Algebraic Multigrid Solver for Discrete Elliptic Second Order Problems

Ferdinand Kickinger
Institut für Mathematik
Johannes Kepler Universität Linz

Abstract

In this paper, an Algebraic Multigrid Solver (AMG++) is presented. This solver is especially suited for the fast solution of large sparse systems $A_h u_h = f_h$ of algebraic equations arising from finite element (FE) discretisation of second-order elliptic boundary value problems (BVP) on unstructured refined meshes in one, two and three dimensions as well. The only information used is recovered from the stiffness matrix A_h. More precisely, the information hidden in the stiffness matrix allows us, to construct "coarse grids", the intergrid transfer operators (restriction and prolongation) and the coarse grid matrices. These tools together with some smoother and a solver for the equations on the coarsest grid build up a complete multigrid algorithm. In special cases, we observe, that the AMS completely coincides with some standard geometrical multigrid algorithms. The AMS has been applied to two and three dimensional examples. The algebraic multigrid algorithm shows the same numerical behaviour as it is known from the geometrical multigrid algorithms. Symmetric variants of the algebraic multigrid method can be used for preconditioning a conjugate gradient solver. This combination is more robust and improves the convergence rate considerably. This paper was supported by the Austrian 'Fonds zur Förderung der wissenschaftlichen Forschung (FWF)' within the project P10643-TEC, 'Domain Decomposition Methods in Structural Mechanics'.

1 Introduction

During the last years, the importance of the numerical simulation has been growing more and more. This is certainly due to the rapid development of the hardware with respect to the resources in memory and arithmetic power of the processors (see table below).

	1985	1995
RAM	640 KByte	512 MByte
MFlop	0.X	X00.
HDD	20 MByte	8 GByte

Parallel machines allow us to multiply these resources by the number of pocessors available. The software is always a little bit behind the progress in the hardware. This is especially true for application software. In industry, technical experiments for developping and improving new products are more and more replaced by computer simulations, provided that they are cheaper and as realistic as experiments, or even better. Imagine, that we have to develop a new product, for example an electrical machine or a combustion engine. The "classical" way to do this is to use experience for developing a prototype, then do some experiments, improve the prototype and so on. Doing the same with numerical simulation, we must not build so many prototypes, because we can test on the computer, and make the nessecary improvements also on the computer. Furthermore, the optimization process can be modeled mathematically, and therefore, it can be carried out on the computer. This is called *automatic optimal design*. The computer aided design and the automatic optimal desingn need both a lot of calls of the simulation routines. This is the reason why we need fast simulation codes. This is already true for two dimensions. Conserning simulations on the basis of Partial Differential Equations

(PDEs) and its Finite Element (FE) discretisations, the numerical algorithm in the heart of the most simulation codes and especially of commercial simulation codes, are far from being optimal with respect to the complexity in aritmetic and memory.

The disadvantage of many mesh generators consists in the fact that they are producing some fine grid with no hierarchical structure in it. Thus, the application of a geometrical standard multigrid solver is not possible as long as no coarsening strategies are available. Recently, such coarsening strategies have been proposed by R. E. Bank and J. Xu in two dimensions (see [16]). Being aware of that problem, we have been working on an Algebraic Multigrid Solver from the very beginning. In this context, "Algebraic" means, that we put a matrix K_h and a righthandside f_h into a black box "Solver" and get out the solution u_h of the algebraic system $K_h u_h = f_h$. No mesh information is required to obtain the solution !! The algebraic multigrid solver should converge at least nearely as fast as the geometrical version. In this case, the disadvantage of the mesh generator mentioned above turns out to be an advantage, because the AMG (Algebraic MultiGrid) works on unstructured fine, nested and caotically refined meshes nearely as well as the standard multigrid method. Another advantage of the AMG is the fact, that it is easy to exchange the solver routine in some standard finite element package by the AMG-routine. We have to put in only the solver, it is not necessary to change the meshing strategie and the whole data structures. Also adaptive mesh refinement causes troubles for standard multigrid (see [12]) which disappear in the algebraic case. The first serious algebraic approach to multigrid methods was made more than 10 years ago by J. W. Ruge and K. Stüben, in [7]. However, this approach was restricted to M-matrices, reqular grids or require the knowledge of the coordinats in every meshpoint.

Recently, some revival of algebraic multigrid techniques has been observed ([8], [9], [17], [18], [19], [4]). This is certainly due to the sucess of multilevel techniques (see [5]) as well as the enormous progress in the hardware. Our approach is based on the topological graph of the matrix defined in [11]. A coarsening algorithm working on this graph gives some coarser graph from which and from the fine grid matrix the coarse grid matrix can be derived by Galerkin projection, provided that some prolongation is defined. At first we define the prolongation matrix and obtain the restriction operator by transposing the prolongation matrix. In order to complete the multigrid algorithm, it remains to define some smoothing iteration and some solver for the system araizing at the "coarsest" grid. These components form a complete multigrid cycle that can be used immediately as a solver, or, in a symmetric version, as preconditioner in the conjugate gradient (CG) iteration. The only conditions that restricts the application of the algebraic multigrid solver consists in the fact, that the underlying PDE is of second order, and the unknown approximate the nodal values of the unknown function (the unknown should not be approximate derivatives !!). This means, that we should use *Lagrangian*-typed finite elements.

2 Coarse Grid Generation Based on Topological Matrix Graphs

The main task in Algebraic Multigrid is based on finding some coarse grids, where the required information needs not to be a real coarsenig of the given mesh. Using Galerkins method, we have to search out a prolongation and a restriction operator, to generate some coarse grid corrections.

We know, that the matrix contains mesh information. This is at least, that unknowns, which are connected in the matrix, are neigbours in the mesh. Therefore we need the following definition.

Definition 1 *Let $A \in \mathbf{K}^{I\times I}$ be a matrix with index set I. As the* **Graph** $G(A)$ *of the matrix $A = [a_{\alpha\beta}]_{\alpha\beta\in I}$ we denote the following subset of $I \times I$:*

$$G(A) = \{(\alpha, \beta) \in I \times I : a_{\alpha,\beta} \neq 0\}. \tag{1}$$

$$
\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}
\qquad
\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 \end{bmatrix}
\tag{2}
$$

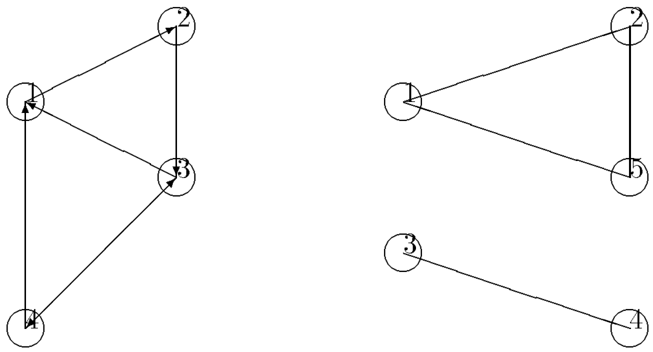


Figure 1: Matrices and corresponding graphs

We can describe the set $G(A)$ as follows: The elements $\alpha \in I$ are denoted as vertices and the $(\alpha, \beta) \in G(A)$ are denoted as edges, going from α to β. If the the matrix A is symmetric, the graph $G(A)$ is symmetric. But also in the case $a_{\alpha,\beta} \neq 0, a_{\beta,\alpha} \neq 0 \quad \forall a_{\alpha,\beta} \neq 0$ the graph is symmetric. (See also Figure 1.)

Let us assume, that the graph is numbered (take the numbering of the matrix), then we can write down an algorithm, where we put in a (symmetric) graph and get out a coarser (symmetric) graph.

Algorithm 1 *Coarse Graph Generator*

1. Put graph $G(A)$ in.

2. Go through the vertices in $G(A)$ and build up a new Graph I.

   - let α be the actual vertex
   - if (this vertex of $G(A)$ is not marked as **visited**)
   - put α into I, and mark this node in $G(A)$ as **visited**
   - go through the connected vertices of α and
     - let β be the actual vertex with $(\alpha, \beta) \in G(A)$
     - if (this vertex of $G(A)$ is not marked as **visited**)
     - put (α, β) into I
     - mark β in $G(A)$ as **visited**

Figure 2 shows us the action of the above algorithm on some simple graph. The nodes in the directed graph I where the edges start, are some kind of topolocical coarse grid points. This way of finding a *coarse grid* is very simple, but useful and fast to compute. The time for calculating the coarse grid structure is about the computation a vector scalar product.

Out of this algorithm, there comes some kind of unknown-clustering, where neigboured unknowns are put together. (In groups of about two, in the one dimensional case, in groups of about four, in the two dimensional case, in groups of about eight, in the three dimensional

$$
\begin{array}{ccccccccccccccccccc}
 & & & & 16 & - & 17 & - & 18 & & & & & & \mathbf{16} & & 17 & & 18 \\
 & & & & | & / & | & / & | & & & & & & & & \uparrow & \nearrow & \\
 & & 12 & - & 13 & - & 14 & - & 15 & & & & 12 & & 13 & & \mathbf{14} & \rightarrow & 15 \\
 & & | & / & | & / & | & / & | & & & & \uparrow & \nearrow & & & & & \\
 & & 8 & - & 9 & - & 10 & - & 11 & \rightarrow & & & \mathbf{8} & & 9 & & 10 & & 11 \\
 & / & | & / & | & / & | & / & | & & & & & & \uparrow & \nearrow & & & \uparrow \\
3 & - & 4 & - & 5 & - & 6 & - & 7 & & 3 & & 4 & & \mathbf{5} & \rightarrow & 6 & & \mathbf{7} \\
| & / & | & / & & & & & & & \uparrow & \nearrow & & & & & & & \\
1 & - & 2 & & & & & & & & \mathbf{1} & \rightarrow & 2 & & & & & &
\end{array}
\tag{3}
$$

Figure 2: Generation of graph I

Figure 3: Coarsening

case.) If we do the coarsening and build up a mesh, putting the new unknown of the coarse grid in the middle of the clusterd unknowns of the fine grid, and interpret this as mesh, we can see what is going on in Figure 3

With this Graph I, we are now going to build up the Galerkin Interpolator.

3 Intergrid Transfer Operators and Coarse Grid Matricies

Once the coarse grid is specified, it is relatively easy to define the prolongator (interpolator) $\mathbf{I}_G$, mapping some vector from the coarse grid space $\mathbf{R}^{N_{corse}}$ to a vector of the fine grid space $\mathbf{R}^{N_{fine}}$ Let us number the *coarse grid points* in I. Then we can write down the Interpolator $\mathbf{I}_G = [a_{ij}] - N_{corse} \times N_{fine} - matrix$ as follows:

$$
\mathbf{I}_G = (a_{ij}) := \left\{ \begin{array}{ll} a_{ij} = 1 & \text{if } j \text{ is the renumberd } i \text{ (when it is a coarse grid point)} \\ & \qquad \text{if } {\scriptstyle (j,i) \in I} \text{ (j is renumberd)} \\ & \qquad 0 \quad \text{otherwise} \end{array} \right\} \tag{4}
$$

The Galerkin Interoplator for Figure 3 looks like:

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}^t \tag{5}$$

Knowing that the restriction

$$\mathbf{R}_G = \mathbf{I}_G^t \tag{6}$$

we can write the *coarse grid system* as

$$\mathbf{R}_G \mathbf{K} \mathbf{I}_G u_{coarse} = \mathbf{R}_G d_{fine}. \tag{7}$$

We demonstrate this algorithm on a few examples: Let us consider the simplest 1D second order boundary value problem:

Example 1 *1D*

$$-u'' = f \ \ in\ (0,1) \quad u(0) = 0, \quad u(1) = 0. \tag{8}$$

Taking the weak formulation, this reads as follows: Find $u \in V_0 = H_0^1(\Omega)$ such that

$$\int_0^1 u'(x)v'(x)dx = \int_0^1 f(x)v(x)dx \quad \forall v \in V_0 \tag{9}$$

with given $f \in L_2(0,1)$. FE-discretisation on an uniform grid with piecewise linear Ansatz-functions and $h = \frac{1}{7}$, results in the stiffness matrix:

$$K = \frac{1}{h} \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}, \tag{10}$$

the interpolator defined above has the matrix representation:

$$\mathbf{I}_G = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}. \tag{11}$$

Now it is easy to compute the Galerkin coarse grid matrix

$$K_{coarse} = \mathbf{I}_G^t K \mathbf{I}_G = \frac{1}{h} \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}. \tag{12}$$

The simplest 2D problem of second order is:

Example 2 *2D*

$$-\triangle u = f \quad \in (0,1)^2 \ \textit{and}\ u = 0 \ \textit{on}\Gamma = \partial\Omega. \tag{13}$$

Taking again the weak formulation, we end up with: Find $u \in V_0 = H_0^1(\Omega)$ such that

$$\int_\Omega \mathbf{grad}u(x)\mathbf{grad}v(x)dx = \int_\Omega f(x)v(x)dx \quad \forall v \in V_0 \tag{14}$$

with given $f \in L_2(\Omega)$. FE-discretisation on an uniform grid with piecewise linear Ansatz-functions and an arbitary h results in the stiffness matrix:

$$K = \begin{pmatrix} 4 & -1 & & -1 & & \\ -1 & \ddots & \ddots & & \ddots & \\ & \ddots & & & & -1 \\ -1 & & & & \ddots & \\ & \ddots & & \ddots & \ddots & -1 \\ & & -1 & & -1 & 4 \end{pmatrix} \tag{15}$$

with the interpolator:

$$\begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \\ \uparrow & \nearrow & \uparrow & \nearrow & \cdots \\ \bullet & \rightarrow & \bullet & \rightarrow & \cdots \\ \uparrow & \nearrow & \uparrow & \nearrow & \cdots \\ \bullet & \rightarrow & \bullet & \rightarrow & \cdots \end{pmatrix} \tag{16}$$

projecting on the coarse grid, we get

$$K_{coarse} = \mathbf{I}_G^t K \mathbf{I}_G = \begin{pmatrix} 8 & -2 & & -2 & & \\ -2 & \ddots & \ddots & & \ddots & \\ & \ddots & & & & -2 \\ -2 & & & & \ddots & \\ & \ddots & & \ddots & \ddots & -2 \\ & & -2 & & -2 & 8 \end{pmatrix} \tag{17}$$

4 Details

In this Algebraic Multigrid Method, we have used a constant interpolation on neigboured nodes. For elliptic equations, with unknowns of the function value, it works well, but what happens on equations of 4th order?

Another thing of interest is, which cycle we should use. Testing a lot of configurations, we find, that the V11-cycle ist the most efficient with respect to computational time for all the tested examples. The fastes way of reducing the error is using a CG-Method, preconditioned with the AMGM V11 procedure. This means one forward Gauss-Seidel smoothing, then the recursiv call of the procedure and one backward call of the Gauss-Seidel smoothing.

Taking a look at the smoother and the according damping factor, we see, that for damping factor $\omega = 1$ ($x^{m+1} = x^m + \omega W^{-1}(Ax - f)$) we obtain the best results (also $\omega > 1$ was tested).

Testing out problems with local refined grids, or with different materials, no problems occur. (See [28])

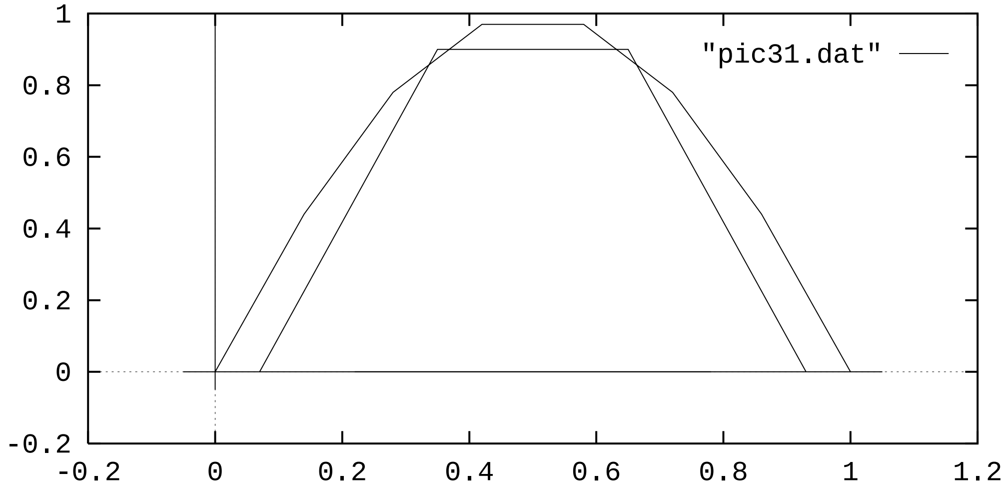


Figure 4: approx. of the Eigen-function

5 Heuristics

5.1 The Coarse Grid Matrices

If we look in simple cases at the araising corse grid matrices, we can interpret them as FEM-matrices of corresponding problems, with different discretisation parameter h.

5.1.1 1D case

We look at the 1D problem as above with start refinement $n = 2^i$. Calculating the FEM-Matrix with linear Ansatz-functions, we get the recursivly defined coarse grid matrices as FEM-matrices according to the following discret problems (*level* is denoting the level of recoarsing where zero is the fine grid):

$$\begin{array}{l} -u'' = f \quad \text{ in } (\sum_{j=1}^{level} \frac{2^j}{4\cdot n}; 1 - \sum_{j=1}^{level} \frac{2^j}{4\cdot n}) \\ \text{with zero dirichlet boundary conditions.} \end{array} \tag{18}$$

This means, we have a coarse grid approximation of the boundary of order h.

Picture 4 shows us the approximation of the coaresest Eigen-function in the model for the fine grid $h = \frac{1}{7}$ and the first coarsening.

5.1.2 2D case

In this case, in the same way, the corresponding problems to the coarse grid matrices (fine grid discretisation choosen, that $n = 2^i$ unknowns are in each direction) are:

$$- \triangle u = f \quad \text{ in } [\sum_{j=1}^{level} \frac{2^j}{4\cdot n}; 1 - \sum_{j=1}^{level} \frac{2^j}{4\cdot n}]^2 \quad u_{|\partial\Omega} = 0 \tag{19}$$

The approximation of the boundary we can see in figure 5.

5.2 A Model Problem

Let us take a look at the model problem $-\Delta u = f$ in one two and three dimensions with dirichlet boundary conditions. Taking the weak formulation we get the following problem:

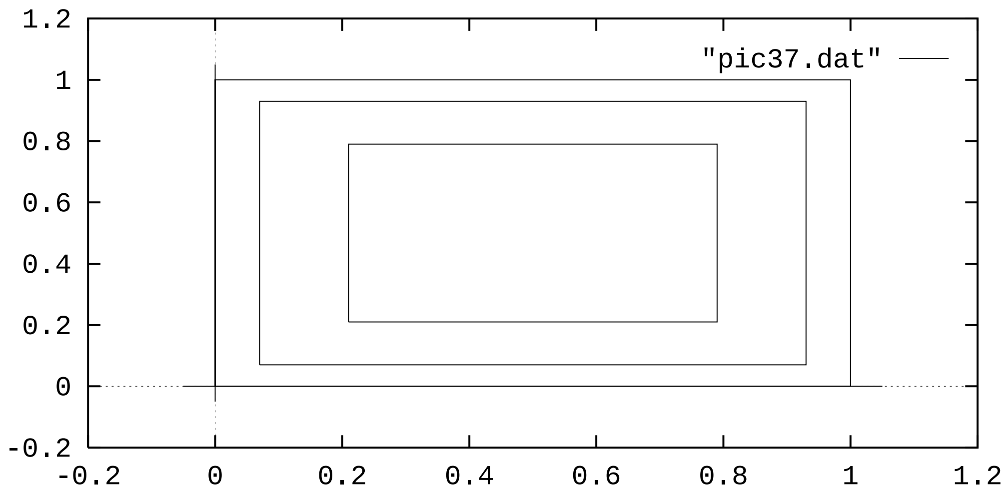


Figure 5: approx. of the boundary of the unit square

Find $u \in V = H_0^1(\Omega)$

$$a(u,v) = f(v), \quad \forall v \in V$$

$$\text{with} \quad a(u,v) = \int_\Omega \bigtriangledown u \cdot \bigtriangledown v dx \text{ and } f(v) = \int_\Omega f v dx \tag{20}$$

For $f \in L_2(\Omega)$ and some regularity conditions on the domain Ω we obtain the following inequality, if we use piecewise linear Ansatz functions and a regular mesh:

$$\|u - u_h\|_1 \leq Ch\|f\|_0, \tag{21}$$

where

$$\begin{array}{rcl} u & - & \text{exact solution,} \\ u_h & - & \text{FEM- approximation of the solution,} \\ h & - & \text{discretisation parameter.} \end{array}$$

Using the argument of Aubin-Nitsche (see [11]) we get also a result for the L_2 Norm:

$$\|u - u_h\|_0 \leq \hat{C}h^2\|f\|_0. \tag{22}$$

Taking the discret system we get the following linear equations:

$$A_h \bar{u}_h = \bar{f}_h \tag{23}$$

where

$$\begin{array}{rcl} A_h & = & [a_{ij}]_{i,j \in I} \text{ with } a_{ij} = a(p_i, p_j), \\ \bar{u}_h & \leftrightarrow & u_h = \sum_{i \in I} [\bar{u}_h]_i p_i, \\ [\bar{f}_h]_i & = & \int_\Omega p_i f dx \end{array}$$

If we take the algebraic coarsening introduced in the sections before we get:

$$\begin{array}{c} a(p_i + p_j, p_k + p_l) = \\ a(p_i, p_k) + a(p_i, p_l) + a(p_j, p_k) + a(p_j, p_l) \end{array} \tag{24}$$

This means, by a renumbering of the unknowns, such that neigboured unknowns are put together, we get a blockstructure where the coarse grid matrix is the matrix of the sum of all entries in each block.

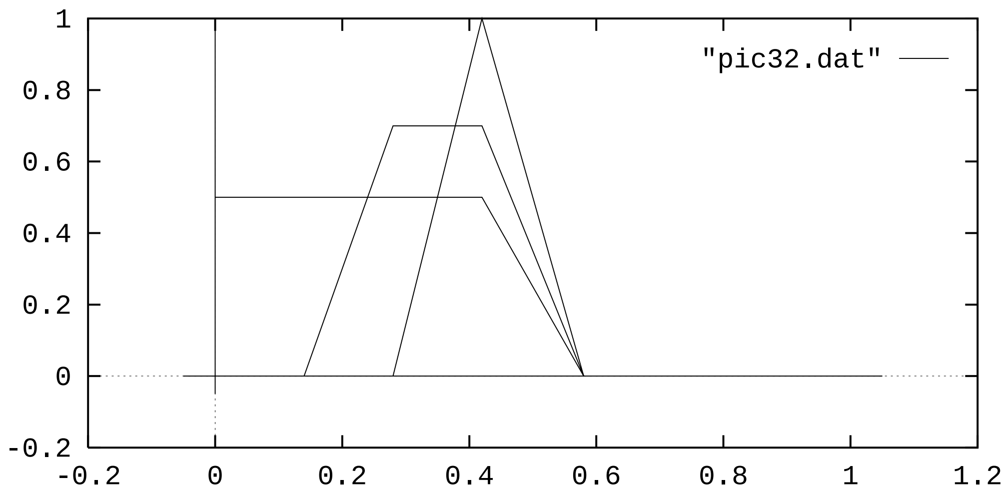


Figure 6: Ansatz-functions

Example 3 $-\Delta u = f \ in \ (0,1) \ h = \frac{1}{7}$

$$\frac{1}{h^2} \cdot \begin{array}{|cc|cc|c|}\hline 2 & -1 & & & \\ -1 & 2 & -1 & & \\ \hline & -1 & 2 & -1 & \\ & & \cdots & \cdots & \\ \hline & & & & \\ & & & & \\ \hline \end{array} \rightarrow \frac{1}{h^2} \cdot \begin{array}{|c|c|c|}\hline 2 & -1 & \\ \hline -1 & 2 & -1 \\ \hline & -1 & 2 \\ \hline \end{array} \tag{25}$$

We get coarse grid Ansatzfunktion like shown in Figure 6.

5.3 A simple Basis Transformation

We start at the discret system

$$A_h \bar{u}_h = \bar{f}_h \tag{26}$$

with the coarse grid system

$$A_H \bar{u}_H = \bar{f}_H, \quad A_H = RA_hP, \quad \bar{f}_H = R\bar{f}_h. \tag{27}$$

in the one dimensional case we get a basis for the coarse grid space by: [[.. 00 11 ..]]. For the $N(R)$ we can give also a basis: [[.. 00 1 − 1 ..]] (see Figure 7).

If we Take the system for $-u'' = f$ in $[0,1]$ $h = \frac{1}{13}$ with homogenious Dirichlet boundary conditions, we get by transforming, the following system:

$$\frac{1}{h^2} \cdot \left|\begin{array}{cccccc|cccccc} 2 & -1 & & & & & & -1 & & & & \\ -1 & 2 & -1 & & & & 1 & & -1 & & & \\ & -1 & 2 & -1 & & & & 1 & & -1 & & \\ & & -1 & 2 & -1 & & & & 1 & & -1 & \\ & & & -1 & 2 & -1 & & & & 1 & & -1 \\ & & & & -1 & 2 & & & & & 1 & \\ \hline & 1 & & & & & 6 & 1 & & & & \\ -1 & & 1 & & & & 1 & 6 & 1 & & & \\ & -1 & & 1 & & & & 1 & 6 & 1 & & \\ & & -1 & & 1 & & & & 1 & 6 & 1 & \\ & & & -1 & & 1 & & & & 1 & 6 & 1 \\ & & & & -1 & & & & & & 1 & 6 \end{array}\right| \cdot \left|\begin{array}{c} u_1+u_2 \\ u_3+u_4 \\ u_5+u_6 \\ u_7+u_8 \\ u_9+u_{10} \\ u_{11}+u_{12} \\ \hline u_1-u_2 \\ u_3-u_4 \\ u_5-u_6 \\ u_7-u_8 \\ u_9-u_{10} \\ u_{11}-u_{12} \end{array}\right| = \left|\begin{array}{c} f_1+f_2 \\ f_3+f_4 \\ f_5+f_6 \\ f_7+f_8 \\ f_9+f_{10} \\ f_{11}+f_{12} \\ \hline f_1-f_2 \\ f_3-f_4 \\ f_5-f_6 \\ f_7-f_8 \\ f_9-f_{10} \\ f_{11}-f_{12} \end{array}\right|, \tag{28}$$

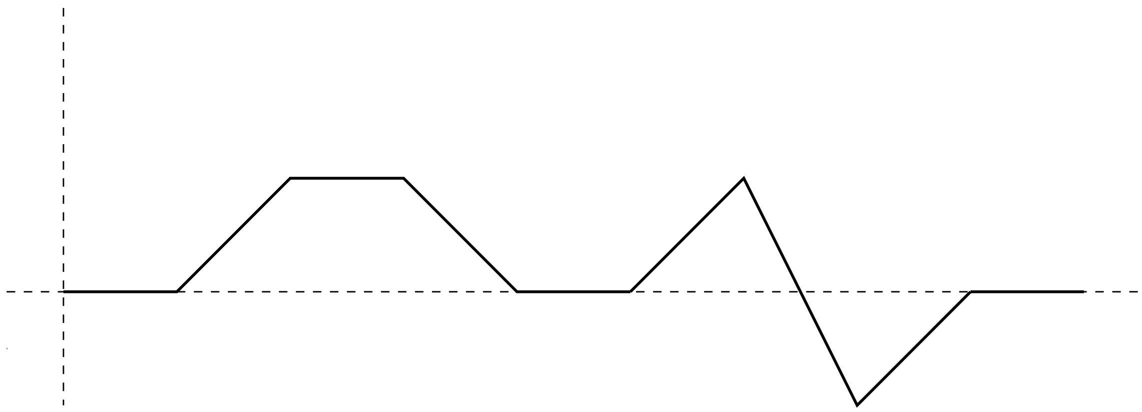


Figure 7: Ansatz-functions for $N(R)^{\perp}$ and $N(R)$

which we can rewrite in the form

$$\begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \cdot \begin{vmatrix} \hat{u} \\ \hat{u}' \end{vmatrix} = \begin{vmatrix} \hat{f} \\ \hat{f}' \end{vmatrix}. \tag{29}$$

We see that A_{11} is our coarse grid matrix. Before we do any calculations in this representation, we see, that the Ansatz-functions for the $N(R)^{\perp}$ are, for $h \to \infty$, an approximation of the Dirac impulse. This means, that elements in $N(R)^{\perp}$ approximate the function value. The basis for $N(R)$ is, for $h \to \infty$, an approximation of the derivative of the Dirac impulse, this means, that elements in $N(R)$ approximate the derivative of the solution.

Recall that the above system is equivalent to the origin FEM-system.

In the infinit dimensional case, if all is sufficient smooth, we can also write formally instead of $-u'' = f$

$$-\begin{vmatrix} \frac{\partial^2}{\partial x^2} & -\frac{1}{2}\frac{\partial}{\partial x} \\ -\frac{1}{2}\frac{\partial}{\partial x} & \frac{1}{2}(I - \frac{\partial^2}{\partial x^2}) \end{vmatrix} \cdot \begin{vmatrix} u \\ u' \end{vmatrix} = \frac{1}{2}\begin{vmatrix} f \\ f' \end{vmatrix}. \tag{30}$$

In the two dimensional case the above calculations can be done again for the special case of $-\Delta u = f, u_{\Gamma} = 0$ in $(0,1)^2$ with unknowns in $i/n, j/n$ with $0 < i, j < n$. We get the basis transformation with the following Ansatz-functions (see Figure 8 and 9).

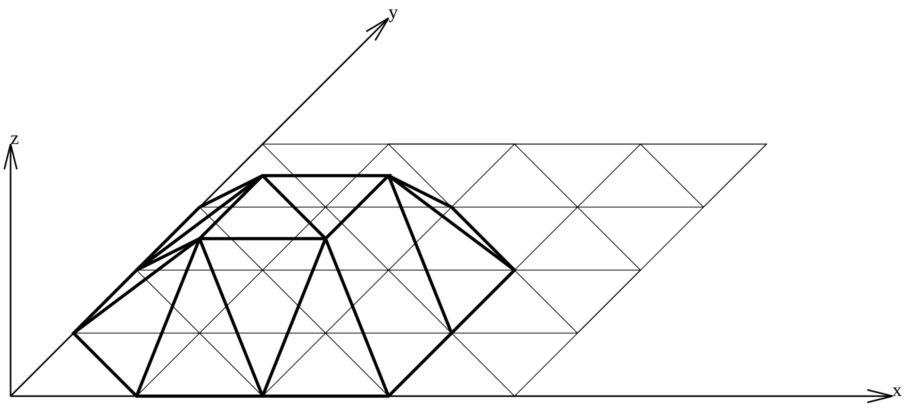
$$\begin{aligned} [...1, 1, 1, 1...] &\quad \ldots \quad u \\ [...1, -1, 1, -1...] &\quad \ldots \quad \frac{\partial u}{\partial x} \\ [...1, 1, -1, -1...] &\quad \ldots \quad \frac{\partial u}{\partial y} \\ [...1, -1, -1, 1...] &\quad \ldots \quad \frac{\partial^2 u}{\partial x \partial y} \end{aligned}$$

Back to our transformed equation

$$\begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \cdot \begin{vmatrix} \hat{u} \\ \hat{u}' \end{vmatrix} = \begin{vmatrix} \hat{f} \\ \hat{f}' \end{vmatrix}. \tag{31}$$

Here we can write for the coarse grid system

$$\begin{vmatrix} A_{11} & 0 \\ 0 & 0 \end{vmatrix} \cdot \begin{vmatrix} \hat{u} \\ 0 \end{vmatrix} = \begin{vmatrix} \hat{f} \\ 0 \end{vmatrix}, \tag{32}$$



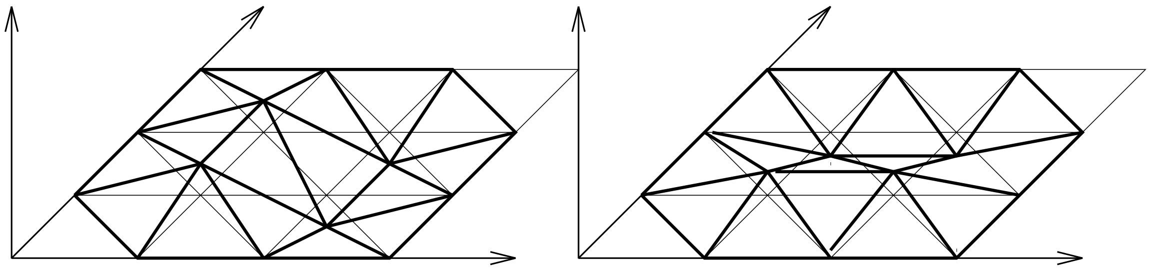


Figure 8: Ansatz-functions for $N(R)^{\perp}$ and $N(R)$ in the 2D case

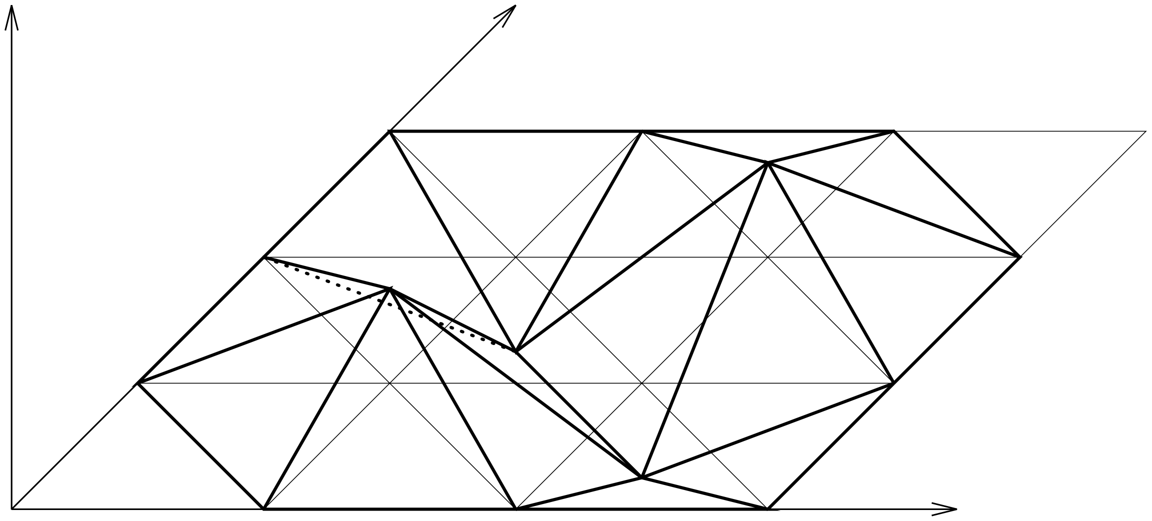


Figure 9: Ansatz-functions for $N(R)$ in the 2D case

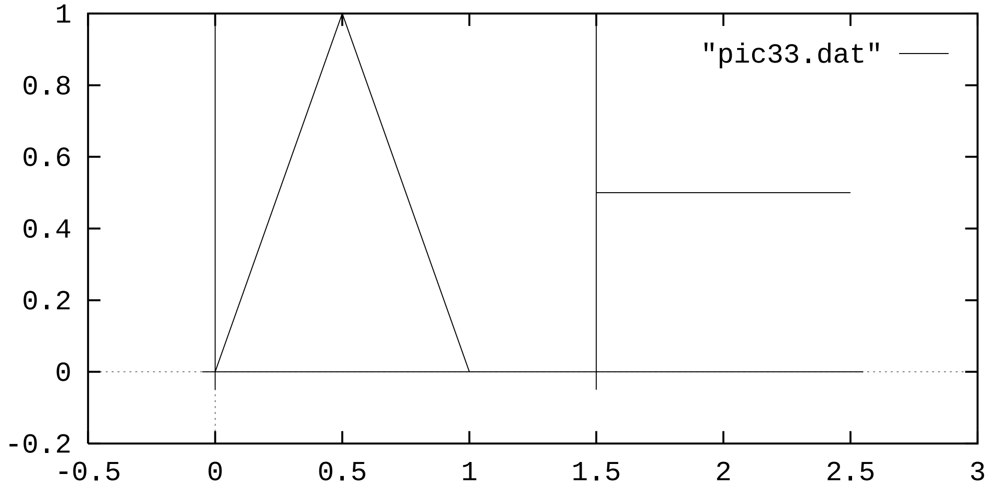


Figure 10: approx. of the Eigen-function MGM/AMGM

with the solution in $N(R)^{\perp}$ as follows:

$$\left| \begin{array}{cc} A_{11}^{-1} & 0 \\ 0 & 0 \end{array} \right| \cdot \left| \begin{array}{c} \hat{f} \\ 0 \end{array} \right| = \left| \begin{array}{c} \hat{u} \\ 0 \end{array} \right|. \tag{33}$$

This allows us, to give an explicit formulation of $A_h^{-1} - PA_HR$

$$\left| \begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array} \right|^{-1} - \left| \begin{array}{cc} A_{11}^{-1} & 0 \\ 0 & 0 \end{array} \right| =$$
$$\left[I - \left| \begin{array}{cc} I & A_{11}^{-1}A_{12} \\ 0 & 0 \end{array} \right| \right] \cdot A_h^{-1},$$

which can be a starting point for further calculations.

We should mention, that the used smoothers like Jacobi or Gauss-Seidel Iterations all fulfil the Smoothing Property, see [10] ,[26].

Another interesting point is the approximation of the coaresest Eigen-function of standard Multigrid compared with Algebraic Multigrid of this kind (see Figure 10).

Another approach using the basis transformation above, results in a orthogonal basis splitting of the space V_h into V_{h1} and V_{h2}. V_{h1} is the space of the coarse grid Ansatz-functions, V_{h2} is the $N(R)$. Because of $V_h = V_{h1} \bigoplus V_{h2}$ we get the existance of projectors $S_1 : V_h \to V_{h1}$ and $S_2 : V_h \to V_{h2}$ with $S_1 + S_2 = I$. In the following $||.||$ means the energienorm defined by the bilinearform $a(.,.)$.

$$||.||^2 = a(.,.) \tag{34}$$

We define further a pseudonorm $\lfloor||.||\rfloor : V_h \to \mathbf{R}^2$ (see [11] by

$$\lfloor||u||\rfloor := \left(\begin{array}{c} ||S_1u|| \\ ||S_2u|| \end{array} \right) \quad \forall u \in V_h. \tag{35}$$

For a bounded linear operator, the pseudonorm in analogous defined by

$$\lfloor||Q||\rfloor := \left(\begin{array}{cc} ||Q||_{11} & ||Q||_{12} \\ ||Q||_{21} & ||Q||_{22} \end{array} \right) \tag{36}$$

with

$$\|Q\|_{ij} := \sup_{v \in V_{hj}, v \neq 0} \frac{\|S_i Q v\|}{\|v\|}, \quad i,j = 1,2. \tag{37}$$

This leads us to

$$\lfloor\|Qu\|\rfloor \leq \lfloor\|Q\|\rfloor \cdot \lfloor\|u\|\rfloor, \tag{38}$$

and

$$\lfloor\|Q_1 \cdot Q_2\|\rfloor \leq \lfloor\|Q_1\|\rfloor \cdot \lfloor\|Q_2\|\rfloor. \tag{39}$$

Lemma 1 *For the spaces V_{h1} and V_{h2} in the one dimensional Dirichlet problem, the following inequality holds:*

$$|a(u,v)| \leq \frac{1}{\sqrt{2}} \|u\| \|v\| \quad \forall u \in V_{h1}, v \in V_{h2}. \tag{40}$$

The proof for this lemma is simple, but technical. We use the fact, that the derivative of the coarse grid functions, is 0 on the clustered elements. That is the reason, why we have only to integrate over halve of the elements. In two dimensions, we get the same result with the constant $\frac{\sqrt{3}}{2}$.

We further define the projector $P_j : V_h \to V_{hj}$.

$$P_j u \in V_{hj} \text{ with } a(P_j u, v) = a(u,v) \quad \forall v \in V_{hj}. \tag{41}$$

For that projector, we can proove the following lemma:

Lemma 2 *For the pseudonorm of the mapping $I - P_j$ (P_j defined above) we get*

$$\lfloor\|I - P_1\|\rfloor \leq \begin{pmatrix} 0 & 0 \\ \frac{1}{\sqrt{2}} & 1 \end{pmatrix}, \quad \lfloor\|I - P_2\|\rfloor \leq \begin{pmatrix} \frac{1}{\sqrt{2}} & 1 \\ 0 & 0 \end{pmatrix}. \tag{42}$$

For details and proof see [11].

By means of this Lemma, we get the convergence of the following theoretical algorithm in the energie norm.

Algorithm 2

$$\begin{aligned} u_h^{k+\frac{1}{2}} &:= u_h^k + v_{1,h}^k \textit{ with } v_{1,h}^k \in V_{h1} & (43) \\ a(v_{1,h}^k, v) &= (f,v) - a(u_h^k, v) \quad \forall v \in V_{h1}. & (44) \\ u_h^{k+1} &:= u_h^{k+\frac{1}{2}} + v_{2,h}^k \textit{ with } v_{2,h}^k \in V_{h2} & (45) \\ a(v_{2,h}^k, v) &= (f,v) - a(u_h^{k+\frac{1}{2}}, v) \quad \forall v \in V_{h2}. & (46) \end{aligned}$$

By the use of the above defined pseudonorm, we can define a norm equivalent to the energie norm as follows:

$$|||u||| := \mathbf{max}\{\|\mathbf{S_1 u}\|, \|\mathbf{S_2 u}\|\}. \tag{47}$$

The following theorem gives us the convergence of the above algorithm. (For details see [11])

Theorem 1 *For an arbitrary starting point $u_h^0 \in V_h$, the algorithm 2 converges to the solution $u_h \in V_h$ of the discrete problem*

$$a(u_h, v_h) = (f, v_h) \quad \forall v_h \in V_h, \tag{48}$$

and the following estimate

$$|||u_h^{k+\frac{3}{2}} - u_h||| \leq \frac{1}{2}|||u_h^k - u_h|||, \quad k = 1, 2, \ldots. \tag{49}$$

holds.

For proof see also [11].

Another different way is, to use the above results, for the coarse grid correction, and to give an estimate for the smoothing strategie, which together form the convergence for the two grid method.

Lemma 3 *The two grid method with the coarse grid correction in V_{h1} and the Richardson smoother with $\theta = \frac{1}{\lambda_{max}}$ converges to the solution $u_h \in V_h$ of the discrete problem*

$$a(u_h, v_h) = (f, v_h) \quad \forall v_h \in V_h, \tag{50}$$

and the following estimate is independent of the discretisation parameter h.

$$|||u_h^{k+\frac{3}{2}} - u_h||| \leq c \cdot |||u_h^k - u_h|||, \quad k = 1, 2, \ldots. \tag{51}$$

with $c < 1$.

Proof:

We have already prooven the following estimate

$$\lfloor ||I - P_1|| \rfloor \leq \begin{pmatrix} 0 & 0 \\ \frac{1}{\sqrt{2}} & 1 \end{pmatrix} \tag{52}$$

With the lemma of Gerschgorin we get (easy) that the pseudonorm of the iteration matrix from the Richardson method with $\theta = \frac{1}{\lambda_{max}}$ can be bounded by:

$$\lfloor ||M|| \rfloor = \lfloor ||I - \theta K_h|| \rfloor \leq \begin{pmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} \end{pmatrix} \tag{53}$$

Thus we get for pre-smoothing, coarse grid correction and post-smoothing

$$\lfloor ||u_h^{k+1} - u_h|| \rfloor \leq \begin{pmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 \\ \frac{1}{\sqrt{2}} & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{2} \end{pmatrix} \tag{54}$$

$$\cdot \qquad \lfloor ||u_h^k - u_h|| \rfloor \tag{55}$$

$$\Downarrow \tag{56}$$

$$|||u_h^{k+1} - u_h||| \leq c \cdot |||u_h^k - u_h||| \quad c < 1 \tag{57}$$

6 Performance

The algorithm is implemented as a **C++** program, and has been tested on a 486 PC. We use a CG-iteration with Algebraic Multigrid preconditioning, where the coarsening is stopped, when the number of unknowns is ≤ 40.

The next results give a comparison of 3 different problems (see Examples 4-6 below). The geometries shown in Figure 12 - 13. In the following we use the notation: $g(x) = g_i$ in $\Omega_i \subset \Omega$.

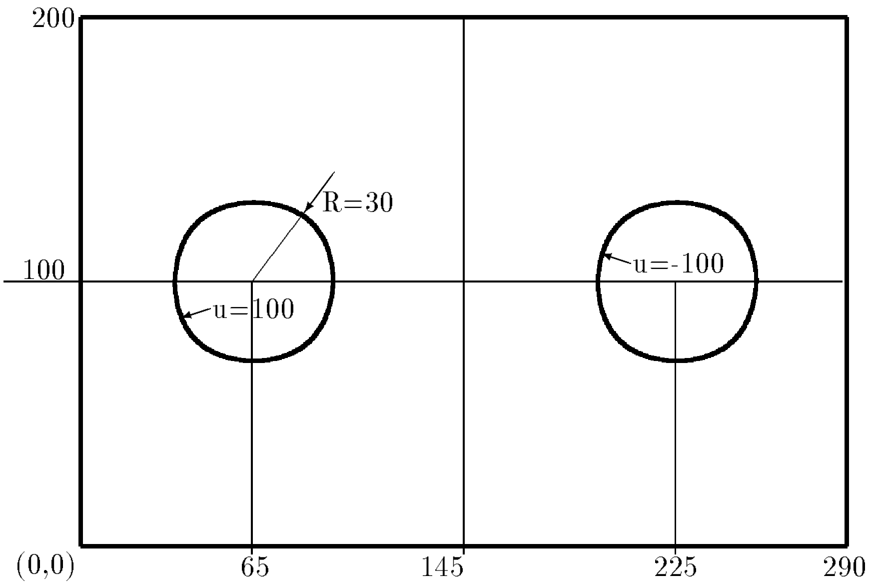


Figure 11: Example 4

Example 4 *(Wire) Find* $u \in V_g = \{v \in H^1(\Omega) | v = 0$ *on the outer boundary,* $v = 100/-100$ *at the interiour boundary* $\}$ *such that*

$$\int_\Omega \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) dx = \int_\Omega f(x) v(x) dx \quad \forall v \in V_0 = H_0^1(\Omega) \tag{58}$$

with given $f = 0$.

Example 5 *(Chip) Find* $u \in V = H^1(\Omega)$ *such that*

$$\int_\Omega \mu_i \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) dx + \int_\Gamma \alpha u v ds = \int_\Omega f(x) v(x) dx + \int_\Gamma \alpha u_A v ds \quad \forall v \in V \tag{59}$$

with Robin boundary conditions on the boundary,

$$\begin{aligned} \mu_1 = 52e-2, \mu_2 = 15e-2, \\ f_1 = 0, f_2 = 100, \\ \alpha = 0.3e4, u_A = 288. \end{aligned} \tag{60}$$

Example 6 *(Electromagnet) Find* $u \in V_0 = H_0^1(\Omega)$ *such that*

$$\int_\Omega \mu_i \mathbf{grad} u(x) \cdot \mathbf{grad} v(x) dx = \int_\Omega f(x) v(x) dx \quad \forall v \in V_0 \tag{61}$$

with homogenious Dirichlet boundary conditions, where

$$\begin{aligned} \mu_1 = 1, \mu_2 = \tfrac{2}{3}, \mu_3 = 1000, \\ f_1 = +/-1, f_2 = 0, f_3 = 0, \\ \Omega_1 \ldots \textit{ iron}, \\ \Omega_2 \ldots \textit{ air}, \\ \Omega_1 \ldots \textit{ kernel}. \end{aligned} \tag{62}$$

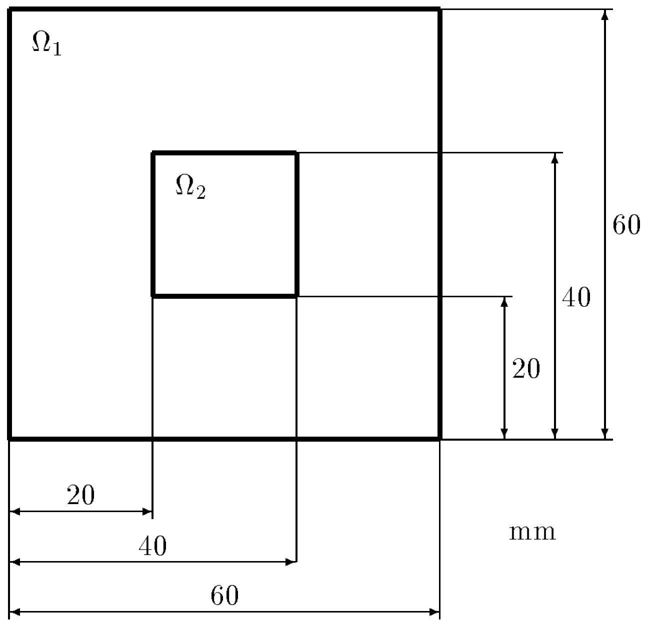


Figure 12: Example 5

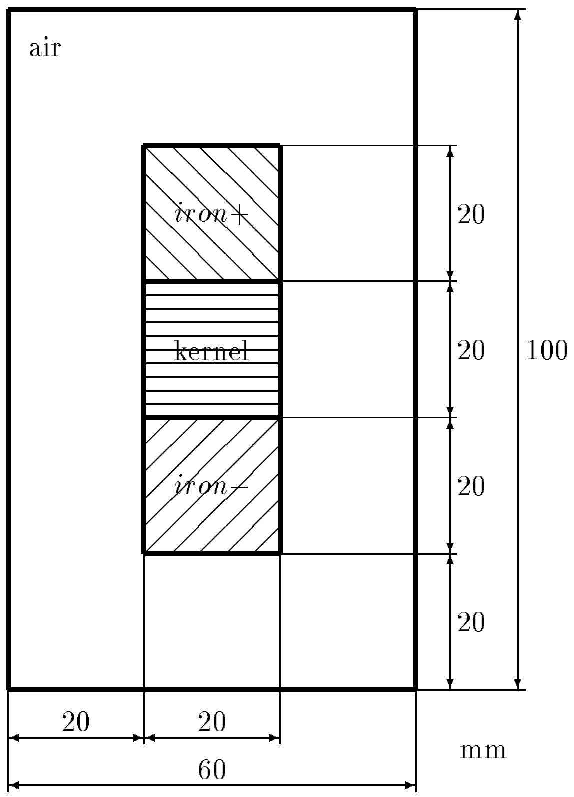


Figure 13: Example 6

The number of unknowns is about the same. For $h = .02 \sim 500$, $h = .01 \sim 2000$ and $h = .005 \sim 8000$. When using a-priori local refining on the boundary (discretisation on the boundary is 4-times finer) the number of unknowns doubles. In Example 4, there is only one material. In Example 5 and Example 6 there are 2 and 3 materials, respectively, where the coefficients differ from 4 to 1e3 (see Example 6). The number of iterations, for reducing the residuals by a factor $\epsilon = 1e - 4$, is shown in the following table:

ex3	$h = .02$	$h = .01$	$h = .005$
V-cycle(1)	8	12	15
V-cycle(.3)	10	17	25
V-cycle(.7)	9	14	21
W-cycle	5	6	8
W-cycle+LR	5	7	7
ex4	$h = .02$	$h = .01$	$h = .005$
V-cycle(1)	6	10	14
V-cycle(.3)	11	16	21
V-cycle(.7)	8	11	17
W-cycle	4	6	7
W-cycle+LR	4	5	7
ex5	$h = .02$	$h = .01$	$h = .005$
V-cycle(1)	7	10	15
V-cycle(.3)	12	16	23
V-cycle(.7)	11	12	18
W-cycle	5	6	7
W-cycle+LR	8	11	14

The parameter in V-cycle(param.) means the damping of the Gauss-Seidel update.

We have to note, that the used grids are totally unstructured, and the size of the triangels vary by a factor 16 because of the local refinement. The according meshes and solutions are shown in the Appendix.

6.1 Adaptive refinement

The next step is to look at adaptive refined grids. To get this grids, we use a *primary mesh*, and compute the solution. This mesh is rather fine, and the solution is computed with AMGPCG. With this solution, we calculate some kind of approximation-function, which contains information whether the local linear approximation is good or not. At least, we get out a refine level for each unknown to improve the approximation. This information goes back in the mesh generator, and we get out some adaptive refined grid. The known solution on the primary grid is interpolated onto this *secondary* grid, and again we make some iterations of AMGPCG. We can control this local refinement in that way, that the memory ressources of the computer are used in an optimal way.

The following tabel shows the iteration numbers for the different examples. The algorithm was first reducing the residuals by a factor 1e-8, then doing some adaptive refinemens, and reducing again the residuals by 1e-4.

ex1	14	$\sim$ 4000 unkn.	ex6	19	$\sim$ 5000
	35			25	
	20	$\sim$ 6400 unkn.		16	$\sim$ 5000
ex2	40	$\sim$ 4400 unkn.	ex7	15	$\sim$ 4400
	37			16	
	17	$\sim$ 6800 unkn.		18	$\sim$ 7600
ex3	9	$\sim$ 500 unkn.	ex8	8	$\sim$ 950
	10			9	
	14	$\sim$ 7000 unkn.		10	$\sim$ 5000
ex4	8	$\sim$ 1900 unkn.	ex9	11	$\sim$ 1100
	13			11	
	19	$\sim$ 7500 unkn.		22	$\sim$ 6600
ex5	19	$\sim$ 1800 unkn.			
	15				
	22	$\sim$ 5500 unkn.			

6.2 3D-examples

Next we tested our AMGM for the Poisson equation with Dirichlet boundary conditons

$$-\triangle u = 1 \text{ in } \Omega \text{ and } u = 273.15 \text{ on } \Gamma = \partial\Omega \tag{63}$$

in three dimensions for different geometries. For details about the geometries see [27].

All numerical experiments were carried out on a Pentium with 100 MHz and 32 MByte RAM. In the following tables the time is given in seconds.

Example 7 *Sphere*

The first test is a sphere with radius $r = 0.2$.

Sphere	h=.04	h=.02	h=.01
Meshpoints	948	6748	50683
3D elements	2225	17911	142999
2D elements	693	2772	11086
Time mesh	1	4	33
Time matrix	1	6	60
Time solver	1	6	120
Num. of It	4	4	5

Example 8 *Torus*

The next test example is a torus with outer radius $r_1 = 0.175$ and inner radius $r_2 = 0.75$. The torus and the mesh (h=0.02) are presented in [27].

Torus	h=.04	h=.02	h=.01
Meshpoints	637	4301	30771
3D elements	1266	10495	82842
2D elements	700	2831	11604
Time mesh	1	4	28
Time matrix	1	3	30
Time solver	1	4	26
Num. of It	4	4	5

Example 9 *Cone*

The third test example is a cone with radius $r = 0.15$ and height $h = 0.3$.

Cone	h=.04	h=.02	h=.01
Meshpoints	228	1615	11079
3D elements	495	4002	29736
2D elements	303	1193	4672
Time mesh	1	3	19
Time matrix	1	2	10
Time solver	1	1	10
Num. of It	4	5	5

Example 10 *Hexaedron with Torus*

Testobj. 1	h=.02	h=.014	h=.012
Meshpoints	8368	23853	39551
3D elements	22426	65277	109604
2D elements	4514	9663	39551
Time mesh	7	18	29
Time matrix	7	23	39
Time solver	8	24	38
Num. of It	4	5	5

Example 11 *Cylinder with Torus*

Testobj. 2	h=.02	h=.01
Meshpoints	3586	28360
3D elements	9488	78758
2D elements	2192	9309
Time mesh	4	27
Time matrix	3	28
Time solver	3	28
Num. of It	4	5

In the above tables, we see the optimal behaviour in time of the components of the FE-code, especially the solver routine. For different generated fine grids, the time for solving the linear system is proportional to the number of unknowns.

7 Further Goals of Developing Algebraic Multigrid Techniques

- Analysis of the Convergence:

  The next step for the introduced method must be the analysis of the solver. A similar method developed by P. Vanek is analysed in that way ([18]), that he proves a convergence rate like $\|e^{n+1}\| \leq (1 - \frac{C}{L})\|e^n\|$, where e^n denotes the error in the n -th iteration of the V11 cycle. $0 < C < 1$ is a constant independen of h and L denotes the number of levels used.

- Test and Analysis of the AMG for 4-th Order Equations:

  Like Vanek did in [18], we have to test this method also for the case of 4-th order equations, like the biharmonic equation. The analysis of the convergence should follow.

- Parallelisation of the Algebraic Multigrid

  If we use this method for three dimensional problems with non-simple geometries, the parallelisation of this algorithm is necessary to get acces to the ressources of parallel machines.

References

[1] R. Dautray J.-L. Lions: **Mathematical Analysis and numerical Methods for Science and Technology** Vol 1. Springer-Verlag Berlin, 1990.

[2] V. Girault, P.-A. Raviart: **Finite Element Approximation of the Navier-Stokes Equations** Springer-Verlag Berlin, 1979.

[3] T. Rossi: **Fictious Domain Methos**. University of Jyväskylä, 1995.

[4] A. A. Reusken: **A Multigrid Method Based on Incomplete Gaussian ELimination**. Eindhoven University of Technology, Department for Mathematics and Computer Science, RANA 95-13, 1995.

[5] J. H. Bramble, J. E. Pasciak, J. Xu: **Parallel Multilevel Preconditioners**. Mathematic of Computation, 55(191):1-22, 1990.

[6] F. Brezzi, M. Fortin: **Mixed and Hybrid Finite Elements**. Springer Verlag, 1991.

[7] J. W. Ruge, K.Stüben **Algebraic Multigrid (AMG)**. Multigrid Methods (St. Mc Cormick, ed.), Frontiers in Applied Mathematics, Vol 5, SIAM, Philadelphia 1986.

[8] P.M. de Zeeuw: **Matrix Dependent Prolongations and Restrictions in a Black Box Multigrid**. J. Comp. and Appl. Mathematics 33, 1-27 1990.

[9] F. Chatelin and W.L. Miranker: **Acceleration by Aggregation of Successive Approximation Methods**. LAA 43, 17-47 1982.

[10] W. Hackbusch **Iterative Löser großer schwachbesetzter Gleichungssysteme** Teubner Studienbücher Mathematik, 1993.

[11] Ch. Großmann H.-G. Roos: **Numerik partieller Differentialgleichungen** Teubner Studienbücher Mathematik, 1994.

[12] S. Margenov, J. Maubach: **Optimal Algebraic Multilevel Preconditioning for Local Refinement along a Line**. Numerical Linear Algebra with Application 2 (4), 347-361, 1995.

[13] B. Heise: **Parallel solvers for linear and nonlinear exterior magnetic field problems based upon FE/BE formulations.** Institutsbericht Nr. 486, Universität Linz, Institut für Mathematik, 1995.

[14] B.Heise: **Comparison of Parallel Solvers for Nonlinear Ellipic Problems Based on Domain Decomposition Ideas**. Institutsbericht Nr. 494, Universität Linz, Institut für Mathematik, 1995.

[15] B. Heise: *A Mixed Variational Formulation for 3D Magnetostatics and Its Finite Element Discretisation.* Tecnical Report 96-3, Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, 1996.

[16] J. Xu **The Auxiliary Space Method and optimal Multigrid Preconditioning Techniques for Unstructured Grids**. Computing, 1996 (to appear).

[17] P. Vanek, J. Krizkova: **Two-Level Method on Unstructured Meshes With Convergence Rate Independent of the Coarse-Space Size**. Report No. 35 University of Coorado at Denver, Center for Computational Mathematics, 1995.

[18] P. Vanek, J. Mandel , M. Brezina: **Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems** . Computing 56, 179-196, 1996.

[19] P. Vanek, J. Krizkova: **Algebraic Multigrid on Unstructured Meshes**. Report No. 34 University of Coorado at Denver, Center for Computational Mathematics, 1994.

[20] T. Greßner, A. Schneider **A 2-D Grid Editing Package**. Report no. 15, Universität Bonn, Institut für Mathematik, 1995.

[21] F. Kickinger: **Algebraic Multigrid for Elliptic Problems of Second Order**. Tecnical Report 96-2, Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerischa Mathematik und Optimierung, 1996.

[22] H. Jin and R. I. Tanner: **Generation of unstructured tetrahedral meheshes by advancing front tequnique**. International Journal of Numerical Methods in Engeneering , Vol 38 , 1995.

[23] C. Yerker, I. Zeid: **Automatic Three-Dimensional Finite Element Mesh Generation via Modified Ray Casting**. International Journal of Numerical Methods in Engeneering , Vol 31 , 1991.

[24] H. Jin and R. I. Tanner: **Unstructured Tetrahedral Mesh Generation for Three-Dimensional Viscous Flow**. International Journal of Numerical Methods in Engeneering , Vol 39 , 1996.

[25] W. Hackbusch, S. A. Sauter: **Adaptive Composite Finite Elements for the Solution of PDEs containing non-uniformly distributed Micro-Scales**. Bericht 95-2, Berichtsreihe des Mathematischen Seminar Kiel, Universität Kiel.

[26] A. Ecker and W. Zulehner:**On the Smoothing Property for the Non-Symmetric Case.** Institutsbericht Nr. 489 ,Universität Linz, Institut für Mathematik, 1995.

[27] F. Kickinger: **Automatic Mesh Generation for 3D Objects**. Tecnical Report 96-1, Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, 1996.

[28] F. Kickinger: **Algebraic Multigrid for Discrete Elliptic Second Order Problems** *a program description.* Tecnical Report 96-5, Universität Linz, Institut für Mathematik, Arbeitsgruppe Numerische Mathematik und Optimierung, 1996.