Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra matematiky

Paralelní víceúrovňové algebraické předpodmiňovače

Ing. Pavla Fraňková, M.Sc.

disertační práce
k získání akademického titulu doktor
v oboru Aplikovaná matematika

Školitel: Doc. Ing. Marek Brandner, Ph.D.

Plzeň, 2019

University of West Bohemia
Faculty of Applied Sciences
Department of Mathematics

Parallel Multilevel Algebraic Preconditioners

Ing. Pavla Fraňková, M.Sc

Thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy (Applied Mathematics)

Supervisor: Doc. Ing. Marek Brandner, Ph.D.

Pilsen, 2019

Anotace

V této práci studujeme víceúrovňové předpodmiňovače. V úvodní části popíšeme teorii víceúrovňových metod (metod multigridu). Pak spojíme dvě zajímavá témata (BPX aditivní multigrid a metodu zhlazených agregací) do nové metody. Představujeme BPX předpodmiňovač založený na metodě zhlazených agregací na modelové úloze. Teoretický výsledek je ověřen sérií numerických experimentů. Poslední kapitola se zabývá aplikací multigridu v modelování transportu neutronů.

Klíčová slova

Algebraický multigrid, zhlazené agregace, BPX, iterační metody, modelování transportu neutronů

Annotation

This thesis studies multilevel preconditioners. In the introduction, we cover the theory of multigrid methods. We then connect two interesting topics (BPX additive multigrid and smoothed aggregation method) into a new multigrid approach. We will present a BPX preconditioner in the smoothed aggregation principle settings on the model case. The theoretical results are validated by the series of numerical experiments. The last chapter covers applications of the multigrid preconditioners in the neutron transport modeling.

Key words

Algebraic multigrid, smoothed aggregation, BPX, iterative methods, neutron transport modeling

Acknowledgments

I would like to thank my supervisors Ing. Petr Vaněk, Ph.D. and Doc. Ing. Marek Brandner, Ph.D. for their assistance, guidance and advises. I would also like to thank other members of the department for their help throughout my studies.

My thanks also belong to my friends and co-workers form Škoda JS, a.s. (Škoda Nuclear Machinery), for introducing me into the field of physics and modeling of neutron transport.

Finally, I would like to thank my mom and Karel for their support.

I hereby declare that this doctoral thesis is the result of my own work, unless clearly stated otherwise.

....
Pavla Fraňková

Contents

1 Mathematical Preliminaries **5**

1.1 Function and Vector Spaces 5

1.1.1 Euclidean Space $\mathbb{R}^n$ 5

1.1.2 Spaces of Continuous Functions 6

1.1.3 Spaces of Integrable Functions 6

1.1.4 Sobolev Spaces 7

1.2 Some Results from the Finite Element Methods 9

1.3 Some Other Results 11

2 Geometric Multigrid Methods **13**

2.1 Iterative Methods 13

2.1.1 Richardson Iterative Method 15

2.1.2 Gauss-Seidel and SOR Methods 17

2.2 Introduction to Multigrid 18

2.3 The Multigid Cycle 22

2.3.1 Multigrid by Subspace Splitting 24

2.3.2 Convergence Estimates 27

2.3.3 Summary of Chapter 1 30

3 Algebraic Multigrid **33**

3.1 Coarsening Process 34

3.2 Smoothed Aggregation Method 36

3.2.1 SA Two-Level Method 37

3.2.2 Coarsening Method 41

3.2.3 Generalized Aggregation 42

3.3 BPX Algorithm in SA Setting 44

3.3.1 Introduction 44

3.4 BPX pre-conditioner in operator setting 46

3.5 Smoothed aggregation prolongators in model case 50

3.6 Verification of the assumptions of the abstract theory 52

4 Implementation and Numerical Experiments **73**
4.1 Sparse Matrices Storage Formats 73
4.2 Multigrid Programming Essentials 75
4.2.1 Parallel Programming Models 77
4.2.2 Parallelization with OpenMP 78
4.3 Implementation of the Model Problem 81
4.3.1 Implementation Details 83
4.3.2 Conjugated Gradients Method 84
4.4 Numerical Experiments 86
4.4.1 Parallelization of Algorithm 4.3.1 87
4.4.2 Other Numerical Experiments 91

5 Applications **97**
5.1 Neutron flux calculations by transport equation 97
5.2 Multigroup Diffusion Approximation 102
5.2.1 Neutron flux calculations 103
5.2.2 Finite Volume Discretization 105
5.3 Numerical experiments 110
5.3.1 Computation of k_{eff} by the Power Method 111
5.3.2 Computation of k_{eff} by Jacobi-Davidson Method 115
5.3.3 Conclusions of Chapter 5 116

List of symbols

Symbols of Chapters 1-4

$\mathbb{R}^d$	$d-$ dimensional Euclidean space	
Ω	problem domain (bounded connected open set in $\mathbb{R}^d$)	
Ω^h	space of fine grid points	(two-level method)
Ω^H	space of coarse grid points	(two-level method)
$\langle\cdot,\cdot\rangle$	vector product	
$\|\cdot\|_{l^p}$	l^p norm	
l^p	space of sequences	
$C^k(\Omega)$	spaces of continuous functions	
$\|u\|_{C^k(\bar{\Omega})}$	C^k norm	
$L_p(\Omega)$	spaces that consist of Lebesgue integrable functions	
$\|u\|_{L_p(\Omega)}$	$L_p(\Omega)$ norm	
$(\cdot,\cdot)$	inner product	
$W_p^k(\Omega)$	Sobolev space	
$H^1(\Omega)$	space $W_2^1(\Omega)$	
$(\cdot,\cdot)_A$	$A-$ inner product	
$\|\cdot\|_A$	$A-$norm	
U	Hilbert space, if not stated otherwise	
$\mathcal{A}(\cdot,\cdot)$	continuous bilinear form	
$\|\cdot\|_{\mathcal{A}}$	energy norm	
$f(\cdot)$	continuous linear form	
A	finite dimensional discretization operator	
A_h	fine level operator	(two-level method)
A_H	coarse level operator	(two-level method)
A_l	finite dim. discretization operator on the level l	
$\mathbf{f}$	load vector or right-hand side	
$\mathbf{r}$	residual	
$\mathbf{e}$	error	
$\mathbf{u}$	solution to a problem $A\mathbf{u}=\mathbf{f}$	
E	error propagation operator	
$\mathcal{V}_h$	finite dimensional Hilbert space	
V	finite dimensional vector space	
L	number of levels	
I_h^H	restriction operator	(two-level method)
I_H^h	prolongation operator	(two-level method)

I_{l-1}^{l}	prolongation operator: $\mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}, l = L, \ldots, 2$	(Sec. 2)
$(I_{l-1}^{l})^{\mathrm{T}}$	restriction operator: $\mathbb{R}^{n_l} \to \mathbb{R}^{n_{l-1}}, l = L, \ldots, 2$	(Sec. 2)
I_{l+1}^{l}	prolongation operator: $\mathbb{R}^{n_{l+1}} \to \mathbb{R}^{n_l}, l = 1, \ldots, L-1$	(Sec. 3, 4)
$(I_{l+1}^{l})^{\mathrm{T}}$	restriction operator: $\mathbb{R}^{n_l} \to \mathbb{R}^{n_{l+1}}, l = 1, \ldots, L-1$	(Sec. 3, 4)
$\mathcal{A}$	aggregate	(Sec. 3, 4)
P	prolongator	
$\mathcal{S}$	smoothing operator	
ν	number of smoothing steps	
S	smoothing iteration matrix	
n	number of fine points	(two-level method)
m	number of coarse points	(two-level method)
n_l	number of points on level l	
B	approximate inverse of$A$	
$\mathcal{B}$	multilevel preconditioner	
$\mathcal{P}$	A orthogonal projection	
$\mathcal{Q}$	L_2 orthogonal projection	
π	finite element interpolator	
φ	basis function	
ϕ	macroelement mapping	
γ	number of MG$(\mathbf{x}, \mathbf{f})$ steps	

Symbols of Chapter 5

E	energy of neutrons
r	position of neutron
t	time
$\mathbf{\Omega}$	unit vector of neutron's direction of motion
ϕ	neutron flux
D	diffusion coefficient
$\bar{D}$	effective diffusion coefficient on the boundary
g	energy group
k_{eff}	effective multiplication factor
$\mathbf{j}$	neutron current
Σ	macroscopic cross section
$\nu\Sigma_f^{h\to g}$	cross section for neutron scattering from energy group g to h
χ	fission spectrum

Shortcuts

$MG(\mathbf{x},\mathbf{f})$	multigrid algorithm
GM	geometric multigrid
AMG	algebraic multigrid
SA	smoothed aggregations
BPX	preconditioner of Bramble, Pasciak and Xu
SSC	successive subspace correction
PSC	parallel subspace correction
CSR	compressed sparse row format
CSC	compressed sparse column format
COO	coordinate format

Introduction

The multigrid methods are strong tools for solving large scale algebraic systems coming from partial differential equations. Throughout the years, it has been developed from a simple idea to a whole family of solvers and efficient preconditioners. We find it convenient to start with a brief history of the method, as it shows that from the very beginning, the method was constantly developed along with the evolution of computational mathematics and programming.

The history of multigrid begins with the work of R.P. Fedorenko, who was, in the very late fifties, a scientist at the Steklov Institute under I.M. Gelfand. Fedorenko described the inception in the note On the history of the Multigrid method creation in 2001. The note was then translated by M. A. Botchev.

Fedorenko was dealing with a Poisson equation in a rectangular domain. In his program, a 48x40 grid was used so that the unknown grid function and the right hand side vector occupied almost all the operational memory. He used a simple iteration method and observed a known fact, that the nonsmooth residual decreased fast and became smooth. After this, the decrease became slow. He then formulated the correction equation as a problem on a coarse grid with the residual at the right hand side. This approach probably came from a Newton method for linear equations, which leads to a similar problem.

Fedorenko continued in his work in 1962 [21], where the method described was called a *relaxation method.* In the paper [22] from 1964 a multigrid algorithm was formulated for a five point finite difference discretization scheme of the Poisson equation on a square. It was proven that the work needed to achieve a certain accuracy was *O(N).* Fedorenko was then followed by Bakhvalov in 196l [2] .

After the first steps of multigrid, its development stopped for a couple of years. Multigrid methods were rediscovered and developed in the seventies by Brandt [9], [8], Hackbusch [29] and other authors and became widely known through the eighties. Since the coarsening process in the standard approach is based on the geometrical grid, the original method is called the geometric multigrid and in this thesis will be referred to as GM.

The Brandt-Hackbusch standard multigrid theory required a "regularity and approximation" assumption. It was proved using elliptic regularity of the underlying partial differential equation as well as approximation and inverse properties of the discrete multilevel spaces. Later, the two-level schemes were proved to converge under weaker assumptions [29], [10]. The estimates for general multilevel method without regularity assumptions were proved in [6]. This included convergence results for multigrid refinement applications, problems with irregular coefficients and problems with high jumps in coefficients.

The subsequent years brought a continuing progress of high performance computers. Larger and more complicated problems were solved on the unstructured meshes and demands on efficient solvers and pre-conditioners were rising. It turned out that problems occurred when, for example, when the anisotropic problems were solved.

In the case of anisotropic problems, more complex smoothers are required in order to still achieve a fast multigrid convergence.

Also, in the geometric multigrid, the prolongation matrices are closely connected with the geometrical structure of the problem and it was impossible to create a black-box solver which would create prolongation matrices based just on the linear systems arising from the differential equations.

The algebraic multigrid (AMG) was introduced by Achi Brandt [10] as a method for solving linear systems based on multigrid principles. In contrast to the geometric multigrid, the interplay between grids was based just on the algebraic structure of the stiffness matrix.

We can say that in the AMG, restriction matrices are created by fast and simple algorithms and an information about the grid is no longer required - the coarsening process is automatic. This fact is balanced by the need of robust smoothers. If we use term *sets of variables* instead of *grids* and *single variables* instead of *grid points*, the AMG approach is formally the same as in the GM. We will also see that smoothness has purely algebraic meaning. In the GM, the error after some smoothing steps is geometrically smooth, but in AMG, the algebraically smoothed error can be geometrically oscillatory.

For these advantages, we have to pay a price - before an actual solution phase starts, the set-up phase has to be done. It has been shown in many examples, that the AMG can be less efficient when the GM can be applied. Also, in many cases the AMG performs better as a pre-conditioner than a stand alone solver.

Since the introduction of the original method, a wide variety of AMG algorithms have been developed. The classical approach was presented by Ruge and Stüben ([42], [45]) and is based on the *strong dependency* of unknowns. Next to the classical approach, there exists a class of AMG solvers based on the *aggregation* principle. The smoothed aggregation (SA) approach showed up to be efficient tool for solving various type of problems [56]. It is strong in the context of solving large scale systems of linear algebraic equations arising from discretization of elliptic problems and their singular perturbations. In this thesis we focus mainly on the SA principle.

An outline of this thesis is as follows:

The text is divided into three main parts. The first part of the thesis consists of Chapters 1 and 2 and contains mainly a theory regarding multigrid methods. Chapter 3 contains numerical aspects of the algorithm, programming issues and numerical experiments. The last chapter contains a comparison of some solvers including multigrid methods in the application coming from neutron diffusion problem.

The first chapter covers an area of the geometric multigrid and the main aim of this chapter is to make a comprehensible introduction to the reader. We wanted to describe a rather wide area of multigrid in a way that it would be understandable to a interested reader without previous knowledge of the multigrid, so we start from the basics and explain how multigrid works and it's main principles. We will see that multigrid can be looked at from different angles as well as there exists different derivations of two grid method. In the end of the chapter we present some most important convergence result of Brandt-Hackbusch and of Bramble, Pasciak, Xu and Wang.

Chapter 2 follows by the introduction to the algebraic multigrid. The AMG section is focused mainly on the smoothed aggregation method by Vaněk, Mandel and Brezina. In the second part of the chapter, we propose a connection of the interesting topics: BPX additive multigrid ([6]) and smoothed aggregation method into a new multigrid approach. According to [6], the classical multigrid can be viewed as a multiplicative method of Schwarz type with inexact subspace solvers given by smoothers. As a consequence, its fundamental components contains inner dependencies and have to be performed in a sequence. Unlike standard multigrid, the so-called BPX pre-conditioner frame of Bramble, Pasciak, and Xu [7] is fully additive, allowing for a parallelism on each level. We derive a convergence result for BPX preconditioner is the SA settings on the model case.

The theoretical part is followed by numerical results and applications in Chapter 3. Before we proceed to the actual results, we describe the essentials of programming the multigrid method in parallel. We programmed a code suitable for solving elliptic problem by various solvers. The numerical experiments are done for the model problem in a serial and parallel version. We provide numerical verification of new theoretical results derived in Chapter 2. Then we present a series of numerical experiments involving convergence of the algorithm and we also discuss the results in context of other multigrid methods.

In the last chapter of the thesis, we focus on the applications - modeling of the transport of neutrons in the reactor core. The model we use is the multi-group diffusion approximation which is solved for the neutron flux and so called critical number of the reactor. It leads to a generalized eigenvalue problem. We implemented a module based on the finite volume method which is then used for solving the eigenvalue problem. The main aim of this section is to do a series of numerical experiments on the calculation of the neutron flux and the critical number and consequently make some recommendations regarding used solvers and preconditioners for the given problem.

Chapter 1

Mathematical Preliminaries

In this chapter, we summarise the basic mathematical preliminaries, which will be necessary in the later chapters. We start with the function spaces and continue with variational formulation and solvability of the elliptic problem. More information can be found in [17]. We then present some known facts from the theory of FEM.

1.1 Function and Vector Spaces

1.1.1 Euclidean Space $\mathbb{R}^n$

Let $\mathbb{R}^n$ be an n-dimensional Euclidean space and $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x} = (x_1, \ldots, x_n)$ a vector. The vector product $\langle \cdot, \cdot \rangle$ is a function

$$\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

defined by the formula

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} x_i y_i.$$

The norm $\|\mathbf{x}\|$ is defined as

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}.$$

We will also use the space of sequences l^p, $0 < p < \infty$. Let $(x_i)_{i=1}^{\infty}$, $x_i \in \mathbb{R}$ be a sequence of real numbers. Then l^p is a subspace of $\mathbb{R}^n$ consisting of sequences (x_n):

$$\sum_{i=1}^{\infty} |x_i|^p < \infty.$$

The l^p norm is given by

$$\|x_n\|_{l^p} = \left(\sum_{i=1}^{n} |x_i|^p\right)^{\frac{1}{p}}.$$

1.1.2 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 the 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

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

1.1.3 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*

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

1.1.4 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 $|\alpha|$)

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

$W_p^k(\Omega)$ is called a *Sobolev space* of order k; it is equipped with the (Sobolev) norm

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

Letting

$$|u|_{W_p^k(\Omega)} := \left(\sum_{|\alpha| = k} \|D^\alpha u\|_{L_p(\Omega)}^p \right)^{1/p},$$

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

$$\|u\|_{W_p^k(\Omega)} := \left(\sum_{j=0}^{k} |u|_{W_p^j(\Omega)}^p \right)^{1/p}.$$

An important special case corresponds to taking $p = 2$, the space $W_2^k(\Omega)$ is then a Hilbert space with inner product

$$(u, v)_{W_2^k(\Omega)} := \sum_{|\alpha|\le k} (D^\alpha u, D^\alpha v).$$

We shall usually write $H^k(\Omega)$ instead of $W_2^k(\Omega)$. The definition of $W_p^k(\Omega)$ and its norm and semi norm, for $p = 2, k = 1$, give:

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

$$\|u\|_{H^1(\Omega)} := \left(\|u\|^2_{L_2(\Omega)} + \sum_{j=1}^n \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2},$$

$$|u|_{H^1(\Omega)} := \left(\sum_{j=1}^n \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2}.$$

For $p = 2, k = 2$, we have

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

$$\|u\|_{H^2(\Omega)} := \left(\|u\|^2_{L_2(\Omega)} + \sum_{j=1}^n \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L_2(\Omega)} + \sum_{i,j=1}^n \left\| \frac{\partial^2 u}{\partial x_i \partial x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2},$$

$$|u|_{H^2(\Omega)} := \left(\sum_{i,j=1}^n \left\| \frac{\partial^2 u}{\partial x_i \partial x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2}.$$

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 $\partial\Omega$, it holds that

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

Definition 1 *Normed linear space B equipped with a norm $\|\cdot\|$ is called a Banach space, if it is complete with respect to the metric*

$$\rho(u, v) = \|u - v\| \quad \forall u, v \in B.$$

Definition 2 *Normed linear space H is called a Hilbert space if it is complete respect to the norm given by a product (u, v):*

$$\|u\| = \sqrt{(u, u)} \quad \forall u, v \in H.$$

1.2 Some Results from the Finite Element Methods

In this section we will deal with the topics connected with the finite element methods (FEM). We start with an existence and uniqueness of a solution of the elliptic problem boundary value problem. To be consistent with our model considered in the next sections, we let $\Omega = (0,1) \times (0,1)$ be a computational domain and $\mathcal{V} = H_0^1(\Omega)$. We consider a variational formulation of an elliptic problem:

$$\text{find } u \in \mathcal{V} : \mathcal{A}(u,v) = l(v) \quad \forall v \in \mathcal{V}. \tag{1.1}$$

In what follows, $\langle \cdot, \cdot \rangle$ denotes the Euclidean (l_2) inner product, in the relevant vector space, $\mathcal{A}(\cdot,\cdot) = (\Delta \cdot, \Delta \cdot)$ is a bilinear form and $l : \mathcal{V} \to \mathbb{R}$ is a linear form. Here we need some properties of $\mathcal{A}$ and l.

Definition 3 *The linear form $l : \mathcal{V} \to \mathbb{R}$ is called continuous or bounded if*

$$\exists C > 0 : |l(v)| \leq C\|v\| \quad \forall v \in \mathcal{V}$$

Definition 4 *The bilinear form $\mathcal{A} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is called continuous or bounded if*

$$\exists C_a > 0 : |\mathcal{A}(v,w)| \leq C_a\|v\|\|w\| \quad \forall v, w \in \mathcal{V}$$

Definition 5 *The bilinear form $\mathcal{A} : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is called $\mathcal{V}$-elliptic if*

$$\exists \alpha > 0 : |\mathcal{A}(v,v)| \geq \alpha\|v\|^2 \quad \forall v \in \mathcal{V}$$

Theorem 1 (Banach fixed point theorem) *Let $\mathcal{V}$ be a Banach space (a complete vector space not necessarily having an inner product) and let $\phi : \mathcal{V} \to \mathcal{V}$ be a contraction, i.e.*

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

There exists a unique $u \in \mathcal{V}$ such that

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

Theorem 2 (Riesz representation theorem) *Let $\mathcal{V}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$. Any element $w \in \mathcal{V}$ defines a continuous linear form $l \in \mathcal{V}'$ by $l := \langle w, v \rangle$. On the other hand, for any continuous linear form $l \in \mathcal{V}'$ there exists a unique element $Rl \in \mathcal{V}$ such that*

$$l(v) = \langle Rl, v \rangle \quad \forall v \in \mathcal{V}$$

Moreover, there holds $\|Rl\| = \|l\|_{\mathcal{V}'}$, i.e.

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

Theorem 3 (Lax-Milgram theorem) *$\mathcal{V}$ is a Hilbert space with inner product $\langle \cdot, \cdot \rangle$ and norm $\| \cdot \|$, $\mathcal{A}(\cdot, \cdot)$ is continuous, $\mathcal{V}$-elliptic bilinear form and $l : \mathcal{V} \to \mathbb{R}$ is continuous linear form. Then the variational problem has a unique solution $u \in \mathcal{V}$.*

Now suppose that $\mathcal{V}_h$ is a finite-dimensional subspace of $\mathcal{V}$. The dimension of the finite element space will be denoted as n. The finite element approximation of 1.1 is:

$$\text{find } u_h \in \mathcal{V}_h : \quad \mathcal{A}(u_h, v_h) = l(v_h) \quad \forall v_h \in \mathcal{V}_h. \tag{1.2}$$

Theorem 4 (Céa's lemma) *The finite element approximation u_h to $u \in H_0^1(\Omega)$, the weak solution to the problem 1.1, is the near-best fit to u:*

$$\|u - u_h\|_{H^1(\Omega)} \leq \frac{C_a}{\alpha} \min_{v_h \in \mathcal{V}_h} \|u - v_h\|_{H^1(\Omega)}. \tag{1.3}$$

Let us define

$$(v, w)_{\mathcal{A}} = \mathcal{A}(v, w) \quad \forall v, w \in H^1(\Omega)_0 \tag{1.4}$$

and let $\| \cdot \|_{\mathcal{A}}$ denote the associated **energy norm** defined by:

$$\|v\|_{\mathcal{A}} = (\mathcal{A}(v, v))^{1/2}.$$

The Céa's lemma can be written in the form

$$\|u - u_h\|_{\mathcal{A}} = \min_{v_h \in \mathcal{V}_h} \|u - v_h\|_{\mathcal{A}}. \tag{1.5}$$

Assume A is a symmetric, positive definite matrix. We use the symbols $\langle \cdot, \cdot \rangle_A$ and $\|\cdot\|_A$ for the usual $A-$inner product $\langle \cdot, \cdot \rangle_A = \langle A \cdot, \cdot \rangle$ and A-norm $\|\cdot\|_A = \langle \cdot, \cdot \rangle_A^{1/2}$.

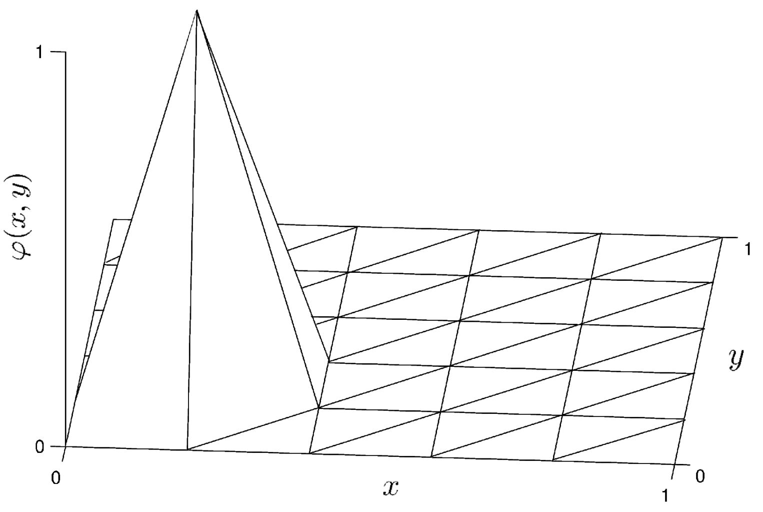


Figure 1.1: The basis function φ_i

Finally we consider the piece-wise linear finite element approximation to the problem 1.1. The space $\mathcal{V}_h = span\{\varphi_i\}_{i=1}^n$ is generated by piece-wise linear basis functions $\varphi_i, \mathcal{V}_h$ (see fig. 1.1). The details of the construction of FEM space can be found in [16] and [47].

1.3 Some Other Results

Let $\mathcal{I}u \in \mathcal{V}_h$ denote the continuous piece-wise linear function which coincides with u at the mesh points x_i, $i = 0, \dots, n$,

$$\mathcal{I}u(x) = \sum_{i=1}^{n-1} u(x_i)\varphi_i(x).$$

The function $\mathcal{I}u(x)$ is called the interpolant of u from $\mathcal{V}_h$. If $u \in H^2(\Omega)$, then there exist a constant C_1 so that

$$\|u - \mathcal{I}u\|_{H^1(\Omega)} \leq C_1 h^2 \|D^2 u\|_L^2(\Omega).$$

Let $\mathcal{M}$ be an index set. The notation $\langle\cdot,\cdot\rangle_{l_2(\mathcal{M})}$ and $\|\cdot\|_{l_2(\mathcal{M})}$ corresponds to the Euclidean inner product and the norm on a discrete domain $\mathcal{M}$ defined as

$$\langle \mathbf{x}, \mathbf{y}\rangle_{l_2(\mathcal{M})} = \sum_{i\in\mathcal{M}} x_i y_i, \quad \|\cdot\|_{l_2(\mathcal{I})} = \langle\cdot,\cdot\rangle_{l_2(\mathcal{M})}^{1/2},$$

respectively. Here, $\mathbf{x}$ and $\mathbf{y}$ are vectors such that their entries $x_i, y_i \in \mathbb{R}$ are defined for all $i \in \mathcal{M}$. On $\mathbb{R}^N$, $\{1, \dots, n\} \supset \mathcal{M}$, $\langle\cdot,\cdot\rangle_{l_2(\mathcal{M})}$ is a semi-product and $\|\cdot\|_{l_2(\mathcal{M})}$ is a semi-norm.

Assume $(U, \|\cdot\|_U)$ and $(V, \|\cdot\|_V)$ are Banach spaces and $L : U \to V$ a linear mapping. We introduce the operator norm of L by

$$\|L\|_{U\to V} = \sup_{u\in U\setminus\{0\}} \frac{\|Lu\|_V}{\|u\|_U}.$$

For a symmetric, positive definite (SPD) matrix A we define a condition number

$$\text{cond}(A) = \lambda_{max}(A)/\lambda_{min}(A).$$

Similarly, for symmetric positive definite matrices A, B, the mutual condition number is given by $\text{cond}(A, B) = \lambda_{max}(BA)/\lambda_{min}(BA)$.

We use generic constants $C, c > 0$ in the usual way, for example, for $\|\mathbf{u}\| \leq C\|\mathbf{v}\|$ and $\|\mathbf{v}\| \leq C\|\mathbf{w}\|$ we write $\|\mathbf{u}\| \leq C\|\mathbf{w}\|$. Typically, C, c are constants independent of the finest level mesh-size and whenever relevant, also on the level and the number of levels.

Theorem 5 (Spectral Equivalence) *Let A and B be two SPD matrices* $(A, B \in \mathbb{R}^{n\times n})$*. Matrices A and B are spectrally equivalent if for* $\alpha_1, \alpha_2 > 0$

$$\alpha_1(Bv, v) \leq (Av, v) \leq \alpha_2(Bv, v), \quad \forall v \in \mathbb{R}^n.$$

If two matrices A and B are spectrally equivalent, then

$$\alpha_1 \leq \frac{(B^{-1}Av, v)_B}{(v, v)_B} \leq \alpha_2$$

and $\alpha_1 \leq \lambda_{min}(B^{-1}A)$, $\lambda_{max}(B^{-1}A) \leq \alpha_2$ and then $cond(B^{-1}A) \leq \frac{\alpha_2}{\alpha_1}$.

Chapter 2

Geometric Multigrid Methods

The purpose of the next chapter is to cover the main ideas of the geometric multigrid. We think that it is useful to start with the basics and follow the steps of the historical development of the method. This chapter is divided into two sections. In the first section, we will start with the iterative methods. We define main iterative methods which are used in multigrid and describe their smoothing properties. Then, in the second section, the two-grid method will follow and then we cover the multigrid by subspace splitting.

2.1 Iterative Methods

Iterative methods play a fundamental role in multigrid as *smoothers*. It has become common in the introductory sections of multigrid-based papers to show an effect of geometrical smoothing on the simple example. We will do no different, since it is nicely shows the smoothing behaviour to the reader.

Let V be a finite dimensional vector space. We will study iterative methods to solve a linear system

$$A\mathbf{u} = \mathbf{f}, \tag{2.1}$$

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

The basic idea of a single step linear iterative methods is to approximate $\mathbf{u}$ by the old solution $\mathbf{u}^{\text{old}}$ and use $\mathbf{u}^{\text{old}}$ to get $\mathbf{u}^{\text{new}}$ by solving the residual equation for an error $\mathbf{e}$. The algorithm may look like this:

Algorithm 2.1.1

Calculate $\mathbf{u}^{\text{new}}$:

```
1: calculate a residual:   $\mathbf{r} = \mathbf{f} - A\mathbf{u}^{\text{old}}$
2: replace   $A\mathbf{e} = \mathbf{r}$   by   $\bar{\mathbf{e}} = B\mathbf{r}$ and solve for $\bar{\mathbf{e}}$
3: update:   $\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + \bar{\mathbf{e}}$
```

Here, B is an approximate inverse of A and a linear operator on V again. We can use this to get Algorithm 2.1.2.

Algorithm 2.1.2

Calculate $\mathbf{u}^{k+1}$:

```
1: given:   $\mathbf{u}^0 \in V$
2: for $k = 0, 1, 2, \dots$ do
3:     calculate:   $\mathbf{u}^{k+1} = \mathbf{u}^k + B(\mathbf{f} - A\mathbf{u}^k)$
4: end for
```

We might also use the form of Algorithm 2.1.2 such that

$$\mathbf{u}^{k+1} = E\mathbf{u}^k + \mathbf{s}, \quad k = 0, 1, 2, \dots, \tag{2.2}$$

where

$$E = I - BA, \quad \mathbf{s} = B\mathbf{f}.$$

Operator B plays the main role in the algorithm. If $B = A^{-1}$, then the first iteration $\mathbf{u}^1$ is the exact solution.

Let us assume a sequence of iterations

$$\mathbf{u}^k, \quad k = 0, 1, 2 \dots \tag{2.3}$$

and $\mathbf{u}^*$ solution to 2.1 . Iteration scheme 2.2 is called *consistent* if it satisfies

$$\mathbf{u}^* = E\mathbf{u}^* + \mathbf{s}. \tag{2.4}$$

If $I - E$ is non-singular, then there is a solution $\mathbf{u}^*$ to the equation (2.4). It gives

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

Theorem 6 *Let 2.2 be a consistent scheme and E be a square matrix such that $\rho(E) < 1$. Then $I - E$ is nonsingular and the iteration (2.3) converges for any $\mathbf{f}$ and $\mathbf{u}^0$. Conversely, if the iteration (2.3) converges for any $\mathbf{f}$ and $\mathbf{u}^0$, 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

$$\mathbf{e}^k = E^k\mathbf{e}^0 \to 0.$$

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 non-singular and the iteration in Algorithm (2.1.1) converges for any initial vector $\mathbf{u}^0$.*

2.1.1 Richardson Iterative Method

The first method we describe is the Richardson method. The matrix B from Algorithm (2.1.1) is given by

$$B = \frac{\omega}{\rho(A)} I, \quad \omega > 0,$$

therefore

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

Let us consider a short example. Just for now, let A be a stiffness matrix given by finite elements discretization of the elliptic problem with zero Dirichlet boundary conditions.

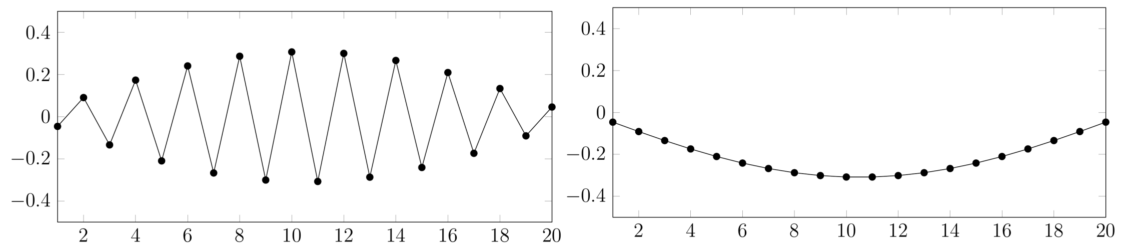
The error of approximation $\mathbf{u}^{k+1}$ is

$$\mathbf{e}^{k+1} = \left(I - \frac{\omega}{\rho(A)} A\right) \mathbf{e}^k.$$

Now, we would like to represent the error $\mathbf{e}^k$ by eigenvectors of matrix A,

$$\mathbf{e}^k = \sum_i \alpha_i \phi_i, \quad A\phi_i = \lambda_i \phi_i,$$

where ϕ_i is an eigenvector corresponding to eigenvalue λ_i. In Fig. 2.1, we see that eigenvector corresponding to the largest eigenvalue is of high frequency, while eigenvector corresponding to the smallest eigenvalue is smooth.



(a) Eigenvector ϕ_i corresponding to $\lambda_i = \rho(A)$

(b) Eigenvector ϕ_i corresponding to the smallest λ_i

Figure 2.1: High and low frequency modes

It holds that

$$\left(I - \frac{\omega}{\rho(A)} A\right) \mathbf{e}^k = \sum_i \alpha_i \left(1 - \omega \frac{\lambda_i}{\rho(A)}\right) \phi_i.$$

For a fixed $\omega = 1$, it is clear that the high frequency modes of the error corresponding to $\lambda_i \approx \rho(A)$ are eliminated fast, while low frequency modes of the error corresponding to $\lambda_i \approx 0$ are eliminated slow.

The smoothing effect of the iterative methods is shown in Fig. 2.2. It shows an initial residual on the left and first iterations on the right. The result is a geometrically smooth graph of the residual. We can also see that the iterative method as a smoother is most effective in the first couple of iterations and then its smoothing efficiency weakens. This is the reason why in multigrid we use just a few smoothing iterations.

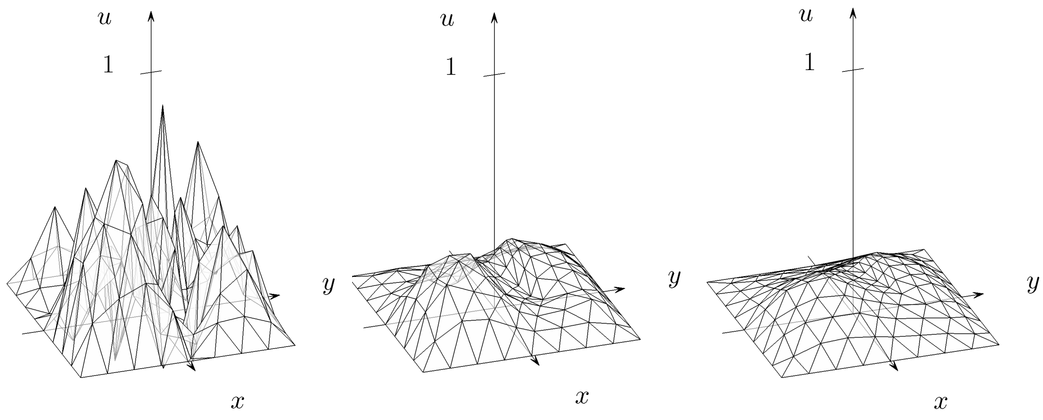


Figure 2.2: Smoothing effect of iterative methods

2.1.2 Gauss-Seidel and SOR Methods

Let $V = \mathbb{R}^n$ and let $A \in \mathbb{R}^{n\times n}$ now be a symmetric positive definite matrix.

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

$$A = D - L - U,$$

we get

$$(D - L)\mathbf{u} = U\mathbf{u} + \mathbf{f}$$

and the iterates are

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

If we solve each equation for u_i, i-th element of $\mathbf{u}$, and use it to update u_{i+1}, the Gauss-Seidel is then of the form

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

The successive over relaxation (SOR) comes from the Gauss-Seidel method and, for $\omega \in (0, 1)$, we get a linear combination of $k - th$ iteration from a Gauss-Seidel method and previous $k - 1th$ itereration,

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

2.2 Introduction to Multigrid

As we have seen in the previous section, relaxation schemes damp the oscillatory modes of the error effectively, but smooth modes are damped slowly. The second main part of multigrid is the residual correction idea, which means that we find a correction of $\mathbf{u}$ in the coarse space by solving the residual equation $A\mathbf{e} = \mathbf{r}$ on the coarse level.

We will start with a two level algorithm which interacts between two grids. Let us denote the fine grid by Ω^h and the coarse grid by Ω^H. The processes which provide the interplay between grids are called a *restriction* and a *prolongation* (or an *interpolation*). The idea of the two-level method is outlined in Algorithm 2.2.1:

Algorithm 2.2.1 Two-grid method (TGM):

```
1: relax on fine mesh:   $\mathbf{u}_h \leftarrow \mathcal{S}^\nu(\mathbf{u}_h, \mathbf{f}_h)$          $\triangleright$ on $\Omega^h$
2: compute residual:   $\mathbf{r}_h = \mathbf{f}_h - A_h\mathbf{u}_h$
3: restrict residual:   $\mathbf{r}_H = I_h^H\mathbf{r}_h$
4: relax on the coarse mesh to get an error on the coarse mesh:
                 $A_H\mathbf{v}_H = \mathbf{r}_H$          $\triangleright$ on $\Omega^H$
5: correct the approximation on the fine mesh:   $\mathbf{u}_h \leftarrow \mathbf{u}_h + I_H^h\mathbf{v}_H$
6: postsmooth:   $\mathbf{u}_h \leftarrow \mathcal{S}^\nu(\mathbf{u}_h, \mathbf{f}_h)$          $\triangleright$ on $\Omega^h$
```

The notation $\mathbf{u}_h \leftarrow \mathcal{S}^\nu(\mathbf{u}_h, \mathbf{f}_h)$ means that $\mathbf{u}_h$ results from a smoothing of $\mathbf{u}_h$ given by ν smoothing steps of $\mathcal{S}$.

We have to define mappings from the coarse to the fine grid. Let us make an example in the 1-D case:

We define a mapping

$$I_H^h : \Omega^H \to \Omega^h.$$

If $\mathbf{u}_h, \mathbf{u}_H$ are defined on Ω_h, Ω_{2h}, then

$$I_H^h\mathbf{u}_H = \mathbf{u}_h,$$

where I_H^h is a prolongator (interpolation operator) if

$$\begin{cases} u_{2j}^h = u_j^H, \\ u_{2j+1}^h = (u_j^H + u_{j+1}^H)/2, \end{cases} \quad \text{for} \quad 0 \leq j \leq \frac{n+1}{2}.$$

(In this notation, the grid index is a bottom index). In a matrix form, this reads

$$\mathbf{u}_h = \frac{1}{2}\begin{bmatrix} 1 & & \\ 2 & & \\ 1 & 1 & \\ & 2 & \\ & 1 & 1 \\ & & \vdots \\ & & 1 \\ & & 2 \\ & & 1 \end{bmatrix}\mathbf{u}_H.$$

The restriction operator $I_h^H : \Omega^h \to \Omega^H$ can be then taken as

$$I_h^H \mathbf{u}_h = \mathbf{u}_H.$$

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

$$u_j^H = \frac{1}{4}(u_{2j-1}^h + 2u_{2j}^h + u_{2j+1}^h).$$

In a matrix form, this reads

$$\mathbf{u}_H = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & 1 & 2 & 1 & & \dots \\ & & & & & & 1 & 2 & 1 \end{bmatrix}\mathbf{u}_h.$$

In the notation of 2.2, we could look at the algorithm as at the iterative process

$$\mathbf{u}_h^{k+1} = M_h \mathbf{u}_h^k + \mathbf{g}_{M_h}.$$

The smoothing process $\mathcal{S}$ is also an iterative method, we can therefore put it into this form

$$\mathbf{u}_h^{k+1} = S\mathbf{u}_h^k + \mathbf{g}_h,$$

where S is the iteration matrix associated with $\mathcal{S}$.

We take $\mathbf{g}_h = 0$ to get

$$\mathbf{u}_h^{k+1} = S\mathbf{u}_h^k,$$

and the residual

$$\mathbf{r}_H = I_h^H(-A_h S \mathbf{u}_h).$$

The algorithm becomes

$$\mathbf{u}_h^{k+1} = S(S\mathbf{u}_h^k + I_H^h A_H^{-1} I_h^H(-A_h S\mathbf{u}_h^k)),$$

therefore

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

The matrix inside the brackets,

$$T = I - I_H^h A_H^{-1} I_h^H A_h, \tag{2.5}$$

is the coarse grid correction operator.

Lemma 2 *When the coarse grid matrix is defined as $A_H = I_h^H A_h I_H^h$, then the coarse grid operator (2.5) is a projector with respect to the A_h - inner product.*

Proof:

Let us show that $I - T = I_H^h A_H^{-1} I_h^H A_h$ is a projector.

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

The adjoint of $I_H^h A_H^{-1} I_h^H A_h$ in a A_h-inner product is

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

T is therefore self-adjoint and $I_H^h\ A_H^{-1} I_h^H A_h$ and A an A-orthogonal projector. We will denote $G = I_H^h\ A_H^{-1} I_h^H A_h$.

Another way to derive the two-grid method

Before we proceed to multigrid section, we would like to show another way to derive the two level method, which is presented in [24]. Instead of previous geometric properties of the iterative methods, it is connected with the algebraic approach to the smooth/fine parts of the solution.

We will now omit the notation of h and H for the prolongator and assume two real vector spaces $\mathbb{R}^m$ and $\mathbb{R}^n, m < n$. We again assume A is a symmetric positive definite matrix of order n and $\mathbf{f} \in \mathbb{R}^n$. We assume now that an injective linear *prolongator* $P : \mathbb{R}^m \to \mathbb{R}^n,\ m < n$ is given.

The two-level method consists in the combination of a *coarse-grid correction* and *smoothing.* The coarse-grid correction is derived by correcting an error $\mathbf{e}$ by a coarse-level vector $\mathbf{v}$ so that the resulting error $\mathbf{e} - P\mathbf{v}$ is minimal in A-norm. In other words, we solve the minimization problem

$$\text{find } \mathbf{v} \in \mathbb{R}^m \text{ so that } \|\mathbf{e} - P\mathbf{v}\|_A \text{ is minimal.} \tag{2.6}$$

The vector $P\mathbf{v}$ is an A-orthogonal projection of the error $\mathbf{e}$ onto Range(P), with the projection operator given by

$$G = P(P^TAP)^{-1}P^TA.$$

From the fundamental theorem of linear algebra, we know that for a matrix A, the space $\mathbb{R}^n$ can be orthogonally decomposed [13]:

$$\mathbb{R}^n = \text{ Range}(A) \oplus \text{ Ker}(A^T).$$

This leads to the following lemma:

Lemma 3 *Operator G is an A-orthogonal projection on the* Range(P) *and T is A-orthogonal projection on the space*

$$\text{Ker}(P^TA) = \{\mathbf{x} \in \mathbb{R}^n : (A\mathbf{x}, \mathbf{w}) = 0 \quad , \forall \mathbf{w} \in \text{ Range}(P)\}.$$

If Ω^h is a space of fine-grid vectors, we can write

$$\Omega^h = \text{ Range}(P) \oplus \text{ Ker}(P^TA)$$

For a vector $\mathbf{e} \in \Omega^h$ it means that it can be written as

$$\mathbf{e} = \mathbf{e}_s + \mathbf{e}_o,$$

where $\mathbf{e}_s \in$ Range(P) and $\mathbf{e}_o \in$ Ker(P^TA). Vector $\mathbf{e}_s$ can be understood as a smooth component of $\mathbf{e}$ and $\mathbf{e}_o$ as the oscillatory component, see [13].

Now, the *error propagation operator* E is an operator such that

$$\mathbf{e}^{k+1} = E\mathbf{e}^k.$$

Thus, the error propagation operator of the coarse-grid correction is given by $I - G = I - P(P^TAP)^{-1}P^TA$ and the error propagation operator of the two-level method by

$$E_{TM} = S_{post}[I - P(P^TAP)^{-1}P^TA]S_{pre}, \tag{2.7}$$

where S_{pre} and S_{post} are error propagation operators of pre- and post- smothing iterations, respectively.

The coarse-grid correction can be algorithmized as

$$\mathbf{u} \leftarrow \mathbf{u} - P(P^TAP)^{-1}P^T(A\mathbf{u} - \mathbf{f})$$

and the variational two-level algorithm with post-smoothing step proceeds as follows:

Algorithm 2.2.2

```
1: pre-smooth:   $\mathbf{u} \leftarrow \mathcal{S}_{pre}(\mathbf{u}, \mathbf{f})$,
2: evaluate the residual:   $\mathbf{r} = A\mathbf{u} - \mathbf{f}$
3: restrict the residual:   $\mathbf{r}_2 = P^T\mathbf{r}$
4: solve a coarse-level problem:   $A_2\mathbf{v} = r_2, \quad A_2 = P^TAP$
5: correct the approximation:   $\mathbf{u} = \mathbf{u} - P\mathbf{v}$
6: post-smooth:   $\mathbf{u} \leftarrow \mathcal{S}_{post}(\mathbf{u}, \mathbf{f})$
```

The coarse-grid correction vector $\mathbf{v}$ is chosen to minimize the error after step 5 of Algorithm 2.2.2. Thus, in the case of a standard multigrid, the coarse-grid correction procedure minimizes the error in an intermediate stage of the iteration. In following sections, we will show that if we minimize the error after coarse-grid correction with subsequent smoothing, we get the smoothed aggregation method.

2.3 The Multigid Cycle

Let us first consider a set of grids defined on the vector spaces V_l, where the total number of grids is L.

The multigrid cycle can be derived from the two-grid cycle. In the two-grid method, we observe that we don't have to solve the coarse level equation exactly, since it is also an approximation. We may replace $\mathbf{v}_H$ by a suitable approximation. We can apply the two grid idea again with even coarser grid. This process can be repeated until all L levels are involved, as we see in Fig. 2.3. Level 1 corresponds to the coarsest level.

Remark 7 We would like to mention that in algebraic multigrid, it is a habit to denote the coarsest level as L and in the AMG section, we will stick to AMG notation.

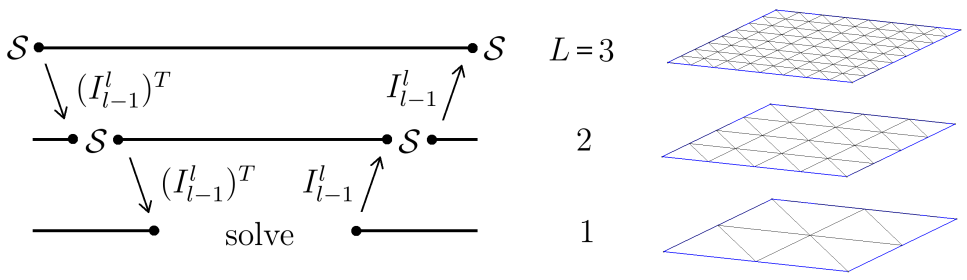


Figure 2.3: Scheme of geometric multigrid

To define the multigrid algorithm, we need the system of linear prolongators

$$I_{l-1}^{l} : \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l},\ n_L = n,\ n_{l-1} < n_l,\ l = 1, \dots, L-1$$

and the smoothing iterative procedures $\mathcal{S}_l(\cdot,\cdot) : \mathbb{R}^{n_l} \times \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$ on all levels $l = L, \dots 2$. Then, the multigrid algorithm is defined as follows:

Given the system 2.1, the prolongators I_{l-1}^{l}, the smoothers $\mathcal{S}_l(\cdot,\cdot)$, $l = L, \dots 2$, the right-hand side $\mathbf{f} \in \mathbb{R}^n$ and parameter $\nu, \gamma > 0$, set $A_L = A$, $A_{l-1} = (I_{l-1}^{l})^T A_l I_{l-1}^{l}$, $l = L, \dots 2$ and $\mathbf{f}^1 = \mathbf{f}$. For a given input iterate $\mathbf{x} \in \mathbb{R}^n$, perform the iteration $\mathbf{x} \leftarrow MG(\mathbf{x}, \mathbf{f})$ given by Algorithm 2.3.1.

Algorithm 2.3.1 MG($\mathbf{x}, \mathbf{f}$)

```
1:  for l=$L$:2 do
2:      pre-smoothing: perform $\nu$ iterations of $\mathbf{x}^l \leftarrow \mathcal{S}_l(\mathbf{x}^l, \mathbf{f}^l)$
3:      coarse-level correction: set $\mathbf{f}^{l-1} = (I_{l-1}^{l})^T(\mathbf{f}^l - A_l\mathbf{x}^l)$
4:      if $l = 2$ then
5:          solve directly: $A_{l-1}\mathbf{x}^{l-1} = \mathbf{f}^{l-1}$
6:      else
7:          set $\mathbf{x}^{l-1} = \mathbf{0}$ and perform $\gamma$ iterations of $\mathbf{x}^{l-1} \leftarrow MG(\mathbf{x}^{l-1}, \mathbf{f}^{l-1})$
8:      end if
9:      correct the approximation on the level $l$ by $\mathbf{x}^l \leftarrow \mathbf{x}^l + I_{l-1}^{l}\mathbf{x}^{l-1}$
10:     post-smoothing: perform $\nu$ iterations of $\mathbf{x}^l \leftarrow \mathcal{S}_l(\mathbf{f}^l, \mathbf{x}^l)$
11: end for
```

Some special choices of L and γ will give us well known types of algorithms. For $L = 2$ algorithm reduces to two-level method, $\gamma = 1$ gives us so called $V-cycle$ and $\gamma \geq 2$ $W-cycle$.

We will be interested especially in the $V-cycle$ and if do not say otherwise, the multigrid method will be referred to as $V-cycle$.

Let $\gamma = 1$, we compare the error propagation operators of MG and the two grid method. The $V-cycle$ error propagation operator E_l is gives by

$$E_0 = 0,$$

$$E_l = S_l^{\nu}(I_l - I_{l-1}^{l}(I_{l-1} - E_{l-1})A_{l-1}^{-1}I_{l-1}^{l}A_l)S_k^{\nu}.$$

The difference between the two-grid operation

$$E_l = S_l^{\nu}(I_l - I_{l-1}^{l}A_{l-1}^{-1}I_l^{l-1}A_l)S_l^{\nu}$$

and the above iteration operator M_l is that A_{l-1}^{-1} is replaced by

$$I_{l-1} - E_{l-1}A_{l-1}^{-1}.$$

2.3.1 Multigrid by Subspace Splitting

In this section, we will look at multigrid in a standard variational setting. We will see that multigrid in general can be seen as a method of a Schwarz type, both additional and multiplicative. In the later sections, we will be interested in the additive one. This topic was studied by Xu in [60], [63] and [61].

We follow Xu and start first with an abstract theory considering problem

$$\mathcal{A}(u,v) = (f,v), \quad \forall v \in \mathcal{V}_h,$$

given a finite dimensional Hilbert space $\mathcal{V}_h$. We consider the decomposition of $\mathcal{V}_h$:

$$\mathcal{V}_h = \sum_{i=1}^{L} \mathcal{V}_i, \tag{2.8}$$

where the finite dimensional subspaces satisfy

$$\mathcal{V}_1 \subset \mathcal{V}_2 \subset, \ldots, \subset \mathcal{V}_h.$$

We define orthogonal projections:

$$\mathcal{P}_l, \mathcal{Q}_l : \mathcal{V}_h \to \mathcal{V}_l$$

by

$$(\mathcal{P}_l u, v_l)_{\mathcal{A}} = (u, v_l)_{\mathcal{A}}, \quad \forall u \in \mathcal{V}_h, v_l \in \mathcal{V}_l,$$

and

$$(\mathcal{Q}_l u, v_l) = (u, v_l), \quad \forall u \in \mathcal{V}_h, v_l \in \mathcal{V}_l.$$

We also define $\mathcal{A}_l : \mathcal{V}_l \to \mathcal{V}_l$ as the restriction of $\mathcal{A}$ on $\mathcal{V}_l$

$$\mathcal{A}(u_l, v_l) = (\mathcal{A}_l u_l, v_l), \quad \forall u_l, v_l \in \mathcal{V}_l. \tag{2.9}$$

It follows that

$$\mathcal{A}_l \mathcal{P}_l = \mathcal{Q}_l \mathcal{A}. \tag{2.10}$$

We will consider two approaches - parallel and successive subspace correction algorithms. The parallel subspace correction (PSC) leads to the so called BPX or MDS additive algorithms and successive subspace correction (SSC) to the multiplicative multigrid.

Parallel subspace correction

We try to solve an equation on each subspace $\mathcal{V}_l$:

$$\mathcal{A}_l e_l = \mathcal{Q}_l r.$$

We, again, solve the equation approximately:

$$\bar{e}_l = \mathcal{R}_l \mathcal{Q}_l r,$$

where $\mathcal{R}_l : \mathcal{V}_l \to \mathcal{V}_l$ is a subspace solver.

Definition 6 (additive preconditioner) *The operator $\mathcal{B}$ from 2.1.2 is then given by*

$$\mathcal{B} = \sum_{i=1}^{L} \mathcal{R}_i \mathcal{Q}_i. \tag{2.11}$$

The additive preconditioner is used in following algorithm:

Algorithm 2.3.2 Additive Schwarz Method/Parallel Subspace Correction

```
given: $u^k \in \mathcal{V}_h$
1: for $l = 1 : L - 1$ do
2:     restrict: $r_l = \mathcal{Q}_l(f - Au^k)$
3:     solve:    $\bar{e}_l = \mathcal{R}_l r_l$
4: end for
5: correct on each subspace:    $u^{k+1} = u^k + \sum_{l=1}^{L} \bar{e}_l$
```

Algorithm 2.3.2 leads to a type of additive multigrid. There exist variants of additive multigrid, for example MDS method [65] or BPX method [6] which will be discussed later.

Succesive subspace correction

The succesive subspace method is given by a multiplicative preconditioner:

Definition 7 (multiplicative preconditioner) *The multiplicative preconditioner $\mathcal{B}$ based on subspaces $\mathcal{V}_l$ is defined as*

$$\mathcal{B} = (I - E)^{-1}, \tag{2.12}$$

where

$$E = (I - T_L) \dots (I - T_1) \quad \textit{and} \quad T_l = \mathcal{R}_l \mathcal{Q}_l \mathcal{A}.$$

The multiplicative preconditioner results in the following method:

Algorithm 2.3.3 Multiplicative Schwarz Method/Successive Subspace Correction

```
1: set $u_0^k = u^k \in \mathcal{V}_h$
2: for $l = 1 : L$ do
3:     $\bar{e}_l = \mathcal{R}_l \mathcal{Q}_l (f - \mathcal{A} u_{l-1}^k)$
4:     $u_l^k = u_{l-1}^k + \bar{e}_l$
5: end for
6: $u^{k+1} = u_L^k$
```

Here, we can see that classical $V - cycle$ can be seen as the SSC method.

Matrix representations of SSC and PSC methods

Let $\mathcal{I}_l : \mathcal{V}_l \to \mathcal{V}_h$ be a natural inclusion and a unique matrix $I_l \in \mathbb{R}^{n \times n_l}$, which is a matrix representation of $\mathcal{I}_l$. For each l, we assume that $\{\varphi_1^l, \ldots, \varphi_{n_l}^l\}$ is a basis of $\mathcal{V}_l$. Since $\mathcal{V}_l \subset \mathcal{V}_h$, every basis function of $\mathcal{V}_l$ can be represented in terms of the basis of $\mathcal{V}_h$ so that

$$(\varphi_1^l, \ldots, \varphi_{n_l}^l) = (\varphi_1, \ldots, \varphi_n) I_l.$$

For every $v_l = \sum_{i=1}^{n_l} c_{l_i} \varphi_{l_i}$, we define a mapping $\pi_l : \mathbb{R}^{n_l} \to \mathcal{V}_l$ as

$$\pi_l c_l := \sum_{i=1}^{n_l} c_{l_i} \varphi_{l_i}$$

and

$$\pi_l^T v_l = ((v_l, \varphi_{l_1}), \ldots, (v_l, \varphi_{l_{n_l}}))^T.$$

We also define

$$G = \{(\varphi_i, \varphi_j)\}_{i,j=1}^n,$$
$$G_l = \{(\varphi_i^l, \varphi_j^l)\}_{i,j=1}^{n_l}.$$

Then L_2 projector $\mathcal{Q}_l$ can be given as

$$\mathcal{Q}_l = \pi_l G_l^{-1} \pi_l^T.$$

Matrix representation of $\mathcal{Q}_l$ is given by

$$Q_l = I_l (I_l^T I_l)_l^{-1} I_l^T. \tag{2.13}$$

If R_l is a matrix representation of $\mathcal{R}_l$, then matrix formulation of the preconditioner $\mathcal{B}$ is

$$B = \sum_{l=0}^{J} I_l R_l I_l^T. \tag{2.14}$$

Remark 8 The difference between PSA and SSC is that in SSC, the residual is updated successively on each level. In PSC, the subspace corrections uses restrictions from the finest level and the subspace corrections can be done separately on each level. The SSC has much better convergence rates, see [40], but PSC is attractive due to it's natural parallelism. In practice, additive algorithms are used as a preconditioners for conjugated gradient method.

2.3.2 Convergence Estimates

In this section, we focus on some important convergence estimates for multigrid algorithm. To show the convergence of the multigrid algorithm, we have to show that for the multigrid error operator E_l,

$$\|E_l\|_A \leq \delta$$

holds with $\delta < 1$.

There exist different approaches to proof that E_l is a contraction. The assumptions and convergence proofs differ for two-level method, $W-cycle$ and $V-cycle$.

The first proof of the multigrid convergence was a work of Braess and Hackbusch [5]. Proofs based on Braess/Hackbusch approach need a certain form of regularity as a assumption and use the concept of smoothing and approximation property. Results coming from this framework variate whether we can provide full or partial regularity and one or more smoothing steps. The proof of convergence, where an assumption of the regularity is not necessary (*regularity free multigrid*) is based on multilevel space splitting and was analyzed by Bramble, Pasciak, Wang and Xu in [6].

The regularity-assuming theorems are based on two properties. The approximation property in a certain sense is a measure of how good the coarse-grid solution is. The efficient smoother has to fulfill the smoothing property.
Let $\mathcal{V}_h$ be a finite dimensional Hilbert space. We follow [4] and introduce two Sobolev spaces V_+, V_- so that

$$\mathcal{V}_h \subset V_+ \hookrightarrow V_-,$$

(which means that the space V_+ is continuously embedded in V_- i.e. for two normed spaces $V_+ \subset V_-$, there exists constant $C \geq 0$ such that for every $x \in V_+$: $\|x\|_{V_-} \leq C\|x\|_{V_+}$).

The Galerkin projection is given by

$$P_h : V_+ \to \mathcal{V}_h.$$

Let α be a positive constant. The approximation property (A1) holds if

$$\|u - P_h u\|_{V_-} \leq ch^{\alpha}\|u\|_{V_+} \quad \forall u \in V_+, . \tag{A1}$$

where $\alpha > 0$ is a positive constant. In other words, the approximation property measures how the coarse-grid solution approximates the fine grid solution.

Now, we introduce a smoother R and smoothing operator $S = I - RA$, which is characterized by the smoothing property (S1):

$$\|S^{\nu} v_h\|_{V_+} \leq \frac{ch^{-\sigma}}{\phi(\nu)}\|v_h\|_{V_-} \quad \forall v_h \in \mathcal{V}_h. \tag{S1}$$

The function $\phi(\nu)$ satisfies $\phi(\nu) \to \infty$ as $\nu \to \infty$. This abstract framework allows for a definition of high frequency elements v_h:

$$\|v_h\|_{V_+} \approx h^{-\alpha}\|v_h\|_{V_-} \tag{2.15}$$

and low frequency elements v_h:

$$\|v_h\|_{V_+} \approx \|v_h\|_{V_-}. \tag{2.16}$$

Let u be a high frequency node as in 2.15, then then inserting u in the smoothing property, smoother S^{ν} is an effective contraction operator.

The effective interplay between approximation and smoothing property means that smoother satisfying smoothing property effectively eliminates the high frequency parts of the error and coarse grid correction eliminates the remaining parts of the error.

As a choice of the two spaces we take $V_+ = H_0^1$, $V = L^2$, $\alpha = 1$ and

$$[V_-, V_+] = [\|\cdot\|, \|\cdot\|_A].$$

Theorem 9 *Assume property (A1) holds, a symmetric smoother R satisfies property (S1) and $\|S^m\| \leq C$ holds. Then the two-grid method converges with sufficient steps ν:*

$$\|E_{TG}\| \leq C\eta(\nu).$$

Proof. Proof can be found in [15]

Self-Adjoint Problem

Let A be a symmetric and positive definite matrix. We also assume that R is a symmetric smoother and satisfies property (R):

$$\|u\|_A \leq (R^{-1}u, u), \quad \forall u \in \mathcal{V}_h. \tag{R}$$

Assume that R is symmetric and $\|S^{\nu}\| \leq C$. The smoothing property (S2) is given as:

$$\|u\| \leq C_S \rho_A (Ru, u), \quad \forall u \in \mathcal{V}_h \tag{S2}$$

Theorem 10 (Braess, Hackbusch) *Consider multigrid error propagation matrix E_l of the $V-cycle$ with a symmetric smoother S. Assume that S satisfies properties (R) and (S2) and approximation property (A1) hold. Then for $C = C_A C_S$*

$$\|E_l\|_A \leq \frac{C}{C + 2\nu}$$

holds.

Proof. Proof is given in [5]

Remark 11 Now, let $\mathcal{A}$ and A be continuous level and finite element discretization operators of problem 1.1. To verify the approximation property, we need an assumption of the regularity such as: given $f \in L_2(\Omega)$ there is a weak solution $u \in H^2(\Omega)$ of $\mathcal{A}u = f$ such that

$$\|u\|_{H^2(\Omega)} \leq C\|f\|_{L^2(\Omega)}.$$

Then it can be shown (e.g. in [15]) that the smoothing and approximation assumptions are satisfied and the multigrid $V - cycle$ has a contraction number smaller than one independent of l (with respect to $\| \cdot \|_A$). The approximation property in L^2 norm is proved by using Aubin-Nitsche duality argument [46] which needs regularity as an assumption.

The convergence proof without the assumption of the regularity was given in [6]. We first set U_l to be a hierarchy of coarse spaces $U_l = \text{Range}(I_1^l)$, where I_1^l is a composite prolongator $I_1^l = I_{l-1}^l \dots I_1^2$.

Theorem 12 (Bramble-Pasciak-Xu-Wang) *Assume there are linear mappings*

$$Q_l : U_L \to U_l,\ l = 1, \dots, L,\ Q_L = I$$

so that

$$\|(Q_l - Q_{l-1})\mathbf{u}\|_l^2 \leq \frac{C_1}{\lambda_l}\|\mathbf{u}\|_A^2 \quad \forall \mathbf{u} \in U_l,\ l = 2, \dots, L$$

and

$$\|Q_l\|_A \leq C_2 \quad \forall l = 1, \dots, L.$$

We also assume smoothers

$$\mathcal{S}_l(\mathbf{x}_l, \mathbf{f}_l) = (I - R_l A_l)\mathbf{x}_l + R_l \mathbf{f}_l,$$

where R_l are symmetric positive definite matrices such that $I - R_l A_l$ are A_l-symmetric positive definite and

$$C_r(R_l \mathbf{u}, \mathbf{u})_{\mathbb{R}^{n_l}} \geq \frac{\|\mathbf{u}\|_{\mathbb{R}^{n_l}}^2}{\rho(A_l)} \quad \forall \mathbf{u} \in \mathbb{R}^{n_l},\ l = 1, \dots, L-1.$$

Then, for the error propagation operator E of the multigrid algorithm holds

$$\|E\|_A \leq 1 - \frac{1}{CL}, \quad C = (1 + C_2^{1/2} + (C_r C_1)^{1/2})^2.$$

Proof. Proof is given in [6]

2.3.3 Summary of Chapter 1

Before we move to the AMG section, we would like to make a short summary to the most important parts of multigrid presented in the first chapter. We will see that the main principles of geometric multigrid are preserved also in the algebraic multigrid, although the main components of the algorithm are constructed purely algebraically.

The multigrid methods belong to a wide family of iterative solvers and uses several grids to effectively eliminates both oscillatory and non-oscillatory parts of the error. This is their main advantage against simple iterative methods (Section 2.1), which operates only on one layer. Simple iterative methods eliminate high frequency elements of the error after a few iteration steps and then the method converges slowly. Multigrid methods use the simple iterative methods to *smooth* the error, but then projects the error to the coarser grid, where the low frequency modes of the error are eliminated by *coarse grid correction*.

Considering two-level method, according to Lemma 3, the the error can be written as a sum of the smooth component ($\in$ Range P) and oscillatory component ($\in$ Ker $(P^T A)$). We have also shown that the two level method is derived from solving the minimization problem 2.6 in A norm.

The projection operator from fine to the coarse level is the *interpolation* operator and the operator from coarse grid to a fine level is the *prolongator*. In our text we consider only prologator I_{l-1}^{l} and set the interpolation operator simply as a transpose of the prolongator I_{l-1}^{l}.

Prongators are usually constructed by the process of refining the mesh from the coarse level to a fine level by representing the coarse level basis functions by the fine level basis functions (Fig. 2.4).

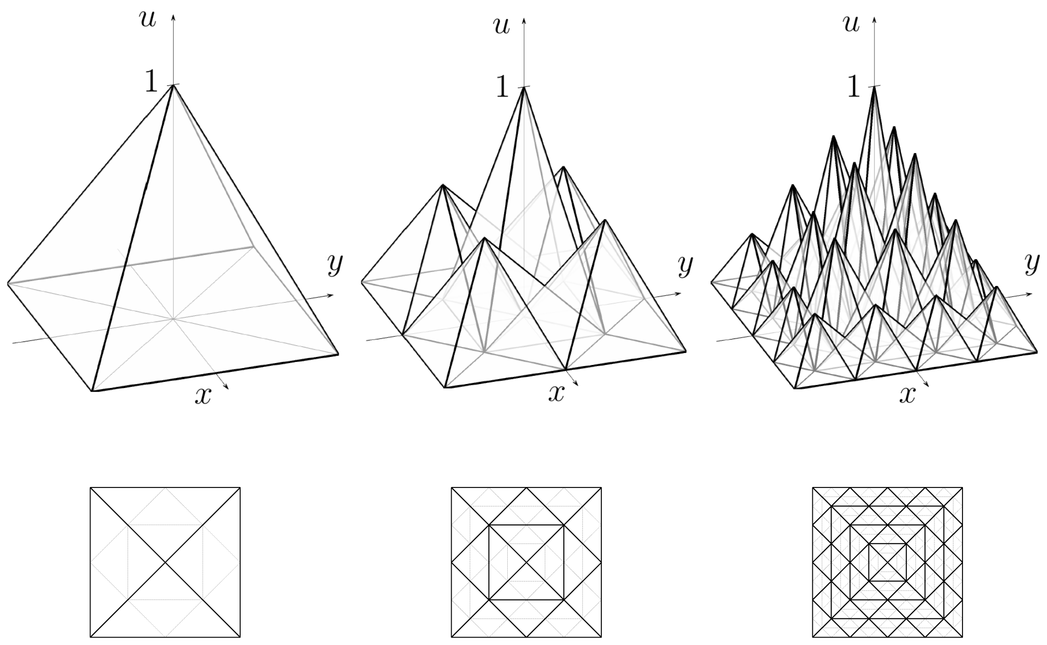


Figure 2.4: Example of basis function on coarse level (left) represented by the basis function o the finer level (right)

The construction of the prolongator operator is geometrical. The situation is simple in 1D case, but sometimes it is very complicated to create a prolongator in a case of complicated meshes in 2D and 3D or when we are not able to create structure of prolongators along with creating of the mesh. In the next chapter, we overcome this problem by constructing the prolongator purely algebraically, given the matrix A.

The uniform convergence of two-level method, $V-cycle$ and $W-cycle$ was proven by various tools. The first proof of uniform convergence of $V-cycle$ was given by Hackbush and uses the introduction of *approximation* and *smoothing* property. The smoothing property ensures that simple iteration method chosen as a smoother will eliminate the oscillatory parts of the error. The smoothed error is then projected to the coarse grids, where the rest of the error is eliminated. The effectivity of the coarse grid correction is ensured by the approximation property.

The proof of uniform convergence based on different principle (regularity-free multigrid) was given by Bramle, Pasciak, Xu and Wang.

Chapter 3

Algebraic Multigrid

In Chapter 1, we have seen that in the standard multigrid, the coarsening process is chosen a priori and the way the prolongators are created is therefore fixed. The smoothing process is chosen so that we are able to ensure an interplay between smoothing and restriction processes. The algebraic multigrid (AMG) is a multilevel method based on a multigrid principle, in which the smoothness is defined purely algebraically. In the AMG, we select a fixed smoother a priori and the coarsening is created so that the whole algorithm converges. This strategy could be motivated by a situation where the prolongation/restriction matrices cannot be created during the coarsening process or when we want to create a black-box solver without previous knowledge of the grid.

In this chapter, we describe the basic idea and components of the algebraic multigrid and then the rest of the chapter chapter will be mainly focused on the smoothed aggregation method considered as a standard method or as setting for the BPX preconditioner.

Remark 13 Previously we described the geometric multigrid in which the prolongation and restriction operators are constructed along with the geometry by refining the coarsest level. The AMG creates a set of coarse spaces automatically from the fine mesh. From now on, the levels will be therefore **numbered from fine to the coarse one**.

In the AMG, we would like to apply similar techniques like in the geometric multigrid, but without having the exact information about the grid. The same fundamental components are used - there is a set of levels, smoother, transfer operators between levels and a solver for the coarsest level.

In the geometric multigrid we had the variables u_i at the grid points, which were known spatial locations. We then selected a subset of these locations as a coarse grid. In the AMG, we don't know the exact location of the variables u_i, but we try to find a subset of u_i as the coarse grid unknowns, given only the algebraic information. If the vector of unknowns has components $\{u_1, \ldots, u_n\}$, then the fine grid points are $\{1, 2, \ldots, n\}$.

The sense of smoothness is also different in the AMG. The local behavior of an algebraically smooth error can be (loosely) expressed as

$$\mathbf{e}_{fine} \approx P\mathbf{e}_{coarse.}$$

As we will see later, we do not necessarily get a geometrically smooth residuum as it was before.

One of the first AMG algorithm [11] was a generalization of geometric multigrid for solving systems of equations. The early convergence results for two level method were obtained in [11], [3] and using variational approach in [36]. Sharper result was given in [66], [19] and [20]. Multilevel result for finite element equation in auxilary space framework is given in [62] for quasi-uniform meshes.

In the following text, the multilevel cycle will be treated as the standard $V-cycle$ (Alg. 2.3) with the AMG components, if not stated otherwise.

3.1 Coarsening Process

The main task for the AMG is to create set of levels and prolongators by some coarsening approaches, given only the fine-level variables. We will describe only the classical approach of Ruge-Stüben and smoothed aggregation, but many coarsening techniques were developed, among others energy-minimization AMG [64], spectral AMGe [33], [12] and unsmoothed and smoothed aggregation AMG [39], [38] and [53].

The Classical Approach

The classical approach is the one of Ruge-Stüben [42], [45] based on strong dependency of the unknowns: to develop an an interplay between two levels, the set of unknowns is split between C or F variables. A subset C of the unknowns is selected for the coarse variables and the rest of the unknowns is called the fine subset F, see Fig. 3.1.

$$\begin{tikzcd}
F & F & F \\
F \arrow[r, dash] & C \arrow[ul, dash] \arrow[u, dash] \arrow[ur, dash] \arrow[r, dash] \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & F \\
F & F & F
\end{tikzcd}$$

Figure 3.1: The C/F splitting

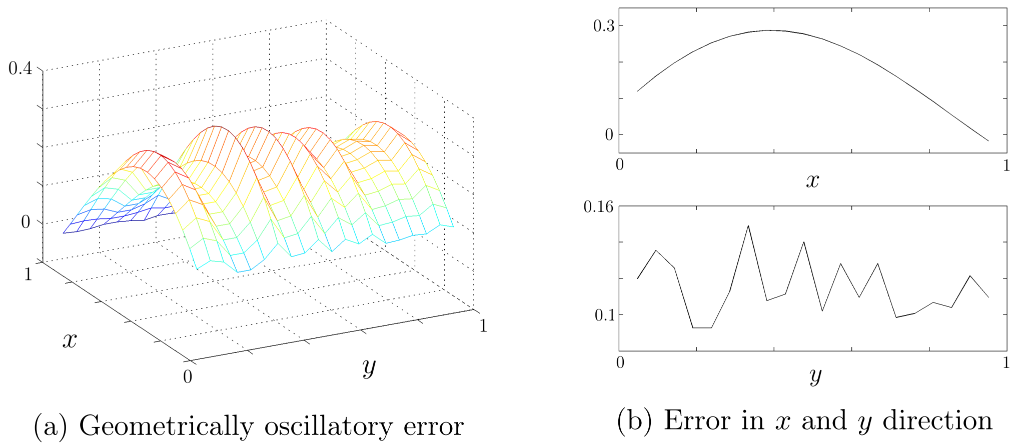
The C/F splitting is based on the strong dependence. The unknown i is strongly dependent on the unknowns j in the set

$$S_i = \{j : -A_{ij} \geq \theta \max_k A_{ij}\},$$

where $\theta < 1$. The actual algorithm is then described in [45].
The algebraical definition of smoothness also suggest that algebraically smooth error doesn't have to be geometrically smooth. In Fig. 3.2a there is an which arised from solving anisotropic problem

$$\epsilon u_{xx} + u_{yy} = 0.$$

We assume our computational domain is a square $\langle -1, 1\rangle \times \langle -1, 1\rangle$ and $0 \leq \epsilon << 1$. In Fig.3.2a there is a algebreaically smooth error which is geometrically smooth in x direction, but geometrically oscillatory in the y direction, due to the anisotropy (Fig.3.2b). The AMG coarsening is done in the x direction only (semicoarsening). The advantage of the AMG is that the semicoarsening arises from the coarsening algorithm in contrast with the geometric multigrid.



(a) Geometrically oscillatory error (b) Error in x and y direction

Figure 3.2: Error after iterations of Gauss-Seidel method

The Aggregation Methods

Other approach to the coarsening belongs to the family of the aggregation methods. The AMG solvers of aggregation type are based on fast and heuristic method of aggregation in which the fine level variables are subdivided into aggregates. An examle of geometrical depiction of aggregates is shown in Fig. 3.3. Every aggregate is then associated with one coarse variable on the next level.

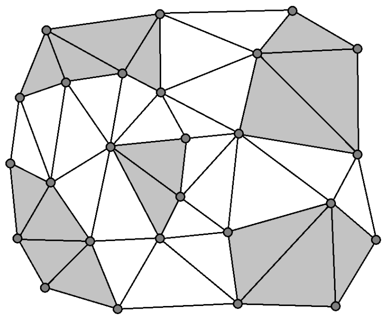


Figure 3.3: Aggregates for 2D problem

It can be viewed as an piecewise constant interpolation from the coarse level variables to the associated aggregates. The aggregation itself is done by some automatic and fast algorithm. The aggreggation-based AMG however are inefficient solvers and also preconditioners. The main two problems of this approach are the poor convergence and also strong $h-dependency$. On the the other hand, the convergence of the constant interpolation can be improved by additional smoothing, which leads to the smoothed aggregation method, described in the next section.

3.2 Smoothed Aggregation Method

The smoothed aggregation method (SA) was developed by Petr Vaněk and first introduced in [50], [49]. This approach is based on the principle of the *tentative prolongator* P^l_{l+1} and the *smoothed prolongator* $I^l_{l+1} = S_l P^l_{l+1}$, where S_l is some polynomial in A_l. The tentative prolongator is created by some fast and easy to compute aggreggation-based algorithm. Most general way of how to obtain the tentative prolongator is by unknowns aggregation technique that is described in [53]. The smoothed prolongator created from the tentative prolongator by smoothing and is used in the actual algorithm. An optimal convergence was proven for the multilevel method in [53] (the convergence result is independent of resolution h).

In the following part of the text, we present and summarize the main ideas of smoothed aggregation. We then focus on the actual process of creating the aggregates and theoretical result of Vaněk, Mandel and Brezina.

3.2.1 SA Two-Level Method

In this section, we describe the two-level SA method.

For the reader's convenience, let us begin with the example in 1D. Consider the the 1D Laplace equation discretized on a mesh consisting of $n_1 = 3n_2$ nodes. We create a system of *aggregates*:

$$\{\{1,2,3\},\{4,5,6\},\dots,\{n_1-2,n_1-1,n_1\}\}$$

forming the disjoint covering of the set $\{1,\dots,n_1\}$.

The simplest tentative prolongator P_2^1 for the problem is constructed so that the columns of P_2^1 are (up to the scaling) 0 or 1 vectors with disjoint nonzero structure. Each column corresponds to disaggregation of one $\mathbb{R}^{n_2}$ variable into three $\mathbb{R}^{n_1}$ variables; nonzero structure of the j-th column corresponds to the j-th *aggregate*. So, P_2^1 can be thought of as a piece-wise constant interpolation in a discrete sense.

P_2^1 is then given by:

$$P_2^1 = \begin{bmatrix} 1 & & \cdot & \\ 1 & & \cdot & \\ 1 & & \cdot & \\ & 1 & \cdot & \\ & 1 & \cdot & \\ & 1 & \cdot & \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ & & \cdot & 1 \\ & & \cdot & 1 \\ & & \cdot & 1 \end{bmatrix}. \tag{3.1}$$

The second step is that the tentative prolongator is smoothed by a linear prolongator smoother. The particular choices of the smoother will be discussed later. The columns of the tentative prolongator creates the discrete basis functions with no overlap. The situation is depicted in Fig. 3.4.

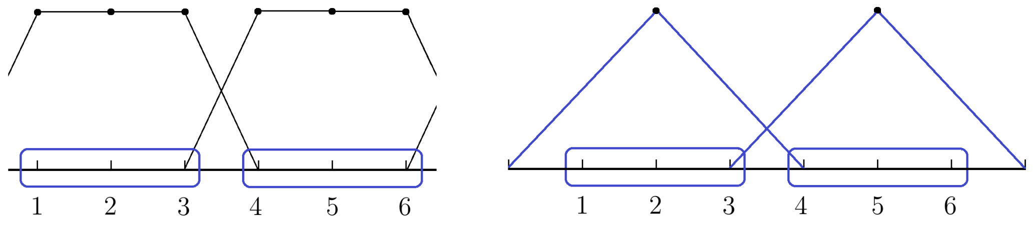


Figure 3.4: Aggregates in 1D and associated basis functions (on the left) and smoothed basis functions (on the right)

Smoothing of the prolongator creates the overlap of discrete basis functions like in the finite elements method. Using just plain aggregation leads to use of constant basis functions and a poor convergence of the method. Smoothing of the prolongator improves the convergence of the method. More detail will be given in the following section.

Another way to derive the two-grid method

Now, we derive the two-level method from solving a minimization problem and compare it with the approach in the geometric MG section (problem 2.6). The standard two-level method minimizes the error in an intermediate stage of the iteration, while we are interested in minimizing the final error after coarse-grid correction and smoothing. The smoothing of prologator manipulates the range of the prolongator to the post-smoother so that the resulting iteration is most efficient.

We will now take a closer look at the two level SA method, as we presented in [24]. In the smoothed aggregation method, we construct the coarse-grid correction to minimize the error *after coarse-grid correction with subsequent smoothing*, which means the final error on the exit of the iteration procedure.

Let S be the error propagation operator of the post-smoother. Throughout this section we assume that S is sparse. This is due to the fact that the above minimization problem leads to smoothed prolongator $I_H^h = SP_H^h$ and we need a sparse coarse-level matrix $A_H = (I_H^h)^T A I_H^h$. The additive point-wise smoothing methods have, in general, sparse error propagation operator; this is the case of Jacobi method or Richardson's iteration.

For a multilevel method with post-smoothing only, the error after coarse-grid correction and subsequent smoothing is given by

$$S(\mathbf{e} - P_H^h \mathbf{v}), \tag{3.2}$$

where $\mathbf{v}$ is a correction vector and $\mathbf{e}$ the error on the entry of the iteration procedure. We choose $\mathbf{v}$ so that the error in (3.2) is minimal in A-norm, that is, for

$n > m$, we solve the minimization problem

$$\text{find } \mathbf{v} \in \mathbb{R}^m \text{ such that } \|S(\mathbf{e} - P_H^h \mathbf{v})\|_A \text{ is minimal.} \tag{3.3}$$

Since $\|S(\mathbf{e} - P_H^h \mathbf{v})\|_A = \|\mathbf{e} - P_H^h \mathbf{v}\|_{S^T A S}$, the minimum is attained for $\mathbf{v}$ satisfying

$$\langle S^T A S(\mathbf{e} - P_H^h \mathbf{v}),\ P_H^h \mathbf{w}\rangle = 0\ \forall \mathbf{w} \in \mathbb{R}^m.$$

We have $\langle S^T A S(\mathbf{e} - P_H^h \mathbf{v}), P_H^h \mathbf{w}\rangle = \langle (P_H^h)^T S^T A S(\mathbf{e} - P_H^h \mathbf{v}), \mathbf{w}\rangle$, hence the above identity is equivalent to $(P_H^h)^T S^T A S P_H^h v = (P_H^h)^T S^T A S \mathbf{e}$ and setting $I_H^h = S P_H^h$, it becomes

$$(I_H^h)^T A I_H^h \mathbf{v} = (I_H^h)^T A S \mathbf{e}. \tag{3.4}$$

Here, $\mathbf{e}$ is the error on the entry of the iteration procedure. Assume for now that I_H^h is injective. Then by (3.4), we have $\mathbf{v} = ((I_H^h)^T A I_H^h)^{-1} (I_H^h)^T A S \mathbf{e}$ and the error after coarse-grid correction and subsequent smoothing is given by

$$S(\mathbf{e} - P_H^h \mathbf{v}) = S\left[\mathbf{e} - I_H^h((I_H^h)^T A I_H^h)^{-1}(I_H^h)^T A S \mathbf{e}\right] = \left[I - I_H^h((I_H^h)^T A I_H^h)^{-1}(I_H^h)^T A\right] S \mathbf{e}. \tag{3.5}$$

By comparing the operator

$$E = \left[I - I_H^h((I_H^h)^T A I_H^h)^{-1}(I_H^h)^T A\right] S \tag{3.6}$$

on the right-hand side of (3.5) with (2.7), we identify E as the error propagation operator of the variational multigrid with smoothed prolongator $I_H^h = S P_H^h$ and pre-smoothing step given by $\mathbf{u} \leftarrow \mathcal{S}(\mathbf{u}, \mathbf{f})$. The algorithm is as follows:

Algorithm 3.2.1

```
1: pre-smooth:   $\mathbf{u} \leftarrow \mathcal{S}(\mathbf{u}, \mathbf{f})$
2: evaluate the residual:   $\mathbf{r} = A\mathbf{u} - \mathbf{f}$
3: restrict the residual:   $\mathbf{r}_H = (I_H^h)^T \mathbf{r}$
4: solve the coarse-level problem:   $A_H v = \mathbf{r}_H, \quad A_H = (I_H^h)^T A I_H^h$
5: correct the approximation:   $u \leftarrow \mathbf{u} - I_H^h \mathbf{v}$
```

We summarize our considerations in the form of a theorem.

Theorem 14 *The error propagation operator E in (3.6) (the error propagation operator of Algorithm 3.2.1) satisfies the identity*

$$\|E\mathbf{e}\|_A = \inf_{\mathbf{v} \in \mathbb{R}^m} \|S(\mathbf{e} - P_H^h \mathbf{v})\|_A$$

for all $\mathbf{e} \in \mathbb{R}^n$.

Proof. The proof follows directly from the fact that Algorithm 3.2.1 was derived from variational objective (3.3). □

Remark 15 One may also start with the variational objective to minimize the final error after performing the pre-smoothing, the coarse-grid correction and the post-smoothing. Such extension is trivial, the pre-smoother has no influence on the coarse-grid correction operator $I - I_H^h((I_H^h)^T A I_H^h)^{-1}(I_H^h)^T A$ and influences only its argument. Indeed, asuming the error propagation operator of the pre-smoother is S^* (the A-adjoint operator), the final error is given by $S(S^*\mathbf{e} - P_H^h\mathbf{v})$ and we solve the minimization problem

$$\text{for } \mathbf{e} \in \mathrm{I\!R}^n \text{ find } \mathbf{v} \in \mathrm{I\!R}^m \ : \ \|S(S^*\mathbf{e} - P_H^h\mathbf{v})\|_A \text{ is minimal.} \tag{3.7}$$

Fundamentally, this is the same minimization problem as (3.3);to derive the corresponding algorithm, it is simply sufficient to follow our manipulations from (3.3) to (3.5) with $\mathbf{e} \leftarrow S^*\mathbf{e}$. This way, we end up with a two-level method that has the error propagation operator

$$E = \left[I - I_H^h((I_H^h)^T A I_H^h)^{-1}(I_H^h)^T A\right] SS^*, \tag{3.8}$$

that is, with the algorithm 3.2.2.

Algorithm 3.2.2

```
1: pre-smooth:   $\mathbf{u} \leftarrow \mathcal{S}_t(\mathbf{u}, \mathbf{f})$
2: pre-smooth:   $\mathbf{u} \leftarrow \mathcal{S}(\mathbf{u}, \mathbf{f})$
3: evaluate the residual:   $\mathbf{r} = A\mathbf{u} - \mathbf{f}$
4: restrict the residual:   $\mathbf{r}_H = (I_H^h)^T\mathbf{r}$
5: solve the coarse-level problem:   $A_H\mathbf{v} = \mathbf{r}_H, \quad A_H = (I_H^h)^T A I_H^h$
6: correct the approximation:   $\mathbf{u} \leftarrow \mathbf{u} - P\mathbf{v}$
```

where $\mathcal{S}_t$ is an iterative method with error propagation operator S^*, $\mathcal{S}$ is an iterative method with error propagation operator S .

We summarize the content of Remark 15 as a theorem.

Theorem 16 *The error propagation operator (3.8) of Algorithm 3.2.2 satisfies the identity*

$$\|E\mathbf{e}\|_A = \inf_{\mathbf{v} \in \mathrm{I\!R}^m} \|S(S^*\mathbf{e} - P_H^h\mathbf{v})\|_A$$

for all $\mathbf{e} \in \mathrm{I\!R}^n$.

Proof. The proof follows directly from the fact that Algorithm 3.2.2 was derived from variational objective (3.7). □

3.2.2 Coarsening Method

In the previous section, we described the advantages of using the smoothed prolongator. It is constructed from a tentative prolongator. The aim is to get the tentative prolongator by a fast and automatic algorithm.

At first, we describe a method by Vaněk, Mandel and Brezina in [55]. We start with a disjoint decomposition of the degrees of freedom on each level into *aggregates*. Every *aggregate* on the level l can be considered as the degree of freedom on the level $l+1$.

For a given ϵ define the strongly coupled neighborhood of node i as

$$N_i^l(\epsilon) = \{j : |a_{ij}| \geq \epsilon\sqrt{a_{ii}a_{jj}}\} \cup \{i\} \tag{3.9}$$

The aggregates are now created in algorithm 3.2.3 and the tentative prolongator P_l (here, we use P_l instead of P_{l+1}^l for simplicity) is defined by the aggregates C_l^i:

$$(P_l)_{ij} = \begin{cases} 1 \text{ if } i \in C_j^l \\ 0 \text{ otherwise} \end{cases} .$$

Algorithm 3.2.3 Aggregation

Let A_l of order n_l and ϵ be given. Generate a disjoint covering $\{C_i^l\}_{i=1}^{n_{l+1}}$ of the set $\{1,\dots,n_l\}$ as follows.

```
1: Set $R = \{1,\dots,n_l\}$ and $j = 0$
2: Select disjoint strongly coupled neighborhoods as the initial attempted covering: If there exists a strongly coupled neighborhood $N_i^l(\epsilon) \subset R$, set $j \leftarrow j+1$, $C_j^l \leftarrow N_i^l(\epsilon)$, $R \leftarrow R \backslash C_j^l$. Repeat until $R$ does not contain any strongly coupled neighborhood.
3: Add each remaining $i \in R$ to one of the sets already selected to which it is strongly connected, if possible:
     • Copy $\tilde{C}_k^l = C_k^l, k = 1,\dots,j$
     • If there exists $i \in R$ and $k$ such that $N_i^l(\epsilon) \cap \tilde{C}_k^l \neq 0$ then set $C_k^l \leftarrow C_k^l \cup \{i\}$
     • Repeat until no such $i$ exists
4: Make the remaining $i \in R$ into aggregates that consist of subset of strongly coupled neighborhoods: If there exists $i \in R$, set $j \leftarrow j+1$ and $C_k^l \cap N_i^l(\epsilon)$. Repeat until $R = 0$.
```

In [55], it is then suggested to improve the tentative prolongator by a smoothing step

$$I_l = (I - \omega D^{-1} A_l^F) P_l,$$

where $A_i^F = (a_{ij}^F)$ is the filtered matrix given by

$$a_{ij}^F = \left\{ \begin{array}{l} a_{ij} \text{ if } j \in N_i^l(\epsilon) \\ 0 \text{ otherwise} \end{array} \right\} \text{ if } i \neq j,$$

$$a_{ii}^F = a_{ii} - \sum_{j=1, j\neq i}^{n_l} (a_{ij} - a_{ij}^F).$$

In [55], it is suggested to use

$$\epsilon = 0.08(0.5)^{l-1}, \quad \omega = \frac{2}{3}.$$

3.2.3 Generalized Aggregation

Now, having the system of aggregates from algorithm 3.2.3, the general way to create a system of tentative prolongators from aggregates is described in [53]. Our aim is to create a hierarchy of tentative prolongators such that for a given $n_1 \times r$ matrix B^1

$$\text{Range}B^1 \subset \text{Range}P_l^1, \quad P_l^1 = P_2^1 \dots P_l^{l-1}, \quad l = 1, \dots, L-1. \tag{3.10}$$

The Range of matrix B^1 specifies, which functions on the finest level will be represented exactly on all levels. Matrix B^1 is chosen as a kernel of stiffness matrix without essential boundary conditions.

We create P_{l+1}^l and a n_{l+1} matrix B^{l+1} so that

$$P_{l+1}^l B^{l+1} = B^l. \tag{3.11}$$

Prolongator P_{l+1}^l is is constructed form a given system of aggregates $\{\mathcal{A}\}_{i=1}^{N_l}$. Aggregates can be constructed as in the method 3.2.3. The property 3.11 is enforced to each of aggregates. The columns of P_{l+1}^l corresponding to aggregate $\mathcal{A}_{i=1}^l$ are created by orthonormalized restrictions of the columns of B^l on aggregate $\mathcal{A}_{i=1}^l$.

Algorithm 3.2.4 Construction of tentative prolongator

```
1: Partition the vector $B^l$ into $n_{l+1}$ blocks $B_i^l, \quad i = 1, \dots, n_l$
2: $Q_i^l \leftarrow \frac{B_i^l}{\|B_i^l\|}$
3: $B_i^{l+1} \leftarrow \|B_i^l\|$
4: $P_{l+1}^l = diag(Q_i^l)$, $B^{l+1} \leftarrow (B_1^{l+1}, \dots B_{n_{l+1}}^{l+1})^T$
```

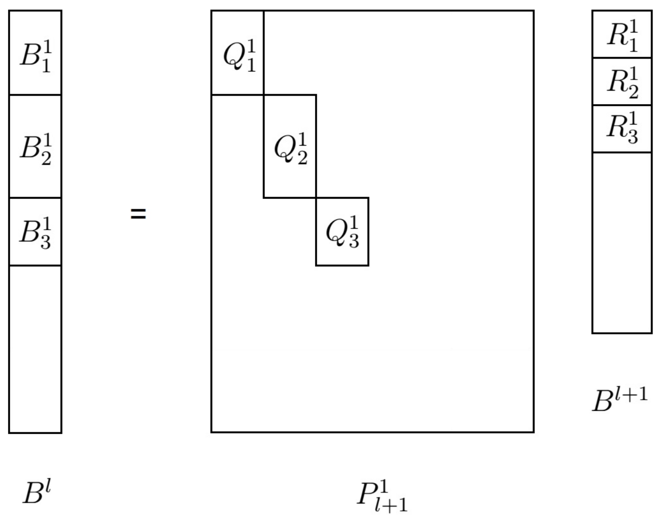


Figure 3.5: The tentative prolongator P_{l+1}^l

Following [53], the smoothing is then given as follows: Let the smoothing of the prolongator is given by a smoother S_l, $S_l : \mathbb{R}^{n_l} \to \mathbb{R}^{n_l}$,

$$S_l = I - \frac{4}{3\bar{\lambda}_l^M} M_l^{-1} A_l, \tag{3.12}$$

where $\bar{\lambda}_l^M \geq \rho(M_l^{-1}A_l)$ and $M_l = (P_l^1)^{\mathrm{T}} P_l^1$

Result of Vanek, Mandel and Brezina

In the end of this section, we would like to show a result of result of Vanek, Mandel and Brezina, [53]. Before we proceed to the main theorem, we now describe the effect of smoothing the prolongator for a two-level method. In [54], it was shown that that the essential convergence condition for a two-level method is a form of weak approximation property: Assume there exists a constant C so that for every $\mathbf{u} \in \mathbb{R}^n$ there exists $\mathbf{v} \in \mathbb{R}^m$

$$\|\mathbf{u} - P\mathbf{v}\|_{\mathbb{R}^n} \leq C/\sqrt{\bar{\lambda}} \|u\|_A, \quad n > m, \tag{3.13}$$

($\bar{\lambda}$ is a known upper bound of $\rho(S^2A)$). We can compare this with similarly looking condition of A. Brandt for the two level method [10]: Let assume there exists a constant C so that for every $\mathbf{u} \in \mathbb{R}^n$ there exists $\mathbf{v} \in \mathbb{R}^m$

$$\|\mathbf{u} - P\mathbf{v}\|_{\mathbb{R}^n} \leq C/\sqrt{\rho(A)} \|\mathbf{u}\|_A.$$

For a special choice of smoother, we can achieve

$$\rho(S^2A) << \rho(A),$$

which means weak approximation property 3.13 is much weaker (easier to satisfy).

In [54], authors suggest a construction of such a smoother.

Now, we present the result of [53]. Let us define a composite aggregate $\tilde{\mathcal{A}}^l_{i=1}$ as the aggregate $\mathcal{A}^l_{i=1}$, understood as the corresponding set of supernodes on the finest level.

Theorem 17 *Let the prolongator smoothers S_l be given by 3.12 and the tentative prolongators given by algorithm 3.2.4. Assume there is a constant $C_{\mathcal{A}} > 0$ such that for every $\mathbf{u} \in \mathbb{R}^{n_1}$ and every $l = 1, \ldots, L-1$*

$$\sum_{i-1}^{N_l} \min_{\mathbf{w} \in \mathbb{R}^r} \|\mathbf{u} - B^1 \mathbf{w}\|^2_{l_2(\tilde{\mathcal{A}}^l_{i=1})} \leq C_{\mathcal{A}} \frac{9^{l-1}}{\bar{\lambda}_1} \|\mathbf{u}\|_A. \tag{3.14}$$

The inequality 3.14 induces, how well is a fine vector approximated by aggregates by vectors B^1.

Remark 18 As an example let B^1 be represented by a single constant on the finest level. To verify 3.14, we have to use Poincaré-Friedrichs inequality

$$\inf_{c \in \mathbb{R}} \|u(x) - c\|_{L^2(\Omega)} \leq C \mathrm{diam}(\Omega) |u|_{H^1(\Omega)} \quad \forall u \in H^1(\Omega) \tag{3.15}$$

aggregate by aggregate.

3.3 BPX Algorithm in SA Setting

In this part, we will present the theory based on the BPX algorithm in the setting of smoothed aggregations. In the previous sections we did not use the bold font for vectors. But in the following section we distinguish between normal and bold font. The notation will be sometimes slightly changed according to our article [25].

3.3.1 Introduction

In the previous text, we dealt with the classical multigrid. It can be seen as a multiplicative method of Schwarz type with inexact subspace solvers given by smoothers ([6]). It has to be performed as a successive algorithm which, preventing from large-scale parallelism. Unlike standard multigrid, the so-called BPX preconditioner frame of Bramble, Pasciak and Xu [7] is fully additive, allowing for parallelism of a single coarse-space basis function on each level. The sufficient conditions for its convergence and the mathematical requirements on its efficient implementation are, however, different from the ones for multiplicative multilevel iterative methods, despite the fact that the sufficient conditions look similar.

The smoothed aggregation algebraic multigrid coarsening technique was proved to be very efficient in solving large systems of linear algebraic equations arising from the discretization of elliptic problems ([49, 53, 52, 54]).

The smoothed aggregation method was, however, developed and analyzed in the context of traditional multiplicative multigrid. Here, we use smoothed aggregation in the BPX frame and analyze the convergence of the resulting iterative method applied to a model example.

In the unpublished technical report [55], authors made a first attempt to analyze the smoothed aggregation method in the context of standard multigrid. The report contains merely a sketch of the theory. It is necessary to establish the resolution-independent equivalence of discrete and continuous L_2-norms

$$\left\| \sum_i x_i \varphi_i^l \right\|_{L_2} \approx \mathit{scaling} \left(\sum_i x_i^2 \right)^{1/2} \tag{3.16}$$

for the hierarchy of coarse-spaces span$\{\varphi_i^l\}_i$. The equivalence was used to prove the weak approximation property needed to verify the assumptions of the regularity-free abstract multigrid convergence theory of [6].

In [53], for the standard multigrid, the need for this equivalence is avoided by fully algebraic means. In the context of the BPX pre-conditioner, however, equivalence (3.16) is unavoidable. The of the BPX pre-conditioner requires the computationally cheap implementation of the approximate l_2-projections onto coarse-spaces; such implementation must avoid the action of the inverse of the Gram matrix. Thus, for the coarse-space basis, we need a Gram matrix that has an inverse that can be approximated by the inverse of its diagonal. For this reason, we will return to the method of analysis outlined in [55] and developed it fully for the case of model geometry.

For BPX pre-conditioner based on smoothed aggregation we prove, assuming model geometry and H^1-equivalent form, that the condition number of the preconditioned system grows at most as $O(L^2)$, where L is the number of levels.

Presented theory requires the coarse-space bases (or their supports) to form a system of disjoint macroelements covering the entire computational domain. The macroelement function is spanned solely by the set of associated basis functions; no other basis functions are allowed to intersect the macroelement with their supports. Such macroelements are obviously formed in the case of regular geometry. In the general case, however, the smoothed aggregation coarse-space bases tend to form the macroelements as well. The equivalence of discrete and continuous L_2-norms is therefore very likely to hold for unstructured aggregation formed on unstructured meshes.

The interpolation estimates (the weak approximation property of the coarse-space bases with the l_2-norm of the left-hand-side measured on the finest level) are more or less standard variation on the finite element theory of [16], used in a variety of forms in many works, for example [53]. In a BPX context, those estimates had to be carried out for smoothed aggregation coarse-space basis functions, that we, using an algebraic trick, avoided in [53]. Here, only the weak approximation property of pure aggregations had to be proved.

3.4 BPX pre-conditioner in operator setting

In this section we define the BPX pre-conditioner and give a convergence bound. At the end of the section, we describe the implementation of the method assuming the system of prolongators is given. Up to minor technical details that suit our purpose, this section follows [7].

In what follows, we often use symbol $\mathcal{A}$ for $\mathcal{A}_1$. Denote σ_l to be the largest eigenvalue of $\mathcal{A}_l$. Assume $\bar{\sigma}_l \geq \sigma_l$, $l = 1, \ldots, L$ is an upper bound, $\bar{\sigma}_{l+1} \leq \bar{\sigma}_l$. Let $\mathcal{Q}_l : U \to U_l$ be an orthogonal projection and $\tilde{\mathcal{Q}}_l : U \to U_l$ its spectrally equivalent approximation. The BPX pre-conditioner is defined by

$$\mathcal{B} = \frac{1}{\bar{\sigma}_1}\tilde{\mathcal{Q}}_1 + \sum_{l=2}^{L}\left(\frac{1}{\bar{\sigma}_l} - \frac{1}{\bar{\sigma}_{l-1}}\right)\tilde{\mathcal{Q}}_l. \tag{3.17}$$

Since $\mathcal{Q}_1 = \mathcal{I}$, where $\mathcal{I}$ denotes the identity mapping, we can also set $\tilde{\mathcal{Q}}_1 = \mathcal{Q}_1 = \mathcal{I}$. Note that by rearranging the sums and setting $\tilde{\mathcal{Q}}_{L+1} = 0$ we get

$$\begin{aligned}
\mathcal{B} &= \frac{1}{\bar{\sigma}_1}\tilde{\mathcal{Q}}_1 + \sum_{l=2}^{L}\frac{1}{\bar{\sigma}_l}\tilde{\mathcal{Q}}_l - \sum_{l=2}^{L}\frac{1}{\bar{\sigma}_{l-1}}\tilde{\mathcal{Q}}_l \\
&= \sum_{l=1}^{L}\frac{1}{\bar{\sigma}_l}\tilde{\mathcal{Q}}_l - \sum_{l=1}^{L-1}\frac{1}{\bar{\sigma}_l}\tilde{\mathcal{Q}}_{l+1} - \frac{1}{\bar{\sigma}_L}\tilde{\mathcal{Q}}_{L+1} \\
&= \sum_{l=1}^{L}\frac{1}{\bar{\sigma}_l}\left(\tilde{\mathcal{Q}}_l - \tilde{\mathcal{Q}}_{l+1}\right).
\end{aligned} \tag{3.18}$$

In the following theorem, we give a convergence bound of [7]. Since the proof is relatively simple and we prove a slightly different statement than the authors of [7] (with the upper bounds $\bar{\sigma}_l$ in the place of the actual maximal eigenvalues σ_l), we provide the proof in detail for readers' convenience.

Theorem 19 *([7]) Assume there is a constant C_1 independent of l and L such that for every $u \in U$ and every level $l = 1, \ldots, L$, the exact projections $\mathcal{Q}_l$, $\mathcal{Q}_{L+1} = 0$,* *satisfy*

$$\|(\mathcal{I} - \mathcal{Q}_{l+1})\, u\|_U^2 \le \frac{C_1}{\bar{\sigma}_l}\, a(u,u). \tag{3.19}$$

In addition, we assume operators $\tilde{\mathcal{Q}}_l$, $l = 1, \ldots, L$, are spectrally equivalent to projections $\mathcal{Q}_l$ in the sense that

$$c_2\, (\mathcal{Q}_l u,\, u)_U \;\le\; \left(\tilde{\mathcal{Q}}_l u,\, u\right)_U \;\le\; C_2\, (\mathcal{Q}_l u,\, u)_U \;\; \forall u \in U,\; l = 1, \ldots, L \tag{3.20}$$

with constants $C_2 \ge c_2 > 0$ independent of l and L, $l = 1, \ldots, L-1$. Last, we assume that $\bar{\sigma}_{l+1} \le \bar{\sigma}_l$ for all levels $l = 1, \ldots, L-1$. Then

$$\frac{c_2}{C_1 L}\, a(u,u) \;\le\; a\,(\mathcal{BA}\, u,\; u) \;\le\; C_2 L\, a(u,u) \;\; \forall u \in U. \tag{3.21}$$

The following proof is a genuine work of Bramble, Pasciak and Xu. The upper bound is more or less straightforward. The proof of coercivity (the lower bound) is similar to the proof of Lion's lemma.

Proof. Let us set $\mathcal{B}_{ex}$ to be operator $\mathcal{B}$ with $\tilde{\mathcal{Q}}_l = \mathcal{Q}_l$ for all levels l. Let $u \in U$. By (3.17) and (3.32), we have

$$\begin{aligned} a\,(\mathcal{BA}\, u,\; u) \;&=\; \frac{1}{\bar{\sigma}_1}\left(\tilde{\mathcal{Q}}_1 \mathcal{A}\, u,\; \mathcal{A}\, u\right)_U \\ &+\; \sum_{l=2}^{L} \left(\frac{1}{\bar{\sigma}_l} - \frac{1}{\bar{\sigma}_{l-1}}\right) \left(\tilde{\mathcal{Q}}_l \mathcal{A}\, u,\; \mathcal{A}\, u\right)_U \end{aligned}$$

and

$$\begin{aligned} a\,(\mathcal{B}_{ex} \mathcal{A}\, u,\; u) \;&=\; \frac{1}{\bar{\sigma}_1}\,(\mathcal{Q}_1 \mathcal{A}\, u,\; \mathcal{A}\, u)_U \\ &+\; \sum_{l=2}^{L} \left(\frac{1}{\bar{\sigma}_l} - \frac{1}{\bar{\sigma}_{l-1}}\right) (\mathcal{Q}_l \mathcal{A}\, u,\; \mathcal{A}\, u)_U \end{aligned}$$

with

$$c_2\, (\mathcal{Q}_l \mathcal{A}\, u,\; \mathcal{A}\, u)_U \le \left(\tilde{\mathcal{Q}}_l \mathcal{A}\, u,\; \mathcal{A}\, u\right)_U \le C_2\, (\mathcal{Q}_l \mathcal{A}\, u,\; \mathcal{A}\, u)_U$$

by (3.20). Therefore

$$c_2\, a\,(\mathcal{B}_{ex} \mathcal{A}\, u,\; u) \;\le a\,(\mathcal{BA}\, u,\; u) \;\le\; C_2\, a\,(\mathcal{B}_{ex} \mathcal{A}\, u,\; u) \;\; \forall u \in U.$$

It is therefore sufficient to prove (3.21) with $\mathcal{B}_{ex}$ in the place of $\mathcal{B}$ and $c_2 = C_2 = 1$.

Set $U_{L+1} = \emptyset$. Define W_l to be the orthogonal complement of U_{l+1} in U_l, that is,

$$W_l = \{u \in U_l : (u, w)_U = 0\; \forall w \in U_{l+1}\},\; l = 1, \ldots, L.$$

Clearly, spaces W_l, $l = 1, \ldots, L$, form an orthogonal decomposition of U and the operators $\mathcal{Q}_l - \mathcal{Q}_{l+1}$ are orthogonal projections onto the respective spaces W_l. As a consequence of this orthogonality, (3.32) and (3.18), using properties of orthogonal projections

$$\mathcal{Q}_l - \mathcal{Q}_{l+1} = (\mathcal{Q}_l - \mathcal{Q}_{l+1})^2 = (\mathcal{Q}_l - \mathcal{Q}_{l+1})^*$$

(* denotes the adjoint operator) and the Pythagorean Theorem,

$$\begin{aligned}
a(\mathcal{B}_{ex}\mathcal{A}\,u,\ u) &= \sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l}\, a\left((\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u,\ u\right) \\
&= \sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l} \left((\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u,\ \mathcal{A}\,u\right)_U \\
&= \sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l} \left((\mathcal{Q}_l - \mathcal{Q}_{l+1})^2\,\mathcal{A}\,u,\ \mathcal{A}\,u\right)_U \\
&= \sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l} \left((\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u,\ (\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\right)_U \\
&= \sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l} \left\|(\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\right\|_U^2 \qquad (3.22) \\
&= \sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l} \left(\|\mathcal{Q}_l\mathcal{A}\,u\|_U^2 - \|\mathcal{Q}_{l+1}\mathcal{A}\,u\|_U^2\right) \\
&\leq \sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l} \|\mathcal{Q}_l\mathcal{A}\,u\|_U^2\,. \qquad (3.23)
\end{aligned}$$

Let $\mathcal{P}_l$ be $a(\cdot,\cdot)-$orthogonal projection onto U_l, $l = 1, \ldots, L$ and $u, v \in U$. Since $\mathcal{P}_l$ is $a(\cdot,\cdot)$-symmetric, $\mathcal{P}_l = \mathcal{I}$ on U_l, $\mathcal{Q}_l$ is symmetric and $\mathcal{I} - \mathcal{Q}_l$ is the orthogonal projection onto $U_l^\perp$ ($\perp$ denotes the orthogonal complement),

$$\begin{aligned}
(\mathcal{Q}_l\mathcal{A}\,u,\ v)_U &= (\mathcal{A}\,u,\ \mathcal{Q}_l v)_U = (\mathcal{A}\,u,\ \mathcal{P}_l\mathcal{Q}_l v)_U = (\mathcal{A}\,\mathcal{P}_l u,\ \mathcal{Q}_l v)_U \\
&= (\mathcal{A}_l\mathcal{P}_l u,\ \mathcal{Q}_l v)_U = (\mathcal{A}_l\mathcal{P}_l u,\ \mathcal{Q}_l v)_U + (\mathcal{A}_l\mathcal{P}_l u,\ (\mathcal{I} - \mathcal{Q}_l)\,v)_U \\
&= (\mathcal{A}_l\mathcal{P}_l u,\ v)_U\,,
\end{aligned}$$

hence

$$\mathcal{Q}_l\mathcal{A} = \mathcal{A}_l\mathcal{P}_l$$

and therefore

$$\|\mathcal{Q}_l\mathcal{A}\,u\|_U^2 = \|\mathcal{A}_l\mathcal{P}_l u\|_U^2 \leq \bar{\sigma}_l\, a(\mathcal{P}_l u, \mathcal{P}_l u) \leq \bar{\sigma}_l a(u,u).$$

This estimate together with (3.23) prove the upper bound of (3.21) with $\mathcal{B}_{ex}$ in the place of $\mathcal{B}$ and $C_2 = 1$.

To establish the lower bound of (3.21), we estimate using

$$\mathcal{I} = \sum_{l=1}^{L} (\mathcal{Q}_l - \mathcal{Q}_{l+1}),$$

the fact that $\mathcal{I} - \mathcal{Q}_l$ is orthogonal projection onto $U_l^{\perp}$ and Cauchy-Schwarz inequality,

$$\begin{aligned}
a(u,u) &= \sum_{l=1}^{L} a\left((\mathcal{Q}_l - \mathcal{Q}_{l+1})\,u,\ u\right) \\
&= \sum_{l=1}^{L} \left((\mathcal{Q}_l - \mathcal{Q}_{l+1})\,u,\ \mathcal{A}\,u\right)_U \\
&= \sum_{l=1}^{L} \left((\mathcal{Q}_l - \mathcal{Q}_{l+1})\,u,\ (\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\right)_U \\
&= \sum_{l=1}^{L} \left[\left((\mathcal{I} - \mathcal{Q}_l)\,u,\ (\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\right)_U\right. \\
&+ \left.\left((\mathcal{Q}_l - \mathcal{Q}_{l+1})\,u,\ (\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\right)_U\right] \\
&= \sum_{l=1}^{L} \left((\mathcal{I} - \mathcal{Q}_{l+1})\,u,\ (\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\right)_U \\
&\le \sum_{l=1}^{L} \|(\mathcal{I} - \mathcal{Q}_{l+1})\,u\|_U \ \|(\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\|_U .
\end{aligned}$$

Thus, by assumption (3.19), Cauchy-Schwarz inequality and (3.22) we get

$$\begin{aligned}
a(u,u) &\le \sum_{l=1}^{L} \left(\frac{C_1}{\bar{\sigma}_l}\right)^{1/2} a^{1/2}(u,u)\,\|(\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\|_U \\
&\le C_1^{1/2}\, a^{1/2}(u,u) \left(\sum_{l=1}^{L} \frac{1}{\bar{\sigma}_l}\ \|(\mathcal{Q}_l - \mathcal{Q}_{l+1})\,\mathcal{A}\,u\|_U^2\right)^{1/2} \left(\sum_{l=1}^{L} 1^2\right)^{1/2} \\
&= (C_1 L)^{1/2}\, a^{1/2}(u,u)\, a\left(\mathcal{B}_{ex} u,\ u\right)^{1/2}.
\end{aligned}$$

Assume $u \neq 0$. Dividing the above estimate by $a^{1/2}(u,u)$ and squaring the result we get

$$(\mathcal{B}_{ex} u, u) \ge \frac{1}{C_1 L}\, a(u,u),$$

proving the first inequality of (3.21) for $\mathcal{B}_{ex}$ in the place of $\mathcal{B}$ and $c_2 = 1$. For $u = 0$, (3.21) holds trivially. This completes our proof. □

Here, we describe implementation of the method. The exact projection operators in the matrix form are

$$\mathcal{Q}_l = Q_l = I_l^1 \left((I_l^1)^T I_l^1\right)^{-1} (I_l^1)^T, \;\; l = 1, \dots, L. \tag{3.24}$$

We choose the inexact projections to be the operators Q_l with the matrix $(I_l^1)^T I_l^1$ replaced by its diagonal, that is

$$\tilde{\mathcal{Q}}_l = \tilde{Q}_l = I_l^1 D_l^{-1} (I_l^1)^T, \; D_l = \text{diag}\left((I_l^1)^T I_l^1\right), \;\; l = 1, \dots, L. \tag{3.25}$$

The action of BPX preconditioner (3.17) is given by the following algorithm:

Algorithm 3.4.1

```
given: $\mathbf{x} \in \mathbb{R}^n$
1: evaluate the action $\mathbf{y} = B\mathbf{x} \in \mathbb{R}^n$ of preconditioner $B = \mathcal{B}$ by
```

$$\mathbf{y} = \frac{1}{\bar{\sigma}_1}\mathbf{x} + \sum_{l=2}^{L} \left(\frac{1}{\bar{\sigma}_l} - \frac{1}{\bar{\sigma}_{l-1}}\right) I_l^1 D_l^{-1} (I_l^1)^T \mathbf{x}, \; D_l = \text{diag}\left((I_l^1)^T I_l^1\right). \tag{3.26}$$

In (3.26), $\bar{\sigma}_l$ is an upper bound of

$$\sigma_l = \lambda_{max}\left(\mathcal{A}_l\right) = \sup_{\mathbf{x} \in \text{ Range}(I_l^1) \setminus \{\mathbf{0}\}} \frac{\|\mathbf{x}\|_A^2}{\|\mathbf{x}\|^2} = \sup_{\mathbf{x} \in \mathbb{R}^{n_l} \setminus \{\mathbf{0}\}} \frac{\|I_l^1 \mathbf{x}\|_A^2}{\|I_l^1 \mathbf{x}\|^2}, \; l = 1, \dots, L. \tag{3.27}$$

The implementation of the method will be discussed more in detail in the later section.

3.5 Smoothed aggregation prolongators in model case

In the smoothed aggregation method ([49, 50, 53]) we create prolongator I_{l+1}^l (assuming prolongators $I_2^1, \dots, I_l^{l-1}$ are already given) in the form

$$I_{l+1}^l = S_l P_{l+1}^l.$$

Here, S_l is an $n_l \times n_l$ sparse linear *prolongator smoother* being the first degree polynomial in $A_l = (I_l^1)^T A I_l^1$ and P_{l+1}^l an $n_l \times n_{l+1}$ *tentative prolongator* given by unknowns aggregation. The tentative prolongator is responsible for the approximation, while the prolongator smoother enforces the smoothness of the coarse-level spaces. The simplest prolongator P^l_{l+1} will be given in this section. For the most general form of tentative prolongator applicable to non-scalar problems on unstructured meshes, see [53].

Let $\Omega = (0,1) \times (0,1)$ be a computational domain. We consider a model problem

$$\text{find } u \in H^1_0(\Omega) : a(u,v) = f(v) \ \forall v \in H^1_0(\Omega). \tag{3.28}$$

Here, $a(\cdot,\ \cdot) = (\nabla\cdot, \nabla\cdot)_{L_2(\Omega)}$ and $f(\cdot) \in (H^1_0(\Omega))^{-1}$. The problem is discretized by $P1$ elements on a uniform triangular mesh obtained from a regular square mesh when each square is broken by connecting its left lower and right upper vertices with a straight edge. We assume the number of interior nodes in the direction of both axes x and y is 3^{L-1}.

On the finest level, we form the aggregates (index sets of vertices) $\{\mathcal{A}^1_i\}_{i=1}^{9^{L-2}}$ by grouping the mesh vertices into 3×3 regular, square groups. For each aggregate, the central vertex represents the aggregate on the second level. Thus, we have mesh vertices on level 2 and the procedure can be repeated, giving rise to the hierarchy of the aggregates $\{\mathcal{A}^l_i\}_{i=1}^{9^{L-l-1}}$, $l = 1, \ldots, L-1$ and the hierarchy of nodal points $\{\mathbf{v}^l_i\}_{i=1}^{n_l}$, $l = 1, \ldots, L$, with $\mathbf{v}^l_i$ being the central point of the aggregate $\mathcal{A}^{l-1}_i$. Then, we define the tentative prolongators

$$(P^l_{l+1})_{ij} = \begin{cases} 1 & \text{for } i \in \mathcal{A}^l_j, \\ 0 & \text{otherwise} \end{cases}, \tag{3.29}$$

$i = 1, \ldots, n_l$, $j = 1, \ldots, n_{l+1}$, $n_l = 9^{L-l}$, $l = 1, \ldots, L-1$. Thus, P^l_{l+1} is a 0/1 matrix with disjoint nonzero structure. Each column of P^l_{l+1} corresponds to the disaggregation of one $\mathbb{R}^{n_{l+1}}$ variable into nine $\mathbb{R}^{n_l}$ variables. Thus, P^l_{l+1} can be thought of as a piecewise constant coarsening in a discrete sense.

Next we specify the prolongator smoother. Let $\bar{\lambda}_1 \geq \lambda_{max}(A)$ be an available upper bound. We set

$$\bar{\lambda}_l = \bar{\lambda}_1 \tag{3.30}$$

for all levels $l = 2, \ldots, L$. In Lemma 4, we will show that $\lambda_{max}(A_l) \leq \bar{\lambda}_l$. Define prolongator smoothers S_l by

$$S_l = I - \frac{4}{3}\frac{1}{\bar{\lambda}_l} A_l, \quad l = 1, \ldots L-1. \tag{3.31}$$

The parameter $\frac{4}{3}$ is chosen because, in a certain sense, it minimizes the upper bound of $\lambda_{max}(S^2_l A_l)$. The details are obvious from (3.37) in the proof of Lemma 4.

The choice of the upper bounds $\bar{\sigma}_l \geq \sigma_l$ needed in (4.7) is addressed by Remark 24.

3.6 Verification of the assumptions of the abstract theory

Let $(U, (\cdot,\cdot)_U, \|\cdot\|_U)$ be a Hilbert space. Consider a problem

$$\text{find } u \in U : a(u,v) = f(v) \; \forall v \in U.$$

Here, $a(\cdot,\cdot)$ is a symmetric bilinear form coercive and continuous on U and $f(\cdot) \in U^{-1}$ (U^{-1} is the dual space). Let

$$U = U_1 \supset U_2 \supset \ldots \supset U_L$$

be a hierarchy of nested Hilbert spaces with the inner product inherited from U. For each $l = 1,\ldots,L$, define operator $\mathcal{A}_l : U_l \to U_l$ by

$$a(u_l, v_l) = (\mathcal{A}_l u_l, v_l)_U \;\; \forall u_l, v_l \in U_l. \tag{3.32}$$

Define the coarse-level basis functions $\varphi_i^l = \pi_1 I_l^1 \mathbf{e}_i^l$, $l = 1,\ldots,L$, $i = 1,\ldots,n_l$. Here, π_1 is the finest level finite element interpolator

$$\pi_1 : \mathbf{x} \in \mathbb{R}^n \mapsto \sum_i x_i \varphi_i^1$$

with $\{\varphi_i^1\}_{i=1}^n$ being the finest level finite element basis and $\mathbf{e}_i^l$ the i-th canonical basis vector of $\mathbb{R}^{n_l}$.

Lemma 4 *Assume $\bar\lambda_1 \geq \lambda_{max}(A)$ is an available upper bound satisfying $\bar\lambda_1 \leq C\lambda_{max}(A)$. Set $\bar\lambda_l = \bar\lambda_1$ for all levels $l = 1,\ldots,L$. There is a constant $C > 0$ independent of the mesh-size h, level l and basis function number i such that for all $l = 1,\ldots,L$, $i = 1,\ldots,n_l$,*

$$\|\varphi_i^l\|_{H^1(\Omega)} \leq C, \tag{3.33}$$

$$\|\varphi_i^l\|_{L_2(\Omega)} \leq Ch_l, \quad h_l = 3^{l-1}h \tag{3.34}$$

and

$$\lambda_{max}(A_l) \leq \bar\lambda_l \leq C. \tag{3.35}$$

Proof. Assume the first inequality of (3.35) holds for level l, $1 \leq l < L$. We will show that $\bar\lambda_{l+1} = \bar\lambda_l = \bar\lambda_1$ satisfies the first inequality of (3.35) as well. We estimate

$$\begin{aligned}
\lambda_{max}(A_{l+1}) &= \sup_{\mathbf{x}\in\mathbb{R}^{n_{l+1}}\setminus\{\mathbf{0}\}} \frac{\langle A_{l+1}\mathbf{x},\mathbf{x}\rangle}{\|\mathbf{x}\|^2} \\
&= \sup_{\mathbf{x}\in\mathbb{R}^{n_{l+1}}\setminus\{\mathbf{0}\}} \frac{\langle A_l S_l P^l_{l+1}\mathbf{x}, S_l P^l_{l+1}\mathbf{x}\rangle}{\|\mathbf{x}\|^2} \\
&= \sup_{\mathbf{x}\in\mathbb{R}^{n_{l+1}}\setminus\{\mathbf{0}\}} \frac{\langle S_l^2 A_l P^l_{l+1}\mathbf{x}, P^l_{l+1}\mathbf{x}\rangle}{\|\mathbf{x}\|^2} \\
&\leq \lambda_{max}(S_l^2 A_l) \sup_{\mathbf{x}\in\mathbb{R}^{n_{l+1}}\setminus\{\mathbf{0}\}} \frac{\|P^l_{l+1}\mathbf{x}\|^2}{\|\mathbf{x}\|^2}. \qquad (3.36)
\end{aligned}$$

Next we estimate $\lambda_{max}(S_l^2 A_l)$ in terms of $\bar{\lambda}_l$. By the spectral mapping theorem,

$$\begin{aligned}
\lambda_{max}(S_l^2 A_l) &= \lambda_{max}\left(\left(I - \frac{4}{3}\frac{1}{\bar{\lambda}_l}A_l\right)^2 A_l\right) \\
&= \max_{\lambda\in\sigma(A_l)} \left(1 - \frac{4}{3}\frac{1}{\bar{\lambda}_l}\lambda\right)^2 \lambda \\
&= \bar{\lambda}_l \max_{\lambda\in\sigma(A_l)} \left(1 - \frac{4}{3}\frac{\lambda}{\bar{\lambda}_l}\right)^2 \frac{\lambda}{\bar{\lambda}_l} \\
&\leq \bar{\lambda}_l \max_{t\in[0,1]} \left(1 - \frac{4}{3}t\right)^2 t \\
&= \frac{1}{9}\bar{\lambda}_l. \qquad (3.37)
\end{aligned}$$

Since each aggregate contains exactly 9 degrees of freedom, it holds that

$$\frac{\|P^l_{l+1}\mathbf{x}\|^2}{\|\mathbf{x}\|^2} = 9 \qquad (3.38)$$

for all $\mathbf{x} \in \mathbb{R}^{n_l}$. This identity, (3.36) and (3.37) give

$$\bar{\lambda}_{l+1} \equiv \bar{\lambda}_l \geq \lambda_{max}(A_{l+1}).$$

The proof of the first inequality of (3.35) follows by induction with $\bar{\lambda}_1 \geq \lambda_{max}(A)$. Now, the well-known bound $\lambda_{max}(A) \leq C$ ([16]) with C independent of h, gives the second inequality of (3.35).

The estimate (3.33) is a consequence (3.35). Indeed,

$$\begin{aligned}
|\varphi_i^l|^2_{H^1(\Omega)} &= (\nabla\pi_1 I_l^1 \mathbf{e}_i^l, \nabla\pi_1 I_l^1 \mathbf{e}_i^l)_{L_2(\Omega)} = \langle A_1 I_l^1 \mathbf{e}_i^l, I_l^1 \mathbf{e}_i^l\rangle \\
&= \langle A_l \mathbf{e}_i^l, \mathbf{e}_i^l\rangle \leq \lambda_{max}(A_l)\|\mathbf{e}_i^l\|^2 \leq C.
\end{aligned}$$

Since the $H^1(\Omega)$-norm and the $H^1(\Omega)$-seminorm are equivalent on $H_0^1(\Omega)$ by Friedrich's inequality, we get the statement (3.33).

Let us prove (3.34). It is well-known ([16]) that

$$ch\|\mathbf{x}\| \leq \|\pi_1\mathbf{x}\|_{L_2(\Omega)} \leq Ch\|\mathbf{x}\| \quad \forall\mathbf{x} \in \mathbb{R}^{n_1} \tag{3.39}$$

with constants $C \geq c > 0$ independent of h. We estimate using $\varrho(S_l) = \lambda_{max}(I - \frac{4}{3}\frac{1}{\lambda_l}A) \leq 1$, (3.38) and (3.39),

$$\begin{array}{rcl} \|\varphi_i^l\|_{L_2(\Omega)} & = & \|\pi_1 I_l^1 \mathbf{e}_i^l\|_{L_2(\Omega)} \\ & \leq & Ch\|I_l^1\mathbf{e}_i^l\| \\ & = & Ch\|S_1P_2^1I_l^2\mathbf{e}_i^l\| \\ & \leq & Ch\varrho(S_1)\|P_2^1I_l^2\mathbf{e}_i^l\| \\ & \leq & Ch\|P_2^1I_l^2\mathbf{e}_i^l\| \\ & = & 3Ch\|I_l^2\mathbf{e}_i^l\| \\ & = & \ldots = \\ & = & 3^{l-1}Ch\|I_l^l\mathbf{e}_i^l\| \\ & = & Ch_l. \end{array}$$

This constitutes the proof of (3.34). □

To make our theory comprehensible, we introduce the notion of macroelement. The macroelement has two aspects: the set of associated basis functions $\{\varphi_i\}_{i\in\tau}$ (τ is an index set) and the geometrical domain T (understood closed) that contains the intersection of their supports, that is,

$$T \supset \bigcap_{i\in\tau} \operatorname{supp}\varphi_i. \tag{3.40}$$

The essential properties of the macroelements $\langle T, \{\varphi_j\}_{j\in\tau}\rangle$ are:

1. Property (3.40),
2. the closed domains T have disjoint interiors and cover the entire computational domain,
3. except for the basis functions φ_i, $i \in \tau$, associated with the macroelement, no other basis functions are allowed to intersect int T with their supports.

The function on macroelement is therefore a linear combination of macroelement basis functions $\{\varphi_i\}$, $i \in \tau$ satisfying (3.40) with no other basis functions involved. The rigorous definition folows:

Definition 8 *Consider a computational domain Ω and a system of basis functions $\{\varphi_i\}$, with (well-defined) supports contained in $\bar{\Omega}$. Let $\{T_i\}$ be a family of closed domains $T_i \subset \bar{\Omega}$ such that*

a)

$$\bigcup_i T_i = \bar{\Omega} \text{ and} \operatorname{int} T_i \,\cap\, \operatorname{int} T_j = \emptyset \text{ for } i \neq j, \tag{3.41}$$

b) for every T_i there is an index set τ_i such that the corresponding set of basis functions $\{\varphi_j\}_{j\in\tau_i}$ satisfies

$$\bigcap_{j\in\tau_i} \operatorname{supp}\varphi_j \subset T_i \text{ and} \operatorname{supp}\varphi_j \cap \operatorname{int} T_i = \emptyset \;\forall\, j \notin \tau_i. \tag{3.42}$$

Then we call the system $\left\{\langle T_i,\ \{\varphi_j\}_{j\in\tau_i}\rangle\right\}_i$ a system of macroelements on Ω.

Remark 20 Clearly, the finite elements as concieved in [16] are, according to Definition 8, also macroelements.

It is a matter of routine to show that for Poisson equation discretized using $P1$ elements on a uniform grid, the coarse-space basis functions obtained by smoothed aggregation using the aggregates consisting of three neighboring nodes are in fact $P1$ basis functions as well, see [56]. The coarse-level resolution is $3\times$ the fine-level resolution. The macroelements are then formed by overlaps of supports of two adjacent basis functions and are identical with coarse $P1$ elements.

Before introducing our macroelements and proving their properties, we need several auxiliary statements:

Lemma 5 *The coarse-level spaces satisfy the following properties:*

a) The coarse-level matrices follow the nine point scheme; entry a^l_{ij} of $A_l = (I^1_l)^T A I^1_l$ can be nonzero only for directly adjacent (in the 9-point scheme) aggregates $\mathcal{A}^{l-1}_i$ and $\mathcal{A}^{l-1}_j$ on the level $l-1$. On the first level, the adjacency of the aggregates is considered assuming an underlying 9-point scheme instead of a 7-point scheme.

b) Apart from the vertices directly adjacent to the boundary with essential boundary condition, the vector of ones $\mathbf{1}_l \in \mathbb{R}^{n_l}$ forms the kernel of A_l, i.e.

$$(A_l\mathbf{1}_l)_i = 0 \tag{3.43}$$

for all vertices $\mathbf{v}^l_i$ not adjacent to the boundary with essential boundary condition. Adjacency to the boundary is considered assuming an underlying 9-point scheme extended to the boundary nodes, see Fig. 3.6.

c) *Apart from boundary with an essential boundary condition, the discrete basis functions $I_l^1 \mathbf{e}_i^l$ form a decomposition of unity in the sense that*

$$(I_l^1 \mathbf{1}_l)_i = \left(\sum_{j=1}^{n_l} I_l^1 \mathbf{e}_j \right)_i = 1 \tag{3.44}$$

for all fine-level vertices $\mathbf{v}_i^1$ that belong to the (closed) square $\bar{\Omega}_{int}^l$ with vertices $\mathbf{v}_{i_{c_1}}^l, \mathbf{v}_{i_{c_2}}^l, \mathbf{v}_{i_{c_3}}^l, \mathbf{v}_{i_{c_4}}^l$ adjacent to the corners of the unit square Ω. The continuous basis functions $\varphi_i^l = \pi_1 I_l^1 \mathbf{e}_i^l$ satisfy

$$\sum_{i=1}^{n_l} \varphi_i^l = 1 \textit{ on } \bar{\Omega}_{int}^l, \tag{3.45}$$

see Fig. 3.6.

d) *Support of each of the basis functions $\varphi_i^l = \pi_1 I_l^1 \mathbf{e}_i^l$ satisfies*

$$\mathrm{supp}\ \varphi_i^l \subset \Omega_{supp,i}^l,$$

with $\Omega_{supp,i}^l$ being a (closed) square with side-length $2h_l = 2 \cdot 3^{l-1}h$ and the center of gravity located in $\mathbf{v}_i^l$. Apart from $\partial\Omega$, the vertices and edge midpoints of $\Omega_{supp,i}^l$ are vertices $\mathbf{v}_j^l$, $j \in \mathcal{N}_i^l \setminus \{i\}$, see Fig. 3.6. Here, $\mathcal{N}_i^l$ denotes the neighbourhood of i in the nine-point scheme.

Proof. Let us prove statement a). Assume the stencil of A^{l-1} follows the nine point scheme. Let $\mathcal{A}_i^{l-1}$ and $\mathcal{A}_j^{l-1}$ be two aggregates. Clearly, a_{ij}^l of $A_l = (I_l^{l-1})^T A_{l-1} I_l^{l-1}$ can be nonzero only if supp $I_l^{l-1}\mathbf{e}_i^l$ and supp $I_l^{l-1}\mathbf{e}_j^l$ are directly adjacent sets (in the 9-point scheme) of vertices on level $l-1$. Since $I_l^{l-1} = S_{l-1}P_l^{l-1}$, where S_l is a first-degree polynomial in A_{l-1} and P_l^{l-1} is given by disaggregation (supp $P_l^{l-1}\mathbf{e}_i^l = \mathcal{A}_i^{l-1}$), the supports supp $I_l^{l-1}\mathbf{e}_i^l$ and supp $I_l^{l-1}\mathbf{e}_j^l$ are adjacent only for two directly adjacent aggregates $\mathcal{A}_i^{l-1}$ and $\mathcal{A}_j^{l-1}$. The proof of a) now follows by induction, with the fact that the matrix $A = A_1$, being a finite element stiffness matrix, follows the seven-point scheme which is a subset of the nine-point scheme.

Let us prove statement b). Assume vertex $\mathbf{v}_i^l$ is not adjacent to the boundary with essential boundary condition. Recall that matrices A_l on all levels follow the nine-point scheme. Assume statement b) holds on the level $l-1$. To prove our statement on the level l, it is sufficient to establish that

$$\sum_{j \in \mathcal{N}_i^l} a_{ij}^l = 0,$$

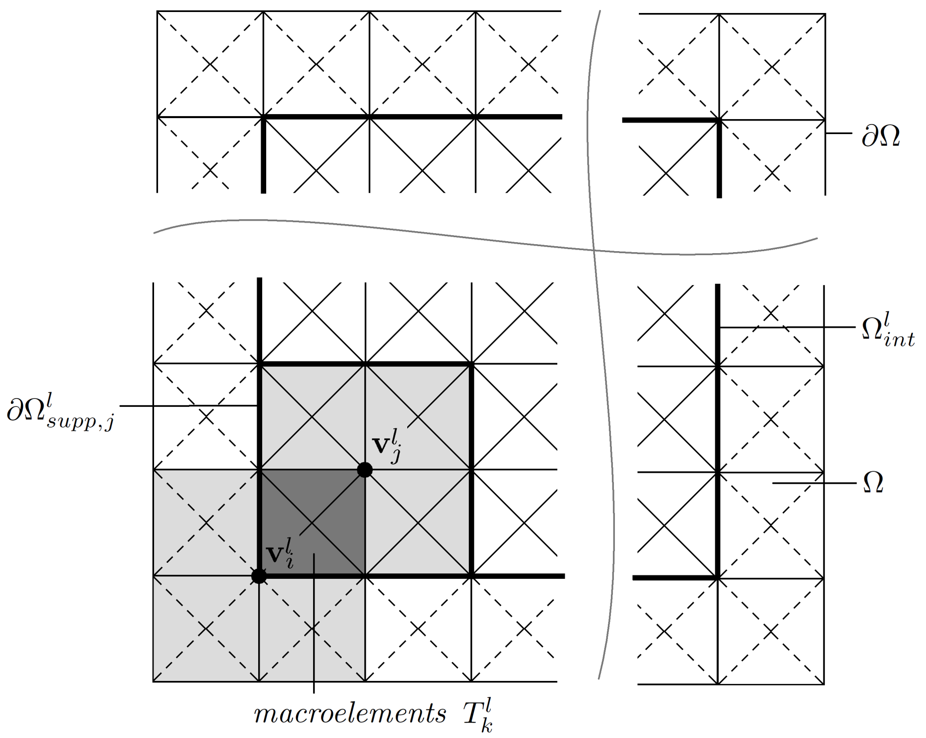


Figure 3.6: Coarse-level geometry

where $\mathcal{N}^l_i$ is the neighborhood of i in the nine-point scheme. Clearly,

$$\begin{aligned}\sum_{j\in\mathcal{N}^l_i} a^l_{ij} &= \sum_{j\in\mathcal{N}^l_i} \langle A^{l-1}\mathbf{e}^l_i, \mathbf{e}^l_j\rangle \\ &= \sum_{j\in\mathcal{N}^l_i} \langle A_{l-1}S^2_{l-1}P^{l-1}_l\mathbf{e}^l_i, P^{l-1}_l\mathbf{e}^l_j\rangle \\ &= \langle A_{l-1}S^2_{l-1}P^{l-1}_l\mathbf{e}^l_i, \sum_{j\in\mathcal{N}^l_i} P^{l-1}_l\mathbf{e}^l_j\rangle. \end{aligned} \tag{3.46}$$

Further, supp $A_{l-1}S^2_{l-1}P^{l-1}_l\mathbf{e}^l_i =$ supp $A_{l-1}(I-\omega A_{l-1})^2P^{l-1}_l\mathbf{e}^l_i$ is contained in $\mathcal{A}^{l-1}_i$ with 3 layers added, which equals $\mathcal{A}^{l-1}_i$ with adjacent aggregates added. Therefore we have

$$\left(\sum_{j\in\mathcal{N}^l_i} P^{l-1}_l\mathbf{e}^l_j\right)_k = 1 \text{ for } k\in\mathcal{N}\equiv\bigcup_{j\in\mathcal{N}^l_i}\mathcal{A}^{l-1}_j \supset \text{supp } A_{l-1}S^2_{l-1}P^{l-1}_l\mathbf{e}^l_i. \tag{3.47}$$

Denote int $\mathcal{N}$ to be the interior of the above set $\mathcal{N}\subset\{1,\ldots,n_{l-1}\}$, defined as

int $\mathcal{N} = \{k : \mathcal{N}^{l-1}_k \subset \mathcal{N}\}\cap\{k : \mathbf{v}^{l-1}_k$ is not a vertex adjacent to the boundary $\partial\Omega\}$.

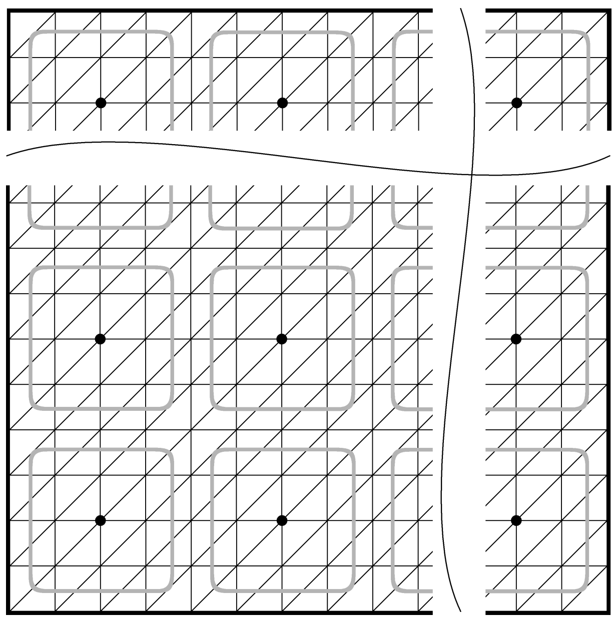


Figure 3.7: Coarse-level geometry

Clearly,

$$\text{supp}\, S_{l-1}^2 P_l^{l-1} \mathbf{e}_i^l \subset \text{int}\, \mathcal{N}.$$

From this, (3.46), and (3.47) it follows that

$$\begin{aligned}\sum_{j\in\mathcal{N}_i^l} a_{ij}^l &= \langle A_{l-1} S_{l-1}^2 P_l^{l-1} \mathbf{e}_i^l, \mathbf{1}_{l-1}\rangle \\ &= \langle S_{l-1}^2 P_l^{l-1} \mathbf{e}_i^l, A_{l-1}\mathbf{1}_l\rangle \\ &= \langle S_{l-1}^2 P_l^{l-1} \mathbf{e}_i^l, A_{l-1}\mathbf{1}_l\rangle_{l_2(\text{int}\,\mathcal{N})} \\ &= 0\end{aligned}$$

as

$$(A_{l-1}\mathbf{1}_{l-1})_k = 0 \;\forall\; k \in \text{int}\,\mathcal{N} \subset \text{int}\,\{1,\ldots,n_{l-1}\},$$

by assumption, proving b) for level l. The proof of b) on all levels follows by induction, with the fact that the finite element stiffness matrix satisfies b).

Let us prove c). Consider a set $\mathcal{M} \subset \{1,\ldots,n_l\}$. Let $\mathbf{x} \in \mathbb{R}^{n_l}$ be the vector such that $x_i^l = 1$ for all $i \in \mathcal{M}$. By property b), $(A_l\mathbf{x})_k = 0 \;\forall\; k \in \text{int}\,\mathcal{M}$ and therefore $\mathbf{y}_l = S_l\mathbf{x} = (I - \omega A_l)\mathbf{x}$ satisfies $y_i = 1$ for all $i \in \text{int}\,\mathcal{M}$. Assume c) holds for an intermediate I_l^k with some $k \in \{2,\ldots,l\}$, that is,

$$(I_l^k \mathbf{1}_l)_p = 1 \;\forall p : \mathbf{v}_p^k \in \bar{\Omega}_{int}^l.$$

(See c).) Then, $(P_k^{k-1} I_l^k \mathbf{1}_l)_p = 1$ for all vertices $\mathbf{v}_p^{k-1} \in \bar{\Omega}_{int}^l$, with one layer of vertexes added. The vector

$$\mathbf{y} = I_l^{k-1}\mathbf{1}_l = S_{k-1} P_k^{k-1} I_l^k \mathbf{1}_l$$

therefore satisfies $y_p = 1$ for all p such that $\mathbf{v}_p^l \in \bar{\Omega}_{int}^l$. The proof now follows by induction, with the fact that $I_l^l \mathbf{1}_l = \mathbf{1}_l$ satisfies c). Property (3.45) is a direct consequence.

To prove d), it is sufficient to show that

$$\{\mathbf{v}_j^1,\ j \in \text{supp } I_l^1 \mathbf{e}_i^l\} \subset \text{int } \Omega_{supp,i}^l. \tag{3.48}$$

Assume

$$\{\mathbf{v}_j^k,\ j \in \text{supp } I_l^k \mathbf{e}_i^l\} \subset \text{int } \Omega_{supp,i}^l \tag{3.49}$$

for some $k \in \{2, \ldots, l\}$. Consider the set

$$\omega_i^{l,k-1} = \text{supp } P_k^{k-1} I_{l-1}^k \mathbf{e}_i^l = \bigcup_{j \in \text{supp } I_{l-1}^k \mathbf{e}_i^l} \mathcal{A}_j^{k-1}.$$

Obviously, for the set $\tilde{\omega}_i^{l,k-1}$ consisting of $\omega_i^{l,k-1}$ with one layer of surrounding vertices added, that is,

$$\tilde{\omega}_i^{l,k-1} = \omega_i^{l,k-1} \cup \{j \in \mathcal{N}_p^{k-1},\ p \in \omega_i^{l,k-1}\},$$

the corresponding set of vertices is contained in int $\Omega_{supp,i}^l$. The proof of (3.49) for $k-1$ in place of k is completed by the fact that

$$\text{supp } I_l^{k-1}\mathbf{e}_i^l = \text{supp } (I - \omega A_{k-1}) P_k^{k-1} I_l^k \mathbf{e}_i^l \subset \tilde{\omega}_i^{l,k-1}.$$

The proof of (3.49) for all $k \in 1, \ldots, l$ follows by induction, with the fact that (3.49) obviously holds for $k = l$. □

For $l > 1$, let us connect vertices $\mathbf{v}_j^l$, $j = 1, \ldots, n_l$ by the regular square mesh extended to the boundary $\partial\Omega$, see Fig. 3.6. This mesh consists of squares; let us choose a numbering of those squares (including those adjacent to the boundary) and denote them $\{T_i^l\}$. For each square T_i^l define an index set τ_i^l of numbers of vertices $\mathbf{v}_i^l$ that are its corner vertices. (Note that there are no vertices $\mathbf{v}_i^l$ located at $\partial\Omega$.)

Lemma 6 *For $l > 1$, the system $\{\langle T_i^l, \{\varphi_j^l\}_{j \in \tau_i^l}\rangle\}_i$, $\varphi_j^l = \pi_1 I_l^1 \mathbf{e}_j^l$, is a system of macroelements on Ω.*

Proof. We verify the conditions of Definition 8.

Obviously, squares T_j^l have disjoint interiors and cover entire computational domain Ω. Thus, (3.41) holds.

By Lemma 5 d), vertices $\mathbf{v}_i^l$ are located at centers of gravity of squares $\Omega_{supp,i}^l \supset$ supp φ_i^l. Clearly,

$$T_i^l = \bigcap_{j \in \tau_i^l} \Omega_{supp,j}^l \supset \bigcap_{j \in \tau_i^l} \operatorname{supp} \varphi_j^l.$$

and for $j \notin \tau_i^l$

$$\operatorname{int} T_i^l \cap \operatorname{supp} \varphi_j^l \subset \operatorname{int} T_i^l \cap \Omega_{supp,j}^l = \emptyset.$$

This proves (3.42). □

Lemma 7 *For $l > 1$, basis functions φ_i^l, $i = 1, \dots, n_l$, satisfy the following properties:*

1. *The $H^1(\Omega)$-seminorm and $L_2(\Omega)$-norm of each φ_i^l, $l = 1, \dots, L$, $i = 1, \dots, n_l$, are bounded by*
$$|\varphi_i^l|_{H^1(\Omega)} \le C \tag{3.50}$$
and
$$\|\varphi_i^l\|_{L_2(\Omega)} \le Ch_l. \tag{3.51}$$
Here, $C > 0$ is a constant independent of mesh-size h, level l, and basis function number i.

2. *For T_i^l that is not adjacent to a boundary with an essential boundary condition, the quadruple of associated basis functions ($\tau_i^l = \{i_1, i_2, i_3, i_4\}$) satisfies*
$$\sum_{j=1}^{4} \varphi_{i_j}^l = 1 \text{ on } T_i^l.$$

3. *On the edges of an interior macroelement T_i^l, the traces of basis functions satisfy*
$$\begin{aligned} \operatorname{tr} \varphi_{i_1}^l &= 0 \text{ on } e_2 \cup e_3, \\ \operatorname{tr} \varphi_{i_2}^l &= 0 \text{ on } e_3 \cup e_4, \\ \operatorname{tr} \varphi_{i_3}^l &= 0 \text{ on } e_4 \cup e_1, \\ \operatorname{tr} \varphi_{i_4}^l &= 0 \text{ on } e_1 \cup e_2. \end{aligned}$$
In the above identity, the edges e_j of T_i^l (in the local numbering) and vertices $\mathbf{v}_{i_j}^l$ are numbered in the same way as on the reference element as depicted in Fig. 3.8.

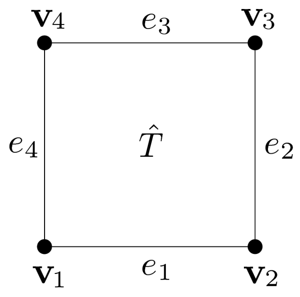


Figure 3.8: Reference square

Proof. The statement No. 1 follows directly from Lemma 4.

Statement No. 3 of our lemma follows by Lemma 5 d). The decomposition of unity (Statement 2.) follows by Lemma 5 c) and d). This completes our proof. □

Lemma 8 *Let $\hat{T}$ be a unit square with edges and vertices as depicted on Fig. 3.8. Define the set $\mathcal{G} = \{(\hat{\varphi}_1, \hat{\varphi}_2, \hat{\varphi}_3, \hat{\varphi}_4)^T\} \subset [H^1(\hat{T})]^4$, where each function $\hat{\varphi}_i$ is associated with vertex $\hat{\mathbf{v}}_i$, by the following properties:*

1. *There is a positive, finite constant C such that*
$$\|\hat{\varphi}_i\|_{H^1(\hat{T})} \leq C, \quad i = 1, \ldots, 4. \tag{3.52}$$
2. *The functions $\hat{\varphi}_i$, $i = 1, \ldots, 4$, satisfy the decomposition of unity*
$$\sum_{i=1}^{4} \hat{\varphi}_i = 1 \text{ on } \hat{T}. \tag{3.53}$$
3. *The traces $\operatorname{tr} \hat{\varphi}_i \in H^{1/2}(\partial\hat{T})$ of functions $\hat{\varphi}_i$, $i = 1, \ldots, 4$, on the boundary $\partial\hat{T}$, denoted later simply as $\hat{\varphi}_i$, satisfy*
$$\begin{aligned} \hat{\varphi}_1 &= 0 \text{ on } \hat{e}_2 \cup \hat{e}_3, \\ \hat{\varphi}_2 &= 0 \text{ on } \hat{e}_3 \cup \hat{e}_4, \\ \hat{\varphi}_3 &= 0 \text{ on } \hat{e}_1 \cup \hat{e}_4, \\ \hat{\varphi}_4 &= 0 \text{ on } \hat{e}_1 \cup \hat{e}_2. \end{aligned}$$
See Fig. 1.

Define the Gram matrix
$$G = G(\hat{\varphi}_1, \hat{\varphi}_2, \hat{\varphi}_3, \hat{\varphi}_4) \equiv \{g_{ij} = (\hat{\varphi}_i, \hat{\varphi}_j)_{L_2(\hat{T})}\}_{ij=1}^4.$$
There is a positive constant c dependent exclusively on C such that
$$\lambda_{min}(G) \geq c > 0$$
for all quadruples $(\hat{\varphi}_1, \hat{\varphi}_5, \hat{\varphi}_3, \hat{\varphi}_4)^T \in \mathcal{G}$.

Proof. Let us prove that the basis functions $\hat{\varphi}_i$, $i = 1, \ldots, 4$ are linearly independent. Assume for now the contrary, i.e.

$$\sum_{i=1}^{4} c_i \hat{\varphi}_i = 0 \text{ with some } c_i \neq 0.$$

By property No. 3, $\hat{\varphi}_3 = \hat{\varphi}_4 = 0$ on $\hat{e}_1$. Hence, by the assumption of the decomposition of unity (3.53),

$$\hat{\varphi}_1 + \hat{\varphi}_2 = 1 \text{ on } \hat{e}_1 \tag{3.54}$$

and, by the assumption of the linear dependence,

$$c_1\hat{\varphi}_1 + c_2\hat{\varphi}_2 = 0 \text{ on } \hat{e}_1, \tag{3.55}$$

with $c_1 \neq 0$ or $c_2 \neq 0$. Let us say that $c_1 \neq 0$. We will show that properties (3.54), (3.55) and $c_1 \neq 0$ exclude each other.

By (3.55) and (3.54) it follows that

$$(c_1 - c_2)\hat{\varphi}_1 = -c_2 \text{ on } \hat{e}_1.$$

Let $c_2 \neq 0$. Then $\hat{\varphi}_1 = \mathit{const} \neq 0$ on $\hat{e}_1$. Since $\hat{\varphi}_1 = 0$ on $\hat{e}_2$ by property No. 3, it follows that there is a jump in the trace of $\hat{\varphi}_1$ at the point $\hat{\mathbf{v}}_2$ and thus, $\hat{\varphi}_1 \notin H^{1/2}(\partial\hat{T})$, which contradicts the trace theorem, as $\|\hat{\varphi}_1\|_{H^1(\hat{T})} \leq C < +\infty$.

Consider now the case of $c_2 = 0$. Then, by (3.55) $c_1\hat{\varphi}_1 = 0$, hence by (3.54) it follows that $c_1(1 - \hat{\varphi}_2) = 0$ on $\hat{e}_1$. Since $c_1 \neq 0$ by the assumption, it follows that $\hat{\varphi}_2 = 1$ on $\hat{e}_1$. Since $\hat{\varphi}_2 = 0$ on $\hat{e}_4$, there is a jump in the trace of $\hat{\varphi}_2$ at the point $\hat{\mathbf{v}}_1$, hence $\hat{\varphi}_2 \notin H^{1/2}(\partial\hat{T})$, which contradicts the trace theorem. Thus, $c_1 \neq 0$ leads to the contradiction for any c_2, hence $c_1 = 0$.

Due to the double axial symmetry of $\hat{T}$ (with respect to both x and y), it follows that $\sum_{i=1}^{4} c_i\hat{\varphi}_i = 0$ implies $c_1 = c_2 = c_3 = c_4 = 0$ and therefore the basis functions $\hat{\varphi}_i$, $i = 1, \ldots, 4$, are linearly independent.

Since G is a Gram matrix corresponding to the linearly independent basis, the functional

$$\Phi(\hat{\varphi}_1, \hat{\varphi}_2, \hat{\varphi}_3, \hat{\varphi}_4) = \lambda_{min}(G(\hat{\varphi}_1, \hat{\varphi}_2, \hat{\varphi}_3, \hat{\varphi}_4))$$

is a positive functional on $\mathcal{G}$. By the Cauchy-Schwarz inequality, entries $g_{ij} = (\hat{\varphi}_i, \hat{\varphi}_j)_{L_2(\hat{T})}$ are continuous bilinear forms on $L_2(\hat{T})$, hence continuous functionals on $\mathcal{G}$. At the same time, eigenvalues of G depend continuously on the entries g_{ij}. Thus, Φ is a continuous, positive functional on $[L_2(\hat{T})]^4$. In the rest of the proof we will show that the set $\mathcal{G}$ is compact in $[L_2(\hat{T})]^4$. Clearly, the set $\mathcal{G}$ is bounded in $[H^1(\hat{T})]^4$, hence weakly precompact. Further, the set $\mathcal{G}$ is convex. Indeed, for two functions $\hat{\varphi}_i$, $\hat{\varphi}_i'$ such that

$$\|\hat{\varphi}_i\|_{H^1(\hat{T})} \leq C, \quad \|\hat{\varphi}_i'\|_{H^1(\hat{T})} \leq C$$

and α, β non-negative numbers such that $\alpha + \beta = 1$, it holds that

$$\|\alpha\hat{\varphi}_i + \beta\hat{\varphi}'_i\|_{H^1(\hat{T})} \leq \alpha\|\hat{\varphi}_i\|_{H^1(\hat{T})} + \beta\|\hat{\varphi}_i\|_{H^1(\hat{T})} \leq (\alpha+\beta)C = C.$$

For two quadruples of functions $\{(\hat{\varphi}_1, \hat{\varphi}_2, \hat{\varphi}_3, \hat{\varphi}_4)^T\}$ and $\{(\hat{\varphi}'_1, \hat{\varphi}'_2, \hat{\varphi}'_3, \hat{\varphi}'_4)^T\}$ satisfying equality constraints No. 2 and No. 3, their convex combination $\{\alpha\hat{\varphi}_i + \beta\hat{\varphi}'_i\}_{i=1}^4$, $\alpha + \beta = 1$, $\alpha, \beta \in \mathbb{R}_0^+$, satisfies conditions No. 2 and No. 3 too. Thus, $\mathcal{G}$ is convex and weakly precompact. Since $\mathcal{G}$ is closed in $[H^1(\hat{T})]^4$ and convex, it is weakly closed, hence weakly compact in $[H^1(\hat{T})]^4$. By Rellich's theorem, $\mathcal{G}$ is compact in $[L_2(\hat{T})]^4$.

Summing up, Φ is a continuous positive functional on $\mathcal{G} \subset [L_2(\hat{T})]^4$, with $\mathcal{G}$ being a compact set. Thus, Φ attains its *positive* minimum on $\mathcal{G}$, proving our statement. □

Remark 21 Let $\hat{\varphi}_i$, $i = 1, \ldots, 4$ be basis functions satisfying assumptions of Lemma 8, G the corresponding $L_2(\hat{T})$-Gram matrix, and $\hat{u} = \sum_{i=1}^4 u_i\hat{\varphi}_i$, $\mathbf{u} = (u_1, u_2, u_3, u_4)^T \in \mathbb{R}^4$. Then,

$$\|\hat{u}\|^2_{L_2(\hat{T})} = \left(\sum_{i=1}^4 u_i\hat{\varphi}_i, \sum_{j=1}^4 u_j\hat{\varphi}_j\right)_{L_2(\hat{T})} = \sum_{i=1}^4\sum_{j=1}^4 (\hat{\varphi}_i, \hat{\varphi}_j)_{L_2(\hat{T})} u_i u_j = \langle G\mathbf{u}, \mathbf{u}\rangle. \tag{3.56}$$

Hence, by Lemma 8 it follows that

$$\|\hat{u}\|^2_{L_2(\hat{T})} = \|\mathbf{u}\|^2_G \geq \lambda_{min}(G)\|\mathbf{u}\|^2 \geq c\|\mathbf{u}\|^2. \tag{3.57}$$

At the same time, (3.56) gives

$$\|\hat{u}\|^2_{L_2(\hat{T})} \leq \lambda_{max}(G)\|\mathbf{u}\|^2, \tag{3.58}$$

where

$$|g_{ij}| = (\hat{\varphi}_i, \hat{\varphi}_j)_{L_2(\hat{T})} \leq \|\hat{\varphi}_i\|_{L_2(\hat{T})}\|\hat{\varphi}_j\|_{L_2(\hat{T})} \leq \|\hat{\varphi}_i\|_{H^1(\hat{T})}\|\hat{\varphi}_j\|_{H^1(\hat{T})} \leq C.$$

Thus, by Gershgorin's theorem,

$$\lambda_{max}(G) \leq C.$$

This bound, (3.58), and coercivity estimate (3.57) yield uniform norm equivalence

$$c\|\mathbf{u}\|^2 \leq \|\hat{u}\|^2_{L_2(\hat{T})} \leq C\|\mathbf{u}\|^2$$

with constants $C \geq c > 0$ dependent exclusively on constant C in (3.52).

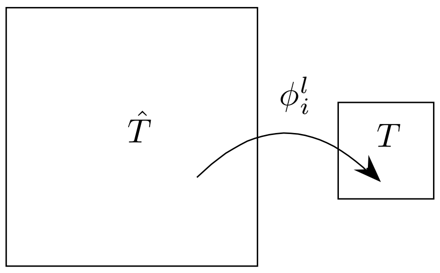


Figure 3.9: The macroelement transformation

Lemma 9 *Consider the affine mapping $\phi(\cdot) : \mathbb{R}^2 \to \mathbb{R}^2$ that maps unit square $\hat{T}$ onto square T with the side-length H and the left lower vertex located at $\mathbf{b} \in \mathbb{R}^2$. The mapping is given by*

$$\phi(\hat{\mathbf{x}}) = \begin{bmatrix} H & 0 \\ 0 & H \end{bmatrix} \hat{\mathbf{x}} + \mathbf{b}. \tag{3.59}$$

See Fig. 3.9. Let $u \in H^1(T)$. Define the transformed function $\hat{u} \in H^1(\hat{T})$ by

$$\hat{u}(\hat{\mathbf{x}}) = u(\phi(\hat{\mathbf{x}})). \tag{3.60}$$

Then it holds that

$$\|\hat{u}\|_{L_2(\hat{T})} = H\|u\|_{L_2(T)} \tag{3.61}$$

$$|\hat{u}|_{H^1(\hat{T})} = |u|_{H^1(T)}. \tag{3.62}$$

Proof. The proof follows by the elementary transformation of the integrals. □

Remark 22 Let T_i^l be an interior macroelement. From Lemma 7 and Lemma 9 it follows that the associated basis functions $\varphi_{i_j}^l$, $j = 1, \ldots, 4$, transformed by the mapping (3.59) via (3.60) (that is, resulting functions $\hat{\varphi}_j$, $j = 1, \ldots, 4$) satisfy the assumptions of Lemma 8. Indeed, by Lemma 7 it follows that $\|\varphi_{i_j}^l\|_{H^1(T_i^l)} \leq C$. Hence by Lemma 9 it follows that $\|\hat{\varphi}_j\|_{H^1(\hat{T})} \leq C$. Assumptions No. 2 and No. 3 of Lemma 8 follow from properties No. 2 and No. 3 proved in Lemma 7.

Lemma 10 *Define the level l interpolation operator*

$$\pi_l : \mathbf{x} \in \mathbb{R}^{n_l} \mapsto \sum_{i=1}^{n_l} x_i \varphi_i^l, \; l = 1, \ldots, L. \tag{3.63}$$

There are positive constants $C \geq c$ independent of mesh-size h and level l such that level $l = 1, \ldots, L$ and every $\mathbf{u} \in \mathbb{R}^{n_l}$, the following norm equivalence holds:

$$ch_l\|\mathbf{u}\| \leq \|\pi_l \mathbf{u}\|_{L_2(\Omega)} \leq Ch_l\|\mathbf{u}\|. \tag{3.64}$$

Proof. Let us prove first the left inequality of (3.64). Define Ω_{int}^l to be the union of all macroelements T_i^l that are not adjacent the boundary with the essential boundary condition, and $\mathcal{T}_i^l$ to be the set of indices of basis functions associated with macroelement T_i^l. Assume T_i^l is an interior macroelement. The entries of the set $\mathcal{T}_i^l = \{j_1, j_2, j_3, j_4\}$ are ordered in the same way as vertices in Fig. 3.8.

Let us consider the affine mapping ϕ_i^l that maps the unit square $\hat{T}$ onto T_i^l as in Lemma 9. Consider a function $u = \pi_l \mathbf{u}$, $\mathbf{u} \in \mathbb{R}^{n_l}$. Clearly,

$$u = \sum_{j \in \mathcal{T}_i^l} u_j \varphi_j^l \text{ on } T_i^l.$$

Define the transformed function

$$\hat{u}(\hat{\mathbf{x}}) = u(\phi_i^l(\hat{\mathbf{x}})), \quad \hat{\mathbf{x}} \in \hat{T}.$$

Further, define the transformed basis

$$\hat{\varphi}_k(\hat{\mathbf{x}}) = \varphi_{j_k}^l(\phi_i^l(\hat{\mathbf{x}})), \ k = 1, \dots, 4.$$

Then,

$$\hat{u}(\hat{\mathbf{x}}) = \sum_{k=1}^4 u_{j_k} \hat{\varphi}_k(\hat{\mathbf{x}}). \tag{3.65}$$

By Remark 22, transformed basis functions $\{\hat{\varphi}_k\}_{k=1}^4$ satisfy the assumptions of Lemma 8. Hence, denoting $G = \{(\hat{\varphi}_i, \hat{\varphi}_j)_{L_2(\hat{T})}\}_{i,j=1}^4$, the Gram matrix corresponding to the transformed basis, we have the estimate

$$\left\| \sum_{i=1}^4 w_i \hat{\varphi}_i \right\|_{L_2(\hat{T})}^2 = \langle G\mathbf{w}, \mathbf{w} \rangle \geq \lambda_{min}(G) \|\mathbf{w}\|^2 \geq c \sum_{i=1}^4 w_i^2 \quad \forall \mathbf{w} \in \mathbb{R}^4.$$

See Remark 21. By this inequality, (3.65), and Lemma 9, it follows that

$$\|u\|_{L_2(T_i^l)}^2 = h_l^2 \|\hat{u}\|_{L_2(\hat{T})}^2 = h_l^2 \left\| \sum_{k=1}^4 u_{j_k} \hat{\varphi}_k \right\|_{L_2(\hat{T})}^2 \geq c h_l^2 \sum_{k=1}^4 u_{j_k}^2 = c h_l^2 \sum_{j \in \mathcal{T}_i^l} u_j^2.$$

Since each degree of freedom i belongs to at least one set $\mathcal{T}_j^l$, the previous inequality gives

$$\|\mathbf{u}\|^2 \leq \sum_{\forall T_i^l \subset \Omega_{int}^l} \sum_{j \in \mathcal{T}_i^l} u_j^2 \leq C^{-1} h_l^{-2} \sum_{\forall T_i^l \subset \Omega_{int}^l} \|u\|_{L_2(T_i^l)}^2 \leq C^{-1} h_l^{-2} \|u\|_{L_2(\Omega)}^2,$$

completing the proof.

The second inequality of (3.64) is more or less trivial. Define the global Gram matrix

$$G^l = \{(\varphi_i^l, \varphi_j^l)_{L_2(\Omega)}\}_{i,j=1}^{n_l}.$$

The (minimal) constant C in the second inequality of (3.64) is then $\sqrt{\lambda_{max}(G^l)}$. (See Remark 21.) The matrix G^l contains at most 9 non-zeroes per row and the non-zeroes can be estimated by the Cauchy-Schwarz inequality and Lemma 7 by

$$|g_{ij}^l| \leq \|\varphi_i^l\|_{L_2(\Omega)}\|\varphi_j^l\|_{L_2(\Omega)} \leq Ch_l^2.$$

By Gershgorin's theorem, $\lambda_{max}(G^l) \leq Ch_l^2$ and the proof follows. □

Corollary 2 *For Gram matrix $M_l = (I_l^1)^T I_l^1$, $l = 1, \ldots, L$, corresponding to the discrete basis $\{I_l^1 \mathbf{e}_i^l\}_{i=1}^{n_l}$, the diagonal matrix $D_l = \mathrm{diag}(M_l)$ is uniformly spectrally equivalent in the sense that the equivalence*

$$c\|\mathbf{x}\|_{M_l} \leq \|\mathbf{x}\|_{D_l} \leq C\|\mathbf{x}\|_{M_l} \ \forall \mathbf{x} \in \mathbb{R}^{n_l} \tag{3.66}$$

holds with constants $C \geq c > 0$ independent of h and l. As a consequence, assumption (3.20) holds for $\tilde{\mathcal{Q}}_l = \tilde{Q}^l$ and $\mathcal{Q}_l = Q_l$ given by (4.5) and (4.6).

Proof. From (3.64) follows the uniform norm equivalence

$$c3^{l-1}\|\mathbf{x}\| \leq \|I_l^1\mathbf{x}\| = \|\mathbf{x}\|_{M_l} \leq C3^{l-1}\|\mathbf{x}\| \ \forall \mathbf{x} \in \mathbb{R}^{n_l}.$$

Hence M_l is well-conditioned. The eigenvalues of D_l satisfy

$$\lambda_i(D_l) \equiv (D_l)_{ii} = \|\mathbf{e}_i^l\|_{M_l}^2 = \frac{\|\mathbf{e}_i^l\|_{M_l}^2}{\|\mathbf{e}_i^l\|^2} \in [\lambda_{min}(M_l), \lambda_{max}(M_l)] \subset [c3^{l-1}, C3^{l-1}],$$

proving (3.66).

From (3.66) follows $C^{-1}\|\cdot\|_{M_l^{-1}} \leq \|\cdot\|_{D_l^{-1}} \leq c^{-1}\|\cdot\|_{M_l^{-1}}$ and therefore

$$\begin{aligned} C^{-1}\langle Q_l\mathbf{u}, \mathbf{u}\rangle &= C^{-1}\langle M_l^{-1}(I_l^1)^T\mathbf{u}, (I_l^1)^T\mathbf{u}\rangle \leq \langle D_l^{-1}(I_l^1)^T\mathbf{u}, (I_l^1)^T\mathbf{u}\rangle \\ &= \langle \tilde{Q}_l\mathbf{u}, \mathbf{u}\rangle \leq c^{-1}\langle M_l^{-1}(I_l^1)^T\mathbf{u}, (I_l^1)^T\mathbf{u}\rangle = c^{-1}\langle Q_l\mathbf{u}, \mathbf{u}\rangle \ \forall \mathbf{u} \in U \end{aligned}$$

with constants $C \geq c > 0$ from (3.66), proving assumption (3.20) of Theorem 19. □

Lemma 11 (Scaled Poincaré and Friedrich's inequality) *Let T be a square domain with side of length H. Then there is a constant $C > 0$ independent of H (and, characteristic for a square) such that (Poincaré inequality)*

$$\inf_{c \in \mathbb{R}} \|u - c\|_{L_2(T)} \leq CH|u|_{H^1(T)} \quad \forall u \in H^1(T). \tag{3.67}$$

and (Friedrich's inequality)

$$\|u\|_{L_2(T)} \leq CH|u|_{H^1(T)} \quad \forall u \in H^1_{0,\Gamma}(T) \equiv \{u \in H^1(T) : \mathrm{tr}\ u = 0 \ on\ \Gamma\}, \tag{3.68}$$

if $\Gamma \subset \partial T$ contains at least one edge of T.

Proof. The proof follows from Poincaré and Friedrich's inequalities on a unit square by scaling, using Lemma 9. $\square$

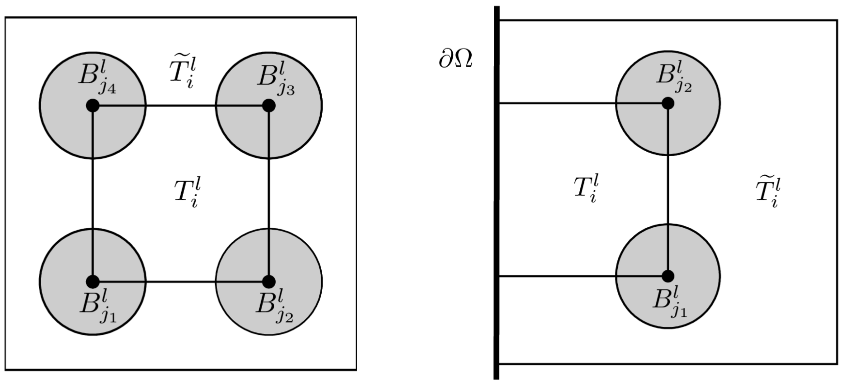


Figure 3.10: Extended macroelement

For each vertex $\mathbf{v}^l_i$ we introduce a ball $B^l_i \subset \Omega$ with center in $\mathbf{v}^l_i$ that has measure about $\mu(T^l_j)$, in the sense that there are constants $C \geq c > 0$ independent of mesh-size h, level l, and basis function number i such that

$$ch_l^2 \leq \mu(B^l_i) \leq Ch_l^2, \tag{3.69}$$

see Fig. 3.10. For convenience, we assume that the balls B^l_i do not intersect each other. We encapsulate each domain

$$T^l_j \cup \bigcup_{i \in \mathcal{T}^l_j} B^l_i$$

into (the nearly smallest possible) square $\tilde{T}^l_j$. Clearly, the intersections of the extended macroelements $\tilde{T}^l_j$ are bounded, that is, there is finite integer N independent of level such that each $\mathbf{x} \in \Omega$ belongs to at most N extended macroelements $\tilde{T}^l_i$.

Both for the interior and for the boundary macroelement, define the local interpolation operator $\Pi^l_i : L_2(\tilde{T}^l_i) \to L_2(T^l_i)$ by

$$\Pi^l_i u = \sum_{j \in \mathcal{T}^l_i} \left(\frac{1}{\mu(B^l_j)} \int_{B^l_j} u d\mathbf{x} \right) \varphi^l_j. \tag{3.70}$$

Here, $\mathcal{T}^l_i$ is an index set of the numbers of basis functions associated with T^l_i.

Next we prove $L_2(\Omega)$-stability of the local interpolation operator.

Lemma 12 *Both for the interior and for the boundary macroelement T_i^l, the mapping Π_i^l is stable in the L_2-norm, in the sense that*

$$\|\Pi_i^l u\|_{L_2(T_i^l)} \le C\|u\|_{L_2(\widetilde{T}_i^l)} \quad \forall u \in L_2(\widetilde{T}_i^l) \tag{3.71}$$

with positive constant C independent of the level l, mesh-size h, and macroelement number i.

Proof. We estimate using the definition of Π_i^l in (3.70), the triangle inequality, the Cauchy-Schwarz inequality, L_2-bound (3.34), and (3.69). We have

$$\begin{aligned}
\|\Pi_i^l u\|_{L_2(T_i^l)} &= \left\| \sum_{j\in\mathcal{T}_i^l} \left(\frac{1}{\mu(B_j^l)} \int_{B_j^l} u d\mathbf{x} \right) \varphi_j^l \right\|_{L_2(T_i^l)} \\
&\le \sum_{j\in\mathcal{T}_i^l} \left(\frac{1}{\mu(B_j^l)} \int_{B_j^l} u d\mathbf{x} \right) \|\varphi_j^l\|_{L_2(T_i^l)} \\
&\le Ch_l \sum_{j\in\mathcal{T}_i^l} \frac{1}{\mu(B_j^l)} (u,1)_{L_2(B_j^l)} \\
&\le Ch_l \sum_{j\in\mathcal{T}_i^l} \frac{1}{\mu(B_j^l)} \|u\|_{L_2(B_j^l)} \|1\|_{L_2(B_j^l)} \\
&\le C \sum_{j\in\mathcal{T}_i^l} \|u\|_{L_2(B_i^l)}.
\end{aligned}$$

Further, by the Cauchy-Schwarz inequality,

$$\sum_{j\in\mathcal{T}_i^l} \|u\|_{L_2(B_j^l)} \le \left(\sum_{j\in\mathcal{T}_i^l} \|u\|^2_{L_2(B_j^l)} \right)^{1/2} \left(\sum_{j\in\mathcal{T}_i^l} 1^2 \right)^{1/2} \le 2\|u\|_{L_2(\tilde{T}_i^l)}.$$

The proof follows from last two estimates. □

The proof of the following lemma uses the key argument of finite element approximation theory.

Lemma 13 *For both the interior and the boundary macroelement T_i^l, the interpolation operator Π_i^l defined in (3.70) satisfies the estimate*

$$\|u - \Pi_i^l u\|_{L_2(T_i^l)} \le Ch_l |u|_{H^1(\tilde{T}_i^l)} \quad \forall u \in H^1_{0,\partial\tilde{T}_i^l\cap\partial\Omega}(\tilde{T}_i^l) \tag{3.72}$$

with constant $C > 0$ independent of h, l, and i. In addition, the interpolation operator

$$\Pi^l : u \in H^1(\Omega) \mapsto \sum_{i=1}^{n_l} \left(\frac{1}{\mu(B_1^l)} \int_{B_i^l} u d\mathbf{x} \right) \varphi_i^l \tag{3.73}$$

satisfies

$$\|u-\Pi^l u\|_{L^2(\Omega)} \le Ch_l|u|_{H^1(\Omega)} \quad \forall u \in H^1_0(\Omega) \tag{3.74}$$

with constant $C>0$ independent of h and l.

Proof. By Lemma 12 it follows that

$$\|I-\Pi^l_i\|_{L^2(\tilde{T}^l_i)\to L_2(T^l_i)} \le \|I\|_{L^2(\tilde{T}^l_i)\to L_2(T^l_i)} + \|\Pi^l_i\|_{L^2(\tilde{T}^l_i)\to L_2(T^l_i)} \le C. \tag{3.75}$$

Let T^l_i be an interior macroelement and $\mathcal{T}^l_i$ the set of basis functions associated with T^l_i. By Lemma 7 it follows that

$$\sum_{j\in\mathcal{T}^l_i} \varphi^l_i = 1 \text{ on } T^l_i.$$

Hence, for any constant function c defined on $\tilde{T}^l_i$ it holds that

$$\Pi^l_i c = \sum_{j\in\mathcal{T}^l_i} \left(\frac{1}{\mu(B^l_i)} \int_{B^l_i} c d\mathbf{x} \right) \varphi^l_j = c \sum_{j\in\mathcal{T}^l_i} \varphi^l_j = c \text{ on } T^l_i. \tag{3.76}$$

Let $u \in H^1(\tilde{T}^l_i)$. By (3.76) and (3.75), for any constant c it holds that

$$\begin{aligned}
\|u-\Pi^l_i u\|_{L_2(T^l_i)} &= \|u-c-(\Pi^l_i u - c)\|_{L_2(T^l_i)} \\
&= \|u-c-(\Pi^l_i u - \Pi^l_i c)\|_{L_2(T^l_i)} \\
&= \|(I-\Pi^l_i)(u-c)\|_{L_2(T^l_i)} \\
&\le \|I-\Pi^l_i\|_{L^2(\tilde{T}^l_i)\to L_2(T^l_i)} \|u-c\|_{L_2(\tilde{T}^l_i)} \\
&\le C\|I-\Pi^l_i\|_{L^2(\tilde{T}^l_i)\to L_2(T^l_i)} \|u-c\|_{L_2(\tilde{T}^l_i)}.
\end{aligned}$$

In the above estimate we choose

$$c = \text{argmin}_{q\in\mathbb{R}} \|u-q\|_{L_2(\tilde{T}^l_i)}.$$

Hence, by the previous inequality and the scaled Poincaré inequality (3.67), (3.72) follows.

To prove (3.72) for a boundary macroelement is even simpler. Let $u \in H^1_{0,\partial\Omega\cap\partial\tilde{T}^l_i}(\Omega)$. We use (3.75) and the scaled Friedrich's inequality (3.68) to estimate

$$\|u-\Pi^l_i u\|_{L_2(T^l_i)} \le \|I-\Pi^l_i\|_{L^2(\tilde{T}^l_i)\to L_2(T^l_i)} \|u\|_{L_2(\tilde{T}^l_i)} \le Ch_l|u|^2_{H^1(\tilde{T}^l_i)}.$$

This completes the proof of (3.72).

To prove (3.74) we use the obvious identity

$$\|u - \Pi^l u\|_{L_2(T_i^l)} = \|u - \Pi_i^l u\|_{L_2(T_i^l)},$$

the local estimate (3.72), and the fact that every point $\mathbf{x} \in \Omega$ belongs to at most $N < +\infty$ extended macroelements $\tilde{T}_i^l$. Thus,

$$\begin{aligned}
\|u - \Pi^l u\|^2_{L^2(\Omega)} &= \sum_{\forall T_i^l \subset \Omega} \|u - \Pi^l u\|^2_{L_2(T_i^l)} \\
&= \sum_{\forall T_i^l \subset \Omega} \|u - \Pi_i^l u\|^2_{L_2(T_i^l)} \\
&\leq C \sum_{\forall T_i^l \subset \Omega} h_l^2 |u|^2_{H^1(\tilde{T}_i^l)} \\
&\leq C h_l^2 |u|^2_{H^1(\Omega)}.
\end{aligned}$$

□

Lemma 14 *There is a constant* $c_\sigma > 0$ *independent of* h*,* l *and* L *such that*

$$\bar{\sigma}_l \equiv \frac{c_\sigma}{9^{l-1}} \geq \sigma_l \tag{3.77}$$

for all $l = 1, \ldots, L$*. In addition, there is a constant* $C > 0$ *independent of* h*,* l*, and* L *such that for every* $\mathbf{u} \in U = \mathbb{R}^{n_1}$*, the exact orthogonal projections* $\mathcal{Q}_l = Q_l : U \to U_l$*,* $U_{L+1} = \emptyset$*,* $Q_{L+1} = 0$*, satisfy*

$$\|\mathbf{u} - Q_{l+1}\mathbf{u}\| \leq \frac{C}{\sqrt{\bar{\sigma}_l}} \|\mathbf{u}\|_A, \quad l = 1, \ldots, L. \tag{3.78}$$

Proof. To estimate the spectral bound (3.77) we use the norm equivalence proved in Lemma 10. By (4.8),

$$\sigma_l = \sup_{\mathbf{x} \in \mathbb{R}^{n_l} \setminus \{\mathbf{0}\}} \frac{\langle A I_l^1 \mathbf{x}, I_l^1 \mathbf{x} \rangle}{\|I_l^1 \mathbf{x}\|^2} = \sup_{\mathbf{x} \in \mathbb{R}^{n_l} \setminus \{\mathbf{0}\}} \frac{\langle A_l \mathbf{x}, \mathbf{x} \rangle}{\|I_l^1 \mathbf{x}\|^2},$$

where by (3.64) and $\pi_l = \pi_1 I_l^1$,

$$\|I_l^1 \mathbf{x}\|^2 \geq c h_1^{-2} \|\pi_1 I_l^1 \mathbf{x}\|^2 = c h_1^{-2} \|\pi_l \mathbf{x}\|^2_{L_2(\Omega)} \geq c \left(\frac{h_l}{h_1}\right)^2 \|\mathbf{x}\|^2 \ \forall \mathbf{x} \in \mathbb{R}^{n_l}. \tag{3.79}$$

The previous two inequalities, together with (3.35), give

$$\sigma_l \leq C \left(\frac{h_1}{h_l}\right)^2 \sup_{\mathbf{x} \in \mathbb{R}^{n_l} \setminus \{\mathbf{0}\}} \frac{\langle A_l \mathbf{x}, \mathbf{x} \rangle}{\|\mathbf{x}\|^2} \leq C \left(\frac{h_1}{h_l}\right)^2 \lambda_{max}(A_l) \leq C \left(\frac{h_1}{h_l}\right)^2,$$

proving (3.77).

We will prove (3.78) for approximate projections $\bar{Q}_l : U \to U_l$ defined by $\bar{Q}_l = \pi_1^{-1}\Pi^l\pi_1$, $l = 2, \ldots, L$ and $\bar{Q}_{L+1} = 0$. The result for the exact projection then follows by minimizing property of the orthogonal projection.

Let $l < L$ and $\mathbf{u} \in U$. We estimate using Lemma 13, norm equivalence (3.39), and $h_{l+1} = 3h_l$,

$$\begin{aligned}
\|\mathbf{u} - \bar{Q}_{l+1}\mathbf{u}\| &= \|(I - \pi_1^{-1}\Pi^{l+1}\pi_1)\mathbf{u}\| \\
&\leq Ch_1^{-1}\|\pi_1(I - \pi_1^{-1}\Pi^{l+1}\pi_1)\mathbf{u}\|_{L_2(\Omega)} \\
&= Ch_1^{-1}\|(I - \Pi^{l+1})\pi_1\mathbf{u}\|_{L_2(\Omega)} \\
&\leq C\frac{h_{l+1}}{h_1}|\pi_1\mathbf{u}|_{H^1(\Omega)} \\
&\leq C\frac{h_l}{h_1}|\pi_1\mathbf{u}|_{H^1(\Omega)} \\
&= C\frac{h_l}{h_1}\|\mathbf{u}\|_A. \qquad (3.80)
\end{aligned}$$

For $l = L$, we have by $h_L = 1/2$ (since there is only one degree of freedom located in the center of Ω), and by Friedrich's inequality (3.68) for $T = \Omega$,

$$\|\mathbf{u} - \bar{Q}_{L+1}\mathbf{u}\| = \|\mathbf{u}\| \leq Ch_1^{-1}\|\pi_1\mathbf{u}\|_{L_2(\Omega)} \leq h_1^{-1}C|\pi_1\mathbf{u}|_{H^1(\Omega)} \leq C\frac{h_L}{h_1}\|\mathbf{u}\|_A,$$

proving (3.80) for $l = L$.

Statement (3.78) follows from (3.77) and estimate (3.80). □

Now we are ready to formulate the final convergence theorem.

Theorem 23 *For model problem (3.28) and smoothed aggregation based coarse-spaces $U_l =$ Range(I_l^1) with prolongators I_{l+1}^l as defined in Section 3.5, the BPX preconditioner $\mathcal{B} = B$ given by Algorithm 3.4.1 satisfies the estimate*

$$\frac{c}{L}\|\mathbf{u}\|_A^2 \leq \langle BA\mathbf{u}, \mathbf{u}\rangle_A \leq CL\|\mathbf{u}\|_A^2 \ \forall \mathbf{u} \in U \qquad (3.81)$$

with constants $C \geq c > 0$ independent of h and L.

Proof. The proof consists in the verification of the assumptions of Theorem 19. Assumption (3.19) follows from Lemma 14. Assumption (3.20) holds by virtue of Corollary 2. □

Remark 24 (Choice of $\bar{\sigma}_l$.) In practice, it is relatively difficult to determine upper bounds $\bar{\sigma}_l \geq \sigma_l$ in (3.27) computationally. From Lemma 14, we know that there is a constant $c_\sigma > 0$ independent of h, l, and L such that

$$\bar{\sigma}_l \equiv \frac{c_\sigma}{9^{l-1}} \geq \sigma_l, \ l = 1, \ldots, L.$$

To get an efficient preconditioner, it is not neccessary to determine the constant c_σ. In (4.7), it is sufficient to use

$$\tilde{\sigma}_l = \frac{1}{9^{l-1}},\ l = 1, \dots, L$$

in the place of $\bar{\sigma}_l = c_\sigma/9^{l-1}$. Obviously, this leads to the scalar multiple $\tilde{B} = c_\sigma^{-1}B$. This simplification does not alter the convergence estimate since

$$\operatorname{cond}(A, \tilde{B}) = \operatorname{cond}(A, c_\sigma^{-1}B) = \operatorname{cond}(A, B) \le \frac{C}{c}L^2.$$

Here, C, c are constants from (3.81).

Remark 25 By the means very similar to those used in Sect. 3.6, for operators $\tilde{Q}_l$, it is possible to verify assumptions of the abstract convergence result of [6] with uniform constants. Then, for a standard multiplicative multigrid, we get the estimate of the convergence rate in the energy norm in the form $1 - C/L$. Compared to the former result of [53], where the convergence rate deteriorates with the power of 3 of L, this is an improvement.

Chapter 4

Implementation and Numerical Experiments

In this chapter, we focus on the numerical aspects of algorithms presented in the previous text. The main aim of this chapter is to do a series of numerical experiments to prove a validity of the theoretical convergence results of the SA BPX algorithm.

Implementation is done in software Matlab and programming language Fortran 90. Before we proceed to the numerical results, we describe all the essential parts that have to be dealt with while programming multigrid algorithm. The first task is to consider carefully which sparse storage format we will use, since we work with large sparse matrices. Then, we make use of the naturally parallel characteristic of the SA BPX algorithm and implement it in parallel (OpenMP parallelization in Fortran). We have to take the algorithm to parts to find out which parts of the multigrid algorithm are suitable to be done in parallel and which are not. As a solver, we use the preconditioned conjugated gradients method. Finally in the end of the chapter, we present a series of numerical experiments.

4.1 Sparse Matrices Storage Formats

Matrices coming from the finite elements discretizations are sparse and potentially large. It is essential to store them in some sparse matrix format. To store an information about an element, we keep usually three arrays with an information about a value and a placement of the element instead of keeping two-dimensional dense array.

In our implementation, we use three storage formats: coordinate, compressed sparse row and compressed sparse column.

Coordinate format

The coordinate format (COO) consists of three real arrays. An array a contains entries of the matrix and two integer arrays ai and aj contain row indices and column indices.

The order of entries is stored in arrays is arbitrary.

Example 1

An example of COO format might look like this:

$$A = \begin{pmatrix} 0 & 1 & 0 & 2 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 5 & 6 \\ 0 & 0 & 7 & 0 & 0 \end{pmatrix},$$

then

$$a = [1, 2, 3, 4, 5, 6, 7],$$

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

$$aj = [2, 3, 3, 4, 3, 4, 5].$$

CSR Structure

The storage of nonzero elements can be done such that we go across all the rows and we store all the elements in the order how they appeared. We denote nnz as the number of nonzero elements. In compressed sparse row format (CSR), the three arrays have following meaning:

a:	a real array of length nnz containing nonzero elements of A,
ai :	an integer array of positions in the column,
aj :	an integer array, where i-th entry points to the beginning of the i-th row in the array ai. The length of the array is $N + 1$, where N is number of rows.

The array aj from previous example is now

$$aj = [1, 3, 4, 4, 7, 8].$$

CSC Structure

The compressed sparse column format (CSC) is similar to the CSR format. In this case we go across columns instead of rows.

Remark 26 In Matlab software, the sparse matrices are created in COO format. In our Fortran implementation we use mainly CSR format (or CSC, if it is convenient). The implementation has to be treated carefully since for some cases it is more practical to use CSR format and in some CSC format. It will be explained later.

4.2 Multigrid Programming Essentials

In every program based on the multigrid algorithm, we have to overcome some problems arising from the algorithm. To be more specific, let us assume we would like to solve a system $A\mathbf{u} = \mathbf{f}$. Then, before we progress to the multigrid solver, we have to create a system of prolongator and restriction operators and representation of A on every level l, $l = 1, \dots, L$. The representation of A on the coarse levels is done by so called *Galerkin principle*: $A_{l+1} = (I^l_{l+1})^{\mathrm{T}} A_l I^l_{l+1}$ (we continue using AMG numbering). In algebraic multigrid, prolongators and coarse matrices A_l are created by algebraic principles and without knowing the exact differential operator.

After we create all the multigrid components on every level, we proceed to the actual solver. Both algorithms can be therefore implemented in two steps, *set-up phase* and *solver phase.*

The set-up phase

Let us look more closely at the set-up phase of an AMG multigrid, particularly at the SA algorithm. It can be summarized in the following scheme:

```
For $l = 1, L$ do
  1.)  coarsening
       - partition fine level nodes into disjoint aggregates by a fast heuristic algorithm
  2.)  definition of prolongators and restrictions
       - define tentative prolongator $P^l_{l+1}$ such that nonzero structure of the $j$-th column corresponds to the $j$-th aggregate
       - define smoothed prolongators $I^l_{l+1}$ by applying one smoothing step on tentative prolongator: $I^l_{l+1} = S_l P^l_{l+1}$
       - define restriction as $(I^l_{l+1})^{\mathrm{T}}$
  3.)  definition of coarse matrices
       $A_{l+1} = (I^l_{l+1})^{\mathrm{T}} A_l I^l_{l+1}$
```

We would like to develop some parts of the set-up phase a bit more:

add 1.) We create aggregates by means of the greedy algorithm described in Alg. 3.2.3. This process works with the conception of *strongly coupled neighborhood* (3.9) which, among others, ensures that the structure of aggregates will follow semi-coarsening pattern for anisotropic problems. The aggregates are created by successive colouring the adjacency graph of matrix A by three sweeps of the greedy method in total.

add 2.) The tentative prolongator is created column-wise by aggregates. Smoothing of the prolongator is usually done by applying one step of a smoother S which is in a polynomial of A. The exception could be problems with aggressive coarsening, see [51]. We have to keep in mind that by smoothing of the prolongator, we have to ensure the nonzero pattern. Adding more nonzero elements to the system would slower whole algorithm.

add 3.) Regardless of whether we use classical or smoothed aggregation AMG, the computation of coarse matrices involves a product of three matrices.

Remark 27 We would like to mention that we are interested in the parallelization of the additive SA-BPX preconditioner and we will not go deeper into the possibilities of parallelization of the set up phase. Depending on the architecture, there are plenty of options of how to implement smoothers, aggregates and coarse matrices in parallel. Reader can see [48], [18], [59] and [30].

The solver phase

After the set-up phase, we proceed to solving a system $A\mathbf{u} = \mathbf{f}$. The multiplicative and additive multigrid have following steps:

1.) *smoothing*
Smoothing is done by some simple iterative method like damped Jacobi method or in the case of SA-BPX the smoother R_l is simply $\lambda_l^{-1}I$.

2.) *restriction*
The operation of restriction and prolongation is a matrix-vector multiplication. In parallel implementation the multiplication of $A\mathbf{x}$ can be done naturally in parallel for a matrix in CSR format. The situation changes, if we consider transposed matrix - vector multiplication. The rows of the sparse matrix becomes its columns and the multiplication therefore cannot be treated in the same way as matrix times vector. We have to think if we want to store both restriction and prolongation matrices in matrix formats

because it is convenient for our parallel implementation or we store only one of them and execute transposed matrix - vector multiplication (it will be discussed in the next section).

3.) *solver*
In the classical multiplicative multigrid on the coarsest level we solve the system, if the system is small, we use a direct method.

4.) *prolongation* (add 2.)

In the additive multigrid, the smoothing and restriction can be done in parallel which will be described later.

4.2.1 Parallel Programming Models

While studying algorithms involved in multigrid, we find out that some parts could be executed independetly on other parts in parallel. In this section we make a brief introduction into the concept of parallel programming.

We now assume a matrix times vector multiplication $y = Ax$.

```
for $i = 1$ to $n$ do
    $sum = 0$
    for $j = 0$ to $n$ do
        $sum = sum + a(i,j)x(j)$
    end for
    $y(i) = sum$
end for
```

The parallelization can be done such that each threads is assigned by a few of rows of the matrix and then the computations for inter loop are done sequentially.

As an example we take an $n = 6$ and divide the tasks into two threads:

```
thread 1:
for $i = 1,2,3$ do
    $sum = 0$
    for $j = 0$ to $n$ do
        $sum = sum + a(i,j)x(j)$
    end for
    $y(i) = sum$
end for
thread 2:
for $i = 4,5,6$ do
    $sum = 0$
    for $j = 0$ to $n$ do
        $sum = sum + a(i,j)x(j)$
    end for
    $y(i) = sum$
  end for
```

We have to be careful when we want to compute a summation in parallel. If we decided to parallelize the $sum = sum + b(j)$ for $j = 1, \dots, N$ similarly like in the example, each of the threads would be assigned by part of vector b and then it would sum the elements of b. But since sum is depending on its value from a previous loop, the results given by each thread would be therefore wrong.

We can ask, if the doubling the number of cores would lead to halving of the time needed to the computations.

Amdahl's Law states that potential program speedup is defined by the fraction of code p that can be parallelized:

$$\text{speedup} = \frac{1}{(1-p)}.$$

However, certain problems can involve increasing speedup by increasing the problem size. Problems that increase the percentage of parallel time with their size are more scalable than problems with a fixed percentage of parallel time.

4.2.2 Parallelization with OpenMP

Some of the problems offer the possibility to be divided into discrete parts that will be treated concurrently. The instructions are then assigned to each part and executed simultaneously:

1. Single Instruction
   The most simple type of computer performs one instruction (such as reading from memory, addition of two values) per cycle, with only one set of data.

2. Multiple Instruction
   This applies to all computers which contain only on processing core (with multi-core CPUs, single-CPU systems can have more than one processing core, making them MIMD systems). Combining several processing cores or processors, a computer can process several instructions and data sets per cycle.

The system architectures are generally:

1. Shared Memory
   In these systems with a shared memory, all processors are connected to a common memory. Usually all the processors are identical and have equal memory access.

2. Distributed Memory
   The processors are connected to each other and each processor has its own local memory. The demands imposed on the communication network are lower than in the case of a shared memory system, as the communication between processors may be slower than the communication between processor and memory.

In the implementation we will use the OpenMP interface. We therefore describe the basics of programming with OpenMP and mention the directives, which we will need laters.

OpenMP is an Application Program Interface (API) that may be used to explicitly direct multi-threaded, this is in shared memory parallelism. Comprised of three primary API components: Compiler Directives, Runtime Library Routines and Environment Variables.

We will use the API specifications for Fortran.

Directives

In Fortran, OpenMP directives are specified by using special comments that are identified by unique sentinels. Compilers can therefore ignore OpenMP directives and conditionally compiled code if support of the OpenMP API is not provided or enabled.

OpenMP directives for Fortran are specified as follows:

```
sentinel directive-name [clause[[,] clause]...]
```

All OpenMP compiler directives must begin with a directive sentinel.

Free Source Form Directives

The following sentinel is recognized in free form source files:

```
!$omp
```

The sentinel can appear in any column as long as it is preceded only by white space (spaces and tab characters). It must appear as a single word with no intervening character.

parallel Construct

```
!$omp parallel [clause[[,] clause]...]
structured-block
!$omp end parallel
```

Description:
When a thread encounters a parallel construct, a team of threads is created to execute the parallel region. The thread that encountered the parallel construct becomes the master thread of the new team, with a thread number of zero for the duration of the new parallel region. All threads in the new team, including the master thread, execute the region. Once the team is created, the number of threads in the team remains constant for the duration of that parallel region.

Worksharing Constructs

The OpenMP API defines the following worksharing constructs, and these are described in the sections that follow:
loop construct
sections construct
single construct
workshare construct

Tasking Constructs

`master Construct`

The master construct specifies a structured block that is executed by the master thread of the team.

`critical Construct`

The critical construct restricts execution of the associated structured block to a single thread at a time.

```
!$omp critical [(name)]
structured-block
!$omp end critical [(name)]
```

`atomic Construct`

The atomic construct ensures that a specific memory update is accessed atomically.

Execution Environment Routines

The `omp_get_num_threads` routine returns the number of threads in the current team.

The `omp_set_num_threads` routine sets the number of threads for the parallel region.

The `omp_get_thread_num` returns the thread number, within the current team, of the calling thread.

4.3 Implementation of the Model Problem

In this section we describe the implementation of BPX in the smoothed aggregation settings (SA BPX) on the model problem.

Here, we want to remind what we mean by solving the model problem. The model problem has been defined before, but we repeat it for the reader's convenience and a brevity of the text.

Let $\Omega = (0,1) \times (0,1)$ be a computational domain. We consider a model problem

$$\text{find } u \in H_0^1(\Omega) : a(u,v) = f(v)\ \forall v \in H_0^1(\Omega). \tag{4.1}$$

Here, $a(\cdot,\ \cdot) = (\nabla\cdot, \nabla\cdot)_{L_2(\Omega)}$ and $f(\cdot) \in (H_0^1(\Omega))^{-1}$. The problem is discretized by $P1$ elements on a uniform triangular mesh obtained from a regular square mesh when each square is broken by connecting its left lower and right upper vertices with a straight edge.

The discretization leads to the system of linear algebraic equations

$$A\mathbf{x} = \mathbf{b},$$

with $n \times n$ symmetric, positive definite matrix A and $\mathbf{b} \in \mathrm{I\!R}^n$. Set $n_1 = n$.

The hierarchy of the aggregates is defined as $\{\mathcal{A}_i^l\}_{i=1}^{9^{L-l-1}}$, $l = 1, \ldots, L-1$ by grouping the mesh vertices into 3×3 regular, square groups. The tentative prolongator is then defined as

$$(P_{l+1}^l)_{ij} = \begin{cases} 1 & \text{for } i \in \mathcal{A}_j^l, \\ 0 & \text{otherwise} \end{cases}, \tag{4.2}$$

$i = 1, \ldots, n_l$, $j = 1, \ldots, n_{l+1}$, $n_l = 9^{L-l}$, $l = 1, \ldots, L-1$. We assume the system of injective linear prolongators

$$I_{l+1}^l : \mathrm{I\!R}^{n_{l+1}} \to \mathrm{I\!R}^{n_l} \quad n_{l+1} < n_l,\ l = 1, \ldots, L-1$$

is given. The system of prolongators is created by applying one smoothing step on the tentative prolongator:

$$I_{l+1}^l = S_l P_{l+1}^l,$$

where the prolongator smoothers S_l are defined by

$$S_l = I - \frac{4}{3}\frac{1}{\bar{\lambda}_l} A_l, \quad l = 1, \dots L-1. \tag{4.3}$$

We set $(U,\ (\cdot,\ \cdot)_U, ; \|\cdot\|_U)$ to be the Euclidean space $(\mathbb{R}^n,\ \langle\cdot,\ \cdot\rangle,\ \|\cdot\|)$ and

$$a(\cdot,\cdot) = \langle A\cdot, \cdot\rangle. \tag{4.4}$$

We introduce *composite prolongators*

$$I_l^1 = I_2^1 I_3^2 \dots I_l^{l-1},\ l = 1, \dots, L.$$

The coarse-spaces are defined by

$$U_l = \text{ Range}(I_l^1),\ l = 1, \dots, L$$

and coarse-level matrices by

$$A_l = (I_l^1)^T A I_l^1.$$

Note that the matrix A_l is the operator $\mathcal{A}_l$ defined by (4.4) and (3.32), represented with respect to the basis given by the columns of I_l^1. The exact projection operators in the matrix form are

$$\mathcal{Q}_l = Q_l = I_l^1 \left((I_l^1)^T I_l^1\right)^{-1} (I_l^1)^T,\ \ l = 1, \dots, L. \tag{4.5}$$

We choose the inexact projections to be the operators Q_l with the matrix $(I_l^1)^T I_l^1$ replaced by its diagonal, that is

$$\tilde{\mathcal{Q}}_l = \tilde{Q}_l = I_l^1 D_l^{-1} (I_l^1)^T,\ D_l = \text{diag}\left((I_l^1)^T I_l^1\right),\ \ l = 1, \dots, L. \tag{4.6}$$

The action of BPX preconditioner (3.17) is given by the following algorithm:

Algorithm 4.3.1

```
given: $\mathbf{x} \in \mathbb{R}^n$
1: evaluate the action $\mathbf{y} = B\mathbf{x} \in \mathbb{R}^n$ of preconditioner $B = \mathcal{B}$ by
```

$$\mathbf{y} = \frac{1}{\bar{\sigma}_1}\mathbf{x} + \sum_{l=2}^{L} \left(\frac{1}{\bar{\sigma}_l} - \frac{1}{\bar{\sigma}_{l-1}}\right) I_l^1 D_l^{-1} (I_l^1)^T \mathbf{x},\ D_l = \text{diag}\left((I_l^1)^T I_l^1\right). \tag{4.7}$$

In (4.7), $\bar{\sigma}_l$ is an upper bound of

$$\sigma_l = \lambda_{max}\left(\mathcal{A}_l\right) = \sup_{\mathbf{x} \in \text{ Range}(I_l^1) \setminus \{\mathbf{0}\}} \frac{\|\mathbf{x}\|_A^2}{\|\mathbf{x}\|^2} = \sup_{\mathbf{x} \in \mathbb{R}^{n_l} \setminus \{\mathbf{0}\}} \frac{\|I_l^1 \mathbf{x}\|_A^2}{\|I_l^1 \mathbf{x}\|^2},\ l = 1, \dots, L. \tag{4.8}$$

The choice of $\bar{\sigma}_l$ is, for our model example, in Remark 24.

Denote $\mathbf{f}_i^l$ to be the i-th column of I_l^1. Note that vectors $\{\mathbf{f}_i^l\}_{i=1}^{n_l}$ form a natural basis of $U_l =$ Range(I_l^1). We would like to implement $\mathbf{y} = \tilde{Q}_l\mathbf{x}$ in parallel. It holds that

$$\mathbf{y} = \tilde{Q}_l\mathbf{x} = I_l^1 D_l^{-1}(I_l^1)^T\mathbf{x},$$

$$\mathbf{y} = \mathbf{f}_1^l\langle\mathbf{f}_1^l, \mathbf{x}\rangle/D_l(1,1) + \mathbf{f}_2^l\langle\mathbf{f}_2^l, \mathbf{x}\rangle/D_l(2,2) + \cdots + \langle\mathbf{f}_{n_l}^l, \mathbf{x}\rangle/D_l(n_l, n_l).$$

It can be implemented using the parallel loop

$$\mathbf{y} = \mathbf{0};\ \text{for } i = 1, \dots, n_l \text{ do in parallel } \mathbf{y} \leftarrow \mathbf{y} + \frac{\langle\mathbf{x}, \mathbf{f}_i^l\rangle}{\|\mathbf{f}_i^l\|^2}\,\mathbf{f}_i^l.$$

The update of $\mathbf{y}$ is a critical section.

Algorithm 4.3.1 can be therefore implemented in parallel using the operation of sparse inner product $\langle\cdot,\cdot\rangle_{l_2(\mathcal{I})}$ as follows:

Algorithm 4.3.2

```
1: setup: given composite prolongators $I_l^1$, $l = 2, \dots, L$, set $\mathbf{f}_i^l$ to be the $i$-th
   column of $I_l^1$ and evaluate $D_{ii}^l$, $l = 2, \dots, L$, $i = 1, \dots, n_l$ as follows:
2: for all $l = 2, \dots, L$, $i = 1, \dots, n_l$ do in parallel
       set $D_{ii}^l = \langle\mathbf{f}_i^l, \mathbf{f}_i^l\rangle_{l_2(\mathrm{supp}(\mathbf{f}_i^l))}$
3: end for
4: given $\mathbf{x} \in \mathbb{R}^{n_1}$, evaluate $\mathbf{y} = B\mathbf{x}$ as follows:
       set $\mathbf{y} = \bar{\sigma}_1^{-1}\mathbf{x}$
       for all $l = 2, \dots, L$, $i = 1, \dots, n_l$ do in parallel
           $\mathbf{y} \leftarrow \mathbf{y} + ((\bar{\sigma}_l^{-1} - \bar{\sigma}_{l-1}^{-1})/D_{ii}^l)\ \langle\mathbf{f}_i^l, \mathbf{x}\rangle_{l_2(\mathrm{supp}(\mathbf{f}_i^l))}\ \mathbf{f}_i^l$
           with the update of $\mathbf{y}$ being a critical section
       end for
```

4.3.1 Implementation Details

The code is written in Fortran and realized via Intel Fortran 90 and GCC Fortran 90. We also write the test versions of the program in software Matlab before we rewrite it in Fortran.

At first we set the main constants of the program and assembly the stiffness matrix and the load vector. Then, there comes a set-up phase, in which we compute prolongators, create smoothed prolongators and coarse matrices. The code itself is separated in Fortran modules. For each level, prologators and coarse matrices are stored in a multigrid container, which is a Fortran derived type. Matrices are stored in CSR or CSC format, depending on what is more convenient for us.

We also have to estimate an upper guess of the eigenvalue of A necessary in the algorithm. To do so, we use the Gershgorin's theorem: 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')`.

To set the stopping criterion, we might consider several types of errors. We consider the exact solution u, vector b, computed solution u_h.
In our code we use the relative residual error defined as

$$\frac{\sqrt{(b - Au_h)^{\mathrm{T}}(b - Au_h)}}{\sqrt{b^{\mathrm{T}}b}}.$$

4.3.2 Conjugated Gradients Method

The conjugated gradients method is a solver for linear system $A\mathbf{u} = \mathbf{f}$, where A is a symmetric positive definite matrix. The method was introduces in 1952 by Hestens and Stiefel [31]. They described the algebraic algorithm and also some problems arising with the finite arithmetic. Later, it was shown that the conjugated gradients method can be derived from the Lanczos algorithm or can be viewed as the optimization of quadratic functional [27].

First, we consider non-preconditioned conjugated gradients method, described by Algorithm 4.3.3. Now, we would like to solve a system $BA\mathbf{u} = B\mathbf{f}$, where B the preconditioner. The preconditioner B is is constructed so that it's action is easy to implement and the algorithm allows for the fast convergence of the conjugated gradients method. The method of preconditioned conjugated gradients is described by algorithm 4.3.4.

Algorithm 4.3.3 CG

```
given: $A, \mathbf{f}, \mathbf{u}^k \in \mathbb{R}^n$
set: $r^k = \mathbf{f} - A\mathbf{u}^k, \mathbf{p}^0 = \mathbf{r}^0$
1: repeat
2:
       $$\alpha_k = \frac{\langle \mathbf{r}^k, \mathbf{r}^k \rangle}{\mathbf{p}^k, A\mathbf{p}^k} \tag{4.9}$$
3:     $\mathbf{u}^{k+1} = \mathbf{u}^k + \alpha_k \mathbf{p}^l$
4:     $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha_k \mathbf{p}^l$
5:
       $$\delta_k = \frac{\langle \mathbf{r}^{k+1}, \mathbf{r}^{k+1} \rangle}{\mathbf{r}^k, \mathbf{r}^k} \tag{4.10}$$
6:     $\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \delta_k \mathbf{p}^k$
7: until convergence
```

Algorithm 4.3.4 PCG

```
given: $B, A, \mathbf{f}, \mathbf{u}^k \in \mathbb{R}^n$
set: $r^k = \mathbf{f} - A\mathbf{u}^k, \mathbf{z}^0 = \mathbf{r}^0, \mathbf{p}^0 = B\mathbf{z}^0$
1: repeat
2:
       $$\alpha_k = \frac{\langle \mathbf{z}^k, \mathbf{r}^k \rangle}{\mathbf{p}^k, A\mathbf{p}^k} \tag{4.11}$$
3:     $\mathbf{u}^{k+1} = \mathbf{u}^k + \alpha_k \mathbf{p}^l$
4:     $\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha_k \mathbf{p}^l$
5:     $\mathbf{z}^{k+1} = B\mathbf{r}^{k1}$
6:
       $$\delta_k = \frac{\langle \mathbf{z}^{k+1}, \mathbf{r}^{k+1} \rangle}{\mathbf{z}^k, \mathbf{r}^k} \tag{4.12}$$
7:     $\mathbf{p}^{k+1} = \mathbf{r}^{k+1} + \delta_k \mathbf{p}^k$
8: until convergence
```

4.4 Numerical Experiments

The model problem has been implemented considering two variants - the model problem with conjugated gradients method (CG) and SA BPX preconditioned CG method (PCG). The SA BPX variant is implemented in the smoothed aggregation settings following Alg. 4.3.1. The set-up of the model problem is defined in Section 3.5 and reminded in Section 4.3.

At first, we present results obtained by an implementation of the serial version of the algorithm. We are interested in convergence of the algorithm compared to non-preconditioned conjugated gradients method and their computational times. In Tables 4.1 − 4.2 there are computational times and number of iterations for the model model problem. The stopping criterion is given by the relative residual error achieving a tolerance ϵ. The number L determines number of levels and, in the case of the model problem, it determines degrees of freedom (DOF). In Table 4.1 we also show elapsed time for the set-up phase of the algorithm.

We can see that the preconditioned conjugated gradients method gives a nearly optimal convergence.

		Set-up phase	Solve phase	
L	DOF	Smooth prolongator	CG	PCG
4	729	0	0	0
5	6561	0.047	0.062	0.046
6	59049	1.357	0.265	0.405
7	531441	2.527	17.09771	4.671
8	4782969	19.750	157.85	56.301

Table 4.1: Computational times elapsed for smoothing the prolongator, CG and PCG solver, $\epsilon = 10^{-5}$

L	DOF	CG	PCG
4	729	39	22
5	6561	119	29
6	59049	362	32
7	531441	1102	35
8	4782969	>1500	37

Table 4.2: Number of iterations for CG and PCG solver, $\epsilon = 10^{-5}$

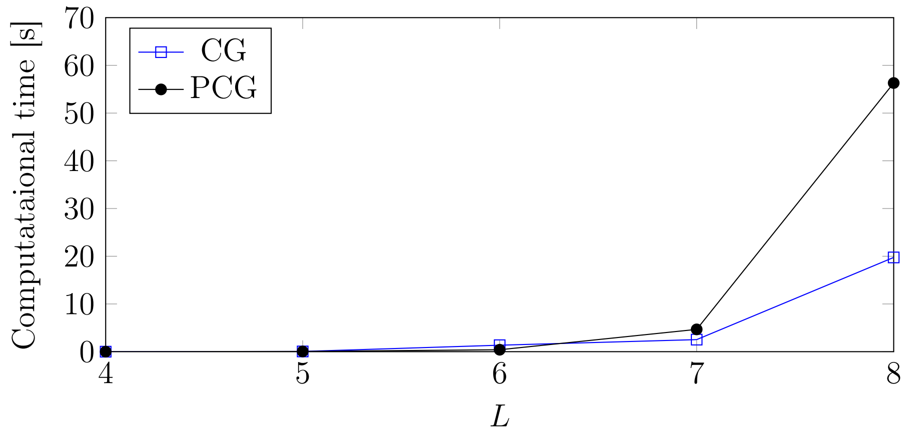


Figure 4.1: Computational times elapsed for smoothing the prolongator, CG and PCG solver, $\epsilon = 10^{-5}$

4.4.1 Parallelization of Algorithm 4.3.1

In the parallel version of the program, we will are interested in possibilities of parallelization of Alg. 4.3.1. In this part we consider three ways.

Case 1

One way of how to parallelize Alg. 4.3.1 is a natural parallelization of

$$\frac{1}{\bar{\sigma}_1}\mathbf{x} + \sum_{l=2}^{L} \left(\frac{1}{\bar{\sigma}_l} - \frac{1}{\bar{\sigma}_{l-1}} \right) I_l^1 D_l^{-1} (I_l^1)^T.$$

The algorithm is an additive method and every addend of the sum can be computed separately. The resulting computational times are given in Table 4.3. Table 4.4 shows a speed-up for $L = 7$ using one to seven threads. Computational times for 1-7 threads are in Fig 4.2.

	Number of levels				
L	4	5	6	7	8
	0.129	0.149	0.499	3.585	54.362

Table 4.3: Computational times for parallel PCG solver, 7 threads $\epsilon = 10^{-5}$

L	*Threads*	Computational times
7	1	4.8821
7	2	4.01
7	3	2.889
7	4	2.699
7	5	2.559
7	6	2.671
7	7	2.685

Table 4.4: Computational times for parallel PCG solver, 1-7 threads, 7 levels, $\epsilon = 10^{-5}$

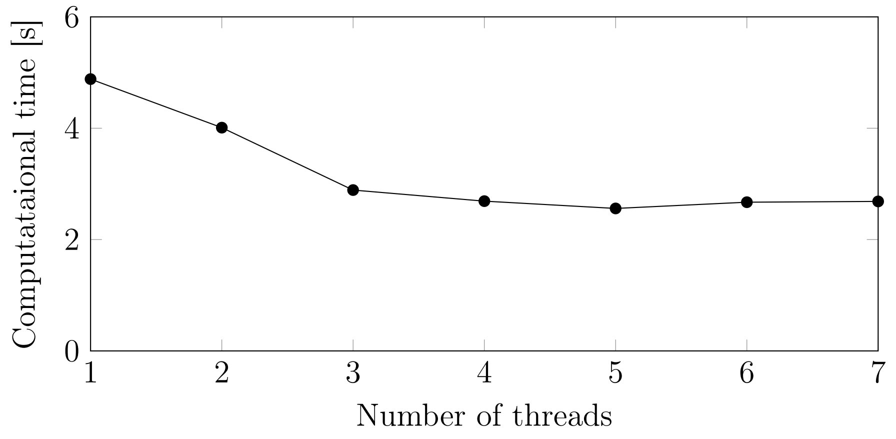


Figure 4.2: Computational times elapsed for parallel PCG solver, 1-7 threads, 7 levels

Case 2

Second way is a parallelization of $\mathbf{y} = \tilde{Q}_l\mathbf{x}$. In this case we implement a parallel matrix-vector multiplication in

$$I_l^1 D_l^{-1} (I_l^1)^T.$$

The resulting computational times are given in Table 4.5. Table 4.6 shows a speed-up for $L = 7$ using one to seven threads. Computational times for 1-7 threads are in Fig 4.3. The results are similar to the previous case.

	Number of levels				
L	4	5	6	7	8
	0.116	0.160	0.494	3.150	52.467

Table 4.5: Computational times for parallel PCG solver, 7 threads $\epsilon = 10^{-5}$

L	*Threads*	Computational times
7	1	4.699
7	2	3.805
7	3	3.642
7	4	2.992
7	5	2.899
7	6	2.545
7	7	2.581

Table 4.6: Computational times for parallel PCG solver, 1-7 threads, 7 levels, $\epsilon = 10^{-5}$

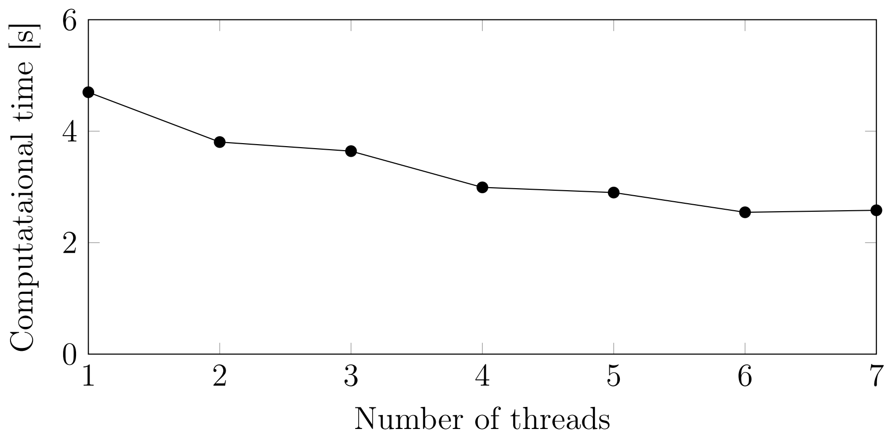


Figure 4.3: Computational times elapsed for parallel PCG solver, 1-12 threads, 7 levels

Case 3

Third way of parallelization is based on splitting of the prolongator into into columns (**f** and make the whole cycle in parallel over $l = 2, \ldots, L$ and $i = 1, \ldots, n_l$. For all $l = 2, \ldots, L$, $i = 1, \ldots, n_l$

in parallel

$$\mathbf{y} \leftarrow \mathbf{y} + ((\bar{\sigma}_l^{-1} - \bar{\sigma}_{l-1}^{-1})/D_{ii}^l) \ \langle \mathbf{f}_i^l, \mathbf{x} \rangle_{l_2(\mathrm{supp}(\mathbf{f}_i^l))} \ \mathbf{f}_i^l$$

with the update of $\mathbf{y}$ being a critical section

The resulting computational times are given in Table 4.7. Table 4.8 shows a speed-up for $L = 7$ using one to twelve threads. Computational times for 1-12 threads are in Fig 4.4. If we look closer at the results in Table 4.8, we see that the solving time using more than 5 threads is about 2,3 faster then if we use one thread, but the computational times are higher than in the previous two cases. One problem could be the fact, that the use of a critical section slows the algorithm significantly. Other problem might be that every we weren't implemented the algorithm such that the set-up phase and critical section is not compensated by speed-up given by parallel implementation..

	Number of levels				
L	4	5	6	7	8
	0.123	0.208	0.836	7.173	203.8667

Table 4.7: Computational times for parallel PCG solver, 12 threads $\epsilon = 10^{-5}$

L	*Threads*	Computiational times
7	1	16.233
7	2	10.891
7	3	8.531
7	4	7.687
7	5	7.191
7	6	6.462
7	7	7.161
7	8	6.789
7	9	6.908
7	10	6.775
7	11	7.071
7	12	7.135

Table 4.8: Computational times for parallel PCG solver, 1-12 threads, 7 levels, $\epsilon = 10^{-5}$

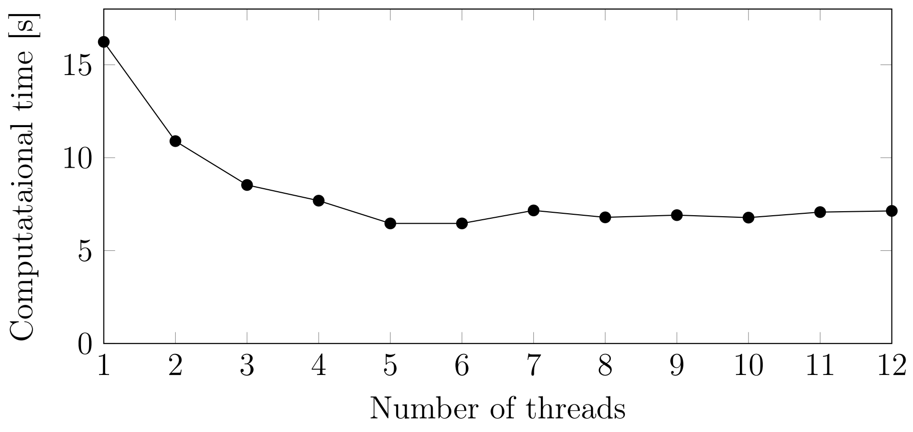


Figure 4.4: Computational times elapsed for parallel PCG solver, 1-12 threads, 7 levels

4.4.2 Other Numerical Experiments

Choice of $\bar{\lambda}$ in the smoother

The smoother of the prolongator contains an upper bound $\bar{\lambda}$. The choice of $\bar{\lambda}$ in the smoother of the prolongator affects the Range of the prolongator and thus a convergence of the algorithm. Results in Table 4.9 shows, that we were able to find a value of $\bar{\lambda}$, for such the convergence of preconditioned conjugated gradients is better than in Table 4.2, where the upper bound is given simply by Gerschgorin Theorem.

L	DOF	CG	PCG
4	729	39	20
5	6561	119	26
6	59049	362	28
7	531441	1102	31
8	4782969	>1500	34

Table 4.9: Number of iterations for CG and PCG solver, $\epsilon = 10^{-5}$

Anisotropic problem

Now, we would like to study a behavior of our algorithm for the anisotropic problem.

We consider the operator

$$\varepsilon \frac{\delta^2}{\delta x_1^2} + \frac{\delta^2}{\delta x_2^2},$$

which describes anisotropic process if $\varepsilon \neq 1$. We discretize it by the finite difference method and obtain matrix stencil

$$\frac{1}{h^2} \begin{bmatrix} & -1 & \\ -\varepsilon & 2+2\varepsilon & -\varepsilon \\ & -1 & \end{bmatrix}.$$

We then solve such a problem with conjugated gradients and preconditioned conjugated gradients method again. Table 4.10 shows number of iterations for CG and Table 4.10 for preconditioned CG for different values of ε and $\epsilon = 10^{-10}$. The results for a preconditioned version of the algorithm are much better then for the unpreconditioned method, but we are loosing almost uniform convergence. In such case, the PCG method is not robust.

	ε			
L	1	0.9	0.1	0.01
3	13	21	25	25
4	56	74	97	140
5	168	217	284	463
6	507	652	850	1409

Table 4.10: Number of iterations for CG solver, $\epsilon = 10^{-10}$

	ε			
L	1	0.9	0.15	0.01
3	15	21	25	26
4	36	40	75	124
5	48	51	123	296
6	52	54	160	432

Table 4.11: Number of iterations for PCG solver, $\epsilon = 10^{-10}$

In comparison, see result in Tables 4.12 and 4.13 with numbers of iterations for smoothed aggregation method (used as a stand-alone solver) and CG preconditioned by smoothed aggregation. In both cases we see that algorithm manages the anisotropy well.

	ε			
L	1	0.5	0.1	0.01
2	7	7	7	7
3	16	16	16	18
4	20	21	27	29
5	28	28	32	38
6	35	36	37	49

Table 4.12: Number of iterations for SA solver, $\epsilon = 10^{-10}$

	ε			
L	1	0.5	0.1	0.01
2	4	4	3	3
3	9	9	10	10
4	12	12	14	15
5	14	14	16	18
6	16	17	18	21

Table 4.13: Number of iterations for SA PCG solver, $\epsilon = 10^{-10}$

Improvements of the Algorithm:
Here, we present results of MDS algorithm [65]. MDS is an improved variant of BPX algorithm involving diagonal scaling. In matrix formulation, the MDS preconditioner is given as

$$B_l = \sum_{l=1}^{L} I_l^1 K_l^{-1} I_1^l,$$

where K_l^{-1} is $diag(A_l)$ and I_1^1 is an identity mapping. If K_l^{-1} is replaced by identity matrices, algorithm becomes BPX algorithm.

In Table 4.14 we see the result for MDS in settings of model problem defined in Section 3.5, so that the coarsening is done by factor 3 in each direction. Table 4.15 shows results for MDS with prolongators given by the smoothed aggregation. In this case, aggregates are created with respect to anisotropies. We can see that results in Table 4.15 are much improved in comparison to Tables 4.14 and 4.11, but they are still outperformed by successive version of the algorithm.

	ε			
L	1	0.5	0.1	0.01
3	15	16	20	30
4	26	27	38	65
5	38	38	57	126
6	46	47	80	185

Table 4.14: Number of iterations for PCG solver, regular coarsening, $\epsilon = 10^{-10}$

	ε			
L	1	0.5	0.1	0.01
3	8	9	11	15
4	20	20	29	50
5	34	34	54	76
6	54	57	81	92

Table 4.15: Number of iterations for PCG solver, SA coarsening, $\epsilon = 10^{-10}$

Comparison with other solvers

Here, we compare the model problem preconditioner with other solvers and preconditioners, namely BPX in the smoothed agregations (SA BPX), unpreconditioned conjugated gradient method (CG), smoothed aggregation as a solver (SA), geometric multigrid as a solver (GM) and BPX. On the one hand, every multigrid algorithm has it's strong and weak side which will not show up while testing on such a simple example, but on the other hand, we wanted to compare the SA BPX algorithm in the context of other well known algorithms. We therefore omit the parallel aspects of additive multigrid.

Throughout time, there were done many numerical experiments comparing geometrical and algebraic multigrid and also successive and parallel multigrid algorithms, see [40] and [28]. In general, we can say that if we have a possibility to use geometric multigrid, the geometric multigrid is a preferred choice over the algebraic multigrid, since in SA, the set up phase could be a costly process. It was also shown (theoretically and numerically) that the parallel algorithms are outperformed by successive algorithms.

Our test example is again problem 1.1 on the unit square. The initial mesh is regular square mesh when each square is divided by an edge connecting bottom left corner with upper right corner. This time, unlike in the model problem, the coarsening is done so that we create a system of nested meshes and coarse basis functions can be geometrically interpreted by a finer level basis functions.

Results are shown in Table 4.16. We see that both additive algorithms are outperformed by multiplicative algorithms which is expected. In this example the SA BPX is BPX preconditioner settings of generalized smoothed aggregation method. The aggregates are now not perfectly regular and convergence is worse. We also have to keep in mind that smoothed aggregation method has a costly set-up phase, so we would prefer to use smoothed aggregation as a multiplicative solver.

	ε				
L	SA BPX PCG	CG	SA	BPX PCG	GM
3	8	9	9	11	10
4	18	30	10	20	13
5	49	65	15	36	14
6	74	131	16	66	15
7	136	264	19	121	15

Table 4.16: Number of iterations for various solvers, $\epsilon = 10^{-10}$

Precision dependence

Results in Tables 4.17 and 4.18 show a behavior of the algorithm for $L = 5$ depending on which precision we use. We set the Fortran *real* kind precision to single, double or quad to see that while the convergence of unpreconditioned conjugated gradient method depends significantly on used precision, the preconditioned algorithm is more stable.

L	single	double	quad
3	23	13	13
4	90	56	56
5	241	168	168
6	1201	507	506
7	>1500	>1500	>1500

Table 4.17: Number of iterations for CG solver, $\epsilon = 10^{-10}$

We can see that the ratio between number of iterations of single and double precision algorithm is 1.7949 in average for CG, while for the preconditioned algorithm we get a ratio 1.1344.

L	single	double	quad
3	20	15	14
4	43	36	34
5	56	52	51
6	60	58	58
7	63	61	60

Table 4.18: Number of iterations for PCG solver, $\epsilon = 10^{-10}$

In our implementation, we use double precision. We think that it would not be convenient to use quad, since it doesn't bring any improvement to the preconditioned algorithm and makes whole implementation slower and more memory demanding.

Chapter 5

Applications

In this chapter we will focus on solving problems coming from the real applications. Our aim is to study behavior of multigrid solvers for problems coming from modeling of neutron transport.

In the following sections we will briefly describe two main mathematical models of neutron transport. We would like to stress that we describe only the theory which is necessary to our calculations, so that the text is readable for a reader not involved in physics. The reader can find more in [26], [32], [58], [57], and [14]. In our calculations, we will use the mathematical model of neutron transport described by a steady state multi group diffusion process. The model problem and following discretization comes from the department of Physics in Škoda JS (Nuclear Machinery).

5.1 Neutron flux calculations by transport equation

In the following text we describe the main quantities and variables which are used in the mathematical modeling. The mathematical model of neutron transport is derived from the balance of neutron in the volume V.
Let us consider a neutron moving in the direction of a unit vector $\mathbf{\Omega}$. Its velocity vector is given by

$$\mathbf{v} = v\mathbf{\Omega}.$$

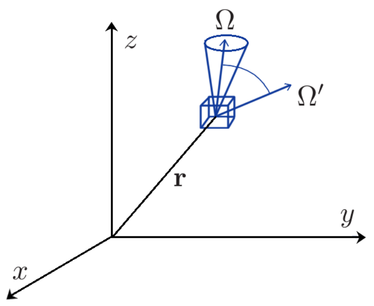


Figure 5.1: Neutron phase plane

The current state of the neutron is described by its

$$\begin{array}{rl} t & \text{time,} \\ \mathbf{r},\ \mathbf{r}=(x,y,z) & \text{spatial position,} \\ \boldsymbol{\Omega},\ \boldsymbol{\Omega}\in\mathbb{R}^3 & \text{streaming direction of neutrons,} \\ E & \text{kinetic energy of neutrons,} \end{array} \tag{5.1}$$

as it is pictured in Fig. 5.1.

One of the most important quantities is the angular neutron density. The angular neutron density is denoted by

$$N(\mathbf{r},\boldsymbol{\Omega},t)$$

and defined as the probable (or expected) number of neutrons at the position $\mathbf{r}$ with direction $\boldsymbol{\Omega}$ and energy E at time t, per unit volume per unit solid angle per unit energy.

The angular neutron flux

$$\phi(\mathbf{r},E,\boldsymbol{\Omega},t)=v(E)N(\mathbf{r},E,\boldsymbol{\Omega},t)$$

is a rate at which the neutrons are passing through a surface of V regardless of its orientation.

The angular neutron current

$$\mathbf{j}(\mathbf{r},E,\boldsymbol{\Omega},t)=v(E)\Omega N(\mathbf{r},E,\boldsymbol{\Omega},t)$$

is a vector quantity which corresponds to a directional flow of neutrons. It's physical interpretation is that

$$\mathbf{j}(\mathbf{r},E,\boldsymbol{\Omega},t)\mathrm{d}A\mathrm{d}E\mathrm{d}\boldsymbol{\Omega}$$

is expected number of neutrons passing through an area dA per unit time with energy E in dE, direction $\boldsymbol{\Omega}$ in d$\boldsymbol{\Omega}$ at time t.

Types of reactions

To assemble the balance of neutrons in the zone, we need to describe which reactions between neutron and nucleus can occur.

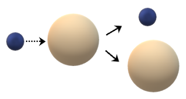
There are many types of reactions, but we will consider main four types that are enough to describe basic processes in the reactor. The reaction in which the neutrons are born is the *fission*. The neutrons born by fission are neutrons of high energy. Such neutrons cannot cause fission reaction due to high energy and the way to lower it is by *scattering*. If neutrons are *absorbed* instead of scattered, they are lost and cannot cause fission any more.

Remark 28 Excited state is denoted as *.

We consider four main interactions:

elastic scattering

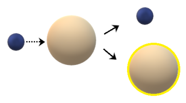
Direction of movement is changed, speed is lowered. Collision of two perfect balls, the energy of neutrons is conserved.



$n + X^A \to (X^{A+1})^* \to n + X^A$

inelastic scattering

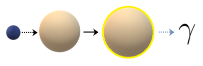
Direction of movement is changed, speed is lowered. The energy is not conserved and escapes.



$n + X^A \to (X^{A+1})^* \to n + (X^A)^*$

absorption

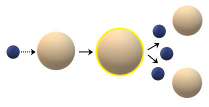
The neutron is absorbed, nucleus attains excited state and then deexcite and emits a particle



$n + X^A \to (X^{A+1})^* \to X^{A+1} + \gamma$

fission

Neutron is absorbed in the nucleus and causes its split and emittion of neutrons.



$n + X^A \to (X^{A+1})^* \to X^{A_2} + X^{A_3} + 3n$

If an energy of the neutron is low enough, it is more probable that it will cause a fission reaction. Neutrons are placed in an environment in which we ensure sufficient scattering. Such a material is called a *moderator*.
In a reactor, we cannot say that a particular reaction will happen for sure, we can

say that it will happen with a certain probability. A probability that a certain reaction happens is described by the cross section. The macroscopic cross section Σ is the probability that a specific interaction occurs when the neutron travels a unit distance through the volume of the homogeneous material. The units are given in cm^{-1}.

Neutron balance

Now, let $V \subset \mathbb{R}^3$ be a balance domain in the reactor core. The total change of amount of neutrons between times t_1 and t_2 in the balance range $\mathrm{d}E\mathrm{d}\mathbf{\Omega}$ in a volume V is given by

$$\int_V \left(N(\mathbf{r}, E, \mathbf{\Omega}, t_2) - N(\mathbf{r}, E, \mathbf{\Omega}, t_1)\right) \mathrm{d}\mathbf{r} \times \mathrm{d}E\mathrm{d}\mathbf{\Omega}.$$

The balance principle in the domain can be described in general as:

$$\begin{aligned} \text{Angular neutron flux density} = &- \text{ neutrons lost via scattering and absorption} \\ &- \text{ neutrons lost via leakage} \\ &+ \text{neutrons gained via scattering} \\ &+ \text{neutrons born in fission} \\ &+ \text{delayed neutron source} \\ &+ \text{other neutrons.} \end{aligned} \tag{5.2}$$

Then the general form of neutron transport equation is given by

$$\begin{aligned} &\left(\frac{1}{v(E)}\frac{\partial}{\partial t} + \mathbf{\Omega}\nabla + \Sigma_s(\mathbf{r}, E, t) + \Sigma_a(\mathbf{r}, E, t)\right)\phi(\mathbf{r}, E, \Omega, t) = \\ &\int_0^{4\pi}\int_0^{\infty} \Sigma_s(\mathbf{r}, E' \to E, \mathbf{\Omega}' \to \mathbf{\Omega}, t)\phi(\mathbf{r}, E', \mathbf{\Omega}', t)\mathrm{d}E'\mathrm{d}\mathbf{\Omega}' + \\ &\int_0^{4\pi}\int_0^{\infty} \chi(\mathbf{r}, E' \to E, \mathbf{\Omega}' \to \mathbf{\Omega}, t)\nu(E')\Sigma_f(\mathbf{r}, E', \mathbf{\Omega}', t)\phi(\mathbf{r}, E', \mathbf{\Omega}', t)\mathrm{d}E'\mathrm{d}\mathbf{\Omega}' + \\ &s_{ex}(\mathbf{r}, E, \mathbf{\Omega}, t). \end{aligned} \tag{5.3}$$

The steady state formulation is used to model neutrons for which the neutron flux doesn't change anymore. The time derivation vanishes and the variables are not dependent on time anymore:

$$\begin{aligned}
&(\boldsymbol{\Omega}\nabla + \Sigma_s(\mathbf{r},E) + \Sigma_a(\mathbf{r},E))\,\phi(\mathbf{r},E,\boldsymbol{\Omega}) = \\
&\quad\int_0^{4\pi}\int_0^{\infty}\Sigma_s(\mathbf{r},E'\to E,\boldsymbol{\Omega}'\to\boldsymbol{\Omega},t)\phi(\mathbf{r},E',\boldsymbol{\Omega}')\mathrm{d}E'\mathrm{d}\boldsymbol{\Omega}'+ \\
&\quad\int_0^{4\pi}\int_0^{\infty}\chi(\mathbf{r},E'\to E,\boldsymbol{\Omega}'\to\boldsymbol{\Omega})\nu(E')\Sigma_f(\mathbf{r},E',\boldsymbol{\Omega}')\phi(\mathbf{r},E',\boldsymbol{\Omega}')\mathrm{d}E'\mathrm{d}\boldsymbol{\Omega}'+ \\
&\quad s_{ex}(\mathbf{r},E,\boldsymbol{\Omega}).
\end{aligned} \tag{5.4}$$

Steps from transport to diffusion

To derive a diffusion equation, we assume:
Isotropic material - to get rid of the angular dependency, we assume that the neutrons are emitted from fission are distributed evenly, as well as we assume scattering is isotropic.
Continuity of flux and current - we assume both flux and current at the boundary of the materials are continuous.

$$\phi(\mathbf{r},E)|_{\partial V-} = \phi(\mathbf{r},E)|_{\partial V+}, \quad \mathbf{j}(\mathbf{r},E)|_{\partial V-} = \mathbf{j}(\mathbf{r},E)|_{\partial V+}.$$

In the diffusion approximation, the transport equation quantities do not depend on the angular variable any more. We define the flux by integrating angular flux over the whole solid angle:

$$\phi(\mathbf{r},E) = \int_0^{4\pi}\phi(\mathbf{r},E,\boldsymbol{\Omega})\mathrm{d}\boldsymbol{\Omega}.$$

and similarly the current:

$$\mathbf{j}(\mathbf{r},E) = \int_0^{4\pi} j(\mathbf{r},E,\boldsymbol{\Omega})\mathrm{d}\boldsymbol{\Omega} = \int_0^{4\pi}\boldsymbol{\Omega}\phi(\mathbf{r},E,\boldsymbol{\Omega})\mathrm{d}\boldsymbol{\Omega}.$$

Using this we get diffusion equation

$$\begin{aligned}
&\nabla\cdot\mathbf{j}(\mathbf{r},E) + (\Sigma_s(\mathbf{r},E) + \Sigma_a(\mathbf{r},E))\phi(r,E) = \\
&\quad\int_0^{\infty}\Sigma_s(\mathbf{r},E'\to E)\phi(\mathbf{r},E',\boldsymbol{\Omega}')\mathrm{d}E'+ \\
&\quad\int_0^{\infty}\chi(\mathbf{r},E'\to E)\nu(E')\Sigma_f(\mathbf{r},E')\phi(\mathbf{r},E')\mathrm{d}E'.
\end{aligned} \tag{5.5}$$

The neutron current will be further approximated such that the neutron flows in the direction of least density by Fick's Law:

$$\mathbf{j}(\mathbf{r},E) = -D(\mathbf{r},E)\nabla\phi(\mathbf{r},E).$$

The coefficient D is the diffusion coefficient.

We then get

$$\begin{aligned} &-\nabla \cdot D(\mathbf{r},E)\nabla\phi(\mathbf{r},E) + (\Sigma_s(\mathbf{r},E) + \sigma_a(\mathbf{r},E))\phi(\mathbf{r},E) = \\ &\int_0^\infty \Sigma_s(\mathbf{r},E' \to E)\phi(\mathbf{r},E',\boldsymbol{\Omega}')\mathrm{d}E' + \\ &\int_0^\infty \chi(\mathbf{r},E' \to E)\nu(E')\Sigma_f(\mathbf{r},E')\phi(\mathbf{r},E')\mathrm{d}E'. \end{aligned} \tag{5.6}$$

5.2 Multigroup Diffusion Approximation

The energy of neutrons in the zone is in a relatively wide range. In practice, we consider several (G) energy groups that correspond to important neutron energy groups (slow, medium, fast, very fast neutrons, etc.):

$$0 = E^g < E^{g-1} < \cdots < E^1 < E^0.$$

In the calculations based on the diffusion model, four or only two groups (fast and slow neutrons) will be considered.
The multigroup approximation of the variables is given by integrating them at intervals of energy groups, for example the group flux ϕ^g is defined as

$$\phi^g(\mathbf{r}) = \int_{E^g}^{E^{g-1}} \phi(\mathbf{r},E)\mathrm{d}E.$$

We then get the multigroup diffusion approximation

$$-D^g\nabla^g\phi^g + (\Sigma_a^g + \sum_{k\neq q}\Sigma_s^{g\to k})\phi^g = \chi^g\sum_k \nu\Sigma_f^k\phi^k + \sum_{k\neq q}\Sigma_s^{k\to g}\phi^k. \tag{5.7}$$

Nuclear chain reaction

It is possible to achieve a state, in which the chain reaction is self- sustained without an external source. We could measure length of life of the neutron from its birth by fission to an end by absorption. This process can be considered as a stochastic process with neutron generations and a certain length of life. The fraction of number of neutrons in two consecutive generations defines a multiplication factor characterizing the chain reaction:

$$k_{eff} = \text{ multiplication factor} = \frac{\text{number of neutrons in one generation}}{\text{number of neutrons in preceding generation}}.$$

There exists another definition of k_{eff} that could be given as

$$k_{eff} = \frac{\text{rate of neutron production in reactor}}{\text{rate of neutron loss (absorbtion + capture) in reactor}}.$$

In practice, we distinguish three states of k_{eff}:

$k_{eff} > 0$ - Number of neutrons grows, chain reactions as well. The reactor is in the supercritical state.

$k_{eff} = 1$ - The reactor is in the critical state, the power of reactor is sustained. Number of neutrons can be maintained in the absence of the source.

$k_{eff} < 0$ - Number of neutrons decreases, the reactor is in the subcritical state.

The problem of criticality can be approached by introducing the variable parameter (eigenvalue) k to the system 5.7 in such way that the neutrons per fission are divided by a constant factor k. The modeled reactor could be critical by varying the number of neutrons emitted in fission.

By it's definition, as the the ratio of neutrons in the next generation to those in the current generation, k is the effective neutron multiplication factor per neutron generation k_{eff}.

For any multiplying system, there exists a unique positive k eigenvalue corresponding to an real and non-negative eigenfunction $\phi_k(\mathbf{r})$, see more theory in [37]. The dominant eigenvalue of the eigenvalue problem satisfies the physical interpretation of the effective multiplicative factor k_{eff}.

The diffusion equation will then have the form

$$-D^g \nabla^g \phi^g + (\Sigma_a^g + \sum_{k \neq q} \Sigma_s^{g \to k}) \phi^g = \frac{\chi^g}{k_{eff}} \sum_k \nu \Sigma_f^k \phi^k + \sum_{k \neq q} \Sigma_s^{k \to g} \phi^k. \tag{5.8}$$

5.2.1 Neutron flux calculations

The problem 5.8 leads to an eigenvalue problem. In general, the are two commonly used ways to compute neutron flux and k_{eff}: by a system of inner and outer iterations and by some fast sparse solver.

Matrix formulation

In the matrix-vector formulation, the problem 5.8 can be written as

$$\mathbf{L}\phi = \frac{1}{k_{eff}} \mathbf{M}\phi, \tag{5.9}$$

where

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \cdot \begin{bmatrix} \phi^1 \\ \phi^2 \end{bmatrix} = \frac{1}{k_{eff}} \begin{bmatrix} M_{11} & M_{22} \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \phi^1 \\ \phi^2 \end{bmatrix},$$

$$\mathbf{L} = \begin{bmatrix} -D^1\nabla^2 + \Sigma_a^1 + \Sigma_s^{1\to 2} & 0 \\ \Sigma_s^{1\to 2} & -D^2\nabla^2 + \Sigma_a^2 \end{bmatrix},$$

$$\mathbf{M} = \begin{bmatrix} \nu\Sigma_f^1 & \nu\Sigma_f^2 \\ & \end{bmatrix}$$

and

$$\chi = [1, 0].$$

By simple manipulations, we get

$$k_{eff}\mathbf{L}\phi = \mathbf{M}\phi,$$

$$\mathbf{L}^{-1}\mathbf{M}\phi = k_{eff}\phi,$$

which leads to solving an eigenvalue problem for the matrix $\mathbf{L}^{-1}\mathbf{M}$. If we decided to solve this eigenvalue problem by the power method, we would get an iterative scheme

$$\phi^{n+1} = \mathbf{L}^{-1}\mathbf{M}\phi^n, \quad n = 1, \ldots \tag{5.10}$$

$$\phi^{n+1} \leftarrow \frac{\phi^{n+1}}{\|\phi^{n+1}\|} \tag{5.11}$$

and

$${k_{eff}}^{n+1} = \frac{\langle \phi^{n+1}, \phi^n \rangle}{\langle \phi^n, \phi^n \rangle} {k_{eff}}^n.$$

In our case it means two step process, in which we first compute the sources $\mathbf{M}\phi$ and then we solve a system of equations for fluxes with sources on the right hand side. This process will be described in detail in the next part.

Neutron Flux by Iterations

The inner and outer iterations lead to searching for the k_{eff} by the power method:

1. Outer iterations - in each step we search for the next update of k_{eff} by power method

2. Inner iterations - in each step we search for the neutron fluxes

For the *two groups*, the inner cycle consists of solving

$$-D^1\nabla^2\phi^1 + (\Sigma_a^1 + \Sigma_s^{1\to 2})\phi^1 = S^1,$$
$$-D^2\nabla^2\phi^2 + \Sigma_a^2\phi^2 = S^2 + \Sigma_s^{1\to 2}\phi^1$$

and for *four groups*, the inner cycle consists of solving

$$\begin{aligned}
-D^1\nabla^2\phi^1 + (\Sigma_a^1 + \Sigma_s^{1\to 2} + \Sigma_s^{1\to 3} + \Sigma_s^{1\to 4})\phi^1 &= S^1,\\
-D^2\nabla^2\phi^2 + (\Sigma_a^2 + \Sigma_s^{2\to 3} + \Sigma_s^{2\to 4})\phi^2 &= S^2 + \Sigma_s^{1\to 2}\phi^1,\\
-D^3\nabla^2\phi^3 + (\Sigma_a^3 + \Sigma_s^{3\to 4})\phi^3 &= S^3 + \Sigma_s^{1\to 3}\phi^1 + \Sigma_s^{2\to 3}\phi^2 + \Sigma_s^{4\to 3}\phi^4,\\
-D^4\nabla^2\phi^4 + \Sigma_a^4\phi^4 &= S^4 + \Sigma_s^{1\to 4}\phi^1 + \Sigma_s^{2\to 4}\phi^2 + \Sigma_s^{3\to 4}\phi^3.
\end{aligned} \tag{5.12}$$

The source S^g is updated by values of ϕ^g in each step as

$$S^g = \frac{\chi^g}{k_{eff}} \sum_k \nu\Sigma_f^k \phi^k.$$

The more compact formula for inner iterations is given as

$$-D^g\nabla^g\phi^g + (\Sigma_a^g + \sum_{k>q}\Sigma_s^{g\to k})\phi^g = \frac{\chi^g}{k_{eff}}\sum_k \nu\Sigma_f^k\phi^k + \sum_{k<q}\Sigma_s^{k\to g}\phi^k.$$

In the case the value of $\Sigma_s^{4\to 3}$ is nonzero, the third equation of 5.12 contains the unknown ϕ^4. We have two options here, we can iterate the value by so called *thermal iteration* or we simply use the value of ϕ^4 from the previous iteration.

After finishing the cycle of inner iterations, we update k_{eff} by some approximation of

$$k_{eff}^{(i)} = \int \nu\Sigma_f^1(\phi^1)^{(i)} + \nu\Sigma_f^2(\phi^2)^{(i)}\mathrm{d}V,$$

or

$$k_{eff}^{(i)} = \frac{\int \psi^{(i)}\mathrm{d}V}{\int \psi^{(i-1)}\mathrm{d}V} k_{eff}^{(i-1)}, \quad \psi = \sum_k \nu\Sigma_f^k\phi^k.$$

5.2.2 Finite Volume Discretization

The neutron flux distribution is, in our case, obtained by finite-volume discretization as in [34] .

The first step is to divide the computing domain into elements. The *hexagon zone* respects the geometry of the reactor core of VVER-440. The mesh consists of hexagons that represents *fuel files/cassettes* whose width is denoted by H_k.

Hexagonal cassettes are further subdivided into hexagonal elements i of volume S_i representing individual *fuel rods* with spacing h and length of the side a, see Fig. (5.2).

Since we will focus on the benchmark problem in 2D, we will derive the formulation in 2D, but the approach would be the same in 3D.

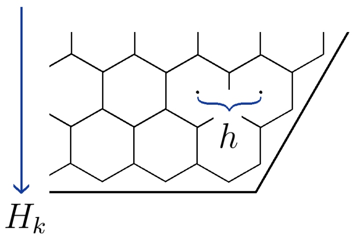


Figure 5.2: Corner of a cassette

On each element, we consider *homogeneous material composition* that is computed from a real heterogeneous distribution.

The second step is to get an approximation by the *finite volume method* [41] for the diffusion equation.

We remind that the diffusion equations is given as

$$\begin{aligned}\nabla \cdot \mathbf{j}(\mathbf{r}, E) + (\Sigma_s(\mathbf{r}, E) + \Sigma_a(\mathbf{r}, E))\phi(r, E) = \\ \int_0^\infty \Sigma_s(\mathbf{r}, E' \to E)\phi(\mathbf{r}, E', \mathbf{\Omega}')\mathrm{d}E' + \\ \int_0^\infty \chi(\mathbf{r}, E' \to E)\nu(E')\Sigma_f(\mathbf{r}, E')\phi(\mathbf{r}, E')\mathrm{d}E'.\end{aligned} \tag{5.13}$$

We then integrate equation 5.13 on each element i and use the divergence theorem:

$$-\sum_j^6 \oint_{S_{i,j}} (\mathbf{j}^g \cdot \overrightarrow{n})\, \mathrm{d}S_{i,j} + \iint_{S_i} \Sigma_r^g \phi^g \mathrm{d}S_i = \iint_{S_i} \left[\chi^g \sum_k \nu \Sigma_f^k \phi^k + \sum_{k \neq q} \Sigma_s^{k \to g} \phi^k \right] \mathrm{d}S_i,$$

where $\overrightarrow{n}$ is an outward-pointing unit normal of $S_{i,j}$ and the compact notation $S_{i,j}$, $j = 1, \ldots, 6$, denotes the boundary lines between element i and it's six adjacent neighbors denoted as j. The situation is described in the Fig 5.3.

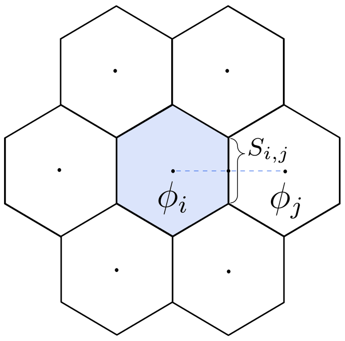


Figure 5.3: The element i and six adjacent elements

The question now remains, how to approximate the circle integral along the boundary $S_{i,j}$, $j = 1, \dots, 6$,

$$-\sum_{j}^{6} \oint_{S_{i,j}} (\mathbf{j}^g \cdot \overrightarrow{n}) \, \mathrm{d}S_{i,j}.$$

First, we assume that the mean value of ϕ^g on each element i is represented by a central value ϕ_i^g. The element will also be represented by one set of material constants $D_i, \Sigma_{a,i}, \Sigma_{f,i}, \Sigma_{s,i}$ and χ_i at the center of the element i.

Now, let us consider two adjacent elements i and j. In the middle of the line $S_{i,j}$, we insert a point s and approximate the circle integral over $S_{i,j}$ by a value $I_s \cdot a$:

$$\oint_{S_{i,j}} (\mathbf{j}^g \cdot \overrightarrow{n}) \mathrm{d}S_{i,j} \approx I_s \cdot a.$$

The value I_s will be further approximated by the two adjacent values of flux ϕ_i and ϕ_j.

The neutron current will be approximated such that the neutron flows in the direction of least density by Fick's Law:

$$\mathbf{j}(\mathbf{r}, E) = -D(\mathbf{r}, E) \nabla \phi(\mathbf{r}, E) \tag{5.14}$$

and the gradient of flux will be approximated by finite differences. Together, we get

$$I_s^+ = D_j^g \frac{\phi_j^g - \phi_{s^+}^g}{h/2}, \tag{5.15}$$

$$I_s^- = D_i^g \frac{\phi_{s^-}^g - \phi_i^g}{h/2}, \tag{5.16}$$

where superscripts + and − have a meaning of the right-hand or left-hand limit.

Now, we have to make some assumptions about continuity of fluxes and currents. We expect

$$\mathbf{j}_s^- = \mathbf{j}_s^+ \tag{5.17}$$

and

$$\phi_s^- = \phi_s^+, \tag{5.18}$$

thus, neutron fluxes and currents are continuous in s, see 5.4.

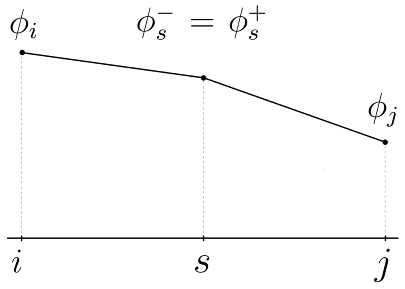


Figure 5.4: Neutron fluxes and currents on the boundary of the two elements

Assumption 5.17 together with 5.15 and 5.16 gives

$$D_j^g \frac{\phi_j^g - \phi_{s+}^g}{h/2} = D_i^g \frac{\phi_{s-}^g - \phi_i^g}{h/2}.$$

We use the assumption 5.18 and we obtain

$$\phi_{s-}^g = \frac{D_i^g \phi_i^g + D_j^g \phi_j^g}{D_i^g + D_j^g}. \tag{5.19}$$

We insert 5.19 to 5.16 and the approximation of the leakage term is

$$I_s = \frac{h}{2} D_i^g D_j^g \frac{\phi_i^g - \phi_j^g}{D_i^g + D_j^g} a.$$

More generally, we could assume that the diffusion coefficient in s are approximated by value $\bar{D}_{ij}^g$, where $\bar{D}_{ij}^g$ is the effective diffusion coefficient on the interface of two cells i and j. Effective diffusion coefficient is then

$$\bar{D}_{ij}^g = h \frac{D_i^g D_j^g}{D_i^g \frac{h}{2} + D_j^g \frac{h}{2}} \tag{5.20}$$

(referred to as harmonic) or geometric:

$$\bar{D}_{ij}^g = \sqrt{D_i^g D_j^g}. \tag{5.21}$$

The leakage term is

$$-\oint_{S_{i,j}} ([D^g \nabla \phi^g]\overrightarrow{n}) \mathrm{d}S_{i,j} = \sum_j \bar{D}_{ij} \frac{\phi_i^g - \phi_j^g}{h} a.$$

The last thing is to approximate the absorption and source term

$$\iint_{S_i} \Sigma_r^g \phi^g \mathrm{d}S_i = \sum_i \phi_i^g S_i$$

Finally, in two dimensions we get

$$\sum_j a\bar{D}_{ij}^g \frac{\phi_i^g - \phi_j^g}{h} + S_i(\Sigma_{a,i}^g + \Sigma_{r,i}^g)\phi_i^g = S_i Z_i. \tag{5.22}$$

The source Z_i includes scattering into a given group, neutrons generated by fission or an external source.

Boundary condition

The system of equations must be equipped with the boundary condition. Our aim is to simulate behavior of the neutron flux at the edge of he zone ∂V. The boundary condition in general is a mixed boundary condition which is implemented in the form of γ-matrix. It holds that $\mathbf{j} = \gamma\phi$, where $\mathbf{j}$ contains neutron currents on the boundary and ϕ is a vector of fluxes on the boundary.

An example of boundary conditions:

- vacuum - $\Phi|_{\partial V} = 0$ - ideal absorption material in which neutrons are not reflected back to the core
- mirror reflector - neutrons reflects back in the sense of mirror reflection
- albedo - $\mathbf{j}^+ = \alpha\mathbf{j}^-$ - albedo is a generalization of the mirror reflection - partition α of neutrons is reflected back to he core ($\mathbf{j}^+$ a $\mathbf{j}^-$ are inner and outer current across the boundary).

In our calculations, we take into account the radial reflector. It is done such that cassettes of the reactor core are surrounded by one row of fictional cassettes, where the geometry of a reflector is modeled. The boundary condition is then applied to the fictional reflector cassettes.

The Feedback Iterations

Solving the system of linear equations from the discretization of the diffusion equation gives us the distribution of the neutron flux Φ and k_{eff}. The input to the system of equations is the set of physical constants for each element of the discretization. The input constants are generally dependent on output quantities, so we have to recalculate the constants. The output quantities include critical boric acid concentration, critical power, critical temperature and more.

A convergence of the feedback iterations is dependent on the convergence of the outer iterations. In general we can say that the convergence of the outer iterations should be in a certain harmony with the feedback iterations so that the overall system converges fast.

5.3 Numerical experiments

In this section we present results of numerical experiments for the benchmark problem Full-Core-440 proposed in [35]. Full-Core-440 is a 2D calculation benchmark based on the VVER-440 core including explicit radial reflector, see Fig. 5.5.

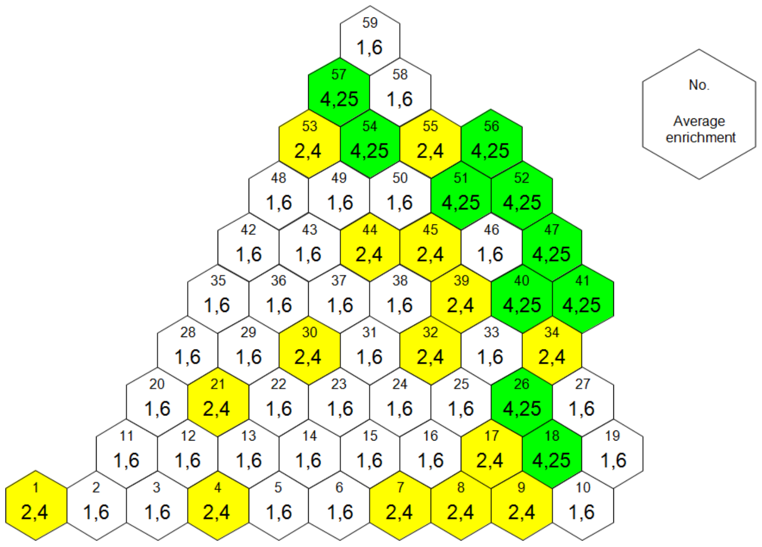


Figure 5.5: Proposal of the benchmark Full-Core-440 (scheme taken from [35])

In the previous section, the pin by pin discretization of the diffusion problem by the finite volume method (FVM) resulted to a generalized eigenvalue problem. Maximal eigenvalue and corresponding eigenvector to this problem are related to a multiplicative coefficient k_{eff} and neutron flux ϕ. An implementation of this problem is done in several steps. We initialize the geometry of the problem, assign all the material constants to particular pins, then the matrices are assembled and finally the eigenvalue problem is solved. A distribution of the neutron flux is used for determination of the power distribution.

It is usually difficult to make significant changes in a large production code. We implemented the FVM code in a way that it is flexible and the numerical experiments can be done easily and then desired changes can be implemented in a production code. Our solution will be compared with a solution of the program Moby-Dick (ŠKODA JS. a.s.) and of MCNM code based on the Monte-Carlo method.

We consider two ways of solving the eigenvalue problem. The first leads to a outer-inner scheme 5.10, where maximal eigenvalue is searched by outer iterations (by the power method) and in each iteration of the power method, the systems of equations are solved for group neutron fluxes. Second way is to solve an enlarged system 5.9 of equations by a method suitable for solving generalized eigenvalue problem. Here, we use Jacobi-Davidson method, see [43].

Remark 29 In our numerical experiments we concentrate on solving the algebraic systems of equations within the inner iterations. Outer iterations are usually accelerated (mostly by Chebyshev technique) and before the actual solver phase, the initial guess of k_{eff} is computed.

Remark 30 The resulting graphs of solutions were generated by the visualization software Map-View.

5.3.1 Computation of k_{eff} by the Power Method

In this section, we present results of computations by the power method. In practice, we are not interested in the flux distribution, but in the power distribution. The power distribution is for each pin i defined as

$$K_r^i = \sum_{g=1}^{G} \phi_i^g \Sigma_{f,i}^g. \tag{5.23}$$

The power distribution is then normalized to the whole core average.

Solution computed by the FVM is compared in Fig 5.6 with MCNM and in Fig. 5.7 and 5.8 with Moby-Dick. The FVM results are in a good agreement with the Moby-Dick code.

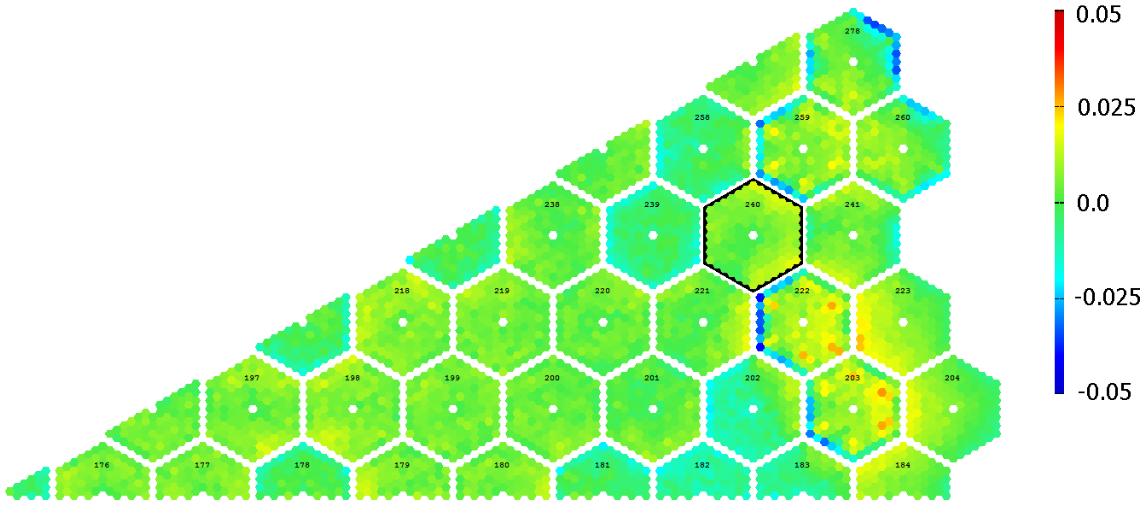


Figure 5.6: Difference between computed solution and MCNP solution

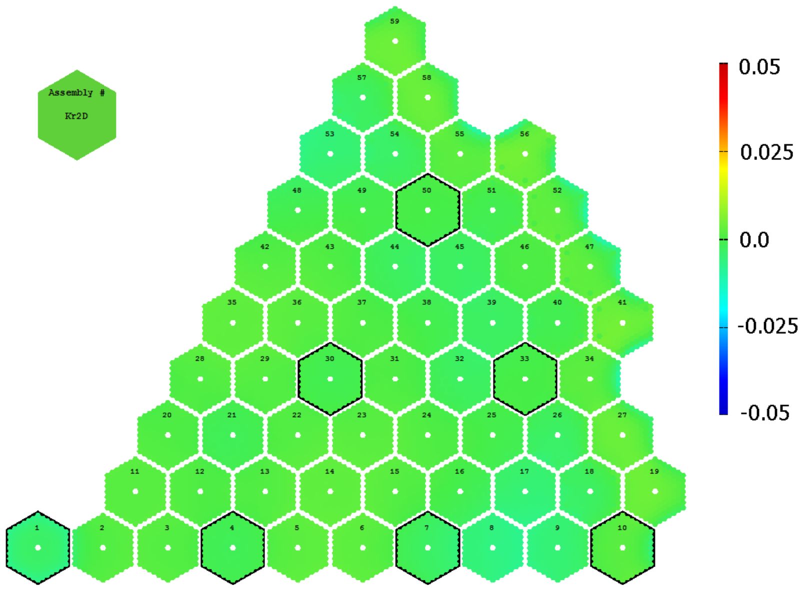


Figure 5.7: Difference between computed solution coefficient and Moby-Dick solution with two rows of reflector cassettes

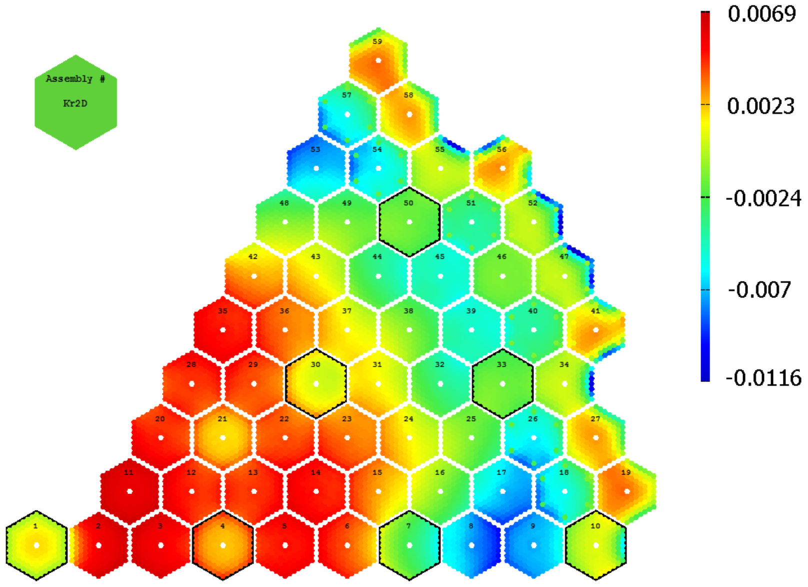


Figure 5.8: Difference between computed solution and Moby-Dick solution with two rows of reflector cassettes (here, color scheme is re-scaled by maximal and minimal error)

The FVM was derived for the harmonic effective diffusion coefficient $\bar{D}$ on the boundary, but we can use also the geometric coefficient. The difference between solution computed with harmonic and geometric $\bar{D}$ on the boundary is depicted in Fig. 5.8. The difference is small and the result is expected, since harmonic and geometric average of two coefficient give almost the same results if the two coefficients do not differ much. This holds for the most of the elements in the core, but the diffusion coefficients differ in the neighborhood of boundary of the zone, boundary of the cassettes or gadolinium rods (their material constants significantly differ from neighboring elements).

The inner iterations imply solving g systems of symmetric positive definite systems, where g in number of energy groups. In the case of the multigrid preconditioning, the fine level variables are given by the pin-by-pin discretization. The coasrening for first two levels is shown in Fig. 5.10. The coarsening pattern is applied to the whole computational domain so that we obtain prolongators for geometric multigrid or regular aggregates.

The convergence result are shown in Table 5.1, the convergence tolerance is set to 10^{-8}. The results show that the SA method is could be used as a effective solver for our problem and even more effective preconditioner for conjugated gradient method.

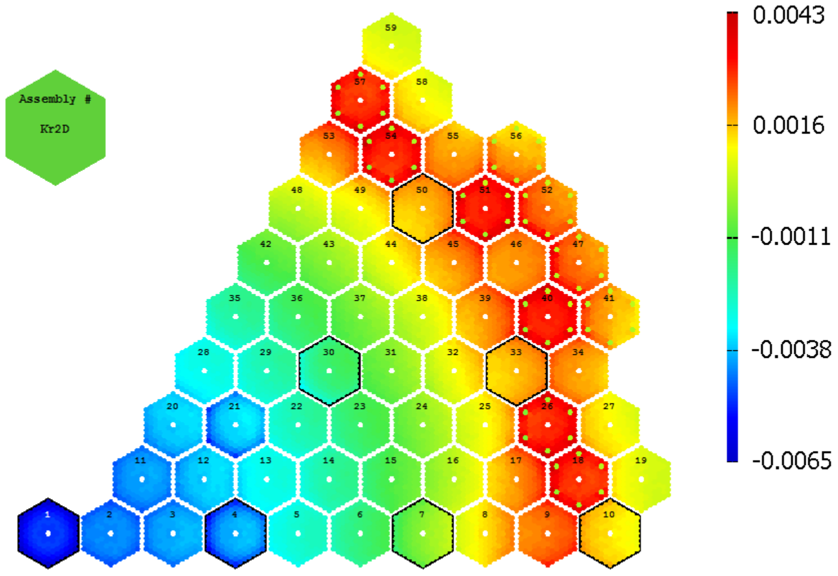


Figure 5.9: Comparison of results computed with harmonical and geometrical effective diffusion coefficient

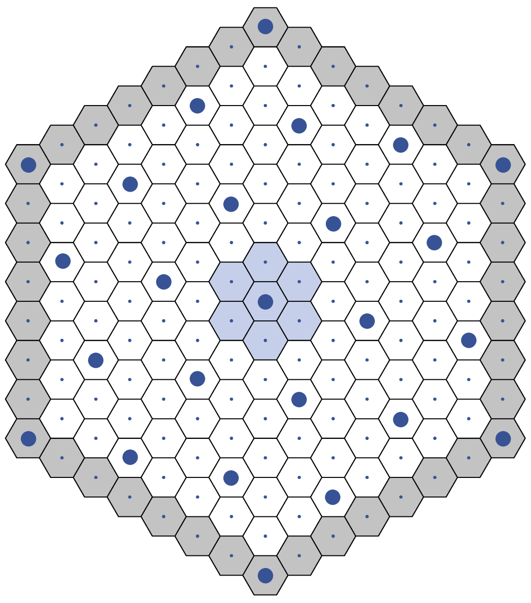


Figure 5.10: Coarsening for the pin-by-pin discretization. The small dots correspond to the fine variables and the bold dots to the coarse variables or centers of the aggregates. The blue area corresponds to one aggregate.

Solver	$It, g=1$	$It, g=2$
SA PCG	12	9
CG	175	66
SA	20	16

Table 5.1: Number of iterations for inner loop, g is the energetic group

5.3.2 Computation of k_{eff} by Jacobi-Davidson Method

The Jacobi-Davidson method ([43]) is a subspace iteration method for problems

$$A\phi = \lambda\phi \tag{5.24}$$

and can also be extended to solve generalized eigenvalue problems.

In the Jacobi-Davidson method the given eigenvalue problem is projected on the search space. The original eigenvalue problem is approximated by a solution to the projected eigenvalue problem. In each iteration step the search subspace is expanded by a correction vector $\mathbf{v}$ which is computed by approximately solving the correction equation. The method is based on solving

$$(I - \phi\phi^T)(A - \lambda I)(I - \phi\phi^T)\mathbf{v} = -\mathbf{r}, \tag{5.25}$$

where ϕ is an eigenvector, λ approximated eigenvalue and $\mathbf{r}$ is the residual vector of the eigenvalue problem

$$\mathbf{r} = (A - \lambda I)\phi. \tag{5.26}$$

For the solution of 5.25, the Krylov subspace methods such as GMRES or BiCG are frequently used. Preconditioning techniques are then applicable for these methods.

The Jacobi-Davidson QZ method (JDQZ) [23] is a Jacobi-Davidson method for the generalized eigenvalue problem

$$A\phi = \lambda B\phi. \tag{5.27}$$

The JDQZ algorithm is used for computing a few eigenvalues with the associated eigenvectors of a matrix pencil $A - \lambda B$. JDQZ can be interpreted as a subspace iteration variant of the QZ algorithm. We use the implementation [1] written by G. Sleijpen.

The results are given in Tables 5.2 - 5.5, where It is the number of JDQZ iterations and MV is the number of matrix multiplications. Except unpreconditioned method we used AMG and diagonal (Jacobi) preconditioning). We see that Jacobi-Davidson method is a efficient solver for our discretization and the AMG preconditioned showed as a efficient preconditioner as well.

In Tables 5.4 and 5.5 we enhanced the JDQZ by initial guess. Here, the initial guess is given by one iteration of power method. The results are much improved for all variants. This is a advantage of this method, since in practice, we usually do not compute jus one state of the reactor core, but a series of consecutive states. Hence, we can use the last distribution of neutron flux as a initial guess for JDQZ.

The convergence tolerance is set to 10^{-8}.

Preconditioner	*It*	*MV*
AMG	27	125
Jacobi	64	385
-	79	472

Table 5.2: Convergence of JDQZ with GMRES

Preconditioner	*It*	*MV*
AMG	4	530
Jacobi	4	810
-	39	7845

Table 5.3: Convergence of JDQZ with BICGstab

Preconditioner	*It*	*MV*
AMG	17	98
Jacobi	62	377
-	67	407

Table 5.4: Convergence of JDQZ with GMRES with initial guess

Preconditioner	*It*	*MV*
AMG	3	231
Jacobi	3	610
-	4	1011

Table 5.5: Convergence of JDQZ with BICGstab with initial guess

5.3.3 Conclusions of Chapter 5

In this section we were focused on the finite volume discretization of the neutron diffusion equation.

The pin-by-pin discretization leads to solving generalized problem which we solved by Jacobi-Davidson and by outer-inner computational scheme. The Inner-outer scheme uses power method in outer loop and inner iterative solver. Inner loop consist of solving symmetric and positive definite problems which is suitable for using conjugated-gradients method.

We implemented the module based on the finite volume method as a tool for making the development of the large macrocode easier and as a tool for numerical experiments.

From a view of multigrid preconditioners, the numerical experiments showed that multigrid preconditioner is effective and easy to apply. The known geometry allows for convenient creation of prolongators in the set-up phase.

The results will serve for future development of the module. We would be careful with pick one eigensolver as the best based on the number of iterations needed for reaching certain tolerance. In practical calcultions, the eigenvalue problem is embedded by the feedback iterations. The convergence of the feedback iterations are in general dependent on the so the choice of the suitable eigensolver must be chised so that the whole system converges fast.

We would also like to mention that here, the assumptions 5.17 and 5.18 lead to a discretization scheme without using the so called discontinuity factors, [44]. In the case that we assume that 5.18 holds using the discontinuity factors on the edge between cells, we can incorporate the discontinuity factors into the discretization scheme.

Conclusions

In this thesis we study parallel multilevel preconditioners, their implementation and application.

The text consists of three main parts. The first part (Chapters 1 and 2) covers a theory of the multigrid methods. Our intention was that the introduction would serve as a review of geometric and algebraic multigrid and would be a comprehensive introduction to a reader, who has not studied multigrid methods before. In the end of this theoretical part, we present a new multigrid approach on the model problem. Second part of the thesis (Chapter 3) contains implementation details and numerical experiments and the third part (Chapter 4) contains application of multigrid solvers and preconditioners on the neutron diffusion problem.

The first chapter starts with a description of basic one-level iterative methods and reasons of their slow convergence. Insufficient convergence rates serve as a motivation for constructing a robust method that would operate on more then just one level. Simple iterative methods are followed by mechanisms of geometric multigrid, derivations of the two-grid method and main convergence theorems. We also remind that multigrid algorithms can be viewed in terms of a multiplicative or an additive methods - while a standard $V-cycle$ is the multiplicative algorithm, the BPX algorithm is the additive method. Chapter 1 is finished with a paragraph summarizing of the previous text.

The second chapter continues on with the review of multigrid methods. In this chapter, we are interested in the algebraic multigrid. AMG uses similar principles as the geometric multigrid (prolongators, smoothing, coarse space correction), but the construction of AMG elements is done in an algebraic way. The classical AMG approach is considered to be the one of Ruge-Stüben, but since we are interested only in the smoothed aggregation method (SA), we talk about the classical approach briefly. After an introduction, we describe the difference between two-level smoothed aggregation and geometric two-level method, main steps of construction of the smoothed prolongator and main convergence proof.

The review of SA method is followed by a section containing our new theory published in [25]. The theory is connecting SA method with BPX preconditioner.

The BPX preconditioner has been always studied in the context of standard variational multigrid. We propose a proof of nearly uniform convergence of the BPX preconditioner with smoothed aggregation, under the assumption that the mesh is regular. We use similar theoretical means as in [53], but here, the equivalence of discrete and continuous L_2-norms for the hierarchy of coarse-spaces $\text{span}\{\varphi_i^l\}_i$, had to be proved.

Chapter 3 focuses on the implementation and numerical experiments. We created a Fortran 90 code parallelized with OpenMP suitable for solving elliptic problems by various multilevel solvers. We managed to implement the preconditioner in parallel, numerically prove a validity of the previous theory, i.e. to show a nearly uniform convergence and to compare new algorithm with existing popular solvers.

In the introduction of Chapter 3, we briefly describe essentials needed for a Fortran implementation of the preconditioner, sparse matrix storage formats and an introduction to OpenMP parallel programming. The smoothed aggregation method is usually implemented in a set-up phase and a solver phase. The set-up phase of the smoothed aggregation consists of several steps including an aggregation of unknowns, construction of the tentative prolongator and smoothing the prolongator. For each step, there exist techniques and theories that offer an efficient parallel implementation, but in our thesis we focus only on the parallelization of the solver phase.

The implementation of an action of the SA BPX preconditioner follows Algorithm 4.3.1 and avoids direct computation of the Gram matrix. We considered three cases of parallelization of Alg. 4.3.1. In the first case, we parallelized the algorithm naturally, as the action can be applied separately on each level. In the second case, we parallelized just the matrix vector multiplications which create an essential part of the action. The third case considered parallelization via all the levels and the columns of the prolongators. In the first two cases, we were able to speed up the solver time twice. In the third case, we also get a similar speed up, but the overall solver time is higher. It seems that in this case OpenMP interface needs more communication time and have to deal more with the critical section.

The next set of numerical experiments involved convergence of the SA BPX preconditioner for the conjugated gradient method (CG). We compare results for the SA BPX preconditioner with the CG without preconditioning on the model problem and show that numerical experiments are in an agreement with the theory. We also studied the convergence for an anisotropic problem. For this problem, the preconditioner doesn't converge nearly uniformly any more. The results were compared with classical smoothed aggregation method used as a stand alone solver and as a preconditioner. The smoothed aggregation method achieve uniform convergence results even for anisotropic problem, since aggregates naturally follows a pattern of semicoarsening. Motivated by that, as an improvement we used generalized aggregation instead of the regular coarsening and we also used MDS algorithm which is a variant of BPX algorithm. The results much improved, but they were still outperformed by the multiplicative smoothed aggregation algorithm.

We then compared the SA BPX algorithm with other solvers and preconditioners including the geometric BPX algorithm. Convergence results of both additive algorithms were outperformed by the multiplicative variants of multigrid. In general, we could say that since smoothed aggregation method has a costly set-up phase, we prefer smoothed aggregation as a multiplicative algorithm over the BPX variant, even if we consider that the algorithm is naturally parallel. Although the convergence results of SA BPX do not compete with the SA method, we think that it was useful to study BPX algorithm in the context of smoothed aggregation method, since, to our knowledge, it hasn't been studied before and the numerical results are new and interesting.

The last set of numerical experiment involved behavior of the preconditioner in the context of used precision kind. The results showed that preconditioned CG was a bit less dependent on used kind. All the numerical experiments were therefore implemented in double precision.

Chapter 4 contains application of the multigrid preconditioners on the neutron diffusion problem. The chapter is introduced by a brief description of the neutron diffusion problem and the finite volume discretization of the problem. Our main achievement of Chapter 4 is development of a module based on the finite volume method (FVM). The FVM code uses similar numerical scheme as it is in the program Moby-Dick belonging to ŠKODA JS. a.s. Moby-Dick is a large production code and it is time consuming to change the code significantly in order to do various numerical experiments. The purpose of the FVM code was to create a flexible code such it could be conveniently used for research purposes and numerical experiments. Promising results of the numerical experiments can be used for future improvements of Moby-Dick.

The numerical experiments in this thesis were tested on the benchmark problem Full-Core-440 and the result can be compared with solution of Moby-Dick and of MCNM based on the Monte-Carlo method. Numerical experiments in this section involved different ways of searching the critical number. We compare the solutions and make recommendations for future development.

List of Reports and Publications

Technical and Scientific Reports for Škoda JS a.s.

[1] P. Fraňková. Alternativní rekonstrukce výkonu v AZ EDU. JS-VÝP/E040/18. *ŠKODA JS a.s., Plzeň*, 2018.

[2] P. Fraňková. Alternativní rekonstrukce výkonu v AZ ETE, JS-VÝP/E041/18. *ŠKODA JS a.s., Plzeň*, 2018.

[3] P. Fraňková. Implementace neregulární mříže AZ, Zpráva Ae 18789/Dok Rev. 0. *ŠKODA JS a.s., Plzeň*, 2018.

[4] P. Fraňková. Paralelizace nápočtu konstant, JS-VÝP/E043/18. *ŠKODA JS a.s., Plzeň*, 2018.

[5] P. Fraňková. Porovnání výpočtů nodální metodou, Zpráva Ae 18218/Dok Rev. 0. *ŠKODA JS a.s., Plzeň*, 2018.

[6] P. Fraňková. Vývoj modulu MKP, Zpráva Ae 18758/Dok Rev. 0. *ŠKODA JS a.s., Plzeň*, 2018.

[7] P. Fraňková and V. Krýsl. Implementace okrajové podmínky závislé na hranách PS v nodálním modelu AZ, Zpráva Ae 18225/Dok Rev. 0. *ŠKODA JS a.s., Plzeň*, 2018.

[8] P. Fraňková and V. Krýsl. Vývoj varianty kódy reflektující iregularity AZ. JS-VÝP/E035/16/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2016.

[9] P. Fraňková and V. Krýsl. Vývoj nodální ho modulu AZ v programu Moby-Dick. Zpráva Ae16914/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2016.

[10] P. Fraňková and V. Krýsl. Aplikace metody konečných prvků v programu Moby-Dick. Zpráva Ae17354/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2017.

[11] P. Fraňková and V. Krýsl. Implementace okrajové podmínky mezi AZ a radiálním reflektorem závislé na hraně PS . Zpráva Ae 17351/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2017.

[12] P. Fraňková and V. Krýsl. Vývoj modulu programu Moby-Dick reflektující neregularity AZ. Zpráva Ae 17353/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2017.

[13] P. Fraňková and V. Krýsl. Vývoj nodálního modulu Hanka v programu Moby-Dick. Zpráva Ae 17352/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2017.

[14] P. Fraňková, V. Krýsl, and J. Šůstek. Vývoj propojení modulu MKP s programem moby-dick, JS-VÝP/E042/18. *ŠKODA JS a.s., Plzeň*, 2018.

[15] P. Fraňková and D. Sprinzl. Výpočetní podklady pro testy fyzikálního spouštění realizované palivové vsázky ETE U1C15 . Zpráva Ae 16773/Dok, Rev. 1. *ŠKODA JS a.s., Plzeň*, 2016.

[16] P. Fraňková and D. Sprinzl. Výpočetní podklady pro testy fyzikálního spouštění realizované palivové vsázky ETE U2C15 . Zpráva Ae 17134/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2017.

[17] P. Fraňková and D. Sprinzl. Výpočetní podklady pro testy fyzikálního spouštění realizované palivové vsázky ETE U2C16. *ŠKODA JS a.s., Plzeň*, 2018.

[18] V. Krýsl, J. Šůstek, and P. Fraňková. Modernizace makroódu Moby-Dick. Zpráva Ae 17073/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2017.

[19] M. Lovecký, P. Fraňková, and V. Krýsl. Radiální okrajové podmínky modelu VVER-440 podle metody Monte-Carlo. Zpráva Ae 17261/Dok, Rev. 0. *ŠKODA JS a.s., Plzeň*, 2017.

Publications

[20] J. Brousek, P. Fraňková, H. Kopincová, R. Kužel, R. Tezaur, P. Vaněk, and Z. Vastl. An overview of multilevel methods with aggressive coarsening and massive polynomial smoothing. *Electronic transactions on numerical analysis ETNA*, 2015.

[21] J. Brousek, P. Fraňková, and P. Vaněk. Improved convergence bounds for smoothed aggregation method. *Applications of Mathematics*, 2016.

[22] P. Fraňková, H. Kopincová, R. Kužel, P. Vaněk, and Z. Vastl. A short philosophical note on the origin of smoothed aggregations. *Applications of Mathematics*, 2013.

[23] P. Fraňková, J. Mandel, and P. Vaněk. Model analysis of BPX preconditioner based on smoothed aggregation. *Applications of Mathematics*, 2015.

[24] P. Fraňková, M. Hanuš, H. Kopincová, R. Kužel, R. Tezaur, P. Vaněk, and Z. Vastl. More radical smoothed aggregation and its convergence theory. *Electronic transactions on numerical analysis ETNA, submitted*, 2014.

Bibliography

[1] JDQR and JDQZ codes, available via http://www.math.uu.nl/people/sleijpen.

[2] N. S. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 1966.

[3] R. E. Bank and C. C. Douglas. Sharp estimates for multigrid rates of convergence with general smoothing and acceleration. *Society for Industrial and Applied Mathematics*, 1985.

[4] F. A. Bornemann and R. Krause. Classical and cascadic multigrid. *Proceedings of the 9th International Conference on Domain Decomposition Methods 1996*, 1996.

[5] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *Society for Industrial and Applied Mathematics*, 1983.

[6] J. M. Bramble, J. E. Pasciak, J. Wang, and J. Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Mathematics of Computation*, 1991.

[7] J. M. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 1990.

[8] A. Brandt. Multi-level adaptive technique (MLAT) for fast numerical solution to boundary-value problems. *Springer-Verlag*, 1973.

[9] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 1977.

[10] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 1986.

[11] A. Brandt, S. F. McCormick, and J. W. Ruge. Algebraic multigrid (AMG) for automatic multigrid solutions with application to geodetic computations. *Inst. for Computational Studies*, 1982.

[12] M. Brezina, A. J. Cleary, R. D. Falgout, V. Henson, J. E. Jones, T. A.Manteuffel, S. F. McCormick, and J. W. Ruge. Algebraic multigrid based on element interpolation (AMGe). *Society for Industrial and Applied Mathematics*, 2001.

[13] W. Briggs, V. E. Hemson, and S. T. McCormick. A multigrid tutorial. *Society for Industrial and Applied Mathematics*, 2000.

[14] Y. A. Chao and Y. A. Shatilla. Conformal mapping and hexagonal nodal meth- ods ii: Implementation in the ANC-H Code. *Nuclear Science and Engineering*, 1995.

[15] L. Chen. Recursive proofs for multigrid methods. 2018.

[16] P. G. Ciarlet. *The finite element method for elliptic problems.* SIAM, 2002.

[17] P. Drábek. *Úvod do funkcionální analýzy.* Západočeská univerzita, Plzeň, 1993.

[18] R. D. Falgout. An introduction to algebraic multigrid. *Computing in Science and Engineering*, 2006.

[19] R. D. Falgout and P. S. Vassilevski. On generalizing the AMG framework. *Society for Industrial and Applied Mathematics*, 2004.

[20] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov. On two-grid convergence estimates. *Numerical linear algebra with applications*, 2005.

[21] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1962.

[22] R. P. Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 1964.

[23] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. *Society for Industrial and Applied Mathematics*, 1998.

[24] P. Fraňková, H. Kopincová, R. Kužel, P. Vaněk, and Z. Vastl. A short philosophical note on the origin of smoothed aggregations. *Applications of Mathematics*, 2013.

[25] P. Fraňková, J. Mandel, and P. Vaněk. Model analysis of BPX preconditioner based on smoothed aggregation. *Applications of Mathematics*, 2015.

[26] W. J. Garlan. *Reactor physics: Point kinetics.* McMaster University, Hamilton, Ontario, Canada, 2005.

[27] G. H. Golub and C. F. V. Loan. *Matrix computations.* Johns Hopkins University Press, Baltimore, 1996.

[28] T. Grauschopf, M. Griebel, and H. Regel. Additive multilevelpreconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second oorder elliptic PDEs. *Technische Univerität München*, 1996.

[29] W. Hackbusch. Multi-grid methods and applications. *Springer-Verlag*, 1985.

[30] V. E. Henson and U. M. Yang. BoomerAMG: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 2002.

[31] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 1952.

[32] B. Heřmanský. *Jaderné reaktory.* SNTL, Praha, 1981.

[33] J. E. Jones and P. Vassilevski. AMGe based on element agglomeration. *Society for Industrial and Applied Mathematics*, 2001.

[34] V. Krýsl, P. Mikoláš, D. Sprinzl, J. Šůstek, J. Švarný, and K. Vlachovský. *Popis programu MOBY-DICK s úpravami pro VVER-1000.* Škoda JS a.s, Plzeň, 2015.

[35] V. Krýsl, P. Mikoláš, D. Sprinzl, and J. Švarný. Full-Core VVER-440 pin power distribution calculation benchmark. *AER Working Groups A&B, Plzeň*, 2011.

[36] S. F. McCormick. Multigrid methods for variational problems: general theory for the V-cycle. *Society for Industrial and Applied Mathematics*, 1985.

[37] J. Mika. Existence and uniqueness of the solution to the critical problem in the neutron transport theory. *Studia Mathematica*, 1971.

[38] S. Míka and P. Vaněk. Acceleration of convergence of a two level algebraic algorithm by aggregation in smoothing process. *Applications of Mathematics*, 1992.

[39] S. Míka and P. Vaněk. A modification of the two-level algorithm with over-correction. *Applications of Mathematics*, 1992.

[40] G. W. P. Bastian, W. Hackbusch. Additive and multiplicative multi-grid - a comparison. *Computing*, 1998.

[41] T. G. R. Eymard and R. Herbin. *Finite Volume Methods.* in Handbook of Numerical Analysis Vol. 7, 2000.

[42] J. W. Ruge and K. Stüben. Algebraic multigrid. *Frontiers in Applied Mathematics and Statistics*, 1987.

[43] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst. Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems. *BIT Numerical Mathematics*, 1996.

[44] K. S. SMITH. Assembly homogenization techniques for light water reactor analysis. *Progress in Nuclear Energy*, 1986.

[45] K. Stüben. Algebraic multigrid (AMG): An introduction with applications. *Gesellschaft für Mathematik und Datenveranbeitung*, 1999.

[46] E. Sulli. Finite element methods for partial differential equations. *Lecture Notes, Oxford*, 2012.

[47] P. Sváček and M. Feistauer. *Metoda konečných prvků.* Vydavatelství ČVUT, Praha, 2006.

[48] R. Tuminaro. Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines. *IEEE*, 2000.

[49] P. Vaněk. Acceleration of convergence of a two-level algorithm by smoothing transfer operator. *Applications of Mathematics*, 1992.

[50] P. Vaněk. Fast multigrid solver. *Applications of Mathematics*, 1995.

[51] P. Vaněk. Smoothed prolongation multigrid with rapid coarsening and massive smoothing. *Applications of Mathematics*, 2012.

[52] P. Vaněk and M. Brezina. Nearly optimal convergence result for multigrid with aggressive coarsening and polynomial smoothing. *Applications of Mathematics*, 2013.

[53] P. Vaněk, M. Brezina, and J. Mandel. Convergence of algebraic multigrid based on smoothed aggregations. *Numerische Mathematik*, 2001.

[54] P. Vaněk, M. Brezina, and R. Tezaur. Two-grid method for linear elasticity on unstructured meshes. *Society for Industrial and Applied Mathematics*, 1999.

[55] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid on unstructured meshes. Technical report, UCD/CCM Report 34, Center for Computational Mathematics, University of Colorado at Denver, 1994.

[56] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 1996.

[57] M. R. Wagner. *Three-Dimensional Nodal Diffusion and Transport Theory Methods for Hexagonal-z Geometry.* Nuclear Science and Engineering, 1989.

[58] M. M. R. Williams. *Mathematical Methods in Particle Transport Theory.* Butterworths, London, 1971.

[59] X. Xiaowen and M. Zeyao. A new grid-coarsening algorithm for parallel algebraic multigrid method. *Chinese Journal of Numerical Mathematics and Applications*, 2006.

[60] J. Xu. Theory of multilevel methods, dissertation. *Cornell University*, 1989.

[61] J. Xu. Iterative methods by space decomposition and subspace correction. *Society for Industrial and Applied Mathematics*, 1992.

[62] J. Xu. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 1996.

[63] J. Xu. An introduction to multilevel methods, lecture notes. *Pennsylvania State University*, 1997.

[64] J. Xu, H. Zhang, and L. Zikatanov. Obtaining optimal coarse spaces for AMG via trace minimization. *Submitted to Journal of Computational Mathematics*, 2016.

[65] X. Zhang. Multilevel schwarz methods. *Numerische Matematik*, 1992.

[66] L. T. Zikatanov. Two-sided bounds on the convergence rate of two-level methods. *Numerical Linear Algebra with Applications*, 2008.