Levels and Scales

Achi Brandt
Dept. of Applied Mathematics & Computer Science
The Weizmann Institute of Science
Rehovot 76100, Israel

November 1984

Open your eyes and you see multi-level processes all around. They have always been here. The organization and operation of military forces is an obvious example: soldiers are grouped in squads, which in turn are grouped in sections, grouped in platoons, then in companies, battalions, regiments, brigades, divisions, corps and armies. Civil society, less strict and more complicated, still operates in a variety of hierarchical structures: geographical, economical, political, judicial, educational, and so forth.

Such hierarchies are necessary because there are very many, sometimes millions, interdependent decisions to be made: they cannot be made by one governor, because of their multitude and their complicated interdependence, nor can they be decided by many independent administrators without being coordinated with each other. The hierarchical structure effectively deals with this situation by exploiting the fact that each decision has a certain "scale". The location of a new hospital, for example, is a decision which has the scale of a district: It strongly affects the district served by it, marginally affects neighboring districts, and very weakly affects others. The decision can therefore be made by a certain administration assigned to the district, in some coordination with neighboring administrations. The decision cannot effectively be negotiated at a too low administrative level, e.g., at the level of individual families living in the district; the relevant information (concerning needs, constraints, etc.) must be gathered into one point of decision. Similarly, the general outline of principal throughways should be decided at the inter-state level, while local back roads should be regulated at the district or town levels.

The two kinds of roads should be connected, of course. The effective way to manage the inter-dependence between the different levels is based on the assumption that global decisions should only marginally be affected by local ones (otherwise the latter would not be local). Thus, ideally, a two-level hierarchical structure should operate in the following way: The global government first gathers some general figures summarized at the local level, representing sum totals of local needs, important overall constraints, etc. Based on these it prepares preliminary global plans. These global plans give the local governments the framework for devising their own, more detailed, plans. In the course of doing that, the local

government may realize that some, usually marginal, aspects of the global plans do not quite fit the local situation and therefore need some adjustments or corrections. So, at a second round, the global government again gathers information summarized at the local level, now representing sum totals of *remaining* needs ("residuals", in the language of numerical analysts). Since in practice this process is seldom fully recognized, let alone fully effectively organized, more such rounds may be needed. When more levels of government are involved, the process is applied recursively, in a variety of manners.

For very much similar reasons, iterative hierarchical procedures, similar to the process we have just described, are very efficient in solving large and complicated problems on computers. Such procedures were naturally introduced to solve problems where the hierarchical structure was already explicit in the problem itself. A good example is the field of production planning, notably in the Soviet Union, where hierarchic divisions into sectors and pyramidal management naturally led to the introduction of iterative "aggregation/disaggregation" (a/d) algorithms, starting in the mid sixties [10] and growing in the seventies into extensive Russian literature on iterative a/d procedures for large linear programming problems (see [21]). Multi-level approaches have in fact quite naturally emerged in all branches of computer technology, as in the structured organization of computer hardware (see for example [16]), the top-down structured design of software ([23] ,[17]), the pyramidal data structures (trees, heaps, etc.) and many of the most efficient algorithms in computer science, such as fast sorting (sorting n numbers in $O(n \log n)$ operations) the "divide and conquer" class of algorithms, etc. (See, e.g., [1] . Most of these procedures are not iterative, though.)

Also for very much the same reasons, multi-level algorithms have come forth as the most efficient algorithms in solving the very large algebraic systems arising in discretizing partial differential boundary value problems, especially those describing steady-state physical systems. Here the fully hierarchical structure is not at all generally explicit in the problem itself, so it took some effort, and interesting historical development, to realize it. Fully effective multi-level algorithms were first developed as *direct*, not iterative, solvers, treating very special situations where it was algebraically clear enough how to recursively construct hierarchical solvers. I refer here to the fast solvers based on fast Fourier transforms (FFT) and/or reduction methods, especially the cyclic odd-even reduction, both of which are clearly recursive, but non-iterative, multi-level processes (see [8] ,[13] and [18]). The total reduction method also belongs to this class (see [20]). The solution of n equations by this kind of solvers requires at most $O(n \log n)$ computer time and storage, but the class of problems for which this full efficiency is attained is quite limited: separable problems, essentially meaning constant-coefficient elliptic equations with constant-coefficient boundary conditions on rectangular domains. This class has been substantially enlarged by using these fast solvers, and various combinations of them, *iteratively*. Thus, for example, if the coefficients are not constant but sufficiently uniform, the iterative application of a constant-coefficient fast solver can be very effective (the number of iterations depending on the uni-

formity of the coefficients, but not on the meshsize). Extensions to domains of arbitrary shapes have been obtained by "capacitance matrix methods", using the fast direct solvers iteratively, with conjugate-gradient acceleration, each solution typically costing the equivalent of some 15 applications of the fast solver (see e.g. [14]). The nested-dissection approach to elimination ordering [11] is another powerful multi-level approach, more general but less efficient than the FFT and reduction solvers.

Meanwhile, steady-state PDE problems and the solution of their discretized equations were examined from two other points of view, which jointly led to the realization that each such problem contains a natural hierarchy of levels, not immediately explicit, but very powerful and much more general than the mechanical hierarchy exploited by the above methods. First, studying the reasons for slow convergence of various relaxation solvers, it became clear that relaxation is a "local" process which cannot efficiently treat "global" or "smooth" solution components: A smooth error component shows relatively small "residuals", i.e., small errors in the individual difference equations. (For smooth components such small residuals have the same sign throughout a large region, and their effect can therefore be accumulated and produce a large error.) Since the relaxation corrections are based on the individual residuals, they are necessarily small compared with the actual error, if the latter is smooth. Thus, in order to get a correction comparable to a smooth error, information concerning the residuals throughout sufficiently large regions must be summed up to one point of decision, very much as in the case of global social decisions mentioned above. The size of the regions over which residuals should be summed up—so that a correct picture about the error magnitude is obtained—must be comparable to the *scale* of the error, i.e., to the typical distance over which the error substantially changes. As long as each relaxation step (each step of correction) works on a much smaller scale, convergence must be slow.

A second, complementary viewpoint evolved from examining the nature of the discretization error, i.e., the difference between the true solution of the differential problem and the exact solution of the discretized equations. The relative magnitude of this error is clearly determined by the relative magnitudes of the discretization meshsize and the solution scale. A smooth solution, which is a large-scale solution, can thus be approximated on a coarse grid. The same is obviously also true for a smooth *error*. Thus, it became clear, exactly those errors that are slow to converge by relaxation processes on some fine grid can be approximated on a coarser grid, where the meshsize is comparable to their scale and hence their convergence need not be slow.

A natural hierarchy of levels emerges, based on viewing the solution to each boundary value problem as a linear combination of components with different scales: Each component is most effectively controlled by grids with meshsize comparable to its scale, and efficient multi-level control can thus be realized as a multi-grid processing.

A two-grid process, for example, is fully analog to the ideal operation of the two-level hierarchical government described above: The problem is first represented on the coarser grid, e.g., by averaging its equations to the scale of that grid. The (approximate) solution to the resulting coarse-grid problem, once computed, is then interpolated to the fine grid, serving there as a first approximation, a framework, to be next improved by fine-grid processes, such as relaxation. This fine-grid processing finds the fine features of the solution which were invisible to the coarser grid, and also, as a result, encounter some residuals of global (smooth) errors, which it cannot efficiently reduce. (These are smooth errors caused by aliasing, i.e., by the previous coarse-grid processing having misinterpreted coarse-grid traces of the fine features. Now that those fine features have been removed from the error by the fine processing, that aliasing error becomes the dominant one.) So, in the next round, the residual problem is approximately transferred, by some averaging, to the coarse grid, where it can efficiently be solved, and its solution is then interpolated back to the fine grid and added as a correction to the previous fine-grid solution.

The process just described is the two-level "full multigrid" (FMG) algorithm. It can be used recursively in a variety of manners in case more levels are involved.

The number of levels that should be used depends on the ratio between the size of the domain and the finest scale one wants to see in the solution. Between these two scales as many scales should be introduced as practical. Namely, the ratio between successive scales (successive meshsizes) should be as small as possible, as long as this does not substantially increase the total number of gridpoints. The ratio 2:1 between successive meshsizes is very convenient: the total number of gridpoints is still dominated by their given number on the finest grid, but successive meshsizes are close enough to effectively treat any solution component. In fact, with such a ratio, the experience so far showed that suitable FMG algorithms could solve all test problems "to the level of truncation error" (i.e., to the point where the error in approximating the differential solution is dominated by the discretization error, not by the error in solving the discrete system) in just few (less than 10) "work units", where the work unit is the amount of operations involved in expressing the given (finest-grid) system of discrete equations. The 2:1 ratio is also most convenient in programming the inter-grid transfers. Note that in this respect the multigrid processing is different from multi-level social structures: Its levels are chosen much more tightly, to achieve maximum efficiency.

Note also that the decomposition of the PDE solution into components of different scales is only implicit; it is used above to motivate and explain the validity and strength of the multigrid process; but the actual multigrid algorithm does not use any such decomposition: It only transfers equations (or residual equations) from fine grids to coarse grids, and solutions (or corrections) from coarse to fine. Decomposition in terms of Fourier components, in particular, can be used as a powerful tool to *analyze*, and even exactly predict, the performance of multigrid algorithms, but the algorithms themselves do not employ such decompositions, and

their efficiency extends far beyond the cases where the Fourier analysis is rigorously valid. (Incidentally, it is important to realize that in some cases Fourier analysis is the wrong tool to separate local effects from global ones. For example: A local discontinuity gives rise to global high-frequency Fourier components. Other tools should then be used to understand quantitatively, and optimize, the multigrid performance.)

The FMG solvers (solving discretized PDEs to the level of truncation errors) have been developed to the point that they are today even faster than the FFT and reduction solvers mentioned before. (The multigrid Poisson solver described in [3] is the fastest we know.) More importantly, of course, these FMG solvers are much more general. They solve with the same efficiency (i.e., in few work units) complicated nonlinear systems on general domains. Moreover, it is possible to integrate into each application of an FMG solver, for small extra computer work, various processes of *local* mesh refinement, mesh optimization and *local* coordinate transformations, making it very effectual in treating singularities, unbounded domains, curved boundaries, boundary layers, discontinuities, etc.

Furthermore, multigrid solvers can directly be applied to "higher" problems—such as optimization, optimal design and optimal control problems, or system identification problems—whose solution would normally be accomplished through solving a *sequence* of boundary value *sub* problems. An important principle indeed is always to try to multigrid the given, *original* problem, instead of merely using fast multigrid solvers to a sequence of intermediate subproblems. The *original* problem (e.g., the optimization problem itself) should first be solved on a coarser grid, then relaxed on the finer grid (including for example local optimization of parameters, in case some of the functions to be optimized do have local scales), then brought back to the coarser grid, etc. The entire solution of the *original* problem may thus cost only few work units. Multigridding the original problem is especially advantageous in case that problem is autonomous (i.e., having solution-dependent but not directly space-dependent coefficients, as for example most fluid dynamic problems) while the subproblems are non-autonomous (having coefficients spatially depending on the solution of a previous subproblem).

Sometimes it is still required to solve a sequence of subproblems. A typical example is the interactive design of a certain structure, where between two solutions of the system the structure is changed in some specific parts and/or in some of its global parameters. The new solution can then be obtained from the old one by a remarkably short multigrid processing, in which the finer grids are relaxed only around the changed parts. This technique may allow the design of a large structure to be done mostly in core memory, since for several design steps only the currently designed parts, and some neighboring parts, should fully be kept in memory while the rest of the structure may be represented by coarser grids (using the FAS version of multigrid). Re-solving by such techniques should be so efficient as to allow the designer to introduce some changes to a large structure and immediately view the new solution on the screen.

Similarly, in evolution problems with implicit time differencing, the solution of a new system of equations seems to be required at each time step. Each of these systems could be solved very efficiently by an FMG algorithm, costing the equivalent of just few *explicit* time steps. But, here again, multigrid techniques may even be more effective if they are designed in terms of the original evolution problem itself. For example, one can often drastically reduce the overall work by exploiting the fact that the solution is a superposition of pure convection and smooth changes, or changes which are smooth throughout most of the domain, hence requiring fine-grid interactions only at some small subdomains.

The full efficiency of multigrid solvers, as described here, is not easy to obtain, though. It sometimes depends on the correct treatment of each feature of the problem at each multigrid stage. Many things can easily go wrong, such as relaxing a certain boundary condition in a way which conflicts with the interior smoothing; or improper fine-to-coarse transfers of boundary conditions and their residuals; or treating at relaxation conditions which seem local but are not; or using a relaxation scheme which is not as powerful a smoother as it should, and can, be; or wrong order or type of interpolations; or any inadequate treatment of any difficulty, from structural singularities (e.g., reentrant corners) to discontinuities in the solution or in the equations, anisotropies, non-ellipticity, etc., etc. A single mistake at any of these may substantially degrade the whole performance, not to mention plain programming bugs, which, due to the corrective nature of the algorithm, may well disguise themselves in the innocent form of slow convergence. To obtain full efficiency it is therefore necessary to construct the algorithmic concepts and the actual programs in a gradual, systematic way, using available knowhow (see [12] ,[4]).

The class of partial differential equations that can be solved by multigrid solvers has been ever extending, from second order equations to arbitrary orders, from linear to nonlinear, from smooth coefficients to strongly discontinuous ones [2] , from definite to indefinite problems, from scalar equations to general systems [4, §3.8], and from elliptic type to other types.

Unlike evolution problems, where properties like hyperbolicity and parabolicity are all important, the only feature that matters concerning the type of the differential operator in (stationary) boundary value problems is whether it is nicely (isotropically) elliptic or not. If it does not have a good ellipticity measure, it does not make any difference whether it is anisotropic elliptic (like $\epsilon\partial 2/\partial x2 + \partial 2/\partial y2$) or semi-elliptic, weakly elliptic (e.g., having elliptic singular perturbation), hyperbolic or any other non-elliptic type: All these types have the same basic difficulty and can be treated essentially by the same approach. The difficulty is that the solution scales discussed above are *not isotropic*. (Incidentally: One dimensional problems, such as $du/dx = f$, are elliptic, unlike their two-dimensional counterparts. Their use as models for treating non-ellipticity is thus erroneous.) In other words, at each point in the domain of the problem there passes a "characteristic line" (sometimes just a characteristic *surface*; and in case of non-scalar PDEs there

may pass *several* such lines or surfaces at each point, corresponding to different solution components) such that a "global" error component (in the sense that its residuals are small compared with its size) can change rapidly in directions perpendicular to the characteristic lines. Such global but rapidly changing components are called *characteristic components*. As global components, their convergence by relaxation is inefficient, but as rapidly changing components they cannot be well approximated by coarser grids.

The key for efficient multigrid treatment of such anisotropic problems is to clearly distinguish between two very different situations, depending on whether or not one wishes to approximate characteristic components far from boundaries, where "far" is meant relative to the component's smaller scale, i.e., the scale of its rapid change perpendicular to the characteristic. To so approximate characteristic components by the discrete system, the grid must (roughly) be aligned with the characteristic lines throughout the domain. Fast multigrid solvers can then be based on these aligned gridlines, either by relaxing these gridlines simultaneously, or by coarsening the grid only along these lines and not in the perpendicular directions. (The latter approach is preferred when the characteristics are surfaces rather than lines.) If, on the other hand, the grid is not consistently aligned with the characteristics, then characteristic components cannot be approximated far from boundaries (except accidentally, in some regions of accidental alignment), and it is then unwise to attempt to have fast algebraic convergence for such components: This convergence, which is meaningless in terms of approximating the *differential* solution, is harder to obtain: costlier and more complicated relaxation schemes and/or inter-grid transfers must be developed, and even then fast convergence is not always guaranteed. It is absurd, we believe, to invest most of your computer resources and programming effort to get fast algebraic convergence exactly for those components whose algebraic solution is generally (outside accidental regions of alignment) no approximation to the differential solution.

This fashion in treating anisotropy, as well as some other developments, have led to the recognition that fast algebraic convergence should not be the main objective of multigrid solvers. The objective is of course to get the desired accuracy, in terms of solving the *differential* problem, for minimal computer (and also human) resources. This is obtained by FMG solvers which, especially in cases of anisotropic equations without corresponding grid alignment, do not necessarily employ uniformly good smoothers, hence nor do they attain uniformly good algebraic convergence rates. Working with such solvers requires of course certain modifications in the traditional "smoothing factor" approach for measuring the effectiveness of relaxation (see [4, §20.3.1]), as well as new approaches for apriori predicting, and aposteriori judging, the overall success of the FMG solver (see [4, §7.4,§7.5 and §1.6] .) An important advantage of these approaches for performance evaluation is that the performance becomes less sensitive to the precise treatment of all problem features at all algorithmic stages.

These approaches also allow the evaluation of important schemes which de-

liberately avoid any algebraic convergence. Such for example is the "double discretization" scheme [4, §10.2] , which employs less accurate, stable discretizations in relaxation processes, and others, more accurate but not necessarily stable ones, in the fine-to-coarse transfers of residuals, thus combining easier local stability with higher global accuracy.

Another interesting development has recently led to the extension of multigrid-like techniques to cases where no grid is actually present. It started, in a way, with the development of usual multigrid algorithms for diffusion problems which were isotropically elliptic but in which the diffusion coefficients were strongly discontinuous. It turned out [2] that to obtain the usual multigrid efficiency in such cases, the coarse-to-fine interpolation of corrections should be based on the difference equations themselves, rather than being the standard polynomial interpolation. This later led to the recognition that interpolation can be based solely on the algebraic equations, without even using the geometry of the grid. In a similar manner, the fine-to-coarse transfer of residual equations can completely be based on the *adjoint* (transposed) algebraic system. Since these intergrid transfers are what give the coarse grid a definite meaning, it was further realized that even the choice of the coarse-grid *variables* can be freed from its traditional geometric context, and purely "algebraic multigrid" (AMG) schemes were introduced [5] , [4, §13.1] , [6] , [15] . In these schemes the selection of coarser levels is based on the principle that each variable of any level should have a sufficiently strong total "algebraic connection" to variables of the next-coarser level. The entire processing is thus made in terms of the given (sparse) algebraic system of equations, with no reference to their geometric origin. AMG algorithms can thus be used as very efficient "black box" solvers for important classes of matrix equations. (For some other classes the current AMG solvers are not suitable.) The typical multigrid efficiency is obtained by AMG even for cases where it would be very difficult to construct conventional (geometric) multigrid algorithms, such as cases of finite element discretization on arbitrary, irregular triangulations, or even cases where topologically rectangular grids are used, but with highly and non-uniformly stretched coordinates (Lagrangian discretizations in particular) or with peculiarly distributed physical coefficients. In addition, AMG solvers can be applied to many large algebraic systems which are not at all derived from continuous problems, such as the geodetic problem treated in [7] .

The scope of multi-level computations has thus been extended very much. To state it most broadly, consider any matrix equation $Ax = b$. (That the system is linear is not really essential; but it simplifies the following statements.) Denote by $\tilde{x}$ the evolving computed approximation to x, and define the error vector $e = x - \tilde{x}$ and the *normalized* residual vector $r = (r_i) = (a_i e / \parallel a_i \parallel)$, where a_i is the i-th row of A and $\parallel \cdot \parallel$ is the ℓ_2 norm. For a suitable relaxation scheme it can be shown [5, Theorem 3.4] that the decrease in $\parallel e \parallel$ per sweep can be slow only when $\parallel r \parallel \ll \parallel e \parallel$. Since r is properly normalized, for most error components $\parallel r \parallel$ is comparable to $\parallel e \parallel$. Hence, convergence can be slow only for *special* types of error components. Slowly converging errors can therefore be approximated by

far fewer parameters, that is, by a much smaller algebraic system—a coarser level. To exploit this fact one of course needs some characterization for those vectors e for which $\| r \| \ll \| e \|$. In case the matrix A approximates a differential operator L, those vectors e approximate functions v for which $\| Lv \| \ll | L | \| v \|$. This usually implies smoothness of v, or, when L is anisotropic, at least smoothness in characteristic directions. In other cases other characterizations can be found, so the general rule which emerges is that slow convergence should always be avoidable.

This corresponds, more or less, to the "golden rule of computational physics", which states that 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. Indeed, multi-level processing is a general vehicle to effect this rule. So, whenever you have stalling computations—either in the form of slowly converging iterative procedures, or in the form of computational grids, in space and/or in time, which almost everywhere tend to excessively over-resolve the scales of real physical changes—try to think in terms of multi-level techniques.

Multi-level methods are now in the process of being introduced into a variety of new fields, including various systems of tomography, image processing and pattern recognition; statistical physics; queuing theory; network simulation and design; geodesy; multivariate interpolation; large transportation problems and linear programming. The aggregation/disaggregation methods developed earlier for linear programming can now be improved by an AMG-type approach, because the latter provides a more mathematical basis for defining the levels, hence "tighter" hierarchies, exploiting implicit levels not necessarily recognized by the real-life systems. Applications to image processing and pattern recognition are in a sense not quite new, either: There is a strong evidence (see [9] , [19] , [22]) that the human vision processes themselves, in our brain, are multilevelled. Thus indeed, when you open your eyes, you see multi-level processes all around—social, biological as well as physical (hierarchical structure of matter—resulting from some primordial evolution toward "efficiency"?), and your seeing itself is multilevelled. Which is fine, of coarse.

REFERENCES

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullmann: *The Design and Analysis of Computer Algorithms.* Addison Wesley, 1974.

[2] R.E. Alcouffe, A. Brandt, J.E. Dendy, Jr. and J.W. Painter: The multi-grid methods for the diffusion equation with strongly discontinuous coefficients. *SIAM J. Sci. Stat. Comput.*, **2** (1981), 430–454.

[3] D. Barkai and A. Brandt: Vectorized multigrid Poisson solver for the CDC Cyber 205. *Applied Math. and Computations*, **13** (1983), 215–229.

[4] A. Brandt: *Multigrid Techniques: 1984 Guide, with Applications to Fluid Dynamics.* A monograph, 187 pages. Weizmann Institute of Science, Re-

hovot, Israel, February 1984. Appeared as Lecture Notes in Computational Fluid Dynamics, von Karman Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium. Available from GMD-F1T, Postfach 1240, D-5205, Sankt Augustin 1, W. Germany.

[5] A. Brandt: Algebraic multigrid theory: The symmetric case. *Preliminary Proc. Int. Multigrid Conf.,* Copper Mountain, Colorado, April, 1983.

[6] A. Brandt, S. McCormick and J. Ruge: Algebraic multigrid (AMG) for sparse matrix equations. *Sparsity and Its Applications* (D.J. Evans, ed.), Cambridge University Press, 1984.

[7] A. Brandt, S. McCormick and J. Ruge: Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Report, Colorado State University, Fort Collins, Colorado, 1983.

[8] B.L. Buzbee, G.H. Golub and C.W. Nielson: On direct methods for solving Poisson's equations. *SIAM J. Num. Anal.*, **7** (1970), 627-656.

[9] F.W.C. Campbell and J. Robson: Application of Fourier Analysis to the visibility of gratings. *J. Physiol. (Lond.)* **197** (1968), 551–566.

[10] L.M. Dudkin and E.B. Yershov: Interindustries input-output models and the material balances of separate products. *Planned Economy* **5** (1965), 54–63.

[11] J.A. George and J.W.H. Lin: *Computer Solution of Large Sparse Positive Definite Systems.* Prentice Hall, Englewood Cliffs, NJ, 1981.

[12] W. Hackbusch and U. Trottenberg (eds.): *Multigrid Methods*, Lecture Notes in Math. **960**, Springer-Verlag, 1982.

[13] R.W. Hockney: The potential calculation and some applications. *Meth. in Comput. Phys.* **9** (1970), 135–211.

[14] W. Proskurowski: Capacitance matrix solvers—a brief survey. *Elliptic Problem Solvers* (M. Schultz, ed.), Academic Press, 1981, 391–398.

[15] J. Ruge and K. Stüben: Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). This book.

[16] H.S. Stone: *Introduction to Computer Organization and Data Structures*, McGraw-Hill, New York, 1972.

[17] G.H. Swann: *Top-Down Structured Design Techniques.* Petrocelli Books, 1978.

[18] C. Temperton: Direct methods for the solution of the discrete Poisson equation: some comparisons. *J. Comp. Phys.*, **31** (1979), 1–20.

[19] D. Terzopoulos: Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics and Image Processing* **24** (1983), 52–96.

[20] U. Trottenberg: Reduction methods for solving discrete ellip-

tic boundary value problems—an approach in matrix terminology. *Fast Elliptic Solvers* (U. Schumann, ed.). Advanced Publications, London, 1977.

[21] I.Y. Vakhutinsky, L.M. Dudkin and A.A. Ryvkin: Iterative aggregation—a new approach to the solution of large-scale problems. *Econometrica* **47** (1979), 821–841.

[22] H.R. Wilson and J.R. Bergen: A four mechanism model for spatial vision. *Vision Res.* **19** (1979), 19–32.

[23] E. Yourdon: *Structured Design.* Yourdon, Inc.