Chapter 8

Full Approximation Scheme (FAS) and Applications

The Full Approximation Scheme (or Full Approximation Storage - FAS) is a widely used version of multigrid inter-grid transfers, explained for example in [Bra76, §4.3.1],[Bra77a, §5],[Bra77b, §1.2], [Bra79b, §2.1],[Bra81b, §2.3]. It has mainly been used in solving nonlinear problems, but it has so many other applications that it should perhaps be used in most advanced programs. The scheme, its programming, and several of its applications are sketched below. Another, perhaps the most important application of FAS is described in §9, and yet another one in §15.

8.1 From CS to FAS

Consider first a **linear** problem $L^h u^h = f^h$ on a certain grid h, with some approximate solution $\tilde{u}^h$ obtained, say, after several relaxation sweeps. Now we like to use coarser grids in order to supply fast h approximations to smooth errors. Thus, it is the corrections $v^h = u^h - \tilde{u}^h$ which we try to approximate on the next coarser grid $H = 2h$. In the simpler multigrid programs, such as in §1.5, the coarse-grid unknown function is indeed v^H, intended to approximate the correction v^h. This multigrid version is therefore called the **Correction Scheme (CS)**. Since $L^h v^h = r^h$, where

$$r^h = f^h - L^h \tilde{u}^h \tag{8.1}$$

is the fine-grid residual, the CS coarse-grid equations are

$$L^H v^H = I_h^H r^h, \tag{8.2}$$

where L^H approximates L^h on the coarse grid. Once (8.2) has been approximately solved, its approximate solution $\tilde{v}^H$ is interpolated to the fine grid and serves as a correction to the fine-grid solution:

$$\tilde{u}^h_{\text{NEW}} = \tilde{u}^h + I_H^h \tilde{v}^H. \tag{8.3}$$

In the **Full Approximation Scheme (FAS)** we perform exactly the same steps, but in terms of another coarse grid variable. Instead of v^H we use

$$\hat{u}^H = \hat{I}_h^H \tilde{u}^h + v^H \tag{8.4}$$

as the coarse-grid unknown function, where $\hat{I}_h^H$ is some fine-to-coarse transfer which need not be similar to I_h^H in (8.2). (They are in principle defined on different spaces.) This coarse-grid variable $\hat{u}^H$ approximates $\hat{I}_h^H u^h$, the full intended solution represented on the coarse grid, hence the name "Full Approximation Scheme". The FAS coarse-grid equations, derived from (8.2) and (8.4), are

$$L^H \hat{u}^H = \hat{f}^H \tag{8.5a}$$

where

$$\hat{f}^H = L^H(\hat{I}_h^H \tilde{u}^h) + I_h^H r^h. \tag{8.5b}$$

Having obtained an approximate solution $\tilde{u}^H$ to (8.5), the approximate coarse-grid correction is of course $\underline{v}^H = \tilde{u}^H - \hat{I}_h^H \tilde{u}^h$, hence FAS interpolation back to the fine grid, equivalent to (8.3), is

$$\tilde{u}^h_{\text{NEW}} = \tilde{u}^h + I_H^h(\tilde{u}^H - \hat{I}_h^H \tilde{u}^h). \tag{8.6}$$

To use directly

$$\tilde{u}^h_{\text{NEW}} = I_H^h \tilde{u}^H. \tag{8.7}$$

would be worse, of course, since it would introduce the interpolation errors of the full solution u^H instead of the interpolation errors of only the correction v^H (but see end of §8.5). Notice that $\hat{I}_h^H \tilde{u}^h$ in (8.6) and in (8.5b) must be identically the same: A common programming mistake is to have a slight difference between the two; this difference may dominate the calculated correction function and hinder convergence. Also it is important to start with identically the same $\hat{I}_h^H \tilde{u}^h$ as the first approximation in solving (8.5).

Note: The FAS equations (8.5) are a model for each interior differential equation *or boundary condition* appearing in the problem: each of them is transferred, separately, by this prescription. Other side conditions, global constraints, etc., are transferred in exactly the same way. See for example (5.4). In case a double discretization is employed at all levels (see §10.2), two functions $\hat{f}^H$ should be calculated, one for each coarse-grid operator L^H [Bra81a, §2.1].

For linear problems, the FAS steps (8.5)–(8.6) are fully equivalent to the CS steps (8.2)–(8.3). Indeed, the safest way to construct a correct FAS program is to start with a linear subcase, write first a CS program, then convert it to FAS and check that, at each iteration, the FAS results on the finest grid are identically the same as the CS results, except for round-off errors. Common mistakes in implementing FAS, especially in treating boundary conditions, are thus easily detected. The conversion of a CS

program to FAS can be done by a trivial addition of three routines [B+78, Lecture 12]. It can be done either for a cycling program or for an FMG program. The simplest examples are the cycling program FASCC and the program FMG1 [MUG84].

8.2 FAS: dual point of view

To see why FAS is preferable to CS in many, even linear situations, we rewrite (8.5), using (8.1), in the form

$$L^H \hat{u}^H = f^H + \tau_h^H, \tag{8.8}$$

where

$$\tau_h^H := L^H(\hat{I}_h^H \tilde{u}^h) - I_h^H(L^h \tilde{u}^h) \tag{8.9}$$

and $f^H = I_h^H f^h$. Observe that (8.8) without the τ_h^H term is the original coarse-grid equation (with the particular discretization $f^H = I_h^H f^h$), and that $\hat{u}^H$ approximates $\hat{I}_h^H \tilde{u}_{\text{NEW}}^h$, and at convergence $\hat{u}^H = \hat{I}_h^H u^h$. Hence τ_h^H is the **fine-to-coarse defect correction**, a correction to the coarse-grid equation designed to make its solution coincide with the fine-grid solution.

We can now reverse our point of view of the entire multigrid process: Instead of regarding the coarse grid as a device for accelerating convergence on the fine grid, we can view the fine grid as a device for calculating the correction τ_h^H to the coarse-grid equations. Since this correction depends on the non-smooth components of the solution, we obtain it by interpolating the solution to the fine grid and correcting its non-smooth components by relaxation. Having obtained the correction τ_h^H by such a *"visit"* to grid h, we continue the solution process on grid H. Later we may *"revisit"* grid h, in order to update τ_h^H. In such a case the interpolation (8.6) should better be used if we do not want to lose the non-smooth components of the solution already obtained by relaxation in previous visits. This entire process will then yield a solution on grid h, which we can improve by inserting into it visits to grid $h/2$. Etc.

Because $\hat{u}^H$ is just an improvement to u^H, we can omit the $\hat{}$ symbol, and keep in mind that the meaning of u^H changes as soon as an approximation $\tilde{u}^h$ exists on the next finer grid h.

This point of view, and the fact that the full fine grid solution is represented on all coarser grids, open up many algorithmic possibilities, as we shall see below.

8.3 Nonlinear problems

The Correction Scheme is not applicable to nonlinear problems, since the correction equation $L^h v^h = r^h$ is valid only for linear L^h. In case L^h is

nonlinear, the correction equation can instead be rewritten using (8.1) in the form

$$L^h(\tilde{u}^h + v^h) - L^h(\tilde{u}^h) = r^h. \tag{8.10}$$

Transferring this equation to the coarse grid (replacing L^h by L^H, $\tilde{u}^h$ by $\hat{I}_h^H \tilde{u}^h$, v^h by v^H and r^h by $I_h^H r^h$), we get the FAS equations (8.5). Thus, one important advantage of FAS over CS is its direct applicability to nonlinear problems. (This is a general property of defect correction schemes – see for example [Lin76], [Ste78].)

The CS scheme can of course be applied to the Newton linearization of L^h around the current approximation $\tilde{u}^h$. But FAS is usually preferable, because:

(A) No global linearization is needed. The only linearization is the local one used in relaxation, and even this is seldom needed (see §3.4). Hence, no extra storage for coefficients of the linearized equation is required.

(B) The programming is very convenient since the FAS equations (8.5) are exactly the same as the original discrete equations, except for a different right-hand side. Hence the same relaxation routine, the same residual transfer routine, and the same boundary relaxation routine, can be reused at all levels of the program.

(C) In the linearized problems, where sight is lost of the original problem, it is much more difficult to employ the most efficient multigrid approaches: One tends to solve far below truncation errors and therefore tailor unnecessarily complicated and less vectorizable "perfect smoothers" (cf. §3.3), getting into unnecessary troubles connected with small ellipticity (cf. §7.4–7.5) or boundary singularity (§5.7). More importantly, one cannot then integrate the FMG algorithm with various processes outside the solver, such as continuation (see §8.3.2), grid adaptation (§9.6), and others (§13).

(D) The FAS multigrid rate of convergence is not constrained by the convergence rate of Newton iterations. It is still mainly determined by the interior smoothing rate. Solving the nonlinear problem is no more expensive than solving, just once, the corresponding linearized problem. (In many cases, though, Newton convergence rate is fast enough to impose no real constraint to an FMG algorithm.)

(E) FAS is useful in various other ways. Particularly important for nonlinear problems are its applications in solving chains of problems (§15), near discontinuities (§8.5) and in automatic local refinement procedures (§9).

On the other hand, Newton linearization may still be preferred in those cases where it is an essential part of the discretization process, as in some

finite-element formulations. This, however, may make such formulations less attractive in multigrid environments.

Although not employed in the FAS multigrid *processing*, we still use Newton linearizations in the local mode *analysis*, to estimate smoothing and convergence factors (§3, 4). See also §5.1 for a debugging technique related to nonlinear problems.

Examples: Various nonlinear problems have been solved by FAS, including transonic flow problems [SB77], [Jam79], [MR82]; steady-state compressible and incompressible Navier-Stokes equations (§19 and 20); the Bratu equation$\Delta u + \lambda e^u = 0$ (see §8.3.2); and complementary problems arising from free boundary problems. A simple example of the latter is to calculate the nonnegative function u which minimizes the Dirichlet integral $\int(\underline{\nabla} u \cdot \underline{\nabla} u + 2fu)dx$. without the $u \geq 0$ constraint the problem is of course equivalent to a problem with the Poisson equation $-\Delta u = f$. But the nonnegativity constraint introduces nonlinearity. Using a FAS-FMG algorithm this nonlinear problem is solved with essentially the same amount of work as Poisson problems [BC83].

8.3.1 Eigenvalue problems

Eigenvalue problems can simply be regarded as nonlinear problems. They are nonlinear since the unknown eigenvalue λ_j multiplies the corresponding unknown eigenfunction u_j. Also, to fix the eigenfunctions, nonlinear orthonormality conditions $(u_i, u_j) = \delta_{ij}$ are added as global constraints. The solution algorithm proceeds as a usual FAS-FMG multi grid, with the global constraints treated basically as in §5.6. The eigenvalues are updated once per cycle, together with a more precise determination of the individual eigenfunctions within the space spanned by them, by a Rayleigh-Ritz process. Experiments for model problems [BMR83] show that an FMG algorithm with one V(2, 1) cycle on each level gives a discrete eigenfunction with algebraic errors much smaller than truncation errors. Similar work is needed for each additional eigenfunction.

8.3.2 Continuation (embedding) techniques.

Nonlinear problems usually have many discrete solutions, and the desired one is obtained only if we start from a close enough first approximation. A common way to obtain a good first approximation, or generally to trace branches of solutions, is by *continuation* (also called "embedding" or "Davidenko method"): The problem, including its discretization and its approximate solution is written as depending on some parameter

$$\gamma_0 \leq \gamma \leq \gamma_*$$

such that for γ_0 the problem is easily solvable (e.g., it is linear), while for γ_* it is the problem we really need to solve. We advance γ from γ_0 to γ_* in

steps $\delta\gamma$ small enough to ensure that the solution to the γ problem can serve as a good first approximation in solving the $\gamma + \delta\gamma$ problem. Sometimes γ is a physical parameter; sometimes the solutions are better defined in terms of a non-physical parameter, such as the arclength of the solutions path [Kel77]; cf. §5.6.

As for the relation between multigrid and continuation, several situations arise. Sometimes, the FMG algorithm is a good continuation process by itself. In particular, in non-elliptic and singular perturbation problems where relaxation adds $O(h)$ artificial viscosity, the FMG algorithm starts from highly viscous solutions (since h is large) and gradually eliminates viscosity as it proceeds to finer grids. This is a natural continuation path since problems with large viscosity terms are well-defined and easier to solve. This continuation is carried even much further when the FMG algorithm is continued to include *local* refinements around thin viscous layers (see §9).

Very often, instead of solving the problem several times for a sequence of γ values, the gradual changes of γ can similarly be integrated into just one FMG solver, advancing γ each time the solver proceeds to a new finest level.

An explicit continuation over the problem path $\gamma_0 \leq \gamma \leq \gamma_*$ should be made because the intermediate problems are either interesting in themselves, or necessary for reaching or even defining the desired γ_* solution. When the intermediate problems are not of interest, they can of course be solved to a lower accuracy, using coarser grids only. The grids cannot all be too coarse, however; the meshsize $\hat{h}$ must participate in the continuation process if components of wavelengths comparable to $\hat{h}$ are needed to keep the solutions in the "attraction region" of the desired solution path.

Even when components with $O(\hat{h})$ wavelengths are needed in the continuation process, in each $\delta\gamma$ step they do not usually change much. We can therefore employ the "frozen τ" techniques described below (§15), and perform most of the $\delta\gamma$ steps using, on most parts of the domain, very coarse grids, with only few "visits" to grid $\hat{h}$: Such a continuation process will often require less computational work than the final step of solving the γ_* problem, at which a higher accuracy is sought.

A favorite example where these techniques are put to test is the Bratu problem $\Delta u + \lambda e^u = 0$ in the unit square, with $u = 0$ on the boundary. In collaboration with Klaus Stueben we have solved the problem using FAS, freeing λ to be unknown and adding the value $u(P)$, where P is the center of the square, as a global constraint, treated basically as in §5.6. In this formulation, the multigrid solver had no problem going around the "limit point" ("turning point") of the solution curve (e.g., the curve of $u(P)$ as function of λ, which is not a unique function). It gave solutions at the same efficiency as corresponding algorithms for Poisson equations. In fact, we could solve by one-cycle FMG algorithm to the level of truncation errors, even problems on the upper branch of the solution curve, and even without

continuation at all. The only region where more lengthy calculations were needed was a region where the discrete solution bifurcated, a phenomenon the algorithm was not designed to deal with efficiently. See more details in [ST82, §5.5.1]. A further work along these lines, also to other problems, is reported in [Sch83].

8.4 Estimating truncation errors. τ-extrapolation

As with other defect-correction schemes, the defect can serve also as an error estimator. That is, τ_h^H is an approximation to the local truncation error τ^H (on the coarse grid H), defined by

$$\tau^H = L^H\left(\hat{I}^H u\right) - I^H\left(Lu\right), \tag{8.11}$$

where u is the true differential solution and I^H and $\hat{I}^H$ are two continuum-to-H transfer operators, defined as follows. I^H is the operator used in our grid-H discretization for $f^H = I^H f$, and $\hat{I}^H$ represents the sense in which we want u^H to approximate u: We want u^H to actually approximate $\hat{I}^H u$. The injection $(\hat{I}^H u)(x^H) = u(x^H)$ is usually meant, but other local averaging are sensible too.

Note the analogy between (8.11) and (8.9). τ^H is the correction to grid-H right-hand side that would make the grid-H solution coincide with the true differential solution $\hat{I}^H u$, while τ_h^H is the correction that would make it, at convergence, coincide with the fine-grid solution $I_h^H u^h$. It is hence clear that at convergence

$$\tau^H \approx \tau^h + \tau_h^H, \tag{8.12}$$

where τ^h is the fine-grid local truncation error, defined with $\hat{I}^h$ such that $\hat{I}^H = \hat{I}_h^H \hat{I}^h$. The sign $\approx$ means an equality up to a higher order in h. Relation (8.12) means, more precisely, that if $\tau^h + \tau_h^H$ were used to correct the H-equations, then u^H would be a higher-order approximation to $\hat{I}^H u$; namely, $u^H - \hat{I}^H u$ would equal $w^H - \hat{I}^H w^h$, where $L^H w^H = I_h^H \tau^h$ and $L^h w^h = \tau^h$.

Relation (8.12) can be used to inexpensively raise the approximation order. If the local approximation order (order of consistency) at the point x is p, i.e., if $\tau^h(x) \approx c(x)h^p$, where $c(x)$ is independent of h, then $\tau^H(x) = 2^p c(x)h^p$, hence $\tau_h^H(x) \approx (2^p - 1)c(x)h^p$, and hence $\tau^H(x) \approx 2^p(2^p-1)^{-1}\tau_h^H(x)$. To raise the approximation order all we have then to do is to change the grid-H equations (8.5) by writing them as in (8.8) and multiplying τ_h^H by the fixed factor $2^p(2^p-1)^{-1}$ This operation is called τ **extrapolation**. It resembles Richardson extrapolation, but it can profitably be done even in cases the latter cannot (e.g., in cases $p = p(x)$ is not constant), because it extrapolates the equation, not the solution.

τ-extrapolation can be shown to be a special case of the higher-order techniques of §10.2 below, but it is especially simple and inexpensive. It costs only one multiplication on the *coarser* grid. It is probably best to use it in an FMG algorithm with $W(\nu, O)$ cycles, since a terminal relaxation with the lower order discretization would impair the approximation (even though its order would remain higher). An option for τ-extrapolation exists in the model program FMG1 [MUG84].

Because of the analogy to the local truncation errors, τ_h^H is also called the **relative local truncation error** – the local truncation error of grid H relative to grid h. It is a by-product of the FAS processing which can be used to estimate the true local truncation errors: $\tau^h \approx (2^p-1)^{-1}\tau_h^H$. Hence it can be used in FMG stopping criteria (see §7.3) and in grid adaptation criteria (§9.5, 15).

8.5 FAS interpolations and transfers

The consideration in determining the residual transfer I_h^H, the correction interpolation I_H^h and the FMG interpolation $\mathbb{I}_H^h$ in FAS are basically the same as in CS, but there are some additional possibilities and we should also specify now the FAS solution transfer $\hat{I}_h^H$.

For linear problems the choice of $\hat{I}_h^H$ does not matter of course; all choices will give identically the same results. The solution efficiency of many nonlinear problems is also insensitive to the exact choice of $\hat{I}_h^H$. The choice does matter only where the problem coefficients drastically vary over a meshsize. By the "problem coefficients" we mean those of the linearized problem. In practice, using FAS , we do not linearize the problem, but the transferred solution $\hat{I}_h^H u^h$ implicitly determines the problem coefficients on the coarse grid H – determining the coefficients may in fact be regarded as the *purpose* of this transfer (although, unlike the CS situation, the coefficients can *change* on grid H, with the changing approximation). When the problem coefficients are highly variable, each coarse grid coefficient should be a suitable average of neighboring values of the corresponding fine-grid coefficients. The coarse-grid problem, in other words, should be a proper "homogenization" of the fine-grid problem. Such homogenization is usually obtained by using full weighting for $\hat{I}_h^H$ (as for I_h^H in (4.5)–(4.6)).

In some, very special situations the dominant solution-dependent term in the coefficients may have the form $g(u)$, where g is a sensitive function; large changes in g are caused by more-or-less normal changes in u over a meshsize. In such a case the weighting $\hat{I}_h^H$ should have the special form

$$\hat{I}_h^H \tilde{u}^h = g^{-1} \bar{I}_h^H \left(g\left(\tilde{u}^h\right)\right), \tag{8.13}$$

where $\bar{I}_h^H$ is a normal full weighting, such as (4.6), $g(\tilde{u}^h)$ is a grid function such that $(g(\tilde{u}^h))(x^h) = g(\tilde{u}^h(x^h))$ for every fine-grid point x^h, and g^{-1} is the inverse function of g; that is, $g^{-1}(g(\alpha)) = \alpha$ for any value α. If several

sensitive functions such as g appear in the coefficients, several $\hat{I}_h^H$ may have to correspondingly be used. (So far we have not seen a practical problem where this was required.)

An important possibility offered by FAS is the usage of a special type of interpolation near a nonlinear interior discontinuity, such as a shock or an interface. The grid-H solution, introducing smooth changes to the grid-h solution, may change the **location** of such a discontinuity, moving it a meshsize or two. Near the discontinuity, the correction $\underline{v}^H = \tilde{u}^H - \hat{I}_h^H \tilde{u}^h$ will then be highly non-smooth; it will look like a pulse function. Interpolating it as a correction to the fine grid will introduce there unintended high oscillations. To avoid this, the FAS interpolation (8.6) should be replaced by (8.7) near the discontinuity. This is easy to implement, by adopting the following, more general rule.

> *Use (8.6) everywhere except near points where $\tilde{u}^H - \hat{I}_h^H \tilde{u}^h$ is comparable to $\tilde{u}^h$, where (8.7) should be used.*

8.6 Application to integral equations

When the integral equation

$$\int K(x,y)u(y)dy = f(x,u(x)) \tag{8.14}$$

is discretized in a usual way on a grid with $n = O(h^{-d})$ points, the unknowns are all connected to each other; the matrix of the (linearized) discrete system is full. A solution by elimination would require $O(n^3)$ operations. An FMG solution would require $O(n^2)$ operations, since each relaxation sweep costs $O(n^2)$ operations. In case (8.14) is nonlinear in u, FAS-FMG would be used. Using the FAS structure, even for linear problems, we can often reduce this operation count very much, by exploiting smoothness properties of K.

In most physical applications $K(x,y)$ has a singularity at $y = x$. So usually $K(x,y)$ becomes either smaller or smoother as the distance $|y-x| = (\sum_{j=1}^{d}(y_j - x_j)^2)^{\frac{1}{2}}$ increases. "Smaller" and "smoother" are in fact related: The former is obtained from the latter by differentiations of (8.14) with respect to $x = (x_1, \ldots, x_d)$ (possibly replacing the integral equation by an integro-differential equation). In either case, one can obtain practically the same accuracy in the numerical integration using meshsizes that increase with $|y - x|$, cutting enormously the work involved in relaxation. Usually u(y) is much less smooth than $K(x,y)$ for large $|y - x|$. The integration with increasing meshsizes cannot then use point values of $\tilde{u}^h$, but should use local averages of $\tilde{u}^h$, taken over boxes whose size increases with $|y - x|$. Exactly such averages are supplied by the sequence of coarser grids in the FAS structure. The FAS solution transfers $\hat{I}_h^H$ should of course represent full weighting. One can increase the accuracy of integration by using, in

addition to the full-weighting averages, higher local moments, represented on additional coarser grids. These techniques have been demonstrated in [BL90] and subsequent works.

8.7 Small storage algorithms

Various effective methods for vastly reducing the storage requirement of the discrete solution without using external storage, can be based on the Full Approximation Scheme. One simple method [Bra79b, §2.2] is to use the fact that a problem whose finest grid is h can satisfactorily be solved by an FMG algorithm with only one h-cycle (see §7.3). This means that only one visit is needed to grid h, including the FMG interpolation I_H^h, a couple of relaxation sweeps, and the residual and solution transfers I_h^H and $\hat{I}_h^H$ back to grid H. All these operations can be made "wave-like" by just one pass over grid h, requiring no more than few columns at a time kept in memory. (When the operations of one relaxation sweep have been completed up to a certain column, the operations of the next sweep can immediately be completed on the next column, etc.) This visit is enough to supply the corrected right-hand side $\hat{f}^H$ on grid H (cf. §8.2), hence enough to calculate $\tilde{u}^H$ without any storage allocated to grid h, except for the few mentioned continuously shifted columns.

$\tilde{u}^H$ is as precise as $\tilde{u}^h_{\text{NEW}}$. The usual terminal sweeps of the h cycle are only done if we need the solution on grid h, their role is to *smooth* the interpolation errors, not to *reduce* the error. Moreover, suppose that what we really need from our calculations is some functional of the solution, $\Phi(u)$ say, so we would like to calculate $\Phi^h(\tilde{u}^h_{\text{NEW}})$. All we have to do is to calculate, incidentally to transferring the solution u^h back to grid H, the values of both $\Phi^h(\tilde{u}^h)$ and $\Phi^H(\hat{I}_h^H \tilde{u}^h)$. Then, having later obtained $\tilde{u}^H$, we can calculate

$$\hat{\Phi}^H(\tilde{u}^H) = \Phi^H(\tilde{u}^H) + \Phi^h(\tilde{u}^h) - \Phi^H(I_h^H \tilde{u}^h), \tag{8.15}$$

which is practically as accurate as $\Phi^h(\tilde{u}^h_{\text{NEW}})$, because $\tilde{u}^H - \hat{I}_h^H \tilde{u}^h$ is small and smooth.

This simple procedure reduces the required storage typically by the factor $2^d - 1$, without increasing the computational work. Other procedures can reduce the storage much further by avoiding the storage of coarser grids too, except for a certain $(n_k h_k) \times (n_k h_k)$ box on each grid $h_k = 2^k h$. The h_k box is shifted within the h_{k-1} box to supply the $\tau_{h_k}^{h_{k-1}}$ corrections. The amount of work increases since on "revisiting" the h_k box we need to reproduce its own $\tau_{h_{k+1}}^{h_k}$ corrections. This can be done only in the interior of the box, distance $O(h_k|\log\varepsilon|)$ from the boundary of the box, where $\varepsilon = O(h^p)$ is the desired accuracy on the finest grid, because closer to that boundary the h_{k+1} high frequency errors are larger than the desired accuracy. Hence we must have $n_k \geq O(|\log\varepsilon|)$. The overall required storage can therefore

be reduced to $O(|\log \varepsilon|^d |\log h|)$ (not just $O(|\log h|)$, as mistakenly calculated in [Bra77a, §7.5]). Such procedures are called *segmental refinement techniques.*

Another small-storage multigrid algorithm, not based on FAS, is described in [Hac80]. It is a region dissection procedure, particularly suited for elongated domains.