several relaxation sweeps in several directions. If for example line relaxation is also required, ADS relaxation may be needed, i.e., four passes over the domain. Each pass may be effective in only part of the domain, but the combined sweep will give excellent smoothing everywhere,

for any combination of semi h-elliptic approximations in two dimensions (and also in three dimensions, if the grid is coarsened in only two directions (cf. §4.2.1. In some particular cases (when the reduced equation is hyperbolic in some time-like direction, and upstream differencing is employed) such schemes yield not only great smoothing but also great convergence, making coarse-grid corrections superfluous.

Our preference today, however, is away from these downstream marching schemes. First, because they are not so good for vector and parallel processing. Also, because in the cases where several downstream directions are required, the programming is complicated and the multi-direction procedure is not fully efficient, since it requires several passes over the grid where one or two (efficient) passes is all that would be needed at each multigrid stage. Hence **ordering-free** schemes were developed, with which good smoothing is obtained for any ordering, including RB and/or zebra (the block-relaxation rules should still be kept). Such ordering-free schemes are obtained either by distributive relaxation [Bra79b, §6], or by using slightly more artificial viscosity than that required for upstream-differencing [Bra80b, §4.3], [Bra81a, §5.7, 6.3, 7.2]. Actually, even these devices (distributive relaxation and/or increased artificial viscosity) are not usually needed, unless one wants a "perfect smoother" in order to reduce algebraic errors far below truncation errors: If all that matters is the fast approximation of the *differential* (not the discrete) solution and an FMG algorithm is employed (see §7), then the simplest direction-free schemes, such as RB, can do [Bra81a]. Moreover, the tendency of downstream marching schemes to yield fast *convergence* (not just fast smoothing) may sometimes be disadvantageous (see §3.2).

3.7 Principle of relaxing general PDE systems

To obtain a good smoother (i.e., a good combination of discretization and relaxation, yielding good *differential* smoothing, as defined in §12) for a given *system* of q partial differential equations $Lu = f$, it is important to understand in advance what smoothing rates are obtainable. The guiding principle here is the following.

The smoothing rate for a given PDE operator L can be as good as the smoothing rates obtainable for the factors of the subprincipal part of $\det(L)$.

Many examples are given in Part III of this Guide (e.g., study §17.3 before proceeding to the general case that follows). To explain this generally, we first show how a smoother for L can be constructed in terms of a smoother for the scalar operator $\det(L)$. One way to do it is through distributive relaxation (cf. §3.4). Such a relaxation is defined by considering the vector of unknown functions u (or their discrete counterparts) to affinely depend on a "ghost" vector of functions w, i.e., $u = Mw + u^0$, where M is a $q \times q$ matrix of differential (or finite-difference) operators, and the vector u^0 is immaterial (since we are interested in *changes* in w,

through which *changes* in u are defined; w itself is not explicitly used). In terms of w we then devise a suitable relaxation for the product operator LM. It is easy to see that the smoothing rate of this relaxation in w will automatically be taken over by the resulting distributive relaxation in u.

(Note that a variable coefficient and a derivative do not generally commute. However, interchanging their order does not change the principal part of the operator. So in fact, it is only in terms of principal or subprincipal parts that the determinant of a matrix operator and its factorization are well defined.)

One particular choice is to take M to be the transposed matrix of cofactors of L, in which case LM equals $\det(L)$ times the $q \times q$ identity operator. One can thus devise for each component of w any relaxation suitable for $\det(L)$; the corresponding distributive relaxation for u will have the same smoothing rate.

As mentioned earlier (§2.1), the only part of L which really participates in devising the smoother is the *subprincipal* part of the linearized operator; the smoothing rates obtained for L are the same as for that part. Thus, for the discussion here, we may think of L as having subprincipal terms only. In that case we can often factor $\det(L)$ into simpler factors. Typically, in many physical problems, the factors are either the Laplacian Δ or the convection-diffusion operator $\Delta + \underline{a} \cdot \underline{\partial}$. Smoothing rates for the latter are discussed for example in [Bra81a], [Ket82] and [ST82].

One general way to devise the relaxation of L in terms of the factorization of $\det(L)$ is to correspondingly factorize L itself, using the following theorem.

Theorem 3.2. *If* $\det(L) = l_1 l_2$, *where each* l_i *is a (scalar) differential operator, then one can factorize the* $q \times q$ *operator* L *into* $L = L_1 L_2$, *where* L_i *are* $q \times q$ *matrix operators such that* $\det(L_i) = l_i$.

The proof, for which we thank Anthony Joseph, is based on Theorem 5 on page 393 in [Lan65]. A nice example is the factorization of the elasticity operator

$$\begin{pmatrix} \mu\Delta + \lambda\partial_{xx} & \lambda\partial_{xy} \\ \lambda\partial_{xy} & \mu\Delta + \lambda\partial_{xy} \end{pmatrix} = \begin{pmatrix} \partial_x & \partial_y \\ \partial_y & -\partial_x \end{pmatrix} \begin{pmatrix} \lambda + \mu & 0 \\ 0 & \mu \end{pmatrix} \begin{pmatrix} \partial_x & \partial_y \\ \partial_y & -\partial_x \end{pmatrix}$$

that is indeed useful for its relaxation (through the scheme of §17.3), especially in case $\mu \ll \lambda$, where simpler schemes fail.

To relax the factorized system $L_1 L_2 u = f$, one can simply introduce the auxiliary vector of unknown functions $v = L_2 u$ and alternating relax the two systems: $L_1 v = f$ and $L_2 u = v$. The combined smoothing rate is asymptotically no worse than the worst of the rates of the two systems. (If these two rates are very different, the system with the slower rate can be relaxed more times [TOS00, App. C], [LB04].) Special care should of course be exercised in relaxing near and on boundaries (see §5.3).

In many cases there is a simpler distributive relaxation which meets the goal of the above guiding principle. It is not necessary that LM be diagonal as in the general approach above; it is enough to get LM to be triangular. Moreover, if the operators on one of the columns of M all have a common divisor, that divisor can be omitted. In this way one can often have each term on the diagonal of LM to be just one separate factor of $\det(L)$ (some factors possibly appearing in more than one diagonal term), in which case no auxiliary functions (such as v above) are needed. The relaxation schemes in §17–20 are all of this kind. Also, instead of distributive relaxation one can obtain the same goal by weighted relaxation schemes. The important upshot in any case is that thanks to the above guiding principle, *the goal is known in advance, so do not settle for any substantially slower rates.* Note that the above smoothing discussion assumes frozen operators, hence may not apply to very coarse grids.

3.8 ILU smoothers

The above list of relaxation schemes, although including some of the most efficient smoothers, does not exhaust all possibilities. Of special interest is the use of incomplete LU decomposition (ILU), and related schemes, as smoothers. Such smoothers, first introduced in [WS80], have been shown to be very robustly efficient for a wide range of 5-point and 9-point difference equations. For an extensive treatment, see [Ket82] and the more recent [SWd85].

The basic ILU process can be described as a Gaussian elimination truncated so as to preserve a certain pattern of sparsity, simply by ignoring (i.e., replacing immediately by zero) any term produced by the elimination process at any matrix position designed to remain zero (e.g., any matrix position which is originally inherent zero). In case of nonlinear equations one can apply this process with the principal linearization (see §3.4).

The robustness of the ILU smoother can be explained by its ability in many cases to automatically produce an approximation to desired block relaxations in varying grid directions. If for example the system contains any sequence of unknowns each of which is strongly coupled only to its predecessor and successor in the sequence, and if the ILU ordering of unknowns conform with the ordering of that sequence (i.e., it gives that sequence upon omitting all other unknowns), then, ignoring weak couplings, the equations of the sequence appear to the ILU process as a separate tridiagonal system, which it automatically solve simultaneously (since Gaussian elimination for a tridiagonal system does not produce new non-zero terms). This is exactly what the block relaxation rule (§3.3) requires. Provided the weak couplings do not somehow accumulate and spoil this picture. There are special situations where that may happen. Indeed, for some special anisotropic equations there are some very special high-frequency error components which are even considerably *magnified* by the simple ILU process. More advanced

"block ILU" (see [Ket82, §3.2.3]) should then be used.

One can produce systems where various sequences of strong couplings are so ordered as to contradict any ordering chosen for the ILU process. But this rarely happens in practice; especially in two dimensions, such systems are artificial. Thus, using ILU we need to worry much less about all directions of relaxation required for a perfect smoother. Note however that such a perfect smoother is usually needed only for solving far below truncation errors (see §3.3). A careful comparison, in which the total amount of operations in an FMG algorithm is counted taking into account the ILU set-up operations shows ILU schemes to be quite comparable to suitable GS schemes [Tho83]. Moreover, in many cases ILU requires much more storage: One needs to store all the non-zero matrix elements, whereas the GS schemes often require storing just the approximate solution itself (e.g., when the problem, whether linear or not, is autonomous, as in all fluid dynamics cases). Also, GS schemes (in red-black ordering) are much faster on vector machines.

Both an advantage and a disadvantage is the fact that ILU is a "package deal", automatic prescription: It tells you exactly what to do in complicated situations, near boundaries for example; but it does not allow you to change the scheme to deal with special needs, such as local relaxation near reentrant corners and other singularities [BB87], other special local treatments, separate smoothing of coupled differential equations, etc. One can however combine ILU smoothers with sophisticated distributive schemes. For example, within the distributive relaxation described in §19.3 for the Navier-Stokes equations, one can use ILU for relaxing each set of momentum equations ((19.4b) for one j) in terms of the corresponding velocity function u_j^h.