Chapter 0

Introduction

0.1 Where and why multigrid can help

The starting point of the multigrid method (or more generally, the Multi-Level Adaptive Technique - MLAT), and indeed also its ultimate upshot, is the following "golden rule":

The amount of computational work should be proportional to the amount of real physical changes in the computed system. Stalling numerical processes must be wrong.

That is, whenever the computer grinds very hard for very small or slow real physical effect, there must be a better computational way to achieve the same goal. Common examples of such stalling are the usual iterative processes for solving the algebraic equations arising from discretizing partial-differential, or integro-differential, boundary-value (steady-state) problems, in which the error has relatively small changes from one iteration to the next. Another example is the solution of time-dependent problems with time-steps (dictated by stability requirements) much smaller than the real scale of change in the solution. Or, more generally, the use of too-fine discretization grids, where in large parts of the computational domain the meshsize and/or the timestep are much smaller than the real scale of solution changes. Etc.

If you have such a problem, multi-level techniques may help. The trouble is usually related to some "stiffness" in your problem; i.e., to the existence of several solution components with different scales, which conflict with each other. For example, smooth components, which are efficiently approximated on coarse grids but are slow to converge in fine-grid processes, conflict with high-frequency components which must be approximated on the fine grids. By employing interactively several scales of discretization, multilevel techniques resolve such conflicts, avoid stalling and do away with the computational waste.

The main development of multilevel techniques has so far been limited to their role as fast solvers of the algebraic equations arising in discretizing boundary-value problems (steady-state problems or implicit steps in evolution problems). The multigrid solution of such problems usually requires just few (four to ten) work units, where a work unit is the amount of computational work involved in *expressing* the algebraic equations (see §7.3). This efficiency is obtained for all problems on which sufficiently research has been made, from simple model problems to complicated nonlinear systems on general domains, including diffusion problems with strongly discontinuous coefficients, integral equations, minimization problems with constraints; from regular to singular-perturbation and non-elliptic boundary-value problems. Due to the iterative nature of the method, nonlinear problems require no more work than the corresponding linearized problems. Linearization is thus neither needed nor usually recommended (see §8.3). Problems with global constraints are solved as fast as the corresponding unconstrained difference equations, using a technique of enforcing the constraints only at the coarse-grid stages of the algorithm (§5.6). Few work units are also all the work required in calculating each eigenfunction of discretized eigenproblems (§8.3.1). Moreover, all multigrid processes can be fully parallelized and vectorized. In 1984, a model multigrid program on the Cyber 205 solved 3 million equations per second [BB83].

Beyond the fast solvers, multilevel techniques can be very useful in other ways related to stiffness. They can provide very efficient grid-adaptation procedures for problems (either boundary-value or evolution problems) in which different scales of discretization are needed in different parts of the domain (see §9). They can give new dimension of efficiency to stiff evolution problems (§16). They can also resolve the conflict between higher accuracy and stability in case of non-elliptic and singular perturbation boundary-value problems (§10.2). In addition, multi-level techniques can enormously reduce the amount of discrete relations employed in solving chains of similar boundary-value problems (as in processes of continuation), and in optimization problems (see §§13, 15), or in solving integral equations (see §8.6). They can also be used to vastly cut the required computer storage (§8.7). Feasibility studies have already established the effectiveness of the multilevel approach for all of these applications, which continue to be active research areas.

Multilevel processes can also cut, sometimes by several orders of magnitude, the computer resources needed to solve some *large systems which do not originate from partial-differential or integral equations.* The common feature in those systems in that they involve many unknowns related in a low-dimensional space; i.e., each unknown u_P is defined at a point $P = (x_1, \ldots, x_d)$ of a low-dimensional space (d is usually 2 or 3) and the coupling between two values u_P and u_Q generally becomes weaker or smoother as the distance between P and Q increases, except perhaps for a small number of particular pairs (P, Q). Examples are: the equations of multivariate interpolation of scattered data [Bra83, Mei79]; geodetic problems of finding the locations of many stations that best fit a large set of local observations [Mei80]; problems in transportation, economy [VDR79], and queuing theory [Kau82]; statistical problems on lattices, arising in statistical mechanics (e.g., Ising model) and in "gauge" theories of elementary particles; and various systems of tomography, image processing, picture reconstruction and pattern recognition [NOR81, San81, GG83]. There is in fact strong evidence that the human vision processes themselves are multilevelled [CR68, WB79, Ter83]. In several of these areas multigrid research has just recently started [Bra10].

0.2 About this guide (the 1984 edition)

The opening chapter of this Guide is dedicated to numerical analysts who have no previous acquaintance with multigrid methods. It also gives some references to other introductory material. (§§1.1 and 1.7 in that chapter may interest veteran multigridders, too.)

The main parts of this Guide are chiefly intended for people with some multigrid knowledge, or even experience. In fact, we were mainly motivated by the following situation, so often encountered in last few years: A good numerical analyst tries a multigrid solver on a new problem. He knows the basics, he has seen it implemented on another problem, so he has no trouble writing the program. He gets results, showing a certain rate of convergence, perhaps improving a former rate obtained with a one-grid program. Now he is confronted with the question: Is this the real multigrid efficiency? or is it many times slower, due to some conceptual error or programming bug? The algorithm has many parts and aspects: relaxation sweeps and coarse-to-fine and fine-to-coarse transfers at interior points and at points near boundaries; relaxation and transfers of the boundary conditions themselves; treatment of boundary and interior singularities and/or discontinuities; choosing the coarse-grid variables and defining its equations; the method of solving on the coarsest grid; the general flow of the algorithm; etc. A single error (a wrong scheme or a bug) in any of these parts may degrade the whole performance very much, but it is still likely to give an improvement over a one-grid method, misleading the analyst to believe he has done a good job. How can an error be suspected and detected? How can one distinguish between various possible troubles? What improved techniques are available?

The key to a fully successful code is to know in advance what efficiency is ideally obtainable, and then to construct the code gradually in a way that ensures approaching that ideal, telling us at each stage which process may be responsible for a slowdown. It is important to work in that spirit: Do not just observe what efficiency is obtained by a given multigrid algorithm, but ask yourself what is the *ideal* efficiency and *find out* how to obtain it.

To guide inexperienced multigridders in that spirit is the main purpose of this Guide.

We believe that any discrete system derived from a continuous problem is solvable "to the level of truncation errors" in just few "work units" (see §7.3). To obtain this performance, the first crucial step is to construct a relaxation scheme with a high "smoothing rate" (see §3). Then the interior inter-grid transfers and coarse-grid operator should be designed (§4), and full numerical experiments can be started with cycling algorithms, aiming at obtaining the interior rate (§§5, 6). Finally, "Full Multi-Grid" (FMG) algorithms can then be implemented, and "solvability in just few work units" can be tested (§7). These stages of development are outlined in Part I below, pointing out many possibilities and technical points, together with theoretical tools needed for quantitative insights into the main processes.

The quantitative aspect in these theoretical tools is important, since we want to distinguish between the efficiency of several candidate multigrid algorithms, all of which may be "asymptotically optimal" (i.e., solving the problem in a uniformly bounded number of work units), but some of which may still be several orders of magnitude faster than others. Except for some model problems, most present-day rigorous mathematical theories of multigrid algorithms do not give us accurate enough insights (see §14), hence the present guide will emphasize the role of "local mode analyses". These analyses (see §§2.1, 3.1, 4.1, 7.4, 7.5) neglect some of the less work-consuming processes so as to obtain a clear and precise picture of the efficiency of the more important processes. The predictions so obtained can be made accurate enough to serve in program optimization and debugging. Experience has taught us that careful incorporation of such theoretical studies is essential for producing reliable programs which fully utilize the potential of the method.

Part II of this Guide summarizes more advanced multigrid techniques and insights. Mainly, it is intended to show how to use the multilevel techniques far beyond their more familiar capacity as fast linear algebraic solvers. See the survey in §0.1 and the list of contents.

In part III we bring applications to fluid dynamics. Whereas in Part I the information about technique for all problems is ordered according to their common stages of development, in Part III we study specific problems, separately from each other. The order is again according to a certain line of development, namely, starting from simple problems and gradually learning our way to more complicated ones. The emphasis is on *systems* of differential equations; scalar problems are not separately treated.

This Guide can be viewed as an extension of an earlier work [Bra82b], with numerous updates, few new sections (§§3.8, 5.7), a new chapter (Chapter 1) and a whole new part (Part III). It is not intended just for teaching, but also for organizing and unifying the material. It is also used as an opportunity to mention some advances which have not appeared in the literature before. In particular, [Bra82b] already included some new relaxation schemes such as "Box Gauss-Seidel" (§3.4) and relaxation with only sub-principal terms (§10.3); the general rule of block relaxation (§3.3); an analysis of the orders of interpolation and residual transfers which should be used in solving *systems* of differential equations (§§4.3, 7.1); the multigrid treatment of global constraints (§5.6); an applications of FAS to obtain much more efficiency discretization to *integral* equations, leading sometimes to solutions in $O(n)$ operations, where n is the number of discrete unknowns (§8.6; [BL90]); some innovations in higher-order techniques (§10.2); unified grid switching and adaptation criteria (§9.6); multi-level approach to optimization (§13; [BR03]); and the Algebraic Multi-Grid (AMG) method (§13.1). Results from a recent work, still unpublished, on non-elliptic and singular perturbation problems [Bra81a]) are also mentioned, including a summary of stability requirements (§2.1); the double-discretization scheme (§10.2); the two-level FMG mode analysis, which tends to replace the usual two-level mode analysis (§§7.4, 7.5); the F cycle (a hybrid of V and W cycles; §6.2). (The main topic from [Bra81a] hardly mentioned here is the multigrid treatment of discontinuities, in which research is currently underway. But see §§2.2 and 8.5). The discussions on the real role of relaxation (§12), on the general approach to coarsening questions (§11), and on "algebraization" and "dealgebraization" trends in multigrid development (§13) were added as a general new viewpoints, related to each other, somewhat philosophical, but certainly useful.

This Guide includes in addition some remarks about the "principal linearization" used in relaxation (usually meaning no linearization at all – see §3.4); the general algebraic property of slowly converging relaxation schemes (§1.1); the superfluity of "perfect smoothers" for non-elliptic or slightly elliptic systems (§§3.3, 3.6); the principle of relaxing general PDE operators in terms of the factors of their subprincipal-part determinant (§3.7); some new debugging devices (e.g., §§4, 5.1); stabilizing coarsening by added global constraints (§5.6); the treatment of structural singularities such as reentrant corners (§§5.7, 9.6); and new numerical results for the Stokes and compressible and incompressible Navier-Stokes equations, including results for non-staggered grids (§§18.6, 19.5).