ADAPTIVE MULTIGRID METHODS: THE UG CONCEPT

P. Bastian
G. Wittum
Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg,
Im Neuenheimer Feld 368, 69120 Heidelberg, Federal Republic of Germany,
wittum@iwr.uni-heidelberg.de

Abstract

In the present paper we discuss the development and practical application of a flexible software toolbox for multigrid methods on unstructured and locally refined grids. Our first aim is to combine modern optimal multigrid methods with the robustness strategies developed for structured grids and in the second part we discuss a parallel implementation of the programming environment on multiprocessors with distributed memory. The various techniques are illustrated with many practical experiments.

1 Introduction

In the first part of this paper we discuss the development and practical application of robust multi-grid methods to solve partial differential equations on adaptively refined grids. Since a couple of years multi-grid methods are well established as fast solvers for large systems of equations arising from the discretization of differential equations. However, it is still a substantial unresolved question to find robust methods, working efficiently for large ranges of parameters e.g. in singularly perturbed problems. This applies to diffusion-convection-reaction equations, arising e.g. from modelling of flow through porous media, the basic equations of fluid mechanics and plate and shell problems from structural mechanics.

Multi-grid methods are known to be of optimal efficiency, i.e. the convergence rate κ does not depend on the dimension of the system, characterized by a stepsize h. Following [28] we call a multi-grid method robust for a singularly perturbed problem, if

$$\kappa(h, \varepsilon) \leq \kappa_0 < 1, \ \forall \varepsilon > 0, \ h > 0, \tag{1}$$

ε denoting the singular perturbation parameter. Up to now multi-grid methods satisfying (1) have been studied in the literature only for special model cases using structured grids, see [24], [25], [17], [27], [28], [29].

Problems of the type mentioned, typically show degenerations in hyperplanes. To resolve these zones special dynamic grid adaptation techniques are necessary. Here it is necessary to rethink standard multi-grid techniques. In §2 we classify several multi-grid approaches for adaptively refined grids. On the one hand adaptively refined grids can substantially weaken the robustness requirement (1) as outlined in §3. On the other hand the unstructured grids generated by adaptive refinement require special numbering techniques so that the smoother does a good job on the problem. It is one of the main

objectives of the present paper to present a strategy to combine the techniques of robust multi-grid and adaptivity.

The practical use of the techniques mentioned so far requires a substantial programming effort. Compared with traditional approaches on structured uniformely refined grids in logically rectangular domains, code complexity is at least an order of magnitude higher according to our experience. It is the aim of the *ug* code described in §4 to find proper programming abstractions that allow reuse of the code for many different applications. This is especially important in the parallel version (§5), where the dynamic management of the distributed data structure complicates the code substantially. It should be noted here that most of the robustness techniques described in the first part of the paper are very hard to parallelize efficiently, so the parallel version uses only simpler parallelizable smoothers at the moment.

Several practical examples illustrating the robustness techniques in the serial version and speedup results for the parallel version are presented in §6.

2 Multi-Grid Strategies

2.1 Basic Multi-Grid Techniques

Let the linear boundary-value problem

$$\begin{aligned} Ku &= f \text{ in } \Omega \\ u &= u_R \text{ on } \partial\Omega \end{aligned} \tag{2}$$

with a differential operator $K : U \to F$ between some function spaces be given on a domain $\Omega \subseteq \mathbf{R}^d$. Let (2) be discretized by some local discretization scheme on a hierarchy of admissible grids (cf. [14])

$$\begin{aligned} &\Omega_l \quad , \quad l = 0, \ldots, l_{max} \\ &\Omega_l \subseteq \Omega_{l+1} \subseteq \Omega \quad . \end{aligned} \tag{3}$$

We use nested grids only for ease of presentation. Most of the methods discussed below can readily be applied to general loosely coupled grids violating (3). The discretized equations on Ω_l are denoted by

$$\begin{aligned} K_l u_l &= f_l \text{ in } \Omega_l, \text{ for } l = 1, \ldots, l_{max} \quad , \\ u_l &= u_{R,l} \text{ on } \partial\Omega_l \end{aligned} \tag{4}$$

with

$$K_l : U_l \to F_l \quad , \tag{5}$$

U_l, F_l denoting the discrete analoga of U and F with finite dimension n. We assume that the discretized equations are sparse. Further let some "smoother"

$$S_l : U_l \to U_l \text{ for } l = 0, \ldots, l_{max} \quad , \tag{6}$$

and "grid transfer operators"

$$p_{l-1} : U_{l-1} \to U_l, \quad r_{l-1} : F_l \to F_{l-1}, \text{ for } l = 1, \ldots, l_{max} \quad , \tag{7}$$

be given.

Multi-grid methods are fast solvers for problem (4). We basically distinguish between additive and multiplicative multi-grid methods. The multiplicative method is the well-known classical multi-grid (cf. [13]) as given in algorithm 2.1:

Algorithm 2.1 *Multiplicative multi-grid method.*

mmgm(l, u, f)
integer l; grid function u, f;
{ grid function v, d; integer j;
  if $(l = 0)$ $u := K_l^{-1} f$;
  else {
    $u := S_l^{\nu_1}(u, f)$;
    $d := r_{l-1}(K_l u - f)$;
    $v := 0$;
    for j:=1 step 1 to γ do mmgm$(l-1, v, d)$;
    $u := u - p_{l-1} v$;
    $u := S_l^{\nu_2}(u, f)$;
  }
}

The additive multi-grid method is given by the following algorithm.

Algorithm 2.2 *Additive multi-grid method.*

amgm(l, u, f)
integer l; grid function $v[l]$, $d[l]$;
{ integer j;
  $d[l] := K_l u - f$; $v[l] := 0$;
  for j:=l step -1 to 1 do { $d[j-1] := r_{j-1} d[j]$; $v[j-1] := 0$;}
  for j:=1 step 1 to l do $v[j] := S_j^{\nu}(v[j], d[j])$;
  $v[0] := K_0^{-1} d[0]$;
  for j:=1 step 1 to l do $v[j] := v[j] + p_{j-1} v[j-1]$;
  $u := u - v[l]$;
}

The structure of both algorithms can be seen from Figs. 1(a) and 1(b). The main difference between these two variants is that in the multiplicative method smoothing and restriction of the defect to the next coarser level are performed on one level after the other sequentially, while in the additve method smoothing on the different levels can be performed in parallel. Restriction and prolongation, however, are sequentially in the additive method too. Usually, the additive methods are applied as preconditioners,

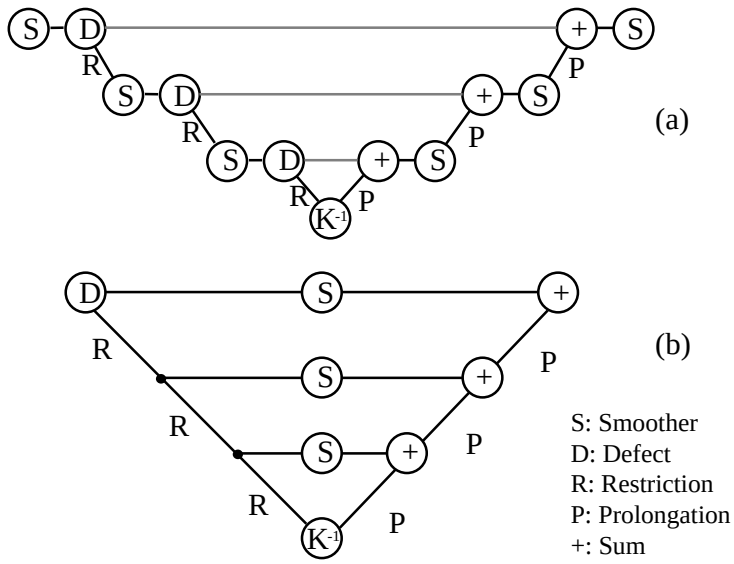


Figure 1: *Outline of the V-cycle multiplicative multigrid algorithm mmgm (a) and of the additive multigrid algorithm amgm (b).*

since acceleration methods like cg directly pick an optimal damping parameter, the multiplicative methods are used as solvers and as preconditioners. According to [31], these methods can be formulated as additive Schwarz methods.

Applying multi-grid methods to problems on locally refined grids one has to think about the basic question, how to associate grid-points with levels in the multi-grid hierarchy. Consider the hierarchy of grids $\{\Omega_l, l = 0, \ldots, l_{max}\}$ from (3). Early multi-grid approaches smooth all points in Ω_l. This may cause a non-optimal amount of work and memory of $O(n \log n)$ per multi-grid step. This problem was the starting point for Yserentant , [32], and Bank-Dupont-Yserentant, [1], to develop the method of hierarchical bases (HB) and the hierarchical basis multi-grid method (HB/MG). These were the first multi-grid methods with optimal amount of work per step for locally refined grids. This is due to the fact that on level l only the unknowns belonging to points in $\Omega_l \setminus \Omega_{l-1}$ are treated by the smoother. However, the convergence rate deteriorates with $\log n$. For the first time this problem was solved by the introduction of the additive method by Bramble, Pasciak and Xu, [9], (BPX). There on level l the smoother treats all the points in $\Omega_l \setminus \Omega_{l-1}$ and their direct neighbours, i.e. all points within the refined region.

Table 1 gives an overview of the multi-grid methods used for the treatment of locally refined grids and classifies the variant we call "local multi-grid". The methods mentioned above differ in the smoothing pattern, i.e. the choice of grid points treated by the smoother. The methods in the first two lines are of optimal complexity for such problems. The amount of work for one step is proportional to the number of unknowns on the finest grid. However, only the methods in the second line, BPX and local multi-grid converge independently of h for scalar elliptic problems. The basic advantage of the multiplicative methods is that they do not need cg-acceleration and thus can be directly applied to unsymmetric problems, further they show a better convergence rate and on a serial computer the additive process does not have any advantage. The local multi-

Table 1: *Multi-grid methods for locally refined grids.*

smoothing pattern	basic structure: additive	basic structure: multiplicative
(1) new points only	*HB Yserentant, 1984, [32]*	*HBMG Bank, Dupont, Yserentant, 1987, [1]*
(2) refined region only	*BPX Bramble, Pasciak, Xu, 1989, [9]*	*local multi-grid, [21], [10], [6]*
(3) all points	*parallel multigrid Greenbaum, 1986, [12]*	*classical multi-grid, [11]*

grid scheme is the natural generalization of the classical multi-grid method to locally refined grids, since in case of global refinement, it is identical with the standard classical multi-grid method.

The local multi-grid has first been introduced by Rivara in [21] and first been analyzed in 1991 by Bramble, Pasciak, Wang and Xu, [10]. They considered it as a multiplicative variant of their so-called BPX-method, [9]. However, they did not consider robustness. Without knowledge of this, the authors developed this method as a variant of standard multi-grid based on the idea of robustness (cf. [6]). The main advantage of this approach is that the application to unsymmetric and non-linear problems is straightforward (cf. [6]). Robustness for singularly perturbed problems is achieved by combining local multi-grid with robust smoothers (cf. [6]), as explained in the next section.

3 Robustness Strategies

3.1 Robust Smoothing

Already in 1981, Wesseling suggested the first robust multi-grid method for singularly perturbed problems discretized on structured grids [24], [25]. The main idea is to apply a smoother which solves the limit case exactly. This is possible e.g. for a convection-diffusion equation using a Gauß-Seidel smoother and numbering the unknowns in convection direction. Wesseling however, suggests to use an incomplete LU-smoother, since this handles the convection dominated case as well as the anisotropic diffusion (cf. [17], [28]). Main ingredients, however, are the use of structured grids and a lexicographic numbering.

A simple analysis of the hierarchical basis methods (HB, HB/MG) shows that the smoothing pattern is too poor to allow robust smoothing.

Remark 3.1 *The hierarchical basis method and the hierarchical basis multigrid method do not allow robust smoothing for a convection-diffusion equation. The smoothing pattern*

used in these methods does not allow the smoother to be an exact solver for the limit case. This holds for uniformly as well as for locally refined grids.

Based on this observation, we extended the smoothing pattern, adding all neighbours of points in $\Omega_l \setminus \Omega_{l-1}$. This allows the smoother to solve the limit case exactly, provided the grid refinement is appropriate. This is confirmed by numerical evidence given in Chapter 5.

Up to now some theory is contained in [28],[29] and the new papers by Stevenson [22], [23] for uniformly refined grids. This theory shows that the basic requirement that the smoother is an exact solver in the limit case is not sufficient to obtain robustness. Additionally it must be guaranteed that the spectrum of the smoother is contained in $[-\vartheta, 1]$ for $0 \leq \vartheta < 1$. This can be achieved by modification (cf. [28], [23]).

3.2 A Robust Smoother for Convection-Diffusion Problems

The construction of a robust smoother, which is exact or very fast in the limit, is the kernel of a robust multigrid method and makes up the main problem when applying this concept to unstructured grids. Here we need special numbering strategies.

In the following we present a strategy for the convection-diffusion equation

$$-\varepsilon \Delta u + c \cdot \nabla u = f \ , \tag{8}$$

with the convection vector c, and $\varepsilon > 0$. Discretizing the convection term by means of an upwind method, we can assign a direction to each link in the graph of the stiffness matrix. If the directed graph generated by this process is cycle-free, it defines a partial ordering of the unknowns. This partial ordering can be used to construct an algorithm for numbering of the unknowns, which brings the convective part of the stiffness matrix to a triangular form. The following numbering algorithm performs such an ordering on general unstructured grids, provided the convection graph is cycle-free.

Algorithm 3.1 *downwind_numbering.*

1. Assign the downwind direction from the discretization of the convective term to each link in the stiffness martix graph. Indifferent links are marked by 0.

2. Put n = number of unknowns.

3. Find all vertices with minimal number of incoming links and put them in a fifo F.

4. Derive a total order from the directed acyclic graph

   For all vertices L initialize Index$(L) = 0$;
   While (F not empty) do
   get E from F;
   (4a) Put $Index(E) := 1$; Put E in fifo FP; $i := 1$;
   (4b) While (FP not empty) and ($i < n$) do
   Get K from FP;
   For all neighbors L of K do
   If (L downwind from K) and (Index$(L) \leq$ Index(K))

$i :=$ Index(L);
Index$(L) :=$ Index(K)+1;
Put L in FP;

5. Call quicksort with the vertex list and the criterion $Index(L) < Index(K) \Rightarrow L < K$. Output: Ordered vertex list.

Remark 3.2 *If the edge graph is cycle-free, loop (4b) terminates in $O(n)$-steps with $FP = \emptyset$. Loop (4) has complexity $O(q \cdot n)$ where q is the number of minimal elements in the edge graph, which is small. Because of calling quicksort in (5) the complexity of the whole algorithm equals $O(q \cdot n \ln n)$.*

If loop (4b) terminates with $FP \neq \emptyset$ and $i \geq n$, the edge graph contains a cycle.

This method has been used for the computations described in Section 5 . Meanwhile it has been improved by Bey (cf. [7]). Cycles in the matrix graph may occur, if there are vortices in the convection c. If c is vortex-free, cycles can occur if several triangles with sharp angles are neighbouring each other and are almost perpendicular to the flow direction (cf. [7]). These numerically caused cycles, however, can be simply eliminated by finding and cutting elementwise cycles. This is possible with $O(n)$ work count.

3.3 Semi-coarsening

Another strategy to obtain a robust multi-grid method is the so-called semi-coarsening approach (cf. [8]). The basic idea is to improve the coarse grid correction instead of the smoother. Starting with a fine and structured grid, coarsening is performed only in those co-ordinate directions, in which the scale of the equation is already resolved. E.g. for the anisotropic model problem

$$-(\varepsilon \partial_{xx} + \partial_{yy})u = f \ , \quad \text{in}\Omega = (0,1) \times (0,1) \tag{9}$$

with corresponding boundary conditions one would coarsen an equidistant cartesian grid in case of small ε as shown in Figure 2(a).

Remark 3.3 *Such a sequence of coarse grids yields a robust multi-grid method for the anisotropic model problem (9) without using a special smoother, since the coarse grid resolves the scale in the direction where the smoother does not work.*

This semi-coarsening approach, however, is based on the use of fine grids which do not resolve the differential scale, otherwise there would be no semi-coarsening. Consequently this approach is not applicable as soon as the finest grid resolves the problem scale, which is crucial when solving differential equations.

This does not apply to so-called multiple semi-coarsening approaches [20], since these methods are able to construct sequences of coarse grids from any structured fine one, no matter if the scale is resolved. Solving practical problems we mainly have to look for an approach which allows to adapt the grid to the differential scale by adaptive refinement and to solve efficiently on the hierarchy of grids generated this way.

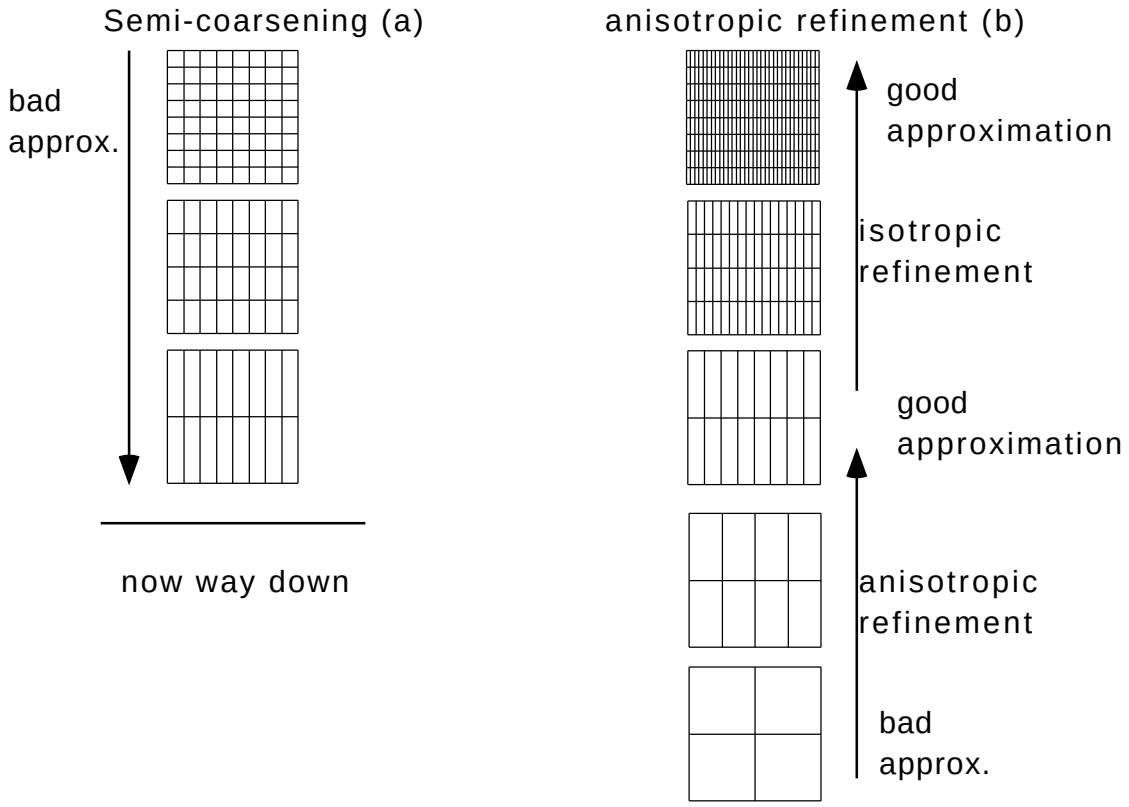


Figure 2: *Illustration of semi-coarsening (a) and anisotropic refinement (b).*

3.4 Anisotropic Refinement

Instead of starting with a fine grid and constructing the gridhierarchy by coarsening we start with a coarse grid and refine that anisotropically in order to resolve the scale successively. Such a refinement process is given e.g. by the "blue refinement strategy" due to Kornhuber,[18]. The basic idea is just to refine quadrilaterals with a "bad aspect ratio" by halving the longer edge. Bad aspect ratios can be introduced either by element geometry or by anisotropic coefficients in the equation. This is shown for the anisotropic model problem (9) in Fig. 2(b). Note that the discretization error is balanced on the *coarsest* grid for semi-coarsening, while it is balanced on the *finest* grid for the anisotropic refinement approach. Kornhuber described how to generalize this approach to triangular unstructured grids. Following this process we finally obtain a grid Ω_l which resolves the scale of the problem.

From this grid on we refine regularly and so the multi-grid process will obviously work without problems.

Remark 3.4 *A proof of robust multi-grid convergence is straightforward since the asymptotic behaviour is determined by the isotropic problem. So we need a robust method only for a finite sequence of grids up to a fixed $h > 0$, weakening the robustness requirement (1) to the relative robustness:*

$$\kappa(h, \varepsilon) \leq \kappa_0 < 1, \quad \forall \overline{\varepsilon} \geq \varepsilon \geq \underline{\varepsilon} > 0, \ \forall h \geq \underline{h} > 0 \ , \tag{10}$$

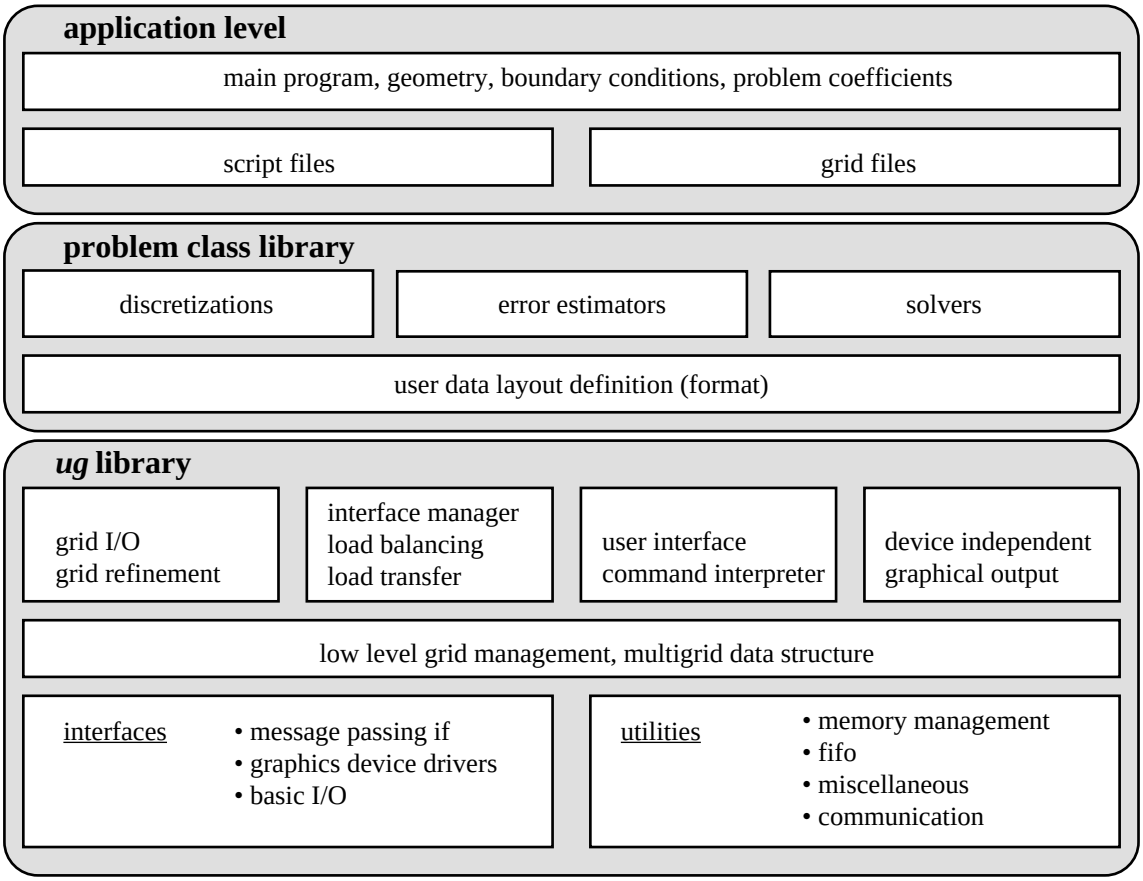


Figure 3: *Overview of the internal structure of the ug code.*

which makes the job much easier. Thus it is sufficient in many cases to use just a lexicographically numbered ILU_β, since we do not need the property that the smoother is exact in the limit case. It is sufficient that it reasonably accounts for the "main connections" up to a fixed range of $\varepsilon > 0$ and for finite h.

Since this process improves the approximation of the differential problem at the same time, this will be the appropriate approach to follow.

An example of that type is the skin problem described in §5.

4 The Software Toolbox ug

A big problem in practice with the modern adaptive multigrid methods introduced above is code complexity when implementing these methods in a computer program. Especially on the parallel computer this is an important point since the dynamic parallel management of the data structure is very complicated. Therefore we developed the software environment *ug* ("unstructured grids") as a problem independent "toolbox" for (parallel) unstructured adaptive multigrid applications. It is a layered construction of several libraries, see Fig. 3 for an overview. The bottom layer contains all components that are totally independent of the PDE to be solved, e. g. grid I/O, grid refinement, device independent graphical output, user interface and dynamic grid management with load balancing in the parallel version. The next layer is the so-called problem class library

that implements discretization, error estimators and solvers for a whole class of PDEs, e. g. a scalar conservation law or incompressible, stationary Navier-Stokes equations. On top of that resides the user's application that provides the domain, boundary conditions and problem coefficients for the lower layers.

The relative code size of these layers indicates that the proper abstractions (interfaces) have been chosen: The *ug* layer typically makes about 75-80% of the executable, the problem class layer takes 15-20% in the convection-diffusion case (with many different solvers) and a main program typically is only 5%. This means in practice:

- Modularization and hierarchical code design is the major tool to get a reliable piece of software.
- 75-80% of the code can be reused *without any change* when switching to more complicated equations. This has been proved already for incompressible Navier-Stokes equations in the serial version of the code.
- The user interested in implementing new numerical algorithms (a problem class library) will never be concerned with low level programming.
- As a consequence of that his code is portable since machine dependencies typically arise only in the *ug* layer. Message passing interfaces are available for several parallel machines (Parsytec, Intel Paragon, PVM).
- The final aim is to have a consistent software environment in 2 and 3 space dimensions on serial and parallel machines.

5 Parallelization

Here we will only cover briefly some basic aspects of the parallel implementation, for a more detailed discussion we refer to [5]. The primary parallelization approach is data partitioning, i.e. the parallelism inherent to multigrid methods is exploited. This poses a restriction on the smoother that can be used in the parallel code. Robustness strategies such as ILU or the special numbering strategies presented in paragraph 3.2 can not be parallelized efficiently, at least on unstructured grids (for structured rectangular grids see [3]). Therefore the parallel implementation currently uses either point Jacobi or inexact Block-Jacobi smoothers (with one or several steps GS or ILU as inner solver).

The data partitioning uses a unique mapping of the elements (of all levels) to the set of processors. For each element t assigned to a processor p, this processor holds also a copy of the nodes of t and of the father element of t and its nodes. This leads to an overlapping storage scheme, where e.g. several copies of one node are stored in different processors. The parallel data management module provides high level routines for exchange of data between these copies.

The stiffness matrix is only assembled per processor and is fully parallel and does not need any communication. Restriction and prolongation proceed without communication as long as each son of an element t is mapped to the same processor as t. The nested grid refinement algorithm can also be parallelized efficiently if there is a refinement rule for each edge refinement pattern possible. In this case the iteration in the green closure of the triangulation can be avoided.

Theoretical investigations and practical experience shows that multiplicative multigrid methods are twice as efficient as additive multigrid methods in terms of computer time on a single computer, since the number of iterations to reach a convergence criterion is doubled, but one iteration of amgm costs not much less than one of mmgm. On a parallel machine the picture may be different for two reasons: First the granularity of amgm is coarser. In the smoother amgm can send all updates for levels $1, \ldots, j$ in one large message, while mmgm must send the updates on all levels seperately, i.e. j smaller messages. The second and more important point is, that different load balancing methods can be used for both methods. In mmgm each grid level must be distributed equally onto all processors while in amgm it is sufficient that the number of elements on all levels is about the same for each processor (if there is no communication in restriction and prolongation).

The most complicated part of the parallelization is the load balancing module which is divided into two parts: The load balancer determines only the new assignment of elements to processors while the load transfer part actually moves the parts of the data structure to their new position. The basic idea for the load balancing part is to first assign the elements to clusters (subsets of elements) and then to assign the clusters to the processors. The clustering process reduces the complexity of the load balancer dramatically, the number of clusters is only proportional to the number of processors. Since the clusters are built via the element hierarchy, they are the key to find a compromise between inter- and intragrid communication. The central element of the cluster assignment algorithm is currently a recursive coordinate bisection strategy. More elaborate techniques will be tested in the future when nonlinear systems are to be solved. Due to the high efficiency of the multigrid method for the simple scalar elliptic problems, care must be taken that the load balancer does not dominate the computation time.

6 Numerical Results

The first four problems show applications of the serial *ug* version, the last two examples give some insight into the performance of the parallel version (more examples can be found in [5]).

6.1 The Skin Problem

As a first test problem we take the following one which is used to model the penetration of drugs through the uppermost layer of the skin (stratum corneum). The stratum corneum is made up of corneocytes which are embedded in a lipid layer. The diffusion is described by the diffusion equation

$$\begin{aligned} -\nabla(D(x,y)\nabla u) + \frac{\partial u}{\partial t} &= 0 \quad \text{in}\Omega \\ u &= 1 \quad \text{on}\Gamma_u \\ u &= 0 \quad \text{on}\Gamma_o \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on}\Gamma_r \cup \Gamma_l \end{aligned} \tag{11}$$

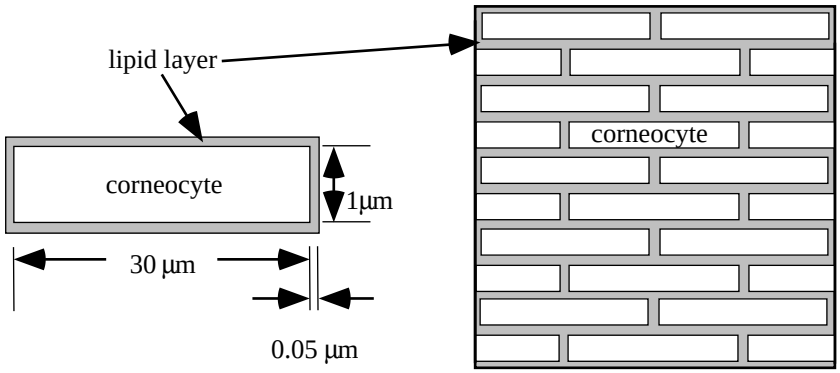


Figure 4: *Right hand side: Structure of skin made up from corneocytes (white) and lipid layers (gray/black). The considered block of stratum corneum is 11μm by 60.2μm. Left hand side: Elementary cell consisting of a corneocyte surrounded by one half of the lipid layer.*

Table 2: *Convergence rate of a (1,1,V)-mmgm applied to the stationary skin problem for various values of D_2 ($D_1 = 1$). The number of unknowns was 54385 on level 5 (6 grid levels).*

D_2	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}
ρ	0.08	0.22	0.39	0.41	0.45	0.45	0.43

where Ω is the unit square and the diffusion coefficient $D(x, y)$ is given by

$$D(x,y) = \begin{cases} D_1 & \text{if } (x,y) \in \text{lipid} \\ D_2 & \text{if } (x,y) \in \text{corneocyte} \end{cases},$$

i.e. it may jump by some orders of magnitude across the corneocyte edges. The corneocytes are very flat and wide cells which in a two-dimensional cross-section are approximated by thin rectangles as shown in Fig. 4.

From Fig. 4 we see that the lipid layer is 0.1μm thick while the corneocytes are 1 by 30μm of size. Since the permeability may jump by some orders of magnitude between lipid and corneocyte, we must align the coarse-grid lines with the interfaces. So we just take the corners of the corneocytes as points for the coarse grid connecting them to form a tensor product grid. Thus we get rid of the problems induced by jumping coefficients. However, we obtain highly anisotropic grid cells in the lipid layer with an aspect ratio of approx. 1:150. Since such an aspect ratio makes the approximation strongly deteriorate and the multi-grid method as well, we use the anisotropic ("blue") refinement strategy to derive a robust multi-grid method and to create a grid which after 5 levels of blue refinement has elements not exceeding an aspect ratio of 1:5. Above that level we refine uniformly. To obtain a robust method on the coarser grids we use an ILU$_\beta$-smoother, cf. [28]. Average convergence factors for a (1,1,V)-cycle are given in Table 2. For more details on this problem see [19].

Table 3: *Robustness of a (1,1,V)-mmgm with ILU-smoother and downwind numbering. The method used 8 locally refined grids to discretize problem the convection-diffusion problem with over 10.000 unknowns on level 8. The convergence rate $\kappa(10)$ is averaged over 10 steps and refers to the finest grid.*

ε	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$\kappa(10)$	0.068	0.067	0.075	0.102	0.092	0.068	0.033	0.018

6.2 Convection-Diffusion Equation

As a second example we show results for the convection-diffusion equation

$$-\varepsilon\Delta u + c \cdot \nabla u = f \tag{12}$$

in the unit square with Dirichlet boundary conditions. We choose c as follows

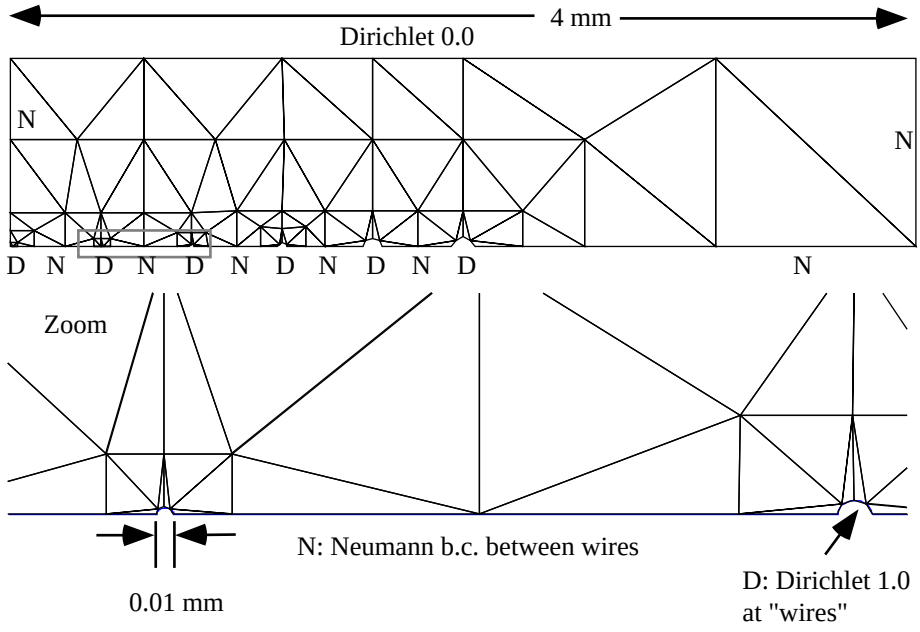
$$c = \left(1 - \sin(\alpha)\left[2\left(x + \frac{1}{4}\right) - 1\right] + 2\cos(\alpha)\left[y - \frac{1}{4}\right]\right)^4 (\cos(\alpha), \sin(\alpha))^T \tag{13}$$

where α is the angle of attack. The boundary conditions are: $u = 0$ on $\{(x,y) : x = 0, 0 \le y \le 1\} \cup \{(x,y) : 0 \le x \le 1, y = 1\} \cup \{(x,y) : x = 1, 0 \le y \le 1\} \cup \{(x,y) : 0 \le x < 0.5, y = 0\}$ and $u = 1$ on $\{(x,y) : 0.5 \le x \le 1, y = 0\}$. The jump in the boundary condition is propagated in direction α. We have div$c = 0$ and c varies strongly on Ω such that the problem is convection dominated in one part of the region and diffusion dominated in another part. As discretization we use a finite volume scheme with first order upwinding for the convective terms on a triangular grid. The grid is refined adaptively using a gradient refinement criterion. As smoother we took a Gauß-Seidel scheme with downwind numbering using algorithm 3.1 in a (1,1,V)-cycle mmgm. It is important to note that the smoother itself is not an exact solver. Thus we should see the benefit of multi-grid in the diffusion dominated part and of the robust smoother in the convection dominated one. This is confirmed by the results given in Table 3. There we show the residual convergence rate averaged over 10 steps for problem (12) on adaptively refined unstructured grids versus ε.

For the same problem with $\varepsilon = 10^{-7}$ the same mmgm but without downwind numbering shows a convergence rate of 0.95 averaged over 40 steps and taking the smoother with downwind numbering but without coarse grid correction as a solver, we end up with a convergence rate of 0.949 as well. This confirms the outlined concept of robust multi-grid. Results of 3d computations can be found in [7].

6.3 Drift Chamber

This problem solves the Laplacian $-\Delta u = 0$ in the domain given by Fig. 5. The boundary conditions are of Dirichlet and Neumann type as indicated in the figure. The feature of this problem are the small wires with Dirichlet boundary conditions that must be resolved on the coarse grid. The smallest wire has a radius of 0.005 mm, while the whole chamber is 4 mm wide and 1 mm thick. So one has to trade off between a coarse grid with



[b]

Figure 5: *Problem definition, coarse grid and zoom for the drift chamber problem.*

few unknowns but a large aspect ratio in grid cells and a coarse grid with equal sized triangles but a large number of unknowns. The grid in Fig. 5 is a reasonable compromise with 85 nodes and 112 triangles but still aspect ratios are large and a robust smoother is required.

Table 4 shows the results of multiplicative and additive multigrid with several different smoothers applied after 3, 4, 5 and 6 levels of uniform refinement. Specifically the smoothers were damped Jacobi with $\omega = 2/3$ (djac), (symmetric) Gauß-Seidel (gs, sgs) and ILU without modification and with $\beta = 0.35$ (ILU, ILU$_\beta$). We make the following remarks:

1. h independent convergence is only achieved with the ILU$_\beta$ smoother. The optimal value was $\beta = 0.35$ but the choice is not very sensitive and good results are achieved with values between 0.2 and 0.5. This corresponds nicely with the theory in [28].

2. The additive method shows qualitatively the same behaviour as the multiplicative multi-grid method but has worse numerical efficiency.

3. Multiplicative multi-grid with a symmetric Gauß-Seidel smoother used as preconditioner in a conjugate gradient method is the only combination giving also relatively satisfactory results, being only a factor 3 slower in computation time than the ILU$_\beta$ smoother.

4. The diverging iteration for ILU without modification can be explained by accumulating roundoff errors. Since the global stiffness matrix is symmetric positive definite but *not* an M-matrix due to obtuse angles the diagonal elements in the ILU decomposition can become very small which leads to instabilities. The modification helps in this case too, since it enlarges the diagonal.

Table 4: *Results for different solver/smoother combinations for the drift chamber problem. Multigrid data: (2,2,V) cycle for Jacobi smoother, $\nu = 1$ for amgm, (2,2,V) cycle for all other smoothers, initial solution $u = 0$, numbers are iterations for a reduction of the residual by 10^{-6} in the euclidean norm. The grid nodes have been ordered lexicographically, iteration numbers exceeding 100 are marked with an asterisk, diverging iterations are marked with $\uparrow$.*

highest level		3	4	5	6
grid nodes		3809	14785	58241	231169
mmgm	djac	*	*	*	*
	gs	79	99	*	*
	sgs	48	59	66	70
	ILU	33	$\uparrow$	$\uparrow$	$\uparrow$
	ILU_β	9	9	9	9
mmgm+cg	djac	31	38	43	43
	sgs	13	16	17	18
	ILU	10	$\uparrow$	$\uparrow$	$\uparrow$
	ILU_β	6	6	6	6
amgm+cg	djac	74	99	*	*
	sgs	36	46	53	57
	ILU	62	$\uparrow$	$\uparrow$	$\uparrow$
	ILU_β	20	24	25	26

6.4 A Shape Design Problem

A nonlinear example computed with *ug* is the following shape design problem. An infinitely long bar consisting of two different materials (hard and soft) and square cross section is to be designed. The ratio of the two materials in a cross section is prescribed and the torsional rigidy is to be maximized. The materials may be mixed at the interface, for details see [16]. The differential equation modelling this situation is given by

$$\begin{aligned} -\mathrm{div}(\varphi(\|\nabla u\|^2)\nabla u) &= 1 \text{ in} (0,1)^2 \\ u &= 0 \text{ on } \partial\Omega \end{aligned} \tag{14}$$

where φ is given by

$$\varphi(x) = \begin{cases} \mu_1 & \text{if } x \geq \frac{2\lambda\mu_2}{\mu_1} \\ \sqrt{\frac{2\lambda\mu_1\mu_2}{x}} & \text{if } \frac{2\lambda\mu_1}{\mu_2} < x < \frac{2\lambda\mu_2}{\mu_1} \\ \mu_2 & \text{if } x \leq \frac{2\lambda\mu_1}{\mu_2} \end{cases}$$

and the constants are: $\lambda = 0.008, \mu_1 = 1, \mu_2 = 2$.

The problem has been solved with global and local grid refinement using either a damped fixed point iteration or a nonlinear multigrid method (with a fixed point iteration as smoother). Table 5 shows the average reduction rates per iteration for the four different methods and different refinement depths.

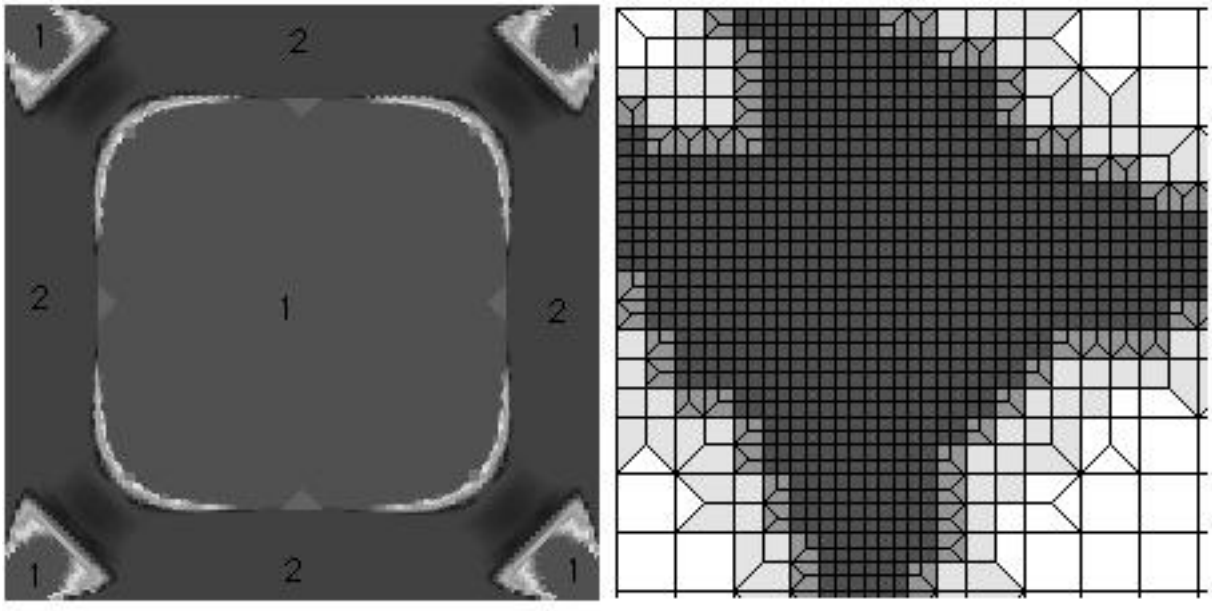


Figure 6: *Left picture shows the function $\varphi(\|\nabla u\|)$ for the shape design problem. Regions labeled with (1) consist of the soft material, regions labeled with (2) consist of the hard material. The right picture shows a zoom into the local refinement in the corner.*

Table 5: *Average convergence rates over 10 iterations for the nonlinear shape design problem using multiplicative multigrid with Gauß-Seidel smoother, and a (2,2,V) cycle.*

j	global refinement		local refinement	
j	nonlinear MG	fixed point	nonlinear MG	fixed point
1	0.05	0.36	0.01	0.1
2	0.40	0.49	0.49	0.1
3	0.73	0.87	0.52	0.57
4	0.62	0.85	0.72	0.76
5	0.78	0.92	0.59	0.62
6			0.65	0.72
7			0.71	0.65

6.5 German Bight

This example is intended to show that multigrid methods on unstructured grids can be parallelized as efficiently as on structured grids. The scalar, time-dependent convection-diffusion equation is solved in the domain given in Fig. 7. The flow field is fixed but nonuniform (it is taken from measurements by the BSH, Hamburg). The coarse grid is already very fine (about 1200 triangles) and is mapped to the processors via orthogonal recursive coordinate bisection [26] (Fig. 7 uses a 3 by 3 processor array).

All parallel results reported below have been computed on a Parsytec transputer system with PARIX 1.2 and 4 MBytes RAM per processor.

Table 6 shows iteration times for a multiplicative multigrid method for up to three levels of refinement (4 grids) and various processor numbers. Execution time increases

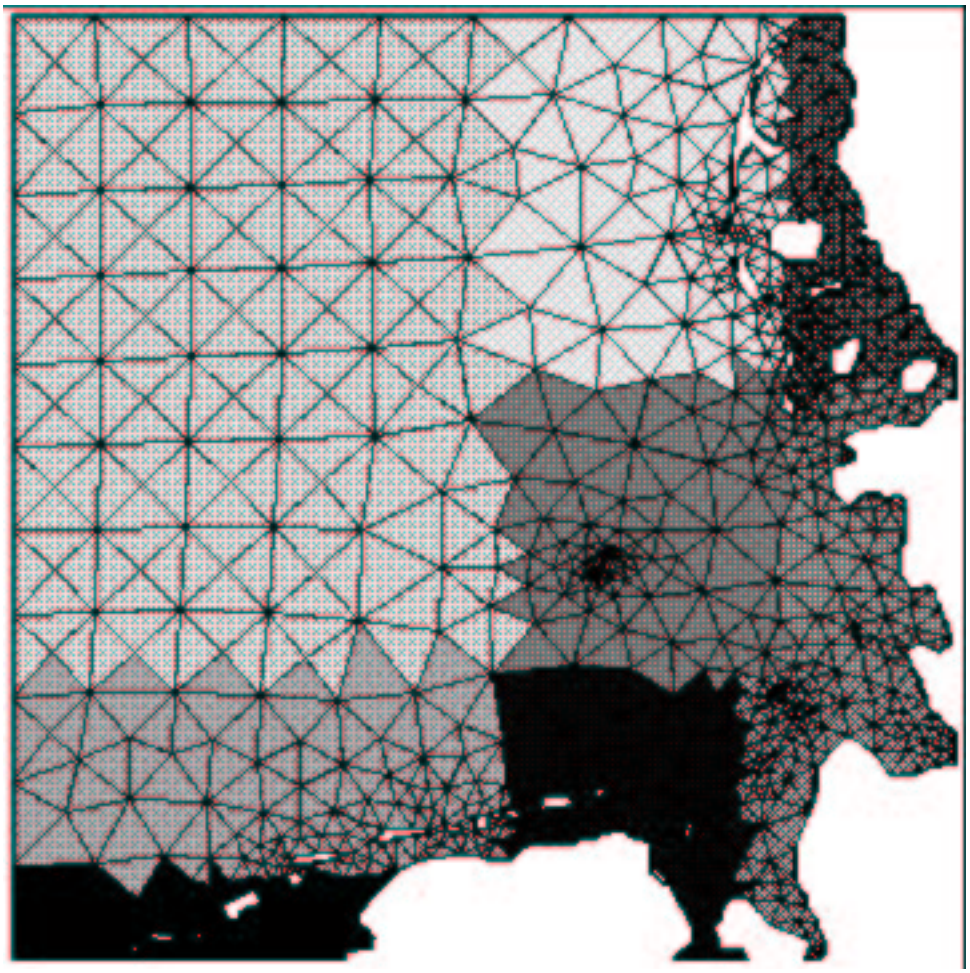


Figure 7: *Triangulation of the German Bight and mapping onto 3 by 3 processors.*

Table 6: *Iteration times in seconds for a (2,2,V) multiplicative multigrid cycle with Block-Jacobi smoother with 2 steps symmetric Gauß-Seidel as inner solver. The residual in the level 0 equation has been reduced by* 10^{-4}.

		Processors						
j	Elements	1	2	4	5	10	16	20
1	5956	7.3	3.8	2.5	1.9	1.3	0.9	0.8
2	23824		16.4	8.8	7.0	3.9	2.6	2.2
3	95296					13.8	8.9	7.3

from 7.3 seconds to 8.9 seconds (22% increase) when going from one to 16 processors with problem size also increased by a factor of 16. Table 7 shows a computation of 50 timesteps on level 2 with various processor numbers from 2 to 20. The computation on 20 processors is 7.1 times faster than on two processors for a fixed problem size.

6.6 A Simple Locally Refined Example

This example illustrates the case of local grid refinement and gives a comparison of additive and multiplicative multigrid in terms of numerical efficiency. The Laplace equation

Table 7: *Total solution time in seconds for 50 steps with implicit euler time integration. Three grid levels have been used with 23824 triangles on the finest grid (fixed problem size for all processor numbers). Within each timestep the residual has been reduced by* 10^{-5} *with the multigrid method from the previous table.*

	Processors					
	2	4	5	10	20	SGI Crimson
Time	3610	1913	1668	964	510	1310
Speedup	1	1.9	2.2	3.7	7.1	

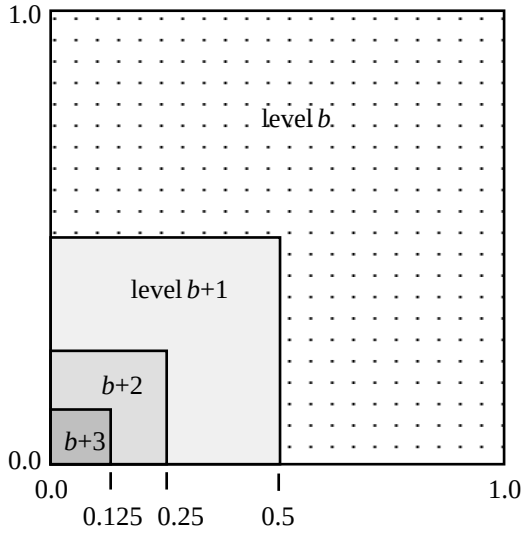


Figure 8: *Grid refinement for the locally refined model problem with* $w = 1$.

with Dirichlet boundary conditions is to be solved in the unit square:

$$\begin{array}{rcll} -\Delta u & = & 0, & \text{in } \Omega = (0,1)^2 \\ u(x,0) & = & \frac{1}{4}x, & x \in [0,1[\\ u(1,y) & = & \frac{1}{4} + \frac{1}{4}y, & y \in [0,1[\\ u(x,1) & = & \frac{1}{2} + \frac{1}{4}(1-x), & x \in [0,1[\\ u(0,y) & = & \frac{3}{4} + \frac{1}{4}(1-y), & y \in]0,1[\end{array} \quad . \tag{15}$$

The initial triangulation T_0 consists of 4 quadrilaterals with one unknown. Refinement is uniform up to some prescribed level b and beginning with level b refinement is such that the grid on level k is restricted to the rectangle $[0, s_k]^2$ with $s_k = (\sqrt{w}/2)^{k-b}, (k > b)$ (see Fig. 8). The factor w is called the "growth factor", since the number of elements on level $k+1$ is defined recursively by $\#T_{k+1} = w\#T_k$. With $w = 4$ one gets uniform refinement and $w = 1$ indicates a case where there is no geometric growth in the number of unknowns. Table 8 shows the results for $w = 1, 2, 3, 4$ and additive and multiplicative multigrid. Note that the problem size is increased with the number of processors. However it is not possible to get always the same number of unknowns per processor for different values of w.

The conclusions drawn from this test are:

Table 8: *Results for different locality of refinement (w, see text) and a varying number of processors. T_{SOL} (total solution time) is for a 10^{-6} reduction in residual norm on level j after a nested iteration. Multigrid data: $\nu_1 = \nu_2 = 1, \gamma = 1$ for mmgm, $\nu = 1$ for amgm with a point Jacobi smoother. Refinement was uniform up to level 4 ($h = 1/32$) except for cases $w = 1$ and $P > 1$ where refinement was uniform up to level 5. N is the number of unknowns after j adaption steps, E_{IT} is the parallel efficiency in one multigrid iteration.*

w		mmgm+jac					cg+amgm+jac				
	P	1	4	16	32	64	1	4	16	32	64
4	j	4	5	6	7	7	4	5	6	7	7
	N	1089	4225	16641	66049	66049	1089	4225	16641	66049	66049
	T_{SOL}	5.95	5.87	6.55	12.49	6.83	9.33	10.17	11.04	20.98	11.43
	E_{IT}		85	75	77	71		88	79	82	76
3	j	5	6	7	7	8	5	6	7	7	8
	N	3553	10657	31656	31656	94440	3553	10657	31656	31656	94440
	T_{SOL}	18.14	15.59	13.09	7.99	15.46	31.38	25.81	21.44	11.97	17.85
	E_{IT}		86	76	71	47		90	80	71	71
2	j	5	7	8	9	10	5	7	8	9	10
	N	2768	12223	24767	49974	99638	2768	12223	24767	49974	99638
	T_{SOL}	15.59	19.05	11.77	12.60	13.44	26.00	28.43	18.68	19.77	20.92
	E_{IT}		78	64	59	55		86	71	74	63
1	j	6	6	10	13	15	6	6	10	13	15
	N	2753	7425	20225	29825	36225	2753	7425	20225	29825	36225
	T_{SOL}	15.39	11.13	10.17	10.69	11.14	24.37	18.49	15.68	13.93	9.96
	E_{IT}		78	60	41	24		85	64	53	45

- The additive method has always better or equal (only one case) efficiencies than the multiplicative method. This is due to coarser grained parallelism and the possibility of using decompositions with smaller interfaces. The latter is the more important point which can be seen by considering the results for $P = 64$ and comparing efficiencies for different w. For $w = 4$ both load balancing methods yield the same decomposition and efficiencies differ not much. For w getting smaller the differences become greater.

- The solution time is always *smaller* for the multiplicative method than for the additive method, except the case of $w = 1, P = 64$. This is not a result of the small number of unknowns per processor. In contrary to the case $w > 1$, one can observe for $w = 1$ that efficiencies do not increase with problem size above level 10 for a fixed number of processors. This is due to the fact that the unknowns do not grow geometrically with the number of levels (no decrease of the surface to volume ratio).

- In the case where additive multigrid is equal or better than multiplicative multigrid in terms of total computation time both methods have to be considered as inefficient. Since additive multigrid is a factor of two (roughly) more expensive on a serial computer, parallel efficiency for multiplicative multigrid must be below 50% in order to allow additive multigrid to be better. Since there are also losses in the latter method, the break even point happens to be at 25% efficiency for multiplicative and 50% efficieny for additive multigrid in this example.

References

[1] R. E. Bank, T. F. Dupont, H. Yserentant: *The Hierarchical Basis Multigrid Method* , Numer. Math., **52**, 427-458 (1988).

[2] R. E. Bank: *PLTMG: A software package for solving elliptic partial differential equations. Users Guide 6.0.* SIAM, Philadelphia, 1990.

[3] P. Bastian, G. Horton: *Parallelization of Robust Multigrid Methods: ILU Factorization and Frequency Decomposition Method.* SIAM J. Sci. Stat. Comput., **12**, No. 6, pp. 1457-1470, 1991.

[4] P. Bastian, G. Wittum: *On Robust and Adaptive Multigrid Methods.* In: Proceedings of the 4^th European Multigrid Conference, Amsterdam, July 1993, to appear.

[5] P. Bastian: *Parallel Adaptive Multigrid Methods.* IWR Report 93-60, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, 1993.

[6] —: *Locally Refined Solution of Unsymmetric and Nonlinear Problems.* In: Hackbusch, W., Wittum, G. (eds.): Incomplete Decompositons - Theory, Algorithms, and Applications, NNFM, vol. 41, Vieweg, Braunschweig, 1993.

[7] J. Bey, G. Wittum: *A Robust Multigrid Method for the Convection-Diffusion Equation on locally refined grids.* In: Adaptive Methods, Proceedings of the Ninth GAMM Seminar, Notes on Numerical Fluid Mechanics, Vieweg Verlag, Braunschweig, 1993, to appear.

[8] A. Brandt: *Guide to Multigrid Development.* in Hackbusch W., Trottenberg U. (eds.): Multigrid Methods. Proceedings Köln-Porz, 1981. Lecture Notes in Mathematics, Bd. 960, Springer, Heidelberg, 1982.

[9] J. H. Bramble, J. E. Pasciak, J. Xu: *Parallel Multilevel Preconditioners*, Math. Comput., **55**, 1-22 (1990).

[10] J. H. Bramble, J. E. Pasciak, J. Wang, and J. Xu, *Convergence estimates for multigrid algorithms without regularity assumptions*, Math. Comp., **57**, (1991), pp. 23–45.

[11] R. P. Fedorenko: *Ein Relaxationsverfahren zur Lösung elliptischer Differentialgleichungen.* (russ.) UdSSR Comput Math Math Phys 1,5 1092-1096 (1961).

[12] A. Greenbaum: *A Multigrid Method for Multiprocessors.* Appl. Math. Comp., **19**, 75-88 (1986).

[13] W. Hackbusch: *Multi-grid methods and applications.* Springer, Berlin, Heidelberg (1985).

[14] —: *Theorie und Numerik elliptischer Differentialgleichungen.* Teubner, Stuttgart, 1986.

[15] —: *The Frequency Decomposition Multi-grid Method.* Part I: Application to Anisotropic Equations. Numer. Math., 1989.

[16] B. Kawohl, J. Stara, G. Wittum: *Analysis and Numerical Studies of a Shape Design Problem.*, Archive for Rational Mechanics, **114**, 349-363, 1991.

[17] R. KETTLER: *Analysis and comparison of relaxation schemes in robust multi-grid and preconditioned conjugate gradient methods.* In: Hackbusch,W., Trottenberg,U. (eds.): Multi-Grid Methods, Lecture Notes in Mathematics, Vol. 960, Springer, Heidelberg, 1982.

[18] R. KORNHUBER, R. ROITZSCH: *On Adaptive Grid Refinement in the Presence of Boundary Layers.* Preprint SC 89-5, ZIB, Berlin, 1989.

[19] R. LIECKFELDT, G. W. J. LEE, G. WITTUM, M. HEISIG: *Diffusant concentration profiles within corneocytes and lipid phase of stratum corneum.* Proceed. Intern. Symp. Rel. Bioact. Mater., 10 (1993) Controlled Release Society, Inc.

[20] W. A. MULDER: *A New Multigrid Approach to Convection Problems.* J. Comp. Phys., **83**, 303-323 (1989).

[21] M. C. RIVARA, *Design and data structure of a fully adaptive multigrid finite element software*, ACM Trans. on Math. Software, 10 (1984), pp. 242–264.

[22] R. STEVENSON: *On the robustness of multi-grid applied to anisotropic equations: Smoothing- and Approximation-Properties.* Preprint Rijksuniversiteit Utrecht, Wiskunde, 1992.

[23] —: *New estimates of the contraction number of V-cycle multi-grid with applications to anisotropic equations.* In: Hackbusch, W., Wittum, G. (eds.) : Incomplete Decompositions, Algorithms, theory, and applications. NNFM, vol 41, Vieweg, Braunschweig, 1993.

[24] P. WESSELING: *A robust and efficient multigrid method.* In: Hackbusch, W., Trottenberg, U. (eds.): Multi-grid methods. Proceedings, Lecture Notes in Math. 960, Springer, Berlin (1982).

[25] —: *Theoretical and practical aspects of a multigrid method.* SIAM J. Sci. Statist. Comp. **3**, (1982), 387-407.

[26] R. D. WILLIAMS: *Performance of Dynamic Load Balancing Algorithms for Unstructured Mesh Calculations*, Report C3P 913, California Institute of Technology, Pasadena, CA., (1990).

[27] G. WITTUM: *Filternde Zerlegungen - Schnelle Löser für große Gleichungssysteme.* Teubner Skripten zur Numerik Band 1, Teubner, Stuttgart, 1992.

[28] —: *On the robustness of ILU-smoothing.* SIAM J. Sci. Stat. Comput., **10**, 699-717 (1989).

[29] —: *Linear iterations as smoothers in multi-grid methods.* Impact of Computing in Science and Engineering, **1**, 180-215 (1989).

[30] J. XU: *Multilevel theory for finite elements.* Thesis, Cornell Univ., 1988.

[31] —: *Iterative Methods by Space Decomposition and Subspace Correction: A Unifying Approach*, SIAM Review, **34**(4), 581-613, (1992).

[32] H. YSERENTANT: *Über die Aufspaltung von Finite-Element-Räumen in Teilräume verschiedener Verfeinerungsstufen.* Habilitationsschrift, RWTH Aachen, 1984.