Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Diplomová práce

Problematika agresivního zhrubování v metodě multigridu pro eliptické rovnice ve 2D

Plzeň, 2013 Pavla Fraňková

University of West Bohemia
Faculty of Applied Sciences
Department of Mathematics

Diploma Thesis

Multigrid with aggressive coarsening for elliptic problems in 2D

Pilsen, 2013

Pavla Fraňková

I would like to thank my supervisor Doc. Ing. Josef Daněk Ph.D. for guiding my thesis and for his advices.

I also thank Ing. Petr Vaněk Ph.D. and RNDr. Roman Kužel Ph.D. for their kind assistance during project.

I hereby declare that this Diploma's Thesis is the result of my own work and that all external sources of information have been duly acknowledged.

.....
Pavla Fraňková

Contents

Abstract

We deal with the geometrical multigrid with aggressive coarsening and polynomial smoothing. We give a brief introduction to the finite elements method and geometric multigrid. We implement the multigrid solver in Matlab. Our main task is to prove results of [18] numerically. We analyze the main parts of the problem and try to make them more effective. We also implement the problem in Fortran 90 code, so that we could refine the mesh and compute the coarsening matrix at the same time.

Key words

Geometric multigrid, polynomial smoothers, aggressive coarsening, iterative methods

Abstrakt

Zabýváme se geometrickým multigridem s agresivním zhrubováním a polynomiálním hlazením. Podáme stručný úvod do metody konečných prvků a multigridu. V další části implementujeme multigrid v Matlabu. Analyzujeme hlavní části programu s ohledem na rychlost výpočtu. Také je provedena implementace ve Fortranu 90, která umožňuje počítat zjemňování sítě a asemblování hrubící matice najednou. Výsledky [18] byly numericky prokázány na několika příkladech.

Klíčová slova

Geometrický multigrid, polynomiální hladič, agresivní zhrubování, iterační metody

Introduction

The multigrid methods are strong tools for solving systems of linear equations. In this thesis we focus on the geometric multigrid. In the geometric multigrid we start with the coarse level and by refining the coarse mesh we obtain a system of nested meshes.
Our main interests lead to the multigrid with aggressive coarsening and polynomial smoothing. When using such a coarsening, the coarse space is much smaller than the fine space. It was theoretically proved that if we use a special polynomial smoother, the uniform convergence can be achieved. This can be done if the degree of the smoother reflects the coarsening factor. It will be our aim to verify the convergence results in [18], which say that when we use such a polynomial smoother S, the convergence is uniform under assumption $deg(S) < C\frac{h_{k+1}}{h_k}$, $C > 0$, where h_k and h_{k+1} are the characteristic resolutions for the finer resp. coarser grid.

Chapter 1

Variational Formulation of Elliptic problems

1.1 Spaces of Continuous Functions

Let $\mathbb{N}$ denote the set of non-negative integers. An n-tuple

$$\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$$

is called a multi-index. The non-negative integer $|\alpha| := \alpha_1 + \ldots \alpha_n$ is referred to as the length of the multi-index $\alpha = (\alpha_1, \ldots, \alpha_n)$. Let

$$D^\alpha = \frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1} \ldots \partial x_n^{\alpha_n}}.$$

Let Ω be an open set in $\mathbb{R}^n$ and $k \in \mathbb{N}$. We denote by $C^k(\Omega)$ the set of all continuous real - valued functions defined on Ω such that $D^\alpha u$ is continuous on Ω for all $\alpha = (\alpha_1, \ldots, \alpha_n)$ with $|\alpha| \leq k$. Assuming that Ω is a bounded set, $C^k(\bar{\Omega})$ will denote the set of all u in $C^k(\Omega)$ such that $D^\alpha u$ can be extended from Ω to a continuous function on $\bar{\Omega}$.
$C^k(\bar{\Omega})$ can be equipped with the norm

$$\|u\|_{C^k(\bar{\Omega})} := \sum_{|\alpha| \leq k} \sup_{x \in \Omega} |D^\alpha u(x)|.$$

We denote the space of continuous functions by $C(\bar{\Omega})$ with norm

$$\|u\|_C(\bar{\Omega}) = \max_{x \in \bar{\Omega}} |u(x)|.$$

The support of a continuous function of u defined on an open set $\Omega \subset \mathbb{R}^n$ is defined as the closure in Ω of the set $\{x \in \Omega : u(x) \neq 0\}$.

We denote by $C_0^k(\Omega)$ the set of all u contained in $C^k(\Omega)$ whose support is a bounded subset of Ω. Let

$$C_0^\infty(\Omega) = \bigcap_{k\geq 0} C_0^k(\Omega).$$

1.2 Spaces of integrable functions

We consider a class of spaces that consist of Lebesgue integrable functions. Let p be a real number, $p \geq 1$, we denote by $L_p(\Omega)$ the set of all real - valued functions defined on an open subset Ω of $\mathbb{R}^n$ such that

$$\int_\Omega |u(x)|^p \mathrm{d}x < \infty.$$

$L_p(\Omega)$ is equipped with the norm

$$\|u\|_{L_p(\Omega)} := \left(\int_\Omega |u(x)|^p \mathrm{d}x\right)^{1/p}.$$

An important case corresponds to taking $p = 2$. Then

$$\|u\|_{L_2(\Omega)} = \left(\int_\Omega |u(x)|^2 \mathrm{d}x\right)^{1/2}.$$

The space $L_2(\Omega)$ can be equipped with the inner product

$$(u, v) := \int_\Omega u(x)v(x)\mathrm{d}x.$$

Clearly $\|u\|_{L_2(\Omega)} = (u, u)^{1/2}$.

Lemma 1 (The Cauchy-Schwarz inequality)**.** *Let u and v belong to $L_2(\Omega)$, then $u, v \in L_1(\Omega)$ and*

$$|(u, v)| \leq \|u\|_{L_2(\Omega)}\|v\|_{L_2(\Omega)}.$$

1.3 Sobolev Spaces

Let k be a nonnegative integer and suppose that $p \in [1, \infty]$. We define (with D^α denoting a derivative of order $|\alpha|$)

$$W_p^k(\Omega) = \{u \in L_p(\Omega) : D^\alpha \in L_p(\Omega), |\alpha| \leq k.\}$$

$W_p^k(\Omega)$ is called a Sobolev space of order k- It is equiped with the (Sobolev) norm

$$\|u\|_{W_p^k(\Omega)} := \left(\sum_{|\alpha| \leq k} \|D^\alpha u\|^p_{L_p(\Omega)} \right)^{1/p} \qquad \text{when } 1 \leq p < \infty.$$

Letting

$$|u|_{W_p^k(\Omega)} := \left(\sum_{|\alpha| = k} \|D^\alpha u\|^p_{L_p(\Omega)} \right)^{1/p} \qquad \text{when } 1 \leq p < \infty,$$

for $p \in [1, \infty)$, we can write

$$|u|_{W_p^k(\Omega)} := \left(\sum_{j=0}^{k} |u|^p_{L_p(\Omega)} \right)^{1/p} \qquad \text{when } 1 \leq p < \infty.$$

An important special case corresponds to taking $p = 2$, the space $W_p^k(\Omega)$ is then a Hilbert space with inner product

$$(u, v)_{W_p^k(\Omega)} := \sum_{|\alpha| \leq k} (D^\alpha, D^\alpha).$$

We shall usually write $H^k(\Omega)$ instead of $W_p^k(\Omega)$. The definition of $W_p^k(\Omega)$ and its norm and semi norm, for $p = 2, k = 1$, give:

$$H^1(\Omega) = \{u \in L_2(\Omega) : \frac{\partial u}{\partial x_j} \in L_2(\Omega), j = 1, \ldots, n\},$$

$$\|u\|_{H^1(\Omega)} := \left(\|u\|^2_{L_p(\Omega)} + \sum_{k=1}^{n} \|\frac{\partial u}{\partial x_i}\|^2_{L_p(\Omega)} \right)^{1/2}.$$

$$|u|_{H^1(\Omega)} := \left(\sum_{k=1}^{n} \|\frac{\partial u}{\partial x_i}\|^2_{L_p(\Omega)} \right)^{1/2}.$$

$$H^2(\Omega) = \left\{ u \in L_2(\Omega) : \frac{\partial u}{\partial x_j} \in L_2(\Omega), j = 1, \ldots, n, \frac{\partial^2 u}{\partial x_i \partial x_j} \in L_2(\Omega), j = 1, \ldots, n \right\},$$

$$\|u\|_{H^2(\Omega)} := \left(\|u\|^2_{L_p(\Omega)} + \sum_{k=1}^{n} \|\frac{\partial u}{\partial x_i}\|^2_{L_p(\Omega)} + \sum_{k=1}^{n} \|\frac{\partial^2 u}{\partial x_i \partial x_j}\|^2_{L_p(\Omega)} \right)^{1/2}.$$

$$|u|_{H^2(\Omega)} := \left(\sum_{k=1}^{n} \| \frac{\partial^2 u}{\partial x_i \partial x_j} \|^2_{L_p(\Omega)} \right)^{1/2}.$$

Finally, we define the Sobolev space $H_0^1(\Omega)$ as the closure of $C_0^\infty(\Omega)$ in the norm $\|\cdot\|_{H^1(\Omega)}$. $H_0^1(\Omega)$ is the set of all $u \in H^1(\Omega)$ such that u is the limit in $H^1(\Omega)$ of sequence $\{u_m\}_{m=1}^\infty$ with $u_m \in C_0^\infty(\Omega)$. For sufficiently smooth $\partial\Omega$ the is

$$H_0^1(\Omega) = \{u \in H^1(\Omega) : u = 0 \text{ on } \partial\Omega.\}$$

1.4 Variational formulation for Poisson Equation

Theorem 1. *For sufficiently smooth functions v and w there holds*

$$\int_\Omega \nabla v \nabla w \mathrm{d}x = \int_{\partial\Omega} \frac{\partial w}{\partial n} \mathrm{d}s - \int_\Omega v \Delta w \mathrm{d}x.$$

The first integral on the right-hand side denotes integration with respect to the arc length s along $\partial\Omega$

Proof. Recall the divergence theorem:

$$\int_\Omega div A \mathrm{d}x = \int_{\partial\Omega} A n \mathrm{d}s.$$

Setting A=(vw,0) and A=(0,vw) we obtain

$$\int_\Omega \frac{\partial v}{\partial x}_i w \mathrm{d}x + \int_\Omega v \frac{\partial w}{\partial x}_i \mathrm{d}x = \int_\partial \Omega v w n_i \mathrm{d}s, \quad i = 1, 2.$$

This gives

$$\int_\Omega \nabla v \nabla w \mathrm{d}x = \int_\Omega \left(\frac{\partial v}{\partial x}_1 \frac{\partial w}{\partial x}_1 + \frac{\partial v}{\partial x}_2 \frac{\partial w}{\partial x}_2 \right) \mathrm{d}x$$

$$= \int_{\partial\Omega} \left(v \frac{\partial w}{\partial x}_1 n_1 + v \frac{\partial w}{\partial x}_2 n_2 \right) \mathrm{d}s - \int_\Omega v \left(\frac{\partial^2 w}{\partial^2 x}_1 + v \frac{\partial^2 w}{\partial^2 x}_2 \right) \mathrm{d}x$$

$$= \int_{\partial\Omega} v \frac{\partial w}{\partial n} \mathrm{d}s - \int_{\partial\Omega} v \Delta w \mathrm{d}x$$

■

For sufficiently smooth bounded domain $\Omega \subset \mathbb{R}^2$ with boundary $\partial\Omega$ we consider the homogeneous Dirichlet problem for the Poisson equation :

$$\begin{array}{rcll} -\Delta u & = & f & \text{in}\,\Omega \\ u & = & f & \text{on}\,\partial\Omega \end{array} . \tag{1.1}$$

Using Green's formula we find the variational formulation of 1.1. We define the space

$$V = \{v : v \text{ is continuous on } \Omega, \frac{\partial v}{\partial x_1}, \frac{\partial v}{\partial x_2} \text{ are p.w. cont. on } \Omega \text{ and } v = 0 \text{ on } \partial\Omega\}.$$

We multiply the Poisson equation by $v \in V$, integrate over Ω and after using the first Green formula we find u that satisfies

$$u \in V : a(u, v) = (f, v) \quad \forall v \in V \tag{1.2}$$

with

$$a(u, v) := \int_\Omega \nabla u \nabla v \mathrm{d}x \text{ and } (f, v) := \int_\Omega f v \mathrm{d}x$$

The corresponding minimization problem is:

$$u \in V : F(u) \leq F(v) \quad \forall v \in V \text{ and } F(v) := \frac{1}{2} a(v, v) - (f, v).$$

It this place, we remind some facts needed for the next section:
Let V be a linear space. $L : V \to \mathbb{R}$ is called a linear form if

$$L(\beta v + \alpha w) = \beta L(v) + \alpha L(w) \quad \forall v, w \in V, \beta, \alpha \in \mathbb{R}.$$

$a(.,.)$ is a bilinear form on $V \times V$ if $a : V \times V \to \mathbb{R}$ and if it is linear in both arguments:

$$a(u, \beta v + \alpha w) = \beta a(u, v) + \alpha a(u, w),$$
$$a(\beta v + \alpha w, u) = \beta a(u, w) + \alpha a(u, v), \quad \forall u, v, w \in V, \beta, \alpha \in \mathbb{R}.$$

The bilinear form is called symmetric if

$$a(u, w) = a(w, u) \quad \forall u, w \in V.$$

1.5 Unique Solvability of Variational Formulation

In this section we will deal with existence and uniqueness of a solution to the problem

$$u \in V : a(u, v) = L(v) \quad \forall v \in V.$$

Here V is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$, $a(\cdot, \cdot)$ is a bilinear form and $L : V \to \mathbb{R}$ is a linear form. Here we need some properties of a and L.

Definition 1. *The linear form $L : V \to \mathbb{R}$ is called continuous or bounded if*

$$\exists C > 0 : |L(v)| \leq C\|v\| \quad \forall v \in V$$

Definition 2. *The bilinear form $a : V \times V \to \mathbb{R}$ is called continuous or bounded if*

$$\exists C_a > 0 : |a(v, w)| \leq C_a\|v\|\|w\| \quad \forall v, w \in V$$

Definition 3. *The bilinear form $a : V \times V \to \mathbb{R}$ is called V-elliptic if*

$$\exists \alpha > 0 : |a(v, v)| \geq \alpha\|v\|^2 \quad \forall v \in V$$

Theorem 2 (Banach fixed point theorem)**.** *Let V be a Banach space (a complete vector space not necessarily having an inner product) and let $\phi : V \to V$ be a contraction, i.e.*

$$\exists c, \quad 0 \leq c < 1 : \quad \|\phi(v) - \phi(w)\| \leq c\|v - w\| \quad \forall v, w \in V.$$

There exists a unique $u \in V$ such that

$$\phi(u) = u.$$

Theorem 3 (Riesz representation theorem)**.** *Let V is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and norm $\|\cdot\|$. Any element $w \in V$ defines a continuoues linear form $L \in V'$ by $L := \langle w, v \rangle$. On the other hand, for any continuous linear form $L \in V'$ there exists a unique element $RL \in V$ such that*

$$L(v) = \langle RL, v \rangle \quad \forall v \in V$$

Moreover, there holds $\|RL\| = \|L\|_{V'}$, i.e.

$$\|R\|_{V' \to V} := \sup_{G \in V' \{0\}} \frac{\|RG\|}{\|G\|_{V'}} = 1$$

Theorem 4 (Lax-Milgram theorem)**.** *V is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and norm $\|\cdot\|$, $a(\cdot, \cdot)$ is continuous, V-elliptic bilinear form and $L : V \to \mathbb{R}$ is continous linear form. Then the variational problem has a unique solution $u \in V$.*

Example 1. *We now consider the two-dimensional variational formulation for Poisson problem with Dirichlet boundary conditions:*

$$a(u, v) = L(v),$$

where $u, v \in H_0^1(\Omega)$, $a(u,v) = \int_\Omega \nabla u \nabla v \mathrm{d}x$ *and* $L(v) = \int_\Omega fv\mathrm{d}x$. *Linearity and bilinearity is clear due to the linearity of the integrals provided that* $f \in L_2(\Omega)$. *To prove the boundeness of* L *we use the Cauchy-Schwarz inequallity*

$$|L(v)| = |\int_\Omega fv\mathrm{d}x| \leq \|f\|_{L_2(\Omega)}\|v\|_{L_2(\Omega)} \leq \|f\|_{L_2(\Omega)}\|v\|_{H^1(\Omega)}.$$

We used the bound $\|v\|_{L_2(\Omega)} \geq \|v\|_{H^1(\Omega)}$ *which holds by definition on the* $H^1(\Omega)$ *norm. The continuity of* a *can be shown with Cauchy-Schwarz again.*

$$|a(v,w)| \leq \|\nabla v\|_{L_2(\Omega)}\|\nabla w\|_{L_2(\Omega)} \leq \|\nabla v\|_{H^1(\Omega)}\|\nabla w\|_{H^1(\Omega)}.$$

Here we get continuity with $C_a = 1$. *We now check the* $H_0^1(\Omega)$*-ellipticity of* a. *We need to find a constant* $\alpha < 0$ *such that there holds*

$$a(v,v) = \int_\Omega |\nabla|^2 \mathrm{d}x \leq \alpha \|v\|^2_{H^1(\Omega)}.$$

We use the Poincaré's inequality

$$\int_\Omega v^2(x)\mathrm{d}x \leq C \int_\Omega |\nabla v(x)|^2 \mathrm{d}x \quad \forall v \in H_0^1(\Omega).$$

We then obtain that holds for $\alpha = \frac{1}{C+1}$.

1.5.1 Natural and Essential Boundary Conditions

We now consider a boundary problem where the normal derivative is prescribed. such a problem is called Neumann boundary problem:

$$\left\{ \begin{array}{rcll} -\Delta u & = & f & \text{in}\,\Omega \\ \frac{\partial u}{\partial n} & = & g & \text{on}\,\partial\Omega \end{array} \right. . \tag{1.3}$$

Here, $\Omega \subset \mathbb{R}^2$ is again a bounded domain with Lipschitz continuous boundary $\partial\Omega$ and $\frac{\partial u}{\partial n}$ denotes the outward normal derivative of u on $\partial\Omega$ The boundary conditions is called Neumann boundary condition. Again, we multiply the differential equation by a test function $v \in H^1(\Omega)$ and integrate over Ω. Using that $\frac{\partial u}{\partial n} = g$ on $\partial\Omega$, the first Green formula gives

$$(f,v) = \int_\Omega -\Delta u v \mathrm{d}x = -\int_{\partial\Omega} \frac{\partial u}{\partial n} v \mathrm{d}s + \int_\Omega \nabla u \nabla v \mathrm{d}x =$$

$$-(g,v)_{\partial\Omega} + (\nabla u, \nabla v) = a(u,v) - (g,v)_{\partial\Omega}.$$

The variational formulation of is

$$u \in H^1(\Omega): \quad a(u,v) = (f,v) + (g,v)_{\partial\Omega}, \quad \forall v \in H^1(\Omega),$$

where

$$a(u,v) = (\nabla u, \nabla v) \quad \text{and} \quad (g,v)_{\partial\Omega} := \int_{\partial\Omega} gv\mathrm{d}s.$$

Remark: Note that the Neumann boundary condition appears in the variational formulation and is not incorporated in the space $V = H^1(\Omega)$. It is therefore called natural boundary condition. In contrast, a Dirichlet boundary condition of the type $u =$ on $\partial\Omega$ enters the variational formulation by choosing V appropriately to reflect this condition, $V \subset H_0^1(\Omega) \subset H^1(\Omega)$ here.Therefore, Dirichlet boundary conditions are called essential boundary conditions.

1.6 Finite Element Approximation

We now introduce the finite element method for the solution of 1.2. Let us assume, for simplicity, that Ω is a polygonal domain. We then consider a triangulation $T_h = \{K_j := j = 1, \ldots, m\}$ of Ω into triangles (elements) K_j, i.e.

$$\Omega = \bigcup_{K \in T_h} K.$$

Here we assume that any two triangles are disjoint or intersect at any single vertex or an entire edge. The triangulation is called a mesh on Ω. We define

$$h_k = \text{diameter of } K = \text{length of longest side of } K,$$

$$\rho_k = \text{diameter of the longest circle in } K.$$

With such a mesh we associate a mesh size defined by

$$h = \max_{K \in T_h} diameter(K).$$

We also require

$$\frac{\rho_k}{h_k} \geq \beta$$

for $\beta > 0$ independent of h and for all $K \in \mathcal{T}_h$.

It will be assumed that we use a regular mesh: any pair of triangles in a triangulation on Ω intersect along a complete edge, at a vertex or not at all. Such meshes are shown in the figure 1

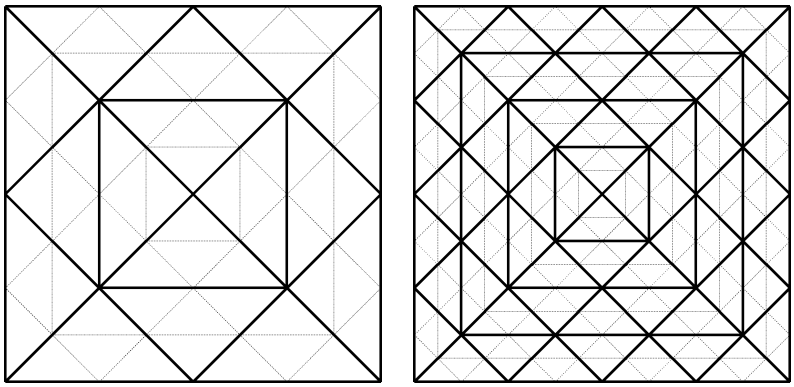

Figure 1.1: Examples of the regular triangulations

We now consider the case $V = H^1(\Omega)$ and

$$V_h = \{v \in V : v|K \in P_0(K) \forall K \in T_h\},$$

where T_h is a triangulation of Ω, which is assumed to be polygonal. Here, $P_0(\Omega)$ denotes the space of polynomials of degree 1 on K.

Our finite element space then is

$$V_h = \{v : v \text{ is continuous on } \Omega, v|_k \text{is linear for } K \in T_h, v = 0 \text{ on } \partial\Omega\}.$$

The dimension of V_h is the number M of interior nodes of the mesh T_h. We now label all the nodes of the triangulation and associate each node N_i to one piecewise linear function φ_i, which is one at the node, see figures 1.2 and 1.3

It is immediate that the functions $\varphi \in V_h$ defined by

$$\varphi(N_i) = \delta_{ij} = \begin{cases} 1 & if \quad i = j \\ 0 & if \quad i \neq j \end{cases}, \quad i, j = 1, \ldots, M$$

form a basis of V_h. We can represent any $v \in V_h$ as a linear combination of the basis functions,

$$v = \sum_{j=1}^{M} \eta_j \varphi_j \quad \text{where } \eta_j = v(N_j).$$

The finite element scheme for 1.2 reads: Find $u_h \in V_h$ such that

$$a(u_h, v) = (f, v) \quad \forall v \in V_h.$$

This is equivalent to the linear system

$$A\xi = b,$$

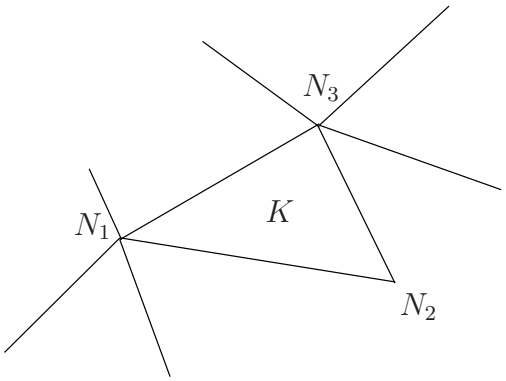

Figure 1.2: Triangulation on Ω

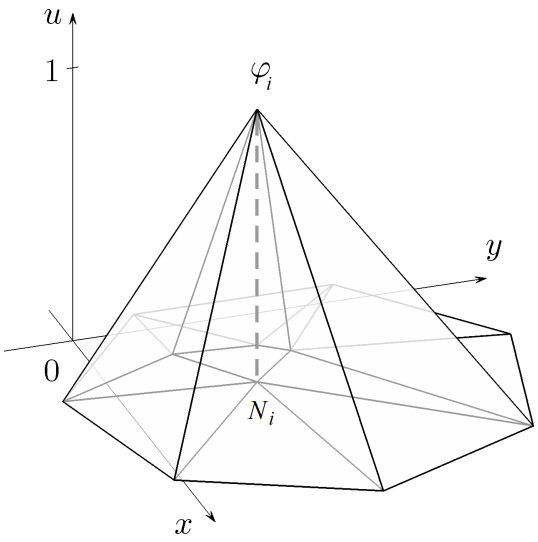

Figure 1.3: Basis function φ_i

where $A = (a_{ij}$ is the $M \times M$ stiffness matrix with elements

$$a_{ij} = a(\varphi_i, \varphi_j) = \int_\Omega \nabla\varphi_i \nabla\varphi_j \mathrm{d}x, \quad i, j, \ldots, M,$$

$b = (b_i) \in R^M$ is the load vector with

$$b_i = (f, \varphi_j) = \int_\Omega f\varphi_i \nabla \mathrm{d}x, \quad i, \ldots, M,$$

and the solution vector $\xi = (\xi_i) \in R^M$ satisfies $\xi_i = u_h(M_i)$, $i = 1, \ldots, M$.

1.6.1 Interpretation of the Galerkin Orthogonality

Now suppose that $V_h$is a finite dimensional subspace of H_0^1. As before, the finite approximation of

$$\text{find } u \in H_0^1(\Omega) \text{ such that } a(u, v) = L(v) \quad \forall v \in H_0^1(\Omega).$$

is:
find u_h in V_h such that $a(u_h, v_h) = L(v_h)$ for all $v_h \in V_h$.

Since it holds for any $v \in V_h$, it also holds for v_h. We therefore have

$$a(u, v_h) = L(v_h) \quad \forall v \in V_h.$$

Subtracting these two equations we get the Galerkin orthogonality

$$a(u - u_h, v_h) = 0 \quad \forall v_h \in V_h.$$

We now continue and denote $v = u - u_h \in H_0^1$. From one of Lax-Milgram assumption (V-ellipticity of a) we have that

$$\|u - u_h\|^2_{H^1(\Omega)} \leq \frac{1}{\alpha} a(u - u_h, u - u_h),$$

$$\|u - u_h\|^2_{H^1(\Omega)} \leq \frac{1}{\alpha} a(u - u_h, u - u_h),$$

From Galerkin orthogonality we have $a(u, v_h) = a(u, u_h)$ $\forall v_h \in V_h$ and it follows that

$$\|u - u_h\|^2_{H^1(\Omega)} \leq \frac{1}{\alpha} a(u - u_h, u - u_v).$$

The boundedness of a gives

$$a(u - u_h, u - u_v) \leq C_a \|u - u_h\|_{H^1(\Omega)} \|u - u_h\|_{H^1(\Omega)}.$$

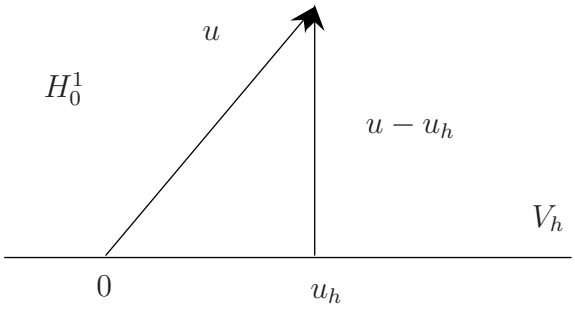

Figure 1.4: Interpretation of the Galerkin orthogonality

Combining the last two inequalities, we obtain that

$$\|u - u_h\|_{H^1(\Omega)} \leq \frac{C_a}{\alpha} \|u - v_h\|_{H^1(\Omega)} \forall v_h \in V_h.$$

This result is known as Céa's lemma:

Lemma 2 (Céa's lemma)**.** *The finite element approximation u_h to $u \in H_0^1(\Omega)$, the weak solution to the problem , is the nearest-best fit to u in the norm $|\cdot\|_{H^1(\Omega)}$:*

$$\|\|u - u_h\|_{H^1(\Omega)} \leq \frac{C_a}{\alpha} \min_{v_h \in V_h} \|u - v_h\|_{H^1(\Omega)}\|.$$

The orthogonality of error $u - u_h$ to the space h is shown in the picture 1.4

1.6.2 Matrix assembly

To efficiently organize this computation one uses an element-oriented strategy. Using the decomposition $\Omega = \bigcup_{K \in T_h} K$ we find for any $i, j, \in \{1, \ldots, M\}$ that

$$a(\varphi_i, \varphi_j) = \int_\Omega \nabla\varphi_i \nabla\varphi_j \mathrm{d}x = \sum_{K \in T_h} \int_K \nabla\varphi_i \nabla\varphi_j \mathrm{d}x = \sum_{K \in T_h} a_K(\varphi_i, \varphi_j).$$

There holds $a_K(\varphi_i, \varphi_j) = 0$ unless both nodes N_i and N_j are vertices of the triangle K. Therefore, to calculate $a_K(\varphi_i, \varphi_j)$, one only needs to consider the numbers

$i, j \in \{1, \ldots, M\}$ which correspond to nodes N_i, N_j of K. For arbitrary fixed $K \in T_h$ let N_i, N_j, N_k denote three vertices. We then call the 3×3-matrix

$$A_k := \begin{pmatrix} a_K(\varphi_i, \varphi_i) & a_K(\varphi_i, \varphi_j) & a_K(\varphi_i, \varphi_k) \\ & a_K(\varphi_j, \varphi_k) & a_K(\varphi_j, \varphi_j) \\ sym & & a_K(\varphi_k, \varphi_k) \end{pmatrix}$$

the local stiffness matrix for K. A is sometimes called global stiffness matrix to distinguish it from the local stiffness matrices. To calculate A_k one obviously needs only the restrictions of the basis functions $\varphi_i, \varphi_j, \varphi_k$. Let us denote these restrictions by

$$\psi_i := \varphi_i | K,\ \psi_j := \varphi_j | K,\ \psi_k := \varphi_k | K.$$

Each of these functions is linear on K and has the value at exactly one vertex and vanishes a the other two vertices. Any linear function w on Kcan be represented by

$$w = w(N_i)\psi_i + w(N_j)\psi_j + w(N_k)\psi_k.$$

The functions ψ_i, ψ_j, ψ_k are called local stiffness basis functions on K.

1.7 Finite element error estimate

Theorem 5. *For a Lipschitz domain $\Omega \subset \mathbb{R}^2$ with polygonal boundary and a given integer $T \geq 2$ let $\{T : T \in \mathcal{T}_h\}$ be a shape regular triangulation of Ω. Then for a piecewise polynomial interpolation operator I_h of degree $t-1$ (piecewise with respect to $\mathcal{T}_h$) there holds*

$$\left(\sum_{T \in \mathcal{T}_h} \|u - I_h u\|^2_{H^m(T)} \right)^{1/2} \leq C h^{t-m} |u|_{H^t}(\Omega)$$

for all $u \in H^t(\Omega)$ an all $0 \leq m \leq t$.

Let $\Omega \subset \mathbb{R}^2$ be a polygon with a quasi-uniform, regular and shape-regular mesh, and I_h be a piecewise linear interpolation operator with respect to the vertices of the mesh. Then

$$\|u - I_h u\|^2_{H^1(\Omega)} = \sum_{T \in \mathcal{T}_h} \|u - I_h u\|^2_{H^2(T)} \leq C h^2 |u|^2_{H^2(\Omega)}.$$

We therefore obtain

$$\|u - I_h u\|^2_{H^1(\Omega)} \leq C h^2 |u|^2_{H^2(\Omega)} \quad \forall u \in H^2(\Omega).$$

Chapter 2

Multgrid Methods

In the next chapter, we give an introduction to the multigrid methods. The crucial part of the multigrid methods is the so called smoothing process. The iterative methods like Jacobi or Gauss - Seidel are the most classic examples if smoothers and we start with them.

2.1 Symmetric and positive definite matrices

A $n \times n$ square matrix $A = (a_{ij})$ is called symmetric if

$$v^T A w = w^T A v$$

for any two vectors v and w. It is equivalent to $a_{ij} = a_{ji}$.

A square matrix A is called positive definite if

$$v^T A v > 0$$

for any non-zero vector v.

For symmteric matrix the following values

$$\min_v \frac{v^T A v}{v^T v} \text{ and } \max_v \frac{v^T A v}{v^T v}$$

characterize the minimal and maximal eigenvalues of A.

Definition 4 (symmetric definition of matrix norm)**.** *For any $n \times m$ rectangular matrix B, the symmetric expression*

$$\max_{v \in \mathbb{R}^n, w \in \mathbb{R}^m} \frac{w^T B v}{\|v\| \|w\|},$$

defines a matrix norm $\|B\|$.

The definition is equivalent to

$$\|B\| = \max_{v \in \mathbb{R}^n} \frac{\|Bv\|}{\|v\|}$$

Proposition 1. *Let A and B be teo s.p.d matrices. Then the inequality*

$$v^T AV \leq v^T VBv \quad \textit{for all } v,$$

implies that

$$v^T B^{-1} V \leq v^T A^{-1} v \quad \textit{for all } v.$$

2.2 Iterative Methods

Let $\mathcal{V}$ be a finite dimensional vector space. We will study iterative methods to solve a linear system

$$Au = f,$$

where $A : \mathcal{V} \to \mathcal{V}$ is a symmetric positive definite linear operator and $f \in \mathcal{V}$ is given.

Let M be an $n \times n$ matrix such that the systems in the form of $My = g$ are easy to solve. We consider the decomposition

$$A = D + L + L^T,$$

where L is strictly lower triangular part of A. For a given M and initial guess x^0 we consider an iterative process

$$M(x^{k+1} - x^k) = f - Ax^k \tag{2.1}$$

for $k = 0, 1, \ldots$.

The term x^{k+1} is called the correction and the right hand-side $r^k = Ax^k$ is called residual.

Algorithm 1 Calculate u^{k+1}

---

given :
$M, \quad x^0 \in \mathcal{V}$
find :

$$x^{k+1} = x^k + M^{-1}(f - Ax^k), \quad k = 0, 1, 2, \ldots. \tag{2.2}$$

Lets assume a sequence of iterations

$$x^k, \quad k = 0, \ldots, n. \tag{2.3}$$

We denote the error $e^k = x^* - x^k$, where x^* is the exact solution of . It holds

$$M(x^{k+1} - x^k) = M(x^{k+1} - x + x - x^k) = f - Ax^k = Ae^k,$$

$$e^{k+1} = (I - M^{-1}A)e^k.$$

Another form of (2.1) can be written such that

$$x^{k+1} = Ex^k + N, \quad k = 0, 1, 2, \ldots, \tag{2.4}$$

where

$$E = I - M^{-1}A, \quad N = M^{-1}f.$$

The matrix E is called the iteration matrix. We recursively obtain

$$x^{k+1} - x^* = E(x^k - x^*) = \cdots = E^{k+1}(x^0 - x^*).$$

We see that the error $e = x^{k+1} - x^*$ is transformed by the matrix E. We therefore say that E is also the error propagation operator.
A complex scalar λ is called an eigenvalue of the square matrix A if a nonzero vector u exists such that $Au = \lambda u$. The vector u is called an eigenvector of A associated with λ. The set of all the eigenvalues of A is called the spectrum of A and is denoted by $\sigma(A)$.

The maximum modulus of the eigenvalues is called the spectral radius and is denoted by $\rho(A)$.

$$\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda|.$$

Theorem 6. *Let E be a square matrix such that $\rho(E) < 1$. Then $I - E$ is non-singular and the iteration (2.3) converges for any f and x^0. Conversely, if the iteration (2.3) converges for any f and x^0, then $\rho(E) < 1$.*

Idea of the proof: Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of E, then there exists a regular matrix T such that

$$E = TJT^{-1},$$

where J is Jordan matrix . It holds

$$E^k = TJ^kT^{-1}.$$

If

$$\max_i |\lambda_i(E)| < 1,$$

then

$$\lim_{k \to \infty} J^k = 0,$$

therefore

$$e^k = E^k e^0 \to 0.$$

In the opposite direction, if $\lim_{k\to\infty} E^k = 0$, then $\lim_{k\to\infty} J^k = 0$ and

$$|\lambda_i| < 1 \quad \forall i.$$

Corollary 1. *Let E be a square matrix such that $||E|| < 1$ for some matrix norm $||.||$. Then $I - E$ is nonsingular and the iteration (2.2) converges for any initial vector x^0.*

In the next part, we are interested in the convergence of the stationary method in energy norm $\|v\|_A = \sqrt{v^T A v}$. For symmetric, positive definite matrix A we define s.p.d matrix $A^{\frac{1}{2}}$. Then it holds:

$$\|v\|_A^2 = v^T A v = v^T A^{\frac{1}{2}} A^{\frac{1}{2}} v = (A^{\frac{1}{2}} v) A^{\frac{1}{2}} v = \|A^{\frac{1}{2}} v\|^2.$$

We use $e^{k+1} = (I - M^{-1}A)e^k$ we have

$$A^{\frac{1}{2}} e^{k+1} = A^{\frac{1}{2}} (I - M^{-1}A) A^{-\frac{1}{2}} (A^{\frac{1}{2}} e^k).$$

Applying norm on both sides, we have

$$\|e^{k+1}\|_A \leq \|I - A^{\frac{1}{2}} M^{-1} A^{\frac{1}{2}}\|_A \|e^k\|_A.$$

We need to estimate $\|I - A^{\frac{1}{2}} M^{-1} A^{\frac{1}{2}}\|_A$.

We denote the expression $\epsilon := \|I - A^{\frac{1}{2}} M^{-1} A^{\frac{1}{2}}\|_A$. We then consider $\epsilon^T \epsilon$:

$$\epsilon^T \epsilon = (I - A^{\frac{1}{2}} M^{-T} A^{\frac{1}{2}})(I - A^{\frac{1}{2}} M^{-T} A^{\frac{1}{2}}),$$

$$= I - A^{\frac{1}{2}} M^{-T} A^{\frac{1}{2}} - A^{\frac{1}{2}} M^{-1} A^{\frac{1}{2}} + A^{\frac{1}{2}} M^{-T} A M^{-1} A^{\frac{1}{2}},$$

$$= I - (A^{\frac{1}{2}} M^{-T})(M + M^T - A)(M^{-1} A^{\frac{1}{2}}).$$

We denote $Y = M^{-1} A^{\frac{1}{2}}$, then

$$I - Y^T (M + M^T - A) Y.$$

As matrix Y is invertible, $Y^T(M + M^T - A)Y$ is s.p.d. if a and only if $M + M^T - A$ is s.p.d.

Theorem 7 (16)**.** *A necessary and sufficient condition for the iteration process to be A- convergent (convergent in A-norm) is*

$$M + M^T - A$$

to be s.p.d.

We now consider a composite iteration using both M and M^T. For given $x^0, k \geq 0$ compute

$$M(x^{k+\frac{1}{2}} - x^k) = b = Ax^k,$$
$$M^T(x^{k+1} - x^{k+\frac{1}{2}}) = b = Ax^{k+\frac{1}{2}}.$$

We obtain

$$\bar{M}(x^{k+1} - x_k) = b = Ax^k,$$

where

$$\bar{M} = M(M + M^T - A)M^T.$$

We then have

$$x^{k+\frac{1}{2}} = x^k + M^{-1}r^k,$$
$$x^{k+1} = x^{k+\frac{1}{2}} + M^{-T}r^{k+\frac{1}{2}}.$$

Then,

$$x^{k+1} = x_k + (M^{-1} + M^{-T} - M^{-T}AM^{-1})r^k = x^k + M^{-T}(M + M^T - A)M^{-1}r^k.$$

This is standard iteration process with the symmetric preconditioner $\bar{M}$,

$$M(x^{k+1} - x^k) = f - Ax^k.$$

We have

$$I - \bar{M}^{-1}A = (I - M^{-T}A)(I - M^{-1}A).$$

Hence if we multiply equation with $A^{\frac{1}{2}}$ from the left and $A^{-\frac{1}{2}}$ from the right. We then obtain

$$A^{\frac{1}{2}}(I - \bar{M}^{-1}A)A^{-\frac{1}{2}} = A^{\frac{1}{2}}(I - M^{-T}A)(I - M^{-1}A)A^{-\frac{1}{2}} = \epsilon^T\epsilon.$$

Here, the composite iteration is composite if and only if the original iteration with M is A-convergent.

2.2.1 Richardson Iterative Method

The first method we describe is the Richardson method. The matrix B from algorithm (2.2) is given by

$$B = \frac{\omega}{\rho(A)}I,$$

therefore

$$u^{k+1} = u^k + \frac{\omega}{\rho(A)}(f - Au^k), \quad k = 0, 1, 2, \ldots.$$

Let

$$A\phi_i = \nu_i\phi_i,$$

with ν_i the eigenvalues of A,

$$\nu_1 < \nu_2 < \ldots \nu_n,$$

ϕ_i the eigenfunctions of A, Then

$$u - u^0 = \sum_i \alpha_i \phi_i$$

and

$$u - u^k = \sum_i \alpha_i (1 - \omega \frac{\nu_i}{\nu_n})^k \phi_i.$$

For a fixed $\omega = 1$, it is clear that if $(1 - \omega\nu_i/\nu_n)^k$ converges to zero as $k \to \infty$, it converges fast for ν_i close to ν_n. This means that the high frequency modes in the error are damped out quickly.

The graphs of solution after some smoothing steps are in the picture 2.1

2.2.2 Jacobi, Gauss-Seidel and SOR Methods

Let $\mathcal{V} = \mathbb{R}^N$ and let $A \in \mathbb{R}^{N \times N}$ now be a symmetric positive definite matrix. We then consider a splitting

$$A = (D - L - U)$$

where D is diagonal and L and U are the strictly lower and upper parts of A. Then

$$\begin{aligned} Au &= f, \\ \Leftrightarrow (D - L - U)u &= f, \\ \Leftrightarrow Du &= (L + U)u + f, \\ \Leftrightarrow u &= D^{-1}(L + U)u + D^{-1}f. \end{aligned}$$

The Jacobi iteration reads

$$u^{k+1} = D^{-1}(L + U)u^k + D^{-1}f$$

or

$$u^{k+1} = u^k - D^{-1}(Au^k - f)$$

The Gauss-Seidel method in matrix form is obtained in the following way. Using the decomposition

$$\begin{aligned} A &= D - L - U, \\ (D - L)u &= Uu + f \end{aligned}$$

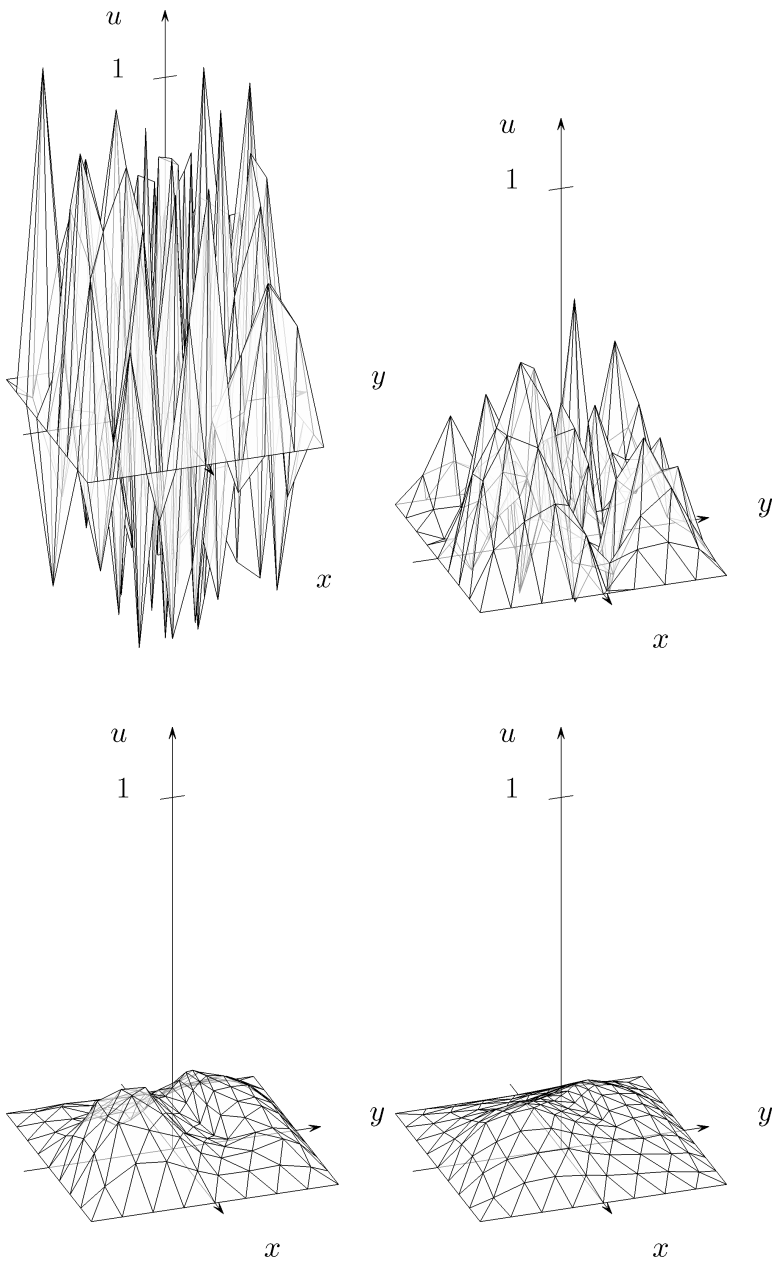

Figure 2.1: Smoothing effect of iterative methods

and the iterates are

$$u^{k+1} = u^k - (D - L)^{-1}(Au^k - f).$$

Another approach is to solve each equation for u_i and use it to update u_{k+1}. The Gauss-Seidel is then of the form

$$u_i^k = (b_i - \sum_{j=1}^{i-1} a_{ij}u_j^k - \sum_{j=i+1}^{n} a_{ij}u_j^{k-1})/a_{ii}.$$

The successive over relaxation (SOR) comes from the Gauss-Seidel method,

$$x_i^{k+1} = x_i^k + r_i^k$$

where

$$r_i^k = (b_i - \sum_{j=1}^{i-1} a_{ij}u_j^k - \sum_{j=i+1}^{n} a_{ij}u_j^{k-1})/a_{ii}.$$

To make an improvement, we use ωr_i^k,

$$x_i^k = x_i^k + \omega r_i^k,$$

we get a linear combination of $k - th$ iteration from a Gauss-Seidel method and previous $k - 1th$ itereration,

$$x_i^k = \omega(b_i - \sum_{j=1}^{i-1} a_{ij}u_j^k - \sum_{j=i+1}^{n} a_{ij}u_j^{k-1})/a_{ii} + (1 - \omega)x_i^k.$$

2.3 Introduction to Multigrid

In this section we give a brief introduction to the multigrid methods. A nice introduction can be found in [4] and in [15]. A bit more general theory can be then found in [1]. Many relaxation schemes damp the oscillatory modes of the error effectively, but smooth modes are damped slowly. If we use a coarse grid, relaxation is cheaper (1/2 of nodes in 1D, 1/4 in 2D) . The second observation might be the residual correction idea: Let u^i be the approximation to u, the residual is $d = f - Au$ and the error $v = u - u^i$ satisfies $Av = d$.

The error will be smoother after relaxing on the fine grid, but more oscillatory on the coarse grid. We can therefore relax the residual equation $Av = d$ with initial guess of error $e = 0$.

Let us denote the fine grid by Ω^h and the coarse grid by Ω^H , where $H = 2h$.

We define a linear mapping $p : \mathbb{R}^m \to \mathbb{R}^n$, $m < n$. The mapping p is called a prolongator and it's range includes such error modes, which cannot be effectively eliminated with smoothing. We find $v \in \mathbb{R}^m$ such that v minimizes error correction $\|e - pv\|_A$. The idea is outlined in the algorithm 3: The minimization of $\|e - pv\|_A$ in A norm leads to the two level algorithm:

$$\frac{\mathrm{d}}{\mathrm{d}t}\|e - p(v + wt)\|_A^2 = 0 \quad \forall e \in \mathbb{R}^m,$$

$$\|e - p(v + wt)\|_A^2 = \|e - pv\|_A^2 + 2t(A(e - pv), pw) + t^2\|pw\|_A^2,$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\|e - p(v + wt)\|_A^2 = 2(A(e - pv), pw).$$

The minimization $\|e - pv\|_A$ fulfills

$$(p^T A(e - pv), pw) = 0 \quad \forall w \in \mathbb{R}^m.$$

The condition for coarse grid correction is therefore

$$p^T A(e - pv) = 0,$$

which is

$$p^T Apv = p^T Ae.$$

Since

$$e = e(x) = x - x^*, \quad x^* = A^{-1}f,$$

The correction on the coarse level is given by solving the equation

$$p^T Apvp^T(Ax - f). \tag{2.5}$$

The expression (2.5) gives the standard two-level algorithm 2. We will now use notation I_H^h for interpolation and I_h^H for restriction.

The notation $u^h = \mathcal{S}^{\nu_2}(A_h, u^h, f^h)$ means that u^h results form a smoothing method after ν smoothing steps for the initial guess u_0, the right hand side f and iteration matrix A.

We have to define mappings from coarse to fine grids. Let us take the 1-D case. We define a mapping

$$I_{2h}^h : \Omega^{2h} \to \Omega^h.$$

If u^h, u^{2h} are defined on Ω_h, Ω_{2h}, then

$$I_{2h}^h u^{2h} = u^h,$$

where I_{2h}^h is an interpolation if

$$\begin{cases} u_{2i}^h = u_i^{2h}, \\ u_{2i+1}^h = (u_i^{2h} + u_{i+1}^{2h})/2, \end{cases} \quad \text{for} \quad 0 \le i \le \frac{N+1}{2}.$$

Algorithm 2 Two-grid method

relax on fine mesh

$u^h = \mathcal{S}^{\nu_1}(A_h, u_0^h, f^h) \qquad on \quad \Omega^h$

compute residuum

$d^h = f^h - A_h u^h$

restrict residuum

$d^H = I_h^H d^h$

relax on the coarse mesh to get the error on the coarse mesh

$A_H v_H = d^H \quad on \quad \Omega^H$

correct the approximation of fine mesh

$u^h = u^h + I_H^h v^H$

post smooth

$u^h = \mathcal{S}^{\nu_2}(A_h, u^h, f^h) \qquad on \quad \Omega^h$

In the matrix form, this reads

$$u^h = \frac{1}{2}\begin{bmatrix} 1 & & & \\ 2 & & & \\ 1 & 1 & & \\ & 2 & & \\ & 1 & 1 & \\ & & \vdots & \\ & & 1 & \\ & & 2 & \\ & & 1 & \end{bmatrix} u^{2h}$$

The restriction operator can be then taken as

$$I_h^{2h} u^h = u^{2h},$$

which is also called the injection operator.

We may also consider a full weighting operator, which is in 1-D case

$$u_i^{2h} = \frac{1}{4}(u_{2i-1}^h + 2u_{2i}^h + u_{2i+1}^h).$$

In matrix form, this reads

$$u^{2h} = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & 1 & 2 & 1 & & \dots \\ & & & & & & 1 & 2 & 1 \end{bmatrix} u^h.$$

We could look at the algorithm as at the iterative process with error propagation operator M_h.

Since we are interested just in M_h, we take $f^h = 0$ and S^{ν_1}, S^{ν_2} corresponding smoothing operators. We now follow the two-level algorithm steps to get

$$u^h \leftarrow S^{\nu_1}(u^h),$$

the residual

$$d^H = I_h^H(-A_h S_h^{\nu_1}(u^h)).$$

The algorithm becomes

$$u^h \leftarrow S^{\nu_2}[S^{\nu_1}(u^h) + I_H^h(I_h^H A_h I_H^h)^{-1}(-A_h S_h^{\nu_1}(u^h))],$$

therefore

$$M^h = S^{\nu_2}[I - I_H^h(I_h^H A_h I_H^h)^{-1} A_h] S_h^{\nu_1}.$$

The matrix inside the brackets,

$$T_H^h = I - I_H^h(I_h^H A_h I_H^h)^{-1} A_h, \tag{2.6}$$

is known as the coarse grid correction. We now remind some facts about projections. Let V be a vector space. We say that linear mapping $P : V \to V$ is a projection, if $P = P \circ P$.

Theorem 8. *Let V be a vector space and $P : V \to V$ projection. Then*

$$V = RangeP \oplus KerP.$$

Proof in [19].

Lemma 3. *When the coarse grid matrix is defined as $A_H = I_h^H A_h I_H^h$, then the coarse grid operator (2.6) is a projector with respect to the A_h - inner product.*

We follow the important proof from in [19]. **Proof.** Let us show that $I - T_h^H = I_H^h A_H^{-1} I_h^H A_h$ is a projector. At first, we check if $P^2 = P$:

$$(I_H^h(I_h^H A_h I_H^h)^{-1} A_h) \cdot (I_H^h(I_h^H A_h I_H^h)^{-1} A_h) =$$

$$I_H^h A_H^{-1}(I_H^h A_H^{-1} I_h^H) A_h^{-1} I_h^H A_h = I_H^h A_H^{-1} I_h^H A_h = I_H^h(I_h^H A_h I_H^h)^{-1} A_h.$$

Next we show that T_h^H is self adjoint. The adjoint of $I_H^h A_H^{-1} I_h^H A_h$ in a A_h-inner product is

$$(T_h^H x, y)_{A_h} = (I_H^h A_H^{-1} I_h^H A_h x, y)_{A_h} = (x, I_H^h A_H^{-1} I_h^H A_h y)_{A_h} = (x, T_h^H y)_{A_h}.$$

T_h^H is therefore self-adjoint and $I_H^h\ A_H^{-1} I_h^H A_h$ and A an A-orthogonal projector. We now show that $I_H^h A_H^{-1} I_h^H A_h$ is a A-orthogonal projection to the range(I_h^H) and T_H^h is A-orthogonal to the A-orthogonal complement to Range(P).
We know that our algorithm is based on the minimization of $\|e - I_H^h v\|_A$. We therefore have

$$\|e - I_H^h A_H^{-1} I_h^H A_h e\|_A = \min_{w \in \text{Range U}} \|e - w\|_A.$$

$U = I_H^h A_H^{-1} I_h^H A_h$ is a A-orthogonal projection to the Range$P$:

$$\{x \in \mathbb{R}^n : (Ax, w) = 0 \quad \forall w \in \text{Range}(I_H^h)\}.$$

Next step is

$$\begin{gathered}\{x \in \mathbb{R}^n : (Ax, w) = 0 \quad \forall w \in \text{Range}(I_H^h)\} = \\ \{x \in \mathbb{R}^n : (Ax, I_H^h v) = 0 \quad \forall v \in \mathbb{R}^m\} = \\ \{x \in \mathbb{R}^n : I_H^h A x = 0\} \\ = \text{ Ker } (I_k^H A).\end{gathered}$$

■

We now look at very important property [19]. Let us consider M^h in the form

$$M^h = S^1[I - I_H^h (I_h^H A_h I_H^h)^{-1} A_h].$$

Here S^1 is Richardson iteration, therefore we denote S by $I - \frac{\omega}{\rho(A)} A$.

We would like to find an upper bound of M^h.

$$\frac{\|ST_H^h e\|_A}{\|e\|_A} = \frac{\|ST_H^h e\|_A}{\|T_H^h e\|_A} \cdot \frac{\|T_H^h e\|_A}{\|e\|_A},$$

$$\sup \frac{\|ST_H^h e\|_A}{\|e\|_A} = \sup \frac{\|ST_H^h e\|_A}{\|T_H^h e\|_A} \cdot \frac{\|T_H^h e\|_A}{\|e\|_A}.$$

By the submultiplicativity of the norm $\|\cdot\|_A$ we obtain

$$\sup \frac{\|ST_H^h e\|_A}{\|e\|_A} \leq \sup \frac{\|ST_H^h e\|_A}{\|T_H^h e\|_A} \cdot \sup \frac{\|T_H^h e\|_A}{\|e\|_A}.$$

The spectral radius of A-orthogonal projection is bounded by 1:

$$\sup \frac{\|ST_H^h e\|_A}{\|e\|_A} \leq \sup \frac{\|ST_H^h e\|_A}{\|T_H^h e\|_A} = \sup_{u \in \text{Rng}_{T_H^h}} \frac{\|Su\|_A}{\|u\|_A}.$$

We now consider

$$\|Su\|_A^2 = \|(I - \frac{\omega}{\rho(A)}A)u\|_A^2 =$$

$$\langle Au, u\rangle - \frac{\omega}{\rho(A)}\langle A^2u, u\rangle - \frac{\omega}{\rho(A)}\langle Au, Au\rangle + \frac{\omega^2}{\rho^2(A)}\langle AAu, Au\rangle,$$

$$\|u\|_A^2 - \frac{2\omega}{\rho(A)}\|Au\|^2 + \frac{\omega^2}{\rho^2(A)}\langle AAu, Au\rangle =$$

$$\|u\|_A^2 - (2-\omega)\frac{\omega}{\rho(A)}\|Au\|^2.$$

Under an assumption $\frac{\|Au\|^2}{\|e\|_A^2} > C\rho(A)$ and taking $\omega = 1$ we have

$$\|u\|_A^2(1 - (2-\omega)\frac{\omega}{\rho(A)}\frac{\|Au\|^2}{\|u\|_A}) \leq \|u\|_A^2(1 - \frac{1}{\rho(A)}C\rho(A)) = (1-C)\|u\|_A^2.$$

We have proved that $(Au, I_H^h) = 0\ \forall v \in \mathbb{R}^m$.

$$\|u\|_A^2 = (Au, u) = (Au, u - I_H^h v) \leq \|Au\|\|u - I_H^h v\|\ \forall v \in \mathbb{R}^m.$$

We assume so called weak approximation property

$$\forall u \in \mathbb{R}^n \exists v \in \mathbb{R}^m : \|u - I_H^h v\| \leq \frac{C}{\sqrt{\rho(A)}}\|u\|_A.$$

2.3.1 Nested Iteration

A nested iteration can be used to obtain a good initial guess from coarser meshes.

Algorithm 3 Nested iteration

set $h := h_0$. Given an initial guess $u_0^h, setu^h = \mathcal{S}^{\nu_1}(A_h, u_0^h, f^h)$
for $l = m - 1$ to 0 **do**
  $u^{h/2} = I_h^{h/2}u^h$
  $h = h/2$
  $u_j^h = \mathcal{S}^{\nu_1}(A_h, u_0^h, f^h)$
end for

2.4 The Multigid Cycle

The multigrid cycle is based on a two - grid cycle. In the two-grid method, we needed to find a solution of

$$A_h v^H = d^H$$

exactly. We observe, that we don't have to solve the coarse level defect equation exactly, since

$$\bar{v} = I_H^h v^H$$

is also an approximation. We may replace v^H by a suitable approximation. We can apply the two grid idea again with even coarser grid. This process can be repeated until all $l+1$ levels are involved. Level 0 corresponds to the coarsest level.

Algorithm 4 Multi-grid method for solving $A_l u^l = f^l, u_k^{m+1} = MGM(k, \gamma, u_k^m, A_k, f_k, \nu_1, \nu_2)$

---

presmoothing
compute $\bar{u}_k^m$ by applying ν_1 smoothing steps to u_k^m.
$\bar{u}_k^m = \mathcal{S}^{\nu_1}(A_k, u_k^m, f^k)$
coarse grid corrections
compute residuum
$\bar{d}_k^m = f_k - A_k \bar{u}_k^m$
restrict residuum
$\bar{d}_{k-1}^m = I_k^{k-1} \bar{d}_k^m$
compute the approximate solution
$A_{k-1} \bar{v}_{k-1}^m = \bar{d}_{k-1}^m$
if $k = 1$ use a direct or fast iterative solver to solve $\bar{d}_{k-1}^m = I_k^{k-1} \bar{d}_k^m$
if $k > 1$,
$\bar{v}_{k-1}^m = MGM(k-1, \gamma, 0, A_{k-1}, \bar{d}_{k-1}^m, \nu_1, \nu_2)$
interpolate the correction
$\bar{v}_k^m = I_{k-1}^k \bar{v}_{k-1}^m$
correct
$u_k^m = \bar{u}_k^m + \bar{v}_k^m$
postsmoothing
$u_k^{m+1} = \mathcal{S}^{\nu_1}(A_k, u_k^m, f^k)$

$$M_0 = 0$$

$$M_k = S_k^{\nu_2}(I_k - I_{k-1}^k (I_{k-1} - (M_{k-1})^\gamma)(A_{k-1})^{-1} I_{k-1}^k L_k) S_k^{\nu_1}$$

$$k = 1, \ldots, l$$

The difference between the two-grid operation

$$M_k^{k-1} = S_k^{\nu_2}(I_k - I_{k-1}^k A_{k-1}^{-1} I_k^{k-1} A_k) S_k^{\nu_1}$$

and the above iteration operator M_k is that A_{k-1}^{-1} is replaced by

$$(I_{k-1} - (M_{k-1})^\gamma)(A_{k-1})^{-1}.$$

2.4.1 Full Multigrid (FGM)

As previously we consider a sequence of discrete approximations and use the notation

$$MGM^r(k+1, \gamma, \omega_k, A_k, f_k, \nu_1, \nu_2)$$

for a procedure consisting of r steps of a suitable iterative (k+1)-grid cycle with cycle index γ. The right hand sides f_k on Ω can be defined recursively by

$$f_k = I_{k+1}^k f_{k+1}$$

with

$$f_l = f|_{\Omega_l}.$$

The aim is to achieve a discretization accuracy on each level within a few multigrid cycles.

In the multigrid cycle, the operator I_{k-1}^k was applied to the corrections, but here we have FMG interpolation $\mathcal{I}_{k-1}^k$,

$$\mathcal{I}_{k-1}^k : \quad \Omega_{k-1} \to \Omega_k,$$

which transfers the approximation of the solution to the fine grid.

Algorithm 5 Full Multigrid

for $k = 0$ solve
$A_0 u_0 = f_0$
$u_0^{FMG} = u_0$
for $k = 1, 2, \ldots, j$
$u_0^k = \mathcal{I}_{k-1}^k u_{k-1}^{FMG}$ $u_k^{FMG} = MGM^r(k+1, \gamma, \omega_k, A_k, f_k, \nu_1, \nu_2)$

2.4.2 Work Estimates

More on this topic in [6]. We now assume MGM algorithm, which consist of these parts:

$$\begin{array}{ccll} u_l = \mathcal{S}(u_l, f_l) & \leq C_S n_l & \text{for all } l \geq 1 \\ d_{l-1} = I_{l-1}^l (A_l u_l - f_l) & \leq C_D n_l & \text{for all } l \geq 1 \\ u_l = u_l - I_l^{l+1} v_{l-1} & \leq C_C n_l & \text{for all } l \geq 1 \\ u_0 = A_0^{-1} f_0 & \leq C_0 & \end{array} \tag{2.7}$$

The n_l is the number of unknowns and equations at level l. We assume A_l to be sparse matrices and bounds proportional to n_l. We now use some results from [6]. The constant

$$C_H = \sup \frac{n_{l-1}}{n_l}$$

has the value

$$C_H = 2^{-d},$$

where d is the dimension of $\mathbb{R}^d$ and mesh size is $h_{l-1} = 2h_l$. The following holds:

Suppose that

$$\gamma C_H < 1.$$

Then one step of the multi-grid iteration requires $C_l n_l$ operations, where

$$C_l < \frac{\nu C_s - C_D + C_C}{1 - \gamma C_H} + (\gamma C_H)^{l-1} \frac{C_0}{n_1}.$$

Generally we have W_l computational work per level l. It is recursively given

$$W_1 = W_1^0 + W_0$$

$$W_{k+1} = W_{k+1}^k + \gamma_k W_k, \quad k = 1, \ldots, l-1$$

The W_{k+1}^k denotes computational work of two-grid cycles except work needed to solve the defect equations on Ω_k.

If γ is independent of k, then

$$W_l = \sum_{k=1}^{l} \gamma^{l-k} W_k^{k-1} + \gamma^{l-1} W_0, \quad l \geq 1.$$

In the 2D case

$$N_k \cong 4N_{k-1}.$$

We then require

$$W_k^{k-1} \leq C N_k.$$

Under these assumptions we get estimate of total computing work W_l

$$W_l \leq \frac{4}{3} C N_l \quad \text{for} \quad \gamma = 1$$
$$W_l \leq 2 C N_l \quad \text{for} \quad \gamma = 2$$
$$W_l \leq 4 C N_l \quad \text{for} \quad \gamma = 3$$

2.4.3 Geometric Multigrid on FEM Spaces

In this section we may refer to the theory in [3] and [2]. Let us now consider a model problem for $\Omega \subset \mathbb{R}^2$,

$$-\Delta u = f, \quad \text{in} \quad \Omega, \tag{2.8}$$
$$u = 0 \quad \text{on} \quad \partial\Omega.$$

The weak formulation of the problem can be formulated into a form

$$a(u, v) = l(v), \quad \forall v \in H_0^1(\Omega),$$
$$a(u, v) = \int_\Omega \nabla u(x) \nabla(v)(x) dx,$$
$$l(v) = \int_\Omega u(x) f(x) dx$$

and

$$(v, w) = \int_\Omega vw \, dxdy.$$

Let us consider a hierarchy of nested finite element spaces

$$V_0 \subset V_1 \cdots \subset V_L$$

where each space is spanned by

$$V_l = span\{\varphi_i^{(l)}\}_{i=1}^{n_l}.$$

The notation $\varphi_{i_c}^{(l)}$ means $i_c - th$ basis function on the level l. Since each $\varphi_{i_c}^{(l)} \in V \subset V_L$ we have the expansion

$$\varphi_{i_c}^{(l)} = \sum_{i=1}^{n} \varphi_{i_c}^{(l)}(x_i) \varphi_i^{(l-1)}.$$

Consider the coefficient vector $\phi_{i_c} = (\varphi_{i_c}^l(x_i))_{i=1}^n$. The matrix $P_{l-1}^l = (\phi_{i_c})_{i_c=1}^{n_l}$ is referred to as the coarsening matrix. The relations among basis function is shown in the picture 2.4.3. We will talk about this topic again in the implementation part.

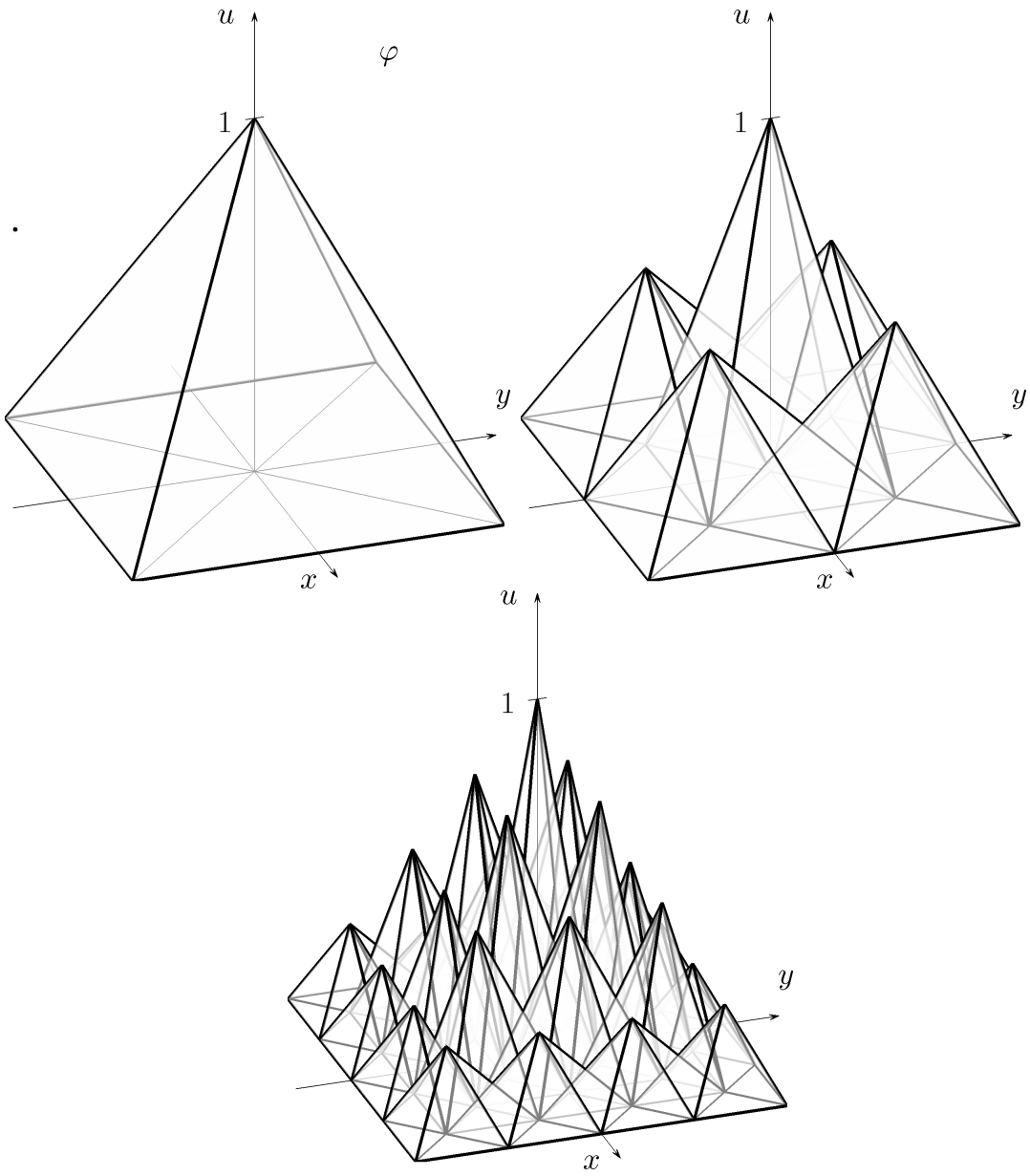

Figure 2.2: The basis function on the coarse level (top left), which can be expanded by the basis function on the finer levels

2.5 Multigrid with Polynomial Smoothers

In this section, let us consider a standard V-cycle multigrid to solve

$$Ax = f,$$

with A s.p.d. matrix of order n. We assume there is a hierarchy of prolongators $\{P_{k-1}^k\}$,

$$P_{k-1}^k : \mathbb{R}_{k-1}^n \to \mathbb{R}^k, \quad n_{k-1} < n_k,\ k = 0, \ldots, L-1$$

and matrices A_k defined in the section 2.3.4 .

Let us consider a polynomial smoother with error propagation operator

$$I - M_k^{-T} A_k = I - M_k^{-1} A_k = S_k^\gamma \left(I - \frac{1}{\bar{\lambda}_{S_k^2 A_k}} S_k^2 A_k \right), \tag{2.9}$$

where S_k is a polynomial in A_k such that $\sigma(S_k) \leq 1$, $\bar{\lambda}_{S_k^2 A_k} \geq \sigma(S_k^2 A_k)$ and γ positive integer (we will assume γ =1). The error propagation operator to the symmetrized smoother $\bar{M}_k$ is

$$I - \bar{M}_k^{-1} A_= (I - M_k^{-T} A_k)(I - M_k^{-1} A_k) = S^{2\gamma} \left(I - \frac{1}{\bar{\lambda}_{S_k^2 A_k}} S_k^2 \right)^2.$$

The symmetrized smoother M_k is then

$$\bar{M}_k^{-1} = A_k^{-1} \left[I - \left(I - \frac{1}{\bar{\lambda}_{S_k^2 A_k}} S_k^2 \right)^2 S_k^{2\gamma} \right].$$

We need a multigrid relaxation on the level k as an iterative process with error propagation operator $I - M_k^{-1} A_k$. As we have proved in the beginning of chapter 2. The process is A_k convergent if $M_k^T + M_k - A_k$ is positive definite matrix. We assume that there is a constant $\alpha > 0$, uniform with respect to $k \geq$ such that,

$$v_k^T (M_k^T + M_l - A_k) v_k \geq \alpha v_k^T A_k v_k \quad \forall v_\in \mathbb{R}^{n_k}. \tag{2.10}$$

Now, we recursively define the V-cycle preconditioner B_k (s.p.d matrix) in the following way:

for $k = 1, \ldots, L$

$$I - B_k^{-1} A_k = (I - M_k^{-T} A_k)(I - P_{k-1}^k B_{k-1}^{-1} (P_{k-1}^k)^T A_k)(I - M_k^{-1} A_k).$$

We start with $B_0 = A_0$ and let $B = B_L$

Lemma 4. *For any $\lambda > 0$ and integer $N > 0$ there is a unique polynomial $p_{\lambda,N}$ of degree N such that*

$$\max_{0 \le t < \lambda} p_{\lambda,N}^2(t) t$$

is minimal under the constraint $p_{\lambda,N}(0) = 1$. The polynomial p is given by

$$p_{\lambda,N}(t) = \left(1 - \frac{t}{r_1}\right) \dots \left(1 - \frac{t}{r_N}\right),$$

$$r_k = \frac{\lambda}{2}\left(1 - \cos\left(\frac{2k\pi}{2N+1}\right)\right), \quad k = 1 \dots, N.$$

Let $\bar{\lambda_k}$ be an upper bound of $\rho(A_k)$ and integer N_k a given degree of a smoothing polynomial. We choose

$$p_{\bar{\lambda} N_k} A_k.$$

We also set

$$\bar{\lambda}_{S_k^2 A_k} = \frac{\bar{\lambda}_k}{(2N+1)^2}.$$

We use following algorithm to implement action 2.9 . Next, we perform the linear part S_k: We do for $i = 1, \dots, N_k = deg(S_k)$,

$$x \leftarrow (I - \alpha_i A_k)x + \alpha_i f, \quad \alpha_i = \left[\frac{\bar{\lambda}_k}{2}(1 - \cos\left(\frac{2i\pi}{2N_k+1}\right))\right]^{-1}, \quad \bar{\lambda}_k \le \rho(A_k).$$

We then perform the iteration with the error propagation operator

$$I - \bar{\lambda}^{-1}_{S_k^2 A_k} S_k^2 A_k,$$

we then do

$$x \leftarrow x - \frac{1}{\bar{\lambda}_{S_k^2 A_k}} S_k^2 (Ax - f),$$

where we evaluate the action of $S_k x$ as a product

$$S_k x = (I - \alpha A_k) \dots (I - \alpha_{N_k} A_k) x.$$

Let us define the quantity $\lambda_{k,j} = \sup x \in \mathbb{R}^{n_k} \frac{\langle A P_k^0 x, P_0^k x \rangle}{\|P_j^k x\|^2}$, $k < j$

Theorem 9. *Let $\bar{\lambda}_{k-1,k} \ge \lambda_{k-1,k}$, we assume the existence of linear mappings $Q_k : V_k \to V_L, Q_L = I$, staisfying*

$$\|(Q_k - Q_{k+1})v\|_k \le \frac{C_a}{\sqrt{\lambda_{k-1,k}}} \|v\|_A \quad \forall v \in V_L, \quad k = L, \dots, 1 \tag{2.11}$$

and

$$\|Q_k\|_A \leq C_s, \quad k = 0, \ldots, L.$$

We assume smoothers satisfy 2.10 and $\bar{M}_k$ *satisfy*

$$\|v\|^2_{\bar{M}_k} \leq \beta(\bar{\lambda}_{k-1,k})\|v\|^2 + \|v\|^2_{A_k}.$$

Then the multigrid operator B *is nearly spectrally equivalent to* A*:*

$$v^T Av \leq v^T Bv \leq \left[C_s^2 + 2L\left(\beta(C_a + 4C_s^2) + \frac{1}{\alpha}C_s^2\right)\right] v^T Av.$$

2.6 Multigrid with smoothed prolongator

Previously, we were interested in the minimization of the $\|e - I_H^h v\|_A$. We are now interested in the minimization of the error after the correction and the smoothing [17]. We want to find such v, that

$$\|s(e - I_H^h v)\|_A \quad \text{is minimal.}$$

We use the very similar manipulation like in the section about classic two-level method. Again, we deal with a quadratic functional $\|S(e - I_H^h)\|_A^2$, so we find its minimum by zero variation:

$$\frac{\mathrm{d}}{\mathrm{d}t}|_{t=0}\|S(e - I_H^h(v + tw))\|_A = 0 \quad \forall w \in \mathbb{R}^m.$$

The minimization condition reads

$$(AS(e - I_H^h v)Spw) = 0 \quad \forall w \in \mathbb{R}^m,$$

hence

$$I_h^H S^T AS I_H^h v = I_H^h S^T ASe.$$

The error propagation operator is given by

$$Ee = S(e - I_H^h v) = [I - SI_H^h((SI_H^h)^T ASI_H^h)^{-1}(SI_H^h)^T A]Se.$$

We now compare it with the previous

$$M^h = S^{\nu_2}[I - I_H^h(I_h^H A_h I_H^h)^{-1} A_h]S_h^{\nu_1}.$$

We have the same method with one pre-smoothing step $\nu_1 = 1$ and the prolongator SI_H^h. We then have a following algorithm SMG:

We consider a polynomial smoothers

$$S = (I - \frac{\omega_1}{\bar{\lambda}}A)(I - \frac{\omega_2}{\bar{\lambda}}A)\ldots(I - \frac{\omega_N}{\bar{\lambda}}A), \ldots \omega_i \in \mathbb{R}_+,\ i = 1, \ldots N.$$

Here $N = deg(S)$. Let $\bar{\lambda} \geq \rho(A)$ and $\bar{\lambda}_S \geq \rho(S^2 A) = \frac{\bar{\lambda}}{(2N+1)^2}$. We consider a following algorithm

Algorithm 6 Two-grid method with smoothed prolongator - SMG

relax on fine mesh

$u^h = \mathcal{S}^1(A_h, u_0^h, f^h) \qquad on \quad \Omega^h$

compute residuum

$d^h = f^h - A_h u^h$

restrict residuum

$d^H = (SI_H^h)^T d^h$

relax on the coarse mesh to get the error on the coarse mesh

$(SI_H^h)^T A_h (SI_H^h) v_H = d^H \quad on \quad \Omega^H$

correct the approximation of fine mesh

$u^h = u^h + (SI_H^h) v^H$

Algorithm 7 Two-grid method with smoothed prolongator -SMG II

for given parameters $N = deg(S)$, $\omega_i \in \mathbb{R}_+$, $i = 1, \dots N$:

for i=1,...,N do

$x \leftarrow (I - \frac{\omega_i}{\overline{\lambda}} A)x + \frac{\omega_i}{\overline{\lambda}} f$

solve

$(SI_H^h)^T A_h (SI_H^h) v_H = d^H \quad on \quad \Omega^H$

correct the approximation of fine mesh

$u^h = u^h + (SI_H^h) v^H$

do

$x \leftarrow (I - \frac{\omega}{\overline{\lambda}_S} S^2 A)x + \frac{\omega}{\overline{\lambda}} S^2 f$

Chapter 3

Implementation and Numerical Experiments

In this chapter, we discuss the numerical implementation of the previous algorithms. We describe our programs and present various numerical experiments. Our main aim is to test the uniform convergence of the algorithm SMG II We also test some aspects of the algorithm changing degree of the polynomial smoother or taking non-polynomial smoothers. We concentrate on Poisson problem on various meshes, boundary conditions and right hand side.

3.1 Implementation

We first discuss the main issues of the multigrid algorithm. The computation consists of preprocessing part and solving part. During the preprocessing part we set up the problem and finest mesh, assembly the matrix and load vector. We then assembly coarsening matrices and compute Galerkin matrices. As soon as we are finished with the preprocessing, we compute the solution in the multigrid cycle.

3.1.1 Error Estimates and Stopping Criterion

Let us consider very simple example

$$-\Delta u = 1$$

with Dirichlet boundary conditions $u = 0$ on Γ, where Γ is the boundary of the square $[0,1] \times [0,1]$.

We would like to measure $\|u - u_h\|_A$, convergence rates and computational times. To investigate the energy norm, we transform it into another form:

$$a(u - u_h, u - u_h) = a(u,u) + a(u_h, u_h) - 2a(u, u_h)$$

$$= a(u,u) + a(u_h,u_h) - a2(u_h,u_h)$$
$$= a(u,u) - a(u_h,u_h)$$

i.e. we have

$$\|u - u_h\|_A^2 = \|u\|_A^2 - \|u_h\|_A^2.$$

The norm $\|u_h\|_A$ is easy to compute, since it is

$$\|u_h\|_a^2 = b^T z,$$

where b is a load vector and z is vector of unknown coefficients.

The norm $\|u\|_A^2$ is usually not known, but we can make a very good guess by Aitken's extrapolation technique: We compute u_h for a sequence of meshes for $i = 1, \ldots, n$. Denote $u_h^i = \gamma_i$, we then have

$$\|u\|_A^2 \approx \frac{\gamma_i\gamma_{i-2} - \gamma_{i-1}^2}{\gamma_i - 2\gamma_{i-1} + \gamma_{i-2}}, \quad i = 3, \ldots, n.$$

In the following computations, we use several types of errors. We work with the exact solution u, vector b, computed solution u_h and, int the case of computational error, we distinguish between computed solution of discrete problem v_h a exact solution of the discretization problem u_h.

ralative error

$$= \frac{\sqrt{(u-u_h)^{\mathrm{T}}(A)(u-u_h)}}{\sqrt{u^{\mathrm{T}}Au}},$$

absolute error

$$= \sqrt{(u-u_h)^{\mathrm{T}}A(u-u_h)},$$

relative residual error

$$= \frac{\sqrt{(b-Au_h)^{\mathrm{T}}(A)(b-Au_h)}}{\sqrt{b^{\mathrm{T}}Ab}},$$

computational error

$$= \frac{\sqrt{(u_h-v_h)^{\mathrm{T}}(A)(u_h-v_h)}}{||u||_A - \sqrt{u_h^{\mathrm{T}}Au_h}}.$$

In our experiments, we are interested in the rate of convergence μ and the order of convergence α.

We say that the sequence has the rate of convergence μ if

$$\lim_{k\to\infty} \frac{e_{k+1}}{e_k^\alpha} = \mu, \quad \mu > 0.$$

We assume

$$e_{n+1} = C|e_n|^{\alpha},$$

then

$$\alpha = \log(|e_{n+1}/e_n|)/\log(|e_n/e_{n-1}|).$$

We also use the convergence rate

$$\mu = e_{NoI}^{1/NoI},$$

where e_{NoI} is relative residual error after NoI iterations.

3.2 Matlab Programs

Our programs has been implemented with the Matlab R2006a software. We use some of the Matlab procedures, especially the `pdetoolbox`.
The main computations are done in the script `multigrid_script` and the function `multigrid_cycle`. To run the multigrid solver, we run the script `multigrid_script`. There are 5 predefined problems. As we need to assembly the stiffness matrix and the load vecctor, we need to get parameters to be put into Matlab pde routines.

We first discuss the `assembpde` command. To be able to assembly the system, we need three matrices p, e and t, coeffitients of the elliptic problem and information about geometry of the problem. The last three number coming to the procedure are (1,0,1) and corresponds to the fact that we solve the Poisson equation. Now we get the geometry and matrices p, e and t. We can use the `initmesh` command or we can use the `pdetool` and export all the necessary coefficients into the data files. In our implementation we consider the both cases.

For our first example we initialize mesh, assembly the system and solve the system of equations in Matlab built-in procedures:

```
[p,e,t]=initmesh('squareg','hmax',2)
for i =1:10
[K,W,B,ud]=assempde('squareb1',p,e,t,1,0,1);
[p,e,t]=refinemesh('squareg',p,e,t);
u1=K\W;
u(i)=W'*u1
end
```

We see that we need to carry the information about boundary nodes. We have two options available. If we solve a Dirichlet problem, one way is to solve the system of n_i equations, where n_i is the number of interior nodes. In this case, we

have to compute some mapping between numbering of the global basis functions and numbering of the interior nodes. Second approach is to include boundary nodes into the matrix, such that the numbering of the global basis functions is the same as the numbering of the nodes. The stiffness matrix is larger, but since there is no need for extra mapping, it will be more comfortable for us to work with such a matrix. Another reason could be that the matrices for Dirichlet and Neumann problem would have the same size, so our algorithms for creating the coarsening matrix could be almost the same.

The last thing before we do some experiments is to make an upper guess of the eigenvalue of A necessary in the algorithm. To do so, we use the Gershgorin's theorem:

Theorem 10. *Every eigenvalue of the matrix A of size N satisfies:*

$$|\lambda - A_{ii}| \leq \sum_{j \neq i} |A_{ij}| \quad i \in \{1, 2, 3, \ldots, N\}$$

In matlab weuse the command `lambda=norm(A,'inf')` .

From [18] we also have take care of the degree of the smoother. The degree must satisfy

$$deg(S) < C\frac{h_{k+1}}{h_k}, \quad C > 0.$$

3.2.1 example

We consider the simple example

$$-\Delta u = f$$

with Dirichlet boundary conditions $u = 0$ on Γ, where Γ is the boundary of the unit circle. The solution u is chosen such that it solves the exation with $f = -12xy$. The exact solution is therefore

$$u(x, y) = xy(1 - x^2 - y^2).$$

We have predefined our problem as *problem* 4, with initial triangulation shown in the picture 3.2.1.

The exact solution is shown in the picture 3.3.

As this is the first example, we also want to talk about matrices involved. The matrices are generally sparse, as we can wee in the picture 3.2. The picture 3.2 shows the nonzero elements of the stiffness matrix. The coarsening matrix is also sparse. The coarsening matrix is created according the section 2.4.3:

The basis on the coarse level is expanded by the basis on the fine level. Consider the coefficients of fine basis functions, which expands the coarse one. It is

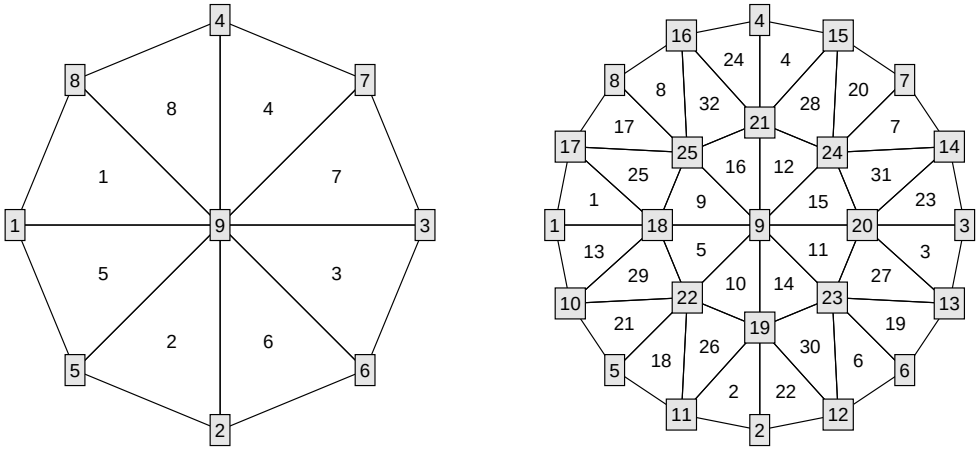

Figure 3.1: Coarsest triangulation of the unit circle on the left and a refinement of the triangulation on the right.

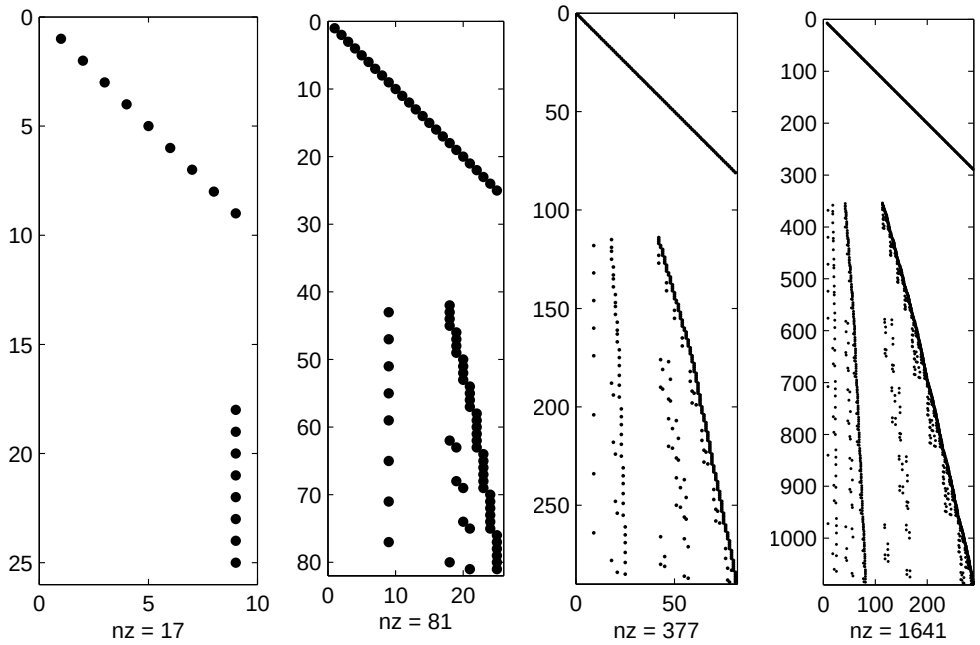

Figure 3.2: Nonzero entires of the stiffness matrix A. The coarsest matrix is on the left and matrix on on the level after three refinements on the right.

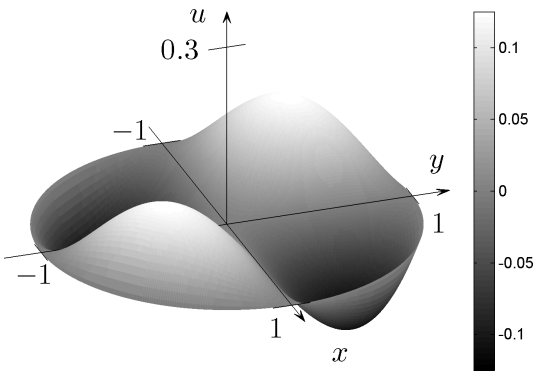

Figure 3.3: Graph of the exact solution u.

obvious that each column of the coarsening matrix consists of the vector of these coefficients. In our case (p_1 polynomials) it is easy to estimate the coefficients, since they can be either 0, 1 or 1/2. Lets now look at the the triangulations for this example.

Matlab `initmesh` labels the edge nodes first. After one refinemesh step the old nodes have the same number and new nodes are created. We use these information to create the coarsening matrices.

The following algorithm is used to compute the coarsening matrix. Vector $w2$ keeps the information about the node - if it is boundary node (0) or not (1). We know that ones will be on the diagonal, as the coarse nodes are also on the fine mesh. Then we go over the coarse elements. The coarse nodes restricted to the the fine mesh have the neighbors, which has the coefficient one half in the coarsening matrix. As we create sparse matrices, we do need an input in the coordinate format. As we go over every interior edge twice, Matlab sums the value, if it finds the coordinate entry twice. We therefore set the value as 1/4.

We can now proceed to the multigrid cycle. If we do not solve the equation where the solution is known, we use the Matlab backslash command to compute a solution, which we take as an exact solution. The first three iterations are shown in the picture 3.4.

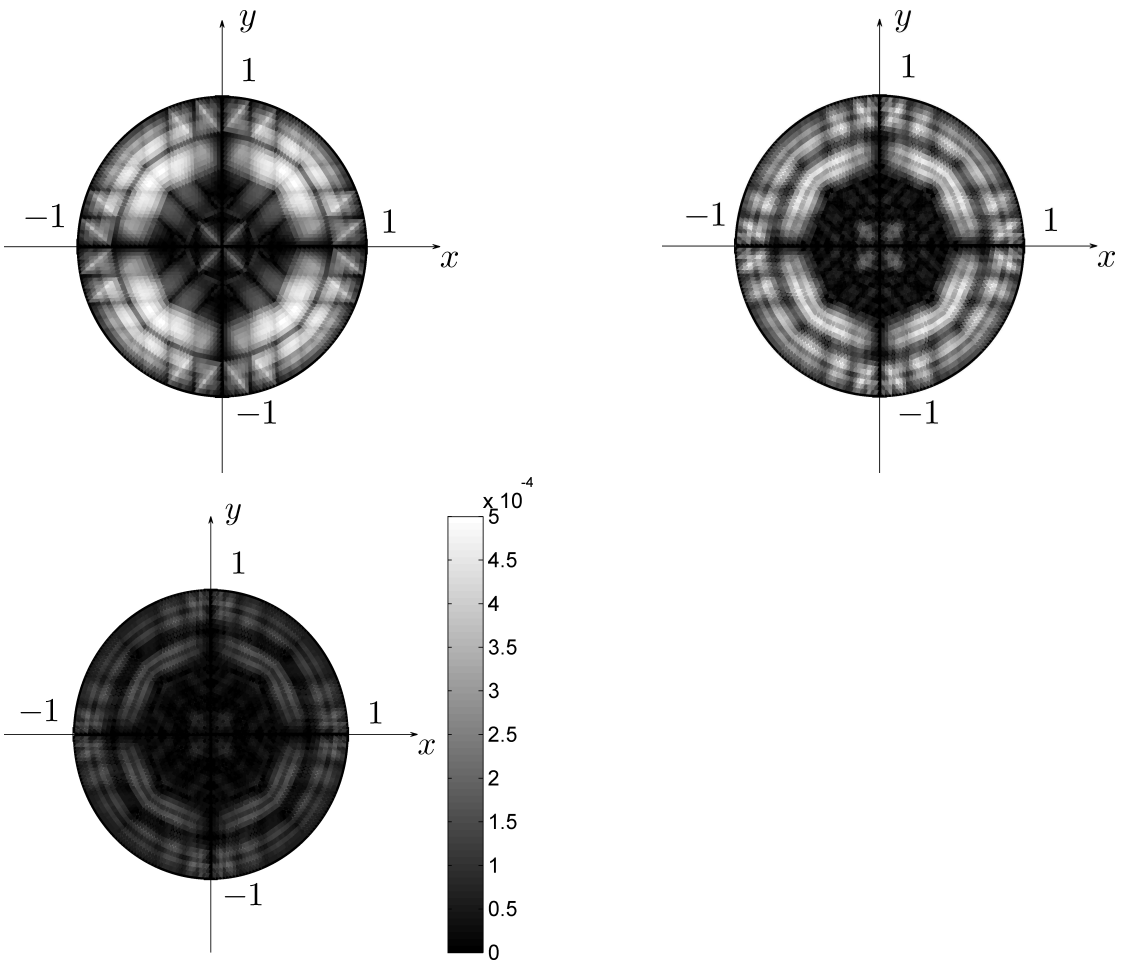

Figure 3.4: The pictures above show graphs of $|u - u_h|$ after the first three iteration of the cycle.

Algorithm 8 Calculation of the coarsening matrix

NCP : number of coarse points
for $k = 1$ to NCP **do**
  $i(s) = k$
  $j(s) = k$
  $s = s + 1$
end for
for $k = 1$ to $3 \cdot nel$ **do**
  if $w2(nodes(k, 1)) == 1$
  $i(s) = nodes(k, 2), j(s) = nodes(k, 1), s = s + 1;$
  $i(s) = nodes(k, 3), j(s) = nodes(k, 1), s = s + 1;$
end for
$u = zeros(length(i), 1)$
$u(:) = 1/4$
$u(1 : length(w2)) = 1$

3.2.2 example

Let us consider

$$-\Delta u = f, \tag{3.1}$$

with $u = 0$ on the boundary. The domain Ω is a unit square $[0,1] \times [0,1]$. We choose the right-hand side f such that

$$u = x^2 y(1-x)(1-y)$$

is the seoluion of $-\Delta u = f$.

$$u_x = -2xy(1-x)(1-y) - x^2 y(1-y),$$

$$u_{xx} = -2y(1-x)(1-y) + 4xy(1-y) + 2x^2(1-x),$$

$$u_y = x^2(1-x)(1-y) - x^2 y(1-x),$$

therefore

$$f = -2y(1-x)(1-y) + 4xy(1-y) + 2x^2(1-x).$$

We can now compute the energy norm of exact solution:

$$\nabla u \cdot \nabla u = 2xy(1-x)(1-y) - x^2 y(1-y))^2 + (x^2(1-x)(1-y) - x^2 y(1-x))^2,$$

$$\|u\|_A = \sqrt{\frac{4}{525}}.$$

At first we set up the problem. This example is predefined as *problem* 0. The initial mesh is generated with one interior node on the coarse level. The mesh is shown in the picture 3.5 on the right and a graph of the exact solution is shown on the left.

In the tables 3.2 - 3.5 we can study the convergence of the multigrid cycle with the polynomial smoother, with Gauss Seidel smoother and just Gauss Seidel iterative scheme. The coarsening factor means the number of levels which we 'skip' during coarsening. It is obvious that Gauss Seidel as a smoother doesn't work for aggressive coarsening, but polynomial smoother works well.

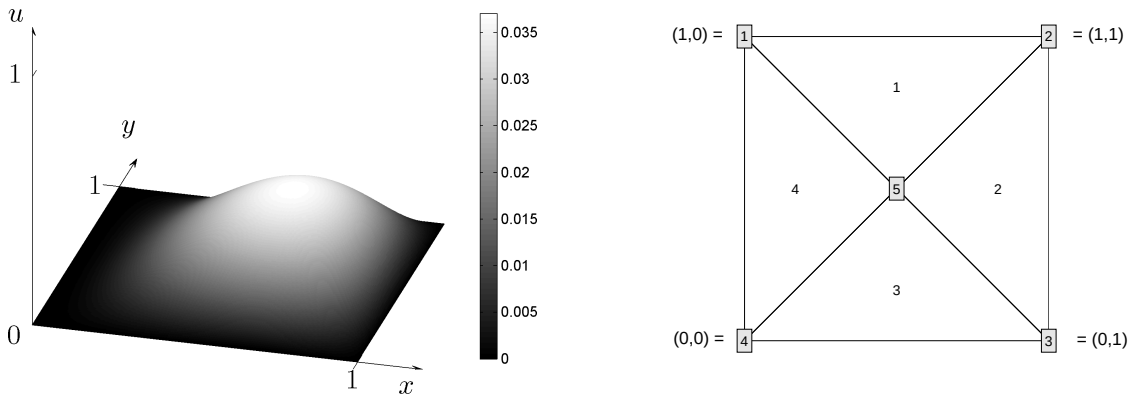

Figure 3.5: The graph of exact solution of (3.1) and the initial mesh (right).

Table 3.1: Computational values for solving (3.1).

m	9
coarsening time	1.565571e+000
matrix assembly time	3.729150e+000
Deg. of freedom	130561
Number of nonzero el. of A	650761
ϵ	10^{-6}

Table 3.2:

coarsening factor	multigrid solver time (s)	polynomial degree	number of iterations	α	rate
1	1.740449e+001	1	16	1.007809	2.060952e-001
2	4.275317e+001	2	18	1.007127	2.563962e-001
4	3.565271e+002	8	15	0.985567	2.662881e-001

Table 3.3:

coarsening factor	multigrid solver GS time (s)	number of iterations	α	rate
1	2.8003e+001	25	1.008	0.37681
2	6.7686e+001	72	0.999	0.737
4	4.3129e+002	400	0.9998	0.9797

Table 3.4:

NoI	comp. error GS	comp. error pol.s.
1	6.4436e+004	8.2088e+003
2	5.7229e+004	1.1257e+003
3	5.2336e+004	1.5492e+002
4	4.8615e+004	2.2786e+001
5	4.5464e+004	3.4662e+000
6	4.2692e+004	5.4181e-001
7	4.0206e+004	8.8056e-002
8	3.7950e+004	1.5096e-002
9	3.5883e+004	2.7938e-003
10	3.3978e+004	5.6757e-004
11	3.2212e+004	1.2646e-004
12	3.0566e+004	3.0319e-005
13	2.9028e+004	7.6425e-006
14	2.7584e+004	1.9901e-006
15	2.6227e+004	5.2995e-007

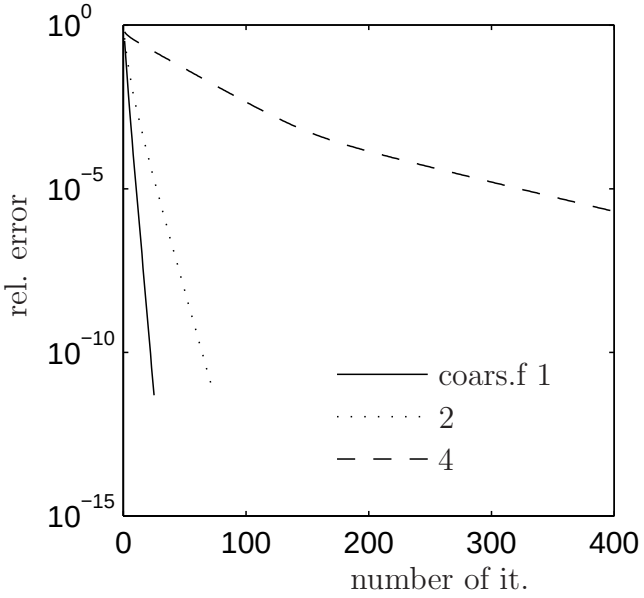

Figure 3.6: Convergence for GS smoothing

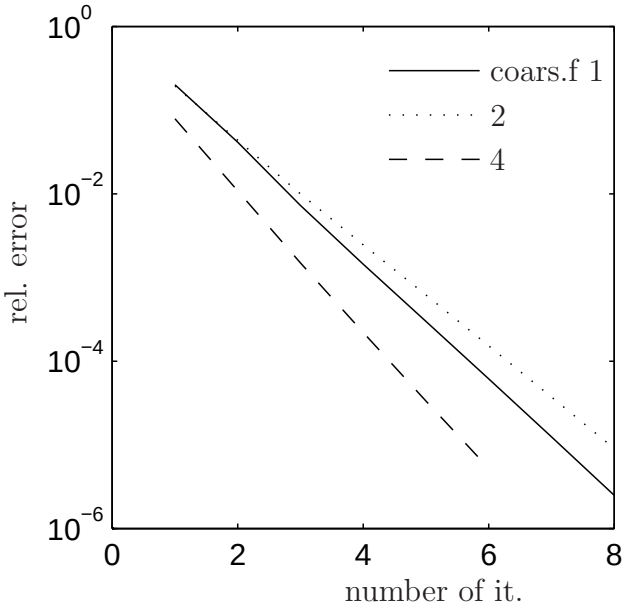

Figure 3.7: Convergence for pol. smoothing

Table 3.5:

NoI	rel error Mg with GS	relerror pol.s.	pure GS rel err.
1	6.2126e-001	7.9144e-002	948.4670609e-003
2	5.5177e-001	1.0854e-002	899.864136e-003
3	5.0459e-001	1.4936e-003	854.7872352e-003
4	4.6872e-001	2.1969e-004	813.0582955e-003
5	4.3834e-001	3.3420e-005	774.3215218e-003
6	4.1161e-001	5.2239e-006	738.2420133e-003
7	3.8764e-001	8.4899e-007	704.5312213e-003
8	3.6589e-001	1.4554e-007	672.9442175e-003
9	3.4597e-001	2.6936e-008	643.272465e-003
10	3.2760e-001	5.4722e-009	615.3369700e-003
11	3.1057e-001	1.2192e-009	588.9827573e-003
12	2.9470e-001	2.9231e-010	564.0746101e-003
13	2.7987e-001	7.3685e-011	540.4937804e-003
14	2.6595e-001	1.9188e-011	518.1354164e-003
15	2.5286e-001	5.1095e-012	496.9065087e-003

3.3 Example

Here, we consider a Neumann problem

$$\begin{aligned} -\Delta u &= 10 \quad \text{in} \quad \Omega, \\ u &= x + y \quad \text{on} \quad \partial\Omega. \end{aligned}$$

We have to deal with the boundary term now. The stiffnesses matrix and the load vector will be computed by command `assempde` again. The geometry files which generates the boundary data can be easily computed in the `pdetool`. Second thing is, that we have to consider the boundary nodes in the computations of prolongators. The diagonal entries are ones again. We now distinguish between boundary nodes and interior nodes. Since we account each interior node twice, the entry to the sparse matrix stays $(i, j, 1/4)$. For the boundary nodes, which are considered ones in the algorithm, the entry is $(i, j, 1/2)$. This is carried by `prolmatrix5`.

Following tables shows results for the algorithm. If we use polynomial smoother, the number of iteration stays relatively the same. Again, we compare polynomial smoother and GS smoother .

Table 3.6:

m	5
coarsening time	1.2196e-001
matrix assembly time	3.6175e-001
galerkin matrices time	1.4877e-002
deg. of freedom	4993
number of nonzero el. of A	34433
ϵ	10^{-6}

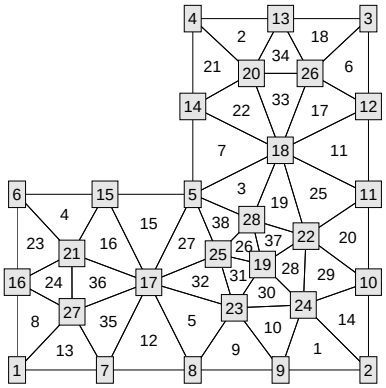

Figure 3.8: Inital mesh

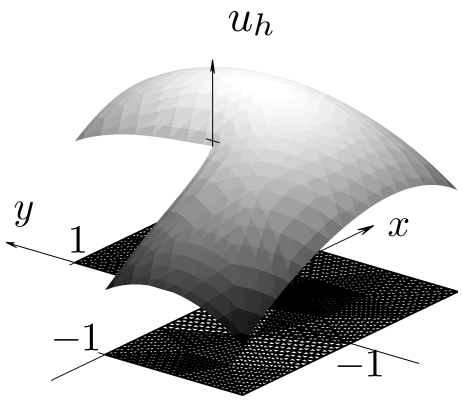

Figure 3.9: Solution to the Neumann problem

Table 3.7:

coarsening factor	multigrid solver time (s)	polynomial degree	number of iterations	α	rate
1	5.3448e-001	1	9	9.8303e-001	1.8674e-001
2	6.0693e-001	2	9	9.9554e-001	1.9223e-001
4	6.2584e+000	8	8	9.8706e-001	1.6871e-001

Table 3.8:

NoI	rel	abs	relres
1	1.1910e-001	1.4429e+000	2.4909e+000
2	1.4143e-002	1.7134e-001	1.0044e-001
3	2.2660e-003	2.7451e-002	1.4975e-002
4	4.0132e-004	4.8618e-003	3.0500e-003
5	7.6168e-005	9.2275e-004	6.6055e-004
6	1.5179e-005	1.8389e-004	1.4185e-004
7	3.1242e-006	3.7849e-005	3.0553e-005
8	6.5633e-007	7.9512e-006	6.5683e-006

Table 3.9:

coarsening factor	multigrid solver time (Gs)	number of iterations	α	rate
1	3.2256e-001	11	9.9313e-001	2.7635e-001
2	6.1448e-001	29	9.9897e-001	6.1448e-001
4	8.6440e+000	377	9.8706e-001	9.6396e-001

Table 3.10:

NoI	rel	abs	relres
1	2.5629e+000	1.5183e+000	2.5629e+000
2	1.3958e+000	1.2155e+000	1.3958e+000
3	9.9521e-001	1.0634e+000	9.9521e-001
4	8.4588e-001	9.6002e-001	8.4588e-001
5	7.4667e-001	8.7608e-001	7.4667e-001
6	6.7121e-001	8.0437e-001	6.7121e-001
7	6.0979e-001	7.4143e-001	6.0979e-001
8	5.5763e-001	6.8528e-001	5.5763e-001

3.3.1 example

Let us consider

$$-\Delta u = f,$$

with $u = 0$ on the boundary.

This is our last example. The domain is shown in the picture 3.10. Following tables and graphs shows the convergence results for the algorithm. The picture 3.13 shows the convergence when we use incorrect degree of the smoother = 1.

The tables 3.13 and 3.14 give the most important results. As we see, the convergence is improving when degree of the polynomial is growing. On the other side, the number of non zeros of the smoother is growing in each step, since the smoother is a polynomial of A. Therefore the matrix-matrix computations are very expensive and a large problem can't be solved in the reasonable time. As we would like to solve large systems, this is a massive complication.

The first four iteration of the method are shown in 3.11. We can see the mark of the initial the initial triangulation and how the error get smoothed.

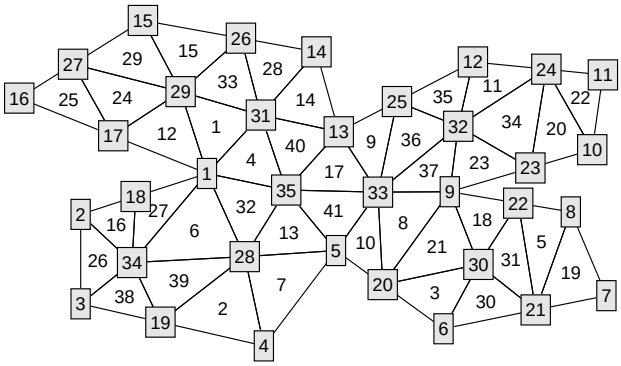

Figure 3.10: Initial mesh

Table 3.11:

m	9
coarsening time	6.5472e-001
matrix assembly time	4.6654e-001
Galerkin assembly time	1.8832e-002
deg. of freedom	5033
number of nonzero el. of A	34379
ϵ	10^{-6}

Table 3.12:

coarsening factor	multigrid solver time (s)	polynomial degree	number of iterations	α	rate
1	6.9622e-001	1	14	9.9955e-001	3.5930e-001
2	9.2971e-001	2	13	9.9994e-001	3.4383e-001
4	1.4025e+001	8	14	9.9980e-001	3.6126e-001

Table 3.13:

multigrid solver time (s)	polynomial degree	number of iterations	rate	NNZ
6.3479e-001	1	14	3.5930e-001	656
8.1637e-001	2	9	1.8037e-001	1562
1.0729e+000	3	7	1.0754e-001	2690
1.4757e+000	4	6	9.6589e-002	3952
1.8088e+000	5	5	6.1993e-002	5296
2.6479e+000	6	5	5.8273e-002	6666
3.6143e+000	7	5	5.8273e-002	7944

Table 3.14:

polynomial degree	number of iterations	rate	NNZ
1	12	2.9146e-001	656
2	7	1.2530e-001	1562
3	5	5.8990e-002	2690
4	5	4.4151e-002	3952
5	4	2.3992e-002	5296
6	4	1.5771e-002	6666
7	4	1.2964e-002	7944
8	3	9.2997e-003	9060

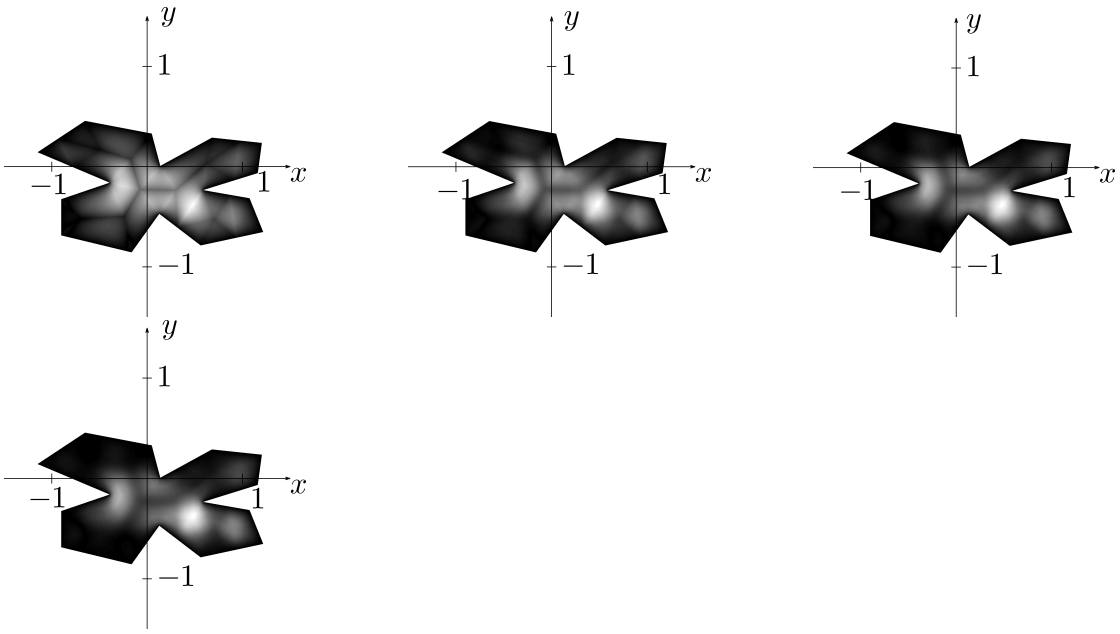

Figure 3.11: First four iterations of the cycle

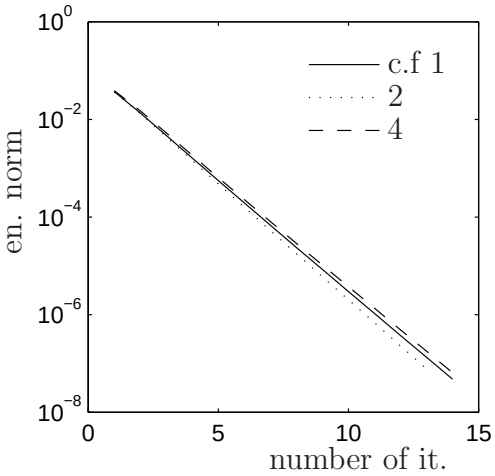

Figure 3.12: Energy norm using polynomial smoother of right degree

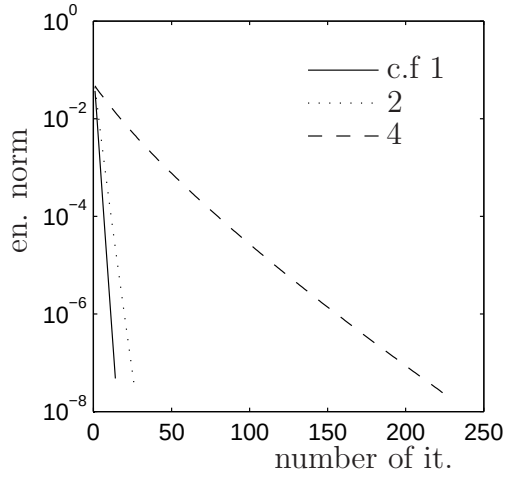

Figure 3.13: Energy norm using polynomial smoother of degree 1

3.4 Fortran implementation

We implemented a solver in Fortran 90 to experiment with the algorithm. In Fortran we are forced to implement everything on our own, on the other side it gives us many options to think about the algorithm more deeply. We work with the main program main.f90, where the initialization, refinemesh, computation of the matrices and solver are held. There are two modules mgtypes.f90 and matrixassembly.f90.

We have implemented everything straight for sparse matrices. The type mg is some kind of a container, where it consists of rhs, x, coarsening matrix and the Galerkin matrix. We create a vector mg(:) - for each level one spot in the vector. The mgtypes also contains sparse matrix type. Since we think this topic is important, we give a short introduction to the sparse storage formats.

3.4.1 Sparse Matrices Storage formats

In the case of piecewise linear basis functions the matrices A^k and C^k are sparse. The sparse matrices has most of the entries equal to zero. It is therefore not necessary to store matrices as dense.

We denote N_z as the number of nonzero elements.

Coordinate format

The coordinate format contains three arrays: a real arrays which contains entries of the matrix, integer array containing row indices and an integer array containing column indices.

The order of entries is stored in arrays is arbitrary. It might look like this:

$$A = \begin{pmatrix} 0 & 2 & 0 & 4 & 0 \\ 0 & 0 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 9 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

then

$$a = [2, 1, 4, 9, 5, 4, 1]$$

$$jr = [1, 5, 4, 4, 2, 1, 4]$$

$$jc = [2, 3, 3, 4, 3, 4, 5]$$

CSR Structure

The storing of nonzero elements can be done such that we go across all the rows (rows format) or columns (column format) and we write all the elements in the order how they appeared. Our three arrays would have following meaning:
a: A real array of length N_Z containing nonzero elements of A,
of : An integer array of positions in the column ,
xm : A pointer array, where i-th entry points to the beginning of i-th row in the arrays. The length of the array is N+1, containing also the number $of(1)+N_Z$.

Taking A from previous example, our arrays would have following form:

$$a = [2, 4, 5, 4, 9, 1, 1]$$

$$of = [2, 4, 3, 3, 4, 5, 3]$$

$$xm = [1, 3, 4, 7, 8]$$

Les A be a symmetric matrix. We then can store only lower triangular elements of the matrix.

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 8 & 2 & 1 \\ 0 & 0 & 2 & 9 & 0 \\ 0 & 0 & 1 & 0 & 10 \end{pmatrix}$$

The storage arrays for a symmetric matrix is

$$a = [1, 3, 8, 2, 9, 1, 10]$$

$$of = [1, 2, 3, 3, 4, 3, 5]$$

$$xm = [1, 2, 3, 4, 6, 8]$$

CSC Structure

The compressed sparse column format is similar to the CSR format, but we go across columns instead of rows.

3.5 Implementation issues

We will now talk shortly about some implementation matters we consider interesting. Previously, in Matlab, we had to refine all the meshes and then compute the coarsening matrix. We had to go over number of coarse elements $\cdot$ 3. In Fortran, we wanted to refine the elements in similar way like it is in Malab, but while

constructing a new mesh, we use the information about the mesh and create coarsening matrix immediately. To coarsen initial mesh from *problem* 0 to 10 levels, we need 10.738 seconds in Matlab and in Fortran just 0.0858 seconds.

If we compare matrix and rhs creating, in Matlab we need 15.675 seconds and 3.01 in Fortran.

The program was created with the idea that it could be parallelized in the future. This is the case of creating the local stiffness matrices and mainly matrix vecotr multiplication. It is well known that CSR times vector can be parallelized easily, but CSC not. We therefore store coarsening matrices in CSC storage format (number of rows is much greater then the number of columns.) Matrix A is stored in CSR format, due to the future Galerkin multiplication. This is however the most problematic part. Since the sparse vectors in general are not entered entry by entry, the computation takes much longer time. Therefore in our Fortran code the time to compute the Galerkin matrices takes too long. Our task to the future will be to make the program run in parallel and implement matrix-matrix multiplication in more optimal way (reordering algorithms, block storage schemes).

Conclusion

In this thesis we were dealing with in the geometrical multigrid methods. The main goal was to implement the algorithms and create the test problems to numerically verify the results of [18]. We have created a set of test problems using Dirichlet and Neumann conditions, various meshes, smoothers and degrees of freedom. In the first and second part we gave a brief introduction to the variational problems. We then continued with the finite element method. We mentioned the most important theorems and described the computational aspects of the finite elements method. The fourth part consists of the introduction to the multigrid. The theory of the geometric multigrid is wide, therefore we focused on the topics which we consider to be close to the article [18].
The last and most important part covers the description of the implementation and numerical experiments. It has been numerically verified that the convergence result is independent of h_{k+1}/h_k. The problematic part of the algorithm is that the smoother is polynomial of A. Large coarsening factor (= aggressive coarsening) reflects larger degree of the smoother, which means that the number of nonzero elements of the smoother grows. As we solve large sparse systems, matrix-matrix multiplication affects the computational time massively. Multigrid with the polynomial solver degree greater than 8 for large problems seems not to be an advantage. Then we implemented the program in Fortran 90. This gives us more freedom to implement our algorithm and a source code base, which can be extended in the future. We were able to implement the program using some ideas we had during Matlab implementations, on the other hand some aspects weren't working for us. It will be therefore our work in the future to improve and extend the code.

Bibliography

[1] Braess, D.: Finite elements : theory, fast solvers, and applications in elasticity theory 3rd ed. Cambridge University Press, 2007.

[2] Bramble, James H.: Multigrid methods, Longman Scientific and Technical, 1993.

[3] Bramble, James H. and Pasciak, Joseph E. and Xu, Jinchao: Parallel Multilevel Preconditioners, Mathematics of Computation 55:1-22, 1990.

[4] Briggs, W. L.: A multigrid tutorial, Society for Industrial and Applied Mathematics, 1987.

[5] Gräser, Carsten; Kornhuber, Ralf: Multigrid methods for obstacle problems. J. Comput. Math. 27, no. 1, 1-44, 35J86 (65K10 65N55), 2009.

[6] Hackbush, W. : Multi-Grid Methods and Applications, Apringer-Verlag, Heidelberg, 1985.

[7] Hoppe, R. H. W.; Kornhuber, R. Adaptive multilevel methods for obstacle problems. SIAM J. Numer. Anal. 31, 1994.

[8] Kornhuber, R.: Monotone iterations for elliptic variational inequalities. Free boundary problems: theory and applications, 1997.

[9] Kornhuber, R.: Adaptive monotone multigrid methods for nonlinear variational problems, . 158 pp. ISBN: 3-519-02722-4 (Reviewer: Dietrich Braess), 65K05 (35R35 65N30 76M25), 1997.

[10] Kornhuber, R.: Monotone multigrid methods for elliptic variational inequalities. II. Numer. Math. 72 , no. 4, 481-499, 65N55 (65K05), 1996.

[11] Kornhuber, Ralf Monotone multigrid methods for elliptic variational inequalities. I. Numer. Math. 69, no. 2, 167-184, 65N55 (65K05), 1994.

[12] Larson, M.G., Bengzon, F. : The Finite Element Method: Theory, Implementation, and Practice, Springer, 2009.

[13] Pacheco, P. S.: Parallel programming with MPI Morgan Kaufmann Publishers, 1997.

[14] Süli, E. : Finite Element Methods for Partial Differential Equations, University of Oxford, 2000.

[15] Trottenberg, U.: Multigrid, Academic Press, 2001.

[16] Vassilevski, P.: Lecture Notes on Multigrid, 2010.

[17] Vaněk, P.: Convergence of Algebraic Multigrid Based on Smoothed Aggregation, 1998.

[18] Vaněk, P., Brezina,M.: Nearly Optimal Convergence Result for Multigrid with Aggresive Coarsening and Polynomial Smoothing.

[19] Vaněk, P.: Základy multigridu a zhlazených agregací

[20] Xu, Jinchao: Iterative methods by space decomposition and subspace correction, SIAM Review34:581-613,1992.