Multilevel Iterative Methods

Chen-Song Zhang

Version 0.8, July 31, 2020

Abstract

Over the past few decades, intensive research has been done on developing efficient iterative solvers for large-scale linear systems arising from PDEs. One particularly powerful technique that has drawn a lot of attention in practice and theoretical analysis is the class of multilevel iterative solvers/preconditioners. In this lecture note, we will focus on algorithms and analysis of multilevel iterative methods, including the well-known geometric and algebraic multigrid methods, for discrete problems arsing from partial differential equations. The main content of this note is presented for the simple Poisson's equation, but a few more complicated applications of multilevel iterative methods are also discussed.

The lecture note is originally prepared for a semester-long course at the Academy of Mathematics and Systems Science, Beijing. It is mainly based on Prof. Jinchao Xu's short courses at the Peking University in 2013 and at the Academy of Mathematics and Systems Science in 2016, as well as Prof. Ludmil Zikatanov's summer school lectures at the Academy of Mathematics and Systems Science in 2015. Special thanks to Dr. Xuefeng Xu, Ms. Huilan Zeng, and Ms. Wenjuan Liu for proof-reading this note.

- Version 0.1: March 18, 2016 — May 10, 2016
- Version 0.2: May 12, 2016 — May 26, 2016
- Version 0.3: June 08, 2016 — Aug 22, 2016
- Version 0.4: Aug 26, 2016 — Dec 31, 2016
- Version 0.5: Feb 01, 2017 — Jan 10, 2018
- Version 0.6: Sep 10, 2018 — Dec 20, 2018
- Version 0.7: May 28, 2019 — July 24, 2019
- Version 0.8: Jan 24, 2020 — July 31, 2020

Contents

Part I

General Theory of Multilevel Iterative Methods

Chapter 1

Introduction

Computer simulation has become an important tool in engineering and sciences. Many physical problems in scientific and engineering computing can be reduced to the numerical solution of certain partial differential equations (PDEs). Finding a viable solution to underlying discretized systems is often expensive, generally consuming a significant portion of the overall cost in a numerical solution procedure of PDEs. Various fast solution techniques, such as adaptive mesh refinement (AMR), domain decomposition (DD) methods, and multigrid (MG) methods, have been developed to address this issue. In certain sense, all these techniques involve "multilevel" iterations.

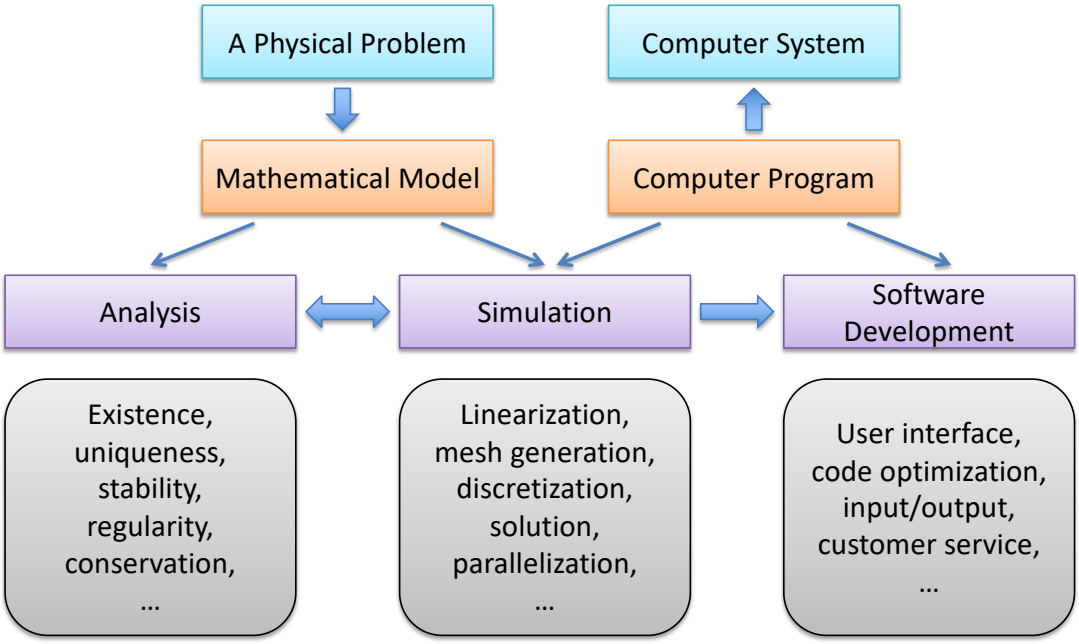


Figure 1.1: Numerical simulation of a physical problem.

The above diagram gives a simple illustration of how a physical problem is "solved" via numerical simulation in general. It is basically an interplay of modeling, mathematical analysis, numerical analysis, scientific computing, and software engineering. A successful computer simulation of complicated physical phenomena requires expertise in many scientific subjects. Hence, nowadays it is difficult for one person to manage all these areas and close collaborations of experts from different areas become crucial.

Effective linear solvers play a key role in many application areas in scientific computing. There are many different types of algorithms for solving linear systems. In this lecture, we focus on studying algorithmic and theoretical aspects of multilevel iterative methods, including geometric multigrid (GMG) and algebraic multigrid (AMG) methods. The basic problem setting for our discussion is: Given an invertible matrix $A : \mathbb{R}^{N\times N}$ and a vector $\vec{f} \in \mathbb{R}^N$, find $\vec{u} \in \mathbb{R}^N$ such that $A\vec{u} = \vec{f}$. There are many features of linear solver that we desire in practice, including:

- Convergence — The method should converge to the solution for any initial guess.
- Robustness — The method should behave similarly in different scenarios.
- Optimality — The method can give a solution with $O(N)$ computational cost.
- Efficiency — The method can give a solution in "reasonably short" wall time.
- Scalability — The method can scale well on modern parallel architectures.
- Reliability — The method should converge to a solution with limited amount of time.
- Usability — The method can be implemented and used relatively easily.

Here we do not mean to define these features rigorously and we will discuss some of them in details later. These features sometimes contradict with each other and we have to find a good balance in practice. There are many different solution methods available, including direct solvers and iterative solvers. In this lecture, we will discuss several popular multilevel iterative methods, including the overlapping domain decomposition methods with coarse space corrections, two-grid methods, geometric multigrid methods, algebraic multigrid methods. And we will mainly study the convergence theory of these methods using the subspace correction framework.

1.1 The model equation

Let $\Omega \subset \mathbb{R}^d$ be an open and bounded domain with Lipschitz boundary and $f \in L^2(\Omega)$. We consider solution of the Poisson's equation with Dirichlet boundary condition

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \tag{1.1}$$

This equation will be our main model equation in most part of this lecture.

Remark 1.1 (Diffusion equation in various applications)**.** The Poisson's equation, or more generally the diffusion equation, appears in many areas of physics, for example, Fick's law for chemical concentration, Fourier's law for temperature, Ohm's law for electrostatic potential, Darcy's law for porous media flow. □

Derivation and classical solution ⋆

The concept of diffusion is widely used in physics, chemistry, biology, sociology, economics, and finance. It is the net movement of the object (e.g. molecules or atoms) from a region of high concentration (or high chemical potential) to a region of low concentration (or low chemical potential). This is also referred to as the movement of a substance down a concentration gradient.

Let $u(x)$ be some diffusive quantity, like pressure, temperature, or concentration of a biological species. We define the operator $\nabla := (\partial_1, \ldots, \partial_d)^T$. So the gradient of scalar function $u : \Omega \mapsto \mathbb{R}$ can be denoted by ∇u. The Laplace operator can be written as $\Delta u = \nabla \cdot \nabla u$. A diffusive flux $\vec{F}$ is usually proportional to the gradient of u, i.e.,

$$\vec{F} = -\kappa \nabla u. \tag{1.2}$$

where κ is the diffusivity (e.g., heat conductivity or permeability). Note that $-\nabla u$ is the so-called steepest descent direction. If a flow is controlled solely by diffusion, then the mass conservation in any volume ω with unit outer normal vectors $\vec{\nu}$ can be written, in the integral form, as

$$\frac{\partial}{\partial t} \int_\omega u \, dx = -\int_{\partial\omega} \vec{F} \cdot \vec{\nu} \, dS$$

or, in the strong form, as

$$\frac{\partial}{\partial t} u = -\nabla \cdot \vec{F}. \tag{1.3}$$

This can be seen by applying the Divergence Theorem

$$\int_\omega \nabla \cdot \vec{F} \, dx = \int_{\partial\omega} \vec{F} \cdot \vec{\nu} dS. \tag{1.4}$$

Now, by plugging (1.2) into (1.3), we obtain an equation

$$\frac{\partial}{\partial t}u = \nabla \cdot (\kappa \nabla u). \tag{1.5}$$

If we assume $\kappa \equiv 1$ or just a constant and there is a source/sink term f on Ω, then we arrive at the heat equation

$$\frac{\partial}{\partial t}u - \Delta u = f. \tag{1.6}$$

The steady-state solution of equation (8.44) satisfies the well-known Poisson's equation

$$-\Delta u = f. \tag{1.7}$$

Remark 1.2 (Laplace equation)**.** In case of the body force or source/sink term is zero, the equation is usually referred to as the Laplace equation

$$-\Delta u = 0. \tag{1.8}$$

If $u \in C^2(\Omega)$ and $-\Delta u = 0$, the u is called a *harmonic function.* □

We have the fundamental solution of the Laplace equation

$$\Phi(x) := \begin{cases} -\frac{1}{2\pi}\log|x|, & d = 2 \\ \frac{1}{d(d-2)\alpha(d)}|x|^{2-d}, & d \geqslant 3 \end{cases} \tag{1.9}$$

where $\alpha(d)$ is the volume of the unit ball in $\mathbb{R}^d$. It is well-known that

$$u(x) = \Phi * f := \int_{\mathbb{R}^d} \Phi(x-y) f(y)\, dy$$

satisfies $-\Delta u = f$ in $\mathbb{R}^d$ and $u \in C^2(\mathbb{R}^d)$; see Evans [53].

Theorem 1.3 (Strong Maximum Principle)**.** If $u \in C^2(\Omega) \bigcap C(\overline{\Omega})$ is harmonic in Ω, then

$$\max_{x \in \overline{\Omega}} u(x) = \max_{x \in \partial\Omega} u(x).$$

If the domain Ω is connected, then $u \equiv C$ if there exists $x_0 \in \Omega$ such that

$$u(x_0) = \max_{x \in \overline{\Omega}} u(x).$$

Using the maximum principle, we can obtain uniqueness of the solution to the Poisson's equation:

Theorem 1.4 (Uniqueness of solution)**.** If $f \in C(\Omega)$, then there exists at most one solution $u \in C^2(\Omega) \bigcap C(\overline{\Omega})$.

Sobolev spaces ⋆

The standard L^∞-norm and L^2-norm will be denoted by $\|\cdot\|_\infty$ and $\|\cdot\|_0$, respectively. The symbol $L^2_0(\Omega)$ denotes a subspace of $L^2(\Omega)$ consisting of functions that have a zero average. The bilinear forms $(\cdot,\cdot)$ and $\langle\cdot,\cdot\rangle$ denote the classical L^2-*inner product* and the *duality pair*, respectively.

Given a natural number $k \in \mathbb{N}$ and $1 \leqslant p \leqslant \infty$, we define the Sobolev spaces

$$W^k_p(\Omega) := \big\{ v : \Omega \mapsto \mathbb{R} \,:\, \nabla^\alpha v \in L^p(\Omega), \text{ for all } |\alpha| \leqslant k \big\}, \tag{1.10}$$

where $\alpha = [\alpha_1, \ldots, \alpha_d]$ is a multi-index and $\nabla^\alpha v := \partial^{\alpha_1}_{x_1} \cdots \partial^{\alpha_d}_{x_d} v$ is the weak derivative of order α. The corresponding norm and semi-norm are then defined as follows: for $1 \leqslant p < \infty$,

$$\|v\|_{W^k_p(\Omega)} := \Big(\sum_{|\alpha|\leqslant k} \|\nabla^\alpha v\|^p_{L^p(\Omega)} \Big)^{\frac{1}{p}}, \quad |v|_{W^k_p(\Omega)} := \Big(\sum_{|\alpha|= k} \|\nabla^\alpha v\|^p_{L^p(\Omega)} \Big)^{\frac{1}{p}}, \tag{1.11}$$

and, for $p = \infty$,

$$\|v\|_{W^k_\infty(\Omega)} := \sup_{|\alpha|\leqslant k} \|\nabla^\alpha v\|_{L^\infty(\Omega)}, \quad |v|_{W^k_\infty(\Omega)} := \sup_{|\alpha|= k} \|\nabla^\alpha v\|_{L^\infty(\Omega)}. \tag{1.12}$$

Definition 1.5 (Sobolev number)**.** Let $\Omega \subset \mathbb{R}^d$ be Lipschitz and bounded, $k \in \mathbb{N}$, and $1 \leqslant p \leqslant \infty$. The Sobolev number is defined by

$$\mathrm{sob}(W^k_p(\Omega)) := k - \frac{d}{p}. \tag{1.13}$$

Remark 1.6 (Natural scaling)**.** There is a natural scaling for the semi-norm $|\cdot|_{W^k_p(\Omega)}$. For $h > 0$, we apply the change of variable $\hat{x} = x/h : \Omega \mapsto \hat{\Omega}$. Then the following scaling result holds

$$\big|\hat{v}\big|_{W^k_p(\hat{\Omega})} = h^{k-\frac{d}{p}} \big|v\big|_{W^k_p(\Omega)} = h^{\mathrm{sob}(W^k_p(\Omega))} \big|v\big|_{W^k_p(\Omega)}.$$

This property is useful in scaling argument (or homogeneity argument) for finite element error estimates. □

If $p = 2$, the spaces $W^k_2(\Omega)$ are Hilbert spaces and we denote them by $H^k(\Omega)$ for short. The inner product is given by

$$(u, v)_{k,\Omega} := (u, v)_{H^k(\Omega)} := \sum_{|\alpha|\leqslant k} \int_\Omega \nabla^\alpha u \, \nabla^\alpha v \, dx.$$

The induced norm of this scalar product is the $W^k_2(\Omega)$-norm. We denote the completion of $C^\infty_0(\Omega)$ in $H^k(\Omega)$ by $H^k_0(\Omega)$. We will also use the fractional Sobolev space $H^{k+\sigma}_0(\Omega)$ where $0 < \sigma < 1$. It is defined as the completion of $C^\infty_0(\Omega)$ in the fraction norm:

$$\|v\|_{H^{k+\sigma}(\Omega)} := \Big(\|v\|^2_{H^k(\Omega)} + |v|^2_{H^{k+\sigma}(\Omega)} \Big)^{\frac{1}{2}},$$

where

$$|v|_{H^{k+\sigma}(\Omega)} := \left(\sum_{|\alpha|=k} \int_\Omega \int_\Omega \frac{|D^\alpha v(x) - D^\alpha v(y)|^2}{|x-y|^{d+2\sigma}} \, dx \, dy \right)^{\frac{1}{2}}.$$

Before we discuss the Poisson's equation in weak formulation, we introduce a few important properties of the Sobolev spaces, which will become important in our later analysis for multigrid methods.

Proposition 1.7 (Sobolev embedding)**.** Let $0 \leqslant k < m$. If $\mathrm{sob}(W_p^m(\Omega)) > \mathrm{sob}(W_q^k(\Omega))$, then the embedding $W_p^m(\Omega) \hookrightarrow W_q^k(\Omega)$ is compact.

Proposition 1.8 (Sobolev embedding to Hölder continuous spaces)**.** Let $0 < m$ and Ω is Lipschitz. If $0 < \mu \leqslant \mathrm{sob}(W_p^m(\Omega))$, then $W_p^m(\Omega) \subset C^{0,\mu}(\Omega) \subset C^0(\Omega)$.

Example 1.9 (Embedding to $C^0(\Omega)$)**.** An example of particular interests is the relation between $H^1(\Omega)$ and continuous functions $C^0(\Omega)$ for $\Omega \subset \mathbb{R}^d$. From Proposition 1.8, we have

$$H^1(\Omega) \subset C^0(\Omega), \text{ if } d = 1; \text{ and } H^1(\Omega) \nsubseteq C^0(\Omega), \text{ if } d > 1.$$

For example, if Ω is the unit disk on $\mathbb{R}^2$, then $u(x, y) = (-\log(x^2 + y^2))^{1/3}$ is not continuous but in $H^1(\Omega)$.

Proposition 1.10 (Poincaré-Wirtinger inequality)**.** For any $v \in H^1(\Omega)$, we have

$$\left\| v - |\Omega|^{-1} \int_\Omega v \, dx \right\|_{0,\Omega} \leqslant C(\Omega) |v|_{1,\Omega}.$$

Proposition 1.11 (Poincaré inequality)**.** For any $v \in H_0^1(\Omega)$, we have

$$\|v\|_{0,\Omega} \leqslant C_d |\Omega|^{1/d} |v|_{1,\Omega}.$$

It is a special case of the more general Friedrichs' inequality on $W_p^k(\Omega)$ with zero trace and it is sometimes referred to as the Friedrichs–Poincaré inequality.

Proposition 1.12 (Trace theorem)**.** There exists a unique linear operator trace : $H^1(\Omega) \mapsto L^2(\partial\Omega)$, such that $\mathrm{trace}(v) = v$, if $v \in C^0(\overline{\Omega}) \bigcap H^1(\Omega)$, and

$$\| \mathrm{trace}(v) \|_{0,\partial\Omega} \leqslant C(\Omega) \|v\|_{1,\Omega}, \quad \forall v \in H^1(\Omega).$$

Moreover, if $g \in H^{\frac{1}{2}}(\partial\Omega)$, there exists $\phi \in H^1(\Omega)$ such that $\phi|_{\partial\Omega} = g$ and

$$\|\phi\|_{1,\Omega} \leqslant C \|g\|_{\frac{1}{2},\partial\Omega}.$$

Weak formulation

Now we consider the Poisson's equation in a weaker sense. A simple motivation is to convert from a point-wise view to an average view:

$$u(x) = 0, \ \text{ a.e.} \quad \Longleftrightarrow \quad \int_\Omega uv\,dx = 0, \quad \forall v \in C_0^\infty(\Omega).$$

Similarly, we can write the Poisson's equation in the weak form (i.e., the integral form). In the one-dimensional case, it is easy to see that

$$-u'' = f, \ \text{ a.e.} \quad \Longleftrightarrow \quad -\int_\Omega (u'' + f)v\,dx = 0, \quad \forall v \in C_0^\infty(\Omega).$$

Let $\mathscr{U}$ be a Hilbert space with an inner product $(\cdot,\cdot)_{\mathscr{U}}$ and its induced norm $\|\cdot\|_{\mathscr{U}}$. Let $\mathscr{V}$ be a Hilbert space with another inner product $(\cdot,\cdot)_{\mathscr{V}}$ and its induced norm $\|\cdot\|_{\mathscr{V}}$. Denote by $\mathscr{V}'$ the dual space of $\mathscr{V}$ equipped with the norm

$$\|f\|_{\mathscr{V}'} := \sup_{v\in\mathscr{V}} \frac{\langle f, v\rangle}{\|v\|_{\mathscr{V}}}, \quad \forall f \in \mathscr{V}'.$$

Definition 1.13 (Continuity)**.** A bilinear form $a[\cdot,\cdot] : \mathscr{U} \times \mathscr{V} \mapsto \mathbb{R}$ is called continuous if and only if there exists a constant C_a such that

$$a[u, v] \leqslant C_a \|u\|_{\mathscr{U}} \|v\|_{\mathscr{V}}, \quad \forall\, u \in \mathscr{U}, v \in \mathscr{V}. \tag{1.14}$$

Consider a continuous bilinear form $a[\cdot,\cdot] : \mathscr{U} \times \mathscr{V} \to \mathbb{R}$ and $f \in \mathscr{V}'$. We formulate a model problem: Find $u \in \mathscr{U}$ such that $\mathcal{A}u = f$ in $\mathscr{V}'$. Or in the weak form, find $u \in \mathscr{U}$ such that

$$a[u, v] = \langle f, v\rangle, \quad \forall v \in \mathscr{V}. \tag{1.15}$$

Example 1.14 (The Poisson equation)**.** The Poisson problem with homogenous Dirichlet boundary was given in (1.1). In this case, we have $\mathcal{A}u := -\Delta u$ and $a[u, v] := (\nabla u, \nabla v)$. Apparently, the bilinear form $a[\cdot,\cdot]$ is continuous due to the Cauchy–Schwarz inequality and $\mathscr{U} = \mathscr{V} = H_0^1(\Omega)$. □

Well-posedness of the weak problem ⋆

We denote the space of all linear and continuous operators from $\mathscr{U}$ to $\mathscr{V}$ as $\mathscr{L}(\mathscr{U};\mathscr{V})$. Here we review a few results on the inf-sup condition due to Nečas [90].

Theorem 1.15 (Banach–Nečas Theorem)**.** Let $a[\cdot,\cdot] : \mathscr{U} \times \mathscr{V} \mapsto \mathbb{R}$ be a continuous bilinear form with a norm defined as

$$\|a[\cdot,\cdot]\| := \sup_{u\in\mathscr{U}} \sup_{v\in\mathscr{V}} \frac{a[u, v]}{\|u\|_{\mathscr{U}} \|v\|_{\mathscr{V}}}.$$

(i) Then there exists a unique linear operator $\mathcal{A} \in \mathscr{L}(\mathscr{U}; \mathscr{V})$ such that

$$(\mathcal{A}u, v)_{\mathscr{V}} = a[u, v], \quad \forall u \in \mathscr{U}, v \in \mathscr{V},$$

with the operator norm

$$\|\mathcal{A}\|_{\mathscr{L}(\mathscr{U};\mathscr{V})} = \|a[\cdot, \cdot]\|.$$

(ii) Moreover, the bilinear form $a[\cdot, \cdot]$ satisfies the inf-sup condition:

$$\exists\, \alpha > 0, \ \text{ such that } \ \alpha \|u\|_{\mathscr{U}} \leqslant \sup_{v \in \mathscr{V}} \frac{a[u, v]}{\|v\|_{\mathscr{V}}}, \quad \forall\, u \in \mathscr{U}, \tag{1.16}$$

$$\text{for any } \ 0 \neq v \in \mathscr{V}, \ \text{ there exists } \ u \in \mathscr{U}, \ \text{ such that } \ a[u, v] \neq 0, \tag{1.17}$$

if and only if $\mathcal{A} : \mathscr{U} \mapsto \mathscr{V}$ is an isomorphism and

$$\|\mathcal{A}^{-1}\|_{\mathscr{L}(\mathscr{V};\mathscr{U})} \leqslant \alpha^{-1}. \tag{1.18}$$

Proof. *(i)* For any fixed $u \in \mathscr{U}$, the mapping $a[u, \cdot]$ belongs to the dual space $\mathscr{V}'$. By the Riesz representation theorem, there exists $\mathcal{A}u \in \mathscr{V}$ such that

$$(\mathcal{A}u, v)_{\mathscr{V}} = a[u, v], \quad \forall\, v \in \mathscr{V}.$$

Since $a[\cdot, \cdot]$ is continuous, we obtain a bounded operator $\mathcal{A} \in \mathscr{L}(\mathscr{U}; \mathscr{V})$. Furthermore,

$$\|\mathcal{A}\|_{\mathscr{L}(\mathscr{U};\mathscr{V})} = \sup_{u \in \mathscr{U}} \frac{\|\mathcal{A}u\|_{\mathscr{V}}}{\|u\|_{\mathscr{U}}} = \sup_{u \in \mathscr{U}} \sup_{v \in \mathscr{V}} \frac{(\mathcal{A}u, v)_{\mathscr{V}}}{\|u\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} = \sup_{u \in \mathscr{U}} \sup_{v \in \mathscr{V}} \frac{a[u, v]}{\|u\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} = \|a[\cdot, \cdot]\|.$$

(ii) $\Longrightarrow$ The inf-sup condition (1.16) guarantees that there exists $\alpha > 0$ such that

$$\alpha \|u\|_{\mathscr{U}} \leqslant \sup_{v \in \mathscr{V}} \frac{a[u, v]}{\|v\|_{\mathscr{V}}} = \sup_{v \in \mathscr{V}} \frac{(\mathcal{A}u, v)_{\mathscr{V}}}{\|v\|_{\mathscr{V}}} = \|\mathcal{A}u\|_{\mathscr{V}}, \quad \forall u \in \mathscr{U}. \tag{1.19}$$

This implies that $\mathcal{A}$ is injective. Let $\{u_k\}_{k=0}^{\infty} \subset \mathscr{U}$ and $v_k := \mathcal{A}u_k$ be a sequence such that $v_k \to v \in \mathscr{V}$. In order to show the range of $\mathcal{A}$ is closed, we need to show $v \in \mathcal{A}(\mathscr{U})$. From the inequality (1.19), we have

$$\alpha \|u_k - u_j\|_{\mathscr{U}} \leqslant \|\mathcal{A}(u_k - u_j)\|_{\mathscr{V}} = \|v_k - v_j\|_{\mathscr{V}} \to 0.$$

Hence, $\{u_k\}_{k=0}^{\infty}$ is a Cauchy sequence and $u_k \to u \in \mathscr{U}$. Moreover,

$$v = \lim_{k \to \infty} v_k = \lim_{k \to \infty} \mathcal{A}u_k = \mathcal{A}u \in \mathcal{A}(\mathscr{U}).$$

Now we assume that $\mathcal{A}(\mathscr{U}) \neq \mathscr{V}$. Since $\mathcal{A}(\mathscr{U})$ is closed, we can decompose $\mathscr{V}$ as

$$\mathscr{V} = \mathcal{A}(\mathscr{U}) \oplus \mathcal{A}(\mathscr{U})^{\perp}$$

and $\mathcal{A}(\mathscr{U})^{\perp}$ is non-trivial. That is to say, there exists $0 \neq v_{\perp} \in \mathcal{A}(\mathscr{U})^{\perp}$, which contradicts the condition (1.17). Hence the assumption $\mathcal{A}(\mathscr{U}) \neq \mathscr{V}$ cannot hold, i.e., $\mathcal{A}$ is surjective. This, in turn, shows that $\mathcal{A}$ is an isomorphism from $\mathscr{U}$ onto $\mathscr{V}$. Moreover, (1.19) shows

$$\alpha \|\mathcal{A}^{-1} v\|_{\mathscr{U}} \leqslant \|v\|_{\mathscr{V}}, \quad \forall\, v \in \mathscr{V}.$$

This proves the inequality (1.18).

(ii) $\Longleftarrow$ We have

$$\begin{aligned} \inf_{u\in\mathscr{U}} \sup_{v\in\mathscr{V}} \frac{a[u,v]}{\|u\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} &= \inf_{u\in\mathscr{U}} \sup_{v\in\mathscr{V}} \frac{(\mathcal{A}u,v)}{\|u\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} = \inf_{u\in\mathscr{U}} \frac{\|\mathcal{A}u\|_{\mathscr{V}}}{\|u\|_{\mathscr{U}}} \\ &= \inf_{v\in\mathscr{V}} \frac{\|v\|_{\mathscr{V}}}{\|\mathcal{A}^{-1}v\|_{\mathscr{U}}} = \Big(\sup_{v\in\mathscr{V}} \frac{\|\mathcal{A}^{-1}v\|_{\mathscr{U}}}{\|v\|_{\mathscr{V}}}\Big)^{-1} = \|\mathcal{A}^{-1}\|^{-1}_{\mathscr{L}(\mathscr{V};\mathscr{U})} \geqslant \alpha. \end{aligned}$$

This is exactly (1.16). Since $\mathcal{A}$ is an isomorphism, for any $0 \neq v \in \mathscr{V}$, there exists $0 \neq u \in \mathscr{U}$, such that $\mathcal{A}u = v$ and

$$a[u,v] = (\mathcal{A}u, v) = \|v\|^2_{\mathscr{V}} \neq 0,$$

which is (1.17). □

Theorem 1.16 (Nečas Theorem)**.** Let $a[\cdot,\cdot] : \mathscr{U} \times \mathscr{V} \mapsto \mathbb{R}$ be a continuous bilinear form. Then the equation (1.15) admits a unique solution $u \in \mathscr{U}$ for all $f \in \mathscr{V}'$, if and only if the bilinear form $a[\cdot,\cdot]$ satisfies one of the equivalent inf-sup conditions:

(1) There exists $\alpha > 0$ such that

$$\sup_{v\in\mathscr{V}} \frac{a[w,v]}{\|v\|_{\mathscr{V}}} \geqslant \alpha \|w\|_{\mathscr{U}}, \quad \forall\, w \in \mathscr{U}; \tag{1.20}$$

and for every $0 \neq v \in \mathscr{V}$, there exists $w \in \mathscr{U}$ such that $a[w,v] \neq 0$.

(2) There holds

$$\inf_{w\in\mathscr{U}} \sup_{v\in\mathscr{V}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} > 0 \qquad \text{and} \qquad \inf_{v\in\mathscr{V}} \sup_{w\in\mathscr{U}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} > 0. \tag{1.21}$$

(3) There exists a positive constant $\alpha > 0$ such that

$$\inf_{w\in\mathscr{U}} \sup_{v\in\mathscr{V}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} = \inf_{v\in\mathscr{V}} \sup_{w\in\mathscr{U}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} = \alpha. \tag{1.22}$$

Furthermore, the solution u satisfies the stability condition

$$\|u\|_{\mathscr{U}} \leqslant \frac{1}{\alpha} \|f\|_{\mathscr{V}'}.$$

Proof. Let $\mathcal{J} : \mathscr{V} \mapsto \mathscr{V}'$ be the isometric Reisz isomorphism. According to Theorem 1.15, we have $\mathcal{A} \in \mathscr{L}(\mathscr{U}; \mathscr{V})$, which is the linear operator corresponding to $a[\cdot, \cdot]$. In this sense, (1.15) is equivalent to

$$u \in \mathscr{U} : \quad \mathcal{A}u = \mathcal{J}^{-1} f \quad \text{in } \mathscr{V}.$$

Assume the condition (1) holds. Then, $\mathcal{A}$ is invertible by Theorem 1.15. The other direction is also easy to see.

Now the interesting part is to show the equivalence of the three conditions, (1), (2), and (3). From the proof of Theorem 1.15, we have seen that

$$\inf_{w \in \mathscr{U}} \sup_{v \in \mathscr{V}} \frac{a[w, v]}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} = \|\mathcal{A}^{-1}\|^{-1}_{\mathscr{L}(\mathscr{V}; \mathscr{U})}.$$

Similarly,

$$\begin{aligned} \inf_{v \in \mathscr{V}} \sup_{w \in \mathscr{U}} \frac{a[w, v]}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} &= \inf_{v \in \mathscr{V}} \sup_{w \in \mathscr{U}} \frac{(\mathcal{A}w, v)_{\mathscr{V}}}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} = \inf_{v \in \mathscr{V}} \sup_{w \in \mathscr{U}} \frac{(w, \mathcal{A}^{\dagger} v)_{\mathscr{U}}}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} \\ &= \|\mathcal{A}^{-\dagger}\|^{-1}_{\mathscr{L}(\mathscr{U}; \mathscr{V})} = \|\mathcal{A}^{-1}\|^{-1}_{\mathscr{L}(\mathscr{V}; \mathscr{U})}, \end{aligned}$$

where $\mathcal{A}^{\dagger}$ denotes the adjoint operator. Furthermore, if the condition

$$\inf_{v \in \mathscr{V}} \sup_{w \in \mathscr{U}} \frac{a[w, v]}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} > 0$$

holds, then for any $v \in \mathscr{V}$, we have

$$\sup_{w \in \mathscr{U}} \frac{a[w, v]}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} > 0.$$

Hence there exists $w \in \mathscr{U}$, such that $a[w, v] \neq 0$. This completes the equivalence proof. □

From the proof of the last two theorems, we have the following observations:

Remark 1.17 (Existence and uniqueness)**.** Solution of the equation (1.15) exists (i.e., $\mathcal{A}$ is *surjective* or *onto*) if and only if

$$\inf_{v \in \mathscr{V}} \sup_{w \in \mathscr{U}} \frac{a[w, v]}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} > 0. \qquad \text{existence or surjective}$$

Solution of (1.15) is unique (i.e., $\mathcal{A}$ is *injective* or *one-to-one*) if and only if

$$\inf_{w \in \mathscr{U}} \sup_{v \in \mathscr{V}} \frac{a[w, v]}{\|w\|_{\mathscr{U}} \|v\|_{\mathscr{V}}} > 0. \qquad \text{uniqueness or injective}$$

That is to say, $\mathcal{A}$ is bijective if and only if the inf-sup conditions (1.21) or its equivalent conditions hold. In finite dimensional spaces, any linear surjective or injective map is also bijective. So we only need one of the above inf-sup conditions to show well-posedness. □

Remark 1.18 (Optimal constant)**.** The constant α in (1.22) is the largest possible constant in (1.20). In general, the first condition in Theorem 1.16 is easier to verify than the third condition. □

Corollary 1.19 (Well-posedness and inf-sup condition)**.** If the weak formulation (1.15) has a unique solution $u \in \mathscr{U}$ for any $f \in \mathscr{V}'$ so that

$$\|u\|_{\mathscr{U}} \leqslant C\|f\|_{\mathscr{V}'},$$

then the bilinear form $a[\cdot,\cdot]$ satisfies the inf-sup condition (1.22) with $\alpha \geqslant C^{-1}$.

Proof. Since (1.15) has a unique solution for all $f \in \mathscr{V}'$, the operator $\mathcal{A} : \mathscr{L}(\mathscr{U};\mathscr{V})$ is invertible and $\mathcal{A}^{-1} : \mathscr{L}(\mathscr{V};\mathscr{U})$ is bounded. Due to the fact $\|u\|_{\mathscr{U}} \leqslant C\|f\|_{\mathscr{V}'}$, we have $\|\mathcal{A}^{-1}\|_{\mathscr{L}(\mathscr{V};\mathscr{U})} \leqslant C$. From the proof of the Nečas theorem, we can immediately see the optimal inf-sup constant $\alpha = \|\mathcal{A}^{-1}\|^{-1}_{\mathscr{L}(\mathscr{V};\mathscr{U})} \geqslant C^{-1}$. □

A simple model problem

Now we consider the simplest case where $\mathscr{V} = \mathscr{U}$ and $\mathcal{A}$ is coercive.

Definition 1.20 (Coercivity)**.** A continuous bilinear form $a[\cdot,\cdot] : \mathscr{V} \times \mathscr{V} \mapsto \mathbb{R}$ is called coercive if there exists $\alpha > 0$ such that

$$a[v,v] \geqslant \alpha\|v\|^2_{\mathscr{V}}, \quad \forall v \in \mathscr{V}. \tag{1.23}$$

We notice that $\sup_{w\in\mathscr{V}} \frac{a[v,w]}{\|w\|_{\mathscr{V}}} \geqslant \frac{a[v,v]}{\|v\|_{\mathscr{V}}} \geqslant \alpha\|v\|_{\mathscr{V}}$, which implies the first inf-sup condition in Theorem 1.16. Hence, for any $f \in \mathscr{V}'$, the coercive variational problem (1.15) has a unique solution and the solution u is continuously depends on f, i.e., $\|u\|_{\mathscr{V}} \leqslant \alpha^{-1}\|f\|_{\mathscr{V}'}$. In this case, Theorem 1.16 is reduced to the well-known *Lax-Milgram theorem*.

Corollary 1.21 (Lax-Milgram theorem)**.** Let $a[\cdot,\cdot] : \mathscr{V} \times \mathscr{V} \mapsto \mathbb{R}$ be a continuous bilinear form which satisfies the coercivity condition (1.23). Then (1.15) has a unique solution $u \in \mathscr{V}$ for any $f \in \mathscr{V}'$ and $\|u\|_{\mathscr{V}} \leqslant \alpha^{-1}\|f\|_{\mathscr{V}'}$.

Remark 1.22 (Energy norm)**.** If the bilinear form $a[\cdot,\cdot] : \mathscr{V} \times \mathscr{V} \mapsto \mathbb{R}$ is symmetric, then, apparently, it defines an inner product on $\mathscr{V}$. Its induced norm is also called the energy norm

$$|\!|\!|v|\!|\!| := a[v,v]^{1/2}.$$

Coercivity and continuity of the bilinear form $a[\cdot,\cdot]$ imply that

$$\alpha\|v\|^2_{\mathscr{V}} \leqslant |\!|\!|v|\!|\!|^2 \leqslant \|a[\cdot,\cdot]\|\,\|v\|^2_{\mathscr{V}} = \|\mathcal{A}\|_{\mathscr{L}(\mathscr{V};\mathscr{V})}\|v\|^2_{\mathscr{V}},$$

namely, the energy norm $|\!|\!|\cdot|\!|\!|$ is equivalent to the $\|\cdot\|_{\mathscr{V}}$-norm. We will denote the dual energy norm by $|\!|\!|\cdot|\!|\!|_*$. □

Remark 1.23 (Poisson is "well-conditioned")**.** We notice that the Poisson's equation is well-posed in the sense that $-\Delta : \mathscr{V} \mapsto \mathscr{V}'$ is an isomorphism with $\mathscr{V} = H_0^1(\Omega)$ and $\mathscr{V}' = H^{-1}(\Omega)$. There exist constants α (coercivity constant) and C_a (continuity constant), such that

$$\alpha \|v\|_{\mathscr{V}}^2 \leqslant a[v,v] = \langle -\Delta v, v\rangle \leqslant C_a \|v\|_{\mathscr{V}}^2, \quad \forall v \in \mathscr{V}.$$

Hence we have the "condition number" of the Laplace operator is bounded

$$\kappa(-\Delta) = \|-\Delta\|_{\mathscr{L}(\mathscr{V};\mathscr{V}')} \cdot \|(-\Delta)^{-1}\|_{\mathscr{L}(\mathscr{V}';\mathscr{V})} \leqslant \frac{C_a}{\alpha}.$$

This means $-\Delta$ is well-conditioned, which is contradicting our experience in solving the Poisson's equation numerically. The problem here lies in that we are working on two different spaces $\mathscr{V}$ and $\mathscr{V}'$. If we consider $-\Delta : L^2(\Omega) \mapsto L^2(\Omega)$ instead, then we lost boundedness. More general theory has been developed in the seminar work by Babuška [4]. □

High-frequency and locality

Consider the eigenvalue problem for one-dimensional Laplace operator with the homogenous Dirichlet boundary condition, i.e., $-u''(x) = \lambda u(x)$ for $x \in (0,1)$ and $u(0) = u(1) = 0$. It is easy to see that the eigenvalues and the corresponding eigenfunctions are

$$\lambda_k = (k\pi)^2 \qquad \text{and} \qquad u_k(x) = \sin(k\pi x), \quad k = 1, 2, \cdots$$

For other types of boundary conditions, the eigenvalues and eigenfunctions can be obtained as well. We notice that larger eigenvalues (larger k) correspond to eigenfunctions of higher frequency. Similar results can be expected for discrete problems which will be discussed later on.

An important observation comes from the analysis to the local problem

$$-u_\delta''(x) = f(x), \;\; x \in B_\delta := (x_0 - \delta, x_0 + \delta) \qquad \text{and} \qquad u_\delta(x_0 - \delta) = u_\delta(x_0 + \delta) = 0.$$

We can obtain the eigenfunctions of this local problem:

$$u_{\delta,k}(x) = \sin\big(\frac{k\pi}{2\delta}(x - x_0 + \delta)\big), \quad k = 1, 2, \cdots.$$

Define the error $e := u - u_\delta$ in B_δ. Hence e is harmonic in B_δ. It is easy to construct a cut-off function $\theta \in C_0^\infty(B_\delta)$, such that it satisfies the following conditions:

$$\text{(i) } \theta(x) > 0; \quad \text{(ii) } \theta(x) = 1, \; \forall x \in B_{\delta/2}; \quad \text{(iii) } |\theta'(x)| \leqslant \tfrac{C}{\delta}.$$

Thus we have

$$\int_{B_{\delta/2}} \left|e'(x)\right|^2 dx \leqslant \int_{B_\delta} \theta^2(x) \left|e'(x)\right|^2 dx = -\int_{B_\delta} \Big((\theta^2)'e' + \theta^2 e''\Big) e\, dx$$
$$\leqslant \frac{2C}{\delta} \int_{B_\delta} \left|\theta e' e\right| dx \leqslant \frac{2C}{\delta} \Big(\int_{B_\delta} \left|\theta e'\right|^2 dx \Big)^{\frac{1}{2}} \Big(\int_{B_\delta} |e|^2 \, dx \Big)^{\frac{1}{2}}.$$

The first and last inequalities immediately imply that

$$\Big(\int_{B_{\delta/2}} \left|e'(x)\right|^2 dx \Big)^{\frac{1}{2}} \leqslant \Big(\int_{B_\delta} \theta^2(x) \left|e'(x)\right|^2 dx \Big)^{\frac{1}{2}} \leqslant \frac{2C}{\delta} \Big(\int_{B_\delta} |e|^2 \, dx \Big)^{\frac{1}{2}}. \tag{1.24}$$

If we plug in the eigenfunctions $u_{\delta,k}$ to the above inequality, we can see that

$$\frac{k\pi}{2\delta} \leqslant \frac{2C}{\delta} \quad \text{or} \quad k \leqslant \frac{4C}{\pi},$$

which suggests only *low-frequency* components are left in the error function e and oscillating components in the distance δ are accurately captured.

Remark 1.24 (High-frequencies)**.** This simple result implies that the high-frequency part of u can be estimated very well by the local solution u_δ for the model problems. Motived by (1.24), we can define geometric high-frequency functions u_k as those with relatively large $\|\nabla u_k\|_{0,\Omega}/\|u_k\|_{0,\Omega}$ ratio. Moreover, we also note that singularities are special forms of high-frequency. Many forms of singularity can be resolved numerically through local mesh refinement. The reason why this type of methods is able to work is such a local behavior of high frequencies. In the later chapters, we will discuss more on this issue from geometric and algebraic perspectives. □

1.2 Discretization methods

Discretization concerns the process of transferring continuous functions, models, or equations into their discrete counterparts. This process is usually carried out as the first step toward making them suitable for numerical evaluation and implementation on modern computers.

Let $\Omega \in \mathbb{R}^d$ be an open domain and $f \in L^2(\Omega)$. We consider the following model problem

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases}$$

Many discretization methods have been developed, such as finite difference (FD) and the finite element (FE) methods, each with specific approaches to discretization. After discretization, we usually end up with a linear algebraic system of equations

$$A\vec{u} = \vec{f}. \tag{1.25}$$

Finite difference method

In one-dimensional case, without loss of generality, we can assume $\Omega = (0,1)$ and the domain is sub-divided into $N+1$ equally spaced pieces. So we get a uniform mesh with meshsize $h = \frac{1}{N+1}$; see the following figure for illustration.

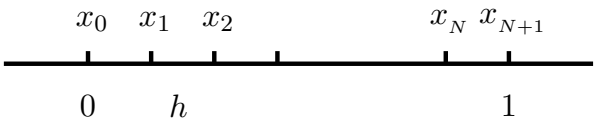


Figure 1.2: Uniform mesh in 1D.

Using the Taylor's expansion, we can easily obtain that

$$\begin{aligned} u''(x_i) &= \frac{1}{h}\Big[u'(x_{i+\frac{1}{2}}) - u'(x_{i-\frac{1}{2}})\Big] + O(h^2) \\ &= \frac{1}{h^2}\Big[u(x_{i-1}) - 2u(x_i) + u(x_{i+1})\Big] + O(h^2). \end{aligned}$$

Let $u_i \approx u(x_i)$ be an approximate solution. Then the FD discretization of the Poisson's equation is

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{pmatrix}. \tag{1.26}$$

That is to say,

$$A := \frac{1}{h^2}\,\text{tridiag}(-1, 2, -1) \qquad \text{and} \qquad \vec{f} := \Big(f_i\Big)_{i=1}^{N} = \Big(f(x_i)\Big)_{i=1}^{N}.$$

We need to solve the linear system $A\vec{u} = \vec{f}$ in order to obtain an approximate solution to the Poisson's equation. It is worth noticing that the coefficient matrix A is symmetric positive definite (SPD), sparse, as well as Toeplitz.

Remark 1.25 (An alternative form of the linear system)**.** Sometimes, it is more convenient (for implementation) to also include the boundary values in $\vec{u}$ and write the linear system as

$$\frac{1}{h^2}\begin{pmatrix} 1 & & & & & \\ -1 & 2 & -1 & & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & & 1 \end{pmatrix}\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_N \\ u_{N+1} \end{pmatrix} = \begin{pmatrix} 0 \\ f_1 \\ \vdots \\ f_N \\ 0 \end{pmatrix}.$$

Apparently this form is equivalent to the discrete problem above. □

Remark 1.26 (Eigenvalues of 1D FD problem)**.** For simplicity we now assume $h \equiv 1$. It is well-known (see HW 1.3) that the eigenvalues of $A := \text{tridiag}(-1, 2, -1)$ are

$$\lambda_k(A) = 2 - 2\cos\left(\frac{k\pi}{N+1}\right) = 4\sin^2\left(\frac{k\pi}{2(N+1)}\right)$$

and the corresponding eigenvectors are

$$\vec{\xi}^k = \left(\xi_i^k\right)_{i=1}^N \in \mathbb{R}^N, \quad \text{with } \xi_i^k := \sin\left(\frac{ik\pi}{N+1}\right).$$

We note that the set of eigenvectors of A, $\vec{\xi}^k = \left(\xi_i^k\right)_{i=1}^N$, forms an orthogonal basis of $\mathbb{R}^N$. Therefore, any $\vec{\xi} \in \mathbb{R}^N$ can be expanded in terms of these eigenvectors:

$$\vec{\xi} = \sum_{k=1}^{N} \alpha_k \vec{\xi}^k.$$

This type of expansion is often called the discrete Fourier expansion. From Figure 1.3, we can

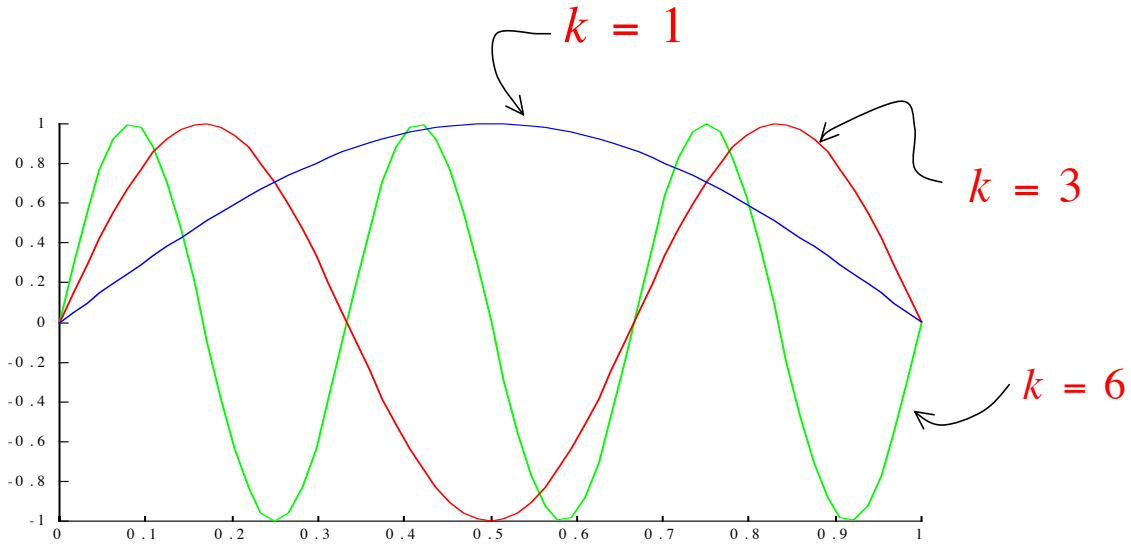


Figure 1.3: Eigenvectors of 1D finite difference system for the Poisson's equation.

easily see that the eigenvectors are "smooth" with small k and are "oscillatory" with large k. Hence the smoothness of $\vec{\xi}$ has a lot to do with the relative size of the coefficients α_k. □

For two-dimensional problems, we can partition the domain uniformly in both x and y-directions into $n+1$ pieces ($N = n^2$). We denote $(x_i, y_j) = \left(\frac{i}{n+1}, \frac{j}{n+1}\right)$ and the Poisson's equation is discretize using the five-point stencil

$$\frac{1}{h^2}\Big[4u_{i,j} - \left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right)\Big] = f(x_i, y_j), \quad i, j = 1, \ldots, n.$$

Then we need to assign an order to the grid points in order to write the unknowns as a vector. There are many ways to order the unknowns for practical purposes. For simplicity, we use the Lexicographic ordering, i.e., $p_{(j-1)n+i} := (x_i, y_j)$. Then we have

$$\frac{1}{h^2}\begin{pmatrix} A_1 & -I & & & \\ -I & A_2 & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & A_{n-1} & -I \\ & & & -I & A_n \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_{N-1} \\ f_N \end{pmatrix},$$

where the block diagonal matrices $A_i := \text{tridiag}(-1, 4, -1)$, $(i = 1, \ldots, n)$ are tridiagonal. Define $C := \text{tridiag}(-1, 0, -1)$. Then it is clear that

$$A = \frac{1}{h^2}\,\text{tridiag}(-I, A_1, -I) = \frac{1}{h^2} I \otimes A_1 + \frac{1}{h^2} C \otimes I.$$

Remark 1.27 (Eigenvalues of the 2D FD problem)**.** Again we assume $h \equiv 1$. Similar to the 1D problem, we can get the eigenvalues

$$\lambda_{i,j}(A) = 4 - 2\cos\frac{i\pi}{n+1} - 2\cos\frac{j\pi}{n+1} = 4\sin^2\frac{i\pi}{2(n+1)} + 4\sin^2\frac{j\pi}{2(n+1)},$$

with eigenvectors

$$\vec{\xi}_{i,j} = \left(\sin\frac{ki\pi}{n+1}\sin\frac{lj\pi}{n+1}\right)_{k,l=1,\ldots,n}.$$

□

Remark 1.28 (Ordering)**.** The shape of the above coefficient matrix A depends on the ordering of degrees of freedom (DOFs). We will see that the ordering also affects the smoothing properties of smoothers and parallelization. Finding the minimal bandwidth ordering is important for some linear solvers, like the LU factorization methods. But it is NP-hard. □

Finite element method

Finite element method (FEM) is a Galerkin method that uses piecewise polynomial spaces for approximate test and trial function spaces. The readers are referred to [46, 71, 17, 38] for more detailed discussion on construction and error analysis of the standard finite element method.

The weak formulation of the model equation can be written as (see Example 1.14): Find $u \in H_0^1(\Omega)$, such that

$$\int_\Omega \nabla u \cdot \nabla v dx = \int_\Omega f v \; dx, \quad \forall v \in H_0^1(\Omega).$$

In 1D, it is easy to explain the main idea of finite element method. Let $\mathcal{P}_k(\tau)$ be the space of all polynomials of degree less than or equal to k on τ. Let

$$V = V_h := \big\{v \in C(\overline{\Omega}) \,:\, v \in \mathcal{P}_1(x_{i-1}, x_i),\ v(0) = v(1) = 0\big\}.$$

Now we can write the discrete variational problem as: Find $u_h \in V_h$, such that

$$a[u_h, v_h] = (f, v_h), \quad \forall v_h \in V_h.$$

Furthermore, we use nodal basis functions $\phi_i \in V_h$, i.e. $\phi_i(x_j) = \delta_{i,j}$. In this way, we can express a given function $u_h \in V_h$ as $u_h(x) = \sum_{j=1}^N u_j \phi_j(x)$. Hence we arrive at the following equation: For any $i = 1, \ldots, N$,

$$\sum_{j=1}^{N} a[\phi_j, \phi_i] u_j = (f, \phi_i) \qquad \text{or} \qquad \sum_j A_{i,j} u_j = f_i.$$

This is a system of algebraic linear equations

$$A\vec{u} = \vec{f}, \tag{1.27}$$

with $\big(A\big)_{i,j} = a_{i,j} := a[\phi_i, \phi_j]$, $\vec{u} := \big(u_i\big)_{i=1}^N$, and $\vec{f} = \big(f_i\big)_{i=1}^N := \Big(\big\langle f, \phi_i \big\rangle\Big)_{i=1}^N$.

If we use the uniform mesh in Figure 1.2, then we have (see HW 1.4) that

$$A := \frac{1}{h}\,\text{tridiag}(-1, 2, -1) \qquad \text{and} \qquad \vec{f} := \big(hf(x_i)\big)_{i=1}^N.$$

Upon solving this finite-dimensional problem, we obtain a discrete approximation u_h. The finite element method has several appealing properties and it will be the main underlying discretization used in this lecture; see §3.1 for more details.

Remark 1.29 (Discrete Poisson's equation is ill-conditioned)**.** Remark 1.23 has shown that the Poisson's equation has a bounded condition number. On the other hand, the discrete problems from FD and FE are both ill-conditioned if meshsize h is small. Later on, we will see that this will cause problems for many iterative methods. The convergence rates of these methods usually depend on the spectrum of the coefficient matrix A. □

Adaptive approximation

We explain the idea of adaptivity with a simple 1D example. Let $u : [0, 1] \mapsto \mathbb{R}$ be a continuous function. Assume that $0 = x_0 < x_1 < \cdots < x_N = 1$ and $h_i := x_i - x_{i-1}$. Let u_N be a piecewise constant function defined on this partition, i.e., $u_N(x) = u(x_{i-1})$ for all $x_{i-1} \leqslant x < x_i$. Then we have

$$|u - u_N| = |u(x) - u(x_{i-1})| = \Big| \int_{x_{i-1}}^{x} u'(t)\, dt \Big| \leqslant \int_{x_{i-1}}^{x_i} |u'(t)|\, dt \leqslant h_i \|u'\|_{L^\infty(x_{i-1}, x_i)}. \tag{1.28}$$

If the partition is quasi-uniform, then we have the approximation estimate

$$\|u - u_N\|_{L^\infty(0,1)} \leqslant \frac{1}{N}\|u'\|_{L^\infty(0,1)}$$

if u is in $W^1_\infty(0,1)$.

The question now is what happens if the function u is less regular (not smooth, singular, rough)? We assume that u is in $W^1_1(0,1)$. In view of the inequalities in (1.28), we notice that we actually need to bound $\|u'\|_{L^1(x_{i-1},x_i)}$. This motivates to give a special (non-uniform) partition such that

$$\int_{x_{i-1}}^{x_i} |u'(t)|\, dt \equiv \frac{1}{N}\|u'\|_{L^1(0,1)}, \quad \text{for } i = 1, 2, \ldots, N.$$

On this partition, we can still obtain a desirable approximation estimate

$$\|u - u_N\|_{L^\infty(0,1)} \leqslant \frac{1}{N}\|u'\|_{L^1(0,1)}.$$

This motivates us that equidistribution of mesh spacing might not be a good choice when the solution is not smooth. Instead, in such cases, we may seek equidistribution of error. Apparently, this type of mesh is u-dependent and obtaining such a mesh is a nonlinear approximation procedure; see more details in Devore [48].

Remark 1.30 (A very useful notation)**.** We use some notations introduced by Xu [118]. The notation $a \lesssim b$ means: there is a generic constant C independent of meshsize h, such that $a \leqslant Cb$. Similarly, we can define "$\gtrsim$" and "$\cong$". This is important because, in our future discussions, we would like to construct solvers/preconditioners that yield convergence rate independent of meshsize h. □

1.3 Simple iterative solvers

There are many different approaches for solving the linear algebraic equations results from the finite difference, finite element, and other discretizations for the Poisson's equation. For example, sparse direct solvers, FFT, and iterative methods. We only discuss iterative solvers in this lecture.

Some examples

Now we give a few well-known examples of simple iterative methods. Consider the linear system $A\vec{u} = \vec{f}$. Assume the coefficient matrix $A \in \mathbb{R}^{N\times N}$ can be partitioned as $A = L + D + U$, where the three matrices $L, D, U \in \mathbb{R}^{N\times N}$ are the lower triangular, diagonal, and upper triangular parts of A, respectively (the rest is set to be zero).

Example 1.31 (Richardson method)**.** The simplest iterative method for solving $A\vec{u} = \vec{f}$ might be the Richardson method

$$\vec{u}^{\,\text{new}} = \vec{u}^{\,\text{old}} + \omega\Big(\vec{f} - A\vec{u}^{\,\text{old}}\Big). \tag{1.29}$$

We can choose an optimal weight ω to improve performance of this method. □

Example 1.32 (Weighted Jacobi method)**.** The weighted or damped Jacobi method can be written as

$$\vec{u}^{\,\text{new}} = \vec{u}^{\,\text{old}} + \omega D^{-1}(\vec{f} - A\vec{u}^{\,\text{old}}). \tag{1.30}$$

This method solves one equation for one variable at a time, simultaneously. Apparently, it is a generalization of the above Richardson method. If $\omega = 1$, then we arrive at the standard Jacobi method. □

Example 1.33 (Gauss–Seidel method)**.** The Gauss–Seidel (G-S) method can be written as

$$\vec{u}^{\,\text{new}} = \vec{u}^{\,\text{old}} + (D + L)^{-1}(\vec{f} - A\vec{u}^{\,\text{old}}).$$

We rewrite this method as

$$(D + L)\vec{u}^{\,\text{new}} = (D + L)\vec{u}^{\,\text{old}} + (\vec{f} - A\vec{u}^{\,\text{old}}) = \vec{f} - U\vec{u}^{\,\text{old}}.$$

Thus we have

$$\vec{u}^{\,\text{new}} = \vec{u}^{\,\text{old}} + D^{-1}\Big(\vec{f} - L\vec{u}^{\,\text{new}} - (D + U)\vec{u}^{\,\text{old}}\Big). \tag{1.31}$$

Compared with the Jacobi method (1.30) ($\omega = 1$), the G-S method uses the most updated solution in each iteration instead of the previous iteration. □

Example 1.34 (Successive over-relaxation method)**.** The successive over-relaxation (SOR) method can be written as

$$(D + \omega L)\vec{u}^{\,\text{new}} = \omega\vec{f} - \Big(\omega U + (\omega - 1)D\Big)\vec{u}^{\,\text{old}}. \tag{1.32}$$

The weight ω is usually in $(1, 2)$. This is in fact the extrapolation of $\vec{u}^{\,\text{old}}$ and $\vec{u}^{\,\text{new}}$ obtained in the G-S method. If $\omega = 1$, then it reduces to the G-S method. □

These preliminary iterative methods have been covered in standard textbooks of numerical analysis. They can be constructed using a classical splitting approach. Here we employ a modified version to give a better view. Let $\alpha \geqslant 0$ be a real parameter and

$$A := A_1 + A_2 = \big(A_1 + \alpha I\big) + \big(A_2 - \alpha I\big).$$

This way we can split the original equation $A\vec{u} = \vec{f}$ as

$$\big(A_1 + \alpha I\big)\vec{u} = \vec{f} - \big(A_2 - \alpha I\big)\vec{u}.$$

This immediately motivates the standard splitting iterative method

$$\vec{u}^{\,\text{new}} = \big(A_1 + \alpha I\big)^{-1}\Big(\vec{f} - \big(A_2 - \alpha I\big)\vec{u}^{\,\text{old}}\Big). \tag{1.33}$$

The method is equivalent to an alternative form, which is the notation we use in this note, as

$$\vec{u}^{\,\text{new}} = \vec{u}^{\,\text{old}} + B\big(\vec{f} - A\vec{u}^{\,\text{old}}\big),$$

with $B := \big(A_1 + \alpha I\big)^{-1}$. Apparently, we can choose the splitting to obtain the above simple iterative methods. For example, by setting $A_1 = 0$, (1.33) yields the Richardson method (1.29); by setting $\alpha = 0$ and $A_1 = \frac{1}{\omega}D$, (1.33) yields the weighted Jacobi method (1.30).

In this setting, the matrix

$$E := -\big(A_1 + \alpha I\big)^{-1}\big(A_2 - \alpha I\big) = I - BA \tag{1.34}$$

is oftentimes called an *iteration matrix* for the iterative method (1.33). It is well-known that the iterative method converges for any initial guess if and only the spectral radius $\rho(E) < 1$.

A simple observation

Many simple iterative methods exhibit different rates of convergence for short and long wavelength error components, suggesting these different scales should be treated differently. We now try to look into this more closely. Let $\lambda_{\max}$ and $\lambda_{\min}$ be the largest eigenvalue and the smallest eigenvalue of A, respectively, and $\vec{\xi}^{\max}$ and $\vec{\xi}^{\min}$ be the corresponding eigenvectors. One interesting observation many people made is: When we use the weighted Jacobi method (1.30) with weight $\omega = 2/3$ to solve the problem $A\vec{u} = \vec{0}$ with the initial guess just equal to $\vec{\xi}^{\max}$, the convergence is very fast. On the other hand, if the weighted Jacobi iteration is used to solve the same equation but with a different initial guess $\vec{\xi}^{\min}$, the convergence becomes slow. See Figure 1.4 for a demonstration.

Note that the reason which causes this difference mainly relies on the fact that the error in the first problem (corresponding to $\vec{\xi}^{\max}$) is oscillatory or of high frequency but the error in the second problem (corresponding to $\vec{\xi}^{\min}$) is smooth or of low frequency. This makes one speculate that the weighted Jacobi method can damp the high frequency part of the error rather quickly, but slowly for the low frequency part; see Remark 1.24.

In Remark 1.26, we have seen that the eigenvalues of the simple finite difference matrix in 1D are

$$\lambda_k(A) = 2 - 2\cos\left(\frac{k\pi}{N+1}\right).$$

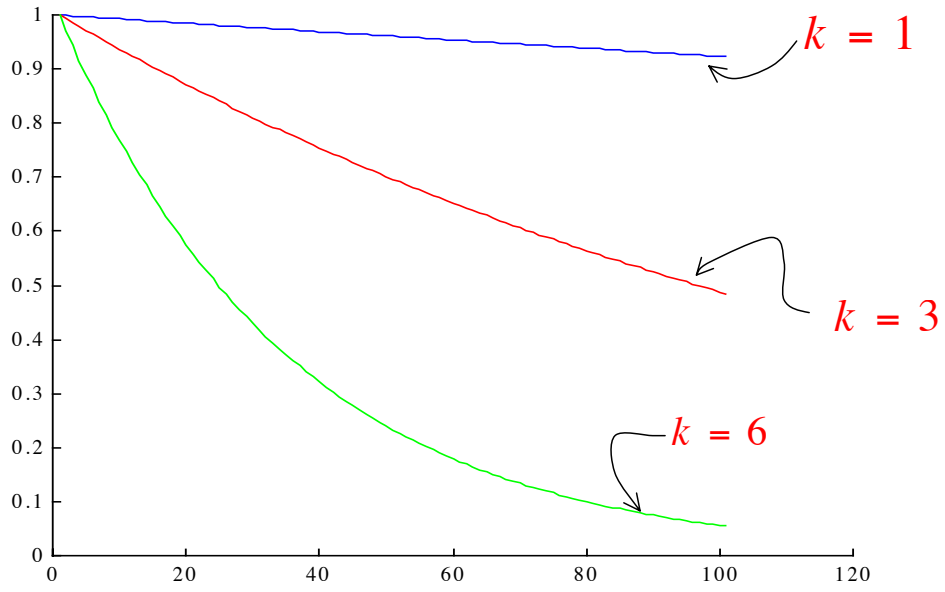


Figure 1.4: Error decay in $\|\cdot\|_\infty$-norm for weighted Jacobi method with initial guess $\vec{\xi}^k$.

Then it is easy to obtain the eigenvalues of the iteration matrix for the weighted Jacobi method

$$\lambda_k(E) = 1 - \omega + \omega \cos\left(\frac{k\pi}{N+1}\right) = \frac{1}{3} + \frac{2}{3}\cos\left(\frac{k\pi}{N+1}\right).$$

From this equation, it is immediately clear that the eigenvalues are small $|\lambda_k(E)| \leqslant \frac{1}{3}$ for larger k ($\frac{N}{2} \leqslant k \leqslant N$). This suggests faster convergence behavior of the weighted Jacobi method for larger k.

Now we can make this simple observation more formal by considering the simple iterative method (1.29), i.e. the Richardson method (it is equivalent to the weighted Jacobi for simple finite difference equations with a constant diagonal), and assume that

$$A\vec{\xi}^k = \lambda_k \vec{\xi}^k, \quad k = 1, \ldots, N,$$

where $0 < \lambda_1 \leqslant \cdots \leqslant \lambda_N$ and we choose $\omega = \frac{1}{\lambda_N}$ for example. Since $\{\vec{\xi}^k\}_{k=1}^N$ forms a basis of $\mathbb{R}^N$, we can write

$$\vec{u} - \vec{u}^{(m)} = \sum_{k=1}^{N} \alpha_k^{(m)} \vec{\xi}^k$$

as an expansion. In the Richardson method, we have

$$\vec{u} - \vec{u}^{(m)} = (I - \omega A)(\vec{u} - \vec{u}^{(m-1)}) = \cdots = (I - \omega A)^m (\vec{u} - \vec{u}^{(0)}).$$

Hence it is easy to see that

$$\sum_{k=1}^{N} \alpha_k^{(m)} \vec{\xi}^k = (I - \omega A)^m \sum_{k=1}^{N} \alpha_k^{(0)} \vec{\xi}^k = \sum_{k=1}^{N} \alpha_k^{(0)} (1 - \omega\lambda_k)^m \vec{\xi}^k.$$

That is to say, we have

$$\alpha_k^{(m)} = (1-\omega\lambda_k)^m \alpha_k^{(0)} = \left(1 - \frac{\lambda_k}{\lambda_N}\right)^m \alpha_k^{(0)}, \quad k = 1, \ldots, N. \tag{1.35}$$

From (1.35), we can see that the convergence speed is fast for high-frequency error components (large k) and slow for low-frequency components (small k).

Smoothing effect of Jacobi method ⋆

In view of Remark 1.26, based on the understanding of the relation between the smoothness and the size of Fourier coefficients, we can analyze the smoothing property using the discrete Fourier expansion. Let $\vec{u}$ be the exact solution of the 1D FD problem on uniform grids and $\vec{u}^{(m)}$ the result of m-th iteration from the damped Jacobi method (or equivalently in this case, the Richardson method). Then

$$\vec{u} - \vec{u}^{(m)} = (I - \omega A)(\vec{u} - \vec{u}^{(m-1)}) = \cdots = (I - \omega A)^m (\vec{u} - \vec{u}^{(0)}).$$

It is straightforward to see that

$$\lambda_k(I - \omega A) = 1 - \omega\lambda_k(A) = 1 - 4\omega \sin^2\left(\frac{k\pi}{2(N+1)}\right).$$

Notice that $\lambda_k(I - \omega A)$ can be viewed as the damping factor for error components corresponding to Fourier mode k; see Remark 1.26. We would like to choose ω such that λ_k's are small.

Consider the Fourier expansion of the initial error:

$$\vec{u} - \vec{u}^{(0)} = \sum_{k=1}^{N} \alpha_k \vec{\xi}^k.$$

Then

$$\vec{u} - \vec{u}^{(m)} = \sum_{k=1}^{N} \alpha_k (I - \omega A)^m \vec{\xi}^k.$$

Note that, for any polynomial p, we have $p(A)\vec{\xi}^k = p(\lambda_k)\vec{\xi}^k$. By choosing $\omega = \frac{1}{4} \approx \frac{1}{\lambda_{\max}(A)}$, we obtain

$$\vec{u} - \vec{u}^{(m)} = \sum_{k=1}^{N} \alpha_k (1 - \omega\lambda_k)^m \vec{\xi}^k = \sum_{k=1}^{N} \alpha_k^{(m)} \vec{\xi}^k,$$

where

$$\alpha_k^{(m)} = \left(1 - \sin^2 \frac{k\pi}{2(N+1)}\right)^m \alpha_k.$$

The above equation implies

$$\alpha_k^{(m)} = \alpha_k \sin^{2m}\left(\frac{N-k+1}{N+1}\frac{\pi}{2}\right) \leqslant \alpha_k \left(\frac{N-k+1}{N+1}\frac{\pi}{2}\right)^{2m},$$

which approaches to 0 very rapidly as $m \to \infty$, if k is close to N (high-frequencies). This means that high frequency error can be damped very quickly. This simple analysis justifies the smoothing property we observed in the beginning of this section.

We can apply the same analysis to the Jacobi method as well and the Fourier coefficient in front of the highest frequency is as follows:

$$\alpha_N^{(m)} = \left(1 - 2\sin^2 \frac{N\pi}{2(N+1)}\right)^m \alpha_N = \cos^m\left(\frac{N\pi}{N+1}\right)\alpha_N \sim (-1)^m\left(1 - \frac{\pi^2}{2(N+1)^2}\right)^m \alpha_N.$$

This suggests that the regular Jacobi method might not have a smoothing property and should not be used as a smoother in general.

1.4 Multigrid method in 1D

In this section, we first give a simple motivation and sneak-peak of the well-known multigrid method, which is a representing example of multilevel iterative methods. The observations of this section will be helpful for our later discussions; see the famous tutorial by Briggs et al. [44] for a quick introduction to the multigrid methods. Consider the *finite difference* scheme (1.26) for the Poisson's equation in 1D, namely

$$A\vec{u} = \vec{f} \qquad \text{with } A = \frac{1}{h^2}\,\text{tridiag}(-1, 2, -1),\ f_i = f(x_i).$$

Nested grids

Multigrid (MG) methods are a group of algorithms for solving partial differential equations using a hierarchy of discretizations. They are very useful in problems exhibiting multiple scales of behavior. In this section, we introduce the simplest multigrid method in 1D.

Suppose there are a hierarchy of $L+1$ grids with mesh sizes $h_l = (\frac{1}{2})^{l+1}$ $(l = 0, 1, \ldots, L)$; see Figure 1.5. It is clear that

$$h_0 > h_1 > h_2 > \cdots > h_L =: h$$

and $N = 2^{L+1} - 1$. We call level L the finest level and level 0 the coarsest level.

Smoothers

We consider how to approximate the solution on each level using some local relaxation method. Assume the 1D Poisson's equation is discretized using the finite difference scheme discussed in the previous section. Then, on each level, we have a linear system of equations

$$A_l\vec{u}_l = \vec{f}_l \qquad \text{with } A_l = h_l^{-2}\,\text{tridiag}(-1, 2, -1).$$

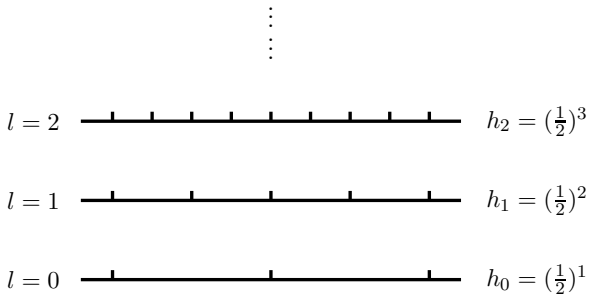


Figure 1.5: Hierarchical grids for 1D multigrid method.

For each of these equations, we can apply the damped Jacobi method (with the damping factor equals to 1/2)

$$\vec{u}_l^{(m+1)} = \vec{u}_l^{(m)} + \frac{1}{2} D_l^{-1} \left(\vec{f}_l - A_l \vec{u}_l^{(m)} \right) \tag{1.36}$$

to obtain an approximate solution. This method is usually referred as a local relaxation or smoother, which will be discussed later in this lecture note.

Prolongation and restriction

Another important component of a multigrid method is to define the transfer operators between different levels. In the 1D case, the transfer operators can be easily given; see Figure 1.6. In another word, we can also write the transfer operators in the matrix form, i.e.,

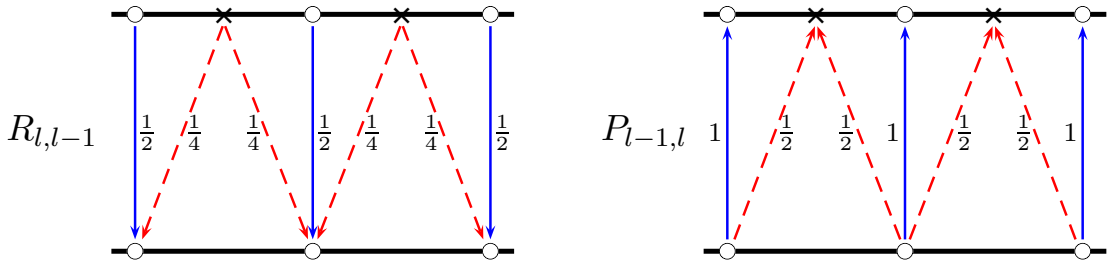


Figure 1.6: Transfer operators between two consecutive levels (Left: restriction operator; right: prolongation operator).

$$R_{l,l-1} := \frac{1}{4}\begin{pmatrix} \ddots & & & & & & \\ & 1 & 2 & 1 & & & \\ & & & 1 & 2 & 1 & \\ & & & & & & \ddots \end{pmatrix} \quad \text{and} \quad P_{l-1,l} := \frac{1}{2}\begin{pmatrix} \ddots & & & \\ & 1 & & \\ & 2 & & \\ & 1 & 1 & \\ & & 2 & \\ & & 1 & \\ & & & \ddots \end{pmatrix}. \tag{1.37}$$

We notice that $R = \frac{1}{2}P^T$. It is straight-forward to check that the coefficient matrices of two consecutive levels satisfy

$$A_{l-1} = R_{l,l-1}A_l P_{l-1,l}.$$

Multigrid algorithm

Let $\vec{f}_l$ be the right-hand side vector and $\vec{u}_l$ be an initial guess or previous iteration on level l. Now we are ready to give one iteration step of the multigrid algorithm (V-cycle).

Algorithm 1.1 (One iteration of multigrid method)**.** $\vec{u}_l = MG(l, \vec{f}_l, \vec{u}_l)$

(i) **Pre-smoothing**: $\vec{u}_l \leftarrow \vec{u}_l + \frac{1}{2}D_l^{-1}\big(\vec{f}_l - A_l\vec{u}_l\big)$

(ii) **Restriction**: $\vec{r}_{l-1} \leftarrow R_{l,l-1}\big(\vec{f}_l - A_l\vec{u}_l\big)$

(iii) **Coarse-grid correction**: If $l = 1$, $\vec{e}_{l-1} \leftarrow A_{l-1}^{-1}\vec{r}_{l-1}$; otherwise, $\vec{e}_{l-1} \leftarrow MG(l-1, \vec{r}_{l-1}, \vec{0}_{l-1})$

(iv) **Prolongation**: $\vec{u}_l \leftarrow \vec{u}_l + P_{l-1,l}\vec{e}_{l-1}$

(v) **Post-smoothing**: $\vec{u}_l \leftarrow \vec{u}_l + \frac{1}{2}D_l^{-1}\big(\vec{f}_l - A_l\vec{u}_l\big)$

Remark 1.35 (Coarse-grid correction)**.** Suppose that there is an approximate solution $\vec{u}^{(m)}$. Then we have

$$A\big(\vec{u} - \vec{u}^{(m)}\big) = \vec{r}^{(m)} := \vec{f} - A\vec{u}^{(m)}$$

and the *error equation* can be written

$$A\vec{e}^{(m)} = \vec{r}^{(m)}. \tag{1.38}$$

If we get $\vec{e}^{(m)}$ or its approximation, we can just update the iterative solution by $\vec{u}^{(m+1)} = \vec{u}^{(m)} + \vec{e}^{(m)}$ to obtain a better approximation of $\vec{u}$. This explains the steps (iii) and (iv) in the above algorithm. □

Remark 1.36 (Coarsest-grid solver)**.** It is clear that, in our setting, the solution on level $l = 0$ is trivial to obtain. In general, we can apply a direct or iterative solver to solve the coarsest-level problem, which is relatively cheap. Sometimes, we have singular problems on the coarsest level, which need to be handled carefully. □

Algorithm 1.1 is one iteration of the multigrid method. We can iterate until the approximation is "satisfactory". For example, we iterate until the relative residual $\|\vec{r}\|_0/\|\vec{f}\|_0$ is less than 10^{-6}; we will discuss stopping criteria later in this lecture. This multigrid algorithm is easy to implement; see HW 1.6. In Table 1.1, we give the numerical results of Algorithm 1.1 for the 1D Poisson's equation (using three G-S iterations as smoother). From the table, we find that, unlike the classical Jacobi and G-S methods, this multigrid method converges uniformly with respect to the meshsize h. This is, of course, a very desirable feature of the multilevel iterative methods, which will be investigated in this lecture.

#Levels	#DOF	#Iter	Contract factor
5	31	4	0.0257
6	63	4	0.0259
7	127	4	0.0260
8	255	4	0.0260
9	511	4	0.0261
10	1023	4	0.0262

Table 1.1: Convergence behavior of 1D geometric multigrid method.

Now it is natural to ask a few questions on such multilevel methods:

- How fast the method converges?
- When does the multigrid method converge?
- How to generalize the method to other problems?
- How to find a good smoother when solving more complicate problems?
- Why the matrices R and P are given as (1.37)? Are there other choices?

And we will mainly focus on these questions in this lecture.

1.5 Tutorial of FASP ⋆

All the numerical examples in this lecture are done using the Fast Auxiliary Space Preconditioning (FASP) package. The FASP package provides C source files[1] to build a library of iterative solvers and preconditioners for the solution of large-scale linear systems of equations. The components of the FASP basic library include several ready-to-use, modern, and efficient iterative solvers used in applications ranging from simple examples of discretized scalar partial differential equations (PDEs) to numerical simulations of complex, multicomponent physical systems.

The main components of the FASP basic library are:

- Basic linear iterative methods;
- Standard Krylov subspace methods;
- Geometric and Algebraic Multigrid (G/AMG) methods;
- Incomplete factorization methods.

The FASP distribution also includes several examples for solving simple benchmark problems. The basic (kernel) FASP distribution is open-source and is licensed under GNU Lesser General Public License or LGPL. Other distributions may have different licensing (contact the developer team for details on this). The most updated version of FASP can be downloaded directly from

http://www.multigrid.org/fasp/download/faspsolver.zip

To build the FASP library for these operating systems. Open a terminal window, where you can issue commands from the command line and do the following: (1) go to the main FASP directory (we will refer to it as `$(faspsolver)` from now on); (2) modify the "*FASP.mk.example*" file to math your system and save it as "FASP.mk"; (3) then execute:

```
> make config
> make install
```

These two commands build the FASP library/header files. By default, it installs the library in `$(faspsolver)/lib` and the header files in `$(faspsolver)/include`. It also creates a file `$(faspsolver)/Config.mk` which contains few of the configuration variables and can be loaded by external project Makefiles. If you do not have "FASP.mk" present in the current directory, default settings will be used for building and installation FASP.

Now, if you would like to try some of the examples that come with FASP, you can build the "tutorial" target and try out the tutorial examples:

[1] The code is C99 (ISO/IEC 9899:1999) compatible.

```
> make tutorial
```

Equivalently, you may also build the test suite and the tutorial examples by using the "local" Makefile in `$(faspsolver)/tutorial`.

```
> make -C tutorial
```

For more information, we refer to the user's guide and reference manual of FASP[2] for technical details on the usage and implementation of FASP. Since FASP is under heavy development, please use this guide with caution because the code might have been changed before this document is updated.

1.6 Homework problems

HW 1.1. Prove the uniqueness of the Poisson's equation. Hint: You can argue by the maximum principle or the energy method.

HW 1.2. Let x_0 and $\delta > 0$ are fixed scales. Find eigenvalues and eigenfunctions of the following local problem

$$-u_\delta''(x) = \lambda_\delta u_\delta, \quad x \in (x_0 - \delta, x_0 + \delta) \qquad \text{and} \qquad u_\delta(x_0 - \delta) = u_\delta(x_0 + \delta) = 0.$$

HW 1.3. Prove the eigenvalues and eigenvectors of $\text{tridiag}(b, a, b) \in \mathbb{R}^{N\times N}$ are

$$\lambda_k = a - 2b\cos\Big(\frac{k\pi}{N+1}\Big) \quad \text{and} \quad \vec{\xi}^k = \Big(\sin\big(\frac{k\pi}{N+1}\big), \ldots, \sin\big(\frac{Nk\pi}{N+1}\big)\Big)^T,$$

respectively. Apply this result to give eigenvalues of the 1D FD matrix A. What are the eigenvalues of $\text{tridiag}(b, a, c) \in \mathbb{R}^{N\times N}$?

HW 1.4. Derive the finite element stiffness matrix for 1D Poisson's equation with homogenous Dirichlet boundary condition using a uniform mesh.

HW 1.5. Derive 1D FD and FE discretizations for the heat equation (8.44) using the backward Euler method for time discretization.

HW 1.6. Implement the geometric multigrid method for the Poisson's equation in 1D using Matlab, C, Fortran, or Python. Try to study the efficiency of your implementation.

HW 1.7. Suppose we need to solve the finite difference equation with coefficient matrix $A := \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{N\times N}$. Plot the eigenvalues of the weighted Jacobi iteration matrix E for $\omega = 1, \frac{2}{3}$, and $\frac{1}{2}$. You can use different problem size N's to get a better view.

[2] Available online at http://www.multigrid.org/fasp. It is also available in "faspsolver/doc/".

Chapter 2

Iterative Solvers and Preconditioners

The term "iterative method" refers to a wide range of numerical techniques that use successive approximations $\left\{u^{(m)}\right\}$ for the exact solution u of a certain problem. In this chapter, we will discuss two types of iterative methods: (1) Stationary iterative method, which performs in each iteration the same operations on the current iteration; (2) Nonstationary iterative method, which has iteration-dependent operations. Stationary methods are simple to understand and implement, but usually not very effective. On the other hand, nonstationary methods are a relatively recent development; their analysis is usually more difficult.

2.1 Stationary linear iterative methods

In this section, we discuss stationary iterative methods; typical examples include the Jacobi method and the Gauss–Seidel method. We will discuss why they are not efficient in general but still widely used. Let V be a finite-dimensional linear vector space, $\mathcal{A} : V \mapsto V$ be a non-singular linear operator, and $f \in V$. We would like to find a $u \in V$, such that

$$\mathcal{A}u = f. \tag{2.1}$$

For example, in the finite difference context discussed in §1.2, $V = \mathbb{R}^N$ and the linear operator $\mathcal{A}$ becomes a matrix A. We just need to solve a system of linear equations: Find $\vec{u} \in \mathbb{R}^N$, such that

$$A\vec{u} = \vec{f}. \tag{2.2}$$

We will discuss the linear systems in both operator and matrix representations.

Remark 2.1 (More general setting)**.** In fact, we can consider iterative methods in a more general setting. For example, let V be a finite-dimensional Hilbert space, V' be its dual, and $\mathcal{A} : V \mapsto V'$ be a linear operator and $f \in V'$. A significant part of this lecture can be generalized to such a setting easily. □

A linear stationary iterative method (one iteration) to solve (2.1) can be expressed in the following general form:

Algorithm 2.1 (Stationary iterative method)**.** $u^{\text{new}} = ITER(u^{\text{old}})$

(i) **Form the residual**: $r = f - \mathcal{A}u^{\text{old}}$

(ii) **Solve or approximate the error equation**: $\mathcal{A}e = r$ by $\hat{e} = \mathcal{B}r$

(iii) **Correct the previous iterative solution**: $u^{\text{new}} = u^{\text{old}} + \hat{e}$

That is to say, the new iteration is obtained by computing

$$u^{\text{new}} = u^{\text{old}} + \mathcal{B}(f - \mathcal{A}u^{\text{old}}), \tag{2.3}$$

where $\mathcal{B}$ is called the *iterator*. Apparently, $\mathcal{B} = \mathcal{A}^{-1}$ for nonsingular operator $\mathcal{A}$ also defines an iterator, which yields a direct method.

Preliminaries and notation

The most-used inner product in this lecture is the Euclidian inner product $(u, v) := \int_\Omega uv\, dx$; and $(u, v) := \sum_{i=1}^N u_i v_i$ if $V = \mathbb{R}^N$. Once we have the inner product, we can define the concept of transpose and symmetry on the Hilbert space V. Define the *adjoint* operator (transpose) of the linear operator $\mathcal{A}$ as $\mathcal{A}^T : V \mapsto V$, such that

$$(\mathcal{A}^T u, v) := (u, \mathcal{A}v), \quad \forall u, v \in V.$$

A linear operator $\mathcal{A}$ on V is *symmetric* if and only if

$$(\mathcal{A}u, v) = (u, \mathcal{A}v), \quad \forall u, v \in \text{domain}(\mathcal{A}) \subseteq V.$$

If $\mathcal{A}$ is densely defined and $\mathcal{A}^T = \mathcal{A}$, then $\mathcal{A}$ is called *self-adjoint*.

Remark 2.2 (Symmetric and self-adjoint operators)**.** A symmetric operator $\mathcal{A}$ is self-adjoint if $\text{domain}(\mathcal{A}) = V$. The difference between symmetric and self-adjoint operators is technical; see [128] for details. □

We denote the null space and the range of $\mathcal{A}$ as

$$\text{null}(\mathcal{A}) := \{v \in V : \mathcal{A}v = 0\}, \tag{2.4}$$

$$\text{range}(\mathcal{A}) := \{u = \mathcal{A}v : v \in V\}. \tag{2.5}$$

Very often, the null space is also called the kernel space and the range is called the image space. The subspaces $\text{null}(\mathcal{A})$ and $\text{range}(\mathcal{A}^T)$ are fundamental subspaces of V. We have

$$\text{null}(\mathcal{A}^T)^\perp = \overline{\text{range}(\mathcal{A})} \quad \text{and} \quad \text{null}(\mathcal{A}^T) = \text{range}(\mathcal{A})^\perp.$$

Remark 2.3 (Non-singularity)**.** If $\text{null}(\mathcal{A}) = \{0\}$, then $\mathcal{A}$ is *injective* or one-to-one. Apparently, $\mathcal{A} : V \mapsto \text{range}(\mathcal{A})$ is *surjective* or onto. If we consider a symmetric operator $\mathcal{A} : \text{null}(\mathcal{A})^\perp \mapsto \text{range}(\mathcal{A})$, then $\mathcal{A}$ is always *non-singular*. □

The set of eigenvalues of $\mathcal{A}$ is called the *spectrum*, denoted as $\sigma(\mathcal{A})$. The spectrum of any bounded symmetric matrix is real, i.e., all eigenvalues are real, although a symmetric operator may have no eigenvalues[1]. We define the spectral radius $\rho(\mathcal{A}) := \sup\{|\lambda| : \lambda \in \sigma(\mathcal{A})\}$. Furthermore,

$$\lambda_{\min}(\mathcal{A}) = \min_{v \in V\backslash\{0\}} \frac{(\mathcal{A}v, v)}{\|v\|^2} \qquad \text{and} \qquad \lambda_{\max}(\mathcal{A}) = \max_{v \in V\backslash\{0\}} \frac{(\mathcal{A}v, v)}{\|v\|^2}.$$

An important class of operators for this lecture is *symmetric positive definite* (SPD) operators. An operator $\mathcal{A}$ is called SPD if and only if $\mathcal{A}$ is symmetric and $(\mathcal{A}v, v) > 0$, for any $v \in V\backslash\{0\}$. Since $\mathcal{A}$ is SPD, all of its eigenvalues are positive. We define the *spectral condition number* or, simply, *condition number* $\kappa(\mathcal{A}) := \frac{\lambda_{\max}(\mathcal{A})}{\lambda_{\min}(\mathcal{A})}$, which is more convenient, compared with spectrum, to characterize convergence rate of iterative methods. For the indefinite case, we can use

$$\kappa(\mathcal{A}) := \frac{\sup_{\lambda\in\sigma(\mathcal{A})} |\lambda|}{\inf_{\lambda\in\sigma(\mathcal{A})} |\lambda|}.$$

More generally, for an isomorphic mapping $\mathcal{A} \in \mathscr{L}(V; V)$, we can define

$$\kappa(\mathcal{A}) := \|\mathcal{A}\|_{\mathscr{L}(V;V)} \|\mathcal{A}^{-1}\|_{\mathscr{L}(V;V)}.$$

And all these definitions are consistent for symmetric positive definite problems.

If $\mathcal{A}$ is an SPD operator, it induces a new inner product, which will be used heavily in our later discussions

$$(u, v)_{\mathcal{A}} := (\mathcal{A}u, v) \qquad \forall u, v \in V. \tag{2.6}$$

It is easy to check $(\cdot, \cdot)_{\mathcal{A}}$ is an inner product on V. For any bounded linear operator $\mathcal{B} : V \mapsto V$, we can define two transposes with respect to the inner products $(\cdot, \cdot)$ and $(\cdot, \cdot)_{\mathcal{A}}$, respectively; namely,

$$(\mathcal{B}^T u, v) = (u, \mathcal{B}v),$$
$$(\mathcal{B}^* u, v)_{\mathcal{A}} = (u, \mathcal{B}v)_{\mathcal{A}}.$$

By the above definitions, it is easy to show (see HW 2.1) that

$$\mathcal{B}^* = \mathcal{A}^{-1}\mathcal{B}^T\mathcal{A}. \tag{2.7}$$

[1] A bounded linear operator on an infinite-dimensional Hilbert space might not have any eigenvalues.

Symmetry is a concept with respect to the underlying inner product. In this chapter, we always refers to the $(\cdot,\cdot)$-inner product for symmetry. By definition, $(\mathcal{BA})^* = \mathcal{B}^T\mathcal{A}$; see HW 2.2 for this equality. If $\mathcal{B}^T = \mathcal{B}$, we do not necessarily have $(\mathcal{BA})^T = \mathcal{BA}$; however, we have a key identity:

$$(\mathcal{BA})^* = \mathcal{B}^T\mathcal{A} = \mathcal{BA}. \tag{2.8}$$

Remark 2.4 (Induced norms)**.** The inner products defined above also induce norms on V by $\|v\| := (v,v)^{\frac{1}{2}}$ and $\|v\|_{\mathcal{A}} := (v,v)_{\mathcal{A}}^{\frac{1}{2}}$. These, in turn, define the operator norms for $\mathcal{B} : V \mapsto V$, i.e.,

$$\|\mathcal{B}\| := \sup_{v\in V\setminus\{0\}} \frac{\|\mathcal{B}v\|}{\|v\|} \qquad \text{and} \qquad \|\mathcal{B}\|_{\mathcal{A}} := \sup_{v\in V\setminus\{0\}} \frac{\|\mathcal{B}v\|_{\mathcal{A}}}{\|v\|_{\mathcal{A}}}.$$

□

It is well-known that, for any consistent norm $\|\cdot\|$, we have $\rho(\mathcal{B}) \leqslant \|\mathcal{B}\|$. Furthermore, we have the following results:

Proposition 2.5 (Spectral radius and norm)**.** Suppose V is Hilbert space with an inner product $(\cdot,\cdot)$ and induced norm $\|\cdot\|$. If $\mathcal{A} : V \mapsto V$ is a bounded linear operator, then

$$\rho(\mathcal{A}) = \lim_{m\to+\infty} \|\mathcal{A}^m\|^{\frac{1}{m}}.$$

Moreover, if $\mathcal{A}$ is self-adjoint, then $\rho(\mathcal{A}) = \|\mathcal{A}\|$.

From this general functional analysis result, we can immediately obtain the following relations:

Lemma 2.6 (Spectral radius of self-adjoint operators)**.** If $\mathcal{B}^T = \mathcal{B}$, then $\rho(\mathcal{B}) = \|\mathcal{B}\|$. Similarly, if $\mathcal{B}^* = \mathcal{B}$, then $\rho(\mathcal{B}) = \|\mathcal{B}\|_{\mathcal{A}}$.

Convergence of stationary iterative methods

Now we consider the convergence analysis of the stationary iterative method (2.3). A method is called convergent if and only if $u^{(m)}$ converges to u for any initial guess $u^{(0)}$.

Notice that each iteration (2.3) only depends on the previous approximate solution u^{old} and does not involve any information of the older iterations; in each iteration, it basically performs the same operations over and over again. It is easy to see that

$$u - u^{(m)} = (\mathcal{I} - \mathcal{BA})\big(u - u^{(m-1)}\big) = \cdots = (\mathcal{I} - \mathcal{BA})^m\big(u - u^{(0)}\big) = \mathcal{E}^m\big(u - u^{(0)}\big),$$

where $\mathcal{I} : V \mapsto V$ is the identity operator and the operator $\mathcal{E} := \mathcal{I} - \mathcal{BA}$ is called the *error propagation operator* (or, sometimes, error reduction operator)[2].

[2]It coincides with the iteration matrix (1.34) or the iterative reduction matrix appeared in the literature on iterative linear solvers.

Lemma 2.6 and (2.8) imply the following identity: If $\mathcal{A}$ is SPD and $\mathcal{B}$ is symmetric, then

$$\rho(\mathcal{I} - \mathcal{B}\mathcal{A}) = \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}. \tag{2.9}$$

Hence we can get the following simple convergence theorem.

Theorem 2.7 (Convergence of Algorithm 2.1)**.** The Algorithm 2.1 converges for any initial guess if the spectral radius $\rho(\mathcal{I}-\mathcal{B}\mathcal{A}) < 1$, which is equivalent to $\lim_{m\to+\infty}(\mathcal{I}-\mathcal{B}\mathcal{A})^m = 0$. The converse direction is also true.

If both $\mathcal{A}$ and $\mathcal{B}$ are SPD, the eigenvalues of $\mathcal{B}\mathcal{A}$ are real and the spectral radius satisfies that

$$\rho(\mathcal{I} - \mathcal{B}\mathcal{A}) = \max\Big(\lambda_{\max}(\mathcal{B}\mathcal{A}) - 1,\, 1 - \lambda_{\min}(\mathcal{B}\mathcal{A})\Big). \tag{2.10}$$

So we can expect that the speed of the stationary linear iterative method is related to the span of spectrum of $\mathcal{B}\mathcal{A}$.

This convergence result is simple but difficult to apply. More importantly, it does not provide any direct information on how fast the convergence could be if the algorithm converges; see the following example for further explanation.

An iterative method converges for any initial guess if and only if the spectral radius of the iteration matrix $\rho(E) < 1$. However, it is important to note that *the spectral radius of E only reflects the asymptotic convergence behavior* of the iterative method. That is to say, we have

$$\frac{\|\vec{e}^{(k+1)}\|}{\|\vec{e}^{(k)}\|} \approx \rho(E),$$

only for very large k.

Example 2.8 (Spectral radius and convergence speed)**.** Suppose we have an iterative method with an error propagation matrix

$$E := \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{N\times N}$$

and the initial error is $\vec{e}^{(0)} := \vec{u} - \vec{u}^{(0)} = (0,\ldots,0,1)^T \in \mathbb{R}^N$. Notice that $\rho(E) \equiv 0$ in this example. However, if applying this error propagation matrix to form a sequence of approximations, we will find the convergence is actually very slow for a large N. In fact,

$$\|\vec{e}^{(0)}\|_2 = \|\vec{e}^{(1)}\|_2 = \cdots = \|\vec{e}^{(N-1)}\|_2 = 1 \qquad \text{and} \qquad \|\vec{e}^{(N)}\|_2 = 0.$$

Hence, analyzing the spectral radius of the iterative matrix alone will not provide much useful information about the speed of an iterative method. □

An alternative measure for convergence speed is to find out whether there is a constant $\delta \in [0,1)$ and a convenient norm $\|\cdot\|$ on $\mathbb{R}^N$, such that $\|\vec{e}^{(m+1)}\| \leqslant \delta \|\vec{e}^{(m)}\|$ for any $\vec{e}^{(0)} \in \mathbb{R}^N$. However, this approach has its own problems because it usually yields pessimistic convergence bound for iterative methods.

Remark 2.9 (Convergence rate of the Richardson method)**.** The simplest iterative method for solving $A\vec{u} = \vec{f}$ might be $B = \omega I$, which is the well-known Richardson method in Example 1.31. In this case, the iteration converges if and only if $\rho(I - \omega A) < 1$, i.e., all eigenvalues of matrix A are in $(0, \frac{2}{\omega})$. Since A is SPD, the iteration converges if $\omega < 2\lambda_{\max}^{-1}(A)$. If we take $\omega = \lambda_{\max}^{-1}(A)$, then

$$\rho\big(I - \lambda_{\max}^{-1}(A)A\big) = 1 - \frac{\lambda_{\min}(A)}{\lambda_{\max}(A)} = 1 - \frac{1}{\kappa(A)}.$$

In fact, the optimal weight is $\omega_{\text{opt}} = \dfrac{2}{\lambda_{\max}(A) + \lambda_{\min}(A)}$ and

$$\rho\big(I - \omega_{\text{opt}}A\big) = \|I - \omega_{\text{opt}}A\| = 1 - \frac{2\lambda_{\min}(A)}{\lambda_{\max}(A) + \lambda_{\min}(A)} = \frac{\kappa(A) - 1}{\kappa(A) + 1}.$$

We can see that the convergence is very slow if A is ill-conditioned. □

Symmetrization

In general, the iterator $\mathcal{B}$ might not be symmetric and it is more convenient to work with symmetric problems. We can apply a simple symmetrization algorithm:

Algorithm 2.2 (Symmetrized iterative method)**.** $u^{\text{new}} = SITER(u^{\text{old}})$

$$u^{(m+\frac{1}{2})} = u^{(m)} + \mathcal{B}\Big(f - \mathcal{A}u^{(m)}\Big), \tag{2.11}$$

$$u^{(m+1)} = u^{(m+\frac{1}{2})} + \mathcal{B}^T\Big(f - \mathcal{A}u^{(m+\frac{1}{2})}\Big). \tag{2.12}$$

In turn, we obtain a new iterative method

$$u - u^{(m+1)} = (\mathcal{I} - \mathcal{B}^T\mathcal{A})(\mathcal{I} - \mathcal{B}\mathcal{A})(u - u^{(m)}) = (\mathcal{I} - \mathcal{B}\mathcal{A})^*(\mathcal{I} - \mathcal{B}\mathcal{A})(u - u^{(m)}).$$

If this new method satisfies the relation

$$u - u^{(m+1)} = (\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})(u - u^{(m)}),$$

then it has a symmetric iteration operator

$$\overline{\mathcal{B}} := \mathcal{B}^T + \mathcal{B} - \mathcal{B}^T\mathcal{A}\mathcal{B} = \mathcal{B}^T(\mathcal{B}^{-T} + \mathcal{B}^{-1} - \mathcal{A})\mathcal{B} =: \mathcal{B}^T\mathcal{K}\mathcal{B}. \tag{2.13}$$

Lemma 2.10 (Error decay property)**.** We have, for any $v \in V$, that

$$\|v\|_{\mathcal{A}}^2 - \|(\mathcal{I} - \mathcal{B}\mathcal{A})v\|_{\mathcal{A}}^2 = \big(\overline{\mathcal{B}}\mathcal{A}v, v\big)_{\mathcal{A}},$$

or equivalently,

$$\big((\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})v, v\big)_{\mathcal{A}} = \|(\mathcal{I} - \mathcal{B}\mathcal{A})v\|_{\mathcal{A}}^2.$$

Proof. Notice that, by the definition of symmetrization,

$$\overline{\mathcal{B}}\mathcal{A} = \mathcal{B}^T(\mathcal{B}^{-T} + \mathcal{B}^{-1} - \mathcal{A})\mathcal{B}\mathcal{A}.$$

This immediately gives

$$\begin{aligned}(\overline{\mathcal{B}}\mathcal{A}v, v)_{\mathcal{A}} &= \big((\mathcal{B}^{-T} + \mathcal{B}^{-1} - \mathcal{A})\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v\big) = (\mathcal{B}\mathcal{A}v, \mathcal{A}v) + (\mathcal{A}v, \mathcal{B}\mathcal{A}v) - (\mathcal{A}\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)\\ &= \big((2\mathcal{I} - \mathcal{B}\mathcal{A})v, \mathcal{B}\mathcal{A}v\big)_{\mathcal{A}}\end{aligned}$$

and the first equality follows immediately. The second equality is trivial. □

Remark 2.11 (Effect of symmetrization)**.** We notice that $\overline{\mathcal{B}}^T = \overline{\mathcal{B}}$ and $(\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})^* = \mathcal{I} - \overline{\mathcal{B}}\mathcal{A}$. Furthermore, Lemma 2.10 shows that $\big((\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})v, v\big)_{\mathcal{A}} = \|(\mathcal{I} - \mathcal{B}\mathcal{A})v\|_{\mathcal{A}}^2$, $\forall v \in V$. Since $\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}$ is self-adjoint w.r.t. $(\cdot,\cdot)_{\mathcal{A}}$, we have $\|\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}\|_{\mathcal{A}} = \rho(\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})$. And as a consequence,

$$\|\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}\|_{\mathcal{A}} = \sup_{\|v\|_{\mathcal{A}}=1} \big((\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})v, v\big)_{\mathcal{A}} = \sup_{\|v\|_{\mathcal{A}}=1} \|(\mathcal{I} - \mathcal{B}\mathcal{A})v\|_{\mathcal{A}}^2 = \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2. \tag{2.14}$$

This immediately gives

$$\rho(\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}) = \|\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}\|_{\mathcal{A}} = \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2 \geqslant \rho(\mathcal{I} - \mathcal{B}\mathcal{A})^2.$$

Hence, if the symmetrized method (2.11)–(2.12) converges, then the original method (2.3) also converges; the opposite direction might not be true though (see Example 2.13). Furthermore, we have obtained the following identity:

$$\|\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}\|_{\mathcal{A}} = \rho(\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}) = \sup_{v \in V\setminus\{0\}} \frac{\big((\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})v, v\big)_{\mathcal{A}}}{\|v\|_{\mathcal{A}}^2}. \tag{2.15}$$

□

For the symmetrized iterative methods, we have the following theorem.

Theorem 2.12 (Convergence of Symmetrized Algorithm)**.** The symmetrized iteration, namely, Algorithm 2.2, is convergent if and only if $\overline{\mathcal{B}}$ is SPD.

Proof. First of all, we notice that

$$\mathcal{I} - \overline{\mathcal{B}}\mathcal{A} = (\mathcal{I} - \mathcal{B}^T\mathcal{A})(\mathcal{I} - \mathcal{B}\mathcal{A}) = \mathcal{A}^{-\frac{1}{2}}(\mathcal{I} - \mathcal{A}^{\frac{1}{2}}\mathcal{B}^T\mathcal{A}^{\frac{1}{2}})(\mathcal{I} - \mathcal{A}^{\frac{1}{2}}\mathcal{B}\mathcal{A}^{\frac{1}{2}})\mathcal{A}^{\frac{1}{2}},$$

which has the same spectrum as the operator $(\mathcal{I}-\mathcal{A}^{\frac{1}{2}}\mathcal{B}^T\mathcal{A}^{\frac{1}{2}})(\mathcal{I}-\mathcal{A}^{\frac{1}{2}}\mathcal{B}\mathcal{A}^{\frac{1}{2}})$. Hence, all eigenvalues of $\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}$ are non-negative, i.e., $\lambda \leqslant 1$ for all $\lambda \in \sigma(\overline{\mathcal{B}}\mathcal{A})$.

The convergence of Algorithm 2.2 is equivalent to $\rho(\mathcal{I}-\overline{\mathcal{B}}\mathcal{A}) < 1$. Since $\sigma(\mathcal{I}-\overline{\mathcal{B}}\mathcal{A}) = \{1-\lambda : \lambda \in \sigma(\overline{\mathcal{B}}\mathcal{A})\}$, it follows that Algorithm 2.2 converges if and only if $\sigma(\overline{\mathcal{B}}\mathcal{A}) \subseteq (0, 2)$. Therefore, the convergence of (2.11)–(2.12) is equivalent to $\sigma(\overline{\mathcal{B}}\mathcal{A}) \subseteq (0, 1]$, i.e., $\overline{\mathcal{B}}\mathcal{A}$ is SPD w.r.t. $(\cdot,\cdot)_{\mathcal{A}}$. Hence the result. □

We can also easily obtain the contraction property in a different way. In Lemma 2.10, we have already seen that

$$\|(\mathcal{I} - \mathcal{B}\mathcal{A})v\|_{\mathcal{A}}^2 = \|v\|_{\mathcal{A}}^2 - (\overline{\mathcal{B}}\mathcal{A}v, \mathcal{A}v).$$

Hence, $\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} < 1$ if and only if $\overline{\mathcal{B}}$ is SPD.

Example 2.13 (Convergence condition)**.** Note that even if $\overline{\mathcal{B}}$ is not SPD, the method defined by $\mathcal{B}$ could still converge. For example, in $\mathbb{R}^2$, if

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -2 \\ 0 & 1 \end{bmatrix}, \quad \text{and} \quad I - BA = \begin{bmatrix} 0 & 2 \\ 0 & 0 \end{bmatrix},$$

then we have

$$\overline{B} = \begin{bmatrix} 1 & 0 \\ 0 & -3 \end{bmatrix} \quad \text{and} \quad I - \overline{B}A = \begin{bmatrix} 0 & 0 \\ 0 & 4 \end{bmatrix}.$$

Hence $\rho(I - BA) = 0 < 4 = \rho(I - \overline{B}A)$. Apparently, the iterator B converges but $\overline{B}$ does not. □

Convergence rate of stationary iterative methods

Remark 2.14 (Contraction property)**.** The stationary iterative method defined by $\mathcal{B}$ is a contraction if $\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} \leqslant \delta_0 < 1$. Apparently, it is equivalent to say

$$\|e\|_{\mathcal{A}}^2 - \|(\mathcal{I} - \mathcal{B}\mathcal{A})e\|_{\mathcal{A}}^2 \geqslant (1 - \delta_0^2)\|e\|_{\mathcal{A}}^2 > 0, \qquad \forall e \neq 0.$$

Lemma 2.10 indicates that $\delta := \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} < 1$ if and only if $\overline{\mathcal{B}}$ is SPD. The constant δ is called the contraction factor of the iterative method. From this point on, we can assume that all the iterators $\mathcal{B}$ are SPD; in fact, if an iterator is not symmetric, we can consider its symmetrization instead.

Based on the identity (2.15), we can prove the convergence rate estimate:

Theorem 2.15 (Convergence rate)**.** If $\overline{\mathcal{B}}$ is SPD, the convergence rate of the stationary iterative method (or its symmetrization) is

$$\|\mathcal{I}-\mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2=\|\mathcal{I}-\overline{\mathcal{B}}\mathcal{A}\|_{\mathcal{A}}=1-\frac{1}{c_1},\quad \text{with } c_1:=\sup_{\|v\|_{\mathcal{A}}=1}(\overline{\mathcal{B}}^{-1}v,v).$$

Proof. The first equality is directly from (2.14). Since $((\mathcal{I}-\overline{\mathcal{B}}\mathcal{A})v,v)_{\mathcal{A}}=\|v\|_{\mathcal{A}}^2-(\overline{\mathcal{B}}\mathcal{A}v,v)_{\mathcal{A}}$, the identity (2.15) yields

$$\|\mathcal{I}-\mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2=1-\inf_{\|v\|_{\mathcal{A}}=1}(\overline{\mathcal{B}}\mathcal{A}v,v)_{\mathcal{A}}=1-\lambda_{\min}(\overline{\mathcal{B}}\mathcal{A})=1-\frac{1}{c_1},$$

where

$$c_1=\lambda_{\max}((\overline{\mathcal{B}}\mathcal{A})^{-1})=\sup_{\|v\|_{\mathcal{A}}=1}\left((\overline{\mathcal{B}}\mathcal{A})^{-1}v,v\right)_{\mathcal{A}}=\sup_{\|v\|_{\mathcal{A}}=1}\left(\overline{\mathcal{B}}^{-1}v,v\right).$$

This in turn gives the second equality. □

Example 2.16 (Jacobi and weighted Jacobi methods)**.** If $A\in\mathbb{R}^{N\times N}$ is SPD and it can be partitioned as $A=L+D+U$, where $L,D,U\in\mathbb{R}^{N\times N}$ are lower triangular, diagonal, upper triangular parts of A, respectively. We can immediately see that $B=D^{-1}$ yields the Jacobi method. In this case, we have

$$\overline{B}=B^T(B^{-T}+B^{-1}-A)B=D^{-T}(D-L-U)D^{-1}.$$

If $K_{\text{Jacobi}}:=D-L-U=2D-A$ is SPD, the Jacobi method converges. In general, it might not converge, but we can apply an appropriate scaling (i.e., the damped Jacobi method) $B_\omega=\omega D^{-1}$. We then derive

$$B_\omega^{-T}+B_\omega^{-1}-A=2\omega^{-1}D-A.$$

The damping factor should satisfy that $\omega<\frac{2}{\rho(D^{-1}A)}$ in order to guarantee convergence. For the 1D finite difference problem of the Poisson's equation, we should use a *damping* factor $0<\omega<1$. □

An example: modified G-S method ⋆

Similar to the weighted Jacobi method (see Example 2.16), we define the weighted G-S method $B_\omega=(\omega^{-1}D+L)^{-1}$. We have

$$B_\omega^{-T}+B_\omega^{-1}-A=(\omega^{-1}D+L)^T+(\omega^{-1}D+L)-(D+L+U)=(2\omega^{-1}-1)D.$$

The weighted G-S method converges if and only if $0<\omega<2$. In fact, $\omega=1$ yields the standard G-S method; $0<\omega<1$ yields the *SUR* method; $1<\omega<2$ yields the *SOR* method. One can select optimal weights for different problems to achieve good convergence result, which is beyond the scope of this lecture.

Motived by the weighted G-S methods, we assume there is an invertible smoother or a local relaxation method S for the equation $A\vec{u} = \vec{f}$, like the damped Jacobi smoother $S = \omega D^{-1}$ $(0 < \omega < 1)$. We can define a general or modified G-S method:

$$B := \left(S^{-1} + L\right)^{-1}. \tag{2.16}$$

This method seems abstract and not very interesting now; but we will employ this idea on block matrices for multilevel iterative methods later on.

We can analyze the convergence rate of this modified G-S method using the same technique discussed above. Since $K = B^{-T} + B^{-1} - A$ is a symmetric operator and we can write (2.13) as $\overline{B} = B^T K B$. If B is the iteration operator defined by (2.16), we have

$$K = (S^{-T} + U) + (S^{-1} + L) - (D + L + U) = S^{-T} + S^{-1} - D.$$

Furthermore, from the definition of K, we find that $B^{-1} = K + A - B^{-T}$. Hence we get an explicit form of $\overline{B}^{-1}$ by simple calculations:

$$\overline{B}^{-1} = (K + A - B^{-T})K^{-1}(K + A - B^{-1}) = A + (A - B^{-T})K^{-1}(A - B^{-1}).$$

This identity and the definition of B yield:

$$\left(\overline{B}^{-1}\vec{v}, \vec{v}\right) = (A\vec{v}, \vec{v}) + \left(K^{-1}(D + U - S^{-1})\vec{v},\, (D + U - S^{-1})\vec{v}\right), \quad \forall \vec{v} \in \mathbb{R}^N.$$

Now we apply Theorem 2.15 and get the following identity for the convergence rate:

Corollary 2.17 (Convergence rate of Modified G-S)**.** If $K = S^{-T} + S^{-1} - D$ is SPD, then the modified G-S method converges and

$$\|I - BA\|_A^2 = \|I - \overline{B}A\|_A = 1 - \frac{1}{1 + c_0}, \quad \text{with } c_0 := \sup_{\|\vec{v}\|_A = 1} \left\| K^{-\frac{1}{2}}\left(D + U - S^{-1}\right)\vec{v} \right\|^2.$$

This simple result will motivate our later analysis for subspace correction methods in Chapter 4.

Example 2.18 (Solving 1D Poisson's equation using G-S)**.** If we apply the G-S method to the 1D FD/FE system (1.26) for the Poisson's equation discussion in §1.2. For simplicity, we first rescale both sides of the equation such that $A := \text{tridiag}(-1, 2, -1)$ and $\vec{f} := \left(h^2 f(x_i)\right)_{i=1}^N$. In this case, $S = D^{-1}$ and $K = D$ in the above modified G-S method. Corollary 2.17 shows that the convergence rate of the G-S iteration satisfies that

$$\|I - BA\|_A^2 = 1 - \frac{1}{1 + c_0}, \quad \text{with } c_0 = \sup_{\vec{v} \in \mathbb{R}^N \setminus \{0\}} \frac{(LD^{-1}U\vec{v}, \vec{v})}{\|\vec{v}\|_A^2}.$$

The positive constant can be further written

$$c_0 = \sup_{\vec{v}\in\mathbb{R}^N\setminus\{0\}} \frac{\left(D^{-1}U\vec{v}, U\vec{v}\right)}{(A\vec{v},\vec{v})} = \sup_{\vec{v}\in\mathbb{R}^N\setminus\{0\}} \frac{\frac{1}{2}\left(U\vec{v}, U\vec{v}\right)}{(A\vec{v},\vec{v})} = \sup_{\vec{v}\in\mathbb{R}^N\setminus\{0\}} \frac{\frac{1}{2}\sum_{i=2}^{N} v_i^2}{(A\vec{v},\vec{v})}.$$

Because we have the eigenvalues of this discrete coefficient matrix A of FD (see Remark 1.26), we can estimate the denominator

$$(A\vec{v},\vec{v}) \geqslant \lambda_{\min}(A)\|\vec{v}\|^2 = 4\sin^2\left(\frac{\pi}{2(N+1)}\right)\|\vec{v}\|^2.$$

Hence, asymptotically, we have the following estimate

$$c_0 \leqslant \sup_{\vec{v}\in\mathbb{R}^N\setminus\{0\}} \frac{\frac{1}{2}\|\vec{v}\|^2}{4\sin^2\left(\frac{\pi}{2(N+1)}\right)\|\vec{v}\|^2} \sim (N+1)^2 = h^{-2}.$$

Hence

$$\|I - BA\|_A \sim \sqrt{1-\tilde{C}h^2} \sim 1 - Ch^2.$$

Similarly, for the FE equation, the condition number also likes $O(h^{-2})$ and convergence rate will deteriorate as the meshsize decreases. □

2.2 Krylov subspace methods

Nonstationary iterative methods are more popular for standard-alone usage. Krylov subspace method (KSM) is a well-known class of nonstationary methods [64]. Let $\mathcal{A} : V \mapsto V$ be an invertible operator. By the Cayley–Hamilton theorem (see HW 2.3), there exists a polynomial of degree no more than $N-1$, $q_{N-1}(\lambda) \in \mathcal{P}_{N-1}$, such that $\mathcal{A}^{-1} = q_{N-1}(\mathcal{A})$. Hence the solution of the linear system has the form $u = q_{N-1}(\mathcal{A})f$. Krylov subspace methods construct iterative approximations to u in

$$\mathcal{K}_m := \operatorname{span}\{f, \mathcal{A}f, \mathcal{A}^2 f, \ldots, \mathcal{A}^{m-1}f\}, \quad m = 1, 2, \ldots$$

Gradient descent method

Let $\mathcal{A} : V \mapsto V$ be an SPD operator. Consider the following convex minimization problem:

$$\min_{u\in V} \mathcal{F}(u) := \frac{1}{2}(\mathcal{A}u, u) - (f, u). \tag{2.17}$$

Suppose we have an initial approximation u^{old} and construct a new approximation

$$u^{\text{new}} = u^{\text{old}} + \alpha\, p$$

with a fixed search direction $p \in V$ and a stepsize α. In order to find the "best possible" stepsize, we can solve an one-dimensional problem (i.e., the exact line-search method):

$$\min_{\alpha\in\mathbb{R}} \mathcal{F}(\alpha) := \frac{1}{2}\big(u^{\text{old}} + \alpha p, u^{\text{old}} + \alpha p\big)_{\mathcal{A}} - (f, u^{\text{old}} + \alpha p).$$

By simple calculation (HW 2.4), we obtain

$$\mathcal{F}(\alpha) := \frac{1}{2}\alpha^2(\mathcal{A}p, p) - \alpha(f - \mathcal{A}u^{\text{old}}, p) + \frac{1}{2}\big(\mathcal{A}u^{\text{old}}, u^{\text{old}}\big) - (f, u^{\text{old}}),$$

and the optimal stepsize is

$$\alpha_{\text{opt}} = \frac{(f - \mathcal{A}u^{\text{old}}, p)}{(\mathcal{A}p, p)} = \frac{(r^{\text{old}}, p)}{(\mathcal{A}p, p)}, \quad \text{with } r^{\text{old}} = f - \mathcal{A}u^{\text{old}}. \tag{2.18}$$

In the previous chapter, we have discussed the Richardson method. A nonstationary version of the Richardson method can be given as:

$$u^{(m+1)} = u^{(m)} + \alpha_m\big(f - \mathcal{A}u^{(m)}\big),$$

which can be viewed as the gradient descent or steepest descent (SD) method with exact line-search for the above convex minimization problem.

Remark 2.19 (Richardson and steepest descent method)**.** If A is a SPD matrix, then $A\vec{u} = \vec{f}$ is equivalent to the unconstrained quadratic minimization problem

$$\operatorname*{argmin}_{\vec{u}\in\mathbb{R}^N} \frac{1}{2}\vec{u}^T A\vec{u} - \vec{f}^T\vec{u}.$$

We immediately notice that the search direction in the Richardson method is exactly the same as the steepest decent method for the above minimization problem. □

This method is easy to implement and cheap in computation (each step only requires 1 matrix-vector multiplication and 2 inner products). Unfortunately, the SD method usually converges very slowly. See the following algorithm description of the SD method:

Listing 2.1: Steepest descent method

```
%% Given an initial guess u and a tolerance ε;
r ← f − Au;
while ‖r‖ > ε
    α ← (r, r)/(Ar, r);
    u ← u + α r;
    r ← r − αAr;
end
```

Example 2.20 (Line-search and the G-S method)**.** Let $V = \mathbb{R}^N$, $A = (a_{i,j}) \in \mathbb{R}^{N\times N}$. Suppose we choose the natural basis as the search directions, i.e., $\vec{p} = \vec{e}_i := (0, \dots, 0, 1, 0, \dots, 0)^T \in V$. Let $\vec{u}^{\,\text{old}} = \vec{u}^{\,(0)}$ be an initial guess. Then the above method yields the iteration:

$$\vec{u}^{\,(i)} = \vec{u}^{\,(i-1)} + \alpha\vec{p} = \vec{u}^{\,(i-1)} + \frac{(\vec{r}^{\,(i-1)}, \vec{p})}{(A\vec{p}, \vec{p})}\vec{p} = \vec{u}^{\,(i-1)} + \frac{(\vec{r}^{\,(i-1)}, \vec{e}_i)}{(A\vec{e}_i, \vec{e}_i)}\vec{e}_i.$$

So we get

$$\vec{u}^{\,(i)} = \vec{u}^{\,(i-1)} + \frac{f_i - \sum_{j=1}^N a_{i,j}\, u_j^{(i-1)}}{a_{i,i}}\vec{e}_i.$$

This means that only one entry is updated in each iteration:

$$u_i^{\text{new}} = u_i^{(i-1)} + \frac{f_i - \sum_{j=1}^N a_{i,j}\, u_j^{(i-1)}}{a_{i,i}} = \frac{1}{a_{i,i}}\Big(f_i - \sum_{j<i} a_{i,j}u_j^{\text{new}} - \sum_{j>i} a_{i,j}u_j^{\text{old}}\Big). \tag{2.19}$$

After N steps ($i = 1, 2, \dots, N$), we obtain a new iteration $\vec{u}^{\,\text{new}}$, which is exactly the G-S iteration. □

Remark 2.21 (The G-S method and Schwarz method)**.** Based on (2.19), we can write the G-S error propagation matrix in a different form

$$I - BA = (I - I_N a_{N,N}^{-1} I_N^T A)\cdots(I - I_1 a_{1,1}^{-1} I_1^T A) = (I - \Pi_N)\cdots(I - \Pi_1), \tag{2.20}$$

where I_i is the natural embedding from $\text{span}\{\vec{e}_i\}$ to $\mathbb{R}^N$ and $\Pi_i = I_i A_i^{-1} I_i^T A$. This form of G-S will be further discussed later in the framework of Schwarz method and subspace correction method. □

Theorem 2.22 (Convergence rate of steepest descent method)**.** If we apply the exact line-search using the stepsize

$$\alpha_m := \frac{\big(r^{(m)}, r^{(m)}\big)}{\big(r^{(m)}, r^{(m)}\big)_{\mathcal{A}}},$$

then the convergence rate of the SD method satisfies that

$$\big\|u - u^{(m)}\big\|_{\mathcal{A}} \leqslant \left(\frac{\kappa(\mathcal{A}) - 1}{\kappa(\mathcal{A}) + 1}\right)^m \big\|u - u^{(0)}\big\|_{\mathcal{A}}. \tag{2.21}$$

Proof. The exact line-search stepsize is easy to obtain by 1D quadratic programming. At the m-th iteration, the energy satisfies that

$$\mathcal{F}(u^{(m+1)}) = \mathcal{F}(u^{(m)} + \alpha_m r^{(m)}) = \mathcal{F}(u^{(m)}) - \alpha_m(r^{(m)}, r^{(m)}) + \frac{1}{2}\alpha_m^2(\mathcal{A}r^{(m)}, r^{(m)}).$$

By plugging the expression of α_m into the right-had side of the above equality, we obtain that

$$\mathcal{F}(u^{(m+1)}) = \mathcal{F}(u^{(m)}) - \frac{1}{2}\frac{(r^{(m)}, r^{(m)})^2}{(\mathcal{A}r^{(m)}, r^{(m)})}.$$

This implies that

$$\begin{aligned}\frac{\mathcal{F}(u^{(m+1)})-\mathcal{F}(u)}{\mathcal{F}(u^{(m)})-\mathcal{F}(u)} &= \frac{\mathcal{F}(u^{(m)})-\frac{(r^{(m)},r^{(m)})^2}{2(\mathcal{A}r^{(m)},r^{(m)})}-\mathcal{F}(u)}{\mathcal{F}(u^{(m)})-\mathcal{F}(u)} \\ &= 1-\frac{(r^{(m)},r^{(m)})^2}{(\mathcal{A}r^{(m)},r^{(m)})(\mathcal{A}^{-1}r^{(m)},r^{(m)})} =: 1-\frac{1}{\beta}\end{aligned}$$

By the Kantorovich inequality, we know $\beta \leqslant \frac{(\lambda_{\max}+\lambda_{\min})^2}{4\lambda_{\max}\lambda_{\min}}$. So it follows

$$\frac{\mathcal{F}(u^{(m+1)})-\mathcal{F}(u)}{\mathcal{F}(u^{(m)})-\mathcal{F}(u)} = 1-\frac{1}{\beta} \leqslant 1-\frac{4\lambda_{\max}\lambda_{\min}}{(\lambda_{\max}+\lambda_{\min})^2} = \frac{(\lambda_{\max}-\lambda_{\min})^2}{(\lambda_{\max}+\lambda_{\min})^2} = \left(\frac{\kappa(\mathcal{A})-1}{\kappa(\mathcal{A})+1}\right)^2.$$

Hence the result. □

Conjugate gradient method

Now we consider a descent direction method with search direction $p^{(m)}$, i.e.

$$u^{(m+1)} = u^{(m)} + \alpha_m p^{(m)}. \tag{2.22}$$

In this case, the "optimal" stepsize from the exact line-search is

$$\alpha_m := \frac{\left(r^{(m)},p^{(m)}\right)}{\left(p^{(m)},p^{(m)}\right)_{\mathcal{A}}}. \tag{2.23}$$

We notice that the residual after one iteration is

$$r^{(m+1)} = r^{(m)} - \alpha_m \mathcal{A}p^{(m)}.$$

In order to keep the iteration going, we wish to construct a new search direction which is orthogonal to the previous search directions. This motives us to define

$$p^{(m+1)} := r^{(m+1)} + \beta_m p^{(m)}, \quad \text{such that } \left(p^{(m)},p^{(m+1)}\right)_{\mathcal{A}} = 0.$$

By simple calculations, we get the weight

$$\beta_m := -\frac{\left(\mathcal{A}r^{(m+1)},p^{(m)}\right)}{\left(\mathcal{A}p^{(m)},p^{(m)}\right)}. \tag{2.24}$$

This is basically the so-called conjugate gradient (CG) method.

Lemma 2.23 (Properties of conjugate directions)**.** For any conjugate gradient step i, we have following identities:

1. $\left(r^{(i)},p^{(i)}\right) = \left(r^{(i)},r^{(i)}\right)$;

2. $\left(r^{(j)}, p^{(i)}\right) = 0, \qquad j > i;$

3. $\left(p^{(j)}, p^{(i)}\right)_{\mathcal{A}} = 0, \qquad j \neq i;$

4. $\left(r^{(j)}, r^{(i)}\right) = 0, \qquad j \neq i.$

This lemma is very simple but important; see HW 2.5. It guarantees we can apply a short recurrence iteration procedure while keep all directions are orthogonal to each other.

Lemma 2.24 (Stepsizes for CG)**.** For the conjugate gradient method, we have following identities:

$$\alpha_m = \frac{\left(r^{(m)}, r^{(m)}\right)}{\left(\mathcal{A}p^{(m)}, p^{(m)}\right)} \qquad \text{and} \qquad \beta_m = \frac{\left(r^{(m+1)}, r^{(m+1)}\right)}{\left(r^{(m)}, r^{(m)}\right)}.$$

The previous lemma may look like some trivial transformations, but it is essential for CG implementation, which is described as follows:

Listing 2.2: Conjugate gradient method

```
%% Given an initial guess u and a tolerance ε;
r ← f − Au,  p ← r;
while ‖r‖ > ε
    α ← (r, r)/(Ap, p);
    ũ ← u + α p;
    r̃ ← r − αAp;
    β ← (r̃, r̃)/(r, r);
    p̃ ← r̃ + β p;
    Update:  u ← ũ,  r ← r̃,  p ← p̃;
end
```

Remark 2.25 (Computational complexity of CG)**.** We find that, in each iteration of the CG method, the complexity is only 1 matrix-vector multiplication and 2 inner products, with a few vector additions. □

The CG method converges much faster than the steepest descent in practice. In fact, we have the following theorem

Theorem 2.26 (Convergence rate of CG)**.** The convergence rate of the CG iteration satisfies the following estimate:

$$\left\|u - u^{(m)}\right\|_{\mathcal{A}} \leqslant 2\left(\frac{\sqrt{\kappa(\mathcal{A})} - 1}{\sqrt{\kappa(\mathcal{A})} + 1}\right)^m \left\|u - u^{(0)}\right\|_{\mathcal{A}}. \tag{2.25}$$

Proof. We only give a sketch of proof here. From Lemma 2.23, the residual $r^{(m)}$ is orthogonal to

$$\mathcal{K}_m = \operatorname{span}\{r^{(0)}, \mathcal{A}r^{(0)}, \ldots, \mathcal{A}^{m-1}r^{(0)}\},$$

namely

$$(\mathcal{A}(u-u^{(m)}),v)=(r^{(m)},v)=0,\quad \forall v\in\mathcal{K}_m.$$

This implies

$$\big((u-u^{(0)})-(u^{(m)}-u^{(0)}),v\big)_{\mathcal{A}}=0,\quad \forall v\in\mathcal{K}_m.$$

The above $\mathcal{A}$-orthogonality gives

$$\begin{aligned}\left\|u-u^{(m)}\right\|_{\mathcal{A}} &= \min_{w\in\mathcal{K}_m}\left\|u-u^{(0)}-w\right\|_{\mathcal{A}} = \min_{q_{m-1}}\left\|u-u^{(0)}-q_{m-1}(\mathcal{A})r^{(0)}\right\|_{\mathcal{A}}\\ &= \min_{q_{m-1}}\left\|\big(I-q_{m-1}(\mathcal{A})\mathcal{A}\big)(u-u^{(0)})\right\|_{\mathcal{A}} = \min_{q_m(0)=1}\left\|q_m(\mathcal{A})(u-u^{(0)})\right\|_{\mathcal{A}}.\end{aligned}$$

The desired estimate can then be obtained by choosing appropriate Chebyshev polynomials; see HW 2.6 as a guideline to complete the proof. □

Remark 2.27 (Minimum residual method)**.** If $\mathcal{A}:V\mapsto V$ is a symmetric isomorphism mapping and it is indefinite, we can apply the minimum residual (MINRES) method characterized by

$$u^{(m)}=\operatorname*{argmin}_{v\in\mathcal{K}_m}\|f-\mathcal{A}v\|_0^2.$$

We can derive analytically that (see, for example, [64])

$$\left\|r^{(m)}\right\|_0\leqslant\min_{q_m(0)=1}\max_{\lambda\in\sigma(\mathcal{A})}|q_m(\lambda)|\left\|r^{(0)}\right\|_0$$

In this case, the following crude convergence estimate holds

$$\left\|r^{(m)}\right\|_0=\left\|\mathcal{A}(u-u^{(m)})\right\|_0\leqslant 2\left(\frac{\kappa(\mathcal{A})-1}{\kappa(\mathcal{A})+1}\right)^m\left\|\mathcal{A}(u-u^{(0)})\right\|_0=2\left(\frac{\kappa(\mathcal{A})-1}{\kappa(\mathcal{A})+1}\right)^m\left\|r^{(0)}\right\|_0.\quad(2.26)$$

If all the eigenvalues are positive, we can get sharp convergence estimate using Chebyshev polynomials. Unfortunately, it is not easy to get a general yet sharp estimate for indefinite problems. □

Effective condition number ⋆

If the spectrum of $\mathcal{A}$ is uniformly distributed in the interval $[\lambda_{\min},\lambda_{\max}]$, then the upper bound in (2.25) is sharp. In fact a few "bad eigenvalues" have almost no effect on the asymptotic convergence of the method. In this case, this bound is not sharp any more. Instead, the asymptotic convergence rate can be estimated by the *effective condition number* [2, 3].

If the spectrum of $\mathcal{A}$ can be decomposed into two parts, $\sigma(\mathcal{A})=\sigma_{\text{eff}}(\mathcal{A})\bigcup\sigma_{\text{iso}}(\mathcal{A})$, with m_0 isolated eigenvalues in $\sigma_{\text{iso}}(\mathcal{A})$. In this case, the above convergence estimate for CG can be modified as

$$\frac{\|u-u^{(m)}\|_{\mathcal{A}}}{\|u-u^{(0)}\|_{\mathcal{A}}}\leqslant 2C\left(\frac{\sqrt{b/a}-1}{\sqrt{b/a}+1}\right)^{m-m_0},\qquad m\geqslant m_0,\quad(2.27)$$

where the constant $C := \max_{\lambda\in\sigma_{\text{eff}}(\mathcal{A})} \prod_{\mu\in\sigma_{\text{iso}}(\mathcal{A})} \left|1 - \frac{\lambda}{\mu}\right|$. It is easy to see that $C \leqslant \big(\kappa(A) - 1\big)^{m_0}$ in general; and, in particular, $C \leqslant 1$ if σ_{iso} contains only isolated large eigenvalues.

Hence we can define the *effective condition number* as

$$\kappa_{\text{eff}}(\mathcal{A}) := \frac{b}{a} = \frac{\max \sigma_{\text{eff}}}{\min \sigma_{\text{eff}}}$$

and use the effective condition number to estimate the rate of convergence of the Krylov subspace methods instead.

Generalizing KSM to Hilbert spaces

It is important to note that the above convergence estimates (2.25) and (2.26) do not depend on the finite dimensionality N. Hence the Krylov subspace methods (KSMs) can be applied for operators $\mathcal{A} : \mathscr{V} \mapsto \mathscr{V}$, where $\mathscr{V}$ is a separable Hilbert space[3]. In view of Remark 1.23, we have

$$\|\mathcal{A}\|_{\mathscr{L}(\mathscr{V};\mathscr{V})} = \sup_{v\in\mathscr{V}} \frac{(\mathcal{A}v, v)}{\|v\|_{\mathscr{V}}^2} = \sup_{v\in\mathscr{V}} \frac{a[v, v]}{\|v\|_{\mathscr{V}}^2} \leqslant C_a$$

and the inf-sup condition (1.16) gives

$$\|\mathcal{A}^{-1}\|_{\mathscr{L}(\mathscr{V};\mathscr{V})}^{-1} = \inf_{v\in\mathscr{V}} \frac{\|\mathcal{A}v\|_{\mathscr{V}}}{\|v\|_{\mathscr{V}}} = \inf_{v\in\mathscr{V}} \sup_{u\in\mathscr{V}} \frac{(\mathcal{A}v, u)}{\|v\|_{\mathscr{V}}\|u\|_{\mathscr{V}}} = \inf_{v\in\mathscr{V}} \sup_{u\in\mathscr{V}} \frac{a[v, u]}{\|v\|_{\mathscr{V}}\|u\|_{\mathscr{V}}} \geqslant \alpha.$$

Hence the condition number $\kappa(\mathcal{A}) \leqslant C_a/\alpha$, which is bounded.

However, in order to employ KSMs for the continuous equations that we are interested in, like the Poisson's equation, we have to consider $\mathcal{A} : \mathscr{V} \mapsto \mathscr{W}$, where $\mathscr{V}$ and $\mathscr{W}$ are both separable Hilbert spaces. Typically, $\mathscr{W} \supset \mathscr{V}$ and most likely $\mathscr{W} = \mathscr{V}'$. For simplicity, we consider a symmetric isomorphism $\mathcal{A} \in \mathscr{L}(\mathscr{V}; \mathscr{V}')$, i.e.,

$$\langle \mathcal{A}u, v\rangle = \langle \mathcal{A}v, u\rangle, \quad u, v \in \mathscr{V},$$

where $\langle\cdot,\cdot\rangle$ is the duality pair. Since $\mathscr{V}' \not\subset \mathscr{V}$, KSMs are not well-defined in this case. The question is how we can apply a KSM method in such a setting.

We need to construct an isomorphism $\mathcal{B}$ mapping $\mathscr{V}'$ back to $\mathscr{V}$. We assume that the map $\mathcal{B}$ is symmetric and positive definite, namely $\langle\cdot, \mathcal{B}\cdot\rangle$ defines an inner product in $\mathscr{V}'$. We immediately notice that $\mathcal{B}$ could be a Riesz operator[4]: For any given $f \in \mathscr{V}'$,

$$(\mathcal{B}f, v)_{\mathscr{V}} = \langle f, v\rangle, \quad \forall v \in \mathscr{V}.$$

As a consequence, $\langle \mathcal{B}^{-1}\cdot,\cdot\rangle$ is an inner product on $\mathscr{V}$, with associated norm equivalent to $\|\cdot\|_{\mathscr{V}}$. This leads to a so-called preconditioned system

$$\mathcal{B}\mathcal{A}u = \mathcal{B}f$$

[3] $\mathscr{V}$ might not be finite dimensional.

[4] We note that, here, $\mathcal{B}$ is inner product dependent.

and $\mathcal{BA}$ is an isomorphism from $\mathscr{V}$ to itself. The Krylov subspace methods can be applied to this preconditioned system and $\mathcal{B}$ is called a preconditioner.

Note that $\mathcal{BA} : \mathscr{V} \mapsto \mathscr{V}$ is symmetric with respect to $(\cdot,\cdot)_{\mathscr{V}}$, i.e.,

$$(\mathcal{BA}u, v)_{\mathscr{V}} = \langle \mathcal{A}u, v\rangle = a[u, v] = (u, \mathcal{BA}v)_{\mathscr{V}}, \quad u, v \in \mathscr{V}.$$

The last equality follows from the symmetry of the bilinear form $a[\cdot,\cdot]$. Furthermore, due to the continuity of $a[\cdot,\cdot]$ (1.14), we obtain

$$\|\mathcal{BA}\|_{\mathscr{L}(\mathscr{V};\mathscr{V})} = \sup_{v\in\mathscr{V}} \frac{|(\mathcal{BA}v, v)_{\mathscr{V}}|}{\|v\|^2_{\mathscr{V}}} = \sup_{v\in\mathscr{V}} \frac{a[v, v]}{\|v\|^2_{\mathscr{V}}} \leqslant C_a$$

and the inf-sup condition (1.16) gives

$$\|(\mathcal{BA})^{-1}\|^{-1}_{\mathscr{L}(\mathscr{V};\mathscr{V})} = \inf_{v\in\mathscr{V}} \frac{\|\mathcal{BA}v\|_{\mathscr{V}}}{\|v\|_{\mathscr{V}}} = \inf_{v\in\mathscr{V}} \sup_{u\in\mathscr{V}} \frac{(\mathcal{BA}v, u)_{\mathscr{V}}}{\|v\|_{\mathscr{V}}\|u\|_{\mathscr{V}}} = \inf_{v\in\mathscr{V}} \sup_{u\in\mathscr{V}} \frac{a[v, u]}{\|v\|_{\mathscr{V}}\|u\|_{\mathscr{V}}} \geqslant \alpha.$$

This discussion directly follows the work by Mardal and Winther [83].

Example 2.28 (Poisson solver as a preconditioner)**.** As an example, we consider a second-order elliptic operator $\mathcal{A} : H^1_0(\Omega) \mapsto H^{-1}(\Omega)$. We need to define

$$(\mathcal{B}f, v)_{H^1_0(\Omega)} := (\nabla\mathcal{B}f, \nabla v)_{0,\Omega} = \langle f, v\rangle.$$

In this sense, we can choose $\mathcal{B} = (-\Delta)^{-1}$ as a preconditioner. We note that other inner products can be used, which will yield different preconditioners. As long as the above continuity condition and the inf-sup condition hold, the preconditioned system is well-conditioned. □

Now we summarize the above discussion on how to construct a "natural" preconditioner:

1. Define an appropriate inner product $(\cdot,\cdot)_{\mathscr{V}}$;
2. Establish the inf-sup condition $\sup_{v\in\mathscr{V}} \frac{a[u,v]}{\|v\|_{\mathscr{V}}} \geqslant \alpha\|u\|_{\mathscr{V}}$ for any $u \in \mathscr{V}$;
3. Define $\mathcal{B}$ as the Reisz operator, i.e., $(\mathcal{B}f, v)_{\mathscr{V}} = \langle f, v\rangle$ for any $v \in \mathscr{V}$;
4. The preconditioned system $\mathcal{BA}$ is symmetric with respect to $(\cdot,\cdot)_{\mathscr{V}}$ and well-conditioned;
5. Construct a discretization which satisfies the corresponding discrete inf-sup condition;
6. Define a spectrally equivalent $\mathcal{B}_h$ as a preconditioner.

2.3 Condition number and preconditioning

The convergence rate of an iterative method depends greatly on the spectrum of the coefficient matrix. Hence, iterative methods usually involve a second matrix that transforms the coefficient matrix into one with a more favorable spectrum. The transformation matrix is called a preconditioner. A good preconditioner $\mathcal{B}$ improves the convergence of the iterative method sufficiently and is relatively cheap to compute, in order to overcome the overhead (extra cost) of constructing and applying the preconditioner. There are a few different ways to apply preconditioners, for example:

$\mathcal{B}\mathcal{A}u = \mathcal{B}f$		Left preconditioning
$\mathcal{A}\mathcal{B}v = f$	$u = \mathcal{B}v$	Right preconditioning
$\mathcal{B}_L\mathcal{A}\mathcal{B}_R v = \mathcal{B}_L f$	$u = \mathcal{B}_R v$	Split preconditioning

Although convergence behavior of iterative methods is not governed by the condition number alone, it provides useful information for a variety of methods. For example, we would hope that $\kappa(\mathcal{B}\mathcal{A}) \ll \kappa(\mathcal{A})$, if we apply a Krylov subspace method to solve a preconditioned linear system.

Construction of preconditioners

It is desirable to have an effective preconditioner which satisfy most, if not all, of the following properties:

- The preconditioned linear systems have improved convergence behavior. Furthermore, the spectral condition number of $\mathcal{B}\mathcal{A}$ should be bounded independently of the size of the problem.
- The preconditioner is relatively easy to setup and cheap to apply. The computational cost of $\mathcal{B}r$ should be proportional to the size of the problem.
- The preconditioner should be robust on different domain shapes, mesh types, jumps in coefficients, etc.
- The preconditioner can be implemented easily and efficiently.

We first introduce a few simple facts that could be helpful when we need to estimate the condition number $\kappa(\mathcal{B}\mathcal{A})$.

Lemma 2.29 (Estimation of condition number)**.** If μ_0 and μ_1 are positive constants satisfying

$$\mu_0\big(\mathcal{A}u, u\big) \leqslant \big(\mathcal{B}^{-1}u, u\big) \leqslant \mu_1\big(\mathcal{A}u, u\big), \quad \forall u \in V, \tag{2.28}$$

then the condition number

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant \frac{\mu_1}{\mu_0}.$$

Proof. By change of variable $u = \mathcal{A}^{-\frac{1}{2}}v$, we have $\sigma\big(\mathcal{A}^{-\frac{1}{2}}\mathcal{B}^{-1}\mathcal{A}^{-\frac{1}{2}}\big) \subseteq \big[\mu_0, \mu_1\big]$ and, hence, $\sigma\big((\mathcal{B}\mathcal{A})^{-1}\big) \subseteq \big[\mu_0, \mu_1\big]$. □

Sometimes, it is more convenient to use some equivalent conditions of (2.28) to analyze condition number; see the following lemma. Proof of the following lemma is left to the readers as an exercise; see HW 2.7.

Lemma 2.30 (Some equivalent conditions)**.** If $\mathcal{A}$ and $\mathcal{B}$ are symmetric positive definite operators on a finite-dimensional space V, then we have the inequalities (2.28) are equivalent to

$$\mu_0(\mathcal{B}u, u) \leqslant (\mathcal{A}^{-1}u, u) \leqslant \mu_1(\mathcal{B}u, u), \quad \forall u \in V, \tag{2.29}$$

or

$$\mu_1^{-1}(\mathcal{A}u, u) \leqslant (\mathcal{A}\mathcal{B}\mathcal{A}u, u) \leqslant \mu_0^{-1}(\mathcal{A}u, u), \quad \forall u \in V, \tag{2.30}$$

or

$$\mu_1^{-1}(\mathcal{B}u, u) \leqslant (\mathcal{B}\mathcal{A}\mathcal{B}u, u) \leqslant \mu_0^{-1}(\mathcal{B}u, u), \quad \forall u \in V. \tag{2.31}$$

Remark 2.31 (Another equivalent condition)**.** If $\mathcal{A}$ and $\mathcal{B}$ are symmetric positive definite operators on a finite-dimensional space V, $\alpha > 0$ and $0 < \delta < 1$, then it is easy to verify the following two conditions are equivalent:

$$-\alpha(\mathcal{A}u, u) \leqslant (\mathcal{A}(\mathcal{I} - \mathcal{B}\mathcal{A})u, u) \leqslant \delta(\mathcal{A}u, u), \quad \forall u \in V \tag{2.32}$$

and

$$(1+\alpha)^{-1}(\mathcal{A}u, u) \leqslant (\mathcal{B}^{-1}u, u) \leqslant (1-\delta)^{-1}(\mathcal{A}u, u), \quad \forall u \in V. \tag{2.33}$$

□

Preconditioned conjugate gradient method

Before we talk about preconditioned KSMs, the first question to answer is why and how CG can be applied to the preconditioned system $\mathcal{B}\mathcal{A}u = \mathcal{B}f$. We have mentioned $\mathcal{B}\mathcal{A}$ is usually not symmetric w.r.t. $(\cdot,\cdot)$ but symmetric w.r.t. $(\cdot,\cdot)_{\mathcal{A}}$. Similarly, we can define a new inner product $(\cdot,\cdot)_{\mathcal{B}^{-1}} := (\mathcal{B}^{-1}\cdot,\cdot)$. Then

$$(\mathcal{B}\mathcal{A}\cdot,\cdot)_{\mathcal{B}^{-1}} = (\mathcal{A}\cdot,\cdot) \quad \Longrightarrow \quad \mathcal{B}\mathcal{A} \text{ is SPD w.r.t. } (\cdot,\cdot)_{\mathcal{B}^{-1}},$$

which means CG can be applied to $\mathcal{B}\mathcal{A}u = \mathcal{B}f$ with the new inner product.

Lemma 2.32 (Stepsizes of PCG)**.** For the preconditioned conjugate gradient method, we have the following identities:

$$\alpha_m = \frac{\big(\mathcal{B}r^{(m)}, r^{(m)}\big)}{\big(\mathcal{A}p^{(m)}, p^{(m)}\big)} \qquad \text{and} \qquad \beta_m = \frac{\big(\mathcal{B}r^{(m+1)}, r^{(m+1)}\big)}{\big(\mathcal{B}r^{(m)}, r^{(m)}\big)}.$$

We notice that $\mathcal{B}^{-1}$ is cancelled out in the above inner products. With the help of this lemma, we can write the pseudo-code of PCG with left preconditioner (compared with regular CG, it just requires one more matrix-vector multiplication):

Listing 2.3: Preconditioned conjugate gradient method

```
%% Given an initial guess u and a tolerance ε;
r ← f − Au,  p ← Br;
while ‖r‖ > ε
    α ← (Br, r)/(Ap, p);
    ũ ← u + α p;
    r̃ ← r − αAp;
    β ← (Br̃, r̃)/(Br, r);
    p̃ ← Br̃ + β p;
    Update:  u ← ũ,  r ← r̃,  p ← p̃;
end
```

Preconditioning v.s. iteration

Let $\mathcal{B}$ be a symmetric iterator of the SPD operator $\mathcal{A}$. We have seen that a sufficient condition for the iterative method to be convergent is that

$$\rho(\mathcal{I} - \mathcal{B}\mathcal{A}) < 1.$$

In this case, $\rho := \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} < 1$. The method is not only converging but also a contraction, i.e., $\|u - u^{(m)}\|_{\mathcal{A}} \leqslant \rho^m \|u - u^{(0)}\|_{\mathcal{A}} \to 0$ as $m \to +\infty$. Similar argument as Theorem 2.12 shows that $\mathcal{B}$ must be SPD. Furthermore, by definition, we have

$$\Big(\big(\mathcal{A} - 2\mathcal{A}\mathcal{B}\mathcal{A} + \mathcal{A}\mathcal{B}\mathcal{A}\mathcal{B}\mathcal{A}\big)u, u\Big) \leqslant \rho^2 (u,u)_{\mathcal{A}}.$$

Changing variable $v = \mathcal{A}^{1/2}u$, we obtain

$$\begin{aligned} \Big(\big(\mathcal{I} - \mathcal{A}^{1/2}\mathcal{B}\mathcal{A}^{1/2}\big)^2 v, v\Big) \leqslant \rho^2 (v,v) \quad &\Longrightarrow \quad \Big|\big((\mathcal{I} - \mathcal{A}^{1/2}\mathcal{B}\mathcal{A}^{1/2})v, v\big)\Big| \leqslant \rho (v,v) \\ &\Longrightarrow \quad \Big|\big((\mathcal{A} - \mathcal{A}\mathcal{B}\mathcal{A})u, u\big)\Big| \leqslant \rho (\mathcal{A}u, u), \quad \forall u \in V. \end{aligned}$$

Hence Remark 2.31 shows (see HW 2.9) that the condition number is uniformly bounded, i.e.,

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant \frac{1+\rho}{1-\rho}.$$

In fact, the above estimate can also be easily obtained from $\rho(\mathcal{I}-\mathcal{B}\mathcal{A})=\rho<1$.

We use the same notation $\mathcal{B}$ for the preconditioner and the iterator, apparently for a reason. Indeed, the convergence rate of the preconditioned CG method (2.25) is equal to

$$\delta_{\mathrm{CG}}:=\frac{\sqrt{\kappa(\mathcal{B}\mathcal{A})}-1}{\sqrt{\kappa(\mathcal{B}\mathcal{A})}+1}\leqslant\frac{\sqrt{\frac{1+\rho}{1-\rho}}-1}{\sqrt{\frac{1+\rho}{1-\rho}}+1}=\frac{1-\sqrt{1-\rho^2}}{\rho}<\rho.$$

The last inequality holds true when $0<\rho<1$. Hence, for any convergent stationary linear iterative method, a preconditioner can be found and its convergence can be accelerated by PCG. Of course, it comes with the extra cost of applying the preconditioners. Preconditioning is so important for practical problems and KSMs are sometimes referred as accelerators.

Stopping criteria ⋆

When an iterative method is employed, sometimes it is hard to determine when to stop the iteration process. Ultimately we would like to have the error $e^{(m)}=u-u^{(m)}$ in certain norm (e.g. the energy norm) to be small enough, i.e., $\left(e^{(m)},e^{(m)}\right)_{\mathcal{A}}^{\frac{1}{2}}<\epsilon$. However, the error is not usually computable. Norms of the residual $r^{(m)}=f-\mathcal{A}u^{(m)}$, which is not only computable but also naturally available in the iterative process, are used instead. According to the standard perturbation analysis, we have

$$\frac{\left\|u-u^{(m)}\right\|}{\|u\|}\leqslant\kappa(\mathcal{A})\frac{\left\|r^{(m)}\right\|}{\|f\|}. \tag{2.34}$$

In fact, $\mathcal{A}(u-u^{(m)})=f-\mathcal{A}u^{(m)}=r^{(m)}$. Hence $\|u-u^{(m)}\|\leqslant\|\mathcal{A}^{-1}\|\|r^{(m)}\|$. On the other hand, it is easy to see that $\|f\|\leqslant\|\mathcal{A}\|\|u\|$. By combining the last two inequalities, we can obtain the desired estimate (2.34). We notice that the right-hand side of (2.34) is the relative residual (with initial guess equals zero) and the left-hand side is just the relative error. Hence this inequality shows that, even if the relative residual is small, the relative error could be still very large, especially for the ill-conditioned problems.

Although L^2-norm of $r^{(m)}$ is usually used in practice, $\left(r^{(m)},r^{(m)}\right)_{\mathcal{B}}^{\frac{1}{2}}$ is a better quantity to monitor for convergence. We notice that

$$(r^{(m)},r^{(m)})_{\mathcal{B}}=\left(\mathcal{A}e^{(m)},\mathcal{A}e^{(m)}\right)_{\mathcal{B}}=\left(\mathcal{A}\mathcal{B}\mathcal{A}e^{(m)},e^{(m)}\right).$$

According to Lemma 2.30, $\left(r^{(m)},r^{(m)}\right)_{\mathcal{B}}^{\frac{1}{2}}$ is equivalent to $\left(e^{(m)},e^{(m)}\right)_{\mathcal{A}}^{\frac{1}{2}}$, if and only if $\mathcal{B}$ is a *good* preconditioner.

Another comment is that we have been using the residual of the original equation instead of the preconditioned equation in PCG. In practice, there might be situations that left part of the preconditioner changes the residual of the equation a lot, which will cause trouble for users to

design stopping criteria. The preconditioned equation has a residual $r_{\mathcal{B}} = \mathcal{B}r = \mathcal{B}(f - \mathcal{A}u)$ and $\|r_{\mathcal{B}}\|$ might be a lot different than $\|r\|$. Thus it is usually not good to use $r_{\mathcal{B}}$ instead of r.

2.4 Domain decomposition methods

In the field of numerical methods for partial differential equations, domain decomposition methods (DDMs) make use of *divide and conquer* techniques by iteratively solving subproblems defined on smaller subdomains. It is a convenient framework for the solution and, more importantly, preconditioning of heterogeneous or multiphysics problems. It can be used in the framework of many discretization methods (e.g., FD and FE) to make their algebraic solution efficient, especially on parallel computers. Roughly speaking, there are two ways of subdividing the computational domain, overlapping and non-overlapping. We will only discuss overlapping domain decomposition methods here.

Divide and conquer

We consider the model boundary value problem

$$\begin{cases} \mathcal{A}u &= f & \text{in } \Omega \\ u &= 0 & \text{on } \partial\Omega \end{cases}$$

Overlapping domain decomposition algorithms are based on a decomposition of the domain Ω into a number of overlapping subdomains. To introduce the main ideas of DDMs, we consider the case of two overlapping subdomains Ω_1 and Ω_2, which form a covering of Ω and $\Omega_1 \bigcap \Omega_2 \neq \emptyset$; see Figure 1. We let Γ_i $(i = 1, 2)$ denote the part of the boundary of Ω_i, which is in the interior of Ω.

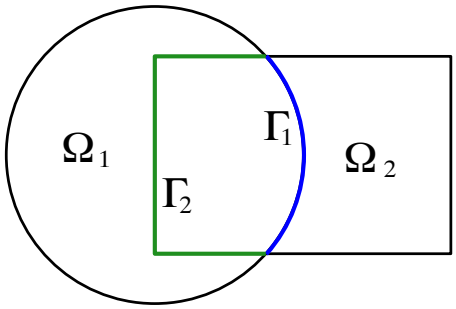


Figure 2.1: Overlapping domain partition with two sub-domains.

If we already have an approximate solution $u^{(m)}$, we can construct a new approximation by

solving the following two equations:

$$\begin{cases} \mathcal{A}u_1^{(m+1)} &= f & \text{in } \Omega_1, \\ u_1^{(m+1)} &= u^{(m)} & \text{on } \Gamma_1, \\ u_1^{(m+1)} &= 0 & \text{on } \partial\Omega_1 \backslash \Gamma_1, \end{cases}$$

and

$$\begin{cases} \mathcal{A}u_2^{(m+1)} &= f & \text{in } \Omega_2, \\ u_2^{(m+1)} &= g^{(m)} & \text{on } \Gamma_2, \\ u_2^{(m+1)} &= 0 & \text{on } \partial\Omega_2 \backslash \Gamma_2. \end{cases}$$

Here we have not specified how to choose the right boundary condition $g^{(m)}$. There are two approaches to apply these two subdomain corrections—the additive approach and the multiplicative approach. In the additive approach, we take $g^{(m)} = u^{(m)}$ and carry out the two corrections simultaneously. In the multiplicative approach, we take $g^{(m)} = u_1^{(m+1)}$ and use the most up-to-date iterative solution. We then define the new iteration as

$$u^{(m+1)}(x) := \begin{cases} u_2^{(m+1)}, & \text{if } x \in \Omega_2; \\ u_1^{(m+1)}, & \text{if } x \in \Omega \backslash \Omega_2. \end{cases}$$

Overlapping DD methods

With the above motivation in mind, we are ready to introduce the standard overlapping domain decomposition method in matrix form:

$$A\vec{u} = \vec{f}, \qquad V = \mathbb{R}^N.$$

Suppose we have an one-dimensional domain partitioning of Ω; see Figure 2.2. Of course, we can use more general partitioning strategies as well.

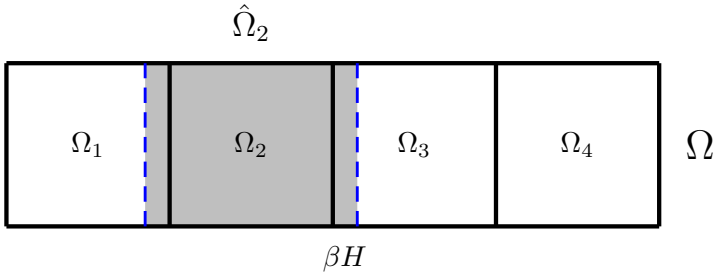


Figure 2.2: Overlapping domain partition with four sub-domains.

Denote the set of grid point indices as $G := \{1, 2, \ldots, N\}$ and it is partitioned into n subdomains. Let $\hat{G}_i$ be the index set of the interior points of $\hat{\Omega}_i$, and $N_i := |\hat{G}_i|$ be the cardinality of

$\hat{G}_i$. Apparently, we have

$$G = \hat{G}_1 \bigcup \hat{G}_2 \bigcup \cdots \bigcup \hat{G}_n \qquad \text{and} \qquad N < N_1 + N_2 + \cdots + N_n.$$

In the matrix form, the injection matrix (natural embedding) $I_i \in \mathbb{R}^{N \times N_i}$ is defined as

$$(I_i \vec{v}_i)_k = \begin{cases} (\vec{v}_i)_k, & \text{if } k \in \hat{G}_i; \\ 0, & \text{if } k \in G \backslash \hat{G}_i. \end{cases} \tag{2.35}$$

It is natural to define sub-problems as $A_i := I_i^T A I_i$ $(i = 1, \ldots, n)$. If we solve each sub-problem exactly, then we have $B_i := I_i A_i^{-1} I_i^T$.

We can define an *additive Schwarz method* (ASM) as

$$B_{\text{as}} := \sum_{i=1}^{n} B_i = \sum_{i=1}^{n} I_i A_i^{-1} I_i^T, \tag{2.36}$$

which generalizes the block Jacobi method. Similarly, a *multiplicative Schwarz method* (MSM) is then defined by the following error propagation operator

$$I - B_{\text{ms}} A := (I - B_n A) \cdots (I - B_1 A) = \prod_{i=n}^{1} (I - B_i A). \tag{2.37}$$

This is a generalization of the block G-S method (with overlapping blocks). In practice, the sub-problem solver A_i^{-1} could be replaced by an approximation, like the ILU method.

Classical convergence results of overlapping DDMs ⋆

These DD methods, especially the ASM version, are usually applied as preconditioners for parallel computing. Its convergence has been analyzed in [49, 50] and we only show the results for the additive version here.

Theorem 2.33 (AS DD preconditioner)**.** The condition number of AS domain decomposition method is independent of the mesh size h and satisfies

$$\kappa(B_{\text{as}} A) \lesssim H^{-2}(1 + \beta^{-2}),$$

where H is size of domain partitions and βH characterizes size of the overlaps.

The DD preconditioner (2.36) performs very well in practice. But the convergence rate still depends on H and the condition number could be large if H is very small. A simple approach to get rid of this dependence on H is to introduce a coarse space $V_0 \subset V$ and a corresponding coarse-level solver, i.e.

$$B_{\text{as},2} := I_0 A_0^{-1} I_0^T + \sum_{i=1}^{n} I_i A_i^{-1} I_i^T,$$

where $I_0 : V_0 \mapsto V$ is the injection matrix and A_0 is the coarse space problem. We then have the following estimate on the condition number:

Theorem 2.34 (Two-level AS DD preconditioner)**.** The condition number of AS domain decomposition method is independent of the mesh size h and satisfies

$$\kappa(B_{\mathrm{as},2}A) \lesssim 1 + \beta^{-1}.$$

The above theorem shows the dependence on meshsize can be removed by introducing an appropriate coarse-level correction. We will construct and analyze two-level and, more generally, multilevel iterative methods in the following chapters.

2.5 Homework problems

HW 2.1. Show the identity (2.7).

HW 2.2. If $\mathcal{B}^T = \mathcal{B}$, show that $(\mathcal{B}\mathcal{A})^* = \mathcal{B}^T\mathcal{A} = \mathcal{B}\mathcal{A}$.

HW 2.3. Let $A \in \mathbb{R}^{N\times N}$ and $q(\lambda) := \big|\lambda I - A\big|$ be the characteristic polynomial of A. Show the Cayley-Hamilton theorem, i.e., $q(A) = 0$.

HW 2.4. Show the optimal stepsize (2.18) for general descent direction method.

HW 2.5. Prove Lemmas 2.23 and 2.24.

HW 2.6. The Chebyshev (or Tchebycheff) polynomial of first kind on $[-1, 1]$ can be defined recursively as

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x).$$

Show that

$$T_n(x) = \frac{1}{2}\Big(\big(x + \sqrt{x^2-1}\big)^n + \big(x - \sqrt{x^2-1}\big)^n\Big)$$

and $\big|T_n(x)\big| \leqslant 1$ for any $x \in [-1, 1]$. Let $0 < \lambda_{\min} \leqslant \lambda_{\max}$. Define

$$S_n(\lambda) := \left[T_n\Big(\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}}\Big)\right]^{-1} T_n\Big(\frac{\lambda_{\max} + \lambda_{\min} - 2\lambda}{\lambda_{\max} - \lambda_{\min}}\Big)$$

and we have

$$\left|T_n\Big(\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}}\Big)\right|^{-1} = \big\|S_n\big\|_{\infty,[\lambda_{\min},\lambda_{\max}]} = \min_{p\in\mathcal{P}_n;\, p(0)=1} \big\|p\big\|_{\infty,[\lambda_{\min},\lambda_{\max}]},$$

where $\mathcal{P}_n$ is the set of polynomials of degree less than or equal to n.

HW 2.7. Prove Lemma 2.30.

HW 2.8. Show that (2.32) and (2.33) are equivalent to each.

HW 2.9. Let $\mathcal{A}$ be SPD and $\mathcal{B}$ be a symmetric iterator. If $\rho = \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} < 1$, then $\mathcal{B}$ is also SPD and

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant \frac{1+\rho}{1-\rho}.$$

Chapter 3

Twogrid Methods

In the previous chapter, we have seen several simple iterative solvers and preconditioners for solving the linear algebraic system (2.1). The convergence rate of these methods deteriorates when meshsize h approaches zero, except for the two-level overlapping domain decomposition method with coarse-grid correction. This motivates our discussions on multilevel iterative methods in the following chapters. In this chapter, we will discuss the twogrid (or more generally, two-level) method for the discrete Poisson's equation:

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \partial\Omega. \end{cases} \quad \Longrightarrow \quad A\vec{u} = \vec{f}.$$

In Chapter 1, we have briefly discussed the finite element approximation for this model problem. From now on, we will mainly discuss in the context of finite element discretizations. Throughout this chapter, we use the standard notations for Sobolev spaces introduced in Chapter 1: $H^k(\Omega)$ denotes the classical Sobolev space of scalar functions on a bounded domain $\Omega \subset \mathbb{R}^d$ whose derivatives up to order k are square integrable, with the full norm $\|\cdot\|_k$ and the corresponding semi-norm $|\cdot|_k$. The symbol $H_0^1(\Omega)$ denotes the subspace of $H^1(\Omega)$ whose trace vanishes on the boundary $\partial\Omega$. We will also discuss the corresponding spaces restricted to the subdomain of Ω.

3.1 Finite element methods

We now take a little detour and say a few more words about the finite element discretizations; see [38] for more details. The linear operator $\mathcal{A} : \mathscr{V} \mapsto \mathscr{V}'$ is defined by

$$(\mathcal{A}u, v) := a[u, v] = \int_\Omega \nabla u \nabla v \, dx, \quad \forall v \in \mathscr{V}$$

and $f \in \mathscr{V}'$ is a function or distribution. Suppose that $\mathcal{A}$ is bounded (1.14), i.e.,

$$a[u,v] \leqslant C_a \|u\|_{\mathscr{V}} \|v\|_{\mathscr{V}}, \quad \forall\, u,v \in \mathscr{V}$$

and coercive (1.23), i.e.,

$$a[v,v] \geqslant \alpha \|v\|_{\mathscr{V}}^2, \quad \forall v \in \mathscr{V}.$$

We would like to find $u \in \mathscr{V}$ such that $\mathcal{A}u = f$ or in the weak form

$$a[u,v] = \langle f, v \rangle, \quad \forall\, v \in \mathscr{V} \tag{3.1}$$

which is well-posed. And we have seen that this problem is well-conditioned in Remark 1.23.

Galerkin approximation

The *Galerkin method* exploits the weak formulation (3.1) and replaces the underlying function space by appropriate finite dimensional subspaces. We choose a finite dimensional space V_N (trial/test space), which is an approximation to the space $\mathscr{V}$ with $\dim(V_N) = N$. When no confusion arises, we shall just drop the subscript and denote the space as $V = V_N$. Then we arrive at the Galerkin discretization:

$$\text{Find } u_N \in V: \quad a[u_N, v_N] = \langle f, v_N \rangle, \quad \forall\, v_N \in V. \tag{3.2}$$

Equation (3.2) yields the so-called *Galerkin discretization*. If the bilinear form $a[\cdot,\cdot]$ is symmetric and coercive, it is called the Ritz–Galerkin discretization. In the finite-dimensional setting, we can identify the dual space V' and V; this way, the duality pair $\langle \cdot, \cdot \rangle$ becomes the l^2-inner product $(\cdot,\cdot)$.

For conforming discretizations, the bilinear form $a[\cdot,\cdot]$ is well-defined on $V \times V$. If the bilinear form $a[\cdot,\cdot]$ is coercive, then we have

$$a[v_N, v_N] \geqslant \alpha_{_N} \|v_N\|_{\mathscr{V}}^2, \quad \forall\, v_N \in V.$$

Since coercivity is inherited from $\mathscr{V}$ to its subspace V, we can see that the constant α_N is bounded from below, i.e.,

$$\alpha_{_N} \geqslant \alpha, \quad \forall N$$

As a consequence, the discrete inf-sup condition holds[1]. It is easy to show the following simple optimality approximation properties.

[1] In general, the continuous inf-sup condition does not imply the discrete one.

Remark 3.1 (Galerkin Orthogonality)**.** Assume $V \subset \mathscr{V}$. The weak formulations of the exact and discrete solutions satisfy

$$\begin{cases} a[u, v] = \langle f, v \rangle, & \forall\ v \in \mathscr{V}; \\ a[u_N, v_N] = \langle f, v_N \rangle, & \forall\ v_N \in V. \end{cases}$$

Taking $v = v_N$ in the first equation and simply subtracting the two equations gives the Galerkin orthogonality, i.e.,

$$a[u - u_N, v_N] = 0, \quad \forall\, v_N \in V. \tag{3.3}$$

If $a[\cdot, \cdot]$ is symmetric and coercive, then (3.3) means the error $u - u_N$ is orthogonal to V in the induced inner product by the bilinear form $a[\cdot, \cdot]$. Apparently, $\Pi_N u := u_N$ is a projection from $\mathscr{V}$ to V with respect to $(\cdot, \cdot)_{\mathcal{A}}$-inner product. It is oftentimes called the *Ritz projection*. □

Lemma 3.2 (Céa's Lemma)**.** If the bilinear form $a[\cdot, \cdot]$ is continuous and coercive, then the Galerkin approximation u_N satisfies

$$\|u - u_N\|_{\mathscr{V}} \leqslant \frac{C_a}{\alpha} \|u - v_N\|_{\mathscr{V}}, \quad \forall\, v_N \in V.$$

More generally, we have the following *quasi-optimality* or *quasi-best-approximation* of the finite-dimensional Galerkin approximation.

Proposition 3.3 (Quasi-Optimality)**.** Suppose $a[\cdot, \cdot] : \mathscr{V} \times \mathscr{V} \mapsto \mathbb{R}$ is continuous. The finite dimensional subspace V in the Galerkin approximation satisfies the discrete inf-sup condition (1.22) with $\alpha_N > 0$. Let u and u_N be the exact solution of (1.15) and the Galerkin solution of (3.2), respectively. Then the error

$$\|u - u_N\|_{\mathscr{V}} \leqslant \frac{\|\mathcal{A}\|}{\alpha_N} \min_{w_N \in V} \|u - w_N\|_{\mathscr{V}}.$$

Proof. For all $w_N \in V$, applying (1.20) and (3.3), we have

$$\alpha_N \|u_N - w_N\|_{\mathscr{V}} \leqslant \sup_{v_N \in V} \frac{a[u_N - w_N, v_N]}{\|v_N\|_{\mathscr{V}}} = \sup_{v_N \in V} \frac{a[u - w_N, v_N]}{\|v_N\|_{\mathscr{V}}} \leqslant \|\mathcal{A}\| \, \|u - w_N\|_{\mathscr{V}}.$$

Then simply applying the triangular inequality gives the estimate.

$$\|u - u_N\|_{\mathscr{V}} \leqslant \frac{\|\mathcal{A}\| + \alpha_N}{\alpha_N} \min_{w_N \in V} \|u - w_N\|_{\mathscr{V}}.$$

Note that this constant in the upper bound is still not sharp. The desired constant in this Proposition was obtained by Xu and Zikatanov [123]. □

Remark 3.4 (Stability)**.** In view of Theorem 1.16, we can see that the Galerkin solution depends on the data continuously, i.e.,

$$\|u_N\|_{\mathscr{V}} \leqslant \frac{1}{\alpha} \|f\|_{\mathscr{V}'}.$$

□

Finite element ⋆

The finite element method (FEM) has a long history in practical use and is widely applied to lots of problems in physics and engineering. It has been proved to be very successful in many areas, like structural mechanics. After decades of extensive development, the subject of classical (conforming) finite element method has become a well-understood and successful area in scientific computation. The most attractive feature of the FEM is its ability to handle complex geometries, boundaries, and operators with relative ease.

Definition 3.5 (Finite element)**.** A triple $(K, \mathcal{P}, \mathcal{N})$ is called a finite element if and only if

(i) $K \subseteq \mathbb{R}^d$ be a bounded closed set with nonempty interior and piecewise smooth boundary;

(ii) $\mathcal{P}$ be a finite-dimensional space of functions on K;

(iii) $\mathcal{N} = \{\mathcal{N}_1, \ldots, \mathcal{N}_k\}$ be a basis of $\mathcal{P}'$.

We usually call K the *element domain*, $\mathcal{P}$ the space of *shape functions*, and $\mathcal{N}$ the set of *nodal variables*.

Definition 3.6 (Nodal basis)**.** Let $(K, \mathcal{P}, \mathcal{N})$ be a finite element. The basis $\{\phi_j\}_{j=1,\ldots,k}$ of $\mathcal{P}$ dual to $\mathcal{N}$, i.e., $\mathcal{N}_i(\phi_j) = \delta_{i,j}$ is called the nodal basis of $\mathcal{P}$.

Example 3.7 (1D Lagrange element)**.** Let $K = [0, 1]$, $\mathcal{P}$ be the set of linear polynomials, and $\mathcal{N} = \{\mathcal{N}_1, \mathcal{N}_2\}$ where $\mathcal{N}_1(v) = v(0)$ and $\mathcal{N}_2(v) = v(1)$. Then $(K, \mathcal{P}, \mathcal{N})$ is a finite element and it is the well-known $\mathcal{P}_1$-Lagrange finite element discussed in Chapter 1. The nodal basis functions are $\phi_1(x) = 1 - x$ and $\phi_2(x) = x$. □

Remark 3.8 (Set of nodal variables)**.** If $\mathcal{P}$ is a k-dimensional space and $\{\mathcal{N}_1, \ldots, \mathcal{N}_k\} \subset \mathcal{P}'$. Then condition (iii) in Definition 3.5 is equivalent to the unisolvence: For any $v \in \mathcal{P}$,

$$\mathcal{N}_i(v) = 0, \quad i = 1, \ldots, k \quad \Longrightarrow \quad v \equiv 0. \quad \square$$

Remark 3.9 (d-dimensional simplex)**.** Let $x^{(1)}, \ldots, x^{(d+1)}$ are $d+1$ points in $\mathbb{R}^d$. Suppose that these points do not lie in one hyper-plane. That is to say, the matrix

$$S = \begin{pmatrix} x_1^{(1)} & x_2^{(2)} & \cdots & x_1^{(d+1)} \\ x_2^{(1)} & x_2^{(2)} & \cdots & x_2^{(d+1)} \\ \vdots & \vdots & \vdots & \vdots \\ x_d^{(1)} & x_d^{(2)} & \cdots & x_d^{(d+1)} \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

is non-singular. The convex hull of the $d+1$ points

$$\tau := \{x = \sum_{i=1}^{d+1} \lambda_i x^{(i)} \,:\, 0 \leqslant \lambda_i \leqslant 1, i = 1 : d+1, \sum_{i=1}^{d+1} \lambda_i = 1\}$$

is called a geometric *d-simplex* generated (spanned) by the vertices $x^{(1)}, \ldots, x^{(d+1)}$. Given any point $x \in \mathbb{R}^d$, we have

$$x = \sum_{i=1}^{d+1} \lambda_i(x) x^{(i)}, \quad \text{with} \quad \sum_{i=1}^{d+1} \lambda_i(x) = 1.$$

Here the numbers $\lambda_1, \ldots, \lambda_{d+1}$ are called the *barycentric coordinates* of x with respect to the simplex τ. □

Now we describe the main steps of discretization using the $(K, \mathcal{P}, \mathcal{N})$-finite element:

Step 1. Domain partitioning: Choose K to be a simplex in $\mathbb{R}^d$. So we first partition the physical domain into simplexes. We discretize a polygonal domain Ω into small triangles or tetrahedrons τ. Let $h_\tau := |\tau|^{\frac{1}{d}}$ be the diameter of $\tau \in \mathcal{M}$ and $h(x)$ be the local meshsize, that is the piecewise constant function with $h|_\tau := h_\tau$ for all $\tau \in \mathcal{M}$. The collection $\mathcal{M}$ of elements is called a mesh or triangulation. We call $\mathcal{M}_h := \mathcal{M}$ *quasi-uniform* if there exists a constant h independent of τ such that

$$h \lesssim h_\tau \lesssim h, \quad \forall\, \tau \in \mathcal{M}.$$

We will only consider *conforming* meshes, i.e., the intersection of any two elements in $\mathcal{M}$ is either an edge ($d = 2$) / a face ($d = 3$), vertex, or empty (see Figure 3.1 for an example). We denote by $G(\mathcal{M})$ the set of all grid points (vertices) in the mesh $\mathcal{M}$. And $\mathring{G}(\mathcal{M}) \subseteq G(\mathcal{M})$ is the set of vertices except those on the Dirichlet boundary. Here we use the subscript h to describe the discrete nature and this does not imply the underlying meshes are quasi-uniform with meshsize h. In the future discussions, we will focus on uniform conforming meshes only.

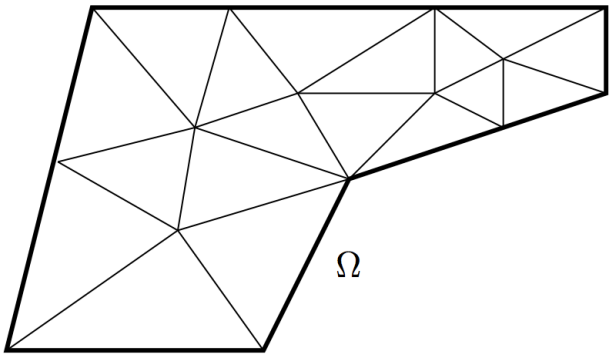


Figure 3.1: A polygonal domain Ω with conforming partition.

Remark 3.10 (Number of geometric entities)**.** We now briefly discuss the relations among the numbers of vertices, edges, faces, and elements in a triangular or tetrahedral partition. We will denote these numbers as $\#\mathsf{V}$, $\#\mathsf{E}$, $\#\mathsf{F}$, and $\#\tau$ respectively. As a convention, $\#\mathsf{F} = \#\tau$ in 2D. In 2D, we can consider the term of *half-edge*, which is defined as a pair of an edge and a face it borders. We can easily see that the number of half-edges is $2\,\#\mathsf{E}$ or $3\,\#\mathsf{F}$. Therefore, we have $2\,\#\mathsf{E} = 3\,\#\mathsf{F}$. Furthermore, according to the famous Euler–Poincaré formula, in any polyhedron, we have that $\#\mathsf{V} - \#\mathsf{E} + \#\mathsf{F} = 2$. Hence we can obtain that $\#\mathsf{F} \approx 2\,\#\mathsf{V}$ and $\#\mathsf{E} \approx 3\,\#\mathsf{V}$. In 3D, we have the following relations asymptotically $\#\mathsf{F} \approx 12\,\#\mathsf{V}$, $\#\mathsf{E} \approx 7\,\#\mathsf{V}$, and $\#\tau \approx 6\,\#\mathsf{V}$. □

Step 2. Finite-dimensional approximation: Let $V_h \subset \mathscr{V}$ be the space of continuous piecewise polynomials over a quasi-uniform conforming mesh $\mathcal{M}_h$, which satisfies appropriate conditions on the boundary $\Gamma := \overline{\Omega}\backslash\Omega$, i.e.,

$$V_h := \big\{v \in C(\overline{\Omega}) \,:\, v|_\tau \in \mathcal{P}_\tau, \text{ for all } \tau \in \mathcal{M}_h\big\} \bigcap \mathscr{V}. \tag{3.4}$$

We notice that there are many ways to approximate the continuous test function space. Different choices will then result in different numerical methods. In this section, we shall focus on the simplest case—linear finite element method on triangles or tetrahedrons, i.e., $v|_\tau$ is a linear polynomial on each $\tau \in \mathcal{M}_h$. The weak form of the finite element approximation reads: Find $u_h \in V_h$ such that

$$a[u_h, v_h] = \langle f, v_h\rangle, \quad \forall\ v_h \in V_h, \tag{3.5}$$

or, equivalently,

$$\mathcal{A}_h u_h = f_h. \tag{3.6}$$

Step 3. Assembling the finite-dimensional problem: Using the finite element definition $(K, \mathcal{P}, \mathcal{N})$, we can give a basis of the finite dimensional approximation space V_h. Suppose $\{\phi_i\}_{i=1}^N$ be a basis of the N-dimensional space V_h. Then (3.6) can be written as an linear algebraic equation

$$\hat{\mathcal{A}}_h \underline{u_h} = \vec{f}_h. \tag{3.7}$$

We are going to discuss this notation later in §3.2.

Properties of finite element methods

There are a few important properties of finite element space and method that will become crucial for our later analysis for multilevel iterative methods.

Proposition 3.11 (Interpolation error)**.** Let $\mathcal{M}_h$ be a uniform mesh and V_h be a $C^\alpha\,(\alpha \geqslant 0)$ finite element space on $\mathcal{M}_h$. The interpolant $\mathcal{J}_h : W_p^m(\Omega) \mapsto V_h$ satisfies

$$\big\|v - \mathcal{J}_h v\big\|_{W_p^k(\Omega)} \lesssim h^{m-k}\big\|v\big\|_{W_p^m(\Omega)}, \quad \forall\, v \in W_p^m(\Omega),\ 0 \leqslant k \leqslant \min\{m, \alpha+1\}.$$

Proposition 3.12 (Inverse estimate)**.** Let $\mathcal{M}_h$ be a uniform mesh and $\mathcal{P} \subseteq W_p^k(K) \bigcap W_q^m(K)$ and $0 \leqslant m \leqslant k$. If V_h is a finite element space for $(K, \mathcal{P}, \mathcal{N})$ on $\mathcal{M}_h$, then we have

$$\Big(\sum_{\tau \in \mathcal{M}_h} \|v\|^p_{W_p^k(\tau)} \Big)^{\frac{1}{p}} \lesssim h^{m-k+\min\{0, \frac{d}{p}-\frac{d}{q}\}} \Big(\sum_{\tau \in \mathcal{M}_h} \|v\|^q_{W_q^m(\tau)} \Big)^{\frac{1}{q}}, \quad \forall\, v \in V_h.$$

Using Proposition 3.12, we can easily see that, for any $v \in V_h$,

$$\begin{cases} \|v\|_{L^\infty(\Omega)} \lesssim h^{-\frac{d}{p}} \|v\|_{L^p(\Omega)}, & p \in [1, \infty); \\ \|v\|_{H^s(\Omega)} \lesssim h^{-s} \|v\|_{L^2(\Omega)}, & s \in [0, 1]; \\ \|v\|_{H^{1+\alpha}(\Omega)} \lesssim h^{-\alpha} \|v\|_{H^1(\Omega)}, & \alpha \in (0, \frac{1}{2}). \end{cases}$$

Moreover, there is a discrete Sobolev inequality at the bottom-line case (when $d = 2$) which is worthy for special attention.

Proposition 3.13 (Discrete Sobolev inequality [29])**.** The following inequality holds

$$\|v\|_{L^\infty(\Omega)} \lesssim C_d(h) \|v\|_{H^1(\Omega)}, \quad \forall\, v \in V_h,$$

where $C_1(h) \equiv 1$, $C_2(h) = |\log h|^{1/2}$, and $C_3(h) = h^{-\frac{1}{2}}$.

Proposition 3.14 (Weighted estimate for L^2 projection [29])**.** Define $\mathcal{Q}_h : L^2(\Omega) \mapsto V_h$ by, for any $v \in L^2(\Omega)$, it holds that

$$(\mathcal{Q}_h v, w) = (v, w), \quad \forall\, w \in V_h.$$

Then we have the following weighted L^2-estimate

$$\big\|v - \mathcal{Q}_h v\big\|_0 + h \big\|\mathcal{Q}_h v\big\|_1 \lesssim h \big\|v\big\|_1, \quad \forall\, v \in H_0^1(\Omega).$$

Remark 3.15 (Simultaneous estimate)**.** From the above weighted L^2-estimate, we can easily show the so-called simultaneous estimate

$$\inf_{w \in V_h} \Big(\big\|v - w\big\|_0 + h \big\|v - w\big\|_1 \Big) \lesssim h \|v\|_1, \quad \forall\, v \in H_0^1(\Omega).$$

□

Remark 3.16 (Spectral radius and condition number of $\mathcal{A}_h$)**.** Suppose that we have a uniform partition with meshsize h. It is clear, from the Poincaré inequality and the inverse inequality, that

$$\|v\|_0^2 \lesssim \|\nabla v\|_0^2 = (\mathcal{A}_h v, v) \leqslant \|v\|_1^2 \lesssim h^{-2} \|v\|_0^2, \quad \forall v \in V_h.$$

In fact, we have $\rho(\mathcal{A}_h) \cong h^{-2}$ and $\kappa(\mathcal{A}_h) \cong h^{-2}$. □

Error analysis ⋆

We now briefly introduce standard error estimates for the continuous linear finite element; see [46, 38] for details. For standard finite element approximation of elliptic equations, the most important property is the following Galerkin orthogonality property (see Remark 3.1)

$$a[u - u_h, v_h] = 0, \quad \forall\ v_h \in V.$$

Using the definition of the energy norm $\|\!|\cdot|\!\| := a[\cdot,\cdot]^{1/2}$, the Galerkin orthogonality (3.3), and the Cauchy-Schwarz inequality, we have

$$\|\!|u - u_h|\!\|^2 = a[u - u_h, u - u_h] = a[u - u_h, u - v_h] \leqslant \|\!|u - u_h|\!\|\, \|\!|u - v_h|\!\|, \quad \forall v_h \in V.$$

Hence, we obtain the *optimality* of the finite element approximation, i.e.,

$$\|\!|u - u_h|\!\| \leqslant \inf_{v_h \in V} \|\!|u - v_h|\!\|. \tag{3.8}$$

This means u_h is the best approximation of u in the subspace V. In general, it is not true for finite element approximations.

Theorem 3.17 (H^1-error estimate)**.** If $u \in H_0^m(\Omega)$ $(1 < m \leqslant 2)$, its $\mathcal{P}_1$-Lagrange finite element approximation $u_h \in V_h \subset \mathscr{V} = H_0^1(\Omega)$ satisfies

$$\|u - u_h\|_{1,\Omega} \lesssim h^{m-1}|u|_{m,\Omega}.$$

If $m = 2$, then we have $\|u - u_h\|_{1,\Omega} \lesssim h\|f\|_{0,\Omega}$.

Theorem 3.18 (L^2-error estimate)**.** If $u \in H_0^2(\Omega)$, its $\mathcal{P}_1$-Lagrange finite element approximation $u_h \in V_h \subset \mathscr{V} = H_0^1(\Omega)$ satisfies

$$\|u - u_h\|_{0,\Omega} \lesssim h|u - u_h|_{1,\Omega} \lesssim h^2|u|_{2,\Omega} \lesssim h^2\|f\|_{0,\Omega}.$$

Remark 3.19 (A posteriori error analysis)**.** A posteriori error estimation relies on the following error equation (or residual equation):

$$a[u - u_h, v] = a[u, v] - a[u_h, v] = \langle f, v\rangle - a[u_h, v] = \langle f - \mathcal{A}u_h, v\rangle, \quad \forall v \in \mathscr{V}.$$

Hence, by the Cauchy-Schwarz inequality, we obtain (see HW 3.1)

$$\|\!|f - \mathcal{A}u_h|\!\|_* \lesssim \|\!|u - u_h|\!\| \lesssim \|\!|f - \mathcal{A}u_h|\!\|_*. \tag{3.9}$$

Here $\|\!|\cdot|\!\|_*$ is the dual norm of $\|\!|\cdot|\!\|$. Notice that, on the right-hand side, we only have the data f and the discrete solution u_h. This upper bound does not depend on the unknown solution u. Of course, to make the upper bound useful in adaptive algorithms, we need it to be local and computable. □

3.2 Matrix representations

In the previous chapters, we have written the discrete problem simply as

$$A\vec{u} = \vec{f}.$$

We will see that, in some sense, it is an abuse-of-notation. Now we would like to clarify (especially for finite element methods) the relation between the general operator form $\mathcal{A}_h u_h = f_h$ and its often-used matrix form (3.7), i.e., $\hat{\mathcal{A}}_h \underline{u_h} = \vec{f_h}$. Sometimes we can drop the subscript h for simplicity.

Vector and matrix representations

Assume that $\{\phi_i\}_{i=1,\dots,N}$ is a basis of the finite-dimensional space V. Any function $v \in V$ can be represented as

$$v = \sum_{i=1}^{N} \underline{v}_i \phi_i$$

and the vector representation (coefficient vector) of v is defined as

$$\underline{v} := \begin{pmatrix} \underline{v}_1 \\ \underline{v}_2 \\ \vdots \\ \underline{v}_N \end{pmatrix} \in \mathbb{R}^N. \tag{3.10}$$

It is not hard to notice that there is another natural and easier-to-compute vector representation

$$\vec{v} := \begin{pmatrix} (v, \phi_1) \\ (v, \phi_2) \\ \vdots \\ (v, \phi_N) \end{pmatrix} \quad \text{and} \quad \vec{v} = M\underline{v}, \tag{3.11}$$

where $M \in \mathbb{R}^{N\times N}$ with $M_{i,j} := (\phi_j, \phi_i) = (\phi_i, \phi_j)$ is the *mass matrix*. $\underline{v}$ and $\vec{v}$ can be referred to as the *primal* and *dual* vector representations of v, respectively. Apparently, we have

$$(\underline{u}, \vec{v}) \equiv (\underline{u}, \vec{v})_{l^2} = \underline{u}^T M \underline{v} = (u, v)_V.$$

Suppose W is another finite-dimensional linear space with a basis $\{\psi_i\}_{i=1,\dots,N'}$. In general, W could be of different dimension than V, namely, $N' \neq N$. For any linear operator $\mathcal{A} : V \mapsto W$, we give a matrix representation (the so-called primal representation), $\underline{\mathcal{A}} \in \mathbb{R}^{N'\times N}$, such that it satisfies that $\sum_{i=1}^{N'} (\underline{\mathcal{A}})_{i,j} \psi_i = \mathcal{A}\phi_j \ \ (j = 1, \dots, N)$, i.e.,

$$(\psi_1, \dots, \psi_{N'})\underline{\mathcal{A}} = \mathcal{A}(\phi_1, \dots, \phi_N). \tag{3.12}$$

On the other hand, the dual representation (the *stiffness matrix*) corresponding to $\mathcal{A}$ is denoted by $\hat{\mathcal{A}} \in \mathbb{R}^{N\times N}$ with entries $\left(\hat{\mathcal{A}}\right)_{i,j} := (\mathcal{A}\phi_j, \phi_i)$.

It is not difficult to check the statements in the following identities; see HW 3.2.

Lemma 3.20 (Matrix representations)**.** If $\mathcal{A}, \mathcal{B} : V \mapsto V$ and $v, u \in V$, we have the following results:

1. $\underline{\mathcal{A}\mathcal{B}} = \underline{\mathcal{A}}\,\underline{\mathcal{B}}$;
2. $\underline{\mathcal{A}v} = \underline{\mathcal{A}}\,\underline{v}$;
3. $\sigma(\mathcal{A}) = \sigma(\underline{\mathcal{A}})$, $\kappa(\mathcal{A}) = \kappa(\underline{\mathcal{A}})$;
4. $\vec{v} = M\,\underline{v}$, $\overrightarrow{\mathcal{A}v} = \hat{\mathcal{A}}\,\underline{v}$;
5. $\hat{\mathcal{A}} = M\,\underline{\mathcal{A}}$;
6. $(u, v) = (M\underline{u}, \underline{v})$.

Example 3.21 (Identity operator)**.** Let $\mathcal{I} : V \mapsto V$ be the identity operator. Its stiffness and mass matrices are equal to each other, i.e., $\hat{\mathcal{I}} = M$. Hence $\underline{\mathcal{I}} = M^{-1}\hat{\mathcal{I}} = I$. Note that this relation is independent of the choice of basis functions. As a consequence, we have

$$I = \underline{\mathcal{I}} = \underline{\mathcal{A}\mathcal{A}^{-1}} = \underline{\mathcal{A}}\,\underline{\mathcal{A}^{-1}},$$

which gives the equality $\underline{\mathcal{A}}^{-1} = \underline{\mathcal{A}^{-1}}$. □

Example 3.22 (Finite difference matrices)**.** For the finite difference methods, we can simply let $\mathcal{A} : \mathbb{R}^N \mapsto \mathbb{R}^N$ be a matrix and the canonical basis $\phi_i = \vec{e}_i := (0, \ldots, 1, \ldots, 0)^T \in \mathbb{R}^N$, then we have $\hat{\mathcal{A}} = \underline{\mathcal{A}}$. Generally speaking, if $\mathcal{A} : V \mapsto V$ and $\{\phi_i\}_{i=1}^N$ is an orthonormal basis of V, then we have $M = I$ and $\hat{\mathcal{A}} = \underline{\mathcal{A}}$. □

Finite element matrices

We now use a few simple examples to demonstrate how to apply these notations. Suppose that $V = V_h$ is the piecewise linear finite element space and $\{\phi_i\}_{i=1,\ldots,N}$ are the basis functions. Let A be the resulting coefficient matrix of (3.2) with $(A)_{i,j} = a_{i,j} := a[\phi_i, \phi_j]$. By definition, $A = \hat{\mathcal{A}} \in \mathbb{R}^{N\times N}$ is the stiffness matrix corresponding to $\mathcal{A}$. Since we are going to focus on the finite element discretization from now on, we will not distinguish A and $\hat{\mathcal{A}}$, when no ambiguity arises.

Let $\underline{u} = \left(u_i\right)_{i=1}^N \in \mathbb{R}^N$ be the vector of coefficients of u_h, namely $\underline{u_h}$. Let $\vec{f} = \left(f_i\right)_{i=1}^N := \left\{\langle f, \phi_i\rangle\right\}_{i=1}^N$. Then $\underline{u}$ satisfies the linear system of equations:

$$\hat{\mathcal{A}}\underline{u} = \vec{f} \qquad \text{or} \qquad A\underline{u} = \vec{f}.$$

Upon solving this finite-dimensional linear system, we are able to obtain a discrete approximation

$$u_h = \sum_{i=1}^{N} \underline{u}_i \phi_i.$$

The main algebraic properties for the stiffness matrix includes: A is *sparse* with $O(N)$ nonzeros, symmetric positive definite (for Dirichlet or mixed boundary condition problems) or symmetric positive semi-definite (for Neumann boundary condition problems). We now summarize this brief introduction of finite element matrices with a few comments. The following results are valid for a large class of finite elements for second-order elliptic boundary value problems in general domains.

Remark 3.23 (Spectrum of mass matrix)**.** Suppose that we have a uniform partition with meshsize h. An often-used matrix is the mass matrix $M \in \mathbb{R}^{N\times N}$, in which $M_{i,j} = (\phi_i, \phi_j)$. In fact, we know that

$$(M\underline{v}, \underline{v}) = \sum_{i,j} \underline{v}_i \, \underline{v}_j \, (\phi_i, \phi_j) = (v, v) = \int_\Omega v^2(x)\, dx \cong h^d \sum_i \underline{v}_i^2 \cong h^d (\underline{v}, \underline{v}). \tag{3.13}$$

It is consistent with the well-known facts that the mass matrix is also SPD and well-conditioned, i.e.,

$$h^d \|\xi\|_0^2 \lesssim \xi^T M\, \xi \lesssim h^d \|\xi\|_0^2, \quad \forall\, \xi \in \mathbb{R}^N.$$

□

Remark 3.24 (Spectrum of stiffness matrix)**.** Suppose that we have a uniform partition with meshsize h. It is also well-known that the stiffness matrix A is SPD and, from Remark 3.16,

$$h^d \|\xi\|_0^2 \lesssim \xi^T A\, \xi \lesssim h^{d-2} \|\xi\|_0^2, \quad \forall\, \xi \in \mathbb{R}^N.$$

Hence the spectral radius $\rho(A) \cong h^{d-2}$ and the condition number $\kappa(A) \cong h^{-2}$. And it has been observed that the CG method becomes slower when h decreases. □

Matrix and operator forms of simple iterative methods

Now we consider the solution of the standard finite element (say the $\mathcal{P}_1$-Lagrange element) for the Poisson's equation, i.e., $\hat{\mathcal{A}}\underline{u} = \vec{f}$. The simplest iterative solver for this finite element equation is probably the well-known Richardson method:

$$\underline{u}^{\text{new}} = \underline{u}^{\text{old}} + \omega\Big(\vec{f} - \hat{\mathcal{A}}\, \underline{u}^{\text{old}}\Big). \tag{3.14}$$

It is equivalent to

$$\underline{u}^{\text{new}} = \underline{u}^{\text{old}} + \omega\Big(M\underline{f} - M\underline{\mathcal{A}}\, \underline{u}^{\text{old}}\Big) = \underline{u}^{\text{old}} + \omega M\Big(\underline{f} - \underline{\mathcal{A}}\, \underline{u}^{\text{old}}\Big).$$

That is to say, the Richardson method, can be written in the operator form as

$$u^{\text{new}} = u^{\text{old}} + \mathcal{B}_\omega\Big(f - \mathcal{A}\, u^{\text{old}}\Big)$$

with an iterator $\mathcal{B}_\omega$, whose matrix representation is $\underline{\mathcal{B}_\omega} = \omega M$. Therefore, it is easy to check (HW 3.3) that the operator form of the Richardson method is

$$\mathcal{B}_\omega v := \omega \sum_{i=1}^{N} (v, \phi_i)\phi_i, \quad \forall v \in V \quad \Longleftrightarrow \quad \underline{\mathcal{B}_\omega} = \omega M. \tag{3.15}$$

In general, a smoother or local relaxation is just a linear stationary iterative method

$$u^{\text{new}} = u^{\text{old}} + \mathcal{S}(f - \mathcal{A}u^{\text{old}})$$

and its matrix representation is

$$\underline{u}^{\text{new}} = \underline{u}^{\text{old}} + \underline{\mathcal{S}}(M^{-1}\vec{f} - M^{-1}\hat{\mathcal{A}}\underline{u}^{\text{old}}) = \underline{u}^{\text{old}} + \underline{\mathcal{S}}M^{-1}(\vec{f} - \hat{\mathcal{A}}\underline{u}^{\text{old}}). \tag{3.16}$$

The above equality indicates that, we shall define a smoother in the matrix form as

$$S := \underline{\mathcal{S}}M^{-1}, \quad \text{i.e.,} \quad \underline{\mathcal{S}} = SM. \tag{3.17}$$

Example 3.25 (Matrix form of the Richardson iteration)**.** If we consider the above Richardson method (3.15) as an example, i.e. $\mathcal{S}_{\text{R}} := \mathcal{B}_\omega$, then

$$S_{\text{R}} = \underline{\mathcal{S}_{\text{R}}}M^{-1} = \underline{\mathcal{B}_\omega}M^{-1} = \omega I.$$

This coincides with the algebraic form of the Richardson method (3.14). □

Now we discuss another important concept for our analysis, the matrix form of the symmetrization. Let $w := \mathcal{S}^T u$. Then we have

$$\vec{w} = \Big((\mathcal{S}^T u, \phi_i)\Big)_{i=1}^N = \Big(\sum_j \underline{u}_j(\mathcal{S}^T\phi_j, \phi_i)\Big)_{i=1}^N = \Big(\sum_j \underline{u}_j(\phi_j, \mathcal{S}\phi_i)\Big)_{i=1}^N = \big(\hat{\mathcal{S}}\big)^T\underline{u}.$$

This immediately gives

$$\underline{\mathcal{S}^T}\,\underline{u} = \underline{\mathcal{S}^T u} = \underline{w} = M^{-1}\vec{w} = M^{-1}\big(\hat{\mathcal{S}}\big)^T\underline{u} = M^{-1}\big(M\underline{\mathcal{S}}\big)^T\underline{u}.$$

In turn, it shows

$$\underline{\mathcal{S}^T} = M^{-1}\big(M\underline{\mathcal{S}}\big)^T = M^{-1}\underline{\mathcal{S}}^T M = S^T M. \tag{3.18}$$

By definition of the primal matrix representation of an operator, we have

$$\mathcal{S}(\phi_1, \ldots, \phi_N) = (\phi_1, \ldots, \phi_N)\underline{\mathcal{S}} \qquad \text{and} \qquad \mathcal{S}^{-1}(\phi_1, \ldots, \phi_N) = (\phi_1, \ldots, \phi_N)\underline{\mathcal{S}^{-1}}.$$

Using Example 3.21, it is easy to see that

$$\underline{\mathcal{S}^{-1}} = \big(\underline{\mathcal{S}}\big)^{-1} = \big(SM\big)^{-1} = M^{-1}S^{-1}. \tag{3.19}$$

Using the definition of symmetrized operator (2.13) and (3.17)–(3.19), we can define the matrix form of the symmetrization

$$\begin{aligned} \overline{S} \;:=\; \overline{\mathcal{S}}M^{-1} \;&=\; S^T M\big(M^{-1}S^{-T} + M^{-1}S^{-1} - M^{-1}\hat{\mathcal{A}}\big)SMM^{-1} \\ &=\; S^T\big(S^{-T} + S^{-1} - A\big)S, \end{aligned} \tag{3.20}$$

which is formally consistent with the definition of symmetrization (2.13).

3.3 Smoothers and smoothing effect

The methods discussed by far, for example the damped Jacobi and Gauss–Seidel methods, are mostly *local relaxation* methods. The name "local relaxation" comes from the fact that these methods just correct the residual vector locally, one variable at a time; see Example 2.20. Although these methods are not very efficient as a solver by themselves, they are key ingredients of modern multilevel iterative methods. These methods can be applied to reduce high-frequency error components; see §1.3. Other methods like the SOR method and incomplete factorizations have similar effects too. In this section, we analyze their smoothing effect using different approaches.

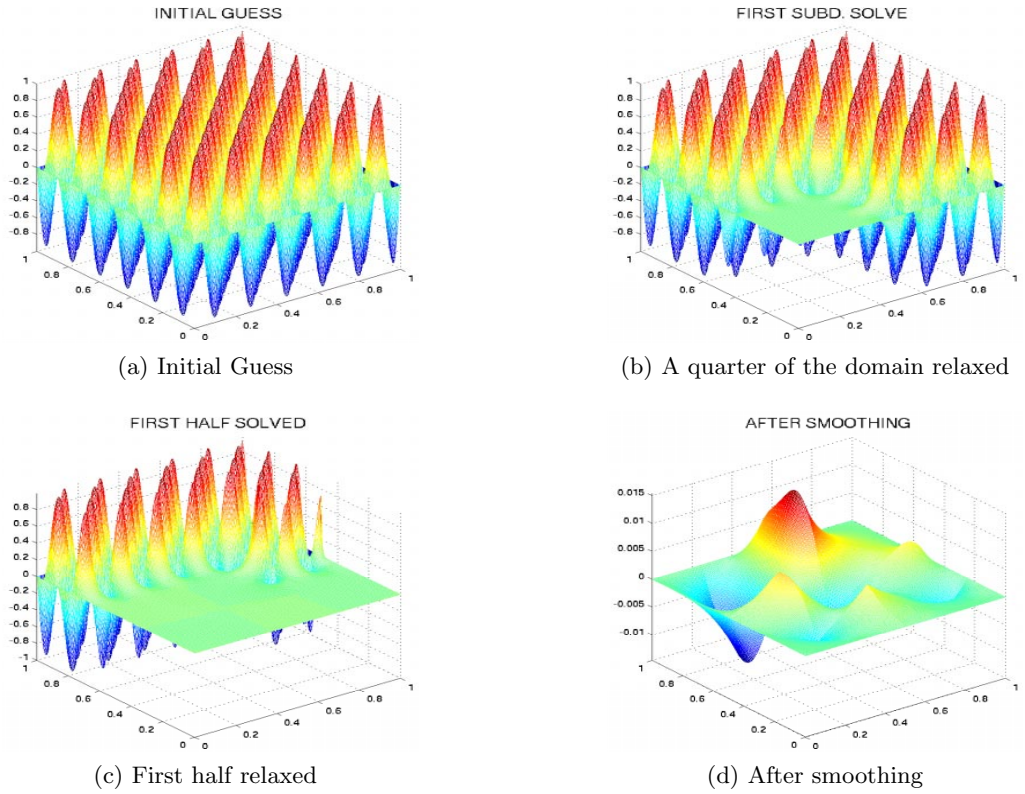
A numerical example

The damped Jacobi and Gauss–Seidel methods are often called *local relaxations* and such relaxation procedures are effective to the error components that are local in nature. Therefore, it is not surprising that both the damped Jacobi and the Gauss–Seidel methods can damp out non-smooth components more easily. These methods are inefficient for relatively smoother components in the error since they are more globally related.

We have observed that the basic stationary linear iterative schemes converge rather fast in the very beginning but then slows down after a few step; see Figure 1.4 for the convergence behavior of the damped Jacobi method. Moreover, these methods not only converges fast in the first few steps, but also smooth out the error function very quickly. In other words, the error becomes a much smoother function after a few iterations. This property of the iterative scheme is naturally called smoothing property and any iterative scheme possessing such smoothing property is called a *smoother*.

Figure 3.2 is a pictorial example for applying multiplicative overlapping domain decomposition method with four subdomains. We can see that, after one iteration, the method smoothes

out the high frequency part and leaves the lower frequency part behind. In fact, basic linear



(a) Initial Guess

(b) A quarter of the domain relaxed

(c) First half relaxed

(d) After smoothing

Figure 3.2: Iterative method in the viewpoint of subdomain relaxation.

relaxation schemes, such as the Richardson, Jacobi, and Gauss–Seidel iterations, are local and can only capture high frequency (local) part of the error, but do not work well on low frequency (global) part.

Local Fourier analysis ⋆

Local Fourier analysis (LFA) or local mode analysis [30, 105, 32] is a very powerful technique to understand and predict the convergence speed of geometric multigrid (GMG) methods. It is well-known that, for model problems on rectangular domains with periodic boundary conditions, LFA can yield the exact convergence rate of GMG; see detailed discussions in, for example, [33, 110, 117]. More recently, LFA has been shown to be applicable to more realistic situations like the Dirichlet boundary condition case—It was proved that, if the problem is compatible to a periodic boundary condition problem, LFA yields rigorous convergence rate of the multigrid schemes [101].

LFA has been developed for multigrid algorithms for very general problems, including problems with nonconstant or nonlinear coefficients. The LFA technique can be applied to different

discretization methods (like finite difference methods, finite volume methods, etc) as long as the resulting discrete problems can be represented in a stencil form. This is restrictive for its application on finite element methods due to their grids are typically not structured. Here, for simplicity, we analyze the simple smoothers using LFA, just to give the readers some flavor on this powerful tool. For more details, the readers are referred to the practical guide on LFA by Wienands and Joppich [117].

In order to analyze the local behavior of iterative methods, we consider the 2D Poisson's equation with homogenous Dirichlet boundary condition on the unit square discretized with a uniform triangulation; see §1.2. We begin with the damped (weighted) Jacobi method as an example. Using the local Fourier analysis, we have the following observation:

1. The standard FD stencil can be written as

$$4u_{i,j} - \big(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\big) = h^2 f_{i,j}, \quad i,j = 1,\ldots,n$$

and the damped Jacobi (or Richardson) method for the above equation reads

$$u_{i,j}^{\text{new}} = (1-\omega)u_{i,j}^{\text{old}} + \frac{\omega}{4}\big(u_{i-1,j}^{\text{old}} + u_{i+1,j}^{\text{old}} + u_{i,j-1}^{\text{old}} + u_{i,j+1}^{\text{old}}\big) + \frac{\omega}{4}h^2 f_{i,j}, \quad i,j = 1,\ldots,n.$$

2. Define the discrete error function $e_{i,j}^{\text{new}} := u_{i,j} - u_{i,j}^{\text{new}}$ and $e_{i,j}^{\text{old}} := u_{i,j} - u_{i,j}^{\text{old}}$, for $i,j = 1,\ldots,n$. It is clear that the error function satisfies the local error equation

$$e_{i,j}^{\text{new}} = (1-\omega)e_{i,j}^{\text{old}} + \frac{\omega}{4}\big(e_{i-1,j}^{\text{old}} + e_{i+1,j}^{\text{old}} + e_{i,j-1}^{\text{old}} + e_{i,j+1}^{\text{old}}\big), \quad i,j = 1,\ldots,n.$$

3. Apply the discrete Fourier transformation:

$$e_{i,j} = \sum_{\theta\in\Theta_n} \alpha_\theta e^{\sqrt{-1}(i\theta_1 + j\theta_2)}$$

and

$$\Theta_n := \Big\{(\theta_1,\theta_2) \,:\, \theta_1 = \frac{2k\pi}{n},\ \theta_2 = \frac{2l\pi}{n},\ \ k,l \in [-m_1, m_2]\Big\},$$

where $m_1 = n/2-1, m_2 = n/2$, if n is even and $m_1 = m_2 = (n-1)/2$, if n is odd. Plugging the discrete Fourier transforms of $e_{i,j}^{\text{new}}$ and $e_{i,j}^{\text{old}}$ to the above error equation, we get the amplification factor of the local mode $e^{\sqrt{-1}(i\theta_1+j\theta_2)}$

$$\lambda(\theta) := \frac{\alpha_\theta^{\text{new}}}{\alpha_\theta^{\text{old}}} = 1 - \omega\Big(1 - \frac{\cos(\theta_1)+\cos(\theta_2)}{2}\Big) \leqslant 1.$$

Furthermore, $\lambda(\theta) \to 1$ when $|\theta| \to 0$ (low-frequency components).

4. Asymptoticly, $m_1 \approx m_2 \approx \frac{n}{2}$. So we can define a *smoothing factor* (i.e. maximal amplification factor corresponding to high-frequency local modes) by

$$\bar{\rho} := \sup_{\theta} \Big\{ \big|\lambda(\theta)\big| \,:\, \frac{\pi}{2} \leqslant |\theta_k| \leqslant \pi,\ k = 1, 2 \Big\}.$$

   By plugging in the end points, we get the the smoothing factor for the damped Jacobi method is

$$\bar{\rho}_{\text{Jacobi}} := \max \Big\{ \Big|1 - 2\omega\Big|, \Big|1 - \frac{1}{2}\omega\Big|, \Big|1 - \frac{3}{2}\omega\Big| \Big\}.$$

Remark 3.26 (Optimal damping factor for smoothing)**.** We notice that, if $\omega = 1$ (the Jacobi method), then $\bar{\rho}_{\text{Jacobi}} = 1$. This confirms the result we obtained in the previous subsection. Apparently, the "best" weight that minimizes the smoothing factor is $\omega = 4/5$, which leads to $\bar{\rho}_{\text{Jacobi}} = 3/5$. □

Remark 3.27 (What is high-frequency error)**.** In the above analysis, we have specify the high-frequency part to be corresponding to $\frac{\pi}{2} \leqslant |\theta_k| \leqslant \pi$. As pointed in Remark 1.24, high-frequencies can be approximated accurately by local behavior. On the contrary, the low-frequencies are those which can be represented well on the coarser grids. Hence this definition is not universal. We have to adjust it to fit the coarsening algorithm under consideration. For example, semi-coarsening or red-black coarsening will lead to different definitions of high-frequency; see, for example, [110]. Later on, we will also discuss how to define this concept from an algebraic point of view. □

It is natural for us to imagine that the G-S method should be better than the Jacobi method in terms of smoothing property. Using the same steps as above, we have the following LFA analysis:

1. The G-S method in lexicographical order reads

$$u_{i,j}^{\text{new}} = \frac{1}{4}\big(u_{i-1,j}^{\text{new}} + u_{i+1,j}^{\text{old}} + u_{i,j-1}^{\text{new}} + u_{i,j+1}^{\text{old}}\big) + \frac{1}{4}h^2 f_{i,j}, \quad i, j = 1, \ldots, n.$$

2. The discrete error function satisfies

$$e_{i,j}^{\text{new}} = \frac{1}{4}\big(e_{i-1,j}^{\text{new}} + e_{i+1,j}^{\text{old}} + e_{i,j-1}^{\text{new}} + e_{i,j+1}^{\text{old}}\big), \quad i, j = 1, \ldots, n.$$

3. Apply the discrete Fourier transform and compute the amplification factor

$$\lambda(\theta) := \frac{\alpha_\theta^{\text{new}}}{\alpha_\theta^{\text{old}}} = \frac{e^{\sqrt{-1}\theta_1} + e^{\sqrt{-1}\theta_2}}{4 - e^{-\sqrt{-1}\theta_1} - e^{-\sqrt{-1}\theta_2}}.$$

4. One can show the smoothing factor for the G-S method is

$$\bar{\rho}_{\text{GS}} := \Big|\lambda\Big(\frac{\pi}{2}, \arccos(4/5)\Big)\Big| = \frac{1}{2}.$$

Remark 3.28 (Anisotropic problems and smoothing effect)**.** Notice that the above analysis only works for uniform partition and isotropic coefficients. When we solve an anisotropic problem, it is important to note that the G-S method (and other point relaxation methods) yields not as good smoothing factor as the isotropic case. In fact, the smoothing factor goes to 1 when ratio between small and large coefficients goes to 0; see Chapter 6 for details. □

Remark 3.29 (Ordering and G-S smoother)**.** For the G-S method, ordering is important. When using the red-black ordering instead of the lexicographical ordering above, one can show the smoothing factor $\bar{\rho}_{\text{RBGS}}$ is $\frac{1}{4}$ [105, 110]. This means the smoothing effect of the red-black ordering for G-S is better. □

Smoothing effect

Considering the Richardson method (3.15), then we have $\mathcal{B}_\omega v = \omega \sum_{i=1}^N (v, \phi_i)\phi_i$. This implies

$$(\mathcal{B}_\omega v, v) = \omega \sum_{i=1}^N (v, \phi_i)^2 = \omega \sum_{i=1}^N (M\underline{v})_i^2 = \omega(M\underline{v}, M\underline{v}) = \omega(M^2\underline{v}, \underline{v}).$$

Since M is SPD, we get

$$(M^2\underline{v}, \underline{v}) = (MM^{\frac{1}{2}}\underline{v},\, M^{\frac{1}{2}}\underline{v}) \cong h^d(M^{\frac{1}{2}}\underline{v},\, M^{\frac{1}{2}}\underline{v}) = h^d(M\underline{v},\, \underline{v}).$$

The estimate (3.13) implies that

$$(\mathcal{B}_\omega v, v) \cong \omega h^d(v, v). \tag{3.21}$$

Now we choose the weight of the Richardson iteration to be $\omega = h^{2-d}$, i.e.,

$$\mathcal{S}_{\text{R}} v := \mathcal{B}_\omega v = h^{2-d} \sum_{i=1}^N (v, \phi_i)\phi_i, \quad \forall v \in V. \tag{3.22}$$

In view of (3.21) and using the fact that the spectral radius of the FE operator is $\rho(\mathcal{A}) \cong h^{-2}$ (see Remark 3.16), we find

$$(\mathcal{S}_{\text{R}} v, v) \cong h^2(v, v) \cong \frac{1}{\rho(\mathcal{A})}(v, v).$$

Roughly speaking, $\mathcal{S}_{\text{R}}$ behaves like $\mathcal{A}^{-1}$ in the high-frequency regime. This is a natural property we will ask for from a smoother later on:

$$(\mathcal{S} v, v) \cong \frac{1}{\rho(\mathcal{A})}(v, v). \tag{3.23}$$

In fact, such conditions are only needed in the range of $\mathcal{S}$.

Apparently, the damped Jacobi method also satisfies this condition. In fact, using the standard scaling argument on each element, we can see that

$$h^{d-2}(\xi,\xi) \lesssim (D\xi,\xi) \lesssim h^{d-2}(\xi,\xi).$$

Hence, using (3.17), we have the Jacobi smoother

$$(\mathcal{S}_{\mathrm{J}} v, v) = (M\underline{\mathcal{S}_{\mathrm{J}}}\underline{v}, \underline{v}) = (MD^{-1}M\underline{v}, \underline{v}) \cong h^{d+2}(\underline{v},\underline{v}) \cong h^2(v,v) \cong \frac{1}{\rho(\mathcal{A})}(v,v).$$

Next, we shall show a more interesting fact that the G-S method behaves in a similar way as in the Jacobi method.

Lemma 3.30 (Smoothing property of G-S in matrix form)**.** Let $\hat{\mathcal{A}}$ be the stiffness matrix and $\hat{\mathcal{A}} = A = D + L + U$. Then the G-S method satisfies

$$\big\|(D+L)\xi\big\|_0 \cong \big\|D\xi\big\|_0 \cong h^{d-2}\|\xi\|_0, \quad \forall \xi \in \mathbb{R}^N.$$

Proof. Locality of the nodal basis functions leads to sparse matrix L; in turn, this gives

$$\big\|(D+L)\xi\big\|_0 \lesssim \big\|D\xi\big\|_0 \lesssim h^{d-2}\|\xi\|_0.$$

The other direction follows from

$$h^{d-2}\|\xi\|_0^2 \lesssim \big(D\xi,\xi\big) \leqslant \big((D+A)\xi,\xi\big) = 2\big((D+L)\xi,\xi\big) \lesssim \|(D+L)\xi\|_0\,\|\xi\|_0.$$

We then get the desired estimates with simple manipulations. □

Similar results for $\mathcal{S}_{\mathrm{GS}}$ follows directly as in the Jacobi method. Now we consider the symmetrized G-S method.

Lemma 3.31 (Smoothing property of SGS)**.** Let $\mathcal{S} : V \mapsto V$ be the symmetrized G-S (SGS) iterator. Then we have

$$\big(\mathcal{S}v, v\big) \cong h^2(v,v) \cong \frac{1}{\rho(\mathcal{A})}(v,v). \tag{3.24}$$

Proof. The matrix form of SGS can be written as

$$\underline{\mathcal{S}} = SM = (D+U)^{-1}D(D+L)^{-1}M.$$

Let $\underline{v}$ be the primal vector representation of $v \in V$. Then we have

$$(\mathcal{S}v, v) = (M\underline{\mathcal{S}v}, \underline{v}) = (M\underline{\mathcal{S}}\,\underline{v}, \underline{v}) = \big\|D^{\frac{1}{2}}(D+L)^{-1}M\underline{v}\big\|_0^2.$$

Hence to show the lemma is equivalent to prove that

$$\big\|D^{\frac{1}{2}}(D+L)^{-1}M\underline{v}\big\|_0^2 \cong h^2(M\underline{v},\underline{v}).$$

By changing of variable $\xi := (D+L)^{-1}M\underline{v} \in \mathbb{R}^N$ and the fact $M \cong h^d$, we can obtain the above equality using

$$h^{d-2}(D\xi,\xi) \cong h^{2(d-2)}\|\xi\|_0^2 \cong \big\|(D+L)\xi\big\|_0^2 = (M\underline{v}, M\underline{v}), \quad \forall\, \xi \in \mathbb{R}^N,$$

which is true due to Lemma 3.30. □

Smoother as preconditioner

From the property (3.24) which is satisfied by all the aforementioned popular smoothers, we can easily see that

$$\rho_{\mathcal{A}}^{-1}(v,v) \lesssim (\mathcal{S}v,v) \lesssim \rho_{\mathcal{A}}^{-1}(v,v), \tag{3.25}$$

where $\rho_{\mathcal{A}} := \rho(\mathcal{A})$. In this note, we call it the smoothing property, which basically means the smoother $\mathcal{S}$ behaves like $\mathcal{A}^{-1}$ on the high frequency regime. Other forms of conditions or assumptions for smoothers have been discussed in the literature, interested readers are referred to the paper by Bramble and Pasciak [24] (general smoothers defined as additive and multiplicative Schwarz methods) and the references therein.

From this property, we have a lower bound for the minimal eigenvalue $\rho_{\mathcal{A}}^{-1} \lesssim \lambda_{\min}(\mathcal{S})$. If the smoother is also symmetric, then the above property indicates that the smoother is SPD. That is to say, the symmetrized version $\overline{\mathcal{S}}$ is apparently SPD and can be used as a preconditioner as well. In view of Remark 3.16 (i.e., $\|v\|_0^2 \lesssim (v,v)_{\mathcal{A}} \leqslant \rho_{\mathcal{A}}\|v\|_0^2$), with simple manipulations, we can derive

$$\rho_{\mathcal{A}}^{-1}(v,v)_{\mathcal{A}} \lesssim \rho_{\mathcal{A}}^{-1}(\mathcal{A}v,\mathcal{A}v) \lesssim (\mathcal{S}\mathcal{A}v,v)_{\mathcal{A}} \lesssim \rho_{\mathcal{A}}^{-1}(\mathcal{A}v,\mathcal{A}v) \leqslant (v,v)_{\mathcal{A}}. \tag{3.26}$$

Due to Lemmas 2.29 and 2.30, (3.26) indicates that $\kappa(\mathcal{S}\mathcal{A}) \lesssim \rho(\mathcal{A}) \cong \kappa(\mathcal{A})$, which means these smoothers, when applied as preconditioners, might not improve the condition number. Hence, to make a reasonable preconditioner requires a lot more than to be a good smoother, which will be the main topic for the rest of this note.

3.4 Twogrid methods

From the analysis in §3.3, we have found that local relaxation methods (smoothers) can damp the oscillatory components of the error rather quickly. Motivated by the two-level DD method in §2.4, we can introduce coarser levels to take care of the smooth components which cannot be treated efficiently by local relaxation methods. A natural idea is then, after a few smoothing steps, to approximate the resulting problem on a coarser grid and continue the iteration with a "coarse version" of the problem. This way, we can resolve the high frequency part of the error

with relaxation schemes and leave the low frequency part to the coarse levels. Before we discuss multilevel methods, we first investigate a much simpler case—the twogrid method.

First of all, we present a simple observation which heuristically explains why the solution on a coarse grid can give a good approximation for smooth error. In fact, smooth functions can be represented on the coarse grid rather accurately. This is the last missing piece of ideas that motivate multilevel iterative methods. We only give a sketch of the proof here and leave the complete proof to the readers (see HW 3.5).

Remark 3.32 (Low frequency error)**.** Let u_h and u_H be the finite element solutions on V_h and $V_H \subset V_h$, respectively. Then we immediately have

$$a[u_h - u_H, v_H] = 0, \quad \forall\, v_H \in V_H.$$

Using the Aubin-Nitsche's argument, we consider a boundary value problem

$$\begin{cases} -\Delta w &= u_h - u_H & \text{in } \Omega, \\ w &= 0 & \text{on } \partial\Omega. \end{cases}$$

Assume that we have full elliptic regularity. Then $\|w\|_2 \leqslant C\|u_h - u_H\|_0$ is bounded. For any $w_H \in V_H$, we get

$$\|u_h - u_H\|_0^2 = a[w, u_h - u_H] = a[w - w_H, u_h - u_H] \leqslant |\!|\!| w - w_H |\!|\!| \, |\!|\!| u_h - u_H |\!|\!| \lesssim H|w|_2 \, |\!|\!| u_h - u_H |\!|\!| .$$

Hence the following inequality holds

$$\|u_h - u_H\|_0 \lesssim H \, |\!|\!| u_h - u_H |\!|\!| \lesssim H \, |\!|\!| u_h |\!|\!| . \tag{3.27}$$

That is to say, if u_h is relatively smooth (small first derivatives), then u_h can be well approximated by u_H. Compare with Remarks 1.24 and 3.27. □

General twogrid methods

Let V_h be fine grid finite element space and V_H be the coarse gird space (usually it is a subspace of V_h.) The twogrid method for equation (3.2) can be described as

Algorithm 3.1 (General twogrid method)**.** Given an initial guess $u^{(0)} \in V_h$.

(i) **Pre-smoothing**: Apply a few relaxation steps to smooth $u^{(0)}$ in the fine space to obtain a new approximation $u^{(1)} \in V_h$;

(ii) **Coarse-grid Correction**: Find $e_H \in V_H$ by solving (exactly or approximately) the error equation

$$(\mathcal{A}e_H, v_H) = (f - \mathcal{A}u^{(1)}, v_H), \quad \forall\, v_H \in V_H$$

in the coarse space, and then set $u^{(2)} = u^{(1)} + e_H$;

(iii) **Post-smoothing**: Apply a few more relaxation steps to smooth $u^{(2)}$ in the fine space to obtain $u^{(3)} \in V_h$.

A more concrete algorithm based on the above abstract algorithm can be introduced. Let V be the fine space associated with meshsize h and $V_c \subset V$ be the coarse space associated with meshsize H. Let $\mathcal{I}_c : V_c \mapsto V$ be the natural embedding (injection), i.e., $\mathcal{I}_c v_c = v_c,\ \forall v_c \in V_c$.

Remark 3.33 (Embedding and projection)**.** By the definition of embedding $\mathcal{I}_c : V_c \mapsto V$ and the fact

$$(\mathcal{I}_c^T v, w_c) = (v, \mathcal{I}_c w_c) = (v, w_c), \quad \forall v \in V, w_c \in V_c,$$

it is easy to see that $\mathcal{I}_c^T = \mathcal{Q}_c$ is the $(\cdot,\cdot)$-projection from V to V_c. And the coarse-level operator can be defined by the Galerkin relation

$$\mathcal{A}_c = \mathcal{I}_c^T \mathcal{A} \mathcal{I}_c = \mathcal{Q}_c \mathcal{A} \mathcal{I}_c.$$

□

Suppose that $\mathcal{S}$ is a smoother and $\mathcal{B}_c$ is a solver or iterator for the coarse-grid problem.

Algorithm 3.2 (Twogrid method)**.** Given an initial guess $u^{(0)} \in V$.

(i) **Pre-smoothing**: $u^{(1)} = u^{(0)} + \mathcal{S}(f - \mathcal{A}u^{(0)})$;

(ii) **Coarse-grid Correction**: $u^{(2)} = u^{(1)} + (\mathcal{I}_c \mathcal{B}_c \mathcal{I}_c^T)(f - \mathcal{A}u^{(1)})$;

(iii) **Post-smoothing**: $u^{(3)} = u^{(2)} + \mathcal{S}^T(f - \mathcal{A}u^{(2)})$.

We note that this algorithm is very similar to the multigrid algorithm discussed in Chapter 1. It mainly contains two processes: the smoothing steps and the coarse grid correction (CGC). When these two parts are complement to each other, we may except high effectiveness of the resulting algorithm. That is to say, we may choose $\mathcal{S}$, V_c, and $\mathcal{B}_c$ to make the method efficient for the equation. The twogrid method is defined in the hope of capturing the high-frequency components of error on the fine grid, and leaving the low-frequency components to the coarser grid. The effect of coarse grid correction is illustrated in Figure 3.3. Note that these two pictures have different scales.

Convergence analysis of twogrid method

In this section, we will estimate convergence rate of twogrid methods. We now give a few simple lemmas. The first lemma is on the norm of oblique projections (also known as the Kato's lemma) which has been proved and reproved in several different fields; see the paper by Szyld [106] for details.

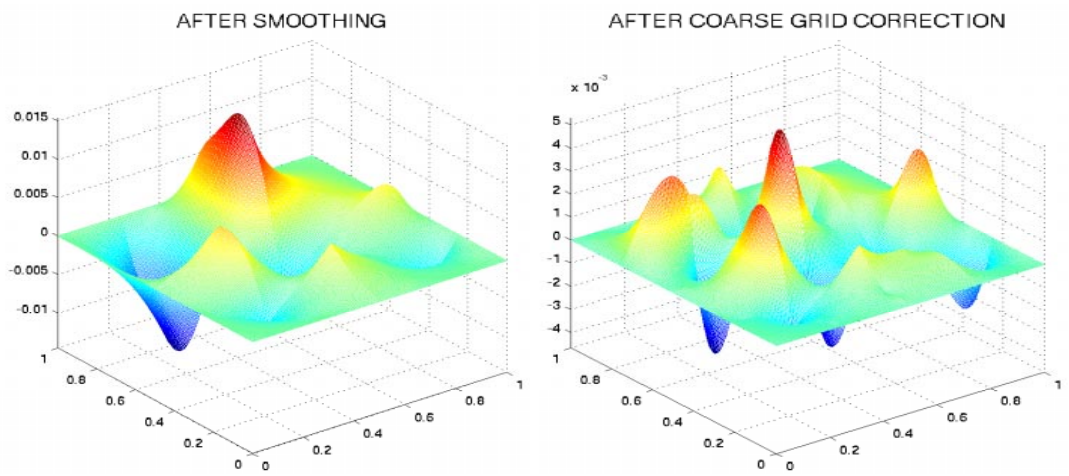


Figure 3.3: After coarse-grid correction, global low frequency is replaced by local high frequency.

Lemma 3.34 (Norm of oblique projections)**.** If Π is a continuous projection onto a Hilbert space $\mathscr{V}$ and Π is neither $\mathcal{I}$ nor 0 , then

$$\|\Pi\| = \|\mathcal{I} - \Pi\|.$$

Proof. Let $u \in \mathscr{V}$ be arbitrary and $\|u\| = 1$. From the assumption on Π, we can take $x := \Pi u \in \text{range}(\Pi)$ and $y := (\mathcal{I} - \Pi)u \in \text{null}(\Pi)$. Then we have

$$1 = \|u\|^2 = \|x\|^2 + \|y\|^2 + 2\,(x, y).$$

If $x = 0$ or $y = 0$, then we have $\Pi u = 0$ or $\|\Pi u\| = 1$, respectively. And, in turn, $\|\Pi u\| \leqslant \|\mathcal{I} - \Pi\|$. If both x and y are nonzero, we define $w := \tilde{x} + \tilde{y} \in \mathscr{V}$, where

$$\tilde{x} := \frac{\|y\|}{\|x\|} x \in \text{range}(\Pi) \quad \text{and} \quad \tilde{y} := \frac{\|x\|}{\|y\|} y \in \text{null}(\Pi).$$

Then $\|w\| = \|x\|^2 + \|y\|^2 + 2\,(x, y) = 1$ and

$$\|\Pi u\| = \|x\| = \|\tilde{y}\| = \|(\mathcal{I} - \Pi)w\| \leqslant \|\mathcal{I} - \Pi\| \quad \Longrightarrow \quad \|\Pi\| \leqslant \|\mathcal{I} - \Pi\|.$$

The other direction can be shown in a similar way and the lemma can be proved. □

Proof of the next two lemmas are straightforward and left to the readers; see HW 3.4.

Lemma 3.35 (Iterator of twogrid method)**.** The twogrid method has a corresponding iterator $\mathcal{B}_{\text{TG}} : V' \to V$ defined as

$$\mathcal{B}_{\text{TG}} = \overline{\mathcal{S}} + \left(\mathcal{I} - \mathcal{S}^T \mathcal{A}\right) \mathcal{I}_c \mathcal{B}_c \mathcal{I}_c^T \left(\mathcal{I} - \mathcal{A}\mathcal{S}\right), \tag{3.28}$$

where $\overline{\mathcal{S}} = \mathcal{S}^T + \mathcal{S} - \mathcal{S}^T \mathcal{A}\mathcal{S}$ is the symmetrization of the smoother $\mathcal{S}$.

Lemma 3.36 (Error propagation of twogrid method)**.** The error propagation operator $\mathcal{E}_{\mathrm{TG}} = \mathcal{I} - \mathcal{B}_{\mathrm{TG}}\mathcal{A}$ for twogrid method is

$$\mathcal{E}_{\mathrm{TG}} = (\mathcal{I} - \mathcal{S}^T\mathcal{A})(\mathcal{I} - \mathcal{B}_c\mathcal{A}_c\Pi_c)(\mathcal{I} - \mathcal{S}\mathcal{A}), \tag{3.29}$$

where Π_c is the $(\cdot,\cdot)_{\mathcal{A}}$-orthogonal projection onto V_c. If the coarse-level solver is exact, namely, $\mathcal{B}_c = \mathcal{A}_c^{-1}$, then we have

$$\mathcal{E}_{\mathrm{TG}} = (\mathcal{I} - \mathcal{S}^T\mathcal{A})(\mathcal{I} - \Pi_c)(\mathcal{I} - \mathcal{S}\mathcal{A}). \tag{3.30}$$

The explicit formula for the projection operator Π_c can be written as $\Pi_c = \mathcal{I}_c\mathcal{A}_c^{-1}\mathcal{I}_c^T\mathcal{A}$. In the above equation (3.30), we observe that it is essential for performance to reduce the norms of the coarse-level correction operator $\mathcal{I} - \Pi_c$ as well as the error reduction operator $\mathcal{I} - \mathcal{S}\mathcal{A}$. Moreover, from Lemma 3.34, analyzing the exact coarse-level correction operator $\|\mathcal{I} - \Pi_c\|_{\mathcal{A}}$ is equivalent to analyze the behavior of $\|\Pi_c\|_{\mathcal{A}}$.

Notice that Π_c is the $\mathcal{A}$-projection from V to V_c. So there is an implicit natural embedding operator $\mathcal{I}_c$ in front of Π_c in the above equality.

We now present a theorem which gives the convergence rate of a simplified twogrid method (Algorithm 3.3) in terms of approximability of the coarser space V_c.

Algorithm 3.3 (Simplified twogrid method)**.** Given an initial guess $u^{(0)} \in V$.

(i) **Coarse-grid Correction**: $u^{(1)} = u^{(0)} + (\mathcal{I}_c\mathcal{B}_c\mathcal{I}_c^T)(f - \mathcal{A}u^{(0)})$;

(ii) **Post-smoothing**: $u^{(2)} = u^{(1)} + \mathcal{S}(f - \mathcal{A}u^{(1)})$.

Assume that $\overline{\mathcal{S}}$ is SPD. In the twogrid method analysis below, we need the following notation

$$\mathcal{T} = \mathcal{T}_{\overline{\mathcal{S}}} := \overline{\mathcal{S}}\mathcal{A} \,:\, V \mapsto V. \tag{3.31}$$

With the above notation, we can define the inner product

$$\big(u, v\big)_{\overline{\mathcal{S}}^{-1}} := \big(\mathcal{T}^{-1}u, v\big)_{\mathcal{A}},$$

the accompanying norm $\|\cdot\|_{\overline{\mathcal{S}}^{-1}}$, and $(\cdot,\cdot)_{\overline{\mathcal{S}}^{-1}}$-orthogonal projection $\mathcal{Q}_{\overline{\mathcal{S}}^{-1}} : V \mapsto V_c$. The convergence rate of the twogrid method is obtained in the following theorem; compare this result with the convergence rate of stationary iterative method in Theorem 2.15.

Theorem 3.37 (Convergence rate of the twogrid method)**.** The convergence rate of the twogrid method, Algorithm 3.3, with the exact coarse-level solver is given by

$$\|\mathcal{E}_{\mathrm{TG}}\|_{\mathcal{A}}^2 = 1 - \frac{1}{c_1(V_c)}, \tag{3.32}$$

where

$$c_1(V_c) := \sup_{v\in V} \frac{\|(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v\|_{\overline{\mathcal{S}}^{-1}}^2}{\|v\|_{\mathcal{A}}^2} = \sup_{v\in V} \inf_{v_c\in V_c} \frac{\|v - v_c\|_{\overline{\mathcal{S}}^{-1}}^2}{\|v\|_{\mathcal{A}}^2}. \tag{3.33}$$

This theorem can be obtained from the X-Z identity; see Theorem 4.15. Here we present a direct proof originally given in [124].

Sketch of the proof. (1) It follows from (3.29) that the simplified TG method has the following error propagation operator

$$\mathcal{E}_{\mathrm{TG}} = (\mathcal{I} - \mathcal{S}\mathcal{A})(\mathcal{I} - \mathit{\Pi}_c).$$

Hence, we can immediately obtain

$$\|\mathcal{E}_{\mathrm{TG}}\|_{\mathcal{A}}^2 = \sup_{v\in V} \frac{\|(\mathcal{I} - \mathcal{S}\mathcal{A})(\mathcal{I} - \mathit{\Pi}_c)v\|_{\mathcal{A}}^2}{\|v\|_{\mathcal{A}}^2} = \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\|(\mathcal{I} - \mathcal{S}\mathcal{A})v\|_{\mathcal{A}}^2}{\|v\|_{\mathcal{A}}^2}.$$

Using the definition of $(\cdot,\cdot)_{\mathcal{A}}$-projection $\mathit{\Pi}_c$, we can show that

$$\|\mathcal{E}_{\mathrm{TG}}\|_{\mathcal{A}}^2 = \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big((\mathcal{I} - \mathcal{T})v, v\big)_{\mathcal{A}}}{\|v\|_{\mathcal{A}}^2} = 1 - \inf_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big(\mathcal{T}v, v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}} = 1 - \inf_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big((\mathcal{I} - \mathit{\Pi}_c)\mathcal{T}v, v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}}.$$

(2) Define

$$\mathcal{X} := (\mathcal{I} - \mathit{\Pi}_c)\mathcal{T} \,:\, V_c^{\perp_{\mathcal{A}}} \mapsto V_c^{\perp_{\mathcal{A}}} \tag{3.34}$$

and it is easy to check that $\mathcal{X}$ is self-adjoint with respect to $(\cdot,\cdot)_{\mathcal{A}}$. A key observation is that the inverse of $\mathcal{X}$ can be explicitly written as

$$\mathcal{Z} = \mathcal{T}^{-1}(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}}).$$

Since $\big(\mathit{\Pi}_c\mathcal{T}^{-1}(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})u, v\big)_{\mathcal{A}} = \big(\mathcal{T}^{-1}(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})u, v\big)_{\mathcal{A}} = \big((\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})u, v\big)_{\overline{\mathcal{S}}^{-1}} = 0$ for any $u \in V_c^{\perp_{\mathcal{A}}}$ and $v \in V_c$, we have $\mathit{\Pi}_c\mathcal{Z} = 0$, which implies that $\mathcal{Z} : V_c^{\perp_{\mathcal{A}}} \mapsto V_c^{\perp_{\mathcal{A}}}$. Furthermore, by the definition of projections, we get

$$\mathcal{X}\mathcal{Z} = (\mathcal{I} - \mathit{\Pi}_c)(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}}) = \mathcal{I} - \mathit{\Pi}_c = \mathcal{I} \quad \text{on } V_c^{\perp_{\mathcal{A}}}.$$

(3) Consequently $\lambda_{\min}(\mathcal{X}) = \lambda_{\max}(\mathcal{Z})^{-1}$. Finally,

$$\begin{aligned}
\lambda_{\max}(\mathcal{Z}) &= \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{(\mathcal{T}^{-1}(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v, v)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}} = \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big((\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v, v\big)_{\overline{\mathcal{S}}^{-1}}}{(v,v)_{\mathcal{A}}} \\
&= \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\|(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v\|_{\overline{\mathcal{S}}^{-1}}^2}{(v,v)_{\mathcal{A}}} = \sup_{v\in V} \frac{\|(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v\|_{\overline{\mathcal{S}}^{-1}}^2}{\|v\|_{\mathcal{A}}^2} =: c_1(V_c).
\end{aligned}$$

The last identity holds because $\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}} = (\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})(\mathcal{I} - \Pi_c)$ and we can then take the supremum back over all $v \in V$ (similar to the argument in the very beginning of this proof). □

Note that, in this theorem, we have only discussed the simplest case where the coarse problem is solved exactly. In practice, the coarse problem is rarely solved exactly. We can also obtain convergence estimates for the inexact twogrid method based on convergence factor of the exact twogrid method; see [93, 126]. This will be discussed later in §6.2.

Optimal coarse space ⋆

Now we discuss how to choose the coarse space to maximize the convergence speed, which will become handy later for developing algebraic multigrid methods (AMGs). We will show that the space spanned by the eigenvectors of $\overline{\mathcal{S}}\mathcal{A}$ corresponding to small eigenvalues gives the "best" coarse space. Here the term "best" refers to the fact that this coarse space minimizes the convergence rate.

Theorem 3.37 provides an estimate on the convergence rate of a twogrid method in terms of $c_1(V_c)$. For a given method, a smaller bound on $c_1(V_c)$ means faster convergence. In particular, the twogrid method is uniformly convergent if $c_1(V_c)$ is uniformly bounded with respect to meshsize. However, one problem for applying Theorem 3.37 is that it is sometimes difficult to work with $\overline{\mathcal{S}}^{-1}$.

A natural approach to overcome such a difficulty is to introduce a simpler but spectrally equivalent SPD operator $\mathcal{D}$, such that

$$C_L \|v\|_{\mathcal{D}}^2 \leqslant \|v\|_{\overline{\mathcal{S}}^{-1}}^2 \leqslant C_U \|v\|_{\mathcal{D}}^2, \quad \forall\, v \in V.$$

Similar to the definition of $c_1(V_c)$, we can introduce the quantity

$$c_1(V_c, \mathcal{D}) = \sup_{v \in V} \frac{\|(\mathcal{I} - \mathcal{Q}_{\mathcal{D}})v\|_{\mathcal{D}}^2}{\|v\|_{\mathcal{A}}^2} = \sup_{v \in V} \inf_{v_c \in V_c} \frac{\|v - v_c\|_{\mathcal{D}}^2}{\|v\|_{\mathcal{A}}^2},$$

where $\mathcal{Q}_{\mathcal{D}} : V \mapsto V_c$ is the $(\cdot,\cdot)_{\mathcal{D}}$-orthogonal projection. Hence

$$C_L\, c_1(V_c, \mathcal{D}) \leqslant c_1(V_c) \leqslant C_U\, c_1(V_c, \mathcal{D}).$$

It is straight-forward to derive the following estimates:

Theorem 3.38 (An estimate of convergence rate of TG)**.** The convergence rate of the twogrid method (3.28) with exact coarse-level solver is given by

$$1 - \frac{1}{C_L\, c_1(V_c, \mathcal{D})} \leqslant \|\mathcal{E}_{\mathrm{TG}}\|_{\mathcal{A}} \leqslant 1 - \frac{1}{C_U\, c_1(V_c, \mathcal{D})} \leqslant 1 - \frac{1}{C_U C}, \tag{3.35}$$

where C is an upper bound of $c_1(V_c, \mathcal{D})$, i.e.,

$$\inf_{v_c \in V_c} \|v - v_c\|_{\mathcal{D}}^2 \leqslant C \|v\|_{\mathcal{A}}^2, \quad \forall\, v \in V. \tag{3.36}$$

The following theorem characterizes the optimal choice of coarse space V_c with a fixed smoother $\mathcal{S}$:

Theorem 3.39 (Optimal coarse space)**.** Given a smoother $\mathcal{S}$, the best coarse space of dimension N_c is given by

$$V_c^{\text{opt}} := \underset{\dim V_c = N_c}{\operatorname{argmin}} \|\mathcal{E}_{\text{TG}}(V_c)\|_{\mathcal{A}} = \operatorname{span}\left\{\xi_k\right\}_{k=1}^{N_c}, \tag{3.37}$$

where $\left\{\xi_k\right\}_{k=1}^{N_c}$ are the eigenfunctions corresponding to the smallest eigenvalues λ_k of $\overline{\mathcal{S}}\mathcal{A}$.

Proof. Recall that $\mathcal{E}_{\text{TG}} = (\mathcal{I} - \mathcal{S}^T\mathcal{A})(\mathcal{I} - \Pi_c)(\mathcal{I} - \mathcal{S}\mathcal{A})$. Since $\mathcal{E}_{\text{TG}}$ depends on V_c we write $\mathcal{E}_{\text{TG}}(V_c)$ and using the same argument as in the proof of Theorem 3.37, we have

$$\|\mathcal{E}_{\text{TG}}(V_c)\|_{\mathcal{A}} \quad = \quad 1 - \min_{v \in V_c^{\perp_{\mathcal{A}}}} \frac{(\overline{\mathcal{S}}\mathcal{A}v, v)_{\mathcal{A}}}{\|v\|_{\mathcal{A}}^2}.$$

Thus,

$$\min_{\dim V_c = N_c} \|\mathcal{E}_{\text{TG}}(V_c)\|_{\mathcal{A}} = 1 - \max_{\dim V_c = N_c} \min_{v \in V_c^{\perp_{\mathcal{A}}}} \frac{(\overline{\mathcal{S}}\mathcal{A}v, v)_{\mathcal{A}}}{\|v\|_{\mathcal{A}}^2}.$$

By the well-known Courant minimax principle [47], we have

$$\max_{\dim V_c = N_c} \min_{v \in V_c^{\perp_{\mathcal{A}}}} \frac{(\overline{\mathcal{S}}\mathcal{A}v, v)_{\mathcal{A}}}{\|v\|_{\mathcal{A}}^2} = \lambda_{N_c+1}$$

and the equality holds if $V_c = V_c^{\text{opt}}$ as given in (3.37). □

Remark 3.40 (Lower bound of contraction factor)**.** Since the coarse space which minimizes the convergence rate is the coarse space which minimizes also $c_1(V_c)$, we have the following inequalities

$$c_1(V_c) = \frac{1}{1 - \|\mathcal{E}_{\text{TG}}\|_{\mathcal{A}}} \geqslant \frac{1}{\lambda_{N_c+1}} \qquad \text{or} \qquad \|\mathcal{E}_{\text{TG}}\|_{\mathcal{A}} \geqslant 1 - \lambda_{N_c+1},$$

which is a lower bound of the contraction factor in terms of size of the small eigenvalues (low frequencies) of $\overline{\mathcal{S}}\mathcal{A}$. □

Since the eigenvalues of $\overline{\mathcal{S}}\mathcal{A}$ are expensive to compute, the practical value of Theorem 3.39 is limited. But it will provide useful guidance in the design practical algebraic multilevel methods in §7.1.

3.5 Matrix representation of the twogrid method

In practice, we have to understand the matrix representation of an abstract algorithm before we can actually implement it. In particular, we wish to answer the questions raised at the end of §1.4. We now explain the matrix representation of the twogrid method in the finite element context.

Grid transfer operators in matrix form

Let $\{\phi_i\}$ be the basis of a finite element space V on the fine-grid. Then the stiffness matrix $\hat{\mathcal{A}}$ reads

$$\left(\hat{\mathcal{A}}\right)_{i,j} = a[\phi_i, \phi_j].$$

Let $\{\phi_l^c\}$ be the basis functions of the coarse-grid subspace $V_c \subset V$ and the stiffness matrix on the coarser space is denote by $\hat{\mathcal{A}}_c$ with $\left(\hat{\mathcal{A}}_c\right)_{k,l} = a[\phi_k^c, \phi_l^c]$. Then ϕ_l^c can be expressed as

$$\phi_l^c = \sum_{i=1}^{N} \left(P\right)_{i,l} \phi_i$$

or

$$(\phi_1^c, \ldots, \phi_{N_c}^c) = (\phi_1, \ldots, \phi_N) P,$$

which defines a *prolongation* matrix $P \in \mathbb{R}^{N \times N_c}$. By definition, this implies that $P = \underline{\mathcal{I}_c}$.

Remark 3.41 (Cannonical prolongation operator)**.** Let $\mathbf{1}_N := (1, 1, \ldots, 1)^T$. Since the basis functions form the partition of unity, it follows that

$$(\phi_1, \cdots, \phi_N)\, \mathbf{1}_N = \sum_{i=1}^{N} \phi_i = 1 = \sum_{l=1}^{N_c} \phi_l^c = (\phi_1^c, \cdots, \phi_{N_c}^c) \mathbf{1}_{N_c} = (\phi_1, \cdots, \phi_N) P\, \mathbf{1}_{N_c}.$$

Hence we have that the prolongation matrix preserves constant away from the boundary, i.e.,

$$P\, \mathbf{1}_{N_c} = \mathbf{1}_N.$$

□

It is important to note that $\underline{\mathcal{I}_c^T} = \underline{\mathcal{Q}_c} \neq \underline{\mathcal{I}_c}^T$, i.e., the matrix representation of adjoint operator is not equal to the transpose of the matrix representation. If we take any $v \in V$, then we have

$$v_c := \mathcal{Q}_c v \qquad \text{and} \qquad v_c = (\phi_1^c, \ldots, \phi_{N_c}^c) \underline{v_c}.$$

On the other hand, with straightforward calculations, we obtain that

$$\vec{v_c} = \Big((v_c, \phi_k^c)\Big)_{k=1}^{N_c} = \Big((v, \phi_k^c)\Big)_{k=1}^{N_c} = \Big(\sum_{j=1}^{N} \underline{v}_j (\phi_j, \phi_k^c)\Big)_{k=1}^{N_c} = \Big(\sum_{j=1}^{N} \underline{v}_j \Big(\underline{\mathcal{I}_c}^T M\Big)_{k,j}\Big)_{k=1}^{N_c} = \underline{\mathcal{I}_c}^T M \underline{v}.$$

In turn, we can obtain the matrix representation of the L^2-projection

$$\underline{\mathcal{Q}_c} \underline{v} = \underline{v_c} = M_c^{-1} \vec{v_c} = M_c^{-1} \underline{\mathcal{I}_c}^T M \underline{v} \quad \Longrightarrow \quad \underline{\mathcal{I}_c^T} = \underline{\mathcal{Q}_c} = M_c^{-1} \underline{\mathcal{I}_c}^T M = M_c^{-1} P^T M. \tag{3.38}$$

Coarse problem in matrix form

Since the coarse-level operator is defined as $\mathcal{A}_c = \mathcal{I}_c^T \mathcal{A} \mathcal{I}_c$, we obtain its matrix representation

$$\underline{\mathcal{A}_c} = \underline{\mathcal{Q}_c}\,\underline{\mathcal{A}}\,\underline{\mathcal{I}_c} \quad \Longrightarrow \quad \hat{\mathcal{A}}_c = M_c \underline{\mathcal{A}_c} = M_c \underline{\mathcal{Q}_c}\,\underline{\mathcal{A}}\,\underline{\mathcal{I}_c} = P^T M \underline{\mathcal{A}} P = P^T \hat{\mathcal{A}} P. \tag{3.39}$$

Then the coarse stiffness matrix satisfies

$$\hat{\mathcal{A}}_c = P^T \hat{\mathcal{A}} P. \tag{3.40}$$

Therefore, the algebraic form (3.40) of the coarse level problem is equivalent to the matrix representation of the operator form.

In the above equality, we observe that, the L^2-projection $\mathcal{Q}_c$ is not needed for implementation. Instead, we only need to use a restriction matrix $R := P^T$.

Remark 3.42 (Finite difference case)**.** Notice that, here, for the finite element stiffness matrices, the restriction matrix is just $R = P^T$. However, we have already noticed that $R \neq P^T$ for the finite difference method in (1.37). In fact, many books (see [44] for example) states $R = \alpha P^T$. This difference comes from the scaling effect caused by h. In the 1D FD example, the coefficient matrices on fine and coarse levels are $A = h^{-1}\hat{\mathcal{A}}$ and $A_c = H^{-1}\hat{\mathcal{A}}_c$, respectively. Hence we get

$$\hat{\mathcal{A}}_c = P^T \hat{\mathcal{A}} P \quad \Longrightarrow \quad A_c = \left(\frac{h}{H} P^T\right) A P =: RAP.$$

This remark explains how we can obtain such the constant α in general. □

Twogrid iterator in matrix form

From (3.28), we have that the twogrid method with exact coarse solver is

$$\mathcal{B}_{\mathrm{TG}} = \overline{\mathcal{S}} + \left(\mathcal{I} - \mathcal{S}^T \mathcal{A}\right) \mathcal{I}_c \mathcal{A}_c^{-1} \mathcal{I}_c^T \left(\mathcal{I} - \mathcal{A}\mathcal{S}\right).$$

We can then write the above equation in matrix form

$$\underline{\mathcal{B}_{\mathrm{TG}}} = \underline{\overline{\mathcal{S}}} + \left(\underline{\mathcal{I}} - \underline{\mathcal{S}^T \mathcal{A}}\right) \underline{\mathcal{I}_c \mathcal{A}_c^{-1} \mathcal{I}_c^T} \left(\underline{\mathcal{I}} - \underline{\mathcal{A}\mathcal{S}}\right).$$

So we define

$$B_{\mathrm{TG}} := \underline{\mathcal{B}_{\mathrm{TG}}} M^{-1} = \underline{\overline{\mathcal{S}}} M^{-1} + \left(\underline{\mathcal{I}} - \underline{\mathcal{S}^T \mathcal{A}}\right) \underline{\mathcal{I}_c \mathcal{A}_c^{-1} \mathcal{I}_c^T} \left(\underline{\mathcal{I}} - \underline{\mathcal{A}\mathcal{S}}\right) M^{-1}.$$

Using the matrix form the symmetrization, inversion, and transpose derived earlier, we can easily get

$$B_{\mathrm{TG}} = \overline{S} + (I - S^T A) P A_c^{-1} P^T (I - AS) = \overline{S} + (I - S^T A) P \left(P^T A P\right)^{-1} P^T (I - AS).$$

Now we are ready to introduce the matrix representation of the twogrid method for solving the linear system $A\underline{u} = \vec{f}$. We describe the twogrid method as a preconditioner action $B_{\mathrm{TG}}(\cdot)$. For any given vector (usually it is the residual vector) $\vec{r} \in \mathbb{R}^N$, we can compute $B_{\mathrm{TG}}(\vec{r})$ in the following steps:

Listing 3.1: A twogrid method

```
%% Given any vector r⃗;
Pre-smoothing: v⃗ ← Sr⃗;
Coarse-grid correction: w⃗ ← v⃗ + P(P^T AP)^{-1} P^T(r⃗ − Av⃗);
Post-smoothing: B_TG r⃗ ← w⃗ + S^T(r⃗ − Aw⃗);
```

Similarly, from (3.30), we have matrix form of the iteration matrix

$$\begin{aligned} E_{\mathrm{TG}} = \underline{\mathcal{E}_{\mathrm{TG}}} &= (I - S^T A)(I - P A_c^{-1} P^T A)(I - SA) \\ &= (I - S^T A)(I - \Pi_c)(I - SA), \end{aligned} \tag{3.41}$$

where $\Pi_c := \underline{\Pi_c} = P A_c^{-1} P^T A$ is the matrix form of the coarse-level correction; see HW 3.8.

In [56], an algebraic analysis of the twogrid method has been given and the convergence rate of the TG method can be written as

$$\rho(E_{\mathrm{TG}}) = 1 - \inf_{v} \frac{v^T (I - \tilde{\Pi}_c) A^{\frac{1}{2}} \overline{S} A^{\frac{1}{2}} (I - \tilde{\Pi}_c) v}{v^T (I - \tilde{\Pi}_c) v},$$

where $\tilde{\Pi}_c := A^{\frac{1}{2}} \Pi_c A^{-\frac{1}{2}} = A^{\frac{1}{2}} P A_c^{-1} P^T A^{\frac{1}{2}}$. This algebraic form is explicit and might be easier to understand compared with Theorem 3.37.

3.6 Homework problems

HW 3.1. Show the a posteriori error bounds (3.9).

HW 3.2. Prove the statements in Lemma 3.20.

HW 3.3. Show the operator form and matrix form (3.15) of the Richardson method.

HW 3.4. Prove Lemma 3.35 and Lemma 3.36.

HW 3.5. Give a complete proof of Remark 3.32.

HW 3.6. Write the 1D multigrid method in §1.4 as a twogrid method (Algorithm 3.2) called recursively and modify your implementation in this way.

HW 3.7. Give the detailed proof of Theorem 3.37. Hint: First show that

$$\sup_{v\in V}\frac{\|(\mathcal{I}-\mathcal{S}\mathcal{A})(\mathcal{I}-\Pi_c)v\|_{\mathcal{A}}^2}{\|v\|_{\mathcal{A}}^2}=\sup_{v\in V}\frac{\|(\mathcal{I}-\mathcal{S}\mathcal{A})(\mathcal{I}-\Pi_c)v\|_{\mathcal{A}}^2}{\|(\mathcal{I}-\Pi_c)v\|_{\mathcal{A}}^2+\|\Pi_c v\|_{\mathcal{A}}^2}=\sup_{v\in V_c^{\perp_{\mathcal{A}}}}\frac{\|(\mathcal{I}-\mathcal{S}\mathcal{A})v\|_{\mathcal{A}}^2}{\|v\|_{\mathcal{A}}^2};$$

Then prove that $\mathcal{X}$ defined in (3.34) is self-adjoint with respect to $(\cdot,\cdot)_{\mathcal{A}}$-inner product.

HW 3.8. Derive the primal matrix representation of Π_c and $\mathcal{E}_{\mathrm{TG}}$ respectively.

Chapter 4

Subspace Correction Methods

In the previous chapters, we have introduced several iterative solvers for the linear equation

$$\mathcal{A}u = f, \tag{4.1}$$

where $\mathcal{A} : V \mapsto V$ is SPD. A linear stationary iterative method can be written as

$$u^{\text{new}} = u^{\text{old}} + \mathcal{B}(f - \mathcal{A}u^{\text{old}}). \tag{4.2}$$

In Chapter 2, we have seen that: If $\mathcal{B}$ is an SPD operator, with proper scaling, the above iterative method (4.2) converges; Furthermore, $\mathcal{B}$ can be applied as a preconditioner of Krylov subspace methods, like PCG.

In this chapter, we present a theoretical framework for analyzing linear iterative methods and/or preconditioners in terms of space decomposition and subspace corrections. This general framework can be used to establish convergence theory for various methods, including the multigrid method, the domain decomposition method, and the twogrid method discussed in the previous chapters.

4.1 Successive and parallel subspace corrections

Suppose we have a subspace decomposition of the solution space

$$V = \sum_{j=1}^{J} V_j \qquad \text{and} \qquad V_j \subset V \ \ (j = 1, \ldots, J).$$

For any $v \in V$, we can write it as $v = \sum_{j=1}^{J} v_j$ with $v_j \in V_j$. Notice that this representation is not unique as there could be redundancy in the subspace decomposition. Later on, it will become clear that such redundancy is crucial for constructing optimal multilevel methods.

Abstract framework for subspace corrections

We first define a few operators which have already been used at different places in the previous chapters.

Definition 4.1. Let V be a finite-dimensional Hilbert space with inner product $(\cdot,\cdot)$ and $V_j \subset V$ be a subspace. We define

$$\begin{cases} \text{subspace problem} & \mathcal{A}_j : V_j \mapsto V_j, & (\mathcal{A}_j v_j, w_j) = (\mathcal{A} v_j, w_j), & \forall\, v_j, w_j \in V_j; \\ (\cdot,\cdot)\text{-projection} & \mathcal{Q}_j : V \mapsto V_j, & (\mathcal{Q}_j v, w_j) = (v, w_j), & \forall\, w_j \in V_j; \\ (\cdot,\cdot)_{\mathcal{A}}\text{-projection} & \Pi_j : V \mapsto V_j, & (\Pi_j v, w_j)_{\mathcal{A}} = (v, w_j)_{\mathcal{A}}, & \forall\, w_j \in V_j. \end{cases}$$

Using Definition 4.1, we have the following elementary results:

Lemma 4.2 (Relation between projections)**.** The following equalities hold:

1. $\mathcal{I}_j^T = \mathcal{Q}_j$, $\mathcal{I}_j^* = \Pi_j$;
2. $\mathcal{Q}_j \mathcal{A} = \mathcal{A}_j \Pi_j$.

Proof. (i) By definition, for any $u \in \mathcal{V}, v_j \in \mathcal{V}_j$, we have

$$(\mathcal{Q}_j u, v_j) = (u, v_j) = (u, \mathcal{I}_j v_j) = (\mathcal{I}_j^T u, v_j),$$
$$(\Pi_j u, v_j)_{\mathcal{A}} = (u, v_j)_{\mathcal{A}} = (u, \mathcal{I}_j v_j)_{\mathcal{A}} = (\mathcal{I}_j^* u, v_j)_{\mathcal{A}}.$$

(ii) For any $u \in \mathcal{V}, v_j \in \mathcal{V}_j$, we have

$$(\mathcal{A}_j \Pi_j u, v_j) = (\Pi_j u, v_j)_{\mathcal{A}} = (u, v_j)_{\mathcal{A}} = (u, \mathcal{I}_j v_j)_{\mathcal{A}} = (\mathcal{A} u, \mathcal{I}_j v_j) = (\mathcal{Q}_j \mathcal{A} u, v_j),$$

which gives the second identity. □

Remark 4.3 (Matrix representation of the $\mathcal{A}$-projection)**.** Let $u_c := \Pi_c u$. Since $\Pi_c : V \mapsto V_c \subset V$ is the $\mathcal{A}$-orthogonal projection operator, for any $u \in V$, we have

$$a[u_c, v_c] = a[\Pi_c u, v_c] = a[u, v_c], \quad \forall\, v_c \in V_c.$$

Using the matrix representation notations introduced in §3.2, we have, for any $v_c \in V_c$, that

$$a[u_c, v_c] = (\mathcal{A} u_c, v_c) = \underline{v_c}^T \hat{\mathcal{A}}_c \, \underline{u_c}, \qquad \forall\, u_c \in V_c; \tag{4.3}$$

$$a[u, v_c] = (\mathcal{A} u, v_c) = (\underline{\mathcal{I}_c v_c})^T \hat{\mathcal{A}} \, \underline{u} = \underline{v_c}^T P^T \hat{\mathcal{A}} \, \underline{u}, \qquad \forall\, u \in V. \tag{4.4}$$

From (4.3) and (4.4), we can derive the matrix representation of the Galerkin projection on the coarse grid

$$\hat{\mathcal{A}}_c \underline{u_c} = P^T \hat{\mathcal{A}} \underline{u} \quad \Longrightarrow \quad \underline{\Pi_c} \, \underline{u} = \underline{\Pi_c u} = \underline{u_c} = \hat{\mathcal{A}}_c^{-1} P^T \hat{\mathcal{A}} \underline{u}.$$

Hence, we obtain the matrix representation of the $\mathcal{A}$-projection operator

$$\underline{\Pi_c} = \hat{\mathcal{A}}_c^{-1} P^T \hat{\mathcal{A}}. \tag{4.5}$$

One can compare this equation with the matrix form of the L^2-projection in (3.38). □

Remark 4.4 (Subspace problems)**.** From the definition of $\mathcal{A}_j$, we get

$$\mathcal{A}_j = \mathcal{I}_j^T \mathcal{A} \mathcal{I}_j = \mathcal{Q}_j \mathcal{A} \mathcal{I}_j = \mathcal{Q}_j \mathcal{A} \mathcal{Q}_j^T.$$

With the help of Lemma 4.2 and simple calculations, we can immediately obtain the error equation on each subspace V_j:

$$\mathcal{A}e = r \quad \Longrightarrow \quad Q_j \mathcal{A} e = \mathcal{Q}_j r \quad \Longrightarrow \quad \mathcal{A}_j \Pi_j e = \mathcal{Q}_j r \quad \Longrightarrow \quad \mathcal{A}_j e_j = r_j,$$

where $r_j = \mathcal{Q}_j r$ and $e_j = \Pi_j e$. □

The main idea of method of subspace corrections (MSC), namely *divide and conquer*, has already been discussed in the domain decomposition method. We first describe the idea of subspace correction in the following abstract algorithm[1], which is just a generalization of Algorithm 2.1:

Algorithm 4.1 (Method of subspace corrections)**.** $u^{\text{new}} = SC(u^{\text{old}})$

(i) **Form residual**: $r = f - \mathcal{A}u^{\text{old}}$

(ii) **Solve error equation on** V_j: $\mathcal{A}_j e_j = r_j$ by $e_j \approx \hat{e}_j = \mathcal{S}_j r_j$

(iii) **Apply correction**: $u^{\text{new}} = u^{\text{old}} + \hat{e}_j$

Notice that, instead of constructing an iterator for the whole system, Algorithm 4.1 only considers one subproblem on the subspace V_j. It is still not clear how to taking all subspaces into account. In fact, the ordering of subspace corrections plays a key role in algorithm construction.

Remark 4.5 (Subspace solvers)**.** It is well-known that

$$u_j = \operatorname*{argmin}_{v \in V_j} \mathcal{F}(v) := \frac{1}{2}(\mathcal{A}v, v) - (f, v)$$

is equivalent to

$$u_j = \operatorname*{argmin}_{v \in V_j} \|u - v\|_{\mathcal{A}}.$$

[1]Note that this procedure is not really an algorithm as it does not specify how to combine the corrections $\hat{e}_j$'s from different subspaces.

We notice that the solution of the subspace problem $\mathcal{A}_j e_j = r_j = \mathcal{Q}_j r^{\text{old}}$ satisfies that

$$\mathcal{F}(u^{\text{old}} + e_j) = \min_{e \in V_j} \mathcal{F}(u^{\text{old}} + e).$$

In order to provide an effective yet practical subspace solver, we should pay attention to the dimension of the subspace and choose an appropriate problem size. □

SSC and PSC methods

Algorithm 4.1 does not specify how to combine the corrections $\hat{e}_j$'s from different subspaces. There are two basic approaches: the successive subspace correction (SSC) and the parallel subspace correction (PSC). SSC can be viewed as the multiplicative Schwarz method (2.37) and PSC can be viewed as the additive Schwarz method (2.36). We now give descriptions of the SSC and PSC algorithms.

Algorithm 4.2 (Successive subspace corrections)**.** $u^{\text{new}} = SSC(u^{\text{old}})$

(i) $v = u^{\text{old}}$

(ii) $v = v + \mathcal{S}_j \mathcal{Q}_j (f - \mathcal{A} v), \quad j = 1, \ldots, J$

(iii) $u^{\text{new}} = v$

Remark 4.6 (Relaxation for subspace solvers)**.** In the above algorithm, we can introduce a relaxation parameter in each subspace correction step

$$v = v + \omega_j \mathcal{S}_j \mathcal{Q}_j (f - \mathcal{A} v), \quad j = 1, \ldots, J.$$

Good relaxation parameters are difficult to obtain in general, but they can improve convergence if optimal values can be found. We will not discuss this modified subspace correction though because ω_j can always be absorbed in $\mathcal{S}_j$. □

Algorithm 4.3 (Parallel subspace corrections)**.** $u^{\text{new}} = PSC(u^{\text{old}})$

(i) $r = f - \mathcal{A} u^{\text{old}}$

(ii) $\hat{e}_j = \mathcal{S}_j \mathcal{Q}_j r, \ j = 1, \ldots, J$

(iii) $u^{\text{new}} = u^{\text{old}} + \sum_{j=1}^{J} \hat{e}_j$

From the above algorithms, it is immediately clear why they are named as PSC and SSC, respectively. As in (3.31), we define an operator

$$\mathcal{T}_j = \mathcal{T}_{\mathcal{S}_j} := \mathcal{S}_j \mathcal{Q}_j \mathcal{A} = \mathcal{S}_j \mathcal{A}_j \Pi_j : V \mapsto V_j.$$

Apparently, if we restrict the domain to V_j, then we have

$$\mathcal{T}_j = \mathcal{T}_{\mathcal{S}_j} = \mathcal{S}_j \mathcal{A}_j : V_j \mapsto V_j.$$

We shall now assume all the subspace solvers (smoothers) $\mathcal{S}_j$ are SPD operators. As $\mathcal{S}_j^T = \mathcal{S}_j$, the operator $\mathcal{T}_j = \mathcal{S}_j \mathcal{A}_j : V_j \mapsto V_j$ is symmetric and positive definite with respect to $(\cdot,\cdot)_{\mathcal{A}}$. If $\mathcal{S}_j = \mathcal{A}_j^{-1}$, i.e., the smoother is the exact solver on each subspace, then we have $\mathcal{T}_j = \mathit{\Pi}_j$.

- The SSC method satisfies:

$$u - u^{\text{new}} = (\mathcal{I} - \mathcal{B}_{\text{SSC}}\mathcal{A})(u - u^{\text{old}}) = (\mathcal{I} - \mathcal{T}_J) \cdots (\mathcal{I} - \mathcal{T}_1)(u - u^{\text{old}}). \tag{4.6}$$

  If $J = N$ and each subspace $V_j = \text{span}\{\phi_j\}$ $(j = 1, \ldots, N)$ and $\mathcal{S}_j = \mathcal{A}_j^{-1}$, then the corresponding SSC method (4.6) is exactly the G-S method; see (2.20).

- For the PSC method, the iterator (or, more often, the preconditioner) satisfies

$$\mathcal{B}_{\text{PSC}} = \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j = \sum_{j=1}^{J} \mathcal{I}_j \mathcal{S}_j \mathcal{Q}_j \qquad \text{and} \qquad \mathcal{B}_{\text{PSC}}\mathcal{A} = \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j \mathcal{A} = \sum_{j=1}^{J} \mathcal{T}_j. \tag{4.7}$$

  If $\mathcal{S}_j$'s $(j = 1, \ldots, J)$ are all SPD, then the preconditioner $\mathcal{B}_{\text{PSC}}$ is also SPD; see HW 4.2. If each subspace $V_j = \text{span}\{\phi_j\}$ $(j = 1, \ldots, N)$ as before, then the resulting PSC methods with $\mathcal{S}_j = \omega(\cdot, \phi_j)\phi_j$ and $\mathcal{S}_j = \mathcal{A}_j^{-1}$ correspond to the Richardson method and the Jacobi method, respectively.

So far, we have not mentioned any multilevel structures in the above methods. In order to introduce multilevel iterative methods in the subspace correction framework, we will need multilevel subspace decompositions.

4.2 Expanded system and block solvers

Recall that, back in §2.1, we discussed a modified block Gauss–Seidel method. In this section, we discuss an expanded system of (4.1) and its block iterative solvers. Moreover, we will show how these block solvers are related to the subspace correction methods for the original linear system (4.1). This relation will become important in the next section for deriving the XZ identity, which gives the convergence rate of SSC.

Expansion of the original problem

Suppose that the finite dimensional vector space V can be decomposed as the summation of linear vector subspaces (might not be linearly independent), $V_1, V_2, \ldots, V_J$, i.e., $V = \sum_{j=1}^{J} V_j$. We define a new vector space

$$\mathbf{V} := V_1 \times V_2 \times \cdots \times V_J.$$

Define an operator $\mathbf{\Pi} : \mathbf{V} \mapsto V$ such that $\mathbf{\Pi u} = \sum_{j=1}^{J} u_j$, where $\mathbf{u} = (u_1, \ldots, u_J)^T \in \mathbf{V}$ with each component $\mathbf{u}_j = u_j \in V_j$. From the definition, it is easy to see that $\mathbf{\Pi}$ is surjective. This operator can be formally interpreted as

$$\mathbf{\Pi} = (\mathcal{I}_1, \ldots, \mathcal{I}_J),$$

where $\mathcal{I}_j$ is the natural embedding from V_j to V. Hence, we obtain

$$\mathbf{\Pi u} = (\mathcal{I}_1, \ldots, \mathcal{I}_J) \begin{pmatrix} u_1 \\ \vdots \\ u_J \end{pmatrix} = \sum_{j=1}^{J} \mathcal{I}_j u_j = \sum_{j=1}^{J} u_j.$$

So we have

$$\mathbf{\Pi}^T = \begin{pmatrix} \mathcal{I}_1^T \\ \vdots \\ \mathcal{I}_J^T \end{pmatrix} = \begin{pmatrix} \mathcal{Q}_1 \\ \vdots \\ \mathcal{Q}_J \end{pmatrix}.$$

Note that $\mathbf{\Pi}\,\mathbf{\Pi}^T \neq \mathcal{I}$ in general.

Define $\mathbf{A} : \mathbf{V} \mapsto \mathbf{V}$ such that $\mathbf{A}_{i,j} = \mathcal{A}_{i,j} := \mathcal{I}_i^T \mathcal{A} \mathcal{I}_j : V_j \mapsto V_i$. And we denote $\mathcal{A}_j := \mathcal{A}_{j,j}$ $(j = 1, \ldots, J)$. Hence we can write the operator $\mathbf{A}$ in a matrix form

$$\mathbf{A} := \mathbf{\Pi}^T \mathcal{A}\, \mathbf{\Pi} = \left(\mathbf{A}_{i,j}\right)_{J \times J} = \begin{pmatrix} \mathcal{A}_{1,1} & \cdots & \mathcal{A}_{1,J} \\ \vdots & \ddots & \vdots \\ \mathcal{A}_{J,1} & \cdots & \mathcal{A}_{J,J} \end{pmatrix}.$$

Given any right hand side function $f \in V$, we define

$$\mathbf{f} := \mathbf{\Pi}^T f = \begin{pmatrix} \mathcal{I}_1^T f \\ \vdots \\ \mathcal{I}_J^T f \end{pmatrix} \in \mathbf{V}.$$

In this setting, we consider the following problem: Find $\mathbf{u} \in \mathbf{V}$, such that

$$\mathbf{A u} = \mathbf{f}. \tag{4.8}$$

This system is called the *expanded equation* of the original linear equation (4.1). We will see how the solution of these two problems are related. If $\mathcal{A}$ is SPD, then $\mathbf{A}$ is a symmetric positive semidefinite (SPSD) operator. Note that $\mathbf{A}$ is usually singular due to its nontrivial null space, null($\mathbf{\Pi}$). However, its diagonal entries $\mathcal{A}_j$ $(j = 1, 2, \ldots, J)$ are non-singular. We can define a semi-norm for $\mathbf{B} : \mathbf{V} \mapsto \mathbf{V}$

$$\|\mathbf{B}\|_{\mathbf{A}} := \sup_{\|\mathbf{v}\|_{\mathbf{A}} \neq 0} \frac{\|\mathbf{B v}\|_{\mathbf{A}}}{\|\mathbf{v}\|_{\mathbf{A}}}.$$

Block solvers for expanded equation

As before, we denote the lower, upper, and diagonal part of $\mathbf{A}$ as $\mathbf{L}$, $\mathbf{U}$, and $\mathbf{D}$, respectively. We can immediately see that the stationary iterative methods discussed in §1.3 can be easily adapted to solve (4.8). The linear stationary iterative methods for (4.8) can be written in the following abstract form

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + \mathbf{B}(\mathbf{f} - \mathbf{A}\mathbf{u}^{\text{old}}), \tag{4.9}$$

where the iterator $\mathbf{B} : \mathbf{V} \mapsto \mathbf{V}$. If $\mathbf{B} = \mathbf{D}^{-1}$, then we have the block Jacobi method for (4.8); if $\mathbf{B} = (\mathbf{D} + \mathbf{L})^{-1}$, then we have the block Gauss–Seidel method.

Motivated by (2.16), we can generalize the block Jacobi and G-S methods. Assume there is a non-singular block diagonal smoother (or relaxation operator) $\mathbf{S} : \mathbf{V} \mapsto \mathbf{V}$, i.e.,

$$\mathbf{S} = \text{diag}(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_J), \quad \text{with } \mathcal{S}_j : V_j \mapsto V_j, \ \ j = 1, 2, \ldots, J.$$

We define modified block Jacobi method by $\mathbf{B} = \mathbf{S}$ and the modified block Gauss–Seidel method by $\mathbf{B} = (\mathbf{S}^{-1} + \mathbf{L})^{-1}$.

Theorem 4.7 (Solution of expanded and original systems)**.** The linear stationary iteration (4.9) for the equation (4.8) reduces to an equivalent stationary iteration (4.2) with the iterator

$$\mathcal{B} = \mathbf{\Pi}\, \mathbf{B}\, \mathbf{\Pi}^T$$

for the original equation (4.1). Moreover, these two methods have the same convergence speed, namely,

$$\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}} = \|\mathbf{I} - \mathbf{B}\mathbf{A}\|_{\mathbf{A}}.$$

Proof. The linear stationary iterative method

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + \mathbf{B}(\mathbf{f} - \mathbf{A}\mathbf{u}^{\text{old}})$$

is equivalent to

$$\begin{aligned} \mathbf{u}_j^{\text{new}} &= \mathbf{u}_j^{\text{old}} + \sum_k \mathbf{B}_{j,k} \Big(\mathcal{I}_k^T f - \sum_i \mathbf{A}_{k,i} \mathbf{u}_i^{\text{old}} \Big) \\ &= \mathbf{u}_j^{\text{old}} + \sum_k \mathbf{B}_{j,k} \mathcal{I}_k^T \Big(f - \sum_i \mathcal{A}\mathcal{I}_i \mathbf{u}_i^{\text{old}} \Big) \ = \ \mathbf{u}_j^{\text{old}} + \sum_k \mathbf{B}_{j,k} \mathcal{I}_k^T \Big(f - \mathcal{A} u^{\text{old}} \Big). \end{aligned}$$

Therefore, we have

$$u^{\text{new}} = \sum_j \mathcal{I}_j \mathbf{u}_j^{\text{new}} = u^{\text{old}} + \sum_{j,k} \mathcal{I}_j \mathbf{B}_{j,k} \mathcal{I}_k^T \Big(f - \mathcal{A} u^{\text{old}} \Big) = u^{\text{old}} + \mathcal{B} \Big(f - \mathcal{A} u^{\text{old}} \Big).$$

This proves the equivalence of (4.9) and (4.2).

A key observation is that

$$(\mathbf{BAv},\mathbf{v})_{\mathbf{A}} = (\mathbf{ABAv},\mathbf{v}) = (\mathbf{\Pi}^T\mathcal{A}\,\mathbf{\Pi}\,\mathbf{B}\,\mathbf{\Pi}^T\mathcal{A}\,\mathbf{\Pi v},\mathbf{v}) = (\mathcal{AB}\mathcal{A}\,\mathbf{\Pi v},\mathbf{\Pi v}) = (\mathcal{BA}\,\mathbf{\Pi v},\mathbf{\Pi v})_{\mathcal{A}}.$$

The contraction factor can be written

$$\begin{aligned}
\|\mathcal{I}-\mathcal{BA}\|_{\mathcal{A}}^2 &= \sup_{v\neq 0}\frac{\|(\mathcal{I}-\mathcal{BA})v\|_{\mathcal{A}}^2}{\|v\|_{\mathcal{A}}^2} = \sup_{v\neq 0}\frac{(v,v)_{\mathcal{A}}-\big((\mathcal{B}^T+\mathcal{B}-\mathcal{B}^T\mathcal{AB})\mathcal{A}v,v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}}\\
&= \sup_{\mathbf{\Pi v}\neq 0}\frac{(\mathbf{\Pi v},\mathbf{\Pi v})_{\mathcal{A}}-\big((\mathcal{B}^T+\mathcal{B}-\mathcal{B}^T\mathcal{AB})\mathcal{A}\mathbf{\Pi v},\mathbf{\Pi v}\big)_{\mathcal{A}}}{(\mathbf{\Pi v},\mathbf{\Pi v})_{\mathcal{A}}}\\
&= \sup_{\|\mathbf{v}\|_{\mathbf{A}}\neq 0}\frac{(\mathbf{v},\mathbf{v})_{\mathbf{A}}-\big((\mathbf{B}^T+\mathbf{B}-\mathbf{B}^T\mathbf{AB})\mathbf{Av},\mathbf{v}\big)_{\mathbf{A}}}{\|\mathbf{v}\|_{\mathbf{A}}^2}\\
&= \|\mathbf{I}-\mathbf{BA}\|_{\mathbf{A}}^2.
\end{aligned}$$

Hence we get the desired result. □

Example 4.8 (Block Jacobi method and PSC)**.** We now apply the block Jacobi method for the expanded system (4.8), i.e.,

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + \mathbf{D}^{-1}(\mathbf{f}-\mathbf{Au}^{\text{old}}).$$

We notice that $\mathbf{D}^{-1}\mathbf{A} = \mathbf{D}^{-1}\mathbf{\Pi}^T\mathcal{A}\mathbf{\Pi}$, which is spectrally equivalent to $\mathbf{\Pi D}^{-1}\mathbf{\Pi}^T\mathcal{A}$ because $\sigma(\mathcal{BA})\backslash\{0\} = \sigma(\mathcal{AB})\backslash\{0\}$. In fact, from Theorem 4.7, we can see that the above iterative method is equivalent to

$$u^{\text{new}} = u^{\text{old}} + \mathbf{\Pi D}^{-1}\mathbf{\Pi}^T(f-\mathcal{A}u^{\text{old}}) = u^{\text{old}} + \sum_{j=1}^{J}\mathcal{I}_j\mathcal{A}_j^{-1}\mathcal{I}_j^T\left(f-\mathcal{A}u^{\text{old}}\right).$$

We immediately recognize that this is the PSC method (or the additive Schwarz method) with exact subspace solvers. □

Example 4.9 (Block G-S method and SSC)**.** Similar to the above example, we can get the block G-S method is just the SSC method (or the multiplicative Schwarz method) for the original problem. We now apply the block G-S method for the expanded system (4.8), i.e.,

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + (\mathbf{D}+\mathbf{L})^{-1}(\mathbf{f}-\mathbf{Au}^{\text{old}}).$$

We can rewrite this method as

$$(\mathbf{D}+\mathbf{L})\mathbf{u}^{\text{new}} = (\mathbf{D}+\mathbf{L})\mathbf{u}^{\text{old}} + (\mathbf{f}-\mathbf{Au}^{\text{old}}).$$

Hence we have

$$\mathbf{Du}^{\text{new}} = \mathbf{Du}^{\text{old}} + \mathbf{f} - \mathbf{Lu}^{\text{new}} - (\mathbf{D}+\mathbf{U})\mathbf{u}^{\text{old}};$$

in turn, we get

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + \mathbf{D}^{-1}\Big(\mathbf{f} - \mathbf{L}\mathbf{u}^{\text{new}} - (\mathbf{D}+\mathbf{U})\mathbf{u}^{\text{old}}\Big).$$

For $j = 1, \ldots, J$, the block G-S method can be written as

$$u_j^{\text{new}} = u_j^{\text{old}} + \mathcal{A}_j^{-1}\Big(\mathcal{I}_j^T f - \sum_{i<j}\mathcal{I}_j^T\mathcal{A}\mathcal{I}_i u_i^{\text{new}} - \sum_{i\geqslant j}\mathcal{I}_j^T\mathcal{A}\mathcal{I}_i u_i^{\text{old}}\Big).$$

We define iteration

$$u^{\frac{j}{J}} := \sum_{i<j} u_i^{\text{new}} + \sum_{i\geqslant j} u_i^{\text{old}} = \sum_{i<j}\mathcal{I}_i u_i^{\text{new}} + \sum_{i\geqslant j}\mathcal{I}_i u_i^{\text{old}}, \quad j = 1, \ldots, J.$$

By this definition, we can see that

$$u^{\frac{j+1}{J}} = u^{\frac{j}{J}} + \mathcal{I}_j u_j^{\text{new}} - \mathcal{I}_j u_j^{\text{old}} = u^{\frac{j}{J}} + \mathcal{I}_j\mathcal{A}_j^{-1}\mathcal{I}_j^T(f - \mathcal{A}u^{\frac{j}{J}}).$$

Here the term $f - \mathcal{A}u^{\frac{j}{J}}$ is sometimes called the *dynamic residual*, which is the residual at an inner iteration of the G-S method. From the above equation, we notice that the block G-S method is just the SSC method with exact subspace solvers $\mathcal{S}_j = \mathcal{A}_j^{-1}$ for the original linear equation (4.1). □

Convergence of block solvers

Motived by the weighted Jacobi and G-S methods, we assume that there is an invertible smoother or local relaxation $\mathbf{S}$ for solving $\mathbf{A}\mathbf{u} = \mathbf{f}$. Similar to the method presented in §2.1, we define a general or modified block G-S method:

$$\mathbf{B} := \big(\mathbf{S}^{-1} + \mathbf{L}\big)^{-1}. \tag{4.10}$$

We analyze the convergence rate of this method. Let $\mathbf{K} := \mathbf{B}^{-T} + \mathbf{B}^{-1} - \mathbf{A}$ be a symmetric operator and the symmetrization operator as $\overline{\mathbf{B}} = \mathbf{B}^T\mathbf{K}\mathbf{B}$. Then we get

$$\Big(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}\Big) = \Big(\mathbf{B}^{-1}\mathbf{K}^{-1}\mathbf{B}^{-T}\mathbf{v}, \mathbf{v}\Big) = \Big(\big(\mathbf{S}^{-1} + \mathbf{L}\big)\mathbf{K}^{-1}\big(\mathbf{S}^{-T} + \mathbf{U}\big)\mathbf{v},\ \mathbf{v}\Big), \quad \forall \mathbf{v} \in \mathbf{V} \tag{4.11}$$

By the definition of $\mathbf{K}$, it is clear that $\mathbf{K}$ is diagonal and

$$\begin{aligned}\mathbf{K} &= (\mathbf{S}^{-T} + \mathbf{U}) + (\mathbf{S}^{-1} + \mathbf{L}) - (\mathbf{D} + \mathbf{L} + \mathbf{U}) \\ &= \mathbf{S}^{-T} + \mathbf{S}^{-1} - \mathbf{D} = \mathbf{S}^{-T}\big(\mathbf{S}^T + \mathbf{S} - \mathbf{S}^T\mathbf{D}\mathbf{S}\big)\mathbf{S}^{-1}.\end{aligned}$$

Hence, its inverse matrix is also diagonal and

$$\mathbf{K}^{-1} = \mathbf{S}(\mathbf{S}^T + \mathbf{S} - \mathbf{S}^T\mathbf{D}\mathbf{S})^{-1}\mathbf{S}^T. \tag{4.12}$$

Using the definition of $\mathbf{K}$, we can obtain that $\mathbf{B}^{-1} = \mathbf{K} + \mathbf{A} - \mathbf{B}^{-T}$. Hence we have a representation of $\overline{\mathbf{B}}^{-1}$ by simple manipulations:

$$\overline{\mathbf{B}}^{-1} = (\mathbf{K} + \mathbf{A} - \mathbf{B}^{-T})\mathbf{K}^{-1}(\mathbf{K} + \mathbf{A} - \mathbf{B}^{-1}) = \mathbf{A} + (\mathbf{A} - \mathbf{B}^{-T})\mathbf{K}^{-1}(\mathbf{A} - \mathbf{B}^{-1}).$$

The last equality and (4.10) immediately yield another important identity:

$$\left(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}\right) = (\mathbf{A}\mathbf{v}, \mathbf{v}) + \left(\mathbf{K}^{-1}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}, (\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\right), \quad \forall \mathbf{v} \in \mathbf{V}. \tag{4.13}$$

Now we apply a modification of Theorem 2.15 (i.e., general convergence rate estimate for SPD problems[2]) and get the following convergence result:

Theorem 4.10 (Convergence rate of modified block G-S)**.** If $\mathbf{K} := \mathbf{S}^{-T} + \mathbf{S}^{-1} - \mathbf{D}$ is SPD, then the modified block G-S method converges and

$$\|\mathbf{I} - \mathbf{B}\mathbf{A}\|_{\mathbf{A}}^2 = 1 - \frac{1}{1 + c_0}, \quad \text{with } c_0 := \sup_{\|\mathbf{v}\|_{\mathbf{A}}=1} \left\|\mathbf{K}^{-\frac{1}{2}}\left(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1}\right)\mathbf{v}\right\|^2.$$

4.3 Convergence analysis of SSC

In the previous section, we have found that the SSC method for the original equation is equivalent to the block G-S method for the expanded equation using the same subspaces $\{V_j\}_{j=1}^J$. Now we try to analyze the convergence rate of the block G-S method for the expanded system. In this way, we can give a convergence analysis for the successive subspace correction method. The proof here follows the discussion in [45].

A technical lemma

Suppose $V = \sum_{j=1}^J V_j$. It is clear that $\mathbf{\Pi} : \mathbf{V} \mapsto V$ is surjective and $\mathbf{\Pi}\mathbf{u} = \sum_{j=1}^J \mathcal{I}_j \mathbf{u}_j$. We have the following simple but useful lemma:

Lemma 4.11. If the iterator $\mathbf{B}$ in (4.9) is SPD, then $\mathcal{B} = \mathbf{\Pi}\,\mathbf{B}\,\mathbf{\Pi}^T$ is also SPD and

$$(\mathcal{B}^{-1}v, v) = \inf_{\substack{\mathbf{v}\in\mathbf{V} \\ \mathbf{\Pi}\mathbf{v}=v}} (\mathbf{B}^{-1}\mathbf{v}, \mathbf{v}), \quad \forall v \in V.$$

Proof. It is clear that $(\mathcal{B}v, v) \geqslant 0$ for any $v \in V$ due to positive definiteness of $\mathbf{B}$. Furthermore, we have

$$0 = (\mathcal{B}v, v) = (\mathbf{B}\mathbf{\Pi}^T v, \mathbf{\Pi}^T v) \quad \Longrightarrow \quad \mathbf{\Pi}^T v = 0 \quad \Longrightarrow \quad v \in \operatorname{null}(\mathbf{\Pi}^T) = \operatorname{range}(\mathbf{\Pi})^{\perp}.$$

[2] In order to apply the convergence rate estimate Theorem 2.15 for stationary iterative methods to a symmetric positive semi-definite problem, we can restrict the domain of operator $\mathbf{A}$ inside the subspace, range($\mathbf{A}$). This way the operator $\mathbf{A}$ is still non-singular.

Since $\mathbf{\Pi}$ is surjective, we have $v = 0$. This proves the iterator $\mathcal{B}$ is SPD.

Define $\mathbf{v}_* := \mathbf{B}\mathbf{\Pi}^T\mathcal{B}^{-1}v$. It is easy to see that

$$\mathbf{\Pi}\mathbf{v}_* = \mathbf{\Pi}\mathbf{B}\mathbf{\Pi}^T\mathcal{B}^{-1}v = \mathcal{B}\mathcal{B}^{-1}v = v, \quad \forall v \in V,$$

and

$$(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{w}) = (\mathbf{\Pi}^T\mathcal{B}^{-1}v, \mathbf{w}) = (\mathcal{B}^{-1}v, \mathbf{\Pi}\mathbf{w}).$$

If $\mathbf{w} \in \text{null}(\mathbf{\Pi})$, then $(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{w}) = 0$. This ensures that, for any vector $\mathbf{v} \in \mathbf{V}$, there exists a $\mathbf{B}^{-1}$-orthogonal decomposition $\mathbf{v} = \mathbf{v}_* + \mathbf{w}$ with $\mathbf{w} \in \text{null}(\mathbf{\Pi})$. Hence, we get

$$(\mathbf{B}^{-1}\mathbf{v}, \mathbf{v}) = \big(\mathbf{B}^{-1}(\mathbf{v}_* + \mathbf{w}), \mathbf{v}_* + \mathbf{w}\big) = \big(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{v}_*\big) + \big(\mathbf{B}^{-1}\mathbf{w}, \mathbf{w}\big).$$

Thus

$$\begin{aligned}\inf_{\substack{\mathbf{v}\in\mathbf{V}\\ \mathbf{\Pi}\mathbf{v}=v}} (\mathbf{B}^{-1}\mathbf{v}, \mathbf{v}) &= \big(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{v}_*\big) + \inf_{\mathbf{w}\in\text{null}(\mathbf{\Pi})} \big(\mathbf{B}^{-1}\mathbf{w}, \mathbf{w}\big) \\ &= \big(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{v}_*\big) = \big(\mathbf{\Pi}^T\mathcal{B}^{-1}v, \mathbf{B}\mathbf{\Pi}^T\mathcal{B}^{-1}v\big) = \big(\mathcal{B}^{-1}v, v\big).\end{aligned}$$

Hence the result. □

Remark 4.12 (Minimizer for the expanded problem)**.** From the above proof, we can easily see $\mathbf{v}_* = \mathbf{B}\mathbf{\Pi}^T\mathcal{B}^{-1}v$ is actually the minimizer for $\inf_{\substack{\mathbf{v}\in\mathbf{V}\\ \mathbf{\Pi}\mathbf{v}=v}} (\mathbf{B}^{-1}\mathbf{v}, \mathbf{v})$. □

Remark 4.13 (Auxiliary space method)**.** The above lemma for relation between the expanded problem and the original problem can also be extended to the auxiliary space lemma: For two vector spaces V and $\tilde{V}$ and a surjective $\Pi : \tilde{V} \mapsto V$, if the iterator $\tilde{\mathcal{B}} : \tilde{V}' \mapsto \tilde{V}$ is SPD, then $\mathcal{B} = \Pi\,\tilde{\mathcal{B}}\,\Pi^T$ is also SPD and

$$(\mathcal{B}^{-1}v, v) = \inf_{\substack{\tilde{v}\in\tilde{V}\\ \Pi\tilde{v}=v}} (\tilde{\mathcal{B}}^{-1}\tilde{v}, \tilde{v}), \quad \forall v \in V.$$

See more discussion on this topic in §4.5. □

We can then derive the following expression for the inverse of the PSC preconditioner, which can be found in [116, 119, 65, 124].

Lemma 4.14. Assume that all $\mathcal{S}_j$'s are SPD. Then

$$(\mathcal{B}_{\text{PSC}}^{-1}v, v) = \inf_{\sum_j v_j = v} \sum_{j=1}^{J} (\mathcal{S}_j^{-1}v_j, v_j), \quad \forall\, v \in V.$$

The XZ identity

We now give the well-known XZ identity originally proved by Xu and Zikatanov [122] which gives the exact convergence rate of the SSC method.

Theorem 4.15 (XZ Identity)**.** Assume that $\mathcal{B}$ is defined by Algorithm 4.2 and, for $j = 1, \ldots, J$, $\mathbf{w}_j := \mathcal{A}_j \Pi_j \sum_{i \geqslant j} \mathbf{v}_i - \mathcal{S}_j^{-1} \mathbf{v}_j$. If $\mathcal{S}_j^{-T} + \mathcal{S}_j^{-1} - \mathcal{A}_j$ are SPD's for $j = 1, \ldots, J$, then

$$\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2 = 1 - \frac{1}{1 + c_0} = 1 - \frac{1}{c_1}, \tag{4.14}$$

where

$$c_0 = \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \|\mathcal{S}_j^T \mathbf{w}_j\|_{\overline{\mathcal{S}}_j^{-1}}^2 \tag{4.15}$$

and

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \left\| \overline{\mathcal{S}}_j \mathcal{S}_j^{-1} \mathbf{v}_j + \mathcal{S}_j^T \mathbf{w}_j \right\|_{\overline{\mathcal{S}}_j^{-1}}^2 . \tag{4.16}$$

Proof. (1) By applying Theorem 2.15 and Lemma 4.11, we know

$$\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2 = 1 - \left(\sup_{\|v\|_{\mathcal{A}}=1} (\overline{\mathcal{B}}^{-1} v, v) \right)^{-1} = 1 - \left(\sup_{\|v\|_{\mathcal{A}}=1} \inf_{\mathbf{\Pi}\mathbf{v}=v} (\overline{\mathbf{B}}^{-1} \mathbf{v}, \mathbf{v}) \right)^{-1}. \tag{4.17}$$

From (4.13), we have, for any $\mathbf{v} \in \mathbf{V}$, that

$$\left(\overline{\mathbf{B}}^{-1} \mathbf{v}, \mathbf{v}\right) = (\mathbf{A}\mathbf{v}, \mathbf{v}) + \left(\mathbf{K}^{-1}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v},\ (\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\right).$$

By simple calculation, we get

$$\begin{aligned} (\mathbf{D} + \mathbf{U})\,\mathbf{v} &= \Big(\sum_{j \geqslant 1} \mathcal{Q}_1 \mathcal{A} \mathcal{Q}_j^T \mathbf{v}_j,\ \sum_{j \geqslant 2} \mathcal{Q}_2 \mathcal{A} \mathcal{Q}_j^T \mathbf{v}_j,\ \cdots \Big)^T \\ &= \Big(\sum_{j \geqslant 1} \mathcal{A}_1 \Pi_1 \mathcal{I}_j \mathbf{v}_j,\ \sum_{j \geqslant 2} \mathcal{A}_2 \Pi_2 \mathcal{I}_j \mathbf{v}_j,\ \cdots \Big)^T \\ &= \Big(\mathcal{A}_1 \Pi_1 \sum_{j \geqslant 1} \mathbf{v}_j,\ \mathcal{A}_2 \Pi_2 \sum_{j \geqslant 2} \mathbf{v}_j,\ \cdots \Big)^T. \end{aligned}$$

Hence we can denote

$$\left(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1}\right) \mathbf{v} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_J)^T, \quad \text{with } \mathbf{w}_j := \mathcal{A}_j \Pi_j \sum_{i \geqslant j} \mathbf{v}_i - \mathcal{S}_j^{-1} \mathbf{v}_j.$$

Due to (4.12) and the fact that $\mathbf{K}$ is diagonal, we have

$$\left(\mathbf{K}^{-1}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v},\ (\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\right) = \sum_{j=1}^{J} \left(\mathcal{S}_j \overline{\mathcal{S}}_j^{-1} \mathcal{S}_j^T \mathbf{w}_j, \mathbf{w}_j\right) = \sum_{j=1}^{J} \left\| \mathcal{S}_j^T \mathbf{w}_j \right\|_{\overline{\mathcal{S}}_j^{-1}}^2,$$

where $\overline{\mathcal{S}}_j := \mathcal{S}_j^T + \mathcal{S}_j - \mathcal{S}_j^T \mathcal{A}_j \mathcal{S}_j$ is the symmetrization of $\mathcal{S}_j$. We then obtain, for any $v \in V$, that

$$\sup_{\|v\|_{\mathcal{A}}=1} \inf_{\mathbf{\Pi v}=v} \left(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}\right) = 1 + \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\mathbf{\Pi v}=v} \sum_{j=1}^{J} \left\|\mathcal{S}_j^T \mathbf{w}_j\right\|_{\overline{\mathcal{S}}_j^{-1}}^2.$$

This gives the desired estimate for the constant c_0.

(2) On the other hand, from (4.11), we have

$$\begin{aligned}\left(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}\right) &= \left(\mathbf{K}^{-1}(\mathbf{S}^{-T} + \mathbf{U})\mathbf{v},\ (\mathbf{S}^{-T} + \mathbf{U})\mathbf{v}\right) \\ &= \sum_{j=1}^{J} \left\|\left(\mathcal{S}_j^{-1} + \mathcal{S}_j^{-T} - \mathcal{A}_j\right)^{-\frac{1}{2}} \left(\mathcal{S}_j^{-T}\mathbf{v}_j + \sum_{i>j} \mathcal{Q}_j \mathcal{A} \mathcal{I}_i \mathbf{v}_i\right)\right\|^2. \end{aligned} \tag{4.18}$$

We notice that

$$\begin{aligned}\mathcal{S}_j^{-T}\mathbf{v}_j + \sum_{i>j} \mathcal{Q}_j \mathcal{A} \mathcal{I}_i \mathbf{v}_i = \mathcal{S}_j^{-T}\mathbf{v}_j + \mathcal{A}_j \mathit{\Pi}_j \sum_{i>j} \mathbf{v}_i &= \left(\mathcal{S}_j^{-T} + \mathcal{S}_j^{-1} - \mathcal{A}_j\right)\mathbf{v}_j + \mathbf{w}_j \\ &= \mathcal{S}_j^{-T}\overline{\mathcal{S}}_j \mathcal{S}_j^{-1}\mathbf{v}_j + \mathbf{w}_j = \mathcal{S}_j^{-T}\left(\overline{\mathcal{S}}_j \mathcal{S}_j^{-1}\mathbf{v}_j + \mathcal{S}_j^T \mathbf{w}_j\right).\end{aligned}$$

Plug this into the previous identity, we get

$$\left(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}\right) = \sum_{j=1}^{J} \left\|\overline{\mathcal{S}}_j \mathcal{S}_j^{-1}\mathbf{v}_j + \mathcal{S}_j^T \mathbf{w}_j\right\|_{\overline{\mathcal{S}}_j^{-1}}^2.$$

Hence the estimate for the constant c_1. □

Remark 4.16 (An equivalent form)**.** We have introduced operators $\mathcal{T}_j := \mathcal{S}_j \mathcal{A}_j : V_j \mapsto V_j$. Hence $\mathcal{T}_{\overline{\mathcal{S}}_j} := \overline{\mathcal{S}}_j \mathcal{A}_j = \mathcal{T}_j + \mathcal{T}_j^* - \mathcal{T}_j^* \mathcal{T}_j$ and we can rewrite the above estimate (4.16) in a slightly different form. Notice that, in (4.18),

$$\mathcal{S}_j^{-T}\mathbf{v}_j + \sum_{i>j} \mathcal{Q}_j \mathcal{A} \mathcal{I}_i \mathbf{v}_i = \mathcal{A}_j \left(\mathcal{S}_j^T \mathcal{A}_j\right)^{-1}\mathbf{v}_j + \mathcal{A}_j \mathit{\Pi}_j \sum_{i>j} \mathbf{v}_i = \mathcal{A}_j \Big[\left(\mathcal{T}_j^*\right)^{-1}\mathbf{v}_j + \mathit{\Pi}_j \sum_{i>j} \mathbf{v}_i\Big]$$

and

$$\left(\mathcal{S}_j^{-1} + \mathcal{S}_j^{-T} - \mathcal{A}_j\right)^{-1} \mathcal{A}_j = \left(\mathcal{T}_j^{-1} + (\mathcal{T}_j^*)^{-1} - \mathcal{I}_j\right)^{-1} = \mathcal{T}_j \mathcal{T}_{\overline{\mathcal{S}}_j}^{-1} \mathcal{T}_j^*.$$

Thus we have

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \left\|\mathcal{T}_{\overline{\mathcal{S}}_j}^{-\frac{1}{2}} \left(\mathbf{v}_j + \mathcal{T}_j^* \mathit{\Pi}_j \sum_{i>j} \mathbf{v}_i\right)\right\|_{\mathcal{A}}^2. \tag{4.19}$$

□

Example 4.17 (Linear stationary iterative method)**.** One-level linear stationary iterative method

$$u^{\text{new}} = u^{\text{old}} + \overline{\mathcal{S}}(f - \mathcal{A}u^{\text{old}}),$$

can be viewed as a special subspace correction method with only one subspace V. Hence, using (4.19), we immediately have

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1} \|\mathcal{T}_{\overline{\mathcal{S}}}^{-\frac{1}{2}} v\|_{\mathcal{A}}^2 = \sup_{\|v\|_{\mathcal{A}}=1} \left((\overline{\mathcal{S}}\mathcal{A})^{-1} v, v\right)_{\mathcal{A}} = \sup_{\|v\|_{\mathcal{A}}=1} \left(\overline{\mathcal{S}}^{-1} v, v\right),$$

which is exactly the convergence rate derived in Theorem 2.15. □

Example 4.18 (Twogrid method)**.** Theorem 3.37 can be viewed as a special case of the XZ identity in the case of space decomposition with two subspaces, i.e., $V = V_c + V$. Suppose we use $\mathcal{A}_c^{-1}$ and $\overline{\mathcal{S}}$ as subspace solvers, respectively. According to (4.19), we get

$$c_1 = \sup_{\|w\|_{\mathcal{A}}=1} \inf_{\substack{w=v_c+v \\ v_c\in V_c, v\in V}} \|v_c + \Pi_c v\|_{\mathcal{A}}^2 + \|(\overline{\mathcal{S}}\mathcal{A})^{-\frac{1}{2}} v\|_{\mathcal{A}}^2.$$

We can prove that

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1} \|\mathcal{T}_{\overline{\mathcal{S}}}^{-\frac{1}{2}} (\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}}) v\|_{\mathcal{A}}^2,$$

which is consistent with (3.33) in Theorem 3.37. For a complete proof of this result, we refer to Zikatanov [133]. □

When we solve each subspace problem exactly, the XZ identity is substantially simpler since $\mathcal{T}_j = \Pi_j$ in this case. This special case of the XZ identity is given in the following corollary.

Corollary 4.19 (SSC with exact subspace solvers)**.** If an exact subspace solver $\mathcal{S}_j = \mathcal{A}_j^{-1}$ for each subspace is used, then we have, in (4.14), that

$$c_0 = \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \Big\| \Pi_j \sum_{i>j} \mathbf{v}_i \Big\|_{\mathcal{A}_j}^2 \tag{4.20}$$

and

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \Big\| \Pi_j \sum_{i\geqslant j} \mathbf{v}_i \Big\|_{\mathcal{A}_j}^2. \tag{4.21}$$

4.4 Convergence analysis of PSC

In this section, we estimate the condition number of the PSC method. In general, PSC might not converge as an iterative method, but we can show that it is uniform convergent as a preconditioner under certain conditions.

Relating PSC to SSC

The following theorem shows the relation between the PSC and SSC methods.

Theorem 4.20 (PSC and SSC)**.** If $\mathcal{S}_j = \mathcal{A}_j^{-1}$ for all j and V_j are subspaces of V, then there exists a constant c_* depends only on topology of the overlaps between the subspaces such that

$$\frac{1}{4}(\mathcal{B}_{\mathrm{PSC}}^{-1}v, v) \leqslant (\overline{\mathcal{B}}_{\mathrm{SSC}}^{-1}v, v) \leqslant c_*(\mathcal{B}_{\mathrm{PSC}}^{-1}v, v), \quad \forall\, v \in V.$$

Proof. Given $v = \sum_{j=1}^J v_j$ with $v_j \in V_j$. It follows that

$$\|v\|_{\mathcal{A}}^2 = \sum_{k,j=1}^J (v_k, v_j)_{\mathcal{A}} = \sum_{k=1}^J (v_k, v_k)_{\mathcal{A}} + 2\sum_{j>k}^J (v_k, v_j)_{\mathcal{A}} = 2\sum_{j\geqslant k}^J (v_k, v_j)_{\mathcal{A}} - \sum_{k=1}^J (v_k, v_k)_{\mathcal{A}}.$$

Hence, since Π_k is a $\mathcal{A}$-projection, it follows that

$$\begin{aligned}\sum_{k=1}^J \|v_k\|_{\mathcal{A}}^2 &\leqslant 2\sum_{k=1}^J \Big(v_k, \sum_{j=k}^J v_j\Big)_{\mathcal{A}} = 2\sum_{k=1}^J \Big(v_k, \Pi_k \sum_{j=k}^J v_j\Big)_{\mathcal{A}} \\ &\leqslant 2\Big(\sum_{k=1}^J \|v_k\|_{\mathcal{A}}^2\Big)^{\frac{1}{2}}\Big(\sum_{k=1}^J \big\|\Pi_k \sum_{j=k}^J v_j\big\|_{\mathcal{A}}^2\Big)^{\frac{1}{2}}.\end{aligned}$$

In turn, it gives $\sum_{k=1}^J \|v_k\|_{\mathcal{A}}^2 \leqslant 4\sum_{k=1}^J \big\|\Pi_k \sum_{j=k}^J v_j\big\|_{\mathcal{A}}^2$. Together with Lemma 4.14, Corollary 4.19, and (4.17), it gives the first inequality. The second one is also easy; see HW 4.5. □

This shows that, if the SSC method works well as an iterative method, then the PSC method based on the same space decomposition also works as a preconditioner. Next, we give direct analysis of the condition number of the PSC method.

Condition number of PSC

To obtain estimates on the condition number of the preconditioned problems, we first give the following assumptions:

Assumption 4.21 (Convergence assumptions for MSC)**.** We assume that

1. For any $v \in V$, there exists a decomposition $v = \sum_{j=1}^J v_j$ with $v_j \in V_j$ such that

$$\sum_{j=1}^J (\mathcal{S}_j^{-1}v_j, v_j) \leqslant K_1(\mathcal{A}v, v); \tag{4.22}$$

2. For any $u, v \in V$,

$$\sum_{(i,j)} (\mathcal{T}_i u, \mathcal{T}_j v)_{\mathcal{A}} \leqslant K_2\Big(\sum_{i=1}^J (\mathcal{T}_i u, u)_{\mathcal{A}}\Big)^{\frac{1}{2}}\Big(\sum_{j=1}^J (\mathcal{T}_j v, v)_{\mathcal{A}}\Big)^{\frac{1}{2}}. \tag{4.23}$$

Theorem 4.22 (Condition number of PSC)**.** If Assumption 4.21 holds true, the PSC method (4.7) satisfies

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant K_1K_2.$$

Proof. (1) For any $v \in V$, suppose that $v = \sum_{j=1}^{J} v_j$ is a decomposition, which satisfies the first condition of Assumption 4.21. It is easy to see that

$$\begin{aligned}(v,v)_{\mathcal{A}} &= \sum_{j=1}^{J}(v_j,v)_{\mathcal{A}} = \sum_{j=1}^{J}(v_j,\Pi_j v)_{\mathcal{A}} = \sum_{j=1}^{J}(v_j,\mathcal{A}_j\Pi_j v) = \sum_{j=1}^{J}\left(\mathcal{S}_j^{-\frac{1}{2}}v_j,\mathcal{S}_j^{\frac{1}{2}}\mathcal{A}_j\Pi_j v\right)\\ &\leqslant \sum_{j=1}^{J}\left(\mathcal{S}_j^{-1}v_j,v_j\right)^{\frac{1}{2}}\left(\mathcal{S}_j\mathcal{A}_j\Pi_j v,\mathcal{A}_j\Pi_j v\right)^{\frac{1}{2}} = \sum_{j=1}^{J}\left(\mathcal{S}_j^{-1}v_j,v_j\right)^{\frac{1}{2}}\left(\mathcal{S}_j\mathcal{A}_j\Pi_j v,v\right)_{\mathcal{A}}^{\frac{1}{2}}\\ &\leqslant \left(\sum_{j=1}^{J}\left(\mathcal{S}_j^{-1}v_j,v_j\right)\right)^{\frac{1}{2}}\left(\sum_{j=1}^{J}\left(\mathcal{T}_j v,v\right)_{\mathcal{A}}\right)^{\frac{1}{2}} \leqslant \sqrt{K_1}\,\|v\|_{\mathcal{A}}\left(\mathcal{B}\mathcal{A}v,v\right)_{\mathcal{A}}^{\frac{1}{2}}.\end{aligned}$$

Consequently, we have the lower bound

$$\frac{1}{K_1}(v,v)_{\mathcal{A}} \leqslant (\mathcal{B}\mathcal{A}v,v)_{\mathcal{A}}, \quad \forall\, v \in V.$$

(2) From the second assumption, we have

$$\|\mathcal{B}\mathcal{A}v\|_{\mathcal{A}}^2 = \sum_{i,j=1}^{J}\left(\mathcal{T}_i v,\mathcal{T}_j v\right)_{\mathcal{A}} \leqslant K_2(\mathcal{B}\mathcal{A}v,v)_{\mathcal{A}} \leqslant K_2\|\mathcal{B}\mathcal{A}v\|_{\mathcal{A}}\|v\|_{\mathcal{A}}.$$

So we obtain the upper bound

$$(\mathcal{B}\mathcal{A}v,v)_{\mathcal{A}} \leqslant K_2(v,v)_{\mathcal{A}}, \quad \forall\, v \in V.$$

Thus Lemmas 2.29 and 2.30 yield the desired estimate. □

According to Theorem 4.22, if we can find a space decomposition and corresponding smoothers with uniform constants K_1 and K_2, then we are able to construct a uniformly convergent preconditioner using the PSC framework. Similar results can obtained for SSC as well.

Remark 4.23 (Similar estimate for SSC)**.** In fact, with the same assumptions (Assumption 4.21), we can also show that the SSC method also converges with

$$\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2 \leqslant 1 - \frac{2-\omega_1}{K_1(1+K_2)^2} \quad \text{and} \quad \omega_1 := \max_j \rho(\mathcal{S}_j\mathcal{A}_j) = \max_j \rho(\mathcal{T}_j).$$

Because a sharp result has been given in §4.3, we will just leave the proof to the readers (cf., for example, [118]). □

Estimates of K_1 and K_2

Assumption 4.21 is not easy to verify directly. So we now give a few useful estimates for the constants in these conditions. We first give a straight-forward estimate of K_1, which clearly separates the condition on space decomposition part and smoother part.

Lemma 4.24 (Estimates of K_1)**.** Assume that, for any $v \in V$, there is a decomposition $v = \sum_{j=1}^{J} v_j$ with $v_j \in V_j$:

(i) If the decomposition satisfies that

$$\sum_{j=1}^{J} (v_j, v_j)_{\mathcal{A}} \leqslant C_1 (v, v)_{\mathcal{A}},$$

then we have

$$K_1 \leqslant C_1/\omega_0, \quad \text{where } \omega_0 := \min_{j=1,\ldots,J} \big\{ \lambda_{\min}(\mathcal{S}_j \mathcal{A}_j) \big\};$$

(ii) If $\rho_j := \rho(\mathcal{A}_j)$ and

$$\sum_{j=1}^{J} \rho_j (v_j, v_j) \leqslant \hat{C}_1 (v, v)_{\mathcal{A}},$$

then we have

$$K_1 \leqslant \hat{C}_1/\hat{\omega}_0, \quad \text{where } \hat{\omega}_0 := \min_{j=1,\ldots,J} \big\{ \rho_j \lambda_{\min}(\mathcal{S}_j) \big\}.$$

Proof. (i) By the definition of ω_0 and the fact that $\lambda(\mathcal{S}_j^{1/2} \mathcal{A}_j \mathcal{S}_j^{1/2}) = \lambda(\mathcal{S}_j \mathcal{A}_j)$, we have

$$(\mathcal{S}_j^{1/2} \mathcal{A}_j \mathcal{S}_j^{1/2} \mathcal{S}_j^{-1/2} v_j, \mathcal{S}_j^{-1/2} v_j) \geqslant \omega_0 (\mathcal{S}_j^{-1} v_j, v_j), \quad j = 1, \ldots, J.$$

Note that

$$\sum_{j=1}^{J} (\mathcal{S}_j^{1/2} \mathcal{A}_j \mathcal{S}_j^{1/2} \mathcal{S}_j^{-1/2} v_j, \mathcal{S}_j^{-1/2} v_j) = \sum_{j=1}^{J} (\mathcal{A}_j v_j, v_j) = \sum_{j=1}^{J} (v_j, v_j)_{\mathcal{A}} \leqslant C_1 (v, v)_{\mathcal{A}}.$$

We then have

$$\omega_0 \sum_{j=1}^{J} (\mathcal{S}_j^{-1} v_j, v_j) \leqslant C_1 (\mathcal{A} v, v) \quad \text{or} \quad \sum_{j=1}^{J} (\mathcal{S}_j^{-1} v_j, v_j) \leqslant \frac{C_1}{\omega_0} (\mathcal{A} v, v),$$

which implies that $K_1 \leqslant \frac{C_1}{\omega_0}$.

(ii) Similar to the previous part, from the definition of $\hat{\omega}_0$, we have

$$\rho_j (v_j, v_j) = \rho_j (\mathcal{S}_j \mathcal{S}_j^{-1/2} v_j, \mathcal{S}_j^{-1/2} v_j) \geqslant \hat{\omega}_0 (\mathcal{S}_j^{-1} v_j, v_j), \quad j = 1, \ldots, J.$$

Hence, we have

$$\hat{\omega}_0 \sum_{j=1}^{J} (\mathcal{S}_j^{-1} v_j, v_j) \leqslant \sum_{j=1}^{J} \rho_j (v_j, v_j) \leqslant \hat{C}_1 (v, v)_{\mathcal{A}},$$

which implies that $K_1 \leqslant \frac{\hat{C}_1}{\hat{\omega}_0}$. □

We introduce a nonnegative symmetric matrix

$$\Sigma = \big(\sigma_{i,j}\big) \in \mathbb{R}^{J\times J}, \tag{4.24}$$

where each entry $\sigma_{i,j}$ is the smallest constant such that

$$\big(\mathcal{T}_i u, \mathcal{T}_j v\big)_{\mathcal{A}} \leqslant \omega_1 \sigma_{i,j} \big(\mathcal{T}_i u, u\big)_{\mathcal{A}}^{\frac{1}{2}} \big(\mathcal{T}_j v, v\big)_{\mathcal{A}}^{\frac{1}{2}}, \quad \forall\, u, v \in V. \tag{4.25}$$

It is clear that $0 \leqslant \sigma_{i,j} \leqslant 1$. Moreover, $\sigma_{i,j} = 0$, if $\Pi_i \Pi_j = 0$.

Lemma 4.25 (Estimate of K_2)**.** The constant $K_2 \leqslant \omega_1 \rho(\Sigma)$. Furthermore, if $\sigma_{i,j} \lesssim \gamma^{|i-j|}$ holds for some parameter $0 < \gamma < 1$, then $\rho(\Sigma) \lesssim (1-\gamma)^{-1}$; in this case, the inequality (4.23) is the well-known *strengthened Cauchy-Schwarz inequality*.

Proof. From the definition of Σ as in (4.24), it is immediately clear that $K_2 \leqslant \omega_1 \rho(\Sigma)$. Furthermore, because the matrix Σ is a real symmetric matrix and $\rho(\Sigma) \leqslant \max_{j=1,\dots,J} \sum_{i=1}^{J} \sigma_{i,j}$, we have

$$\rho(\Sigma) \leqslant \max_{1\leqslant j\leqslant J} \sum_{i=1}^{J} \sigma_{i,j} \lesssim \sum_{i=1}^{J} \gamma^{i-1} \leqslant \frac{1}{1-\gamma}.$$

Hence the result. □

4.5 Auxiliary space method ⋆

Sometimes, we cannot apply subspace correction methods directly due to difficulties in obtaining an appropriate space decomposition. In this case, we can introduce an auxiliary or fictitious space $\tilde{V}$ for assistance. Suppose $\Pi : \tilde{V} \mapsto V$ is surjective and satisfies the following two conditions:

- Firstly,
$$\|\Pi\tilde{v}\|_{\mathcal{A}} \leqslant \mu_1 \|\tilde{v}\|_{\tilde{\mathcal{A}}}, \qquad \forall\, \tilde{v} \in \tilde{V}.$$
- Secondly, for any $v \in V$, there exists $\tilde{v} \in \tilde{V}$ such that $\Pi\tilde{v} = v$ and
$$\mu_0 \|\tilde{v}\|_{\tilde{\mathcal{A}}} \leqslant \|v\|_{\mathcal{A}}, \qquad \forall\, \tilde{v} \in \tilde{V}.$$

Under the above assumptions, if $\tilde{\mathcal{B}}$ is a SPD preconditioner for $\tilde{\mathcal{A}}$, then $\mathcal{B} = \Pi\tilde{\mathcal{B}}\Pi^T$ is SPD and

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant \left(\frac{\mu_1}{\mu_0}\right)^2 \kappa(\tilde{\mathcal{B}}\tilde{\mathcal{A}}).$$

This suggests that we can construct a subspace correction method on $\tilde{V}$ instead of the original space V. This result is also known as the *Fictitious Space Lemma* or the *Fictitious Domain Lemma*; see [91, 120].

The fictitious domain method is a large class of methods which is usually for problems in geometrically complex, and most likely moving, domains. By embedding the original physical domain in a larger artificial domain, we can discretize the partial differential equations on a more structured grid and, hence, solve the resulting linear algebraic systems more quickly. Of course, the boundary conditions have to be handled with great care; see [63] for details.

4.6 Homework problems

HW 4.1. Prove the statements in Remark 4.5.

HW 4.2. If $\mathcal{S}_j$ $(j = 1, \ldots, J)$ are all SPD, then the preconditioner $\mathcal{B} = \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j$ is also SPD.

HW 4.3. Show that the block G-S method for the expanded system is just the SSC method for the original problem.

HW 4.4. Prove Theorem 4.10.

HW 4.5. Prove Theorem 4.20. What is the constant c_*?

Part II

Examples of Multilevel Iterative Methods

Chapter 5

Subspace Correction Preconditioners

In Chapter 4, we have discussed stationary iterative methods in the framework of method of subspace correction (MSC). In this chapter, we give a few examples of multilevel methods and their convergence analysis based on the framework of subspace corrections.

5.1 Two-level overlapping DDM

In this section, we will investigate the two-level overlapping domain decomposition method (DDM) presented in §2.4 using the MSC framework.

Two-level space decomposition

Based on the previous discussions, it is now easy to understand that the additive and multiplicative Schwarz domain decomposition methods can be considered as PSC and SSC, respectively. For proof-of-concept, we use the Poisson's equation on Ω as an example. In this case, $\mathscr{V} = H_0^1(\Omega)$, $\Omega = \bigcup_{j=1}^J \Omega_j$, and $\mathscr{V}_j := \{v \in \mathscr{V} : \operatorname{supp} v \subset \hat{\Omega}_j\} \subset \mathscr{V}$; see Figure 2.2. Suppose we have a finite-dimensional coarse space $V_0 \subset \mathscr{V}$ on a quasi-uniform mesh of meshsize $H = \operatorname{diam}(\Omega_j)$. Apparently, this way, we have a space decomposition

$$\mathscr{V} = V_0 + \mathscr{V}_1 + \cdots + \mathscr{V}_J.$$

The SSC method based on this space decomposition with exact subspace solvers on each subdomain as well as on the coarse space gives an abstract multiplicative Schwarz DDM method[1].

We first define a partition of unity function $\theta_j \in C^1(\overline{\Omega})$ $(j = 1, \ldots, J)$ such that

(1) $0 \leqslant \theta_j \leqslant 1$ and $\sum_{j=1}^J \theta_j = 1$;

[1] It is an abstract algorithm because we did not discretize each sub-domain problems.

(2) $\operatorname{supp}\theta_j \subset \hat{\Omega}_j$;

(3) $\max|\nabla\theta_j| \leqslant C_\beta/H$, where C_β depends on the relative overlap size β.

This way, for any function $v \in \mathscr{V}$, we can define a decomposition

$$v = v_0 + v_1 + \cdots + v_J,$$

where

$$v_0 \in V_0 \quad \text{and} \quad v_j := \theta_j(v - v_0) \in \mathscr{V}_j, \ \ j = 1, \ldots, J.$$

Convergence analysis of DDM

Based on the above decomposition, we have $\sum_{j=1}^J v_j = v - v_0$ and

$$\sum_{j=0}^J \Big|\Pi_j \sum_{i=j+1}^J v_i\Big|_1^2 = \sum_{j=0}^J \Big|\Pi_j \sum_{i=j+1}^J \theta_i(v-v_0)\Big|_1^2 = \Big|\Pi_0(v-v_0)\Big|_1^2 + \sum_{j=1}^J \Big|\Pi_j \sum_{i=j+1}^J \theta_i(v-v_0)\Big|_1^2.$$

Since Π_j's : $\mathscr{V} \mapsto \mathscr{V}_j$ $(j = 1, \ldots, J)$ are $\mathcal{A}$-projections, it is easy to see that $|\Pi_j w|_1 \leqslant |w|_1$. Furthermore,

$$\begin{aligned}
\Big|\Pi_j \sum_{i=j+1}^J \theta_i(v-v_0)\Big|_1^2 &= \Big|\Pi_j \sum_{i=j+1}^J \theta_i(v-v_0)\Big|_{1,\hat{\Omega}_j}^2 \leqslant \Big|\sum_{i=j+1}^J \theta_i(v-v_0)\Big|_{1,\hat{\Omega}_j}^2 \\
&\leqslant \Big\|\big(\sum_{i>j}\theta_i\big)\nabla(v-v_0)\Big\|_{0,\hat{\Omega}_j}^2 + \Big\|\nabla\big(\sum_{i>j}\theta_i\big)(v-v_0)\Big\|_{0,\hat{\Omega}_j}^2 \\
&\leqslant |v-v_0|_{1,\hat{\Omega}_j}^2 + C_\beta^2 H^{-2}\|v-v_0\|_{0,\hat{\Omega}_j}^2.
\end{aligned}$$

By summing up all the terms, we have

$$\begin{aligned}
\sum_{j=0}^J \Big|\Pi_j \sum_{i=j+1}^J v_i\Big|_1^2 &\leqslant |v-v_0|_1^2 + \sum_{j=1}^J |v-v_0|_{1,\hat{\Omega}_j}^2 + C_\beta^2 H^{-2}\sum_{j=1}^J \|v-v_0\|_{0,\hat{\Omega}_j}^2 \\
&\lesssim |v-v_0|_1^2 + C_\beta^2 H^{-2}\|v-v_0\|_0^2,
\end{aligned}$$

where the constant in the last inequality depends on the maximal number of overlaps in domain decomposition. Because v_0 could be any function in V_0, in view of Proposition 3.14 or the so-called simultaneous estimate in Remark 3.15, we can obtain

$$\sum_{j=0}^J \Big|\Pi_j \sum_{i=j+1}^J v_i\Big|_1^2 \lesssim |v|_1^2.$$

Using the X-Z identity (Corollary 4.19), we can get the following uniform convergence result.

Proposition 5.1 (Uniform convergence of two-level DDM)**.** The abstract domain decomposition method with coarse space correction converges uniformly.

We leave the full proof to the interested readers; see HW 5.16.

Remark 5.2 (DDM without coarse space)**.** From the above analysis, we immediately see the importance of having the coarse space V_0. In fact, with a similar proof, one can show that the convergence rate depends on H^{-2} if not applying the coarse space correction. □

5.2 HB preconditioner

In the previous section, we have seen a two-level domain decomposition method in the setting of subspace correction. Now we investigate a multilevel example.

Nested space decomposition

We consider the Poisson's equation on a sequence of nested meshes $\mathcal{M}_l$ $(l = 0, \ldots, L)$ generated from an initial mesh $\mathcal{M}_0$ by uniform regular refinements. Hence meshsize h_l of $\mathcal{M}_l$ is proportional to γ^{2l} with $\gamma \in (0, 1)$. For example, in Figure 1.5, there is a hierarchy of grids with $h_l = (1/2)^{l+1}$ $(l = 0, 1, \ldots, L)$. Clearly,

$$h_0 > h_1 > h_2 > \cdots > h_L =: h.$$

Define continuous piecewise linear finite element spaces on the mesh $\mathcal{M}_l$ as

$$V_l := \big\{ v \in \mathscr{V} \,:\, v|_\tau \in \mathcal{P}_1(\tau),\ \forall\, \tau \in \mathcal{M}_l \big\}. \tag{5.1}$$

This way, we build a nested subspaces

$$V_0 \subset V_1 \subset \cdots \subset V_L =: V \subset \mathscr{V} = H_0^1(\Omega).$$

The set of interior grid points on the l-th level is denoted as $x_{l,i} \in \mathring{G}(\mathcal{M}_l)$ $(i = 1, \ldots, n_l)$. The subspace V_l is assigned with a nodal basis $\{\phi_{l,i}\}_{i=1}^{n_l}$, where $n_l := \big|\mathring{G}(\mathcal{M}_l)\big|$. The space V_l can be further decomposed as the sum of the one-dimensional subspaces spanned with the nodal basis $V_{l,i} := \operatorname{span}\{\phi_{l,i}\}$ $(i = 1, \ldots, n_l)$.

We then define

$$W_l := \big\{ v \in V_l \,:\, v(x) = 0,\ \forall\, x \in \mathring{G}(\mathcal{M}_{l-1}) \big\} \tag{5.2}$$

and obtain a multilevel space decomposition

$$V = W_0 \oplus W_1 \oplus \cdots \oplus W_L. \tag{5.3}$$

Let $\mathcal{J}_l : V \mapsto V_l$ be the cannonical interpolation operator and define $\mathcal{J}_{-1} := 0$. It is easy to see that

$$W_l = (\mathcal{J}_l - \mathcal{J}_{l-1})V = (\mathcal{I} - \mathcal{J}_{l-1})V_l, \quad l = 0, \ldots, L.$$

For level $l = 0, \ldots, L$, we define a nodal basis function

$$\psi_{l,i}(x) = \phi_{l,i}(x), \quad \text{for } x_{l,i} \in \mathring{G}(\mathcal{M}_l)\backslash\mathring{G}(\mathcal{M}_{l-1}) \text{ and } i = 1, \ldots, m_l := n_l - n_{l-1}.$$

Apparently, $\sum_{l=0}^{L} m_l = n_L = N$. This basis

$$\{\psi_{l,i}(x) \,:\, i = 1, \ldots, m_l, \;\; l = 0, \ldots, L\} \tag{5.4}$$

is the so-called *hierarchical basis*.

Notice that the decomposition (5.3) is a direct sum and there is no redundancy in this decomposition at all.

Telescope expansions

Using notations in Definition 4.1, we have

$$\begin{cases} \mathcal{A}_l : V_l \mapsto V_l & (\mathcal{A}_l u_l, v_l) = a[u_l, v_l], \quad \forall\, u_l, v_l \in V_l; \\ \mathcal{Q}_l : L^2 \mapsto V_l & (\mathcal{Q}_l u, v_l) = (u, v_l), \quad \forall\, v_l \in V_l; \\ \mathit{\Pi}_l : \mathscr{V} \mapsto V_l & (\mathit{\Pi}_l u, v_l) = a[u, v_l], \quad \forall\, v_l \in V_l. \end{cases} \tag{5.5}$$

We introduce a new notation $i \wedge j := \min(i, j)$. It is trivial to see

$$\mathcal{Q}_i \mathcal{Q}_j = \mathcal{Q}_{i\wedge j}, \quad \mathit{\Pi}_i \mathit{\Pi}_j = \mathit{\Pi}_{i\wedge j}, \tag{5.6}$$

and

$$(\mathcal{Q}_i - \mathcal{Q}_{i-1})(\mathcal{Q}_j - \mathcal{Q}_{j-1}) = (\mathit{\Pi}_i - \mathit{\Pi}_{i-1})(\mathit{\Pi}_j - \mathit{\Pi}_{j-1}) = 0, \quad \forall\, i \neq j. \tag{5.7}$$

If we define $\mathcal{Q}_{-1} = \mathit{\Pi}_{-1} = 0$, we have the following possible decompositions

$$v = \sum_{l=0}^{L} (\mathcal{Q}_l - \mathcal{Q}_{l-1})v = \sum_{l=0}^{L} (\mathit{\Pi}_l - \mathit{\Pi}_{l-1})v. \tag{5.8}$$

Hierarchical basis preconditioner

We now use the Richardson iteration discussed in §3.3 as the subspace solver, i.e.,

$$\mathcal{S}_{l,i}\mathcal{Q}_{l,i}v = h_l^{2-d}\big(\mathcal{Q}_{l,i}v, \psi_{l,i}\big)\,\psi_{l,i} = h_l^{2-d}\big(v, \psi_{l,i}\big)\,\psi_{l,i}.$$

The PSC method based on the space decomposition (5.3) can then be written

$$\mathcal{B}_{\mathrm{HB}} r = \sum_{j=1}^{N} \mathcal{S}_j \mathcal{Q}_j r = \sum_{l=0}^{L} \left(h_l^{2-d} \sum_{i=1}^{m_l} (r, \psi_{l,i}) \psi_{l,i} \right). \tag{5.9}$$

And this is the explicit form of the well-known *hierarchical basis* (HB) *preconditioner* proposed by Yserentant [129].

We shall now analyze this preconditioner in the framework of PSC in §4.4. In order to do that, we need a few important estimates.

Lemma 5.3 (H^1-stability of interpolation)**.** We have

$$\|(\mathcal{J}_l - \mathcal{J}_{l-1})v\|_0^2 + h_l^2 |\mathcal{J}_l v|_1^2 \lesssim c_d(l) h_l^2 |v|_1^2, \quad \forall v \in V,$$

where $c_1(l) \equiv 1$, $c_2(l) = L - l$, and $c_3(l) = \gamma^{-2(L-l)}$.

Proof. Using Proposition 3.11, we have

$$\|(\mathcal{J}_l - \mathcal{J}_{l-1})v\|_0 = \|\mathcal{J}_l v - \mathcal{J}_{l-1}\mathcal{J}_l v\|_0 \lesssim h_l |\mathcal{J}_l v|_1.$$

Let $\tau \in \mathcal{M}_l$ and $v_\tau := |\tau|^{-1} \int_\tau v \, dx$ be the average of v on τ. Using the standard scaling argument for $|\cdot|_{1,\tau}$, the discrete Sobolev inequality Proposition 3.13, and the Poincaré inequality Proposition 1.10, we can obtain that

$$|\mathcal{J}_l v|_{1,\tau} = |\mathcal{J}_l v - v_\tau|_{1,\tau} \lesssim \|\mathcal{J}_l v - v_\tau\|_{\infty,\tau} \leqslant \|v - v_\tau\|_{\infty,\tau} \lesssim C_d \|v - v_\tau\|_{1,\tau} \lesssim C_d |v|_{1,\tau}.$$

Hence the desired result follows by summing up terms on all elements in $\mathcal{M}_l$. □

Remark 5.4 (Condition number in hierarchical basis)**.** The above lemma suggests that, if $v \in W_l$ for any $0 \leqslant l \leqslant L$, we have

$$c_d^{-1}(l) h_l^{-2}(v, v) \lesssim a[v, v].$$

Compare this with the general Poincaré inequality in Proposition 1.11. Furthermore, from the inverse inequality Proposition 3.12, we always have

$$a[v, v] = |v|_1^2 \lesssim h_l^{-2} \|v\|_0^2 = h_l^{-2}(v, v).$$

Hence the operator $\mathcal{A}_l$ is "well-conditioned" up to a constant $c_d(l)$; compare this property with the standard Lagrange finite element basis case in Remark 3.16. □

Strengthened Cauchy-Schwarz inequality

Lemma 5.5 (Inner product between two levels)**.** If $i \leqslant j$, we have

$$a[u,v] \lesssim \gamma^{j-i} h_j^{-1} |u|_1 \|v\|_0, \quad \forall\, u \in V_i, v \in V_j.$$

Proof. We first restrict our attention to an element $\tau_i \in \mathcal{M}_i$. For $v \in \mathcal{M}_j$, there is a unique function $v_1 \in V$, such that v_1 vanishes on $\partial\tau_i$ and equals to v at all other grid points. Let $v_0 := v - v_1$. Because $u \in W_i$ is a linear function on τ_i, we have $\int_{\tau_i} \nabla u \nabla v_1 = 0$.

Define $T := \bigcup_{\tau_j \in \mathcal{M}_j, \overline{\tau}_j \bigcap \partial\tau_i \neq \emptyset} \tau_j$. Then $|T| \cong \left(\frac{h_i}{h_j}\right)^{d-1} h_j^d = h_i^{d-1} h_j$ and $\operatorname{supp} v_0 \subset \overline{T}$. We have

$$\|\nabla v_0\|_{0,\tau_i}^2 \lesssim \sum_{x \in \mathring{G}(\mathcal{M}_j) \bigcap \partial\tau_i} h_j^d h_j^{-2}\, v_0^2(x) = \sum_{x \in \mathring{G}(\mathcal{M}_j) \bigcap \partial\tau_i} h_j^{d-2}\, v^2(x) \lesssim h_j^{-2} \|v\|_{0,\tau_i}^2.$$

Since ∇u is a constant on τ_i, we have

$$\|\nabla u\|_{0,T} = \frac{|T|^{1/2}}{|\tau_i|^{1/2}} \|\nabla u\|_{0,\tau_i} \lesssim \left(\frac{h_i^{d-1} h_j}{h_i^d}\right)^{1/2} \|\nabla u\|_{0,\tau_i} \lesssim \gamma^{j-i} |u|_{1,\tau_i}.$$

Combining the above two inequalities, we have

$$\int_{\tau_i} \nabla u \cdot \nabla v = \int_{\tau_i} \nabla u \cdot \nabla v_0 \lesssim \gamma^{j-i} h_j^{-1} |u|_{1,\tau_i} \|v\|_{0,\tau_i}, \quad \forall \tau_i \in \mathcal{M}_i.$$

By the Cauchy-Schwarz inequality, we obtain the estimate:

$$\begin{aligned} a[u,v] &= \sum_{\tau_i \in \mathcal{M}_i} \int_{\tau_i} \nabla u \cdot \nabla v \lesssim \gamma^{j-i} h_j^{-1} \sum_{\tau_i \in \mathcal{M}_i} |u|_{1,\tau_i} \|v\|_{0,\tau_i} \\ &\leqslant \gamma^{j-i} h_j^{-1} \Big(\sum_{\tau_i \in \mathcal{M}_i} |u|_{1,\tau_i}^2 \Big)^{1/2} \Big(\sum_{\tau_i \in \mathcal{M}_i} \|v\|_{0,\tau_i}^2 \Big)^{1/2} = \gamma^{j-i} h_j^{-1} |u|_1 \|v\|_0. \end{aligned}$$

Hence the result. □

Lemma 5.6 (Strengthened Cauchy-Schwarz inequality for interpolations)**.** If $u, v \in V$, let $u_i := (\mathcal{J}_i - \mathcal{J}_{i-1})u$, and $v_j := (\mathcal{J}_j - \mathcal{J}_{j-1})v$, then we have

$$a[u_i, v_j] \lesssim \gamma^{|i-j|} \|u_i\|_{\mathcal{A}} \|v_j\|_{\mathcal{A}}.$$

Proof. If $j \geqslant i$, we have $v_j = v_j - \mathcal{J}_{j-1} v_j$. So $\|v_j\|_0 = \|v_j - \mathcal{J}_{j-1} v_j\|_0 \lesssim h_j \|v_j\|_{\mathcal{A}}$ follows from Proposition 3.11. If $i \geqslant j$, we can argue in a similar way. Hence the result follows directly from Lemma 5.5. □

Lemma 5.7 (Estimating K_2)**.** Assume that $\mathcal{T}_j = \mathcal{S}_j \mathcal{A}_j \Pi_j$ and the subspace smoother $\mathcal{S}_j : V_j \mapsto V_j$ satisfies

$$\|\mathcal{S}_j \mathcal{A}_j v\|_0^2 \lesssim \rho_j^{-1} (\mathcal{A}_j v, v), \quad \forall\, v \in V_j,$$

where $\rho_j := \rho(\mathcal{A}_j)$. Then, if $i < j$, we have

$$(u_i, \mathcal{T}_j v)_{\mathcal{A}} \lesssim \gamma^{j-i} \|u_i\|_{\mathcal{A}} \|v\|_{\mathcal{A}}, \quad \forall\, u_i \in V_i, v \in V. \tag{5.10}$$

For $0 \leqslant i, j \leqslant L$, we have the strengthened Cauchy-Schwarz inequality

$$(\mathcal{T}_i u, \mathcal{T}_j v)_{\mathcal{A}} \lesssim \gamma^{|j-i|/2} (\mathcal{T}_i u, u)_{\mathcal{A}}^{\frac{1}{2}} (\mathcal{T}_j v, v)_{\mathcal{A}}^{\frac{1}{2}}, \quad \forall\, u, v \in V. \tag{5.11}$$

Proof. By applying Lemma 5.5, we get

$$(u_i, \mathcal{T}_j v)_{\mathcal{A}} = a[u_i, \mathcal{T}_j v] \lesssim \gamma^{j-i} h_j^{-1} \|u_i\|_{\mathcal{A}} \|\mathcal{T}_j v\|_0.$$

Furthermore, we have

$$\|\mathcal{T}_j v\|_0 = \|\mathcal{S}_j \mathcal{A}_j \Pi_j v\|_0 \lesssim h_j \|\mathcal{A}_j^{1/2} \Pi_j v\|_0 \leqslant h_j \|\Pi_j v\|_{\mathcal{A}} \leqslant h_j \|v\|_{\mathcal{A}}.$$

This proves the first inequality (5.10).

First consider the case when $j \geqslant i$. By the Cauchy-Schwarz inequality and the inequality (5.10), we get

$$(\mathcal{T}_i u, \mathcal{T}_j v)_{\mathcal{A}} \leqslant (\mathcal{T}_j \mathcal{T}_i u, \mathcal{T}_i u)_{\mathcal{A}}^{\frac{1}{2}} (\mathcal{T}_j v, v)_{\mathcal{A}}^{\frac{1}{2}} \lesssim \gamma^{(j-i)/2} \|\mathcal{T}_i u\|_{\mathcal{A}} (\mathcal{T}_j v, v)_{\mathcal{A}}^{\frac{1}{2}}.$$

Also observe that $(\mathcal{T}_i u, \mathcal{T}_i u)_{\mathcal{A}} \lesssim \|\mathcal{T}_i u\|_{\mathcal{A}} (\mathcal{T}_i u, u)_{\mathcal{A}}^{\frac{1}{2}}$ and the second inequality (5.11) follows immediately. □

Convergence analysis of HB preconditioner

Theorem 5.8 (Convergence of HB preconditioner)**.** The multilevel PSC preconditioner $\mathcal{B}_{\mathrm{HB}}$ defined in (5.9) satisfies

$$\kappa(\mathcal{B}_{\mathrm{HB}} \mathcal{A}) \lesssim C_d(h),$$

where $C_1(h) \equiv 1$, $C_2(h) = |\log h|^2$, and $C_3(h) = h^{-1}$.

Proof. We choose a decomposition $v = \sum_{l=0}^{L} v_l := \sum_{l=0}^{L} (\mathcal{J}_l - \mathcal{J}_{l-1}) v$, where $\mathcal{J}_{-1} = 0$. With careful calculations, Proposition 3.12 and Lemma 5.3 ($\mathcal{J}_l = \Pi_l$ in 1D) yield

$$\sum_{l=0}^{L} \|v_l\|_{\mathcal{A}}^2 \lesssim \sum_{l=0}^{L} h_l^{-2} \|v_l\|_0^2 \cong \sum_{l=0}^{L} \rho_l \|v_l\|_0^2 \lesssim C_d(h) \|v\|_{\mathcal{A}}^2. \tag{5.12}$$

On the other hand, we know $\hat{\omega}_0 = \min_l \rho_l \lambda_{\min}(\mathcal{S}_l) \lesssim 1$. Therefore $K_1 \lesssim C_d(h)$ due to Lemma 4.24. The strengthened Cauchy-Schwarz inequality (5.11) and Lemma 4.25 give that $K_2 \lesssim 1$. The convergence result then follows directly from the general theory in Theorem 4.22. □

This theorem shows the HB preconditioner converges very fast when combined with some Krylov subspace method. However the conditioner number still depends on the meshsize h, especially in 3D. Now we discuss a little bit on how to remove such dependency.

Define an operator $\mathcal{H}: V \mapsto V$ such that

$$(\mathcal{H}v, w) := \sum_{l=0}^{L} \sum_{x_i \in \mathring{G}(\mathcal{M}_l)\backslash\mathring{G}(\mathcal{M}_{l-1})} h_l^{d-2}\Big((\mathcal{J}_l v - \mathcal{J}_{l-1}v)(x_i),\, (\mathcal{J}_l w - \mathcal{J}_{l-1}w)(x_i)\Big).$$

Hence we get

$$(\mathcal{H}v, v) = \sum_{l=0}^{L} \sum_{x_i \in \mathring{G}(\mathcal{M}_l)\backslash\mathring{G}(\mathcal{M}_{l-1})} h_l^{d-2}\Big|(\mathcal{J}_l v - \mathcal{J}_{l-1}v)(x_i)\Big|^2, \quad \forall\, v \in V.$$

This operator is in fact the inverse of the HB preconditioner, i.e., $\mathcal{H} = \mathcal{B}_{\mathrm{HB}}^{-1}$; see [130]. In fact, in the proof of Theorem 5.8, we have shown the following norm equivalence result:

$$\|v\|_{\mathcal{A}}^2 \lesssim (\mathcal{H}v, v) = \sum_{l=0}^{L} h_l^{-2}\|(\mathcal{J}_l - \mathcal{J}_{l-1})v\|_0^2 \lesssim C_d(h)\|v\|_{\mathcal{A}}^2. \tag{5.13}$$

Since Π_l is the $(\cdot,\cdot)_{\mathcal{A}}$-projection from V to V_l, it is easy to check that

$$a\big[(\Pi_i - \Pi_{i-1})v, (\Pi_j - \Pi_{j-1})v\big] = 0, \quad \forall i \neq j.$$

We can then obtain that

$$\begin{aligned}\|v\|_{\mathcal{A}}^2 &= \Big\|\sum_{l=0}^{L}(\Pi_l - \Pi_{l-1})v\Big\|_{\mathcal{A}}^2 = \sum_{0\leqslant i,j\leqslant L} a\big[(\Pi_i - \Pi_{i-1})v, (\Pi_j - \Pi_{j-1})v\big] \\ &= \sum_{l=0}^{L} a\big[(\Pi_l - \Pi_{l-1})v, (\Pi_l - \Pi_{l-1})v\big] = \sum_{l=0}^{L}\Big|(\Pi_l - \Pi_{l-1})v\Big|_1^2.\end{aligned}$$

Notice that this is corresponding to the telescope sum of the Ritz-projections in (5.7). Motivated by the above norm equivalence and (5.13), one can easily construct another multilevel PSC method

$$\mathcal{B} = \sum_{j=1}^{J} \mathcal{S}_j \Pi_j.$$

However, Π_j is not good for computation in general except for $d = 1$ in which $\Pi_j = \mathcal{J}_j$ is just the interpolation[2]. In the next section, we explore the idea of telescope expansion using the L^2-projection (5.7) instead of the interpolation or the Ritz-projection. And it turns out to give rise to the well-known BPX preconditioner.

[2]Note that this is equivalent to the HB preconditioner in 1D.

5.3 BPX preconditioner

In the previous section, along with the hierarchical basis decomposition, we have also obtained a natural multilevel space decomposition

$$V = \sum_{l=0}^{L} V_l = \sum_{l=0}^{L} \sum_{i=1}^{n_l} V_{l,i}, \tag{5.14}$$

which contains a lot of "redundancy". Heuristically, one might want to avoid such redundancy in their algorithms. However, it turns out these extra subspaces are not redundant for optimal convergence.

Using the multilevel space decomposition (5.14), we can construct multilevel subspace correction methods. Among them, the most prominent (multilevel) example of PSC methods is the BPX preconditioner [28] based on the multilevel subspace decomposition (5.14):

$$\mathcal{B} = \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j, \quad \text{with } J = \sum_{l=0}^{L} n_l, \tag{5.15}$$

which is computationally more appealing and converges uniformly. The HB and BPX preconditioners both belong to the class of *multilevel nodal basis* preconditioners.

Norm equivalence

Now we shall show why the BPX preconditioner is "better" than the HB preconditioner. We notice that the HB preconditioner is not optimal for higher dimensions than 1D due to the H^1-stability property of the interpolations. Now we can expect it should be better for the L^2 projections.

Lemma 5.9 (Telescope sum of L^2-projections)**.** For any $v \in V$, we have

$$\left|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right|_1 \eqsim h_l^{-1} \left\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right\|_0.$$

Proof. Using the inverse inequality, Proposition 3.12, we get

$$\left|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right|_1 \lesssim h_l^{-1} \left\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right\|_0.$$

Proposition 3.14, together with the trivial equality

$$(\mathcal{Q}_l - \mathcal{Q}_{l-1})v = (\mathcal{I} - \mathcal{Q}_{l-1})(\mathcal{Q}_l - \mathcal{Q}_{l-1})v,$$

gives the other direction. □

Lemma 5.10 (Strengthened Cauchy-Schwarz inequality for L^2-projections)**.** If $u, v \in V$, let $u_i := (\mathcal{Q}_i - \mathcal{Q}_{i-1})u$, and $v_j := (\mathcal{Q}_j - \mathcal{Q}_{j-1})v$, then we have

$$a[u_i, v_j] \lesssim \gamma^{|i-j|}\|u_i\|_{\mathcal{A}}\|v_j\|_{\mathcal{A}}.$$

Proof. If $j \geqslant i$, Lemma 5.9 shows that $\|v_j\|_0 \lesssim h_j\|v_j\|_{\mathcal{A}}$. Hence the desirable result follows directly from Lemma 5.5. If $i \geqslant j$, we can argue in a similar way. □

Lemma 5.11 (Norm equivalence)**.** For any $v \in V$, we have

$$\sum_{l=0}^{L} \|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\|_1^2 \cong \|v\|_1^2.$$

Proof. (i) Since $\mathcal{Q}_l$ is a L^2-projection, we have $\|\mathcal{Q}_l v\|_0 \leqslant \|v\|_0$, $\forall\, v \in L^2(\Omega)$. Furthermore, using Proposition 3.14, we obtain

$$\|\mathcal{Q}_l v\|_1 \leqslant \|v\|_1, \quad \forall\, v \in \mathscr{V}.$$

By space interpolation, we have, for any $\sigma \in (0, \frac{1}{2})$, that

$$\|\mathcal{Q}_l v\|_\sigma \leqslant \|v\|_\sigma, \quad \forall\, v \in \mathscr{V}.$$

Let $\alpha \in (\frac{1}{2}, 1)$. If $\Pi_l : \mathscr{V} \mapsto V_l$ is the standard H^1-projection, the finite element theory gives

$$\|v - \Pi_l v\|_{1-\alpha} \lesssim h_l^\alpha \|v\|_1, \quad \forall\, v \in \mathscr{V}. \tag{5.16}$$

Let $v_i := (\Pi_i - \Pi_{i-1})v$. Note that $\rho_l = \rho(\mathcal{A}_l) \cong h_l^{-2}$. It is easy to show, with help from the inverse inequality (Proposition 3.12) and (5.16), that

$$\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v_i\|_1^2 \lesssim h_l^{-2\alpha}\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v_i\|_{1-\alpha}^2 \lesssim h_l^{-2\alpha}\|v_i\|_{1-\alpha}^2 \lesssim h_l^{-2\alpha}h_i^{2\alpha}\|v_i\|_1^2 \cong \rho_l^\alpha h_i^{2\alpha}\|v_i\|_1^2.$$

Using this inequality and the Cauchy-Schwarz inequality, we can derive that

$$\sum_l \sum_{i,j} \big(\nabla(\mathcal{Q}_l - \mathcal{Q}_{l-1})v_i, \nabla(\mathcal{Q}_l - \mathcal{Q}_{l-1})v_j\big) = \sum_{i,j}\sum_{l=1}^{i\wedge j} \big(\nabla(\mathcal{Q}_l - \mathcal{Q}_{l-1})v_i, \nabla(\mathcal{Q}_l - \mathcal{Q}_{l-1})v_j\big)$$

$$\lesssim \sum_{i,j}\sum_{l=1}^{i\wedge j} \rho_l^\alpha h_i^\alpha h_j^\alpha \|v_i\|_1\|v_j\|_1 \lesssim \sum_{i,j} \rho_{i\wedge j}^\alpha h_i^\alpha h_j^\alpha \|v_i\|_1\|v_j\|_1 \lesssim \sum_{i,j} \gamma^{\alpha|i-j|}\|v_i\|_1\|v_j\|_1.$$

Note that here l, i, and j are all level indices and we can apply summation by parts.

We can show that $\sum_{i,j}\gamma^{\alpha|i-j|}\|v_i\|_1\|v_j\|_1 \lesssim \sum_i \|v_i\|_1^2 = \|v\|_1^2$, which, in turn, gives

$$\sum_l \|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\|_1^2 \lesssim \|v\|_1^2.$$

(ii) On the other hand, using Lemma 5.10, we obtain

$$\begin{aligned} |v|_1^2 &= \sum_{l,m} \big(\nabla(\mathcal{Q}_l - \mathcal{Q}_{l-1})v, \nabla(\mathcal{Q}_m - \mathcal{Q}_{m-1})v\big) \\ &\lesssim \sum_{l,m} \gamma^{|l-m|} \big\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\big\|_1 \big\|(\mathcal{Q}_m - \mathcal{Q}_{m-1})v\big\|_1 \lesssim \sum_{l} \big\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\big\|_1^2. \end{aligned}$$

Hence we get the norm equivalence using Proposition 1.11. □

Remark 5.12 (Fractional norm)**.** We have shown the norm equivalence in H^1-norm. In fact, similar results also hold for $H^\alpha(\Omega)$ with $\frac{1}{2} < \alpha < \frac{3}{2}$. □

Convergence analysis for BPX preconditioner

All subspaces in (5.14) are one-dimensional and, thus, the subspace problems are very easy to solve. We can write the subspace solver (exact solver on each one-dimensional subspace) as follows:

$$\mathcal{S}_l^0 v := \sum_{i=1}^{n_l} (\mathcal{A}\phi_{l,i}, \phi_{l,i})^{-1} (v, \phi_{l,i})\, \phi_{l,i} = \sum_{i=1}^{n_l} (\nabla\phi_{l,i}, \nabla\phi_{l,i})^{-1} (v, \phi_{l,i})\, \phi_{l,i}.$$

Since we are now considering the uniform refinement for the linear finite element discretization, we can use an approximation of $\mathcal{S}_l^0$, for example a local relaxation method:

$$\mathcal{S}_l v := \sum_{i=1}^{n_l} h_l^{2-d} (v, \phi_{l,i})\, \phi_{l,i} \quad (\approx \mathcal{S}_l^0 v). \tag{5.17}$$

This simplification helps us to reduce the cost of computation as well as implementation. Apparently, we have

$$(\mathcal{S}_l v, v) = h_l^{2-d} (\vec{v}, \vec{v}) = h_l^2 (v, v). \tag{5.18}$$

We have seen that the Richardson method, the damped Jacobi method, and the G-S method all satisfy such a condition; see (3.23).

Remark 5.13 (Behavior of the smoother)**.** Note that the method (5.17) is just the Richardson method with a weight $\omega = h_l^{2-d}$ on level l. □

Using the above space decomposition and subspace solvers $\mathcal{S}_l$, the PSC method yields the well-known *BPX preconditioner*

$$\mathcal{B} = \sum_{l=0}^{L} \mathcal{S}_l \mathcal{Q}_l = \sum_{l=0}^{L} \mathcal{I}_l \mathcal{S}_l \mathcal{Q}_l = \sum_{l=0}^{L} \mathcal{I}_l \mathcal{S}_l \mathcal{I}_l^T \tag{5.19}$$

in operator form [28].

Theorem 5.14 (Uniform convergence of BPX)**.** The BPX preconditioner (5.19) is uniformly convergent, i.e., $\kappa(\mathcal{B}\mathcal{A}) \lesssim 1$.

Proof. We take a decomposition $v = \sum_{l=0}^{L} v_l := \sum_{l=0}^{L} (\mathcal{Q}_l - \mathcal{Q}_{l-1})v$, where $\mathcal{Q}_{-1} = 0$. Then we can obtain, from Lemmas 5.11 and 5.9, that

$$(\mathcal{A}v, v) \cong \sum_{l=0}^{L} \big|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\big|_1^2 \cong \sum_{l=0}^{L} h_l^{-2} \|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\|_0^2 = \Big(\sum_{l=0}^{L} h_l^{-2} (\mathcal{Q}_l - \mathcal{Q}_{l-1})v,\, v \Big).$$

Define $\tilde{\mathcal{A}} := \sum_{l=0}^{L} h_l^{-2}(\mathcal{Q}_l - \mathcal{Q}_{l-1})$. Apparently, $(\mathcal{A}v, v) \cong (\tilde{\mathcal{A}}v, v)$, $\forall v \in V$. Using (5.6) and (5.7), we can easily verify that (see HW 5.17)

$$\tilde{\mathcal{A}}^{-1} = \sum_{l=0}^{L} h_l^2 (\mathcal{Q}_l - \mathcal{Q}_{l-1}).$$

Hence

$$(\tilde{\mathcal{A}}^{-1}v, v) = \sum_{l=0}^{L} h_l^2 (\mathcal{Q}_l v, v) - \sum_{l=0}^{L} h_l^2 (\mathcal{Q}_{l-1} v, v) = h_L^2 (\mathcal{Q}_L v, v) + \sum_{l=0}^{L-1} (1-\gamma^2) h_l^2 (\mathcal{Q}_l v, v).$$

Namely, $(\tilde{\mathcal{A}}^{-1}v, v) \cong (\mathcal{B}v, v)$. That is to say, $(\mathcal{A}v, v) \cong (\tilde{\mathcal{A}}v, v) \cong (\mathcal{B}^{-1}v, v)$. Hence it gives the uniform convergence result by Lemma 2.29. □

Remark 5.15 (Multilevel decomposition according to frequencies)**.** From the above analysis, we find that, for any $v \in V$,

$$\big|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\big|_1 \cong h_l^{-1} \big\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\big\|_0 \quad \Longrightarrow \quad \|\nabla v_l\|_0 \sim \|h_l^{-1} v_l\|_0.$$

This fact draws close comparison with the Fourier expansion. That is to say $v = \sum_{l=0}^{L} v_l$ is a multilevel decomposition to different frequencies. Hence $\tilde{\mathcal{A}}$ can be viewed as a multi-resolution expansion of $\mathcal{A}$ and $\kappa(\tilde{\mathcal{A}}^{-1}\mathcal{A}) \lesssim 1$. □

Matrix representation of BPX

Using the matrix representation notations introduced in §3.2 and §3.5, the equation (3.38) in particular, we immediately obtain the matrix representation of the BPX method:

$$\underline{\mathcal{B}}\,\underline{u} = \underline{\mathcal{B}u} = \sum_{l=0}^{L} \underline{\mathcal{I}_l}\,\underline{\mathcal{S}_l}\,\underline{\mathcal{Q}_l}\,\underline{u} = \sum_{l=0}^{L} P_l \,(h_l^{2-d} M_l)\,(M_l^{-1} P_l^T M)\,\underline{u} = \sum_{l=0}^{L} h_l^{2-d} P_l\, P_l^T M\, \underline{u}.$$

In view of (3.16), we get the matrix form of the BPX preconditioner

$$B := \underline{\mathcal{B}} M^{-1} = \sum_{l=0}^{L} h_l^{2-d} P_l\, P_l^T. \tag{5.20}$$

This is the matrix form of the BPX preconditioner when we implement it. To improve efficiency, we can use prolongation between two consecutive levels to obtain P_l.

5.4 Homework problems

Problem 5.16. Give the complete proof of the uniform convergence of the two-level domain decomposition method (Proposition 5.1). What will happen if we do not include the coarse-level correction (Remark 5.2)?

Problem 5.17. Let $\tilde{\mathcal{A}} := \sum_{l=0}^{L} h_l^{-2}(\mathcal{Q}_l - \mathcal{Q}_{l-1})$. Show that $\tilde{\mathcal{A}}^{-1} = \sum_{l=0}^{L} h_l^{2}(\mathcal{Q}_l - \mathcal{Q}_{l-1})$.

Problem 5.18. Implement the BPX preconditioner for the Poisson's equation on a uniform grid. You can choose your favorite discretization method.

Chapter 6

Geometric Multigrid Methods

Multigrid methods are a group of algorithms for solving differential equations using a hierarchy of discretizations. The idea of multigrid was proposed initially by Fedorenko [57] in 1962 for 2D finite difference systems arising from the Poisson's equation. It accelerates the convergence of a basic iterative method (known as a relaxation or smoother) by global corrections from time to time, accomplished by solving a coarse problem approximately. The coarse problem is "similar" to the fine grid problem, while much cheaper to solve. This recursive process is repeated until a coarse grid is reached where the cost of direct solution is negligible compared to the cost of one relaxation sweep on the finest grid. In 1970's, Widlund, Hackbusch, Brandt et al. [67, 30] noticed that this iterative procedure was considerably faster than standard relaxation methods and brought it to attention in the western scientific community.

6.1 Geometric multigrid method

Geometric multigrid (GMG) method is an optimal iterative solver for linear algebraic system (2.1) arising from discretizations of partial differential equations. It is based on two important observations we have pointed out earlier in Chapter 3:

- A local relaxation method damps out the non-smooth (high-frequency) error components and the residual becomes relatively smooth after a few relaxation sweeps;
- A smooth (low-frequency) vector can be well approximated on coarse spaces.

GMG establishes and makes use of hierarchical structures. It is also a good example of the idea of *divide and conquer*. This idea has been applied in two-grid methods; see §3.4. Unfortunately, for large-scale problems, the coarse grid problem might be still too large to be solved efficiently. This makes introducing more than two nested meshes a natural idea. The key steps in the multigrid method (see Figure 6.1) are listed as follows:

- **Smoothing**: Reduce high-frequency error using a few smoothing steps based on a simple iterative method;
- **Restriction**: Restrict the residual on a finer grid to a coarser grid;
- **Coarse grid correction**: Solve an approximate problem on a coarse grid;
- **Prolongation**: Represent the correction computed on a coarser grid to a finer grid.

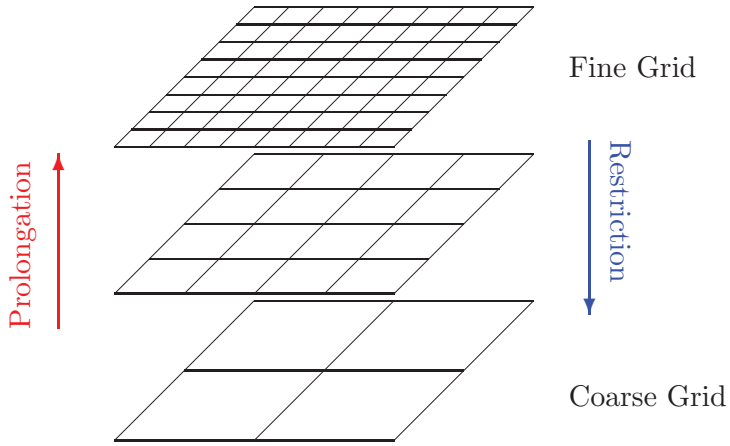


Figure 6.1: Pictorial representation of a multigrid method with three grid levels.

V-cycle multigrid method

Now we will explain the multigrid algorithms using the $\mathcal{P}_1$ finite element method for the Poisson's equation on $\Omega \subset \mathbb{R}^d$ as an example. Suppose we have a sequence of meshes $\mathcal{M}_l$ $(l = 0, \ldots, L)$ generated from an initial mesh $\mathcal{M}_0$ by (uniform) regular refinements. Hence meshsize h_l of $\mathcal{M}_l$ is proportional to γ^l with $\gamma \in (0, 1)$. Clearly,

$$h_0 > h_1 > h_2 > \cdots > h_L =: h.$$

It is easy to see that a multigrid method can be viewed a recursive two-grid method. So we only need to introduce how to do the iteration on two consecutive levels. We denote $\mathcal{I}_{l-1,l} : V_{l-1} \mapsto V_l$ $(l = 1, \cdots, L)$ as the natural embedding and $\mathcal{Q}_{l,l-1} = \mathcal{I}_{l-1,l}^T : V_l \mapsto V_{l-1}$ as the $(\cdot, \cdot)$-projection. Define $\mathcal{A}_l$ $(l = 1, \cdots, L)$ as the operator form of $\mathcal{A}$ on the subspace V_l in (5.1). Then a V-cycle multigrid method is given as follows:

Algorithm 6.1 (One iteration of MG V-cycle)**.** Assume that $\mathcal{B}_{l-1} : V_{l-1} \mapsto V_{l-1}$ is defined and the coarsest level solver $\mathcal{B}_0 = \mathcal{A}_0^{-1}$ is exact. We shall define, recursively, $\mathcal{B}_l : V_l \mapsto V_l$, which is

an iterator for the equation $\mathcal{A}_l v_l = r_l$. Let v_l be the initial guess on each level, i.e., $v_L = u^{(0)}$ and $v_l = 0$ for $0 < l < L$. Do the following steps:

(1) **Pre-smoothing**: For $k = 1, 2, \ldots, m$, compute

$$v_l \leftarrow v_l + \mathcal{S}_l\big(r_l - \mathcal{A}_l v_l\big);$$

(2) **Coarse grid correction**: Find an approximate solution $e_{l-1} \in V_{l-1}$ of the residual equation on level $l-1$, i.e., $\mathcal{A}_{l-1} e_{l-1} = \mathcal{Q}_{l,l-1}\big(r_l - \mathcal{A}_l v_l\big)$, by an iterative method:

$$e_{l-1} \leftarrow \mathcal{B}_{l-1}\mathcal{Q}_{l,l-1}(r_l - \mathcal{A}_l v_l), \quad v_l \leftarrow v_l + \mathcal{I}_{l-1,l} e_{l-1};$$

(3) **Post-smoothing**: For $k = 1, 2, \ldots, m$, compute

$$v_l \leftarrow v_l + \mathcal{S}_l^T\big(r_l - \mathcal{A}_l v_l\big).$$

From this algorithm, we can see this V-cycle multigrid method is just a generalization of Algorithm 3.2 (the abstract two-grid method). Clearly, this geometric multigrid method (with one G-S iteration as pre-smoothing and one backward G-S iteration as post-smoothing) is actually a special successive subspace correction (SSC) method based on the following multilevel space decomposition

$$V = \sum_{j=1}^{J} \tilde{V}_j = \sum_{l=L:-1:1} \sum_{i=1:n_l} V_{l,i} \;+\; V_0 \;+\; \sum_{l=1:L} \sum_{i=n_l:-1:1} V_{l,i},$$

which is a modification of (5.14). Furthermore, on each one-dimensional subspace $\tilde{V}_j$, the subspace problem is solved exactly.

According to Lemma 3.36, the error transfer operator of V-cycle on the l-th level can be written as

$$\mathcal{E}_l := \mathcal{I} - \mathcal{B}_l\mathcal{A}_l = \big(\mathcal{I} - \mathcal{S}_l^T\mathcal{A}_l\big)\big(\mathcal{I} - \mathcal{B}_{l-1}\mathcal{A}_{l-1}\mathit{\Pi}_{l-1}\big)\big(\mathcal{I} - \mathcal{S}_l\mathcal{A}_l\big),$$

where $\mathit{\Pi}_{l-1}$ is the Ritz-projection from V to V_{l-1}. By applying this operator recursively, we obtain the error transfer operator for the MG V-cycle:

$$\mathcal{E}_L = \mathcal{I} - \mathcal{B}_L\mathcal{A}_L\mathit{\Pi}_L = \big(\mathcal{I} - \mathcal{S}_L^T\mathcal{A}_L\big)\cdots\big(\mathcal{I} - \mathcal{S}_1^T\mathcal{A}_1\big)\big(\mathcal{I} - \mathit{\Pi}_0\big)\big(\mathcal{I} - \mathcal{S}_1\mathcal{A}_1\big)\cdots\big(\mathcal{I} - \mathcal{S}_L\mathcal{A}_L\big).$$

Matrix representation of GMG

Similar to the matrix representation of two-grid method discussed in §3.4, we can write the matrix representation of multigrid method. By definition, we have

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

Hence,

$$(\mathcal{A}_l \mathcal{Q}_l u, \mathcal{Q}_l v) = (\mathcal{I}_l^T \mathcal{A} \mathcal{I}_l \mathcal{Q}_l u, \mathcal{Q}_l v) = (\mathcal{A} \mathcal{I}_l \mathcal{Q}_l u, \mathcal{I}_l \mathcal{Q}_l v), \quad \forall\, u, v \in V.$$

It is easy to see that

$$\mathcal{A}_l = \mathcal{I}_l^T \mathcal{A} \mathcal{I}_l \quad \Longrightarrow \quad \underline{\mathcal{A}_l} = \underline{\mathcal{I}_l^T \mathcal{A} \mathcal{I}_l} = \underline{\mathcal{I}_l^T}\, \underline{\mathcal{A}}\, \underline{\mathcal{I}_l}.$$

This and (3.38), in turn, give the inter-grid transformations:

$$\hat{\mathcal{A}}_l = M_l \underline{\mathcal{A}_l} = M_l \underline{\mathcal{I}_l^T}\, \underline{\mathcal{A}}\, \underline{\mathcal{I}_l} = M_l\, \underline{\mathcal{Q}_l}\, M^{-1} \hat{\mathcal{A}}\, \underline{\mathcal{I}_l} = \underline{\mathcal{I}_l}^T \hat{\mathcal{A}}\, \underline{\mathcal{I}_l}, \quad 0 \leqslant l < L.$$

Hence we get the matrix form of the coarse level operator

$$\hat{\mathcal{A}}_l = P_l^T \hat{\mathcal{A}}\, P_l, \quad 0 \leqslant l < L. \tag{6.1}$$

Anisotropic problems ⋆

For GMG, smoothness of error is in the usual geometric sense. But it is not trivial for problems on unstructured meshes or problems with complicated coefficients. A representative example is the second-order elliptic problem

$$-\epsilon u_{xx} - u_{yy} = f(x, y), \quad \forall\, (x, y) \in \Omega, \tag{6.2}$$

where $\epsilon > 0$ is usually small. Other examples include problems with high-contrast coefficients, problems on anisotropic meshes, etc.

If we just naively apply the standard finite difference discretization in §1.2 on the uniform $n \times n$ tensor-product grid for this problem, or equivalently the $\mathcal{P}_1$ finite element discretization on uniform triangular grid from regular refinements, then the coefficient matrix for (6.2) is

$$A_\epsilon = I \otimes A_{1,\epsilon} + C \otimes I, \quad \text{with } A_{1,\epsilon} = \text{tridiag}(-\epsilon, 2 + 2\epsilon, -\epsilon),\ C = \text{tridiag}(-1, 0, -1).$$

The eigenvalues of A are given

$$\lambda_{i,j}(A_\epsilon) = 2(1 + \epsilon) - 2\epsilon \cos \frac{i\pi}{n+1} - 2 \cos \frac{j\pi}{n+1} = 4\epsilon \sin^2 \frac{i\pi}{2(n+1)} + 4 \sin^2 \frac{j\pi}{2(n+1)},$$

with eigenvectors

$$\vec{\xi}_{i,j} = \Big(\sin \frac{ki\pi}{n+1} \sin \frac{lj\pi}{n+1} \Big)_{k,l=1,\ldots,n}.$$

If $\epsilon \ll 1$, then $\lambda_{1,1} < \lambda_{2,1} < \cdots < \lambda_{n,1} < \lambda_{1,2} < \lambda_{2,2} < \cdots$. We notice that, unlike the Poisson's equation, these eigenvalues are ordered in a different pattern. The geometric low-frequencies can be highly oscillatory in the x-direction. It is natural to expect such a behavior from the PDE itself as the x-direction is much less diffusive than the y-direction. We call the x-direction (with smaller coefficient) the weak direction and the y-direction the strong direction.

We can also view this problem from a different perspective. Using the LFA analysis in §3.3, we obtain that the error of the G-S method satisfies

$$(2+2\epsilon)e_{i,j}^{\text{new}} = \epsilon e_{i-1,j}^{\text{new}} + \epsilon e_{i+1,j}^{\text{old}} + e_{i,j-1}^{\text{new}} + e_{i,j+1}^{\text{old}}, \quad i,j = 1,\ldots,n.$$

According to the local Fourier analysis, we can obtain that

$$\lambda(\theta_1,\theta_2) := \frac{\alpha_\theta^{\text{new}}}{\alpha_\theta^{\text{old}}} = \frac{\epsilon e^{\sqrt{-1}\theta_1} + e^{\sqrt{-1}\theta_2}}{2+2\epsilon - \epsilon e^{-\sqrt{-1}\theta_1} - e^{-\sqrt{-1}\theta_2}}.$$

In this case, the smoothing factor of the G-S method is

$$\bar{\rho}_{\text{GS}} = \lambda\left(\frac{\pi}{2}, \arctan\left(\frac{\epsilon(1-\bar{\rho}_{\text{GS}}^2)}{2(\epsilon+1)\bar{\rho}_{\text{GS}}^2}\right)\right) = \frac{\sqrt{5\epsilon^2-2\epsilon+1}+2}{5\epsilon+3} \longrightarrow 1, \quad \text{as } \epsilon \to 0.$$

This means the standard G-S method barely have any smoothing effect on the anisotropic problem when ϵ is small.

On the other hand, if we apply the line G-S smoother, things will be a lot different. Suppose we apply the line smoother in natural ordering (from left to right), namely,

$$(2+2\epsilon)u_{i,j}^{\text{new}} = \epsilon u_{i-1,j}^{\text{new}} + \epsilon u_{i+1,j}^{\text{old}} + u_{i,j-1}^{\text{new}} + u_{i,j+1}^{\text{new}}, \quad j = 1,\ldots,n, \quad i = 1,\ldots,n.$$

Then the error satisfies

$$(2+2\epsilon)e_{i,j}^{\text{new}} = \epsilon e_{i-1,j}^{\text{new}} + \epsilon e_{i+1,j}^{\text{old}} + e_{i,j-1}^{\text{new}} + e_{i,j+1}^{\text{new}}, \quad j = 1,\ldots,n, \quad i = 1,\ldots,n.$$

And we get

$$\lambda(\theta_1,\theta_2) := \frac{\alpha_\theta^{\text{new}}}{\alpha_\theta^{\text{old}}} = \frac{\epsilon e^{\sqrt{-1}\theta_1}}{2+2\epsilon - \epsilon e^{-\sqrt{-1}\theta_1} - 2e^{-\sqrt{-1}\theta_2}}.$$

The maximal smoothing factor is then

$$\bar{\rho}_{\text{LGS}} = \max\left\{\frac{\epsilon}{2+\epsilon}, \frac{\sqrt{5}}{5}\right\}.$$

If $0 < \epsilon \leqslant 1$, we always have $\bar{\rho}_{\text{LGS}} = \sqrt{5}/5 < 1$ independent of ϵ.

In the multigrid setting, one can handle such an equation using special techniques like: (1) an line smoother (group all y-variables corresponding to the same x-coordinate together), or (2) semi-coarsening (only coarse in y-direction), or (3) operator-dependent interpolations. In the next chapter, we will turn our attention to the third approach, which leads to algebraic multigrid methods for solving such difficult problems.

6.2 Nested iterations

The solve phase approximates corresponding problems by calling a two-grid algorithm recursively. There are different approaches for the solve phase; for example, we have seen the V-cycle method in §6.1. In this section, we discuss a few popular methods for the solve phase.

V-cycle and its generalizations

The V-cycle iterator $\mathcal{B}$, Algorithm 6.1, is a two-grid method with an inexact coarse-level solver defined recursively, i.e., the coarse-level iterator $\mathcal{B}_c$ is just $\mathcal{B}$ on the coarse grid. On the coarse level, we start from the initial guess $u_c^{\text{old}} = 0$ and then iterate

$$u_c^{\text{new}} = u_c^{\text{old}} + \mathcal{B}_c\big(f_c - \mathcal{A}_c u_c^{\text{old}}\big), \quad \text{where } \mathcal{B}_c \text{ is the two-grid method for } \mathcal{A}_c.$$

In the the V-cycle, we only apply the above iteration once on the coarse-level. Apparently, this procedure can be generalized. For example, we can iterate multiple steps:

$$u_c^{(0)} = 0, \quad u_c^{(k)} = u_c^{(k-1)} + \mathcal{B}_c\big(f_c - \mathcal{A}_c u_c^{(k-1)}\big), \quad k = 1, \ldots, \nu.$$

This gives the following equation

$$u_c^{(\nu)} = \mathcal{B}_c f_c + (\mathcal{I} - \mathcal{B}_c\mathcal{A}_c)u_c^{(\nu-1)} = \mathcal{B}_c f_c + \mathcal{E}_c u_c^{(\nu-1)} = \cdots = \big(\mathcal{I} + \mathcal{E}_c + \cdots + \mathcal{E}_c^{\nu-1}\big)\mathcal{B}_c f_c,$$

where $\mathcal{E}_c := \mathcal{I} - \mathcal{B}_c\mathcal{A}_c$. We can define a new iterator $\mathcal{B}_{c,\nu}$ such that

$$\mathcal{B}_{c,\nu} f_c := \big(\mathcal{I} - \mathcal{E}_c^{\nu}\big)\big(\mathcal{I} - \mathcal{E}_c\big)^{-1}\mathcal{B}_c f_c = \big(\mathcal{I} - \mathcal{E}_c^{\nu}\big)\mathcal{A}_c^{-1} f_c. \tag{6.3}$$

Motivated by (6.3), we can introduce a polynomial $q_\nu(t) := (1-t)^\nu \in \mathcal{P}_\nu$ and let

$$\mathcal{B}_{c,\nu} := \Big(\mathcal{I} - q_\nu\big(\mathcal{B}_c\mathcal{A}_c\big)\Big)\mathcal{A}_c^{-1}.$$

Then $\nu = 1$ yields the V-cycle apparently. The first non-trivial example is the well-known W-cycle ($\nu = 2$), which is a simple extension of the V-cycle algorithm; see Figure 6.2. By calling the coarse correction steps twice as in (6.4), we can obtain $\mathcal{B}_{c,2}$ (the W-cycle); see HW 6.4. The parameter ν is often called the *cycle index*.

Algorithm 6.2 (One iteration of multigrid cycle)**.** Assume that $\mathcal{B}_{l-1} : V_{l-1} \mapsto V_{l-1}$ is defined and the coarsest level solver $\mathcal{B}_0 = \mathcal{A}_0^{-1}$ is exact. We shall recursively define $\mathcal{B}_l : V_l \mapsto V_l$ which is an iterator for the equation $\mathcal{A}_l v = r_l$. Let $v = v^{(0)}$ be the initial guess.

(1) **Pre-smoothing**: For $k = 1, 2, \ldots, m$, compute

$$v \leftarrow v + \mathcal{S}_l\big(r_l - \mathcal{A}_l v\big);$$

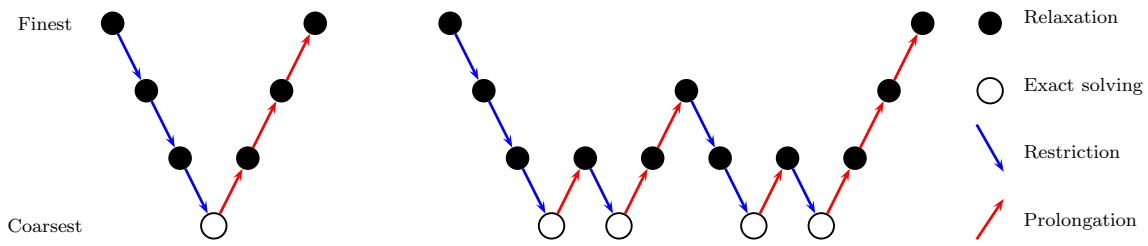


Figure 6.2: Multigrid V-cycle (left) and W-cycle (right).

(2) **Coarse grid correction**: Find an approximate solution $e_{l-1} \in V_{l-1}$ of the residual equation on level $l-1$, i.e., $\mathcal{A}_{l-1}e_{l-1} = \mathcal{Q}_{l,l-1}\big(r_l - \mathcal{A}_l v\big)$ using the iteration: Let $e_{l-1} = 0$ initially. For $k = 1, \ldots, \nu$, compute

$$e_{l-1} \leftarrow e_{l-1} + \mathcal{B}_{l-1}\Big(\mathcal{Q}_{l,l-1}(r_l - \mathcal{A}_l v) - \mathcal{A}_{l-1}e_{l-1}\Big); \tag{6.4}$$

Update the solution with

$$v \leftarrow v + \mathcal{I}_{l-1,l}e_{l-1};$$

(3) **Post-smoothing**: For $k = 1, 2, \ldots, m$, compute

$$v \leftarrow v + \mathcal{S}_l^T\big(r_l - \mathcal{A}_l v\big).$$

In the above general algorithm, the numbers of pre-smoothing and post-smoothing steps could be different from each other or level from level. The notation like V$(1,2)$ means the V-cycle multigrid with 1 pre-smoothing and 2 post-smoothing steps.

From the previous discussion, we have seen that there is a lot of freedom in the choice of $q_\nu(t)$. If $\nu = 1$, then Algorithm 6.2 is the V-cycle; if $\nu = 2$, then Algorithm 6.2 is the W-cycle. Apparently, the cycle index ν on each level does not have to be a fixed integer and one can use $\nu_{l-1} > 0$ to balance convergence and computation complexity; see Remark 6.3 for an alternative scheme.

In V-cycle and W-cycle, the iterators on all the coarser levels are the same. We can also use different polynomial orders ν_l on different levels l $(0 < l < L)$. For example, we can use a polynomial $q_\nu(t)$ such that $q_\nu(0) = 1$ and $0 \leqslant q_\nu(t) < 1$ on the spectrum of $\mathcal{B}_c\mathcal{A}_c$. This type of methods are referred to as the AMLI-cycle (Algebraic Multi-Level Iteration cycle[1]); see [1] and references therein for details.

Remark 6.1 (Nonlinear AMLI cycles)**.** Indeed, we can choose some optimal polynomial $q_\nu(t)$ like the Chebyshev polynomials. This reminds us about the Krylov subspace methods discussed

[1]Here "algebraic" stands for the fact that certain inner polynomial iterations are used in the multilevel cycle.

in §2.2. Inspired by this similarity, we can apply a preconditioned Krylov methods (like Flexible CG or GCR methods) on some of the coarse levels to improve convergence. This type of methods are called Krylov-cycle (K-cycle) methods or Nonlinear AMLI methods [97]. □

Example 6.2 (A simple AMLI-cycle)**.** A simple AMLI-cycle method is to give $l_0 \geqslant 1$, $\mu_1 \geqslant \mu_2 \geqslant 1$, and use the following polynomial orders

$$\nu_l := \begin{cases} \mu_1, & \text{if } \ l = kl_0; \\ \mu_2, & \text{otherwise.} \end{cases}$$

It is clear that, if $l_0 = 1$ and $\mu_1 = \mu_2 = 1$, then this method is just the standard V-cycle. □

Complexity of multigrid iterations

Now we turn our attention to the work estimate of nested iterations. For simplicity, we consider the AMLI-cycle with $\mu_2 \equiv 1$ only. Denote the work needed by $\mathcal{B}_l$ is W_l. Assume the each smoothing sweep costs $O(N_l)$ operations and $N_l \sim h_l^{-d} \sim \gamma^{-ld}$. Then it requires $2m\,O(N_l)$ operations for the pre- and post-smoothing on level l. The prolongation and restriction also requires $O(N_l)$ operations. Hence, for the AMLI-cycle, we have

$$\begin{aligned} W_{(k+1)l_0} &= \mu_1 O\big(N_{(k+1)l_0}\big) + O\big(N_{kl_0+1} + \cdots + N_{kl_0+l_0}\big) + \mu_1 W_{kl_0} \\ &= \mu_1 O\big(N_{(k+1)l_0}\big) + \mu_1 W_{kl_0} \\ &= \mu_1 O\big(N_{(k+1)l_0}\big) + \mu_1^2 O(N_{kl_0}) + \mu_1^2 W_{(k-1)l_0} \\ &= \cdots\cdots \\ &= O\Big(\sum_{j=2}^{k+1} \mu_1^{k+2-j} N_{jl_0}\Big) + \mu_1^k W_{l_0} \\ &= O\Big(\sum_{j=1}^{k+1} \mu_1^{k+2-j} N_{jl_0}\Big) \\ &= O\big(N_{(k+1)l_0}\big) \sum_{j=1}^{k+1} \big(\mu_1 \gamma^{d\,l_0}\big)^j. \end{aligned}$$

Let $N = N_L$ be the number of unknowns on the finest grid. This AMLI method costs $O(N)$ operations in each cycle, if we choose an appropriate μ_1 such that $\mu_1 \gamma^{d\,l_0} < 1$. Apparently, this analysis also yields computational complexity of the standard multigrid cycles like V-cycle and W-cycle quickly.

Remark 6.3 (Variable V-cycle)**.** Sometimes it is very desirable to use more smoothing steps on the coarse meshes to achieve better convergence. For example, we can modify the V-cycle algorithm by making the number of smoothing steps vary with the level l. Namely, we can

replace m in Algorithm 6.1 with m_l, where $m_l = \beta^l m$ with a fixed integer $\beta > 1$. Usually in practice $\beta = 2$ and $m = 1$ are taken and then $m_l = 2^{L-l}$. Note that the computational complexity is still optimal $\mathcal{O}(N)$ as the number of grid points decreases geometrically. □

Full multigrid method ⋆

The multigrid methods discussed above converge uniformly with respect to the meshsize h and requires $O(N)$ operations in each cycle. This means the computation cost is $O(N)$ to reach a fixed tolerance. On the other hand, when we solve a discrete partial differential equation, we need to solve the linear systems increasingly accurate use smaller tolerances for finer meshes, in order to obtain discretization accuracy. This leads to the fact that, to reach the discretization accuracy, the V-cycle multigrid method requires $O(N \log N)$ operations.

One way to further improve the cycling algorithms (for example, the V-cycle algorithm) is to provide good initial guesses using coarse approximations (cheap in computation). This idea leads to a nested iteration method, i.e., the so-called full multigrid (FMG) cycle; see Figure 6.3. From this figure, we can see the full multigrid method can be viewed as a sequence of V-cycles on different levels. Note that FMG prolongations are different than the usual prolongations because they must control error and decide when to proceed to the next finer level.

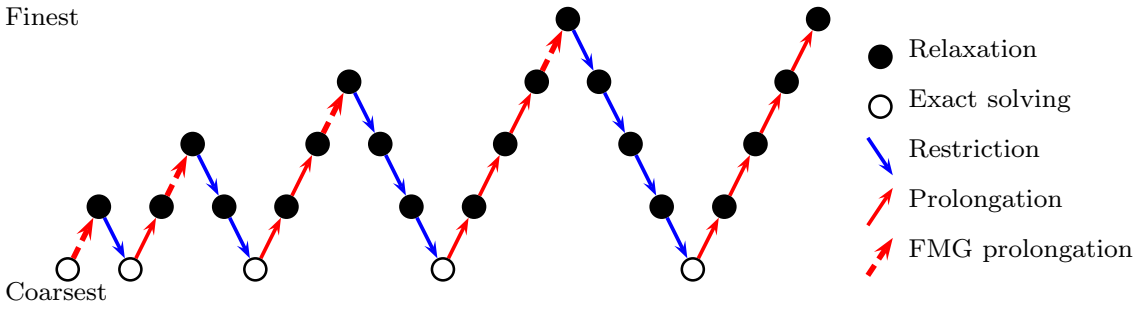


Figure 6.3: Full multigrid cycle.

We can write the concrete algorithm as follows:

Listing 6.1: Full multigrid method

```
1 ũ_0 ← A_0^{-1} f_0;
2 for l = 1, ..., L
3     u_l^{(0)} ← I_{l-1,l} ũ_{l-1};
4     u_l^{(k)} ← V-cycle(l, f_l, u_l^{(k-1)}),   k = 1, ..., ν;
5     ũ_l ← u_l^{(ν)};
6 end
```

Theorem 6.4 (Full multigrid convergence)**.** Assume that the l-th level iteration is a contraction with contraction factor $0 < \delta < 1$ independent of level l. If ν is large enough, then we have

$$\|\!|u_l - \tilde{u}_l|\!\| \lesssim h_l |u|_2,$$

where u_l is the exact solution of finite element problem on level l and $\tilde{u}_l$ is the full multigrid approximation solution on the l-th level.

Proof. Let $e_l := u_l - \tilde{u}_l$. Apparently, on the coarsest level, we have $e_0 = 0$ initially. On the l-th level ($0 < l \leqslant L$), we have

$$\begin{aligned} \|\!|e_l|\!\| &\leqslant \delta^\nu \|\!|u_l - \tilde{u}_{l-1}|\!\| \leqslant \delta^\nu \Big(\|\!|u_l - u|\!\| + \|\!|u_{l-1} - u|\!\| + \|\!|u_{l-1} - \tilde{u}_{l-1}|\!\| \Big) \\ &\leqslant \delta^\nu \Big(C h_l |u|_2 + \|\!|e_{l-1}|\!\| \Big). \end{aligned}$$

By iteration, we obtain that

$$\begin{aligned} \|\!|e_l|\!\| &\leqslant C\Big(\delta^\nu h_l + \delta^{2\nu} h_{l-1} + \cdots + \delta^{l\nu} h_1 \Big) |u|_2 \\ &= C\delta^\nu h_l \Big(1 + \delta^\nu \gamma + \cdots + \delta^{(l-1)\nu} \gamma^{l-1} \Big) |u|_2. \end{aligned}$$

Furthermore, if $\delta^\nu < \gamma$,

$$\|\!|e_l|\!\| \leqslant \frac{C\delta^\nu h_l}{1 - \gamma^{-1}\delta^\nu} |u|_2 \lesssim h_l |u|_2.$$

Hence the result. □

The above theorem indicates that, if we do enough number of V-cycles on each level (independent of meshsize h_l), we can obtain an approximate solution within the accuracy of discretization error. That is to say, $\|\!|u - \tilde{u}_l|\!\| \leqslant \|\!|u - u_l|\!\| + \|\!|u_l - \tilde{u}_l|\!\| \lesssim h_l |u|_2$. This means that FMG can reach discretization error tolerance within $O(N)$ operations.

6.3 Convergence analysis of multigrid methods

In this section, we show the slash cycle or the sawtooth cycle (i.e., /-cycle) method converges uniformly (h-independently) using the XZ identity discussed before. For simplicity, we will only discuss the proof in 1D here. The multidimensional cases and other MG methods can be analyzed in the subspace correction framework as well, but more technically involved; see [118] for example.

Convergence analysis of GMG method

Assume the subspace problems are solved exactly, i.e., $\mathcal{S}_{l,i} = \mathcal{A}_{l,i}^{-1}$, for $i = 1, \ldots, n_l$ and $l = 0, \ldots, L$. We denote the canonical interpolation operators from V to V_l as $\mathcal{J}_l$. That is to say, for any function $v \in V$,

$$\big(\mathcal{J}_l v\big)(x) = \sum_{i=1}^{n_l} v(x_i^l)\, \phi_i^l(x), \quad l = 0, \ldots, L.$$

Let $\mathcal{J}_{-1} v := 0$, $v_0 := \mathcal{J}_0 v$, and $v_l := (\mathcal{J}_l - \mathcal{J}_{l-1})v$, $l = 1, \ldots, L$. Using the interpolants in multilevel spaces, we can write

$$v = \mathcal{J}_L v = \sum_{l=0}^{L} \big(\mathcal{J}_l - \mathcal{J}_{l-1}\big)v = \sum_{l=0}^{L} v_l. \tag{6.5}$$

We also have

$$v = \sum_{l=0}^{L} v_l = \sum_{l=0}^{L} \sum_{i=1}^{n_l} v(x_i^l)\, \phi_i^l(x) =: \sum_{l=0}^{L} \sum_{i=1}^{n_l} v_{l,i}.$$

It is easy to check that

$$(\mathcal{I} - \mathcal{J}_k)v = \sum_{l=k+1}^{L} v_l = \sum_{l=k+1}^{L} \sum_{j=1}^{n_l} v_{l,j}$$

To estimate the convergence rate, in view of Corollary 4.19, we only need to estimate the quantity:

$$c_1 := \sup_{|v|_1 = 1} \inf_{\sum_{l,i} v_{l,i} = v} \sum_{l=0}^{L} \sum_{i=1}^{n_l} \Big| \Pi_{l,i} \sum_{(k,j) \geqslant (l,i)} v_{k,j} \Big|_1^2.$$

We now define and estimate

$$c_1(v) := \sum_{l=0}^{L} \sum_{i=1}^{n_l} \Big| \Pi_{l,i} \Big(\sum_{j=i}^{n_l} v_{l,j} + \sum_{k=l+1}^{L} \sum_{j=1}^{n_l} v_{k,j} \Big) \Big|_1^2.$$

We use the same notations introduced in Chapter 4 for projections, $\Pi_{l,i} : V \mapsto V_{l,i}$ is the $(\cdot,\cdot)_{\mathcal{A}}$-projection. For one-dimensional problems, it is easy to see that $\Pi_l = \mathcal{J}_l$; see HW 6.2. This leads to the following identity

$$\Pi_{l,i}(\mathcal{I} - \mathcal{J}_l) = 0, \quad \forall\, 1 \leqslant i \leqslant n_l,\ 0 \leqslant l \leqslant L.$$

Furthermore, we also have $\Pi_{l,i}(\sum_{j \geqslant i} v_{l,j}) = \Pi_{l,i}\big(v_{l,i} + v_{l,i+1}\big)$. Using these properties, we have

$$\begin{aligned}
c_1(v) &= \sum_{l=0}^{L} \sum_{i=1}^{n_l} \big| \Pi_{l,i}\big(v_{l,i} + v_{l,i+1}\big) + \Pi_{l,i}(\mathcal{I} - \mathcal{J}_l)v \big|_1^2 \\
&= \sum_{l=0}^{L} \sum_{i=1}^{n_l} \big| \Pi_{l,i}\big(v_{l,i} + v_{l,i+1}\big) \big|_1^2 \ \lesssim \ \sum_{l=0}^{L} \sum_{i=1}^{n_l} |v_{l,i}|_1^2 \\
&= \sum_{l=0}^{L} h_l^{-2} \big\| (\mathcal{J}_l - \mathcal{J}_{l-1})v \big\|_0^2 \ \lesssim \ \sum_{l=0}^{L} |v_l|_1^2 \ = \ |v|_1^2.
\end{aligned}$$

The last equality is easy to check; see HW 6.3. This estimate shows the convergence rate of MG is uniformly bounded.

Remark 6.5 (Relation with the HB preconditioner)**.** Note that several places in the above analysis depend on the one-dimensionality ($d = 1$) assumption for simplicity of presentation. In fact, the decomposition (6.5) used in this proof is the hierarchical basis (HB) decomposition in §5.2. We have already seen that the convergence rate of the HB method is actually not optimal in multidimensional cases ($d > 1$). So the proof must be changed in higher dimensions. We will not go into the details using this approach. There are several ways to prove the optimality of geometric multigrid methods in the literature and we review them briefly in the following subsection. □

Some historical remarks ⋆

The theoretical analysis in this note has been closely following the argument of theory of subspace corrections. We now pause a little and take a quick look at the history of multigrid convergence theory. It is not possible to review all the relevant literature about multigrid theory here though; the interested readers are referred to the monographs [69, 85, 20, 44, 110, 114], the survey papers [118, 131], and the references therein for further reading.

In early 1960's, the multigrid method was first introduced and analyzed by Fedorenko [57, 58] for finite difference equations from the Poisson's equation on the unit square. The result was extended to a more complex case with variable coefficients by Bakhvalov [5]. Nicolaides [92] gave an analysis for finite element discretizations of second-order elliptic equations. In the late 1970's, Hackbusch [67] and Brandt [30] made the historical breakthrough and showed that the multigrid technique is highly efficient. These seminar work made the idea of multigrid increasingly popular and extensive efforts had been made in order to give a general convergence theory since then. The simplest case is of course a hierarchy with only two levels. Bank and Dupont developed a two-level hierarchical basis (HB) finite element method [7] and gave the convergence proof of two-grid methods in the finite element setting [6]. Based on this two-grid theory, in some circumstances, one can show that the corresponding W-cycle (or more costly) multigrid with sufficient number of smoothing steps also converges with "similar" efficiency as the two-grid method; see [6, 68, 69, 110] for example. However, the uniform convergence of the V-cycle multigrid, which is more important in practice, cannot be proved in this way [68].

Hackbusch [68] and Braess and Hackbusch [18] first gave a general convergence theory for multigrid, including the V-cycle. The classical book by Hackbusch [69] summarized early development of convergence and optimality of multigrid methods. Hackbusch and collaborators reduced the conditions for the V-cycle convergence to the *smoothing* and *approximation* prop-

erties, namely,

$$(v_l, \mathcal{A}_l v_l) \lesssim (v_l, \mathcal{B}_l^{-1} v_l), \qquad \forall\, v_l \in V_l; \tag{6.6}$$

$$(w_l, \mathcal{B}_l^{-1} w_l) \lesssim \kappa(w_l, \mathcal{A}_l w_l), \qquad \forall\, w_l \in W_l := \big\{(\Pi_l - \Pi_{l-1})v \,:\, v \in V\big\}. \tag{6.7}$$

If the above conditions hold, then there is a positive mesh-independent constant C such that V(m, m)-cycle multigrid converges uniformly and

$$\|\mathcal{I} - \mathcal{B}^{\text{V-cycle}}\mathcal{A}\|_{\mathcal{A}} \leqslant \frac{C}{C+m},$$

which indicates the convergence factor goes to zero as the number of smoothing steps increases. The approximation property (6.7) often requires *full elliptic regularity* on the boundary value problem and quasi-uniformness of the underlying meshes. These restrictions made the classical theory not applicable in many situations where the multigrid methods are still effective. There are some exceptional cases where full elliptic regularity is not necessary; see, for example, [16, 10]. Bramble and Pasciak [22] introduced a *regularity and approximation* condition to show convergence of multigrid methods including the V-cycle for any positive m. Bank and Yserentant [10] presented the classical convergence theory of the multigrid methods from an algebraic point of view.

An alternative convergence theory is the framework of subspace corrections, with which inexact subspace solvers can be analyzed, very general meshes can be treated, and restrictive regularity assumptions can be removed. The subspace correction methods (or the Schwarz methods) emerged and analyzed in both multigrid and domain decomposition communities. Closely related to the multigrid methods (which can be viewed as multiplicative Schwarz methods), additive versions of the multilevel Schwarz method also gained popularity as parallel computers emerged and became the dominant computing environment. Yserentant [129] and Bank, Dupont, and Yserentant [8] extended the two-level HB idea to the multilevel case and obtained the HB preconditioner (additive) and the HBMG method (multiplicative), respectively. The HB-type methods (see §5.2) are easy to implement and very efficient in many cases, especially so in 2D.

Bramble, Pasciak, and Xu [28] proposed a parallel version of V-cycle multigrid called the multilevel nodal basis preconditioner, which is better known as the BPX preconditioner. In this seminar paper, the authors suggested an L^2-type telescope sum (see §5.3) to construct a stable decomposition, which is a break-through and motivated a lot of research. Such a tool also allowed Bramble, Pasciak, Wang, and Xu [27, 26] to analyze the V-cycle multigrid and domain decomposition methods on nonuniform meshes. This analysis gave convergence estimates for the multilevel Schwarz methods mildly depending on mesh size (i.e., depending on the number of levels only). Dryja and Widlund [51] also showed similar convergence estimates for the multilevel

additive Schwarz methods in a more general setting. Later, these results were improved and the multilevel Schwarz methods were finally shown to converge uniformly with respect to mesh size and number of levels (without regularity nor quasi-uniformity assumptions) in different ways [98, 132, 118, 25, 15, 65].

Xu [118] gave a unified theory on subspace correction methods based on stable subspace decomposition of finite element spaces and laid solid foundation for further studies in this field. Yserentant [131] reviewed the classical proof and the subspace correction proof for the convergence of multigrid methods. By combining the two convergence theories, Brenner [37] proved that convergence factor for some V-cycle methods decreases as number of smoothing steps increases without full elliptic regularity assumption. Moreover, Xu and Zikatanov [122] considered methods of subspace corrections in an abstract setting and showed that the convergence factor of successive subspace correction methods can be characterized by a precise estimate

$$\|\mathcal{I} - \mathcal{B}^{\text{V-cycle}}\mathcal{A}\|_{\mathcal{A}}^2 = 1 - \frac{1}{c_1},$$

which is known as the XZ identity (Theorem 4.15). This theory does not depend on the number of smoothing steps explicitly.

By far, we have mainly discussed general convergence theories for the multigrid methods. These theoretical results indicate that the convergence factor of multilevel iterative methods is independent of mesh size h without telling how big the convergence rate accurately is. Such qualitative theories usually do not give satisfactorily sharp or realistic predictions of the actual convergence factor in practice [110]. This seems contradictory as the XZ identity gives an exact equality for the convergence factor instead of an upper bound. But an optimal space decomposition in the XZ identity is not readily available practically speaking and, hence, it is not always easy to obtain a quantitative convergence estimate with the identity. Algebraic convergence estimates can be applied to obtain reasonable quantitative convergence speed for multigrid methods; see [84, 87] for more details. More algebraic convergence analysis results will be reviewed in Chapter 7.

Although the aforementioned qualitative results show h-independent convergent speed of multigrid methods, they still do not fully reflect high efficiency of multigrid algorithms (like the so-called textbook multigrid efficiency). Moreover, these results can not provide much assistance for designing an optimal algorithm. On the other hand, quantitative analysis tools, including *rigorous Fourier analysis* and *local Fourier analysis*, have been developed in the literature to analyze practical performance of multigrid methods for rather general problems. For some cases, they can even provide *exact* convergence factor of the multigrid algorithms (in the sense this convergence factor can be obtained by the worst case mode); see [32, 101].

For a particular problem, it was recommended to apply quantitative analysis (especially LFA)

to get some ideas on how fast the multigrid algorithms could converge. A general procedure for developing multigrid programs named *the LFA ladder* was employed in practice:

1. Choose a suitable discretization method for the problem;
2. Find a good smoother with satisfactory convergence factor μ using LFA;
3. Choose transfer operators and find the two-grid LFA convergence factor σ;
4. Check whether the two-grid LFA convergence factor σ is close to μ;
5. Check whether the convergence factor of the multigrid program approximates σ;
6. Apply the full multigrid and check whether the discretization accuracy is obtained.

This procedure help software engineers to make development decisions and improve development efficiency. We recommend the readers to [110, 117] for more details on these technical tools.

6.4 Two-grid estimates for multigrid analysis

In this section, we introduce a simple tool for estimating convergence speed of the multigrid methods using the two-grid convergence factor. As we mentioned earlier, although this classical approach works well for the W-cycle or more complicated cycles only, it is relatively easy to give practitioners some idea how fast a multigrid code should be quantitatively.

From two-grid to multigrid

It is well-known that, if the exact two-grid method converges sufficiently fast, then the corresponding W-cycle multigrid method will also converge fast [6, 68, 110]. This is very helpful, for practical purposes, to assess how fast a multigrid algorithm will work for a particular problem.

A more rigorous analysis has been given by Notay [93] through a closer look at convergence rate of the inexact (or perturbed) two-grid methods $\mathcal{B}_{\mathrm{TG}}$ in Algorithm 3.2. As we have seen earlier, the multigrid methods can be viewed as recursive calls of the two-grid method. Hence they are indeed inexact two-grid methods. Moreover, we have the following relations between the general two-grid method $\mathcal{B}_{\mathrm{TG}}$ and the exact two-grid method $\mathcal{B}_{\mathrm{eTG}}$:

$$\lambda_{\max}(\mathcal{B}_{\mathrm{TG}}\mathcal{A}) \leqslant \lambda_{\max}(\mathcal{B}_{\mathrm{eTG}}\mathcal{A}) \max\Big\{\lambda_{\max}(\mathcal{B}_c\mathcal{A}_c), 1\Big\},$$
$$\lambda_{\min}(\mathcal{B}_{\mathrm{TG}}\mathcal{A}) \geqslant \lambda_{\min}(\mathcal{B}_{\mathrm{eTG}}\mathcal{A}) \min\Big\{\lambda_{\min}(\mathcal{B}_c\mathcal{A}_c), 1\Big\}.$$

Let $\rho_l := \rho(\mathcal{I}_l - \mathcal{B}_l\mathcal{A}_l)$. In view of the above inequalities and (2.10), we obtain the following estimate

$$\rho_l^{\text{W-cycle}} \leqslant 1 - \Big(1 - \rho_l^{\text{eTG}}\Big)\Big(1 - \big(\rho_{l-1}^{\text{W-cycle}}\big)^2\Big), \quad l = 2, 3, \ldots, L.$$

If $\rho_l^{\text{eTG}} \leqslant \sigma < 1/2$ and $\rho_{l-1}^{\text{W-cycle}} \leqslant \frac{\sigma}{1-\sigma}$, then we can derive, by recursion, that

$$\rho_l^{\text{W-cycle}} \leqslant \frac{\sigma}{1-\sigma}, \quad l = 2, 3, \ldots, L.$$

This is a uniform estimate of the "convergence speed" of the W-cycle multigrid method with respect to the number of levels. This result confirms and quantifies the common wisdom about the W-cycle convergence speed.

Limitations of two-grid theory for GMG ⋆

However, as we mentioned earlier, this approach does not yield uniform convergence estimate for the V-cycle multigrid. This fact shows there is a fundamental difference between two-level and V-cycle multigrid iterations in terms of conditions on convergence. When the above technique is applied to the V-cycle multigrid method, we can easily obtain that: If $\lambda_{\max}(\mathcal{B}_l^{\text{eTG}}\mathcal{A}_l) \leqslant 1$ holds for all levels, then

$$\rho_l^{\text{V-cycle}} \leqslant 1 - \Big(1 - \rho_l^{\text{eTG}}\Big)\Big(1 - \big(\rho_{l-1}^{\text{V-cycle}}\big)\Big), \qquad l = 2, 3, \ldots, L.$$

For example, suppose that $\rho_1^{\text{V-cycle}} = \rho_1^{\text{eTG}}$ and the exact two-grid method converges uniformly with $\rho_l^{\text{eTG}} \leqslant 0.2$ for all $l > 0$. Then it yields the following non-uniform convergence estimates for the V-cycle multigrid:

$$\begin{aligned}
\rho_2^{\text{V-cycle}} &\leqslant 1 - 0.8 \times (1 - 0.200) = 0.360,\\
\rho_3^{\text{V-cycle}} &\leqslant 1 - 0.8 \times (1 - 0.360) = 0.488,\\
\rho_4^{\text{V-cycle}} &\leqslant 1 - 0.8 \times (1 - 0.488) \approx 0.590,\\
\rho_5^{\text{V-cycle}} &\leqslant 1 - 0.8 \times (1 - 0.590) \approx 0.672,\\
\rho_6^{\text{V-cycle}} &\leqslant 1 - 0.8 \times (1 - 0.672) \approx 0.738,\\
&\vdots
\end{aligned}$$

In general, uniform two-grid convergence is not sufficient to guarantee uniform convergence for the V-cycle multigrid; see [86] for example. To give uniform estimate for the V-cycle multigrid, there are additional conditions to be satisfied; see the work by Napov and Notay [88]. Nevertheless, from the above discussion, we find that the analysis for two-grid methods can improve understanding of the convergence behavior of multilevel iterative methods. It is simple yet very powerful. Furthermore, the analysis of inexact two-grid methods indicates that it is

possible to apply the inexact (possibly non-Galerkin) coarse-level operators and might lead to new multigrid algorithms (particularly, algebraic multigrid methods). More discussions can be found in the PhD thesis of Xuefeng Xu [126].

6.5 Implementation of multigrid methods

In this section, we will briefly discuss how to implement the multigrid V-cycle (Algorithm 6.1) for solving the finite element equation $Au = f$ with $A \in \mathbb{R}^{N\times N}$ (N is usually very large). There are a couple of different ways for implementing the multigrid V-cycle algorithm. Here we use a matrix-based implementation to allow generality.

A sparse matrix data structure

First of all, we discuss how to represent a sparse matrix in practice. Apparently we do not wish to store all the zeros in A. There are many different ways to store a sparse matrix with optimal storage complexity. More importantly, which storage format to use usually depends on the hardware architecture. A widely-used general purpose data structure is the so-called *Compressed Sparse Row* (CSR) format [103]. The CSR storage format of a sparse matrix A consists of three arrays, defined as follows:

1. An integer array of row pointers, IA, of size $N + 1$;
2. An integer array of column indices, JA, of size nnz;
3. A double array of non-zero entries corresponding to the column indices, val, of size nnz.

More precisely, the index $IA(i)$ points to the beginning of the i-th row in JA and val. Moreover, the nonzero entries of a sparse matrix are stored in the array val row after row consecutively, that is to say, the i-th row begins at $val(IA(i))$ and ends at $val(IA(i+1)-1)$. In a similar way, $JA(IA(i))$ to $JA(IA(i+1)-1)$ contain the column indices of the nonzeros in row i. Thus IA is of size $N + 1$ (number of rows plus one), JA, and val are of size equal to the number of nonzeros. The number of nonzeros in the i-th row is then equal to $IA(i+1)-IA(i)$ and the total number of nonzeros is equal to $IA(N+1)-IA(1)$. Note that, as a convention, we always start the indices from 1 instead of 0.

When the matrix is a boolean (i.e., all entries are either true or false), the actual nonzeros are not stored because there is no need to store them.

Example 6.6 (A simple CSR matrix)**.** Consider the following 4×5 matrix

$$\begin{pmatrix} 1.0 & 1.5 & 0 & 0 & 1.2 \\ 0 & 1.0 & 6.0 & 7.0 & 1.0 \\ 3.0 & 0 & 6.0 & 0 & 0 \\ 1.0 & 0 & 2.0 & 0 & 5.0 \end{pmatrix}$$

When in the CSR format, this matrix is stored in the following way:

- IA is of size 5 and
$$IA = \left\| \, 1 \, \middle| \, 4 \, \middle| \, 8 \, \middle| \, 10 \, \middle| \, 13 \, \right\|$$
- JA is of size $IA(5) - IA(1) = 12$
$$JA = \left\| \, 1 \, \middle| \, 2 \, \middle| \, 5 \, \right\| \, 2 \, \middle| \, 4 \, \middle| \, 3 \, \middle| \, 5 \, \right\| \, 1 \, \middle| \, 3 \, \right\| \, 3 \, \middle| \, 5 \, \middle| \, 1 \, \right\|$$
- val is of the same size as JA and
$$val = \left\| \, 1.0 \, \middle| \, 1.5 \, \middle| \, 1.2 \, \right\| \, 1.0 \, \middle| \, 7.0 \, \middle| \, 6.0 \, \middle| \, 1.0 \, \right\| \, 3.0 \, \middle| \, 6.0 \, \right\| \, 2.0 \, \middle| \, 5.0 \, \middle| \, 1.0 \, \right\|$$

Note that the indices in JA need not be sorted in ascending order as seen in this example. □

With a sparse matrix stored in the CSR format, the matrix-vector multiplication $y = Ax$ can be performed in the following simple way:

Listing 6.2: Sparse matrix-vector multiplication

```
1 for i = 1, ..., N
2     t ← 0;
3     for k = IA(i), ..., IA(i + 1) − 1
4         j ← JA(k);
5         t ← t + val(k) ∗ x(j);
6     end
7     y(i) ← t;
8 end
```

Now we give a pseudo code of the GS method for solving $Au = f$. We assume that the initial guess is stored in the vector u. The pseudo code below uses an ordering given by a permutation array π, which takes value from 1 to N. Note that if $\pi(\ell) = \ell$ for any ℓ, then the code just yields the forward Gauss–Seidel method. It is important to notice that the positions of the diagonal entries of A in JA and val are not known in advance.

Listing 6.3: Gauss–Seidel method with ordering

```
1 for ℓ = 1, ..., N
```

```
    i ← π(ℓ);
    t ← f(i);
    for k ← IA(i), ..., IA(i + 1) − 1
        j ← JA(k);
        if ( j == i )
            diag ← val(k);
        else
            t ← t − val(k) ∗ u(j);
        end
    end
    u(i) ← t/diag;
end
```

We immediately notice that this pseudo cod is very similar to the previous matrix-vector multiplication and can be implemented very easily.

Assembling finite element matrix

Geometric multigrid methods are often implemented in a matrix-free fashion and, hence, there is no need to assemble the global stiffness matrix. However, we are going to use a matrix-base implementation. So we now discuss the assembling of finite element matrix first. Consider the mesh depicted in Figure 6.4. Note that this mesh has two different types of elements, triangles and quadrilaterals.

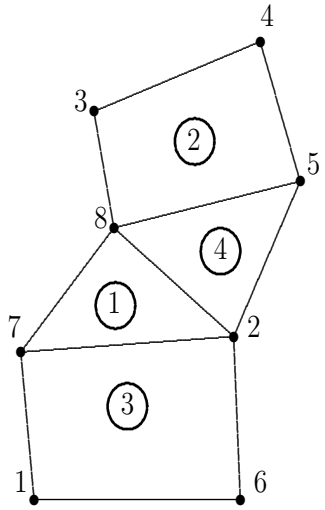


Figure 6.4: A mesh with 4 elements and 8 nodes

Most finite element basis functions are constructed to be locally supported. Very often, a "natural" assumption can be made about how a stiffness matrix is constructed from a finite element mesh:

We have a nonzero in the stiffness matrix at the (i, j)-entry if and only if the nodes i and j appear in the same element.

In order to assemble the stiffness matrix, the following two steps are performed:

1. Find the sparsity pattern of the stiffness matrix A, i.e., IA and JA;
2. Loop through elements and compute the actual entries in A.

The second step (the actual assembly of the entries) is usually easier and need to be done case by case. We will leave this step to the readers as a homework problem. Here we will only explain the first step in the following abstract algorithm:

Algorithm 6.3 (Finding sparsity)**.** Suppose a finite element mesh $\mathcal{M}$ is given.

(1) For each element, find the indices of nodes that belong to it;

(2) For each node, find the indices of elements that it belongs to $\Longrightarrow$ Patch(i);

(3) Obtain the sparsity pattern of A:

for $i \in$ Nodes($\mathcal{M}$)

for $e \in$ Patch(i)

Add all nodes in element e to the list of possible nonzeros in row i[2].

It just remains to show how we can perform steps (1) and (2) of Algorithm 6.3. This can be easily achieved by thinking of the *element–node* correspondence as a sparse matrix of size #elements×#nodes. For example, the so-called element topology is trivially represented by a 4×8 matrix E for which $E_{ij} = 1$ if and only if the node j is in the element i. Using the mesh in Figure 6.4 as an example, E is given below:

$$E = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Since this matrix has value either 1 or 0, we can represent E in the compressed sparse row format with the sparsity pattern only:

$$IE = \| 1 \,|\, 4 \,|\, 8 \,|\, 12 \,|\, 15 \,\|$$

$$JE = \| 7 \,|\, 8 \,|\, 2 \,\| 8 \,|\, 3 \,|\, 4 \,|\, 5 \,\| 1 \,|\, 6 \,|\, 7 \,|\, 2 \,\| 2 \,|\, 5 \,|\, 8 \,\|$$

[2]Make sure that you do not add anything twice. This can be done using an additional indicator array.

This concludes Step (1) of Algorithm 6.3.

On the other hand, Step (2) is also easy to achieve because the columns of E represent the correspondence

$$\text{Node} \mapsto \text{Patch}.$$

This means that for step (2) we can use the transpose matrix E^T.

Remark 6.7 (How to find transpose of a CSR matrix)**.** The *nontrivial* task here is to perform the transposition without using any additional memory. There is an algorithm to do that due to F. Gustavson, which dates to the 1970's and can be found in [99]. During transposition by this algorithm, the column indices in each row come out in increasing order, just for free. □

Matrix form of transfer operators

As seen in (6.1), we can obtain coarse level stiffness matrices using the algebraic Galerkin relation:

$$A_l = P_l^T A P_l, \quad 0 \leqslant l < L.$$

Since we can construct A_l's level by level, we only need the prolongation matrix $P_{l-1,l}$ for $0 < l \leqslant L$. In the operator form, it is trivial to define the prolongation, which is just the natural inclusion operator. But for implementation, we need to find the algebraic form of the prolongations. For geometric multigrid methods, we usually have to write the prolongation subroutines for different cases, and it makes the multigrid code almost a white box.

Here we are going to use a matrix-based implementation. Such a strategy is easy to be adapted to different discretization methods and have an almost identical structure as the AMG methods we will discuss next. Since we have seen how to apply a matrix-vector multiplication and how to apply the smoothers, we are left with construct prolongations as a sparse matrix. But, of course, obtaining such flexibility will cost us more storage as well as computational time. To complete the algorithm we have to give the action of $P_{l-1,l}^T$ and $P_{l-1,l}$, which are just matrix-vector multiplications to transfer data between two consecutive levels. The only programming difficulty here is keeping track of who on the fine grid is interpolated by whom on the coarse grid. We now focus on the particular case in which V_h is the classical linear finite element space corresponding to a uniform grid with size 2^{-L}.

Remark 6.8 (Matrix-free implementation of prolongation)**.** One can easily observe that, there is actually no need for A_{l-1} to be computed as $P_{l-1,l}^T A_l P_{l-1,l}$ because A_{l-1} is just the stiffness matrix corresponding to finite element discretization on a grid with size 2^{1-l}. In such a case, we do not need to store P themselves, but only the action of prolongations. □

In the following example, it is shown how to perform the actions of prolongation for $l = 1$ of grid with meshsize 1/4 on the unit square. We can easily obtain the finite element matrix on

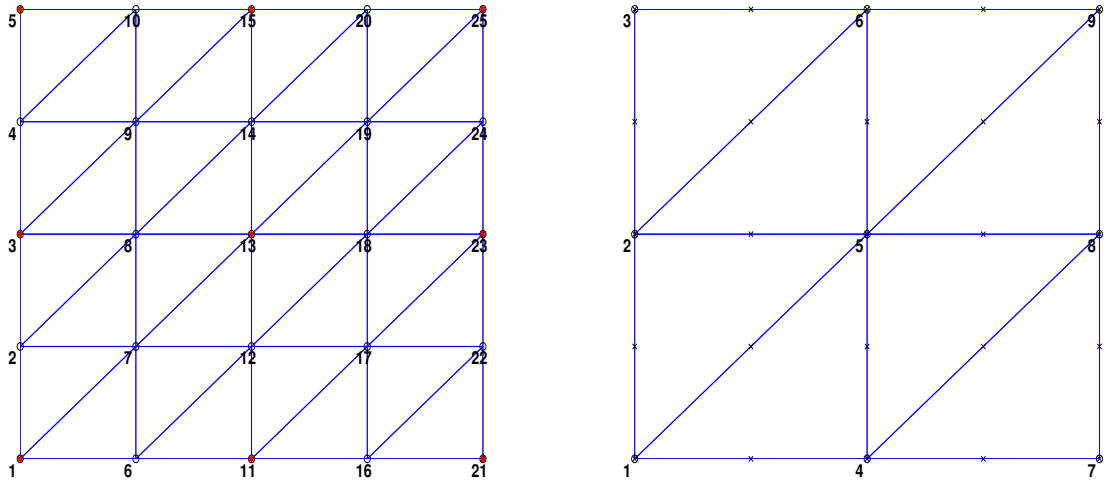


Figure 6.5: Fine and coarse meshes.

the fine mesh; see Table 6.1. Also the matrix corresponding to the coarse grid can be computed.

i	j	$(A_1)_{ij}$
1	1	1.0
1	4	−0.5
1	2	−0.5
2	1	−0.5
2	5	−1.0
2	2	2.0
2	3	−0.5
3	2	−0.5
3	6	−0.5

i	j	$(A_1)_{ij}$
3	3	1.0
4	1	−0.5
4	4	2.0
4	5	−1.0
4	7	−0.5
5	4	−1.0
5	5	4.0
5	2	−1.0

i	j	$(A_1)_{ij}$
5	6	−1.0
5	8	−1.0
6	5	−1.0
6	6	2.0
6	3	−0.5
6	9	−0.5
7	4	−0.5
7	7	1.0

i	j	$(A_1)_{ij}$
7	8	−0.5
8	7	−0.5
8	8	2.0
8	5	−1.0
8	9	−0.5
9	8	−0.5
9	9	1.0
9	6	−0.5

Table 6.1: The nonzero entries of the stiffness matrix A_1 on the fine grid.

Let $e(i)$, $i = 1, 2, \ldots, 9$ be a given vector corresponding to the representation of e_H on the coarse grid. Let $r(i)$, $i = 1, 2, \ldots, 25$ be a residual vector on the fine grid. According to the numbering

given in Figure 6.5, we have the following formulae for computing Pe and $P^T r$:

$$\begin{aligned}
(Pe)(13) &= e(5);\\
(Pe)(12) &= 0.5 * \big(e(4) + e(5)\big);\\
(Pe)(8) &= 0.5 * \big(e(2) + e(5)\big);\\
(Pe)(17) &= 0.5 * \big(e(4) + e(8)\big);\\
&\vdots\\
(P^T r)(5) &= r(13) + 0.5 * \big(r(7) + r(8) + r(12) + r(14) + r(18) + r(19)\big);\\
(P^T r)(1) &= r(1) + 0.5 * \big(r(2) + r(6) + r(7)\big);\\
(P^T r)(4) &= r(11) + 0.5 * \big(r(6) + r(12) + r(17) + r(16)\big);\\
&\vdots
\end{aligned}$$

The remaining values in Pe and $P^T r$ at other grid points can be obtained in similar way.

6.6 Homework problems

HW 6.1. Show the geometric multigrid V-cycle (Algorithm 6.1) is uniformly convergent in $\mathbb{R}^d$.

HW 6.2. If $\mathcal{A} = -\Delta$, show that the interpolant $\mathcal{J}_l : V \mapsto V_l$ is equal to the $(\cdot,\cdot)_{\mathcal{A}}$-projection $\mathit{\Pi}_l : V \mapsto V_l$.

HW 6.3. Let $\Omega = (0,1)$ and $v \in V_h$ be a $\mathcal{P}_1$ Lagrange finite element function. Show that $|v|_1^2 = \sum_{l=l}^{L} |v_l|_1^2$.

HW 6.4. Let $q(t) = (1-t)^2$. Show that $\mathcal{B}_{c,2} = \big(\mathcal{I} - q(\mathcal{B}_c\mathcal{A}_c)\big)\mathcal{A}_c^{-1}$ can be obtained by (6.4).

HW 6.5. Show the work estimate of the full multigrid method is $O(N)$.

Chapter 7

Algebraic Multigrid Methods

Consider the system of equations arising from the Poisson's equation on unstructured meshes or the second-order elliptic equation with anisotropic coefficients

$$Au = f, \quad \text{where } A \in \mathbb{R}^{N\times N} \text{ is SPD and } u, f \in \mathbb{R}^N.$$

Problems with anisotropic coefficients on regular meshes, or problems with isotropic coefficients but on anisotropic meshes, will cause troubles for geometric multigrid methods. While geometric multigrid (GMG) essentially relies on the availability of robust smoothers, algebraic multigrid (AMG) takes a different approach [35, 36, 102] by focusing on constructing suitable coarse space. AMG is a means to generalize GMG and to improve its robustness. There are several situations where AMG can be used but GMG is not; for example, problems on complex domain or irregular triangulation, problems with discontinues coefficients, and purely algebraic problems.

7.1 From GMG to AMG

How to make multigrid methods more robust in practice is a very important question from the early stage of the method development. And AMG is one of the approaches to improve robustness. In this section, we first show some motivations for the algebraic multigrid methods.

General procedure of multigrid methods

From our previous discussions, we observe that a typical MG algorithm contains two phases—the "*setup*" phase and the "*solve*" phase. The setup phase automatically initializes a hierarchical structure, including coarse spaces, prolongations and restrictions, and coarse problem solution methods for multilevel iterations. Notice that the setup phase only needs to be called once before iterations; sometimes, the same setup phase can be used at different time levels for time-dependent problems. For geometric multigrid (GMG) methods, the setup phase is trivial using

the hierarchical grid structure. However, GMG methods are difficult to apply for equations on general domains with unstructured grids. Algebraic multigrid (AMG) methods can be viewed as a generalization of geometric multigrid methods; see [125] and references therein for details.

We now explain how to perform multigrid setup phase in a relatively general setting. Once the setup is done, an appropriate nested iteration scheme should be chosen for the solve phase; see §6.2. It is immediately clear that we only need to discuss how to setup hierarchical information in two consecutive grids/levels for multigrid methods. We can summarize a general multigrid setup procedure as the following steps:

Step 1. **Selecting a smoother:** Choose a smoother S for $Au = f$.

Step 2. **Coarsening:** Identify a coarse space $V_c \subset V$, which contains smooth vectors.

Step 3. **Constructing a prolongation:** Construct a prolongation P in two steps:

3a. Decide, for each fine variable, which coarse variables are used for interpolation;

3b. Determine the weights for prolongation P.

Step 4. **Multilevel cycling:** Apply the same algorithm one or more times for the coarse problem $A_c u_c = f_c$, where $A_c = P^T A P$ and $f_c = P^T f$.

For GMG methods discussed in Chapter 6, Steps 2–4 are determined by the information of nested grids and the users can only find an appropriate smoother S. For example, in §1.4, we have presented a 1D GMG method in a purely algebraic fashion. We have observed that:

(1) GMG coarsening explores the topology of the graph representing the stiffness matrices on different levels are explicitly clear from the geometric refinement procedure;

(2) Prolongation and restriction for GMG usually depend only on the topological structure of the graph without knowing the grid coordinates;

(3) For GMG, smoothness of error is in the geometric sense and, in more general settings, smooth error can be geometrically non-smooth.

The key to an efficient GMG algorithm is to construct effective and cheap smoothers for the problem at hand. On the contrary, for AMG, we focus on how to pick coarse space and constructing interpolation to approximate the error components that cannot be effectively reduced by smoothing. AMG usually employs a simple relaxation process (typically point-wise relaxation) and then attempts to construct a suitable operator-dependent interpolation using the algebraic information of A to treat the error components that cannot be reduced by the relaxation process.

Sparse matrices and graphs ⋆

A sparse matrix can be represented as a graph. As the sparse matrices that we consider are mainly symmetric in the following we only discuss undirected graphs here. We first introduce a few elementary concepts from the graph theory. An *undirected graph* (or simply a *graph*) G is a pair (V, E), where V is a finite set of points called *vertices* and E is a finite set of *edges*. As set of vertices we always consider subsets of $\{1, \ldots, N\}$. An edge in E is an unordered pair $(j,\, k)$ with $j,\, k \in \mathsf{V}$. A graph $\mathsf{G}_0 = (\mathsf{V}_0, \mathsf{E}_0)$ is called a *subgraph* of $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, if $\mathsf{V}_0 \subset \mathsf{V}$ and $\mathsf{E}_0 \subset \mathsf{E}$.

If $(j, k) \in \mathsf{E}$ is an edge in an undirected graph $\mathsf{G} = (\mathsf{V},\, \mathsf{E})$, vertices j and k are said to be *adjacent*. The set of neighboring vertices of i is the set of all vertices that are adjacent to i; and it is denoted as $\mathsf{N}_i \subseteq \mathsf{V}$. An *independent set* of V is a set of vertices of G, no two of which are adjacent. A *maximal independent set* (MIS) or *maximal stable set* is an independent set such that adding any other vertex to the set will introduce at least one adjacent pair. A graph may have many MIS's of different sizes; the largest, or possibly several equally large, MIS of a graph is called a *maximum independent set*.

A *path* from a vertex i to another vertex j is a sequence of edges

$$\big\{(i,\, j_1),\, (j_1, j_2),\, \ldots,\, (j_{l-2}, j_{l-1}),\, (j_{l-1}, j)\big\} \subseteq \mathsf{E}$$

and the number of edges l is called the length of this path. A vertex j is *connected* to a vertex k if there is a path from j to k. The distance between j and k is defined as the length of the shortest path between these two vertices. Apparently, the distance between two vertices is equal to 1 if they are adjacent and is set to ∞ if they are not connected. An undirected graph $\mathsf{G} = (\mathsf{V},\, \mathsf{E})$ is *connected* if any pair of vertices are connected by a path, otherwise G is said to be *disconnected*.

Let $A \in \mathbb{R}^{N\times N}$ be a sparse matrix. The *adjacency graph* of A, denoted by $\mathsf{G}(A)$, is a graph $\mathsf{G} = (\mathsf{V},\, \mathsf{E})$ with $\mathsf{V} := \{1, 2, \ldots, N\}$ and

$$\mathsf{E} := \big\{(j,\, k) \,:\, a_{j,k} \neq 0\big\}.$$

As a general rule, sparse matrices do not provide any geometric information for the underlying graph and only the combinatorial/topological properties of $\mathsf{G}(A)$ or its subgraphs; see Figure 7.1. We note that two different discretizations on different meshes could lead to same sparse coefficient matrix A and, hence, same graph $\mathsf{G}(A)$.

Let A be the coefficient matrix corresponding to the finite element discretization of the second-order elliptic equation with Neumann boundary condition. Apparently A has zero row sum. Hence we can write

$$(Au, v) = \sum_{\substack{(i,j)\in\mathsf{E}\\ i<j}} -a_{i,j}(u_i - u_j)(v_i - v_j). \tag{7.1}$$

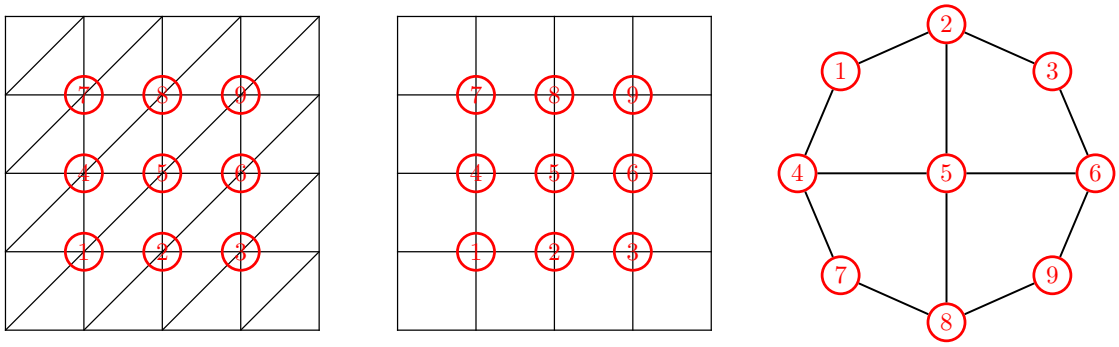


Figure 7.1: Finite element grid (left), difference grid (middle), and graph of their corresponding stiffness matrices (right).

We can also easily derive the corresponding equality for the Dirichlet boundary condition or the mixed boundary condition:

$$(Au, v) = \sum_{\substack{(i,j)\in \mathsf{E} \\ i<j}} -a_{i,j}(u_i - u_j)(v_i - v_j), \qquad \text{if} \quad u_j = v_j = 0, \ \ \forall\, x_j \in \Gamma_D. \tag{7.2}$$

M-matrix and Delaunay triangulation ⋆

We first introduce the concept of M-matrix. We call A an *M-matrix* if it is irreducible (i.e., the graph $\mathsf{G}(A)$ is connected) and

$$a_{i,i} > 0, \quad a_{i,j} \leqslant 0 \ \ (i \neq j), \quad a_{j,j} \geqslant \sum_{i\neq j} \big|a_{i,j}\big|, \quad a_{j,j} > \sum_{i\neq j} \big|a_{i,j}\big| \ \text{ for at least one } j.$$

Apparently, the stiffness matrix in (1.27) is an *M-matrix*. The classical convergence theory for AMG has been developed for this class of symmetric M-matrices [31, 102]. It is in general not the case for the stiffness matrices from finite element discretizations, even for the Poisson's equation. In fact, whether a stiffness matrix is an M-matrix depends on the underlying mesh $\mathcal{M}$. In practice, many AMG algorithms use simple filtering schemes to construct an M-matrix based on A. Xu and Zikatanov [124] introduced the concept of M-matrix relatives to analyze such cases.

First we introduce a few notations using Figure 7.2. In any give simplicial element τ in $\mathbb{R}^3$; similar definitions can be introduced in $\mathbb{R}^d$ for $d \geqslant 2$. An edge (i, j) has two vertices x_i and x_j and denote this edge as E. Let $\kappa_E(\tau) := F_i \bigcap F_j$ and $\theta_E(\tau)$ be the angle between faces F_i and F_j. Define a quantity

$$\omega_E(\tau) := \frac{1}{d(d-1)} \big|\kappa_E(\tau)\big| \cot \theta_E(\tau). \tag{7.3}$$

We then have the following result; see [121] for details.

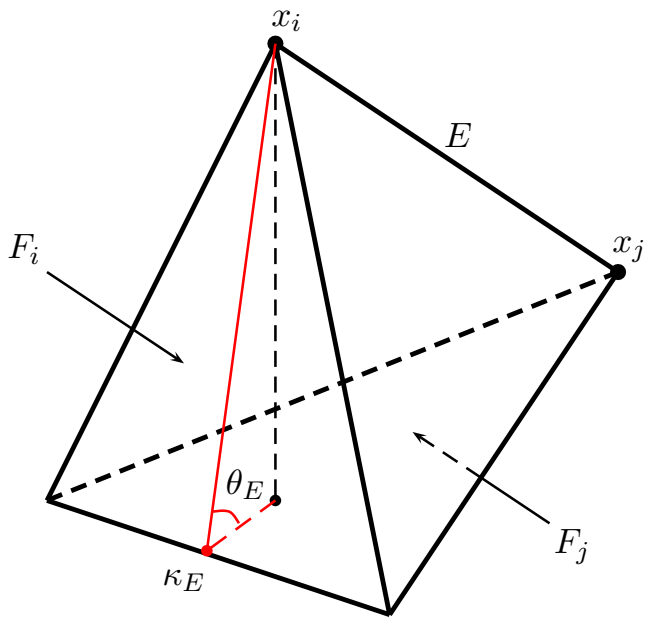


Figure 7.2: Definition of θ_E and κ_E in a simplex in 3D.

Proposition 7.1 (Condition for M-matrix)**.** The stiffness matrix for the Poisson's equation is an M-matrix if and only if, for any edge E, $\sum_{\tau \supset E} \omega_E(\tau) \geqslant 0$ with $\omega_E(\tau)$ defined in (7.3).

Remark 7.2 (Delaunay triangulation and M-matrix)**.** In $\mathbb{R}^2$, the above proposition simply means the sum of the angle opposite to any edge is less than or equal to π, which means the underlying triangulation must be Delaunay. Hence the stiffness matrix for the Poisson's equation is an M-matrix if the triangulation is Delaunay. And the condition is almost sharp[1]. □

For a given mesh $\mathcal{M}_h$, the stiffness matrix of $\mathcal{P}_1$-finite element method for the Poisson's equation is not necessarily an M-matrix. However, it can be estimated by an M-matrix. More specifically, if we keep all the vertices on $\mathcal{M}_h$ and swap internal edges, we can obtain a Delaunay triangulation $\mathcal{M}_h^D$. We have

$$\left(A_{\mathcal{M}_h^D} v, v\right) \leqslant \left(A_{\mathcal{M}_h} v, v\right), \quad \forall\, v \in \mathbb{R}^N;$$

moreover, the equality in the above inequality holds if and only if $\mathcal{M}_h$ is Delaunay. We refer the interested readers to [100] for details. Let $\phi_{\mathcal{M}_h} \in V_h$ is a piecewise linear function and $\phi_{\mathcal{M}_h}(x) = \sum_{i=1}^N v_i \phi_{i,\mathcal{M}_h}(x)$. Then we have

$$\left|\phi_{\mathcal{M}_h^D}\right|_1^2 \leqslant \left|\phi_{\mathcal{M}_h}\right|_1^2, \quad \forall\, v \in \mathbb{R}^N.$$

This means the Delaunay triangulation results in lower roughness of finite element functions among all possible triangulations on a fixed set of vertices.

[1]The opposite direction is true with a few possible exceptions near the boundary

Tarjan's algorithm ⋆

By far we have not assigned any kind of ordering for the unknowns in the solution vector. Sometimes, it is very important for the iterative methods like the Gauss–Seidel method. For example, in Remark 3.29, we have shown that the ordering is important using the local Fourier analysis. In AMG methods, the underlying meshes are not accessible and the natural ordering or C/F ordering can be used. We can also order the unknowns based on algebraic information. In particular, when we solve a flow problem, we would like to order the unknowns following the direction of the flow. Such an ordering (or permutation) results in a matrix which has all its "big" entries in the lower triangle and this technique can enhance the performance of the Gauss–Seidel smoother.

The first question is of course how to find such an ordering. In this section we present the Tarjan's algorithm [109] in the graph theory to find the "best ordering" for the Gauss–Seidel method. Generally, we do this in two steps:

1. Drop some of the entries in the matrix A, which are considered non-essential. This will transform the graph corresponding to A to a directed one.

2. Find the strongly connected components in this directed graph. Each one of these components will correspond to a diagonal block in the stiffness matrix after the permutation.

In fact, after permutation according to the strongly connected component ordering obtained by the Tarjan's algorithm, the matrix A will have the following structure:

$$A = \begin{bmatrix} \boxed{A_{11}} & \approx \epsilon & \approx \epsilon & \cdots & \cdots \\ \boxed{A_{21}} & \boxed{A_{22}} & \approx \epsilon & \cdots & \cdots \\ \boxed{A_{31}} & \boxed{A_{32}} & \boxed{A_{33}} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \ddots & \cdots \\ \boxed{A_{K1}} & \boxed{A_{K2}} & \boxed{A_{K3}} & \cdots & \boxed{A_{KK}} \end{bmatrix}. \tag{7.4}$$

The actual algorithm for finding the strongly connected components in the digraph will be written below. Before that, we have a look at Figure 7.3. The bold edges in this figure represent precisely the strong connections. If we number the blocks from left to right (the unknowns in each block can be in arbitrary order), the stiffness matrix will have the type of structure as the one in (7.4).

Imagine now that the graph represents a town, the edges are streets and the vertices are houses. You are walking along the streets, some of them are one way (directed). You may go and arrive at a house for the first time; other than that, there are two situations which may occur:

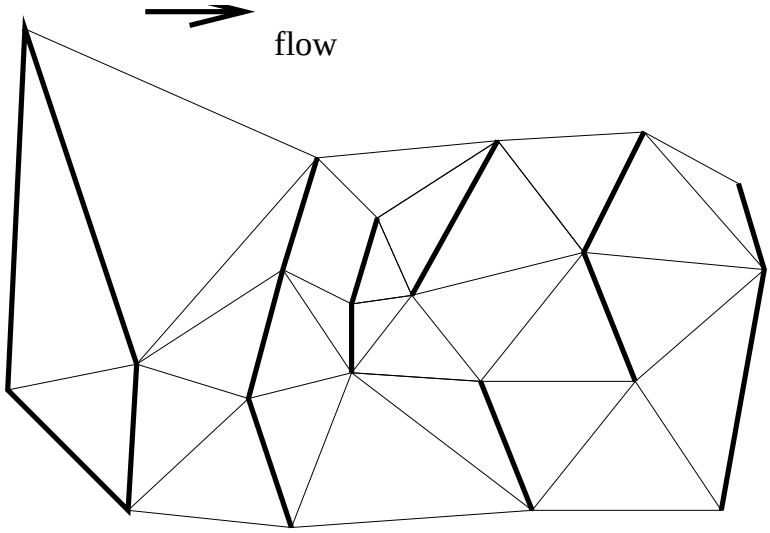


Figure 7.3: A sample mesh with a specify flow direction

1. Either you arrive at a house (vertex) you have already visited, or
2. You are at a house with no way out of it, i.e. a vertex with all edges pointing to it.

Having this in mind it is obvious that, if we return at a place we have been before (encountering a cycle), this corresponds to a so-called *strongly connected component*. In the second case, it is precisely the vertex we would like to number last, because all edges are sinking into it, i.e., it is at the end of the flow. The algorithm then is as follows:

Algorithm 7.1 (Simplified Tarjan's algorithm)**.** Given a directed graph G with N vertices.

1. If all vertices of G have been numbered, stop.
2. Set $i = 0$.
3. Choose any unnumbered vertex $v \in \mathsf{G}$.
4. If v has no edge out, we number it $N - i$, set $i = i + 1$, and return to Step 3.
5. If v has been visited before (encounter a cycle), then
   - Collapse all the vertices in the cycle as a single vertex v_{macro};
   - Connect v_{macro} with all vertices which were connected to member(s) of v_{macro};
   - Thus we obtain a new graph G'. Goto Step 2 and continue with G'.

Example 7.3 (Finding strongly connected components of a graph)**.** The above algorithm is visualized in Figures 7.4, which we have a 2D flow problem. First we assume that we start from

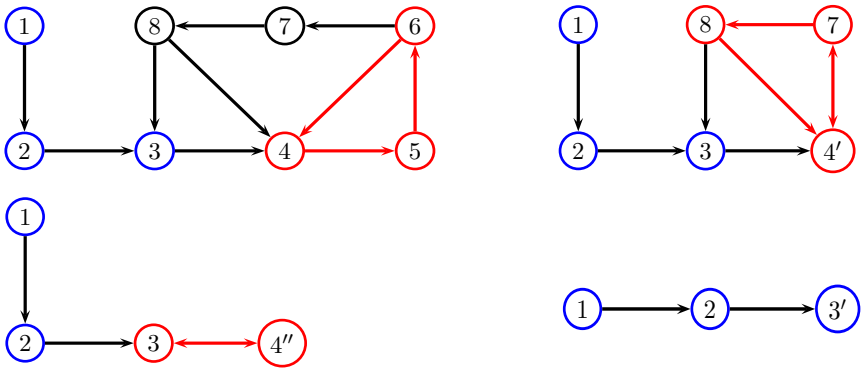


Figure 7.4: Finding strongly connected components of a directed graph

vertex 1 and then follow the path

$$1 \to 2 \to 3 \to 4 \to 5 \to 6 \to 4,$$

and we encounter a cycle. We collapse $\{4, 5, 6\}$ as a single vertex $v_{\text{macro}} = 4'$ and return from the beginning. Following the path

$$1 \to 2 \to 3 \to 4' \to 7 \to 8 \to 4',$$

again we have a cycle. We collapse the cycle and set $4'' = \{4', 7, 8\}$. The next step is again collapsing a cycle $\{4'', 3, 4''\}$ to a vertex $3'$. What is left after this step is a simple graph with three vertices, which precisely looks like the one corresponding to 1D convection dominated problem. □

Apparently, this algorithm has a drawback that there might be quite a lot of renumbering when collapsing the cycles. The fix is to use a stack as proposed by Tarjan and do not renumber anything until the whole connected component is in the stack. This reduces the renumbering dramatically and such an algorithm is linear in the total number of vertices and edges in the graph. In turn, for finite element stiffness matrices and their graphs, this algorithm is linear in the number of unknowns, because these matrices have just a few number of non-zeros per row, i.e., each vertex is incident with only few edges (only a bounded number independent of the mesh size h). A computer program realizing the above Tarjan's algorithm can be found in the article by Gustavson [66]. A good explanation and a lots of examples related to the Tarjan's algorithm are to be found in [61]. A better and more general algorithm is known as the Cross-Wind-Block method by Wang and Xu [115].

Remark 7.4 (Preprocessing to a get directed graph)**.** We comment that sometimes the graph corresponding to A (for example, the finite element stiffness matrix of the Poisson's equation)

is undirected. However, if we are consider a non symmetric problem, situation could be very different. In any event, suppose that we can make the graph be directed by "dropping" some of the "insignificant" entries of A. For example, by setting a threshold $\epsilon \in (0,1)$, we drop all a_{ki} such that $|a_{ki}/a_{ik}| < \epsilon$. Then we can apply the Tarjan's algorithm for finding the strongly connected components in the digraph. □

7.2 Motivations of algebraic multigrid methods

In §6.3, we have discussed general convergence theory for multigrid methods. In this section, we briefly review the convergence theory that are applicable to AMG methods and give on the construction of AMG methods. Following the seminar work by Brandt et al. [35, 36, 31] on the convergence analysis applicable to AMG methods, there have been a lot of discussions on the AMG convergence theory; see [102, 40, 104, 55, 56, 114] for example. The readers are referred to the recent survey paper by Xu and Zikatanov [124].

Algebraic convergence theory

Since the Fourier analysis is not available, algebraic convergence theory appears to be the right tool for developing and analyzing AMG algorithms. Since sharp and computable estimates for general AMG schemes are still lacking [76, 96, 124], we mainly focus on the classical two-level theory of AMG for symmetric positive-definite (SPD) problems. For the development on non-symmetric problems, we refer to [78, 94, 80, 79]; for analysis based on aggregation-type AMG algorithms, we refer to [111, 113, 41, 86, 89, 39].

We have shown the exact convergence factor estimate of two-level methods in Theorem **??**. Now we derive a convergence estimate from an algebraic viewpoint. In particular, we wish to give conditions on the grid-transfer matrices, like P, such that two-level AMG methods converge. We mainly follow the argument in a recent survey by MacLachlan and Olson [76]. Throughout this chapter, we assume that

Assumption 7.5 (General AMG setting)**.** The coefficient matrix A is SPD, the prolongation P has full column-rank, and the given smoother S itself is A-convergent (i.e., $\|I - SA\|_A < 1$).

Let $V = \mathbb{R}^N$ and $V_c = \mathbb{R}^{N_c}$ denote the fine and coarse spaces, respectively. For simplicity, we focus on Algorithm 3.3, V$(0,1)$ two-grid method. The CGC operator corresponds to the matrix $I - \Pi_c$ and

$$\Pi_c = PA_c^{-1}P^TA = P(P^TAP)^{-1}P^TA$$

is a projection onto range(P). The error reduction matrix for the two-gird method in Algorithm 3.3 can then be written as

$$E_{\mathrm{TG}} := (I - SA)(I - \Pi_c). \tag{7.5}$$

From Theorem **??**, the convergence rate of the two-grid method depends on effectiveness of the smoother S and approximability of the coarse space range(P). Our goal is to give an estimate in the form of

$$\|E_{\mathrm{TG}}\|_A^2 = \sup_{e\neq 0} \frac{\|(I - SA)(I - \Pi_c)e\|_A^2}{\|e\|_A^2} = 1 - \delta^*, \tag{7.6}$$

where δ^* yields the sharp and parameter-independent two-grid convergence factor. Of course, it is essential to pose conditions only on the prolongation P to ensure convergence, as the rest components in (7.5) are considered given.

Theorem 7.6 (Convergence factor of two-level algorithm)**.** If there exists $\delta > 0$ such that

$$\big\|(I - SA)e\big\|_A^2 \leqslant \|e\|_A^2 - \delta\big\|(I - \Pi_c)e\big\|_A^2, \quad \forall\, e \in V, \tag{7.7}$$

then the V$(0, 1)$ two-grid method satisfies that

$$\|E_{\mathrm{TG}}\|_A^2 = 1 - \hat{\delta} \quad \text{with} \quad \hat{\delta} := \inf_{(I-\Pi_c)e\neq 0} \frac{\|e\|_A^2 - \|(I - SA)e\|_A^2}{\|(I - \Pi_c)e\|_A^2} \geqslant \delta.$$

Proof. Notice that $\|e\|_A^2 = \|\Pi_c e\|_A^2 + \|(I - \Pi_c)e\|_A^2$ because Π_c is an A-orthogonal projection. Since $(I - \Pi_c)e = 0$ yields $(I - SA)(I - \Pi_c)e = 0$ as well, we have

$$\|E_{\mathrm{TG}}\|_A^2 = \sup_{e\neq 0} \frac{\|(I - SA)(I - \Pi_c)e\|_A^2}{\|e\|_A^2} = \sup_{(I-\Pi_c)e\neq 0} \frac{\|(I - SA)(I - \Pi_c)e\|_A^2}{\|(I - \Pi_c)e\|_A^2 + \|\Pi_c e\|_A^2}.$$

If $\hat{e}$ achieves the above supremum, then $(I - \Pi_c)\hat{e}$ also achieves the supremum because

$$\frac{\|(I - SA)(I - \Pi_c)^2\hat{e}\|_A^2}{\|(I - \Pi_c)^2\hat{e}\|_A^2 + \|\Pi_c(I - \Pi_c)\hat{e}\|_A^2} = \frac{\|(I - SA)(I - \Pi_c)\hat{e}\|_A^2}{\|(I - \Pi_c)\hat{e}\|_A^2} \geqslant \frac{\|(I - SA)(I - \Pi_c)\hat{e}\|_A^2}{\|\hat{e}\|_A^2}.$$

So the convergence factor achieves the supremum when $\Pi_c\hat{e} = 0$. That is to say, from the definition (7.6),

$$\big\|E_{\mathrm{TG}}\big\|_A^2 = \sup_{(I-\Pi_c)e\neq 0} \frac{\big\|(I - SA)e\big\|_A^2}{\big\|(I - \Pi_c)e\big\|_A^2}.$$

Hence the result. □

Note that, if we further assume that the parameter $\hat{\delta}$ in Theorem 7.6 is bounded uniformly on all levels, we can also obtain a uniform bound for the V-cycle convergence factor by recursion [84]. This bound also gives reasonable estimates numerically [87].

Basically, the assumption (7.7) implies that the smoother S is efficient for the components that cannot be treated by CGC efficiently. On the one hand, for the error components that cannot be reduced by CGC, the smoother S must be effective uniformly; on the other hand, for the error components that can be reduced by CGC efficiently, S is allowed to be ineffective. The components for which S is ineffective are called *smooth* and they have to be in the range of the interpolation, range(P), roughly. So (7.7) is natural to assume in order to get an efficient TG algorithm.

However, such a $\hat{\delta}$ is difficult to obtain in practice and we need to give some positive lower bounds of $\hat{\delta}$. So we introduce a nonnegative function $g(e) \geqslant 0$ and define

$$\alpha_g(e) := \frac{\|e\|_A^2 - \|(I-SA)e\|_A^2}{g(e)} \quad \text{and} \quad \beta_g(e) := \frac{\|(I-\Pi_c)e\|_A^2}{g(e)}.$$

Let $\hat{\alpha}_g := \inf_{g(e)\neq 0} \alpha_g(e)$ and $\hat{\beta}_g := \sup_{g(e)\neq 0} \beta_g(e)$. Due to the fact

$$\begin{aligned}
\|E_{\mathrm{TG}}e\|_A^2 &\leqslant \big\|(I-\Pi_c)e\big\|_A^2 - \hat{\alpha}_g g\big((I-\Pi_c)e\big) \\
&\leqslant \big\|(I-\Pi_c)e\big\|_A^2 - \hat{\alpha}_g\hat{\beta}_g^{-1}\big\|(I-\Pi_c)e\big\|_A^2 \\
&= \Big(1-\hat{\alpha}_g\hat{\beta}_g^{-1}\Big)\big\|(I-\Pi_c)e\big\|_A^2 \qquad (7.8)\\
&\leqslant \Big(1-\hat{\alpha}_g\hat{\beta}_g^{-1}\Big)\|e\|_A^2, \qquad (7.9)
\end{aligned}$$

we have $\hat{\delta} \geqslant \hat{\alpha}_g\hat{\beta}_g^{-1}$, i.e.,

$$\|E_{\mathrm{TG}}\|_A^2 \leqslant 1 - \hat{\alpha}_g\hat{\beta}_g^{-1}.$$

In view of the above estimate, we can give two separate assumptions:

$$\big\|(I-SA)e\big\|_A^2 \leqslant \|e\|_A^2 - \bar{\alpha}_g\, g(e), \quad \forall\, e \in V, \tag{7.10}$$

and

$$\exists\, \bar{\beta}_{g,\mathrm{s}}, \text{ such that } \big\|(I-\Pi_c)e\big\|_A^2 \leqslant \bar{\beta}_{g,\mathrm{s}}\, g(e), \quad \forall\, e \in V. \tag{7.11}$$

The condition (7.10) is a smoothing property and the condition (7.11) is a type of approximation property. The condition (7.11) is oftentimes called the *strong approximation assumption*. In view of (7.8), we can further weaken this condition and assume the *weak approximation assumption*:

$$\exists\, \bar{\beta}_{g,\mathrm{w}}, \text{ such that } \big\|(I-\Pi_c)e\big\|_A^2 \leqslant \bar{\beta}_{g,\mathrm{w}}\, g\big((I-\Pi_c)e\big), \quad \forall\, e \in V. \tag{7.12}$$

From the above analysis, we can easily deduce the following theorem.

Theorem 7.7 (Convergence estimate of two-level AMG)**.** If (7.10) and (7.11) (or its weaker version (7.12)) hold, then V(0, 1) two-grid method satisfies

$$\|E_{\mathrm{TG}}\|_A^2 \leqslant 1 - \bar{\alpha}_g\, \bar{\beta}_g^{-1}.$$

Remark 7.8 (Strong and weak approximation properties)**.** The strong approximation assumption (7.11) can be used to show convergence of V-cycle AMG methods via a recursion [84, 102]. But the weak approximation assumption (7.12) is not sufficient for V-cycle to converge [31]. □

It is worthy to notice that, even if we are able to provide simple conditions on coarsening such that the approximation assumptions hold to obtain a convergent two-level or multilevel AMG method, it is still not clear how to provide an algorithm to meet these assumptions based on pure algebraic information. Actually it is difficult to do so in a strict sense. The coarsening process consists of identifying coarse variables and constructing prolongation matrices. And these two procedures are usually coupled together. In the rest of this chapter, we shall discuss more practical steps on constructing the coarsening space.

Interpolation operators

Now the question is how to choose such a function $g(e)$? Furthermore, how to apply Theorem 7.7 to enforce convergence conditions on the prolongation (or interpolation) matrix P to guarantee good AMG performance?

In case $g(e) := \|(I - \Pi_c)e\|_A^2$, we have $\hat{\alpha}_g = \hat{\delta}$ and $\hat{\beta}_g \equiv 1$. Another possible choice is given by Ruge and Stüben [102]:

$$g(e) := \|e\|_{AD^{-1}A}^2.$$

In this case, by definition, the strong approximation assumption (7.11) can be rewritten as

$$\inf_{e_c \in V_c} \|e - Pe_c\|_A^2 \leqslant \bar{\beta}_{\mathrm{s}} \|e\|_{AD^{-1}A}^2, \quad \forall\, e \in V. \tag{7.13}$$

On the other hand, we have

$$\begin{aligned} \|(I - \Pi_c)e\|_A^2 &= \big((I - \Pi_c)e, (I - \Pi_c)e\big)_A = \big((I - \Pi_c)e, (I - \Pi_c)e - Pe_c\big)_A \\ &\leqslant \|(I - \Pi_c)e\|_{AD^{-1}A} \cdot \|(I - \Pi_c)e - Pe_c\|_D \end{aligned}$$

If we assume, instead of (7.13), that

$$\inf_{e_c \in V_c} \|e - Pe_c\|_D^2 \leqslant \bar{\beta}_{\mathrm{w}} \|e\|_A^2, \quad \forall\, e \in V, \tag{7.14}$$

then

$$\begin{aligned} \|(I - \Pi_c)e\|_A^2 &\leqslant \|(I - \Pi_c)e\|_{AD^{-1}A} \cdot \|(I - \Pi_c)e - Pe_c\|_D \\ &\leqslant \|(I - \Pi_c)e\|_{AD^{-1}A} \cdot \bar{\beta}_{\mathrm{w}}^{\frac{1}{2}} \|(I - \Pi_c)e\|_A, \end{aligned}$$

which yields the weak approximation property (7.12). In this way, we obtained two alternative bounds (7.13) and (7.14) for the strong and weak approximation assumptions, respectively. Using (7.14), we can also get convergence bound for the two-level method.

As we mentioned before, the weak approximation property (7.14) is usually not sufficient to guarantee a good interpolation P for V-cycle. More conditions shall be enforced for practical construction of AMG methods.

Let $Q \in \mathbb{R}^{N\times N}$ be a projection onto range(P). So, by definition, it can be written as $Q = PR$, where $R \in \mathbb{R}^{N_c\times N}$ satisfies $RP = I_c$. If $R_{\rm s} := (P^TAP)^{-1}P^TA$, then it is easy to see that $Q_{\rm s} = PR_{\rm s} = \Pi_c$ is such an example. We can also give a simplified choice $R_{\rm w} := (P^TDP)^{-1}P^TD$. For any vector $0 \neq e \in V$, we can assume that

$$\inf_{e_c\in V_c} \frac{\|e - Pe_c\|_A^2}{\|e\|_{AD^{-1}A}^2} \leqslant \frac{\|e - Qe\|_A^2}{\|e\|_{AD^{-1}A}^2} \leqslant \bar{\beta}_{\rm s} \quad \text{and} \quad \inf_{e_c\in V_c} \frac{\|e - Pe_c\|_D^2}{\|e\|_A^2} \leqslant \frac{\|e - Qe\|_D^2}{\|e\|_A^2} \leqslant \bar{\beta}_{\rm w}, \tag{7.15}$$

to give upper bounds for the strong and weak approximation assumptions, respectively. These inequalities give bounds for constructing P such that the two-level method converges according to Theorem 7.7.

We notice that, in the above inequality, the measure like

$$\mu_D(Q, e) := \frac{\|(I - Q)e\|_D^2}{\|e\|_A^2}, \quad \forall\, e \neq 0$$

can be generalized to

$$\mu_X(Q, e) := \frac{\|(I - Q)e\|_X^2}{\|e\|_A^2}, \quad \forall\, e \neq 0,$$

where X is an SPD matrix. We assume that $\mu_X(Q, e) \leqslant \kappa$. Then

$$\inf_{e_c\in V_c} \frac{\|e - Pe_c\|_X^2}{\|e\|_A^2} \leqslant \frac{\|e - Qe\|_X^2}{\|e\|_A^2} = \mu_X(Q, e) \leqslant \kappa. \tag{7.16}$$

If we minimize $\sup_{e\neq 0} \mu_X(PR, e)$ to find the "best possible" interpolation operator P, then it is the so-called *ideal interpolation* [55, 127].

In particular, if $X = \overline{S}^{-1}$, then $\bar{\alpha} \equiv 1$ for the smoothing assumption and the convergence factor of TG is bounded by

$$\|E_{\rm TG}\|_A^2 \leqslant 1 - \frac{1}{\kappa}. \tag{7.17}$$

Algebraic smooth error

In §3.4 (Theorem **??**, in particular), we have seen the following theoretical result: For any given smoother S, the best coarse space of dimension N_c is given by

$$V_c^{\rm opt} := \text{span}\{\phi_k\}_{k=1}^{N_c}, \tag{7.18}$$

where $\{\phi_k\}_{k=1}^{N_c}$ are the eigenfunctions corresponding to the smallest eigenvalues $\lambda_k(\overline{S}A)$. So the *desirable coarse space* should well approximate the lower end of the spectrum of $\overline{S}A$, which can

also be called the *near-null space*. However, it is difficult to find small eigenvalues of $\overline{S}A$ in practice.

A good interpretation of smooth error in algebraic sense could lead to an efficient AMG method. In view of (3.26), we know that the standard pointwise relaxation methods, like the Richardson, weighted Jacobi, and Gauss–Seidel methods, satisfy that

$$\rho_A^{-1}(v,v)_A \lesssim (\overline{S}Av,v)_A \lesssim (v,v)_A.$$

Together with (7.10), it motivates the following definition of the algebraic smooth vector:

Definition 7.9 (Algebraic smoothness)**.** Let $\varepsilon \in (0,1)$ be a small parameter. If $e \in V$ satisfies

$$(\overline{S}Ae,e)_A \leqslant \varepsilon (e,e)_A,$$

then e is algebraically ε-smooth (or the ε-algebraic low-frequency) with respect to A.

By adding and subtraction and (2.13), we immediately have

$$\Big((I-\overline{S}A)e,e\Big)_A \geqslant (1-\varepsilon)(e,e)_A \quad \Longrightarrow \quad \frac{\|(I-SA)e\|_A^2}{\|e\|_A^2} \geqslant 1-\varepsilon.$$

Apparently, the contraction factor for this error component e is close 1 if ε is small. Basically, this means the algebraically smooth error components are those which the smoother S or $\overline{S}$ cannot damp efficiently. That is to say, an error cannot be eliminated by the smoother is a smooth error; see Remark 1.24 for geometric smooth error.

Since $\overline{S}$ is SPD, the algebraically smooth vectors satisfy

$$\|e\|_A^2 = (\overline{S}^{\frac{1}{2}}Ae, \overline{S}^{-\frac{1}{2}}e) \leqslant (\overline{S}Ae,Ae)^{1/2}\,(\overline{S}^{-1}e,e)^{1/2} \leqslant \varepsilon^{1/2}\|e\|_A(\overline{S}^{-1}e,e)^{1/2}.$$

Then we can derive the following estimate

$$\|e\|_A^2 \leqslant \varepsilon \|e\|_{\overline{S}^{-1}}^2, \tag{7.19}$$

which can be viewed as an alternative characterisation of algebraically smooth vectors. Similar to algebraic low-frequency components, we can define algebraic high-frequency as follows

Definition 7.10 (Algebraic high-frequency)**.** Let $\zeta \in (0,1]$. If $e \in V$ satisfies

$$\|e\|_A^2 \geqslant \zeta \|e\|_{\overline{S}^{-1}}^2,$$

then e is called the ζ-algebraic high-frequency vector with respect to A.

With this notion, we can obtain the following convergence estimate [124]:

Theorem 7.11 (Convergence estimate based on space decomposition)**.** Let $V_c \subset V$ be the coarse space and V_{hf} consist of ζ-algebraic high frequencies. Suppose $V = V_c + V_{\text{hf}}$ is a stable decomposition, i.e., for any $v \in V$, there exist $v_c \in V_c$ and $v_{\text{hf}} \in V_{\text{hf}}$ such that $v = Pv_c + v_{\text{hf}}$ and $\|v_{\text{hf}}\|_A^2 \leqslant \beta \|v\|_A^2$. Then the resulting two-level AMG satisfies

$$\|E_{\text{TG}}\|_A \leqslant 1 - \zeta\beta^{-1}.$$

Proof. Since we have the following estimate

$$\inf_{w_c \in V_c} \|v - Pw_c\|_{\overline{S}^{-1}}^2 \leqslant \|v_{\text{hf}}\|_{\overline{S}^{-1}}^2 \leqslant \frac{1}{\zeta}\|v_{\text{hf}}\|_A^2 \leqslant \frac{\beta}{\zeta}\|v\|_A^2,$$

we can prove the theorem using (7.17). □

Remark 7.12 (Local adaptation of AMG)**.** In AMG methods, it is not important whether S smooths the error in any geometric sense or not. On the contrary, the key point is that the error after smoothing sweeps can be characterized algebraically to a degree which makes it possible to construct coarse levels and define interpolations which are locally adapted to the properties of the given smoother. □

Remark 7.13 (Smooth error and Classical AMG)**.** A simpler characterization of smooth error is used in methods like the Classical AMG. If the vector e corresponds to the low-end of eigenvalues, then we have $Ae \ll 1$ in the entry-by-entry sense. According to (7.1) and (7.19), the algebraically smooth error e satisfies that

$$(Ae, e) = \sum_{i<j} -a_{i,j}(e_i - e_j)^2 \ll 1. \tag{7.20}$$

This inequality provides an important motivation for the Classical AMG: Smooth error varies slowly in the direction of relatively large (negative) coefficients of the matrix. And it motivates the notion of strongly negative coupled variables. □

Construction of coarse spaces

We now discuss a few guidelines on how to construct coarse spaces and prolongation matrices based on the AMG theory developed above. In §6.1, we have discussed a general procedure of multigrid setup phase. The coarsening algorithms are automatic procedures for determining the coarse-level variables. Such algorithms are usually based on selecting or combining vertices in the adjacency graph corresponding to the (filtered) coefficient matrix A. We shall discuss more concrete examples of coarsening algorithms in the following sections.

A natural choice of the coarse-level DOFs is to use a subset of fine-level DOFs. Under proper re-ordering (coarse variables first and then fine variables) $R = (I, 0) \in \mathbb{R}^{N_c \times N}$. According to

Theorem **??**, we can use the diagonal matrix $D \in \mathbb{R}^{N\times N}$ of A to analyze the smoother S defined by the point-wise Gauss–Seidel method. This result motivates that we should construct a coarse space, such that

$$\|v - \mathcal{Q}_D v\|_D^2 = \inf_{v_c \in V_c} \|v - v_c\|_D^2 \leqslant \beta \|v\|_A^2, \quad \forall\, v \in V,$$

where the constant β should be small and uniform with respect to interested parameters (like the meshsize h). If v is smooth, i.e., $\|\nabla v\|$ is small, then v can be approximated well in the coarse space V_c. This condition is a sufficient condition for the convergence of the two-grid method. Motivated by Lemma 3.31, we can further simplify it and just choose $D := \|A\|I$, for example.

Heuristically, the error becomes quite smooth after a few relaxation steps and we can expect the coarse space can approximate a smooth vector v rather accurately if the coarse space is chosen appropriately. Motivated by Theorem 7.7 and (7.15), we give Assumption 7.14, which is equivalent to that V_c reproduces local constant.

Assumption 7.14 (Weak approximability)**.** $\|(I - PR)v\|_D \leqslant \beta \|v\|_A, \ \ \forall\, v \in V.$

In view of Remark 3.41, we assume that the prolongation operator preserves the constant (Assumption 7.15). In fact, from the weak approximation property (Assumption 7.14) and let $D := \|A\|I$, we have

$$\|A\|^{1/2} \|v - PR\, v\| \leqslant \beta \|v\|_A.$$

If v is in the near-null space of A, i.e., $\|v\|_A \approx 0$, then $PR\, v \approx v$. Hence we get the following simplified assumption:

Assumption 7.15 (Constant preserving)**.** $P\mathbf{1}_{N_c} = \mathbf{1}_N$.

Unlike in the geometric setting, it is not meaningful to only look at the convergence in the algebraic multigrid context. This is because, the computation complexity of each AMG cycle could be prohibitively large; compare with GMG complexity discussed in §6.2. Even a uniformly convergent AMG method could be very slow. Hence complexity of the multilevel hierarchy for AMG is crucial.

Remark 7.16 (Operator complexity)**.** When constructing the prolongation P, we must control the sparsity of the coarse level matrices. For efficient overall performance, convergence speed is only one aspect. An equally important aspect is the complexity (sparsity) of the coarser level matrices produced by AMG. We now define a measurement of sparsity, i.e., the operator complexity

$$C_A := \frac{\sum_{l=0}^{L} \operatorname{nnz}(A_l)}{\operatorname{nnz}(A)},$$

where nnz(·) is the number of nonzeros of a matrix. Apparently, $C_A \geqslant 1$ is always true and $C_A = 1$ corresponds to the one-level case. When constructing an interpolation operator, we would like to make C_A as close to 1 as possible while keeping good convergence performance. This is not always the case when using the Galerkin-type coarse operator as we discussed in this note. Usually, the coarser matrices A_{l-1} becomes more dense than A_l. This problem becomes more serious when solving very large linear systems. Sometimes, we have to truncate the "insignificant" nonzero entries or specify sparse patterns to maintain low complexity [54]. □

7.3 Classical algebraic multigrid methods

The original AMG [35] idea (the classical AMG) was developed under the assumption that such a problem with A being an M-matrix was solved. The multilevel hierarchy is constructed based on the coefficient matrix only. Later, the AMG algorithm was further generalized using many heuristics that served to extend its applicability to more general problems. For simplicity, we suppose that $A = (a_{i,j}) \in \mathbb{R}^{N\times N}$ is an SPD M-matrix and $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ is the corresponding graph of A.

General AMG setup phase

We have presented a rather abstract framework for multigrid methods in §7.1. Now we give a general two-level setup phase suitable for AMG methods (including the classical AMG and aggregation-based AMG methods). This algorithm can be applied recursively to obtain a multilevel hierarchy until N_c is small enough or A_c is too dense to continue.

Algorithm 7.2 (General algebraic setup algorithm)**.** Given a sparse matrix $A \in \mathbb{R}^{N\times N}$.

1. Filter A to obtain a suitable matrix for coarsening A_f (usually $A_f = A$);
2. Define coarse space with N_c variables;
3. Construct the interpolation $P \in \mathbb{R}^{N\times N_c}$:
   - 3.1. Give a sparsity pattern for the interpolation P;
   - 3.2. Determine weights of the interpolation P;
4. Construct the restriction $R \in \mathbb{R}^{N_c\times N}$ (for example, $R = P^T$);
5. Form the coarse-level coefficient matrix (for example, $A_c = RA_fP$);
6. Give a sparser approximation of A_c if necessary.

The above framework is abstract and general enough to describe a variety of algorithms. Now we give a few comments on this algorithm:

1. If the coefficient matrix A is not symmetric or not an M-matrix, we might want to perform a preprocessing step to get a more suitable matrix A_f. This step can also be used as a way to introduce auxiliary space method.

2. In the classical AMG methods, we use the so-called C/F splitting, namely, split all N variables into N_c C-variables and N_f F-variables, i.e., $N = N_c + N_f$; on the other hand, in the aggregation-based AMG, we form aggregates of F-variables.

3. As we observed before, forming an interpolation P that satisfies the weak approximation property is crucial for convergence. This task can be further divided into two stages: (1) giving sparsity pattern and (2) determine weights of P. Sometimes, we can truncate some of the small entries if it is not sparse enough.

4. For symmetric problems, it is natural to use the Galerkin relation to assume $R = P^T$. But for nonsymmetric problems, we might need to construct R as well.

5. In GMG, the coarse-level problems can also be given by discretization on a coarser grid. But in AMG, we must use the restriction, interpolation, and the fine-level coefficient matrices to compute A_c using the triple-matrix product, which can easily become the most time-consuming part of the setup phase. Implementation of this part requires attention, especially for parallel efficiency.

6. Sometimes, A_c might not be sparse enough even after P is truncated when the Galerkin relation is employed. In this case, one might have to further modify A_c to obtain a sparser approximation.

Strength of connections

In coarsening, we need to find coarse-level variables. Let $\theta_{\text{str}} \in (0, 1)$ be a given real number, usually called *relative strength* parameter. In view of Remark 7.13, we give the following definition: If a pair of indices (i, j) satisfies that

$$-a_{i,j} \geqslant \theta_{\text{str}} \big| \min_k a_{i,k} \big|,$$

then this pair is called *strongly negatively coupled* or *strongly n-coupled*. More precisely, we say that the variable i is strongly negatively coupled to the variable j. Note that, by this definition, (i, j) and (j, i) are two different pairs. We can easily generalize this concept to *strongly coupling* by considering the positive coupling.

Remark 7.17 (Alternative definitions for strong coupling)**.** There are different ways to define strongly coupled pairs. For example, we can call i and j strongly negatively coupled, if

$$a_{i,j} < 0 \quad \text{and} \quad |a_{i,j}| > \theta_{\text{str}} \sqrt{a_{i,i} a_{j,j}}$$

or

$$-a_{i,j} > \theta_{\text{str}} \sqrt{a_{i,i} a_{j,j}}.$$

Such definitions will be used to define aggregation-based methods in the next section. □

Denote further

$$\mathsf{S}_i := \{ j \in \mathsf{N}_i : \ j \ \text{ strongly coupled to } \ i \} \quad \text{and} \quad \mathsf{S}_i^T := \{ j \in \mathsf{V} : \ i \in \mathsf{S}_j \}.$$

So S_i is the set of indices which *affects* i and S_i^T is the ones which are *affected* by i. After finding the strongly coupled variables, we can filter the coefficient matrix to obtain a filtered matrix A_{S} by removing non-strongly coupled connections.

The above definition of strongly coupled variables applies to the direct connections. Sometimes we also need to consider indirect (i.e., long-range) connections; for example, in aggressive coarsening (see Remark 7.20). A variable i is said *strongly connected* to another variable j along a path of length l if there exists a sequence of edges $\{(i, j_1), (j_1, j_2), \ldots, (j_{l-2}, j_{l-1}), (j_{l-1}, j)\} \subseteq \mathsf{E}$ such that $j_{k+1} \in \mathsf{S}_{j_k}$. If there exist at least one path of length less than or equal to ℓ such that i strongly connects to j, then we say that i is ℓ-strongly connects to j and denoted by $j \in \mathsf{S}_i^\ell$.

We note that, based on the nonzero pattern of the original matrix A^ℓ or a filtered matrix A_{S}^ℓ, one can tell whether there are paths between i and j of length ℓ or not. For example, if we consider five-point stencil finite difference scheme on the mesh given in Figure 7.5 (left). Consider the vertex at the center, the point 13. Then

$$\mathsf{S}_{13} = \{12, 8, 14, 18\} \quad \text{and} \quad \mathsf{S}_{13}^2 = \{12, 8, 14, 18, 11, 3, 15, 23, 7, 9, 19, 17\}.$$

And we give the weights of A and A^2 in Figure 7.5. See Figures 7.6, 7.7, and 7.8 for details.

C/F splitting

The classical Ruge-Stüben method is to split the set of vertices V to a sum of two non-intersecting sets, the fine variables F and the coarse variables C, such that all the indices in F will be affected by some index in C, while C is expected to contain as few entries as possible. Then F will be chosen as the set of indices of finer grid nodes, and C will be chosen as the set of indices of coarse grid nodes. The indices of nodes are assigned to be coarse or fine successively. Denote by U the set of indices of nodes that have not been assigned yet, and we summarize the algorithm in the following subroutine:

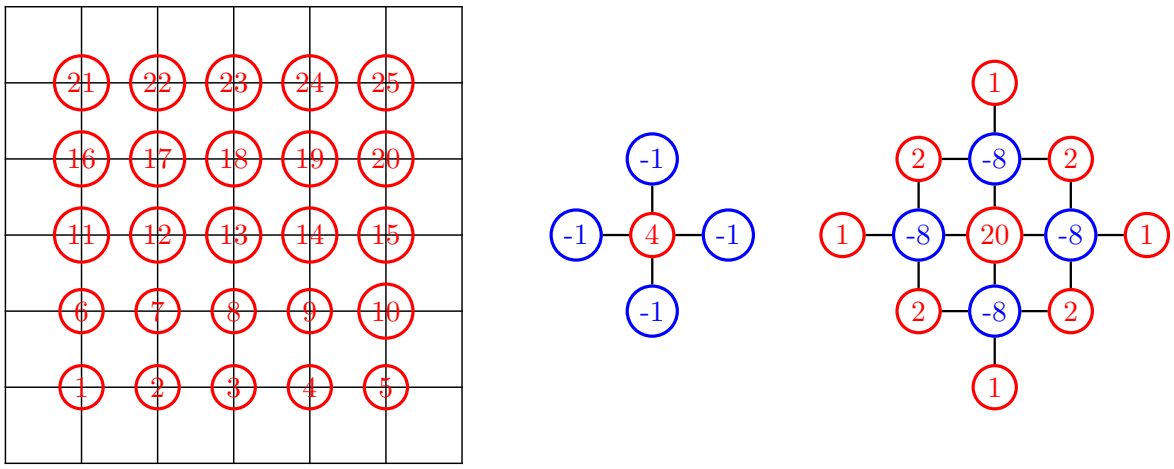


Figure 7.5: Finite difference grid (left), strong connections and weights in A of vertex 13 (middle), and 2-strong connections and weights in A^2 of 13 (right).

$A =$

$$\begin{bmatrix}
4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 4
\end{bmatrix}$$

Figure 7.6: The stiffness matrix A for five-point stencil finite difference scheme on the mesh given in Figure 7.5 (left).

```
>> A*A

ans =

    18    -8     1     0     0    -8     2     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
    -8    19    -8     1     0     2    -8     2     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0
     1    -8    19    -8     1     0     2    -8     2     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0
     0     1    -8    19    -8     0     0     2    -8     2     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0
     0     0     1    -8    18     0     0     0     2    -8     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0
    -8     2     0     0     0    19    -8     1     0     0    -8     2     0     0     0     1     0     0     0     0     0     0     0     0     0
     2    -8     2     0     0    -8    20    -8     1     0     2    -8     2     0     0     0     1     0     0     0     0     0     0     0     0
     0     2    -8     2     0     1    -8    20    -8     1     0     2    -8     2     0     0     0     1     0     0     0     0     0     0     0
     0     0     2    -8     2     0     1    -8    20    -8     0     0     2    -8     2     0     0     0     1     0     0     0     0     0     0
     0     0     0     2    -8     0     0     1    -8    19     0     0     0     2    -8     0     0     0     0     1     0     0     0     0     0
     1     0     0     0     0    -8     2     0     0     0    19    -8     1     0     0    -8     2     0     0     0     1     0     0     0     0
     0     1     0     0     0     2    -8     2     0     0    -8    20    -8     1     0     2    -8     2     0     0     0     1     0     0     0
     0     0     1     0     0     0     2    -8     2     0     1    -8    20    -8     1     0     2    -8     2     0     0     0     1     0     0
     0     0     0     1     0     0     0     2    -8     2     0     1    -8    20    -8     0     0     2    -8     2     0     0     0     1     0
     0     0     0     0     1     0     0     0     2    -8     0     0     1    -8    19     0     0     0     2    -8     0     0     0     0     1
     0     0     0     0     0     1     0     0     0     0    -8     2     0     0     0    19    -8     1     0     0    -8     2     0     0     0
     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     0    -8    20    -8     1     0     2    -8     2     0     0
     0     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     1    -8    20    -8     1     0     2    -8     2     0
     0     0     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     1    -8    20    -8     0     0     2    -8     2
     0     0     0     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     1    -8    19     0     0     0     2    -8
     0     0     0     0     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     0    -8    18    -8     1     0     0
     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     0    -8    19    -8     1     0
     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     1    -8    19    -8     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     2    -8     2     0     1    -8    19    -8
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     2    -8     0     0     1    -8    18
```

Figure 7.7: The matrix A^2 for five-point stencil finite difference scheme on the mesh given in Figure 7.5 (left).

```
>> A*A*A

ans =

    88   -53    12    -1     0   -53    24    -3     0     0    12    -3     0     0     0    -1     0     0     0     0     0     0     0     0     0
   -53   100   -54    12    -1    24   -56    24    -3     0    -3    12    -3     0     0     0    -1     0     0     0     0     0     0     0     0
    12   -54   100   -54    12    -3    24   -56    24    -3     0    -3    12    -3     0     0     0    -1     0     0     0     0     0     0     0
    -1    12   -54   100   -53     0    -3    24   -56    24     0     0    -3    12    -3     0     0     0    -1     0     0     0     0     0     0
     0    -1    12   -53    88     0     0    -3    24   -53     0     0     0    -3    12     0     0     0     0    -1     0     0     0     0     0
   -53    24    -3     0     0   100   -56    12    -1     0   -54    24    -3     0     0    12    -3     0     0     0    -1     0     0     0     0
    24   -56    24    -3     0   -56   112   -57    12    -1    24   -57    24    -3     0    -3    12    -3     0     0     0    -1     0     0     0
    -3    24   -56    24    -3    12   -57   112   -57    12    -3    24   -57    24    -3     0    -3    12    -3     0     0     0    -1     0     0
     0    -3    24   -56    24    -1    12   -57   112   -56     0    -3    24   -57    24     0     0    -3    12    -3     0     0     0    -1     0
     0     0    -3    24   -53     0    -1    12   -56   100     0     0    -3    24   -54     0     0     0    -3    12     0     0     0     0    -1
    12    -3     0     0     0   -54    24    -3     0     0   100   -56    12    -1     0   -54    24    -3     0     0    12    -3     0     0     0
    -3    12    -3     0     0    24   -57    24    -3     0   -56   112   -57    12    -1    24   -57    24    -3     0    -3    12    -3     0     0
     0    -3    12    -3     0    -3    24   -57    24    -3    12   -57   112   -57    12    -3    24   -57    24    -3     0    -3    12    -3     0
     0     0    -3    12    -3     0    -3    24   -57    24    -1    12   -57   112   -56     0    -3    24   -57    24     0     0    -3    12    -3
     0     0     0    -3    12     0     0    -3    24   -54     0    -1    12   -56   100     0     0    -3    24   -54     0     0     0    -3    12
    -1     0     0     0     0    12    -3     0     0     0   -54    24    -3     0     0   100   -56    12    -1     0   -53    24    -3     0     0
     0    -1     0     0     0    -3    12    -3     0     0    24   -57    24    -3     0   -56   112   -57    12    -1    24   -56    24    -3     0
     0     0    -1     0     0     0    -3    12    -3     0    -3    24   -57    24    -3    12   -57   112   -57    12    -3    24   -56    24    -3
     0     0     0    -1     0     0     0    -3    12    -3     0    -3    24   -57    24    -1    12   -57   112   -56     0    -3    24   -56    24
     0     0     0     0    -1     0     0     0    -3    12     0     0    -3    24   -54     0    -1    12   -56   100     0     0    -3    24   -53
     0     0     0     0     0    -1     0     0     0     0    12    -3     0     0     0   -53    24    -3     0     0    88   -53    12    -1     0
     0     0     0     0     0     0    -1     0     0     0    -3    12    -3     0     0    24   -56    24    -3     0   -53   100   -54    12    -1
     0     0     0     0     0     0     0    -1     0     0     0    -3    12    -3     0    -3    24   -56    24    -3    12   -54   100   -54    12
     0     0     0     0     0     0     0     0    -1     0     0     0    -3    12    -3     0    -3    24   -56    24    -1    12   -54   100   -53
     0     0     0     0     0     0     0     0     0    -1     0     0     0    -3    12     0     0    -3    24   -53     0    -1    12   -53    88
```

Figure 7.8: The matrix A^3 for five-point stencil finite difference scheme on the mesh given in Figure 7.5 (left).

Listing 7.1: Classical C/F splitting method

```
U ← V, C ← ∅, F ← ∅;
while U ≠ ∅
    λ_i ← 2|S_i^T ⋂ F| + |S_i^T ⋂ U|, i ∈ U;
    k ← arg max{λ_i, i ∈ U};
    C ← C ⋃ {k}, U ← U\{k};
    F ← F ⋃ S_k^T, U ← U\S_k^T;
end
```

Note that λ_i is a *measure of importance*—It is a measurement about how many points are affected by i. If λ_i is big, we would like to include this point in C. In this way, we can make C contains less points to get bigger coarsening ratio, which is very important for the classical AMG because it usually yields relatively small coarsening ratio.

- We weight more on $|\mathsf{S}_i^T \bigcap \mathsf{F}|$ than $|\mathsf{S}_i^T \bigcap \mathsf{U}|$ due to the fact that the first part has already been determined to be on the fine grid.
- In the early stage of coarsening procedure, F does not contain many points, the above algorithm selects a coarse point with as many as neighbors that strongly coupled to it.
- In the later stage, vertices that strongly coupled to many F-variables are preferred to be selected.

There are a few special cases which require careful treatment during the C/F splitting procedure. We now summarize them in the following remarks:

Remark 7.18 (Isolated points)**.** Before we start the above algorithm, we usually need to filter out the isolated points (like the Dirichlet boundary points) and define them as F-variables. Similarly, if a point has very strong diagonal dominance, we can also call them isolated and move them to F. These are the trivial cases. □

Remark 7.19 (Termination of C/F splitting)**.** If successfully terminated, the set C is an independent set of vertices of the underlying graph G. All F-variables have at least one strongly negatively coupled C-variable, except the trivial ones in the previous remark. However, there might be some U-variables left (with measure $\lambda_i = 0$)—They are not strongly negatively coupled to any C-variables or themselves. Furthermore, there are no F-variables are strongly negatively coupled to these points. In order to interpolate at these points, we can add them as F-variables and interpolate indirectly through the F-variables, to which they are strongly coupled. □

Remark 7.20 (Aggressive coarsening)**.** In practice, the standard C/F splitting scheme given above usually results in high operator complexity (refer to Remark 7.16), which leads to high

Coarsening method	Standard	Aggressive
Operator complexity	2.889	1.606
Setup time (sec)	1.536	1.036
Number of iterations	6	38
Solve time (sec)	0.791	3.293
Time per iteration (sec)	0.132	0.087

Table 7.1: Solving 2D five-point stencil of the Poisson's equation with 1 million DOF using different coarsening methods in the classical AMG method (stopping criteria for PCG is the relative residual smaller than 10^{-6}).

computational and storage demands; see Table 7.1. In such cases, we can apply the so-called aggressive coarsening by considering strong connections of length ℓ. Oftentimes a small ℓ, for example $\ell = 2$, is used. However, A_{S}^2 is expensive to compute and we can apply the regular C/F splitting twice—At the first pass, find C-variables among all variables using A_{S}; at the second pass, apply the C/F splitting on the selected C-variables from the first pass using A_{S}^2 (but on C only, we don't need all entries of A_{S}^2). □

Example 7.21 (Anisotropic elliptic PDE)**.** To illustrate the effect of the above C/F splitting algorithm, we consider an anisotropic diffusion example in §6.1. The computational domain is a unit square. Let us consider the anisotropic diffusion equation

$$-\epsilon u_{xx} - u_{yy} = 0 \quad (\epsilon > 0).$$

Roughly speaking, we have $\epsilon\|u_{xx}\| \approx \|u_{yy}\|$. This means the solution is smooth in y-direction (low-frequencies); but rough in x-direction (high-frequencies). We consider the five-point stencil. The difference equation at the node (x_i, y_j) is

$$-\epsilon\frac{2u_{i,j} - u_{i+1,j} - u_{i-1,j}}{h_x^2} - \frac{2u_{i,j} - u_{i,j-1} - u_{i,j+1}}{h_y^2} = 0.$$

If $\frac{\epsilon}{h_x^2} \ll \frac{1}{h_y^2}$, then $u_{i,j}$ depends on $u_{i,j+1}$ and $u_{i,j-1}$ only. Thus if we process the C/F procedure, the coarsening will take place indeed in one direction only (semi-coarsening); see Figure 7.9. □

Construction of prolongation

After obtaining a C/F splitting, upon a reordering of indices, we can always assume that the indices of the nodes in C is from 1 to N_c, and those in F are from $N_c + 1$ to N. We can write the stiffness matrix in the following block structure

$$\begin{pmatrix} A_{\mathsf{C},\mathsf{C}} & A_{\mathsf{C},\mathsf{F}} \\ A_{\mathsf{F},\mathsf{C}} & A_{\mathsf{F},\mathsf{F}} \end{pmatrix} \begin{pmatrix} u_{\mathsf{C}} \\ u_{\mathsf{F}} \end{pmatrix} = \begin{pmatrix} f_{\mathsf{C}} \\ f_{\mathsf{F}} \end{pmatrix}$$

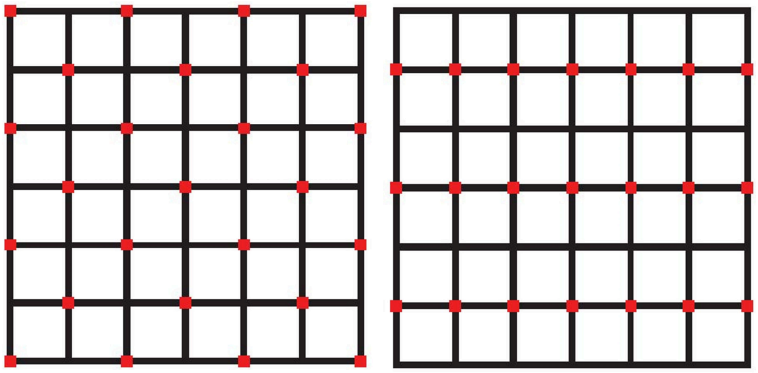


Figure 7.9: C/F splitting for the 2D elliptic problem with $\epsilon = 1$ (left) and $\epsilon \ll 1$ (right), where the red points are C-variables and the black points are F-variables.

Let $e^H \in \mathbb{R}^{N_c}$ be a vector corresponding to the variables on the coarse grid. We now consider how to prolongate it to $e^h \in \mathbb{R}^N$ corresponding to the variables on the fine grid.

We first use the geometric multigrid method for linear finite element method on uniform grids for the 1D Poisson's equation as an example. Let $\{\phi_i^h\}_{i=1}^N$ be the basis of the fine space V and $\{\phi_j^H\}_{j=1}^{N_c}$ be the basis of the coarse space V_c. From the geometrical multigrid point of view, it is natural to expect

$$a(\phi_j^H, \phi_i^h) = 0, \quad j \in \mathsf{C},\ i \in \mathsf{F}. \tag{7.21}$$

In fact, the fine grid (high-frequency) part can be captured by the fine grid approximation, i.e.,

$$a(u^h - \Pi_H u^h, \phi^h) = a(u^h, \phi^h),$$

if ϕ^h is a basis function corresponding to the difference between fine and coarse grid functions.

It is trivial to see that, we should have $(Pe^H)_j = e_j^H$, if $j \in \mathsf{C}$. Define

$$P := \begin{pmatrix} I_{\mathsf{C}} \\ Q \end{pmatrix},$$

where $I \in \mathbb{R}^{N_c \times N_c}$ is the identity matrix and $Q \in \mathbb{R}^{(N-N_c)\times N_c}$. In the matrix form, the condition (7.21) can be written as

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{\mathsf{F}} \end{pmatrix} \begin{pmatrix} A_{\mathsf{C},\mathsf{C}} & A_{\mathsf{C},\mathsf{F}} \\ A_{\mathsf{F},\mathsf{C}} & A_{\mathsf{F},\mathsf{F}} \end{pmatrix} \begin{pmatrix} I_{\mathsf{C}} \\ Q \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}.$$

That is to say, $A_{\mathsf{F},\mathsf{C}} + A_{\mathsf{F},\mathsf{F}}Q = \mathbf{0}$ or $Q = -A_{\mathsf{F},\mathsf{F}}^{-1}A_{\mathsf{F},\mathsf{C}}$. It is easy to check that this prolongation matrix P satisfies Assumption 7.15 if the row-sum of A is zero. However, this prolongation is too expensive to compute in practice and there are many different ways to approximate Q by a simpler sparse matrix W.

1) Direct interpolation scheme

For the error $e^h \in \mathbb{R}^N$, we have

$$A_{\mathsf{F},\mathsf{F}} e^h_{\mathsf{F}} + A_{\mathsf{F},\mathsf{C}} e^h_{\mathsf{C}} \ll \mathbf{1} \quad \Longrightarrow \quad \sum_{j=1}^{N} a_{i,j} e^h_j \approx 0, \quad i \in \mathsf{F}.$$

Motived by the above observation, we can assume

$$a_{i,i} e^h_i + \sum_{j \in \mathsf{N}_i} a_{i,j} e^h_j = 0, \quad i \in \mathsf{F}. \tag{7.22}$$

This would be an interpolation scheme itself if all points in N_i are C-variables. Of course, it is not always the case. Alternatively, we can throw out the entries that are not strongly negatively coupled and obtain

$$a_{i,i} e^h_i + \sum_{j \in \mathsf{S}_i} a_{i,j} e^h_j = 0, \quad i \in \mathsf{F}. \tag{7.23}$$

We approximate the above equation (7.22) with

$$a_{i,i} e^h_i + \alpha_i \sum_{j \in \mathsf{N}_i \bigcap \mathsf{C}} a_{i,j} e^h_j = 0, \quad \alpha_i = \frac{\sum_{k \in \mathsf{N}_i} a_{i,k}}{\sum_{k \in \mathsf{N}_i \bigcap \mathsf{C}} a_{i,k}}, \quad i \in \mathsf{F}.$$

If the i-th row has zero row-sum, then $\alpha_i = -\frac{a_{i,i}}{\sum_{k \in \mathsf{N}_i \bigcap \mathsf{C}} a_{i,k}}$ and we get an interpolation method

$$e^h_i = \sum_{j \in \mathsf{N}_i \bigcap \mathsf{C}} w_{i,j} e^H_j \qquad \text{and} \qquad w_{i,j} = \frac{a_{i,j}}{\sum_{k \in \mathsf{N}_i \bigcap \mathsf{C}} a_{i,k}}. \tag{7.24}$$

In this case, the matrix form is just $W = \big(\operatorname{diag}(A_{\mathsf{F},\mathsf{C}}\mathbf{1})\big)^{-1} A_{\mathsf{F},\mathsf{C}}$. It is straightforward to show that Assumption 7.15 holds in this case.

We can make W more sparse by shrinking the support slightly. Define an interpolation set (support) $\mathsf{P}_i := \mathsf{S}_i \bigcap \mathsf{C}$ for $i \in \mathsf{F}$. After further sparsifying the interpolation (by keeping the strongly negatively coupled C-variables only), we get

$$a_{i,i} e^h_i + \alpha_i \sum_{j \in \mathsf{P}_i} a_{i,j} e^h_j = 0, \quad \alpha_i = \frac{\sum_{k \in \mathsf{N}_i} a_{i,k}}{\sum_{k \in \mathsf{P}_i} a_{i,k}}, \quad \forall\, i \in \mathsf{F}.$$

If the i-th row has zero row-sum, then this gives the well-known direct interpolation

$$e^h_i = \sum_{j \in \mathsf{P}_i} w_{i,j} e^H_j \qquad \text{and} \qquad w_{i,j} = \frac{a_{i,j}}{\sum_{k \in \mathsf{P}_i} a_{i,k}}. \tag{7.25}$$

2) Standard interpolation scheme

In the equation (7.22), we can first eliminate all e^h_j for $j \in \mathsf{S}_i \bigcap \mathsf{F}$, using the j-th equation, by the approximation

$$e^h_j := -\frac{1}{a_{j,j}} \sum_{k \in \mathsf{N}_j} a_{j,k} e^h_k.$$

This results in a new equation for e_i^h:

$$\hat{a}_{i,i}e_i^h + \sum_{j\in\hat{\mathsf{N}}_i} \hat{a}_{i,j}e_j^h = 0, \quad i \in \mathsf{F},$$

with $\hat{\mathsf{N}}_i = \{j \neq i : \hat{a}_{i,j} \neq 0\}$. Define a new interpolation set $\hat{\mathsf{P}}_i = (\bigcup_{j\in\mathsf{S}_i\bigcap\mathsf{F}} \mathsf{S}_j)\bigcup(\mathsf{S}_i\bigcap\mathsf{C})$. Then we apply the above direct interpolation for this new equation and arrive at the so-called standard interpolation scheme.

3) Jacobi interpolation scheme

We can rewrite the equation (7.23) as

$$a_{i,i}e_i^h + \sum_{j\in\mathsf{P}_i} a_{i,j}e_j^H + \sum_{j\in\mathsf{S}_i\setminus\mathsf{P}_i} a_{i,j}e_j^h = 0, \quad i \in \mathsf{F}.$$

Therefore, in order to obtain an interpolation matrix Q, we just need to approximately solve the above equations for e_i^h ($i \in \mathsf{F}$). For example, we can just apply one Jacobi iteration using $\tilde{e}_j^h \approx \frac{\sum_{k\in\mathsf{P}_i} a_{i,k}e_k^h}{\sum_{k\in\mathsf{P}_i} a_{i,k}}$ as the initial guess of e_j^h, $j \in \mathsf{F}$ ($j \neq i$). Then the prolongation can be defined as

$$\begin{cases} e_i^h = e_i^H, & i \in \mathsf{C} \\ a_{i,i}e_i^h + \sum_{j\in\mathsf{P}_i} a_{i,j}e_j^H + \sum_{j\in\mathsf{S}_i\setminus\mathsf{P}_i} a_{i,j}\dfrac{\sum_{k\in\mathsf{P}_i} a_{i,k}e_k^h}{\sum_{k\in\mathsf{P}_i} a_{i,k}} = 0, & i \in \mathsf{F}. \end{cases} \tag{7.26}$$

This is the so-called Jacobi interpolation method.

Remark 7.22 (Some simple alternatives)**.** The biggest advantage of the above approach is that it is simple and local: For the i-th entry, we only need the information on the i-th row of the matrix. We can improve this prolongation matrix P using some straightforward modifications. For example, the initial guess for the same entry can be different for different entries; an alternative initial guess could be

$$\tilde{e}_j^h \approx \frac{\sum_{k\in\mathsf{P}_j} a_{j,k}e_k^h}{\sum_{k\in\mathsf{P}_j} a_{j,k}}, \quad j \in \mathsf{F}.$$

And a few more steps of Jacobi iteration might improve performance. □

Remark 7.23 (Initial guess of weights)**.** If the initial guess $W^{(0)}$ preserves constants, then we get

$$Q - W^{(k)} = \left(I - D_{\mathsf{F},\mathsf{F}}^{-1}A_{\mathsf{F},\mathsf{F}}\right)^k \left(Q - W^{(0)}\right).$$

Since both Q and $W^{(0)}$ preserves constants, all improved weights $W^{(k)}$ also preserve constants by iteration. □

7.4 Aggregation-based algebraic multigrid methods

In this section, we consider the aggregation-base AMG methods whose easy-to-implement feature has drawn a lot of attention recently. The idea is to sub-divide the set of vertices into non-intersecting sets (or aggregates), i.e., $\mathsf{V} = \bigcup_{j=1,\ldots,N_c} \mathsf{C}_j$. Each aggregate C_j corresponds to a coarser variable.

Unsmoothed aggregation AMG

There are several different sophisticated ways to form aggregates. In principle, any combinatorial graph partitioning algorithms can be applied to form aggregation. We first give a simple greedy algorithm to form such an aggregation based on the concept of maximum independent set discussed in §7.1.

Listing 7.2: A greedy aggregation method

```
1  N_c ← 0,  U ← V;
2  for i ∈ U
3     if N_i ⊆ U
4          N_c ← N_c + 1;
5          C_{N_c} ← {i} ⋃ N_i,  U ← U\C_{N_c};
6     end
7  end
```

It is possible to have some "left-over" vertices which do not belong to any aggregate after the above procedure. We can, for example, add them to their neighboring aggregates with least points.

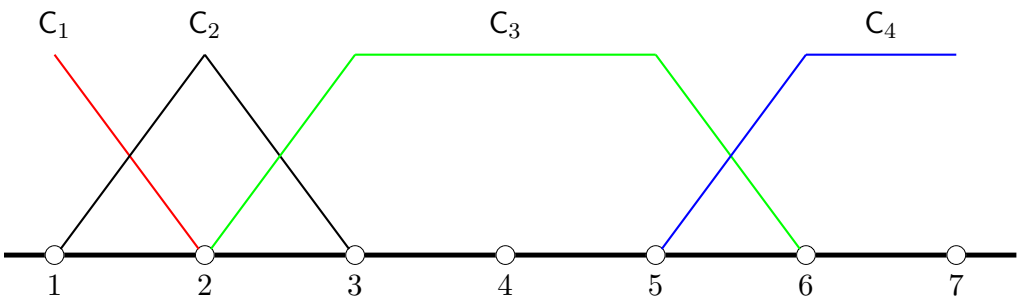


Figure 7.10: Aggregates and prolongation corresponding to (7.27).

Whence an aggregation is given, it is easy to define the prolongation matrix, for $1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant N_c$, by

$$(P)_{i,j} = \begin{cases} 1, & \text{if } i \in \mathsf{C}_j; \\ 0, & \text{if } i \notin \mathsf{C}_j. \end{cases}$$

With this interpolation, it is straight-forward to see that $P\mathbf{1}_{N_c} = \mathbf{1}_N$. We now give an example to explain P in one dimension. Let

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N \times N_c}. \tag{7.27}$$

Figure 7.10 shows the aggregation corresponding to the prolongation P in (7.27).

Of course, there are different ways to form aggregates and we give another approach here. The algorithm to construct coarse grid and prolongation based on the concept of strong coupling is

Listing 7.3: Another aggregation method

```
1  U ← V;
2  for i ∈ U
3     S_i ← {j ∈ U : j is strongly coupled to i};
4     construct a column of prolongation P based on S_i;
5     U ← U\({i} ⋃ S_i);
6  end
```

Smoothed aggregation AMG

The unsmoothed aggregation methods are very simple but usually converge slowly. There are two ways to improve their convergence behavior. One way is to employ a more complicated multilevel iteration, like the K-cycle multigrid method discussed in §6.2. And the other way is to enlarge the aggregates and smooth out the basis functions. The latter approach gives the smoothed aggregation AMG methods, which is based on the idea of minimizing the energy of the coarse basis functions among the set of all functions with same L^2-norm.

Assume that all variables are partitioned into non-overlapping subsets $\{\mathsf{C}_i\}_{i=1}^{N_c}$. We further assume that each C_i has at least one interior point, i.e., there exists an index $k_i \in \mathsf{C}_i$ such that $\big(A\big)_{k_i,j} = 0$ for any $j \notin \mathsf{C}_i$. Suppose that $\mathbf{1}$ is in the null space of A, namely, $A\mathbf{1} = \mathbf{0}$. Define a vector for each aggregate:

$$\mathbf{1}_i(x_j) := \begin{cases} \mathbf{1}(x_j), & \text{if } \ j \in \mathsf{C}_i; \\ 0, & \text{otherwise.} \end{cases}$$

Apparently, $\sum_i \mathbf{1}_i = \mathbf{1}$ and $\big(A\mathbf{1}_i\big)_{k_i} = 0$.

We now smooth out these piecewise basis functions by, for example, one step of weighted Jacobi iteration

$$\psi_i = (I - \omega D^{-1} A)\, \mathbf{1}_i.$$

Hence we have the partition of unity

$$\sum_i \psi_i = (I - \omega D^{-1} A) \sum_i \mathbf{1}_i = (I - \omega D^{-1} A)\, \mathbf{1} = \mathbf{1}.$$

Thus we can obtain

$$\mathbf{1}(x_{k_i}) = \sum_j \psi_j(x_{k_i}) = \sum_j (I - \omega D^{-1} A)\, \mathbf{1}_i(x_{k_i}) = \mathbf{1}_i(x_{k_i}) - \omega D^{-1} A\, \mathbf{1}_i(x_{k_i}),$$

which implies that $D^{-1} A\, \mathbf{1}_i(x_{k_i}) = 0$ and $\psi_i(x_{k_i}) = 1$.

We can define the prolongation

$$P_{\mathrm{SA}} := \big(\psi_1, \psi_2, \ldots, \psi_{N_c}\big).$$

Define $\mathbf{1}_c := (1, \ldots, 1)^T \in \mathbb{R}^{N_c}$. Hence we have $P_{\mathrm{SA}} \mathbf{1}_c = \mathbf{1}$. Furthermore, the coarse level matrix $A_c = P_{\mathrm{SA}}^T A P_{\mathrm{SA}}$ satisfies that

$$A_c \mathbf{1}_c = (P_{\mathrm{SA}}^T A P_{\mathrm{SA}}) \mathbf{1}_c = P_{\mathrm{SA}}^T A \mathbf{1} = \mathbf{0}.$$

By applying this definition recursively, we can finish the AMG setup for the smoothed aggregation method.

Listing 7.4: Smoothed aggregation method

```
1 U ← V;
2 for i ∈ U
3     S_i ← {j ∈ U : j is strongly coupled to i};
4     construct a column of prolongation P based on S_i;
5     U ← U\({i} ⋃ S_i);
6 end
7 Smooth the basis functions using the weighted Jacobi method P_SA = (I − ωD^{-1}A)P;
```

We have mentioned in the previous subsection that there are different ways to form aggregates. After forming aggregates one can apply UA or SA to give prolongation. Now we do preliminary tests on aggregation methods for solving the 2D Poisson's equation using the five-point stencil; see Table 7.2. The AMG methods are applied as preconditioners of PCG. Note that, for the SA method, we use the standard V-cycle multigrid in the solve phase; on the other hand, for the UA methods, we use the K-cycle multigrid for better convergence behavior.

Aggregation method	SA [111]	UA [111]	Pairwise UA [95]
Number of levels	5	5	7
Operator complexity	1.364	1.264	1.332
Setup time (sec)	0.557	0.171	0.277
Number of iterations	16	21	12
Solve time (sec)	1.223	1.696	1.336

Table 7.2: Solving 2D five-point stencil of the Poisson's equation with 1 million DOF using aggregation-based AMG methods (stopping criteria for PCG is the relative residual smaller than 10^{-6}).

Part III

Applications of Multilevel Iterative Methods

Chapter 8

Fluid Problems

Computational fluid dynamics (CFD) is a branch of fluid mechanics that uses numerical analysis and algorithms to solve and analyze fluid problems. Computers are used to perform the calculations required to simulate liquids or/and gases with surfaces defined by boundary conditions. The fundamental basis of most CFD problems are the Navier–Stokes (NS) equations, which define single-phase fluid flows. These equations can be simplified by removing terms describing viscous actions to yield the Euler equations. These equations can be simplified by dropping the nonlinear convection term to yield the Stokes equation. In this chapter, we discuss multilevel iterative methods suitable for problems arising from CFD.

8.1 The Navier–Stokes equations ⋆

The Navier–Stokes equations describe the motion of viscous fluid substances. These balance equations arise from applying the Newton's second law to fluid motion, together with the assumption that the stress in the fluid is the sum of a diffusing viscous term (proportional to the gradient of velocity) and a pressure term.

Flow map

Let Ω_0 be an open bounded set in $\mathbb{R}^d$ $(d = 2, 3)$. As a convention, we denote the location of a particle in Ω_0 by $X = (X_1, \ldots, X_d)$. This is the configuration at time $t = 0$, which is also called the initial configuration. To describe movement of particles, we denote the current configuration as Ω_t at any time $t \geqslant 0$. The position of a particle at time t is denoted by $x = (x_1, \ldots, x_d)$; see Figure 8.1. The Lagrangian specification of the flow field is a way of looking at particle motion where the observer follows an individual particle as it moves through space and time; see the right figure in Figure 8.1. The Eulerian specification of the flow field is a way of looking at

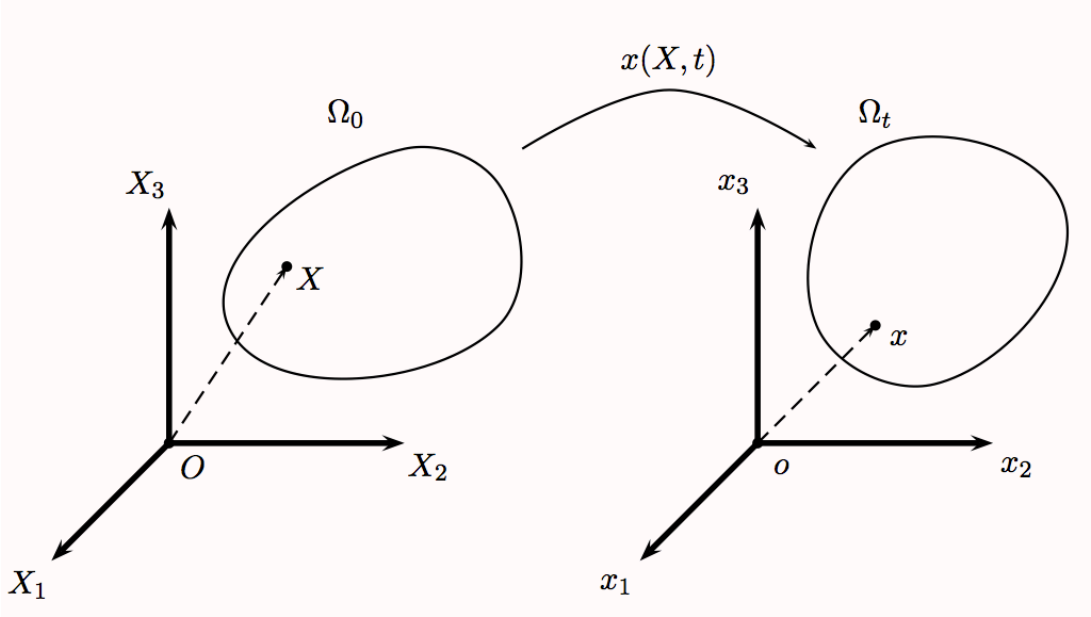


Figure 8.1: From initial configuration Ω_0 to current configuration Ω_t.

particle motion that focuses on specific locations in the space through which the fluid flows as time passes; see the left figure in Figure 8.1.

For a vector-valued function $\mathbf{f} : \Omega_t \mapsto \mathbb{R}^d$, the divergence operator can then be written as $\nabla \cdot \mathbf{f} := \sum_{i=1}^d \partial_i \mathbf{f}_i$. The gradient tensor $\nabla \mathbf{f}$ with $(\nabla \mathbf{f})_{i,j} = \partial_j \mathbf{f}_i$. Let $\mathbf{a} \in \mathbb{R}^d$ be a constant vector field and $(\mathbf{a} \cdot \nabla)\mathbf{f} = (\sum_{i=1}^d \mathbf{a}_i \partial_i)\mathbf{f}$. We define an inner product of two gradient matrices $\nabla \mathbf{f} : \nabla \mathbf{g} = \sum_{i=1}^d \nabla \mathbf{f}_i \cdot \nabla \mathbf{g}_i$. Let $\mathbf{u}(\cdot, t) : \Omega_t \mapsto \mathbb{R}^d$ be the velocity field at a fixed time t. The gradient of $\mathbf{u}$ is denoted by $\nabla \mathbf{u} = (\partial_j \mathbf{u}_i)_{i,j}$. Furthermore, $\nabla \mathbf{u}$ is often divided into the symmetric part and the anti-symmetric part. The symmetric gradient is denoted as $\boldsymbol{\varepsilon}(\mathbf{u}) := \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T)$ and it is the so-called *strain rate*.

We are ready to introduce an important concept to describe trajectory of particles, namely, the flow map $x(X, t)$, which is the trajectory of a particle X along time. We define that

$$\dot{x} = \frac{dx(X,t)}{dt} = \mathbf{u}(x,t) \quad \text{and} \quad x(X,0) = X. \tag{8.1}$$

This simple one-dimensional ordinary differential equation (ODE) is called the characteristic equation. Hence $x(\cdot, t)$ is a mapping from the initial configuration Ω_0 to the current configuration Ω_t, or deformation. The deformation gradient and its determinant are then defined as

$$F := \frac{\partial x}{\partial X} \quad \text{and} \quad J := |F| = \det(F), \tag{8.2}$$

respectively. F is also called the Jacobian matrix.

For any function $f(\cdot, t) : \Omega_t \mapsto \mathbb{R}$, we can easily derive that

$$\dot{f} := \frac{df(x,t)}{dt} = \nabla f \cdot \frac{dx}{dt} + f_t = f_t + \mathbf{u} \cdot \nabla f, \tag{8.3}$$

which is usually called the material derivative of f. Apparently, F and J are functions of t. Using the well-known Jacobi's formula in matrix calculus, we can show that

$$\dot{J} = J \operatorname{tr}(F^{-1}\dot{F}). \tag{8.4}$$

Hence we can immediately obtain

$$\dot{J} = J \operatorname{tr}(\frac{\partial X}{\partial x}\frac{\partial \dot{x}}{\partial X}) = J \operatorname{tr}(\nabla \mathbf{u}) = J(\nabla \cdot \mathbf{u}). \tag{8.5}$$

This way, we get an ODE for J, i.e.

$$\dot{J} = (\nabla \cdot \mathbf{u})J \quad \text{and} \quad J(0) = 1. \tag{8.6}$$

In fact, we can also obtain the variation of the determinant of F,

$$\delta|F| = |F| \operatorname{tr}(F^{-1}\delta F).$$

We can also derive similar results for the deformation gradient F itself:

$$\dot{F} = \frac{d}{dt}(\frac{\partial x}{\partial X}) = \frac{\partial \dot{x}}{\partial X} = \frac{\partial \mathbf{u}}{\partial X} = \nabla \mathbf{u} F. \tag{8.7}$$

We can easily immediately see that

$$F_t + \mathbf{u} \cdot \nabla F = \nabla \mathbf{u}\, F \quad \text{and} \quad F(0) = I. \tag{8.8}$$

Volume and mass conservation

A very useful trick for doing calculus in continuum mechanics is the pull-back (from Ω_t to Ω_0) and push-forward (from Ω_0 to Ω_t) argument. We first give an example:

$$\begin{aligned}
\frac{d}{dt}\int_{\Omega_t} f(x,t)\,dx &= \frac{d}{dt}\int_{\Omega_0} f(x(X,t),t)J\,dX \\
&= \int_{\Omega_0} \frac{d}{dt} f(x(X,t),t)J\,dX + \int_{\Omega_0} f(x(X,t),t)\dot{J}\,dX \\
&= \int_{\Omega_0} (f_t + \mathbf{u}\cdot\nabla f + f\nabla\cdot\mathbf{u})J\,dX \\
&= \int_{\Omega_t} \dot{f} + f\nabla\cdot\mathbf{u}\,dx \;= \int_{\Omega_t} f_t + \nabla\cdot(f\mathbf{u})\,dx.
\end{aligned} \tag{8.9}$$

This identity is often called the *transport formula*.

Lemma 8.1 (Transport formula)**.** For a function $f : \Omega_t \mapsto \mathbb{R}$ and $\mathbf{u}(x,t) := \frac{dx(X,t)}{dt}$, we have

$$\frac{d}{dt}\int_{\Omega_t} f(x,t)\,dx = \int_{\Omega_t} f_t + \nabla\cdot(f\mathbf{u})\,dx = \int_{\Omega_0} (f_t + \mathbf{u}\cdot\nabla f + f\nabla\cdot\mathbf{u})J\,dX.$$

For a domain $\Omega \subset \mathbb{R}^d$, we denote its volume (or area) as $|\Omega|$. We then find that

$$|\Omega_t| = \int_{\Omega_t} 1 \, dx = \int_{\Omega_0} J \, dX = J|\Omega_0|$$

For *incompressible* fluids, we have that the volume preserving property

$$|\Omega_t| \equiv |\Omega_0| \quad \text{or} \quad J(t) \equiv 1.$$

From the equation (8.6), we can derive that $\nabla \cdot \mathbf{u} = 0$. This is the so-called *divergence-free* condition.

Denote the density of the material occupying Ω_t by $\rho(x,t)$. According to the equation (8.9), for any region $\omega_t \subset \Omega_t$, we have that

$$\frac{d}{dt} \int_{\omega_t} \rho(x,t) \, dx = \int_{\omega_t} \rho_t + \nabla \cdot (\rho \mathbf{u}) \, dx$$

Since this identity holds for any ω, we immediately see that

$$\rho_t + \nabla \cdot (\rho \mathbf{u}) = 0 \quad \text{and} \quad \dot{\rho} + \rho \nabla \cdot \mathbf{u} = 0, \tag{8.10}$$

which is called the equation of mass conservation or the continuity equation.

It is clear that integrating the density over any domain ω_t gives the mass. Due to mass conservation, we have that

$$\int_{\omega_0} \rho_0(X) \, dX = \int_{\omega_t} \rho(x,t) \, dx = \int_{\omega_0} \rho(x(X,t),t) J \, dX.$$

Hence, we have the relation

$$\rho(x(X,t),t) = \frac{\rho_0(X)}{J}. \tag{8.11}$$

If the incompressible condition $\nabla \cdot \mathbf{u} = 0$ holds, we obtain that $\rho(x(X,t),t) = \rho_0(X)$.

If $\rho \equiv \rho_0$ is a constant, then (8.10) gives the divergence-free condition immediately. On the other hand, if we assume incompressibility, we can get a simplified equation:

$$\rho_t + (\mathbf{u} \cdot \nabla)\rho = 0 \quad \text{or} \quad \dot{\rho} = 0. \tag{8.12}$$

Together with $\rho(X,0) = \rho_0$ being a constant, we can get $\rho \equiv \rho_0$ for all time $t \in [0,T]$.

Balance of momentum

Now we consider the incompressible Newtonian fluids. Due to the Newton's Second Law, we have the balance of momentum

$$\frac{d}{dt} \int_{\Omega_t} \rho \mathbf{u} \, dx = \text{Force}(\Omega_t). \tag{8.13}$$

The left-hand side of the above equation is the rate of change for the momentum. Using the transport formula (Lemma 8.1), we derive that

$$\frac{d}{dt}\int_{\Omega_t}\rho\mathbf{u}\,dx = \int_{\Omega_0}(\rho_t + \mathbf{u}\cdot\nabla\rho)\mathbf{u} + \rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u})\,dX.$$

Due to the mass conservation and incompressibility (8.12), we then have

$$\frac{d}{dt}\int_{\Omega_t}\rho\mathbf{u}\,dx = \int_{\Omega_t}\rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u})\,dx. \tag{8.14}$$

On the other hand, the right-hand side of the Newton's Second Law is the total force acting on Ω_t. We have, from the divergence theorem, that

$$\text{Force}(\Omega_t) := \int_{\Omega_t}\mathbf{f}\,dx + \int_{\partial\Omega_t}\mathbf{T}\cdot\mathbf{n}\,dS = \int_{\Omega_t}\mathbf{f} + \nabla\cdot\mathbf{T}\,dx,$$

where $\mathbf{f}$ is the total external body force (like gravity), $\mathbf{T}$ is the traction tensor on the boundary of Ω_t, and $\mathbf{n}$ is the outer normal direction on the boundary $\partial\Omega_t$. The exact form of $\mathbf{T}$ depends on the underlying constitutive laws. For Newtonian fluids, the traction can be defined as

$$\mathbf{T} := -pI + 2\mu\boldsymbol{\varepsilon}(\mathbf{u}), \tag{8.15}$$

where p is the pressure and μ is the viscosity.

For incompressible fluids, we have $\nabla\cdot\mathbf{u} = 0$. In turn, we can obtain (see HW 8.1) that

$$\big(\nabla\cdot(2\boldsymbol{\varepsilon}(\mathbf{u}))\big)_j = \sum_{i=1}^{d}\partial_i(\mathbf{u}_{i,j} + \mathbf{u}_{j,i}) = \sum_{i=1}^{d}\partial_j\mathbf{u}_{i,i} + \sum_{i=1}^{d}\partial_i\mathbf{u}_{j,i} = \Delta\mathbf{u}_j,$$

which means

$$2\,\nabla\cdot\boldsymbol{\varepsilon}(\mathbf{u}) = \Delta\mathbf{u}. \tag{8.16}$$

This way we can get the momentum equation (balance of force) for incompressible Newtonian fluids:

$$\rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) = -\nabla p + \mu\Delta\mathbf{u}. \tag{8.17}$$

If the density ρ is a constant, we further simplify the above equation (by modifying the definition of p and μ) to give

$$\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \mu\Delta\mathbf{u}. \tag{8.18}$$

Mathematical models

To summarize, we have derived the mathematical model for incompressible Newtonian fluids, i.e., the *Navier–Stokes* (NS) equations:

$$\begin{cases} \rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) - \mu\Delta\mathbf{u} + \nabla p = \mathbf{f}, & \Omega_t & \text{balance of momentum;} \\ \rho_t + \nabla\cdot(\rho\mathbf{u}) = 0, & \Omega_t & \text{conservation of mass;} \\ \nabla\cdot\mathbf{u} = 0, & \Omega_t & \text{incompressibility;} \\ \mathbf{u} = 0, & \partial\Omega_t & \text{no-slip boundary;} \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega_t & \text{initial condition.} \end{cases} \tag{8.19}$$

If we assume the density ρ is a constant, then we can write (8.19) as follows:

$$\begin{cases} \mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} - \mu\Delta\mathbf{u} + \nabla p = \mathbf{f}, & \Omega_t & \text{momentum equation;} \\ \nabla\cdot\mathbf{u} = 0, & \Omega_t & \text{continuity equation;} \\ \mathbf{u} = 0, & \partial\Omega_t & \text{no-slip boundary;} \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega_t & \text{initial condition.} \end{cases} \tag{8.20}$$

Now we have the mathematical model for incompressible viscous Newtonian fluids. If we consider ideal fluids (viscosity $\mu = 0$) and assume that there is no external body force ($\mathbf{f} = 0$), then we get the incompressible *Euler* equations:

$$\begin{cases} \rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) + \nabla p = 0, & \Omega_t & \text{balance of momentum;} \\ \rho_t + \nabla\cdot(\rho\mathbf{u}) = 0, & \Omega_t & \text{conservation of mass;} \\ \nabla\cdot\mathbf{u} = 0, & \Omega_t & \text{incompressibility;} \\ \mathbf{u}\cdot\mathbf{n} = 0, & \partial\Omega_t & \text{no-flow boundary;} \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega_t & \text{initial condition.} \end{cases} \tag{8.21}$$

If the density ρ is a constant, then we have the following simplified form:

$$\begin{cases} \mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = 0, & \Omega_t & \text{momentum equation;} \\ \nabla\cdot\mathbf{u} = 0, & \Omega_t & \text{continuity equation;} \\ \mathbf{u}\cdot\mathbf{n} = 0, & \partial\Omega_t & \text{no-flow boundary;} \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega_t & \text{initial condition.} \end{cases} \tag{8.22}$$

For numerical simulation of the Navier–Stokes and Euler equations, there are several technical difficulties. First of all, the incompressibility condition is a constraint on the velocity field and appropriate finite element spaces need to be selected to discretize this mixed problem. Secondly, these equations have a nonlinear convection term; when the viscosity coefficient μ is small (corresponding to high Reynolds number), the convection is essentially dominant.

8.2 The Stokes-type equations

For simplicity, we now focus on a linearized problem of the Navier–Stokes equation, namely the Stokes equation.

The time-dependent Stokes equation

On an open bounded set $\Omega \subset \mathbb{R}^d$, we consider

$$\begin{cases} \mathbf{u}_t - \mu\Delta\mathbf{u} + \nabla p &= \mathbf{f}, & \Omega; \\ \nabla\cdot\mathbf{u} &= 0, & \Omega; \\ \mathbf{u} &= 0, & \partial\Omega; \\ \mathbf{u}|_{t=0} &= \mathbf{u}_0, & \Omega. \end{cases} \tag{8.23}$$

This set of equations is usually referred to as the time-dependent Stokes equations. After time discretization, we need to solve the Stokes-like equations

$$\begin{cases} (\mathcal{I} - \epsilon^2\Delta)\mathbf{u} + \nabla p &= \mathbf{f}, & \Omega; \\ \nabla\cdot\mathbf{u} &= 0, & \Omega; \\ \mathbf{u} &= 0, & \partial\Omega. \end{cases} \tag{8.24}$$

We can further simplify the discussion and only consider the following steady-state Stokes equations, i.e.,

$$\begin{cases} -\Delta\mathbf{u} + \nabla p &= \mathbf{f}, & \Omega; \\ \nabla\cdot\mathbf{u} &= 0, & \Omega; \\ \mathbf{u} &= 0, & \partial\Omega. \end{cases} \tag{8.25}$$

Let $\mathscr{V} := \left[H_0^1(\Omega)\right]^d$ and $\mathscr{Q} := L_0^2(\Omega) = \left\{q \in L^2(\Omega) : \int_\Omega q = 0\right\}$. The weak form of the Stokes equation (8.25) can be written as: Find $\mathbf{u} \in \mathscr{V}$ and $p \in \mathscr{Q}$, such that

$$\begin{cases} 2\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\, dx + (p, \nabla\cdot\mathbf{v}) &= (\mathbf{f}, \mathbf{v}), & \forall\, \mathbf{v} \in \mathscr{V}; \\ (\nabla\cdot\mathbf{u}, q) &= 0, & \forall\, q \in \mathscr{Q}. \end{cases} \tag{8.26}$$

The derivation is straightforward and hence leave to the readers; see HW 8.2.

Remark 8.2 (Constrained energy minimization)**.** We can view the Stokes equations as a constrained energy minimization problem

$$\min_{\mathbf{v}\in\mathscr{Z}} \int_\Omega \boldsymbol{\varepsilon}(\mathbf{v}) : \boldsymbol{\varepsilon}(\mathbf{v})\, dx - \int_\Omega \mathbf{f}\cdot\mathbf{v}\, dx,$$

where $\mathscr{Z} := \{\mathbf{v} \in \mathscr{V} : \nabla\cdot\mathbf{v} = 0\}$ is the subspace of divergence-free functions. The equation (8.26) is the first-order optimality condition of this constrained minimization problem and p is the Lagrange multiplier. □

The Brezzi theory

Let $\mathscr{V}'$ and $\mathscr{Q}'$ be the dual spaces of $\mathscr{V}$ and $\mathscr{Q}$, respectively. Generally speaking, we can put the Stokes problem in an abstract framework and consider the following *saddle-point* problem: For any given $(f, g) \in \mathscr{V}' \times \mathscr{Q}'$, find a pair $(u, p) \in \mathscr{V} \times \mathscr{Q}$, such that the following system holds

$$\begin{cases} a[u, v] + b[v, p] &= \langle f, v \rangle, \quad \forall v \in \mathscr{V}; \\ b[u, q] &= \langle g, q \rangle, \quad \forall q \in \mathscr{Q}. \end{cases} \tag{8.27}$$

Here $a[\cdot, \cdot] : \mathscr{V} \times \mathscr{V} \mapsto \mathbb{R}$ and $b[\cdot, \cdot] : \mathscr{V} \times \mathscr{Q} \mapsto \mathbb{R}$ are *continuous* bilinear forms, i.e.,

$$\begin{aligned} a[u, v] &\leqslant C_a \|u\|_{\mathscr{V}} \|v\|_{\mathscr{V}}, \quad \forall u, v \in \mathscr{V}, \\ b[u, p] &\leqslant C_b \|u\|_{\mathscr{V}} \|p\|_{\mathscr{Q}}, \quad \forall u \in \mathscr{V}, p \in \mathscr{Q}. \end{aligned}$$

We can identify a linear operator $\mathcal{A} : \mathscr{V} \mapsto \mathscr{V}'$ such that

$$\langle \mathcal{A}u, v \rangle = a[u, v], \quad \forall u \in \mathscr{V},\ v \in \mathscr{V}$$

and another linear operator $\mathcal{B} : \mathscr{V} \mapsto \mathscr{Q}'$ (or its adjoint $\mathcal{B}^T : \mathscr{Q} \mapsto \mathscr{V}'$) such that

$$\langle \mathcal{B}u, p \rangle = \langle u, \mathcal{B}^T p \rangle = b[u, p], \quad \forall u \in \mathscr{V},\ p \in \mathscr{Q}.$$

Hence (8.27) can be written in the following operator form

$$\begin{cases} \mathcal{A}u + \mathcal{B}^T p &= f, \\ \mathcal{B}u &= g. \end{cases}$$

We now analyze under what condition(s) the weak formulation (8.27) is well-posed. We define the kernel space of $\mathcal{B}$ as

$$\mathscr{Z} := \mathrm{null}(\mathcal{B}) = \left\{ v \in \mathscr{V} \,:\, b[v, q] = 0,\ \forall q \in \mathscr{Q} \right\} \subset \mathscr{V}.$$

Because $b[\cdot, \cdot]$ is continuous, $\mathscr{Z}$ is closed. Hence we can give an orthogonal decomposition

$$\mathscr{V} = \mathscr{Z} \oplus \mathscr{Z}^{\perp} = \mathrm{null}(\mathcal{B}) \oplus \mathrm{null}(\mathcal{B})^{\perp}.$$

For any $u \in \mathscr{V}$, we have $u = u_0 + u_{\perp}$, with $u_0 \in \mathrm{null}(\mathcal{B})$ and $u_{\perp} \in \mathrm{null}(\mathcal{B})^{\perp}$.

In order to solve $\mathcal{B}u = g$, we only need to solve $\mathcal{B}u_{\perp} = g$. Using the inf-sup theory discussed in §1.1, we can see that, if $\mathcal{B}$ is surjective, namely,

$$\inf_{q \in \mathscr{Q}} \sup_{v \in \mathscr{V}} \frac{b[v, q]}{\|v\|_{\mathscr{V}} \|q\|_{\mathscr{Q}}} = \beta > 0, \tag{8.28}$$

then $u_\perp$ exists. Furthermore, it is easy to see that $u_\perp$ is also unique[1]. Hence we have $\mathcal{B}: \mathscr{Z}^\perp \mapsto \mathscr{Q}'$ and $\mathcal{B}^T: \mathscr{Q} \mapsto (\mathscr{Z}^\perp)'$ are isomorphisms.

Now we only need to show the existence and uniqueness of the following problem: Find $u_0 \in \mathscr{Z}$, such that

$$a[u_0, v] = \langle f, v\rangle - a[u_\perp, v], \quad \forall\, v \in \mathscr{Z}.$$

According to the Nečas Theorem 1.16, we know that the existence and uniqueness of u_0 is equivalent to the following inf-sup conditions

$$\inf_{u\in\mathscr{Z}} \sup_{v\in\mathscr{Z}} \frac{a[u,v]}{\|u\|_{\mathscr{V}}\|v\|_{\mathscr{V}}} = \inf_{v\in\mathscr{Z}} \sup_{u\in\mathscr{Z}} \frac{a[u,v]}{\|u\|_{\mathscr{V}}\|v\|_{\mathscr{V}}} = \alpha > 0. \tag{8.29}$$

With the conditions (8.29) and (8.28), we obtain a unique solution $u = u_0 + u_\perp$.

We can find the solution for the pressure variable by solving

$$\mathcal{B}^T p = f - \mathcal{A}u. \tag{8.30}$$

For any $v \in \mathscr{Z} = \mathrm{null}(\mathcal{B})$, it is easy to see that

$$\langle f - \mathcal{A}u, v\rangle = \langle \mathcal{B}^T p, v\rangle = \langle p, \mathcal{B}v\rangle = 0.$$

Hence, $f - \mathcal{A}u \in (\mathscr{Z}^\perp)' = \{w \in \mathscr{V}' : \langle w, v\rangle = 0,\ \forall v \in \mathscr{Z}\}$. Because $\mathcal{B}^T : \mathscr{Q} \mapsto (\mathscr{Z}^\perp)'$ is an isomorphism, there is a unique solution to (8.30).

Hence we obtain the following well-posedness result [42, Theorem 1.1]:

Theorem 8.3 (Brezzi Theorem)**.** For continuous bilinear forms $a[\cdot,\cdot]$ and $b[\cdot,\cdot]$, the saddle-point problem (8.27) is well-posed if and only if (8.29) and (8.28) hold. Furthermore, the solution (u, p) satisfies the stability condition

$$\|u\|_{\mathscr{V}} + \|p\|_{\mathscr{Q}} \lesssim \|f\|_{\mathscr{V}'} + \|g\|_{\mathscr{Q}'}.$$

Remark 8.4 (Inf-sup condition of the mixed formulation)**.** Let $\mathscr{X} := \mathscr{V} \times \mathscr{Q}$. We define a new bilinear form $\tilde{a}: \mathscr{X} \times \mathscr{X} \mapsto \mathbb{R}$

$$\tilde{a}[(u,p),(v,q)] := a[u,v] + b[v,p] + b[u,q].$$

Then the saddle-point problem (8.27) is equivalent to finding $(u, p) \in \mathscr{X}$ such that

$$\tilde{a}[(u,p),(v,q)] = \langle f, v\rangle + \langle g, q\rangle, \quad \forall (v,q) \in \mathscr{X}. \tag{8.31}$$

If both $a[\cdot,\cdot]$ and $b[\cdot,\cdot]$ are continuous, then $\tilde{a}[\cdot,\cdot]$ is also continuous. If $a[\cdot,\cdot]$ and $b[\cdot,\cdot]$ satisfy the standard Brezzi conditions (8.29) and (8.28), respectively, then $\tilde{a}[\cdot,\cdot]$ satisfies the inf-sup condition as well. □

[1] Suppose there is another solution $\tilde{u}_\perp$, then $\mathcal{B}(u_\perp - \tilde{u}_\perp) = 0$. In turn, we have $u_\perp - \tilde{u}_\perp$ is in $\mathrm{null}(\mathcal{B})$. Due to $u_\perp - \tilde{u}_\perp \in \mathrm{null}(\mathcal{B})^\perp$, we find $u_\perp - \tilde{u}_\perp = 0$.

Well-posedness of the Stokes equation

In view of the general theory developed in the previous subsection, we can define

$$a[\mathbf{u},\mathbf{v}] := 2\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\, dx \qquad \mathcal{A} := -\Delta \tag{8.32}$$

$$b[\mathbf{v}, q] := -\int_\Omega \nabla\cdot\mathbf{v}\, q\, dx \qquad \mathcal{B} := -\nabla\cdot,\ \mathcal{B}^T := \nabla \tag{8.33}$$

In this case, the inf-sup condition (8.29) is trivial since the coercive condition holds, i.e.,

$$\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{u}) \geqslant \alpha\|\mathbf{u}\|_1^2, \quad \forall \mathbf{u} \in [H_0^1(\Omega)]^d.$$

Hence we only need to check the inf-sup condition for $b[\cdot,\cdot]$.

Lemma 8.5 (Inf-sup condition for divergence operator)**.** For any $q \in \mathscr{Q} = L_0^2(\Omega)$, there exists $\mathbf{v} \in \mathscr{V} = [H_0^1(\Omega)]^d$ such that

$$\nabla\cdot\mathbf{v} = q \quad \text{and} \quad \|\mathbf{v}\|_1 \lesssim \|q\|_0.$$

So the inf-sup condition (8.28) holds.

Proof. This non-trivial result goes back to Nečas and a proof can be found in [60, III.3.1]. □

Remark 8.6 (Existence of solution)**.** It has been shown in the above lemma that $\text{range}(\mathcal{B}) = L^2(\Omega)/\mathbb{R} \cong \mathscr{Q}$. Or equivalently, we have $\text{null}(\mathcal{B}^T) \bigcap \mathscr{Q} = \{0\}$. □

Using the previous lemma and the Brezzi theorem, we can easily get the following result:

Theorem 8.7 (Well-posedness of the Stokes equation)**.** There exists a unique solution $(\mathbf{u}, p) \in [H_0^1(\Omega)]^d \times L_0^2(\Omega)$ to the weak form of the Stokes equation (8.26) and

$$\|\mathbf{u}\|_1 + \|p\|_0 \lesssim \|\mathbf{f}\|_{-1}.$$

Penalty method for the Stokes equation ⋆

In general, there are two approaches to approximate the Stokes problem. The first one is to approximate (8.26) directly. An alternative method is to formulate the original problem using a penalty method as

$$\text{Find}\ \ \mathbf{u} \in \mathscr{V}: \quad 2\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\, dx + \gamma(\nabla\cdot\mathbf{u},\, \nabla\cdot\mathbf{v}) = (\mathbf{f}, \mathbf{v}), \quad \forall\, \mathbf{v} \in \mathscr{V}. \tag{8.34}$$

The above equation can also be seen in the linear elasticity problems and it is known for causing the *locking* phenomena[2] for many finite element methods when γ is big. This is usually caused by overly constraint on the velocity space. To cure such a problem, penalty methods introduce selective or reduced integration procedures. It has been shown that penalty methods are sometimes equivalent to mixed methods [77].

[2] The computed velocity is vanishing or unnaturally small for big λ.

8.3 Mixed finite element methods

In this section, we consider conforming mixed finite element methods for the Stokes equations. Let $V_h \subset \mathscr{V} = [H_0^1(\Omega)]^d$ and $Q_h \subset \mathscr{Q} = L_0^2(\Omega)$ be finite dimensional spaces. Find $\mathbf{u}_h \in V_h$ and $p_h \in Q_h$, such that

$$\left\{ \begin{array}{lcll} 2\displaystyle\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}_h) : \boldsymbol{\varepsilon}(\mathbf{v}_h)\, dx - (p_h,\, \nabla\cdot\mathbf{v}_h) & = & (\mathbf{f}, \mathbf{v}_h), & \forall\, \mathbf{v}_h \in V_h, \\ (\nabla\cdot\mathbf{u}_h,\, q_h) & = & 0, & \forall\, q_h \in Q_h. \end{array} \right. \tag{8.35}$$

The existence of the discrete solution $(\mathbf{u}_h, p_h)$ is straightforward due to the conformity of the approximation spaces.

Well-posedness and convergence

Let $Z_h = \mathrm{null}(\mathcal{B}_h)$ be the kernel of the discrete divergence operator. In fact, the coercivity of $a[\cdot,\cdot]$ yields that

$$\inf_{\mathbf{u}_h\in Z_h} \sup_{\mathbf{v}_h\in Z_h} \frac{a[\mathbf{u}_h, \mathbf{v}_h]}{\|\mathbf{u}_h\|_1 \|\mathbf{v}_h\|_1} = \alpha_h > 0. \tag{8.36}$$

If $Z_h \subset \mathscr{Z}$ and the coercivity condition holds, we have the following optimal approximation property by the Céa's lemma (Lemma 3.2):

$$\|\mathbf{u} - \mathbf{u}_h\|_{\mathscr{V}} \leqslant \frac{C_a}{\alpha_h} \inf_{\mathbf{v}_h\in Z_h} \|\mathbf{u} - \mathbf{v}_h\|_{\mathscr{V}}.$$

However, it is not easy to make the finite element kernel space $Z_h \subset \mathscr{Z}$. A sufficient condition for this inclusion property is $\mathcal{B}(V_h) \subset Q_h$, which suggests Q_h should be large enough for a fixed space V_h. In fact, we have

$$\mathcal{B}_h\mathbf{u}_h = 0, \quad \text{in } Q_h' \quad \Longleftrightarrow \quad (\mathcal{B}\mathbf{u}_h, q_h) = 0, \quad \forall q_h \in Q_h.$$

Furthermore, we also have

$$\mathcal{B}\mathbf{u}_h = 0, \quad \text{in } \mathscr{Q}' \quad \Longleftrightarrow \quad (\mathcal{B}\mathbf{u}_h, q) = 0, \quad \forall q \in \mathscr{Q}.$$

If $\mathbf{u}_h \in Z_h$ and $q \in \mathscr{Q}$, then $(\mathcal{B}\mathbf{u}_h, q) = (\mathcal{B}\mathbf{u}_h, q_0 + q_\perp) = (\mathcal{B}\mathbf{u}_h, q_0) + (\mathcal{B}\mathbf{u}_h, q_\perp) = 0$, where $q = q_0 + q_\perp$ with $q_0 \in Q_h$. Notice that $(\mathcal{B}\mathbf{u}_h, q_\perp) = 0$ because the inclusion condition $\mathcal{B}(V_h) \subset Q_h$.

If $Z_h \nsubseteq \mathscr{Z}$, then there is a variational crime and we have following estimate:

$$\|\mathbf{u} - \mathbf{u}_h\|_{\mathscr{V}} \leqslant \Big(1 + \frac{C_a}{\alpha_h}\Big) \inf_{\mathbf{v}\in Z_h} \|\mathbf{u} - \mathbf{v}\|_{\mathscr{V}} + \frac{1}{\alpha_h} \sup_{\mathbf{w}\in Z_h\setminus\{0\}} \frac{\big|a[\mathbf{u} - \mathbf{u}_h, \mathbf{w}]\big|}{\|\mathbf{w}\|_{\mathscr{V}}}.$$

For $\mathbf{w} \in Z_h$, we have

$$a[\mathbf{u} - \mathbf{u}_h, \mathbf{w}] = a[\mathbf{u}, \mathbf{w}] - (\mathbf{f}, \mathbf{v}) = -b[\mathbf{w}, p] = -b[\mathbf{w}, p - q],$$

for any $q \in Q_h$. Because $b[\cdot,\cdot]$ is continues, we find that

$$\big|a[\mathbf{u}-\mathbf{u}_h,\mathbf{w}]\big| \leqslant C_b\|\mathbf{w}\|_{\mathscr{V}}\|p-q\|_{\mathscr{Q}}.$$

We can then conclude with the following best approximation result:

Lemma 8.8 (Quasi-optimality for velocity)**.** Let $V_h \subset \mathscr{V}$ and $Q_h \subset \mathscr{Q}$. If the bilinear form $a[\cdot,\cdot]$ is coercive, then we have

$$\|\mathbf{u}-\mathbf{u}_h\|_{\mathscr{V}} \leqslant \Big(1+\frac{C_a}{\alpha_h}\Big)\inf_{\mathbf{v}\in Z_h}\|\mathbf{u}-\mathbf{v}\|_{\mathscr{V}} + \frac{C_b}{\alpha_h}\inf_{q\in Q_h}\|p-q\|_{\mathscr{Q}}.$$

We have the identity

$$(\mathcal{B}_h\mathbf{u}_h, q_h) = b[\mathbf{u}_h,q_h] = (\mathcal{B}\mathbf{u}_h,q_h), \quad \forall q_h \in Q_h.$$

In the other words, $\mathcal{B}_h\mathbf{u}_h$ is the L^2-projection of $\mathcal{B}\mathbf{u}_h$ onto Q_h. If $\mathrm{null}(\mathcal{B}_h^T)$ is not trivial, then $\mathrm{range}(\mathcal{B}_h)$ is strictly included in Q_h. This could lead to ill-posed problems. For a fixed Q_h, the velocity approximation space V_h should be rich enough in order to guarantee the discrete inf-sup condition:

$$\inf_{q_h\in Q_h}\sup_{\mathbf{v}_h\in V_h}\frac{b(\mathbf{v}_h,q_h)}{\|\mathbf{v}_h\|_1\|q_h\|_0} = \beta_h > 0. \tag{8.37}$$

The condition $\mathrm{null}(\mathcal{B}_h^T) = \{0\}$ is necessary for the inf-sup condition above. If $\mathrm{null}(\mathcal{B}_h^T)$ is non-trivial, then the numerical solution p_h is not unique, namely, $p_h + s_h$ is also a solution when $s_h \in \mathrm{null}(\mathcal{B}_h^T)$. In this case, we usually find the computed pressure is oscillatory and, hence, $\mathrm{null}(\mathcal{B}_h^T)$ is often referred to as the space of spurious pressure modes.

Theorem 8.9 (Quasi-optimality)**.** Let $V_h \subset \mathscr{V}$ and $Q_h \subset \mathscr{Q}$. If the bilinear form $a[\cdot,\cdot]$ is coercive and the inf-sup condition (8.37) holds with $\beta_h \geqslant \beta_0 > 0$, then we have

$$\|\mathbf{u}-\mathbf{u}_h\|_{\mathscr{V}} + \|p-p_h\|_{\mathscr{Q}} \lesssim \inf_{\mathbf{v}\in Z_h}\|\mathbf{u}-\mathbf{v}\|_{\mathscr{V}} + \inf_{q\in Q_h}\|p-q\|_{\mathscr{Q}}.$$

Some stable finite element pairs ⋆

From the above discussions, we conclude that: To balance computational efforts and convergence rates for the velocity in $[H_0^1(\Omega)]^d$ and the pressure in $L_0^2(\Omega)$, it is better to use $(k+1)$-th degree of polynomials for V_h and k-th degree of polynomials for Q_h.

Remark 8.10 (Constraint ratio)**.** An empirical approach has been used to check the balance between velocity and pressure approximation spaces. The so-called constraint ratio is defined as

$$C_r := \mathrm{dim}Q_h/\mathrm{dim}V_h.$$

Apparently, if $C_r > 1$ then number of constraints exceeds the number of variables, which will usually cause locking. On the other hand, if C_r is too small, then divergence free condition is not approximated accurately enough. □

The easiest and seemingly natural choice for the mixed finite element spaces is the pair of the lowest order polynomials $P_h^{1,0}$–P_h^0. Unfortunately, this pair does not satisfy the discrete inf-sup condition and we have to either enlarge velocity field finite element space or restrict the pressure space. There are many possible stable pairs; see the survey paper [14] and references therein for more details. Here we just name a few:

- $\left[P_h^{k,0}\right]^d$–$P_h^{k-1,0}$ for $k \geqslant 2$, Taylor–Hood
- $\left[Q_h^{k,0}\right]^d$–$Q_h^{k-1,0}$ for $k \geqslant 2$, Taylor–Hood
- $\left[P_h^{1,0} \oplus \mathsf{B}_\tau^3\right]^2$–$P_h^0$, where B_τ^3 are cubic bubble functions, MINI
- $\left[P_{h/2}^{1,0}\right]^2$–$P_h^0$
- $\left[P_h^{2,0}\right]^d$–P_h^0, important theoretically, but degree not matching
- $\left[P_h^{2,0} \oplus \mathsf{B}_\tau^3\right]^2$–$P^{1,-1}$, Crouzeix–Raviart
- $\left[P_h^{2,0} \oplus \mathsf{B}_\tau^4\right]^3$–$P^{1,-1}$, Crouzeix–Raviart
- $\left[P_h^{1,\mathrm{NC}}\right]^d$–$P_h^0$, non-conforming Crouzeix–Raviart
- $\left[P_h^{k,0}\right]^2$–$P_h^{k-1,-1}$ for $k \geqslant 4$, Scott–Vogelius
- $\left[Q_h^{k,0}\right]^d$–$P_h^{k-1,-1}$ for $k \geqslant 2$

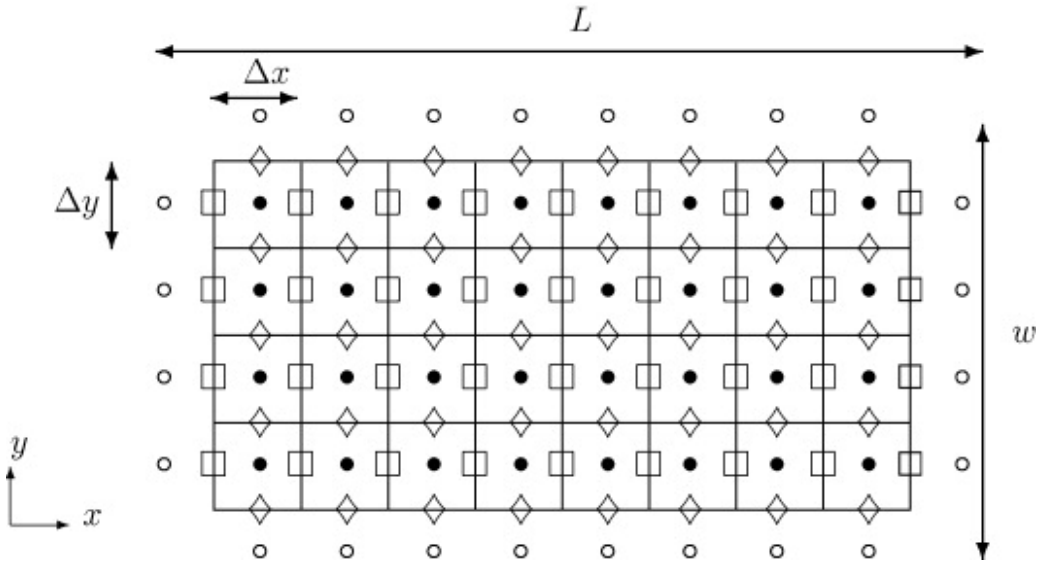


Figure 8.2: A sample discretization using the MAC scheme

Constructing stable finite difference schemes for the Stokes equation lacks of theoretical guidance like the Babuška–Brezzi condition discussed above. However we can expect that the standard five-point stencil does not work for the Stokes equation. This is because the five-point stencil can be viewed as $Q_h^{1,0} - Q_h^{1,0}$ finite element with a specific quadrature rule. If we change the pressure discretization to the center of cells, then it yields $Q_h^{1,0} - Q_h^{0,-1}$. And, apparently, both finite element pairs are not stable. The main idea of the Marker-and-Cell (MAC) scheme is to place the degrees of freedom for velocity and pressure at different locations. More specifically, the pressure p is defined at the cell centers, the velocity component $\mathbf{u}_1$ is defined at the middle points of vertical edges, and the velocity component $\mathbf{u}_2$ defined at the middle points of horizontal edges; see Figure 8.2. This method is same as the RT_0 finite element on rectangular grids.

Mixed methods for the Poisson's equation ⋆

Mixed finite element methods have been applied to our model problem, the Poisson's equation, as well. By introducing an artificial variable p, a general mixed formulation of the Poisson's equation can be written as

$$\begin{cases} \mathbf{u} - \nabla p &= \mathbf{f}, \quad \text{in } \Omega; \\ \nabla \cdot \mathbf{u} &= g, \quad \text{in } \Omega; \\ \mathbf{u} \cdot \mathbf{n} &= 0, \quad \text{on } \partial\Omega \end{cases} \tag{8.38}$$

In this section, we use this model problem to further explain how to construct preconditioners arising from the saddle-point problems.

Sometimes the mixed formulation of the Poisson's equation is used for numerical treatment: Find $(\mathbf{u}, p) \in H_0(\mathrm{div}, \Omega) \times L_0^2(\Omega)$ such that

$$\begin{cases} (\mathbf{u}, \mathbf{v}) + (p, \nabla \cdot \mathbf{v}) &= \langle \mathbf{f}, \mathbf{v} \rangle, \quad \forall\, \mathbf{v} \in H_0(\mathrm{div}, \Omega); \\ (\nabla \cdot \mathbf{u}, q) &= \langle g, q \rangle, \quad \forall\, q \in L_0^2(\Omega). \end{cases} \tag{8.39}$$

Here $H(\mathrm{div}, \Omega)$ consists of all functions in $[L^2(\Omega)]^d$ with divergence in $L^2(\Omega)$ and $H_0(\mathrm{div}, \Omega)$ contains the $H(\mathrm{div}, \Omega)$-functions with vanishing normal components on the boundary $\partial\Omega$. Define an inner product

$$(\mathbf{u}, \mathbf{v})_{H_0(\mathrm{div},\Omega)} := (\mathbf{u}, \mathbf{v}) + (\nabla \cdot \mathbf{u}, \nabla \cdot \mathbf{v}). \tag{8.40}$$

This problem corresponds to the mixed formulation of the Poisson's equation with the Neumann boundary condition.

If $\mathbf{u} \in \mathscr{Z}$ is divergence free, then $\|\mathbf{u}\|_{H_0(\mathrm{div},\Omega)} = \|\mathbf{u}\|_{0,\Omega}$. Hence we can easily verify the Brezzi conditions hold for this problem. As a consequence, the operator

$$\tilde{\mathcal{A}}_0 = \begin{pmatrix} \mathcal{I} & -\operatorname{grad} \\ \operatorname{div} & 0 \end{pmatrix} : H_0(\mathrm{div}, \Omega) \times L_0^2(\Omega) \mapsto H_0(\mathrm{div}, \Omega)' \times L_0^2(\Omega)$$

is an isomorphism. The canonical preconditioner is a block diagonal isomorphism

$$\tilde{\mathcal{D}}_0^{(1)} = \begin{pmatrix} (\mathcal{I} - \operatorname{grad}\operatorname{div})^{-1} & 0 \\ 0 & \mathcal{I} \end{pmatrix} : H_0(\operatorname{div},\Omega)' \times L_0^2(\Omega) \mapsto H_0(\operatorname{div},\Omega) \times L_0^2(\Omega).$$

There is an alternative mixed formulation for the Poisson's equation: Find $(\mathbf{u}, p) \in [L^2(\Omega)]^d \times \left(H^1(\Omega) \bigcap L_0^2(\Omega)\right)$ such that

$$\begin{cases} (\mathbf{u},\mathbf{v}) - (\nabla p, \mathbf{v}) &= \langle \mathbf{f}, \mathbf{v}\rangle, \quad \forall\, \mathbf{v} \in [L^2(\Omega)]^d; \\ -(\mathbf{u}, \nabla q) &= \langle g, q\rangle, \quad \forall\, q \in H^1(\Omega) \bigcap L_0^2(\Omega). \end{cases} \tag{8.41}$$

The Brezzi conditions can be verified using the Poincaré's inequality. Hence $\tilde{\mathcal{A}}$ is also well-defined on $[L^2(\Omega)]^d \times \left(H^1(\Omega) \bigcap L_0^2(\Omega)\right)$. And in this case, the canonical preconditioner is

$$\tilde{\mathcal{D}}_0^{(2)} = \begin{pmatrix} \mathcal{I} & 0 \\ 0 & (-\Delta)^{-1} \end{pmatrix} : [L^2(\Omega)]^d \times \left(H^1(\Omega) \bigcap L_0^2(\Omega)\right)' \mapsto [L^2(\Omega)]^d \times \left(H^1(\Omega) \bigcap L_0^2(\Omega)\right).$$

Apparently, this preconditioner is significantly different than the one given in the previous subsection. As a result, different choices of approximation space and its norm can yield very different solution methods.

8.4 Canonical preconditioners

In this section, we discuss how to construct canonical preconditioners for the saddle-point problems, like the Stokes equation and the time-dependent Stokes equation. The basic idea follows the discussion in §2.2.

Preconditioning the Stokes equation

We notice that the corresponding operator of the Stokes system

$$\tilde{\mathcal{A}} := \begin{pmatrix} -\Delta & -\operatorname{grad} \\ \operatorname{div} & 0 \end{pmatrix}$$

is an isomorphism mapping from $[H_0^1(\Omega)]^d \times L_0^2(\Omega)$ onto $[H^{-1}(\Omega)]^d \times L_0^2(\Omega)$. A natural preconditioner would be the classical block diagonal preconditioner

$$\tilde{\mathcal{D}} = \begin{pmatrix} (-\Delta)^{-1} & 0 \\ 0 & \mathcal{I} \end{pmatrix}.$$

This observation immediately motivates the classical block diagonal preconditioner [23].

Similar to the continuous case, we can construct natural preconditioners based on the mapping properties. Let $\{X_h\}$ be a family of finite element spaces and it is conforming in the sense that $X_h \subset \mathscr{X} := [H_0^1(\Omega)]^d \times L_0^2(\Omega)$. Consider the discrete Stokes problem: Find $(\mathbf{u}_h, p_h) \in X_h$ such that

$$\tilde{a}[(\mathbf{u}_h, p_h), (\mathbf{v}_h, q_h)] = \langle f, \mathbf{v}_h \rangle, \quad \forall (\mathbf{v}_h, q_h) \in X_h.$$

The corresponding linear map $\tilde{\mathcal{A}}_h : X_h \mapsto X_h'$ is given by

$$\langle \tilde{\mathcal{A}}_h x, y \rangle = \tilde{a}[x, y], \quad \forall x, y \in X_h.$$

Note that, in this case, $\tilde{a}$ is not positive definite and the system $\tilde{\mathcal{A}}_h$ can be singular.

According to Remark 8.4, the stable discretizations can be characterized by a discrete inf-sup condition: There exists a constant α_0, independent of h, such that

$$\inf_{x \in X_h} \sup_{y \in X_h} \frac{\tilde{a}[x, y]}{\|x\|_{\mathscr{X}} \|y\|_{\mathscr{X}}} \geqslant \alpha_0 > 0. \tag{8.42}$$

This condition does not follow from the corresponding continuous inf-sup condition. Similar to the continuous case, we can define a preconditioner $\tilde{\mathcal{D}}_h : X_h' \mapsto X_h$ by

$$(\tilde{\mathcal{D}}_h f, y)_{\mathscr{X}} = \langle f, y \rangle, \quad \forall y \in X_h.$$

That is to say

$$\tilde{\mathcal{D}}_h := \begin{pmatrix} (-\Delta_h)^{-1} & 0 \\ 0 & \mathcal{I}_h^{-1} \end{pmatrix}. \tag{8.43}$$

Apparently, if $\tilde{\mathcal{A}}_h$ is symmetric, $\tilde{\mathcal{D}}_h \tilde{\mathcal{A}}_h$ is symmetric with respect to $(\cdot, \cdot)_{\mathscr{X}}$ and

$$\|\tilde{\mathcal{D}}_h \tilde{\mathcal{A}}_h\|_{\mathscr{L}(X_h; X_h)} \leqslant C_a, \quad \|(\tilde{\mathcal{D}}_h \tilde{\mathcal{A}}_h)^{-1}\|_{\mathscr{L}(X_h; X_h)} \leqslant \alpha_0^{-1}.$$

Hence the condition number $\kappa(\tilde{\mathcal{D}}_h \tilde{\mathcal{A}}_h)$ is uniformly bounded.

Preconditioning the time-dependent Stokes equation ⋆

Now we are in position to develop preconditioners for the time-dependent Stokes problem (8.23). Like in many other applications, it is crucial to get robust or parameter-independent performance for problems with small or large parameters. One of the useful technique is to define proper parameter-dependent spaces and norms, such that the operator-norms of the coefficient operator can be bounded uniformly with respect to the parameters [83].

According to the classical theory of intersections and sums of Hilbert spaces [13], we can introduce the norms for $\mathscr{X}_1 \bigcap \mathscr{X}_2$ and $\mathscr{X}_1 + \mathscr{X}_2$ as

$$\|u\|_{\mathscr{X}_1 \bigcap \mathscr{X}_2} := \left(\|u\|_{\mathscr{X}_1}^2 + \|u\|_{\mathscr{X}_2}^2 \right)^{\frac{1}{2}}$$

and

$$\|u\|_{\mathscr{X}_1+\mathscr{X}_2} := \inf_{\substack{u=u_1+u_2\\ u_1\in\mathscr{X}_1, u_2\in\mathscr{X}_2}} \left(\|u_1\|^2_{\mathscr{X}_1} + \|u_2\|^2_{\mathscr{X}_2}\right)^{\frac{1}{2}}.$$

If $\mathscr{X}_1 \bigcap \mathscr{X}_2$ is dense in both $\mathscr{X}_1$ and $\mathscr{X}_2$, then

$$(\mathscr{X}_1 \bigcap \mathscr{X}_2)' = \mathscr{X}_1' + \mathscr{X}_2' \quad \text{and} \quad (\mathscr{X}_1 + \mathscr{X}_2)' = \mathscr{X}_1' \bigcap \mathscr{X}_2'.$$

If $\mathcal{F} \in \mathscr{L}(\mathscr{X}_1;\mathscr{Y}_1) \bigcap \mathscr{L}(\mathscr{X}_2;\mathscr{Y}_2)$, then

$$\mathcal{F} \in \mathscr{L}\big(\mathscr{X}_1 \bigcap \mathscr{X}_2; \mathscr{Y}_1 \bigcap \mathscr{Y}_2\big) \bigcap \mathscr{L}\big(\mathscr{X}_1 + \mathscr{X}_2; \mathscr{Y}_1 + \mathscr{Y}_2\big).$$

For our purpose, we assume that $\mathscr{X}_1$ and $\mathscr{X}_2$ are real separable Hilbert spaces and $\mathscr{X}_2 \subset \mathscr{X}_1$. Hence it is natural to assume $\|u\|_{\mathscr{X}_1} \leqslant \|u\|_{\mathscr{X}_2}$. For a real positive parameter $\epsilon > 0$, we consider the norm for the space $\mathscr{X}_1 \bigcap \epsilon\mathscr{X}_2$ and its dual, respectively, by

$$\|u\|_{\mathscr{X}_1\bigcap\epsilon\mathscr{X}_2} := \left(\|u\|^2_{\mathscr{X}_1} + \epsilon^2\|u\|^2_{\mathscr{X}_2}\right)^{\frac{1}{2}}, \quad \|f\|_{\mathscr{X}_1'+\epsilon^{-1}\mathscr{X}_2'} := \inf_{\substack{f=f_1+f_2\\ f_1\in\mathscr{X}_1', f_2\in\mathscr{X}_2'}} \left(\|f_1\|^2_{\mathscr{X}_1'} + \epsilon^{-2}\|f_2\|^2_{\mathscr{X}_2'}\right)^{\frac{1}{2}}.$$

Apparently, $\mathscr{X}_\epsilon := \mathscr{X}_1 \bigcap \epsilon\mathscr{X}_2$ is the same as $\mathscr{X}_2$ as a set. Furthermore, as ϵ tends to zero, the norm of $\mathscr{X}_\epsilon$ approaches the norm of $\|\cdot\|_{\mathscr{X}_1}$. Similarly, $\mathscr{X}_\epsilon' := \mathscr{X}_1' + \epsilon^{-1}\mathscr{X}_2'$, as a set, is the same as $\mathscr{X}_2'$ and its norm approaches $\|\cdot\|_{\mathscr{X}_1'}$ when ϵ tends to zero.

Consider preconditioning the time-dependent Stokes problem (8.24) where the coefficient operator is defined as

$$\tilde{\mathcal{A}}_\epsilon := \begin{pmatrix} \mathcal{I} - \epsilon^2\Delta & -\operatorname{grad} \\ \operatorname{div} & 0 \end{pmatrix}$$

For this problem, we shall construct a preconditioner which is uniformly convergent with respect to both h and ϵ.

① In view of §8.3, we know that $\tilde{\mathcal{A}}_0$ is bounded from $H_0(\operatorname{div},\Omega)\times L^2_0(\Omega)$ into its dual space. Hence we consider the operator $\tilde{\mathcal{A}}_\epsilon$ on

$$\mathscr{X}_\epsilon := \left(H_0(\operatorname{div},\Omega)\bigcap \epsilon[H^1_0(\Omega)]^d\right)\times L^2_0(\Omega) \quad \text{and} \quad \mathscr{X}_\epsilon' := \left(H_0(\operatorname{div},\Omega)' + \epsilon^{-1}[H^{-1}(\Omega)]^d\right)\times L^2_0(\Omega).$$

In this case, the two Brezzi conditions holds and $\tilde{\mathcal{A}}_\epsilon$ is an isomorphism. In turn, the canonical preconditioner is of the form

$$\tilde{\mathcal{D}}^{(1)}_\epsilon = \begin{pmatrix} (\mathcal{I} - \operatorname{grad}\operatorname{div} - \epsilon^2\Delta)^{-1} & 0 \\ 0 & \mathcal{I} \end{pmatrix}.$$

② We have seen that $\tilde{\mathcal{A}}_0$ is also bounded on $[L^2(\Omega)]^d \times \left(H^1(\Omega)\bigcap L^2_0(\Omega)\right)$ into its dual space. Furthermore, in order to guarantee the inf-sup condition, the proper norm for the pressure unknown is [81, 82]:

$$\sup_{\mathbf{v}\in[H^1_0(\Omega)]^d} \frac{(q, \nabla\cdot\mathbf{v})}{\|\mathbf{v}\|_{L^2\bigcap\epsilon H^1}} = \|\nabla q\|_{L^2+\epsilon^{-1}H^{-1}} \sim \|q\|_{H^1+\epsilon^{-1}L^2}.$$

Motivated by these observations, we can consider

$$\mathscr{X}_\epsilon := \Big[L^2(\Omega)\bigcap \epsilon H_0^1(\Omega)\Big]^d \times \Big(H^1(\Omega)\bigcap L_0^2(\Omega) + \epsilon^{-1}L_0^2(\Omega)\Big)$$

and

$$\mathscr{X}_\epsilon' := \Big[L^2(\Omega) + \epsilon^{-1}H^{-1}(\Omega)\Big]^d \times \Big(\big(H^1(\Omega)\bigcap L_0^2(\Omega)\big)'\bigcap \epsilon L_0^2(\Omega)\Big).$$

This choice of spaces gives a preconditioner of the form

$$\tilde{\mathcal{D}}_\epsilon^{(2)} = \begin{pmatrix} (\mathcal{I}-\epsilon^2\Delta)^{-1} & 0 \\ 0 & (-\Delta)^{-1}+\epsilon^2\mathcal{I} \end{pmatrix}.$$

Along this line, we can construct discrete block diagonal preconditioners for the time-dependent Stokes problem [52, 21].

Preconditioning the heat equation ⋆

In order to introduce a uniform preconditioner for the time-dependent Stokes equation, we still need to give a reasonable solver for $\mathcal{I}-\epsilon^2\Delta$ in $\tilde{\mathcal{D}}_\epsilon^{(2)}$. And this problem is in fact much more general. For example, it also appears in a simpler scalar time-dependent problem—the heat equation:

$$\begin{cases} u_t - \Delta u &= f, \quad \Omega; \\ u &= 0, \quad \partial\Omega; \\ u|_{t=0} &= u_0, \quad \Omega. \end{cases} \tag{8.44}$$

We discretize the first equation in (8.44) using the Backward Euler method for the time variable to obtain that

$$\frac{u_m - u_{m-1}}{t_m - t_{m-1}} - \Delta u_m = f_m,$$

where u_m and f_m are approximations to u and f, respectively, at time level t_m. Since u_0 is given, we can iteration over m to obtain approximate solutions $\{u_m\}_{m=0,1,\dots}$ to $u(t_m,\cdot)$, namely

$$(\mathcal{I}-\epsilon^2\Delta)u_m = f_m'. \tag{8.45}$$

In this case, $\epsilon^2 := t_m - t_{m-1}$ equals the time step-size and $f_m' := u_{m-1} + (t_m - t_{m-1})f_m$ is known. So we need to find out how to construct a preconditioner for operators like $\mathcal{A}_\epsilon := \mathcal{I}-\epsilon^2\Delta$ corresponding to the reaction-diffusion equation.

In particular, in order to solve the reaction-diffusion equation $\mathcal{A}_\epsilon u = f$ in Ω and $u|_{\partial\Omega}=0$ in the previous subsection, we have

$$\mathscr{X}_\epsilon := L^2(\Omega)\bigcap \epsilon H_0^1(\Omega) \quad \text{and} \quad \mathscr{X}_\epsilon' = L^2(\Omega) + \epsilon^{-1}H^{-1}(\Omega).$$

As ϵ goes to zero, both norms approaches the L^2-norm and $\mathcal{A}_\epsilon$ also tends to the identity.

In this setting, we have

$$\begin{aligned}\langle f,(\mathcal{I}-\epsilon^2\Delta)^{-1}f\rangle &= \langle(\mathcal{I}-\epsilon^2\Delta)(\mathcal{I}-\epsilon^2\Delta)^{-1}f,(\mathcal{I}-\epsilon^2\Delta)^{-1}f\rangle\\ &= \langle(\mathcal{I}-\epsilon^2\Delta)^{-1}f,(\mathcal{I}-\epsilon^2\Delta)^{-1}f\rangle-\epsilon^2\langle\Delta(\mathcal{I}-\epsilon^2\Delta)^{-1}f,(\mathcal{I}-\epsilon^2\Delta)^{-1}f\rangle\\ &= \|f_0\|_0^2+\epsilon^{-2}\|f_1\|_{-1}^2,\end{aligned}$$

where $f_0 := (\mathcal{I}-\epsilon^2\Delta)^{-1}f$ and $f_1 := -\epsilon^2\Delta(\mathcal{I}-\epsilon^2\Delta)^{-1}f$. Furthermore, we can get (cf. [83, Example 4.1])

$$\|f\|_{\mathscr{X}_\epsilon'}^2 = \langle f,(\mathcal{I}-\epsilon^2\Delta)^{-1}f\rangle = \langle(\mathcal{I}-\epsilon^2\Delta)u,u\rangle.$$

We can easily see the natural norm is

$$\|u\|_{\mathscr{X}_\epsilon} = \|u\|_{L^2\bigcap\epsilon H_0^1} := \left(\|u\|_0^2+\epsilon^2\|\nabla u\|_0^2\right)^{\frac{1}{2}}.$$

Hence, it is clear that

$$\|u\|_{\mathscr{X}_\epsilon} = \|f\|_{\mathscr{X}_\epsilon'}.$$

Although $\mathcal{I}-\epsilon^2\Delta$ is norm preserving from the above analysis, it is not yet clear how to construct a practical algorithm to solve it. We notice that the above semi-discrete problem or temporal discrete problem resembles our model problem—the Poisson's equation. In order to construct an efficient preconditioner for this equation, we can use the BPX preconditioner (5.19) in §5.3. In view of (5.18), on level l, we wish to have a smoother $\mathcal{S}_l$ behaves like

$$(\mathcal{S}_l v,v) = \frac{h_l^2}{h_l^2+\epsilon^2}(v,v), \quad \forall\, v\in V_l.$$

This smoother then defines the corresponding BPX preconditioner for the semi-discrete problem (8.45). Such a simple example shows how to handle a new problem from geometric point of view and it can be used as a component when solving the time-dependent Stokes problem.

8.5 Block preconditioners

In the previous section, we discussed how to construct canonical (natural) preconditioners based on the mapping property of the continuous Stokes equation. Now we shall consider the discrete Stokes problem arising in the mixed finite element method (such as the Taylor–Hood finite element method) in algebraic setting, i.e.,

$$\tilde{A}\begin{pmatrix}u\\p\end{pmatrix} = \begin{pmatrix}f\\g\end{pmatrix} \quad\text{and}\quad \tilde{A} := \begin{pmatrix}A & B^T\\ B & 0\end{pmatrix}. \tag{8.46}$$

Suppose $A\in\mathbb{R}^{n\times n}$, $B\in\mathbb{R}^{m\times n}$, $u\in\mathbb{R}^n$, and $p\in\mathbb{R}^n$. Let $N=n+m$. Assume that A is SPD and B has full rank. It is well-known that the coupled system $\tilde{A}$ is symmetric, indefinite, and non-singular.

Block diagonal and lower triangular method

If we consider the block diagonal preconditioner given in the previous section, the preconditioner can be written as

$$\tilde{D} := \begin{pmatrix} A^{-1} & 0 \\ 0 & M_p^{-1} \end{pmatrix}, \tag{8.47}$$

where M_p is the mass matrix corresponding to the pressure approximation space and, hence, it is well-conditioned; see Remark 3.23. It is easy to check that (8.47) is exactly the algebraic form of (8.43). Because both A and M_p are symmetric positive definite matrices, the preconditioner is well-defined.

Remark 8.11 (Block factorizations)**.** We can apply the following block factorizations to the matrix $\tilde{A}$ such that

$$\begin{aligned} \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} &= \begin{pmatrix} I_u & 0 \\ BA^{-1} & I_p \end{pmatrix} \begin{pmatrix} A & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I_u & A^{-1}B^T \\ 0 & -I_p \end{pmatrix} \\ &= \begin{pmatrix} A & 0 \\ B & S \end{pmatrix} \begin{pmatrix} I_u & A^{-1}B^T \\ 0 & -I_p \end{pmatrix} = \begin{pmatrix} I_u & 0 \\ BA^{-1} & -I_p \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & S \end{pmatrix}, \end{aligned}$$

where the matrix $S := BA^{-1}B^T$ is the Schur complement. In fact, $\tilde{D}$ in (8.47) can be viewed as an approximation of $\mathrm{diag}(A^{-1}, S^{-1})$. □

Remark 8.12 (Schur complement)**.** Since the A is SPD, the Schur complement $S = BA^{-1}B^T$ is symmetric and positive semi-definite. Moreover, if B has full rank, S is also SPD and we can apply the CG method to solve the Schur complement equation. However, generally speaking, $S^{-1}p$ cannot be computed efficiently with acceptable computational cost. Hence the Schur complement S should be approximated by some approximation $\hat{S}$. There are many different ways based on approximation of the Schur complement; see the survey paper [11]. □

We can also use the block lower triangular matrix to construct a preconditioner

$$\tilde{T} := \begin{pmatrix} A & 0 \\ B & \hat{S} \end{pmatrix}^{-1}. \tag{8.48}$$

In particular, if we replace A by its diagonal part D in the LU decomposition of Remark 8.11, then we get the so-called SIMPLE preconditioner

$$\tilde{T}_{\mathrm{SIMPLE}} := \begin{pmatrix} I_u & D^{-1}B^T \\ 0 & -I_p \end{pmatrix}^{-1} \begin{pmatrix} A & 0 \\ B & BD^{-1}B^T \end{pmatrix}^{-1}. \tag{8.49}$$

The name comes from the widely-used SIMPLE method for fluid problems.

Augmented Lagrangian method

One of the most well-known iterative method for solving (8.46) is probably the Uzawa method. As the last decomposition in Remark 8.11, we can factorize the coefficient matrix as

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} I_u & 0 \\ BA^{-1} & -I_p \end{pmatrix} \begin{pmatrix} A & B^T \\ 0 & S \end{pmatrix}.$$

This means the original linear system can be rewritten as

$$\begin{pmatrix} A & B^T \\ 0 & S \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ BA^{-1}f - g \end{pmatrix}.$$

As discussed in Remark 8.12, the pressure Schur complement equation might be too expensive to be solved exactly. We can apply an iterative method to solve it. For example, we can apply the Richardson's iteration for the second equation in the above system, i.e.,

$$p^{\text{new}} = p^{\text{old}} + \omega\Big(BA^{-1}f - g - Sp^{\text{old}}\Big) = p^{\text{old}} - \omega\Big(g - BA^{-1}f + BA^{-1}B^Tp^{\text{old}}\Big).$$

Hence we can write the above iteration as an alternative direction method

$$Au^{\text{new}} = f - B^Tp^{\text{old}}, \quad p^{\text{new}} = p^{\text{old}} - \omega(g - Bu^{\text{new}}). \tag{8.50}$$

The method (8.50) is called the Uzawa iteration and it is just the Richardson iteration for the Schur complement equation. As we have discussed in §2.1, the method converges with an appropriate scaling factor ω but the convergence rate is usually very slow. One way to speedup the convergence is to apply the Augmented Lagrangian method (cf., for example, [59]):

$$(A + \epsilon^{-1}B^TB)u^{\text{new}} = f + \epsilon^{-1}B^Tg - B^Tp^{\text{old}}, \quad p^{\text{new}} = p^{\text{old}} - \epsilon^{-1}(g - Bu^{\text{new}}). \tag{8.51}$$

Remark 8.13 (Uzawa method and Augmented Lagrangian method)**.** It is easy to see that the Augmented Lagrangian (AL) method is just the Uzawa method for the modified equation

$$\tilde{A}_\epsilon \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f + \epsilon^{-1}B^Tg \\ g \end{pmatrix}, \quad \text{where } \tilde{A}_\epsilon := \begin{pmatrix} A + \epsilon^{-1}B^TB & B^T \\ B & 0 \end{pmatrix}. \tag{8.52}$$

Furthermore, the damping factor ω is chosen to be ϵ^{-1}. □

Theorem 8.14 (Convergence rate of Augmented Lagrangian method)**.** Let $(u^{(0)}, p^{(0)})$ be a given initial guess and $(u^{(m)}, p^{(m)})$ be the iterates obtained via the Augmented Lagrangian method (8.51). Then we have

$$\begin{aligned} \|p - p^{(m)}\|_0 &\leqslant \Big(\frac{\epsilon}{\epsilon + \lambda_1}\Big)^m \|p - p^{(0)}\|_0, \\ \|u - u^{(m)}\|_A &\leqslant \sqrt{\epsilon}\|p - p^{(m-1)}\|_0 \leqslant \sqrt{\epsilon}\Big(\frac{\epsilon}{\epsilon + \lambda_1}\Big)^{m-1} \|p - p^{(0)}\|_0, \end{aligned}$$

where λ_1 is the minimal eigenvalue of $S = BA^{-1}B^T$.

Sketch of proof. From (8.51) and (8.52), we have

$$(A+\epsilon^{-1}B^TB)\big(u-u^{(m)}\big)=-B^T\big(p-p^{(m-1)}\big)$$

and

$$p-p^{(m)}=\Big(I-B(\epsilon A+B^TB)^{-1}B^T\Big)\big(p-p^{(m-1)}\big).$$

By the Shermann–Morrison–Woodburry formula, we have

$$Z:=B(\epsilon A+B^TB)^{-1}B^T=S_\epsilon-S_\epsilon(I+S_\epsilon)^{-1}S_\epsilon,\quad S_\epsilon:=\epsilon^{-1}BA^{-1}B^T.$$

It is easy to verify that

$$I-B(\epsilon A+B^TB)^{-1}B^T=I-S_\epsilon+S_\epsilon(I+S_\epsilon)^{-1}S_\epsilon=(I+S_\epsilon)^{-1}.$$

The above equality shows $\rho(Z)\leqslant 1$ and $p-p^{(m)}=(I+S_\epsilon)^{-1}\big(p-p^{(m-1)}\big)$. So the first estimate follows immediately. The second estimate is obtained by observing

$$\begin{aligned}\big\|u-u^{(m)}\big\|_A^2&=\Big((A+\epsilon^{-1}B^TB-\epsilon^{-1}B^TB)\big(u-u^{(m)}\big),u-u^{(m)}\Big)\\&\leqslant\epsilon\big(Z(p-p^{(m-1)}),p-p^{(m-1)}\big)\end{aligned}$$

and then applying the first estimate. □

According to Theorem 8.14, we can make the convergence as fast as we want by adjusting the parameter ϵ. However, the price to pay is that, in each iteration, we have to solve a nearly-singular system with coefficient matrix $A+\epsilon^{-1}B^TB$, which was discussed in [75]. We can also apply the Augmented Lagrangian method as a preconditioner

$$\tilde{T}_{\mathrm{AL}}:=\begin{pmatrix}A+\epsilon^{-1}B^TB & 0\\ B & \epsilon I\end{pmatrix}^{-1},\tag{8.53}$$

which is often referred to as the AL preconditioner [12].

The method is closely related to the grad-div stabilization [43] of the Stokes (or Navier–Stokes) problem:

$$\begin{cases}(I-\mu\Delta)\mathbf{u}-\epsilon^{-1}\nabla\nabla\cdot\mathbf{u}+\nabla p &=\mathbf{f},\quad\Omega;\\ \nabla\cdot\mathbf{u} &=0,\quad\Omega;\\ \mathbf{u} &=0,\quad\partial\Omega.\end{cases}\tag{8.54}$$

In this modified problem, the coercivity condition automatically holds on the discrete level for the $H_0(\mathrm{div})$-norm defined by (8.40). After discrezation by some mixed finite element method, we obtain discrete systems in the form of (8.51). We can apply the block preconditioners discussed in the previous subsection to solve the resulting discrete problems; see the survey and numerical experiments by He and Vuik [70].

8.6 Multigrid methods for the Stokes equation

Using a general multilevel iterative procedure, we can construct coupled geometric multigrid methods for the saddle-point problem (8.46) as well. For the transfer operators, by applying the similar ideas as in multigrid methods for scalar equations, we can construct prolongations and restrictions for velocity and pressure variables separately. Coarse-level solvers can also apply the same multilevel cycles as in §6.2. So we only discuss smoothers for the Stokes system. Analysis and numerical experiments using different smoothers have been reviewed in the survey by Larin and Reusken [74]. Apparently, the block preconditioners discussed in the previous section can also be applied as smoothers for coupled multigrid methods. In this section, we discuss two other widely-used smoothers in practice.

Braess–Sarazin smoother

The Braess–Sarazin smoother was introduced in [19] and can be written as

$$\begin{pmatrix} u^{(m+1)} \\ p^{(m+1)} \end{pmatrix} = \begin{pmatrix} u^{(m)} \\ p^{(m)} \end{pmatrix} + \begin{pmatrix} \omega D & B^T \\ B & 0 \end{pmatrix}^{-1} \left[\begin{pmatrix} f \\ 0 \end{pmatrix} - \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u^{(m)} \\ p^{(m)} \end{pmatrix} \right], \tag{8.55}$$

where ω is a positive parameter. This method mimics the damped Jacobi smoother for the Poisson's equation.

We need to solve, in each smoothing step, the following the linear system

$$\begin{pmatrix} \omega D & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta u^{(m)} \\ \delta p^{(m)} \end{pmatrix} = \begin{pmatrix} f - Au^{(m)} - B^T p^{(m)} \\ -Bu^{(m)} \end{pmatrix}.$$

The second equation ensures the discrete divergence free condition, i.e.,

$$Bu^{(m+1)} = B\big(u^{(m)} + \delta u^{(m)}\big) = 0, \quad m = 1, 2, \ldots$$

Apparently, the Braess–Sarazin smoother can be reduced to an auxiliary pressure equation

$$\big(BD^{-1}B^T\big)\,\delta p^{(m)} = \omega Bu^{(m)} + BD^{-1}\big(f - Au^{(m)} - B^T p^{(m)}\big).$$

The coefficient matrix $\hat{S} := BD^{-1}B^T$ is similar to a scaled discrete Laplace operator on the pressure space. In practice, we can solve it approximately using an iterative method for example.

Vanka smoother

Next we introduce a smoother originally proposed by Vanka [112]. In the context of finite element methods, the Vanka-type smoothers are just block Gauss–Seidel (or Jacobi) methods. Each block contains degrees of freedom in an element or a set of elements. One of the popular

variant of Vanka-type smoothers is the so-called *pressure-oriented Vanka smoother* for continuous pressure approximations. We only discuss this special case of Vanka smoother here.

For each pressure variable indexed by i ($1 \leqslant i \leqslant m$), let the set of velocity indices that are "connected" to i as

$$\mathsf{S}_i := \{1 \leqslant j \leqslant n \,:\, b_{i,j} \neq 0\},$$

where $b_{i,j}$ is the (i,j)-entry of the matrix B. So we can define an injection to the set of variables $\{u_j \,(j \in \mathsf{S}_i),\, p_i\}$, i.e.,

$$I_i = \begin{pmatrix} I_{u,i} & 0 \\ 0 & I_{p,i} \end{pmatrix} \in \mathbb{R}^{(|\mathsf{S}_i|+1)\times(n+m)},$$

where $I_{p,i}p = p_i$ and $I_{u,i}u = (u_j)_{j\in\mathsf{S}_i}$ are the corresponding injection matrices for velocity and pressure, respectively.

We can then apply a multiplicative Schwarz method (or the so-called Full Vanka smoother):

$$I - \tilde{T}_{\text{FVanka}}\tilde{A} = \prod_{i=1}^{m}\Big(I - I_i^T \tilde{A}_i^{-1} I_i \tilde{A}\Big), \tag{8.56}$$

where

$$\tilde{A}_i = I_i \tilde{A} I_i^T = \begin{pmatrix} A_i & B_i^T \\ B_i & 0 \end{pmatrix} \in \mathbb{R}^{(|\mathsf{S}_i|+1)\times(|\mathsf{S}_i|+1)}.$$

We can also use a simplified version (i.e., the Diagonal Vanka smoother):

$$I - \tilde{T}_{\text{DVanka}}\tilde{A} = \prod_{i=1}^{m}\Big(I - I_i^T \tilde{D}_i^{-1} I_i \tilde{A}\Big), \tag{8.57}$$

where

$$\tilde{D}_i = \begin{pmatrix} D_i & B_i^T \\ B_i & 0 \end{pmatrix} \in \mathbb{R}^{(|\mathsf{S}_i|+1)\times(|\mathsf{S}_i|+1)}.$$

In this case, due to the special nonzero pattern of $\tilde{D}_i$, it can be solved very efficiently.

8.7 Homework problems

HW 8.1. Show the equation (8.16). Hint: In $\mathbb{R}^2$, taking divergence of the symmetric gradient, we get

$$\begin{aligned} \nabla\cdot\boldsymbol{\varepsilon}(\mathbf{u}) &= \begin{pmatrix} \partial_1^2 u_1 + \frac{1}{2}\partial_2(\partial_2 u_1 + \partial_1 u_2) \\ \partial_2^2 u_2 + \frac{1}{2}\partial_1(\partial_1 u_2 + \partial_2 u_1) \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{2}(\partial_1^2 u_1 + \partial_2^2 u_1) + \frac{1}{2}\partial_1(\partial_1 u_1 + \partial_2 u_2) \\ \frac{1}{2}(\partial_1^2 u_2 + \partial_2^2 u_2) + \frac{1}{2}\partial_2(\partial_1 u_1 + \partial_2 u_2) \end{pmatrix} = \frac{1}{2}\Delta\mathbf{u} + \frac{1}{2}\nabla\nabla\cdot\mathbf{u}. \end{aligned}$$

HW 8.2. Derive the weak form (8.26) of the Stokes equations (8.25).

HW 8.3. Give the complete proof of Theorem 8.14.

Chapter 9

Optimization Problems

Mathematical optimization (mathematical programming or optimization) is the selection of a "best" element (with regard to certain criterion) from some set of available alternatives. Many optimization problems can be written as variational inequalities (VIs); for example, many problems in economics, operations research, and transportation equilibrium problems. In this chapter, we discuss multilevel iterative methods for solving finite-dimensional variational inequalities.

9.1 Model problems

VIs arise from a wide range of application areas, like mechanics, control theory, engineering, and finance. After several decades of development, this subject has become very rich on both theory and numerics. For a general discussion on the existence and regularity, we refer the interested readers to [72]. For a comprehensive discussion on numerical methods for VIs, we refer to Glowinski [62].

A model variational inequality

Let $a[\cdot,\cdot]$ and $f(\cdot)$ be a symmetric bilinear form and a linear form, respectively, and $\chi \in H_0^1(\Omega)$ be an admissible obstacle (for simplicity, we assume the zero boundary condition). Consider the following elliptic variational inequality (or the obstacle problem): Find $u \in \mathcal{K}_\chi := \{v \in H_0^1(\Omega) : v \geqslant \chi\}$, such that

$$a[u, v-u] \geqslant f(v-u), \quad \forall\, v \in \mathcal{K}_\chi. \tag{9.1}$$

After transformation $w := u - \chi$, we arrive at a new problem with a simple inequality constraint: Find $w \in \mathcal{K}_0 := \{v \in H_0^1(\Omega) : v \geqslant 0\}$, such that

$$a[w, v-w] \geqslant f_0(v-w) := f(v-w) - a[\chi, v-w], \quad \forall\, v \in \mathcal{K}_0. \tag{9.2}$$

For problem (9.1), the *Lagrange multiplier* can be defined as σ_1 such that

$$\langle \sigma_1(u), \varphi \rangle := f(\varphi) - a[u, \varphi], \quad \forall\ \varphi \in H_0^1(\Omega). \tag{9.3}$$

On the other hand, for (9.2), notice, for any $\varphi \in H_0^1(\Omega)$, that

$$\langle \sigma_2(w), \varphi \rangle = f_0(\varphi) - a[w, \varphi] = f(\varphi) - a[u, \varphi] = \langle \sigma_1(u), \varphi \rangle .$$

It is easy to see that

$$\langle \sigma_1(u), v - u \rangle \leqslant 0, \quad \forall\ v \in \mathcal{K}_\chi, \tag{9.4}$$

or

$$\langle \sigma_2(w), v - w \rangle \leqslant 0, \quad \forall\ v \in \mathcal{K}_0.$$

On the other hand, if σ is the Lagrange multiplier of (9.1), we have

$$\langle \sigma(v) - \sigma(u), \varphi \rangle = -a[v - u, \varphi], \quad \forall\ \varphi \in H_0^1(\Omega).$$

Hence,

$$\langle \sigma(v) - \sigma(u), v - u \rangle = -a[v - u, v - u] = -\left\| v - u \right\|^2, \quad \forall\ v, u \in H_0^1(\Omega). \tag{9.5}$$

Hence, we have $\langle \sigma(v) - \sigma(u), v - u \rangle \leqslant 0$, for any $v, u \in H_0^1(\Omega)$, i.e., σ is a *monotone* operator.

Remark 9.1 (Uniqueness of solution)**.** Notice that if both u_1 and u_2 are solutions of the variational inequality (9.1), by the monotonicity of σ, $\left\| u_1 - u_2 \right\| = 0$ and then we obtain the uniqueness.

As before, we assume that $\mathcal{A} : H_0^1(\Omega) \mapsto H^{-1}(\Omega)$ be the operator corresponding to $a[\cdot, \cdot]$. An frequently equivalent formulation of (9.1) is the so-called linear complementarity problem (LCP): Find a solution $u \in H_0^1(\Omega)$ such that

$$\begin{cases} \mathcal{A}u - f \geqslant 0 \\ u - \chi \geqslant 0 \\ \langle \mathcal{A}u - f, u - \chi \rangle = 0. \end{cases} \tag{9.6}$$

The last equation is the so-called *complementarity condition.*

Proof. If u is a solution of LCP (9.6), then for any $v \in H_0^1(\Omega)$ and $v \geqslant \chi$ we have

$$\langle \mathcal{A}u - f, u - v \rangle = \langle \mathcal{A}u - f, \chi - v \rangle \leqslant 0,$$

in view of the complementarity condition and the sign condition of $\mathcal{A}u - f$. On the other hand, if u is solution of (9.1), it is trivial to see that u satisfies the first two conditions of LCP. The complementarity condition is obtained by taking $v = u + (u - \chi)$ and $v = \chi$. □

Finite element discretization for VIs

As discussed in §3.1, the domain Ω is partitioned into a quasi-uniform simplexes of size h; this mesh is denoted by $\mathcal{M}_h$. Let $V_h \subset W_0^{1,\infty}(\Omega)$ be the continuous piecewise linear finite element space associated with $\mathcal{M}_h$. The obstacle problem (9.2) can be approximated by a finite element function $u_h \in \mathcal{K}_0 \bigcap V_h$ satisfying:

$$a[u_h, v_h - u_h] \geqslant f_0(v_h - u_h), \quad \forall\ v_h \in \mathcal{K}_0 \bigcap V_h. \tag{9.7}$$

As before, we denote all the interior nodes of the partition $\mathcal{M}_h$ by $\mathring{G}(\mathcal{M}_h)$. Let $\{\phi_z\}_{z\in\mathring{G}(\mathcal{M}_h)}$ be the canonical linear finite element basis of the mesh $\mathcal{M}_h$. Let $u = u_h := \sum_{z\in\mathring{G}(\mathcal{M}_h)} u_z\phi_z$ and $\underline{u} = (u_z)_{z\in\mathring{G}(\mathcal{M}_h)}$, the discrete solution and its nodal value vector (primal vector form), respectively. Hence we have the following linear system

$$(\underline{v} - \underline{u})^T(A\underline{u} - \vec{f_0}) \geqslant 0, \quad \forall\ \underline{v} \geqslant 0, \tag{9.8}$$

where A is the corresponding stiffness matrix of the bilinear form and $\vec{f_0}$ is the dual vector form of f_0.

Remark 9.2. One can prove (see for example [34]) that the l^2-error between the exact solution $\underline{u}$ of (9.8) and any approximation solution $\underline{v}$ satisfies that

$$\|\underline{v} - \underline{u}\|_0 \lesssim \|(\vec{f_0} - A\underline{v})_+\|_0,$$

where the vector $(\vec{f_0} - A\underline{v})_+$ is defined element-wise by

$$(\vec{f_0} - A\underline{v})_{+,i} = \begin{cases} (\vec{f_0} - A\underline{v})_i & \text{if } \underline{v}_i > 0 \\ \min\{(\vec{f_0} - A\underline{v})_i, 0\} & \text{if } \underline{v}_i = 0. \end{cases}$$

Error and residual

As usual, we define the energy functional as following

$$\mathcal{F}(v) := \frac{1}{2}a[v, v] - f(v).$$

Then it follows that

$$\mathcal{F}(v) - \mathcal{F}(u) = \frac{1}{2}\left|\!\left|\!\left| v - u \right|\!\right|\!\right|^2 - \langle \sigma, v - u\rangle, \quad \forall\ v \in \mathcal{K}_\chi. \tag{9.9}$$

Consider finite element solutions, u_h and w_h for problems (9.1) and (9.2), respectively. The differences, in terms of energy, between the finite element solutions and the exact solutions can be written as

$$\begin{aligned} \mathcal{F}(u_h) - \mathcal{F}(u) &= \frac{1}{2}\left|\!\left|\!\left| u_h - u \right|\!\right|\!\right|^2 - \langle \sigma, u_h - u\rangle \\ \mathcal{F}(w_h) - \mathcal{F}(w) &= \frac{1}{2}\left|\!\left|\!\left| w_h - w \right|\!\right|\!\right|^2 - \langle \sigma, w_h - w\rangle. \end{aligned} \tag{9.10}$$

It is easy to see that the variational inequality (9.2) can be written as the following quadratic minimization problem:

$$\min_{w\in\mathcal{K}_0} \frac{1}{2}a[w,w] - f_0(w). \tag{9.11}$$

For finite element approximation, we compute the finite dimensional minimization problem

$$\min_{w_h\in V_h\bigcap\mathcal{K}_0} \frac{1}{2}a[w_h,w_h] - f_0(w_h). \tag{9.12}$$

Suppose $\hat{w}_h$ is an approximate solution of the above minimization problem. Then the defect $e_h := w_h - \hat{w}_h$ satisfies

$$\min_{\hat{w}_h+e_h\in V_h\bigcap\mathcal{K}_0} \frac{1}{2}a[\hat{w}_h+e_h,\hat{w}_h+e_h] - f_0(\hat{w}_h+e_h) = \frac{1}{2}a[e_h,e_h] - f_0(e_h) + a[\hat{w}_h,e_h] + C,$$

i.e.,

$$\min_{\hat{w}_h+e_h\in V_h\bigcap\mathcal{K}_0} \frac{1}{2}a[e_h,e_h] - \langle\sigma(\hat{w}_h),e_h\rangle\,. \tag{9.13}$$

Notice that it is in the same form as (9.12) but replacing f_0 by $\sigma(\hat{w}_h)$. Hence the above problem can be viewed as the error problem; compare this with the error equation in the linear case (1.38). Whence we have e_h, we can update $w_h = \hat{w}_h + e_h$ as in the linear case.

9.2 Nonlinear equation and unconstrained minimization

We first consider the unconstrained optimization problem

$$u = \operatorname*{argmin}_{v\in\mathscr{V}} \mathcal{F}(v). \tag{9.14}$$

If $\mathcal{F} : \mathscr{V} \mapsto \mathbb{R}$ is a convex function, then the problem is called a convex optimization (or convex programming). If $\mathcal{F}$ is differentiable, a minimizer satisfies the well-known *first-order optimization condition*

$$\mathcal{G}(u) := \mathcal{F}'(u) = 0, \tag{9.15}$$

where $\mathcal{G} : \mathscr{V} \mapsto \mathbb{R}$ is the Frechet derivative of $\mathcal{F}$. If $\mathcal{F}$ is convex, then (9.14) is equivalent for solving the nonlinear equation (9.15). In particular, if $\mathcal{F}$ is quadratic, then the problem is called a quadratic optimization. Apparently, if $\mathcal{F}$ is a convex quadratic functional, then the problem (9.14) is equivalent to our model problem (2.1), $\mathcal{A}u = f$, with an SPD operator $\mathcal{A} = \mathcal{G}'$.

Nonlinear solvers

In general, the problem (9.14) is much more difficult to solve than (2.1) due to its nonlinearity. We can employ a nonlinear iterative solver to linearize (9.15) to obtain a linear (differential) equation, i.e., linearization then discretization. For example, we may use the standard

approaches, like the Picard method or the Newton–Raphson method. Another strategy is to discretize the continuous problem (9.14) or (9.15) in order to obtain a nonlinear algebraic problem

$$u = \operatorname*{argmin}_{v \in \mathbb{R}^N} \mathcal{F}(v) \tag{9.16}$$

or

$$\mathcal{G}(u) = 0. \tag{9.17}$$

The idea of coarse-grid correction used in Algorithm 3.1 does not apply any more here because the classical residual equation is linear. There are basically two approaches to apply the multilevel idea on this problem—The first approach is to linearize the problem and then apply multigrid methods to linear problems; The second one is to apply multigrid directly to the nonlinear problem using the so-called *Full Approximation Scheme* (FAS).

Newton–Raphson method

There are different ways to linearize a nonlinear problem like (9.15). For simplicity, we now only consider discrete version of the nonlinear equation, i.e., $\mathscr{V} = \mathbb{R}^N$. The most popular approach is the so-called Newton–Raphson (or Newton) linearization. We apply second-order Taylor expansion to approximate the objective function near the current iteration $u^{(k)} \in \mathbb{R}^N$, i.e.,

$$\mathcal{F}(u^{(k)} + e) \approx \mathcal{F}(u^{(k)}) + (\nabla \mathcal{F}(u^{(k)}),\, e) + \frac{1}{2}(\nabla^2 \mathcal{F}(u^{(k)})e,\, e).$$

In order to find a good incremental correction step, we can consider

$$e^{(k)} = \operatorname*{argmin}_{e \in \mathbb{R}^N} \frac{1}{2}(\nabla^2 \mathcal{F}(u^{(k)})e,\, e) + (\nabla \mathcal{F}(u^{(k)}),\, e) = -\big[\nabla^2 \mathcal{F}(u^{(k)})\big]^{-1} \nabla \mathcal{F}(u^{(k)}).$$

This is the Newton–Raphson iteration

$$u^{(k+1)} = u^{(k)} - \big[\nabla^2 \mathcal{F}(u^{(k)})\big]^{-1} \nabla \mathcal{F}(u^{(k)}). \tag{9.18}$$

In the above iteration step, we need to solve a linear system, the Jacobian equation:

$$\mathcal{A} e^{(k)} := \big[\nabla^2 \mathcal{F}(u^{(k)})\big] e^{(k)} = -\nabla \mathcal{F}(u^{(k)}) =: r^{(k)}. \tag{9.19}$$

We can employ the methods discussed in the previous chapters to solve such equations.

Listing 9.1: Newton–Raphson method

```
Given an initial guess u ∈ 𝒱 and set r ← −∇ℱ(u);
while ‖r‖ > ε
    solve the Jacobian equation ∇²ℱ(u)e = r;
    find a good stepsize α > 0;
    u ← u + α e;  r ← −∇ℱ(u);
end
```

The Newton-Raphson method converges very fast (second-order convergence) if the initial guess is close enough to the exact solution. So if a good initial guess is available, the main computation cost of the above algorithm is assembling the Jacobian systems and solving it to acceptable accuracy. If we apply a multigrid algorithm to solve the Jacobian systems, then this method is usually called Newton-Multigrid method. Similarly, another wide-used approach to apply a domain decomposition preconditioned Krylov method to solve the Jacobian systems, then this method is called Newton-Schwarz-Krylov method. Note that we might not need to assemble the Jacobian system explicitly; instead, we can use a Jacobian-free scheme.

Full approximation scheme

For the nonlinear equation (9.15), the residual corresponding to an approximate solution v can be defined as

$$r := -\mathcal{G}(v) = \mathcal{G}(u) - \mathcal{G}(v) \tag{9.20}$$

However, because $\mathcal{G}$ is not linear, $r \neq \mathcal{G}(u - v)$. In FAS, instead of considering the residual equation as in the linear case, the full equation is solved on the coarse grids.

We now use the following two-grid method to demonstrate the basic idea of FAS. Let $u^{(1)}$ be an approximate solution on the fine grid after several steps of relaxation. On the coarse grid, according to (9.20), we need to solve the following nonlinear equation

$$\mathcal{G}_c\big(u_c^{(1)}\big) - \mathcal{G}_c\big(\mathcal{I}_c^T u^{(1)}\big) = r_c = \mathcal{I}_c^T r = -\mathcal{I}_c^T \mathcal{G}\big(u^{(1)}\big). \tag{9.21}$$

This means, on the coarse level, a problem similar to the original problem (with different right-hand side) should be solved

$$\mathcal{G}_c\big(u_c^{(1)}\big) = \mathcal{G}_c\big(\mathcal{I}_c^T u^{(1)}\big) - \mathcal{I}_c^T \mathcal{G}\big(u^{(1)}\big). \tag{9.22}$$

Usually the right-hand side of the above equation is denoted as $\tau_c(u^{(1)})$ and is called the *tau correction*. Note that the coarse-level equation $\mathcal{G}_c$ can be obtained from the discretization on the coarse grid. We can also use the Galerkin method

$$\mathcal{G}_c(u_c) := \mathcal{I}_c^T \mathcal{G}(\mathcal{I}_c u_c).$$

Once the problem (9.22) is solved, we correct the approximation as

$$u^{(2)} = u^{(1)} + \mathcal{I}_c\big(u_c^{(1)} - \mathcal{I}_c^T u^{(1)}\big). \tag{9.23}$$

Apparently the above idea can be applied recursively as we discussed in §6.2. Because the coarse-grid problem is solved for the full approximation, rather than the error, the method is named as the Full Approximation Scheme. In this algorithm, evaluating the nonlinear function is usually the most expensive part computationally. We summarize the two-grid FAS algorithm as follows:

Listing 9.2: Full Approximation Scheme

```
1 Given an initial guess u ∈ V;
2 Solve the nonlinear equation G_c(u_c) = G_c(I_c^T u) − I_c^T G(u);
3 u ← u + I_c(u_c − I_c^T u);
```

Subspace correction methods for convex minimization

Apparently, the idea of subspace correction methods can be easily extended to unconstrained convex minimization problems here. The convergence analysis of SSC and PSC methods has been given by Tai and Xu [108].

9.3 Constrained minimization

In this section, we consider multilevel solvers for constrained minimization problems

$$u = \underset{v \in \mathcal{K}_0}{\operatorname{argmin}} \mathcal{F}(v) := \frac{1}{2} a[v, v] - f(v), \tag{9.24}$$

which is equivalent to the variational inequality (9.2).

Projected full approximation method

Since the the above problem is nonlinear, we can apply the Full Approximation Scheme introduced in the previous section to solve this problem. And this is the so-called Projected Fully Approximation Scheme (PFAS) by Brandt and Cryer [34].

As we have discussed in the previous chapters, we first need to find a relatively simple iterative procedure which is able to dump the high-frequency part of the error quickly. In order to obtain a smoother for (9.24), we can employ the simple iterative methods discussed in §2.1 and then apply a projection step to ensure the new iteration stays in the feasible set. For example, if u^{old} is the previous iteration and u^{GS} is the iteration after one or several Gauss-Seidel sweeps, then $u^{\text{new}} := \max\{0, u^{\text{GS}}\} \in \mathcal{K}_0$ is the new iteration. This method is naturally called the Projected Gauss-Seidel (PGS) method.

At some point PGS will not reduce error efficiently any more, we then apply FAS to approximate the error on a coarser level and continue this procedure until the coarsest level where the nonlinear problem can be solved quickly and accurately. To ease the notation, we explain the idea using a two-grid algorithm for now. We first solve the general LCP problem on a fine level with a given right-hand side f_l

$$\begin{cases} \mathcal{A}u \geqslant f \\ u \geqslant 0 \\ \langle \mathcal{A}u - f, u \rangle = 0. \end{cases}$$

using the PGS method or some other smoother to obtain an approximate solution $u^{(1)}$. Then we solve the above LCP on a coarse level with the right-hand side

$$f_c := \mathcal{I}_c^T \left(f - \mathcal{A}u^{(1)}\right) + \mathcal{A}_c \mathcal{I}_c^T u^{(1)}$$

to obtain an approximation $u_c^{(1)}$. In turn, an improved approximation is given by

$$u^{(2)} = u^{(1)} + \mathcal{I}_c\big(u_c^{(1)} - \mathcal{I}_c^T u^{(1)}\big).$$

Interior point method

For simplicity, we now consider the constrained minimization problem (9.2) on the finite dimensional space $\mathbb{R}^N$, that is to say

$$u = \operatorname*{argmin}_{v \geqslant 0,\, v \in \mathbb{R}^N} \mathcal{F}(v) := \frac{1}{2} v^T A v - f^T v. \tag{9.25}$$

In this case, the Lagrange multiplier $\sigma \in \mathbb{R}^N$ satisfies that $\sigma = -\mathcal{G}(u)$. Then we have the first-order optimality condition

$$\begin{aligned} &\sigma + \mathcal{G}(u) = 0, \qquad && \sigma \leqslant 0, \\ &U\sigma = 0, && u \geqslant 0. \end{aligned}$$

Here we use a convention often employed in the literature $U := \operatorname{diag}\{u_1, \ldots, u_N\}$; similarly, we will denote $\Sigma := \operatorname{diag}\{\sigma_1, \ldots, \sigma_N\}$.

The condition $U\sigma = 0$ (or equivalently, $u_i \sigma_i = 0$ for any $i = 1, \ldots, N$) is usually called the complementarity condition. We now try to relax this condition such that $U\sigma = \mu \mathbf{1}$, where μ is a positive penalty parameter and $\mathbf{1}$ is an all-one vector. At the same time, we try to maintain the iterative solution (u, σ) strictly in the primal-dual feasible set, i.e., $u > 0$ and $\sigma < 0$. Hence we need to solve a system of nonlinear equations:

$$\begin{cases} \sigma + \mathcal{G}(u) &= 0, \\ U\sigma - \mu \mathbf{1} &= 0. \end{cases}$$

We apply the Newton's method for this system and obtain an iterative method

$$\begin{cases} A\delta u + \delta\sigma &= -\sigma - \mathcal{G}(u) \\ \Sigma \delta u + U \delta\sigma &= \mu \mathbf{1} - U\sigma \end{cases} \quad \text{or} \quad \begin{pmatrix} A & I \\ \Sigma & U \end{pmatrix} \begin{pmatrix} \delta u \\ \delta \sigma \end{pmatrix} = \begin{pmatrix} f - Au - \sigma \\ \mu \mathbf{1} - U\sigma \end{pmatrix}.$$

Upon solving this linear system, we can obtain a new iteration. Furthermore, in the above system, I, Σ, and U are all known diagonal matrices, we only need to solve the Schur complement problem

$$(A - U^{-1}\Sigma)\delta u = \mu U^{-1}\mathbf{1} + f - Au. \tag{9.26}$$

Moreover, since $\sigma < 0$ and $u > 0$, the above equation is well-defined and the coefficient matrix is SPD. We can then apply a multilevel iterative method discussed in the previous chapters to solve it efficiently; see [9] for details.

Monotone multigrid method

Now suppose we hierarchical meshes, $\{\mathcal{M}_h^0, \ldots, \mathcal{M}_h^j\}$ and let A_l, b_l, $l = 0, \ldots, j$ are the stiffness matrices and right-hand-side vectors corresponding to the partition $\mathcal{M}_h^l$, respectively. As usual, $\mathcal{M}_h^j$ is the finest mesh. We denote the linear finite element space by V_h^l associated with mesh $\mathcal{M}_h^l$.

We need two kinds of orthogonal projections onto the finite element space V_h^l. The L^2-projections $Q_l : V_h^j \to V_h^l$ are defined by

$$(Q_l v_h, \phi_l) = (v_h, \phi_l), \quad \phi_l \in V_h^l, \tag{9.27}$$

and the energy projections $\Pi_l : V_h^j \to V_h^l$ by

$$a[\Pi_l v_h, \phi_l] = a[v_h, \phi_l], \quad \phi_l \in V_h^l. \tag{9.28}$$

We first define multigrid methods recursively. For a given initial guess $w_j^{(0)} \in V_h^j \bigcap \mathcal{K}_0$. A coarse grid correction is performed: computing the approximate defect $e_{j-1}^{(0)} = \Pi_{j-1}(w_h - w_j^{(0)}) \in V_h^{j-1}$ as the solution of the quadratic programming problem

$$\min_{e_{j-1}^{(0)} \in V_h^{j-1},\ w_j^{(0)} + e_{j-1}^{(0)} \bigcap \mathcal{K}_0} \frac{1}{2} a[e_{j-1}^{(0)}, e_{j-1}^{(0)}] - \langle \sigma(w_j^{(0)}), e_{j-1}^{(0)} \rangle. \tag{9.29}$$

Then let $w_j^{(1)} = w_j^{(0)} + e_{j-1}^{(0)}$. Then we apply m steps of post-smoothing scheme, like *projected SOR* to obtain $w_j^{(m+1)}$. For the coarse correction step, instead of solving the problem on the coarser level $j-1$ exactly, we can solve it by the same multigrid procedure described here. In this way, we obtain a recursive multigrid V-cycle. If we perform coarse grid correction twice at each level, then we get a W-cycle.

One problem with this procedure is that e_{j-1} and w_j are in different levels. To avoid this difficulty, we propose the following coarse grid correction scheme instead of (9.29):

$$\min_{d_{j-1}^{(0)} \in V_h^{j-1} \bigcap \mathcal{K}_0} \frac{1}{2} a[d_{j-1}^{(0)}, d_{j-1}^{(0)}] - \langle \sigma(w_j^{(0)}), d_{j-1}^{(0)} \rangle. \tag{9.30}$$

And then $w_j^{(1)} = w_j^{(0)} + d_{j-1}^{(0)}$ which is always in $\mathcal{K}_0$ because both $w_j^{(0)}$ and $d_{j-1}^{(0)}$ are in $\mathcal{K}_0$ by definition. It is easy to check that the local obstacles in this method are monotone in the sense of Kornhuber [73]. Then we get the similar V-cycle or W-cycle multigrid method as for linear problems expect we need to add a projection step to project the iterates to $\mathcal{K}_0$.

Remark 9.3. This method is shown to be not very good by Tai's test example. The reason is that the coarse grid correction only works when the current approximation is less than the exact solution in the method.

9.4 Constraint decomposition method

It is known the general V-cycle can be written as a successive subspace correction method. For a sequence of search directions $\{\phi_i\}_{i=1}^N$ such that $V_h^j := \text{span}\{\phi_i\}_{i=1}^N$. We can construct a numerical method for find the minimizer of (9.12) as a sequential quadratic programming method. Starting from an initial guess $w_j^{(0)} \in V_h^j \bigcap \mathcal{K}_0$, at each iteration, we solve

$$\min_{w_j^{(0)}+\alpha\phi_1 \in V_h^j \bigcap \mathcal{K}_0} \frac{1}{2} a[w_j^{(0)} + \alpha\phi_1, w_j^{(0)} + \alpha\phi_1] - f_0(w_j^{(0)} + \alpha\phi_1). \tag{9.31}$$

Similar to the discussion in the previous section, we need to solve a discrete problem

$$\min_{w_j^{(0)}+\alpha\phi_1 \in V_h^j \bigcap \mathcal{K}_0} \frac{1}{2} a[\phi_1, \phi_1]\alpha^2 - \langle \sigma(w_j^{(0)}), \phi_1 \rangle \alpha. \tag{9.32}$$

Then the new iterate is obtained by $w_j^{(1)} = w_j^{(0)} + \alpha\phi_1$. Similarly, we start from $w_j^{(1)}$ and search in the direction ϕ_2 to obtain $w_j^{(2)}$, and so on.

If we choose $\text{span}\{\phi_i\}_{i=1}^N$ as the canonical nodal basis of V_h^j, then it is just usual nonlinear or projected Gauss-Seidel method. To take advantage of multilevel basis, it is natural to choose $\text{span}\{\phi_i\}_{i=1}^N = \{\phi_1^j, \ldots, \phi_{N_j}^j, \phi_1^{j-1}, \ldots, \phi_{N_{j-1}}^{j-1}, \ldots, \phi_1^1, \ldots, \phi_{N_1}^1\}$. It falls into the category of *extended relaxation methods*. The problem with this procedure is that ϕ_i might not be in the finest level j, which costs extra computation effort to enforce the constraints $w_j^{(i-1)} + \alpha\phi_i \in V_h^j \bigcap \mathcal{K}_0$.

See Tai [107] for details.

Bibliography

[1] O. Axelsson. A survey of algebraic multilevel iteration (AMLI) methods. *BIT Numerical Mathematics*, 43:863–879, 2003.

[2] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numerische Mathematik*, 48(5):479–498, 1986.

[3] O. Axelsson and G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

[4] I. Babuška. Error-bounds for finite element method. *Numerische Mathematik*, 16(4):322–333, 1971.

[5] N. S. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966.

[6] R. E. Bank and T. Dupont. An optimal order process for solving finite element equations. *Mathematics of Computation*, 36(153):35–51, 1981.

[7] R. E. Bank and T. F. Dupont. Analysis of a two-level scheme for solving finite element equations. Technical report, 1980.

[8] R. E. Bank, T. F. Dupont, and H. Yserentant. The hierarchical basis multigrid method. *Numerische Mathematik*, 52(4):427–458, 1988.

[9] R. E. Bank, P. E. Gill, and R. F. Marcia. Interior methods for a class of elliptic variational inequalities. In L. T. Biegler, M. Heinkenschloss, O. Ghattas, and B. van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, pages 218–235. 2003.

[10] R. E. Bank and H. Yserentant. Multigrid convergence: A brief trip down memory lane. *Computing and Visualization in Science*, 13(4):147–152, 2010.

[11] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, May 2005.

[12] M. Benzi and M. A. Olshanskii. An Augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing*, 28(6):2095–2113, 2006.

[13] J. Bergh and J. Lofstrom. *Interpolation spaces: an introduction*, volume 223. Springer Science & Business Media, 2012.

[14] D. Boffi, F. Brezzi, and M. Fortin. Finite elements for the Stokes problem. In *Mixed Finite Elements, Compatibility Conditions, and Applications*, pages 45–100. Springer Berlin Heidelberg, 2008.

[15] F. Bornemann and H. Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numerische Mathematik*, 64(1):455–476, 1993.

[16] D. Braess. The contraction number of a multigrid method for solving the poisson equation. *Numerische Mathematik*, 37(3):387–404, 1981.

[17] D. Braess. *Finite elements*. Cambridge University Press, Cambridge, second edition, 2001. Theory, fast solvers, and applications in solid mechanics, Translated from the 1992 German edition by Larry L. Schumaker.

[18] D. Braess and W. Hackbusch. A new convergence proof for the multigrid method including the V-cycle. *SIAM journal on numerical analysis*, 20(5):967–975, 1983.

[19] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Applied Numerical Mathematics*, 23(1):3–19, feb 1997.

[20] J. Bramble. *Multigrid methods*. Pitman research notes in mathematics series. Longman Scientific & Technical, 1993.

[21] J. Bramble and J. Pasciak. Iterative techniques for time dependent Stokes problems. *Computers Math. Applic.*, 33:13–30, 1997.

[22] J. H. Bramble and J. E. Pasciak. New convergence estimates for multigrid algorithms. *Mathematics of computation*, 49(180):311–329, 1987.

[23] J. H. Bramble and J. E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Mathematics of Computation*, 50(181):1, jan 1988.

[24] J. H. Bramble and J. E. Pasciak. The analysis of smoothers for multigrid algorithms. *Mathematics of Computation*, 58(198):467–488, 1992.

[25] J. H. Bramble and J. E. Pasciak. New estimates for multilevel algorithms including the V-cycle. *Mathematics of computation*, 60(202):447–471, 1993.

[26] J. H. Bramble, J. E. Pasciak, J. P. Wang, and J. Xu. Convergence estimates for multigrid algorithms without regularity assumptions. *Mathematics of Computation*, 57(195):23–45, 1991.

[27] J. H. Bramble, J. E. Pasciak, J. P. Wang, and J. Xu. Convergence estimates for product iterative methods with applications to domain decomposition. *Mathematics of Computation*, 57(195):1–21, 1991.

[28] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 55(191):1–22, Jul. 1990.

[29] J. H. Bramble and J. Xu. Some estimates for a weighted L^2 projection. *Mathematics of Computation*, 56:463–476, 1991.

[30] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[31] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56, jul 1986.

[32] A. Brandt. Rigorous Quantitative Analysis of Multigrid, I. Constant Coefficients Two-Level Cycle with L_2-Norm. *SIAM Journal on Numerical Analysis*, 31(6):1695–1730, 1994.

[33] A. Brandt. Multigrid guide. Technical report, 2011.

[34] A. Brandt and C. W. Cryer. Multigrid algorithms for the solution of linear complementarity problems arising from free boundary problems. *SIAM J. Sci. Statist. Comput.*, 4(4):655–684, 1983.

[35] A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (amg) for automatic multigrid solutions with application to geodetic computations. *Report, Inst. for computational Studies, Fort collins, colo*, 1982.

[36] A. Brandt, S. McCormick, and J. W. Ruge. Algebraic multigrid for sparse matrix equations. In D. Evans, editor, *Sparsity and its Application*, pages 257–284. Cambridge University Press, 1984.

[37] S. Brenner. Convergence of the multigrid V-cycle algorithm for second-order boundary value problems without full elliptic regularity. *Mathematics of Computation*, 71(238):507–525, 2002.

[38] S. C. Brenner and L. R. Scott. *The Mathematical Theory of Finite Element Methods*, volume 15 of *Texts in Applied Mathematics*. Springer-Verlag, New York, second edition, 2002.

[39] M. Brezina. An improved convergence analysis of smoothed aggregation algebraic multigrid. *Numerical Linear Algebra with Applications*, 19:441—-469, 2012.

[40] M. Brezina, A. J. Cleary, R. D. Falgout, V. E. Henson, J. E. Jones, T. A. Manteuffel, S. F. Mccormick, and J. W. Ruge. Algebraic multigrid based on element interpolation (AMGe). *SIAM J. Sci. Comput.*, 22:1570–1592, 2000.

[41] M. Brezina, R. Falgout, S. MacLachlan, T. a. Manteuffel, S. Mccormick, and J. Ruge. Aggregation (αSA) Multigrid. *SIAM Review*, 47(2):317–346, 2005.

[42] F. Brezzi. On the existence, uniqueness and approximation of saddle-point problems arising from lagrangian multipliers. *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 8(R2):129–151, 1974.

[43] F. Brezzi, M. Fortin, and L. D. Marini. Mixed finite element methods with continuous stresses. *Mathematical Models and Methods in applied sciences*, 3(02):275–287, 1993.

[44] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. Siam, 2000.

[45] L. Chen. Deriving the XZ identity from auxiliary space method. In *Domain Decomposition Methods in Science and Engineering XIX*, pages 309–316. Springer, 2011.

[46] P. G. Ciarlet. *The Finite Element Method for Elliptic Problems*, volume 4 of *Studies in Mathematics and its Applications*. North-Holland Publishing Co., Amsterdam-New York-Oxford, 1978.

[47] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Number v. 1 in Methods of Mathematical Physics. Wiley, 1991.

[48] R. a. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51, nov 2008.

[49] M. Dryja and O. Widlund. Additive schwarz methods for elliptic finite element problems in three dimensions. In *Fifth International Conference on Domain Decomposition Methods*. SIAM, 1992.

[50] M. Dryja and O. B. Widlund. Some domain decomposition algorithms for elliptic problems. In *Iterative Methods for Large Linear Systems*. Academic Press Professional, Inc., 1989.

[51] M. Dryja and O. B. Widlund. Additive schwarz methods for elliptic finite element problems in three dimensions. In *Parallel Algorithms for Partial Differential Equations Proceedings, Kiel 1990*. New York University. Courant Institute of Mathematical Sciences., 1991.

[52] H. C. Elman, D. J. Silvester, and A. J. Wathen. *Finite elements and fast iterative solvers: with applications in incompressible fluid dynamics*. Oxford University Press, USA, 2005.

[53] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[54] R. D. Falgout and J. B. Schroder. Non-Galerkin coarse grids for algebraic multigrid. *SIAM J. Sci. Comput.*, 36:C309–C334, 2014.

[55] R. D. Falgout and P. S. Vassilevski. On generalizing the algebraic multigrid framework. *SIAM J. Numer. Anal.*, 42:1669–1693, 2004.

[56] R. D. Falgout, P. S. Vassilevski, and L. T. Zikatanov. On two-grid convergence estimates. *Numerical Linear Algebra with Applications*, 12(5-6):471–494, 2005.

[57] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1(4):1092–1096, 1962.

[58] R. P. Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 4(3):227–235, 1964.

[59] M. Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*, volume 15. Elsevier, 2000.

[60] G. P. Galdi. *An introduction to the mathematical theory of the Navier-Stokes equations: Steady-state problems*. Springer Science & Business Media, 2011.

[61] A. Gibbons. *Algorithmic graph theory*. Cambridge university press, 1985.

[62] R. Glowinski. *Numerical methods for nonlinear variational problems*. Springer-Verlag, New York, 1984.

[63] R. Glowinski, T.-W. Pan, and J. Periaux. A fictitious domain method for dirichlet problem and applications. *Computer Methods in Applied Mechanics and Engineering*, 111(3-4):283–303, 1994.

[64] G. H. Golub and C. F. Van Loan. *Matrix Computations, Third Edition*, volume 10 of *Johns Hopkins Studies in the Mathematical Sciences*. Johns Hopkins University Press, 1996.

[65] M. Griebel and P. Oswald. On the abstract theory of additive and multiplicative Schwarz algorithms. *Numerische Mathematik*, 180:163–180, 1995.

[66] F. Gustavson. Finding the block lower triangular form of a sparse matrix. In *Sparse matrix computations*, pages 275–289. Elsevier, 1976.

[67] W. Hackbusch. Ein iteratives verfahren zur schnellen auflösung elliptischer randwertprobleme. Technical report, 1976.

[68] W. Hackbusch. Multi-grid convergence theory. In *Multigrid methods*, pages 177–219. Springer, 1982.

[69] W. Hackbusch. *Multi-grid methods and applications*. Springer, 1985.

[70] X. He and C. Vuik. Comparison of some preconditioners for the incompressible Navier-Stokes equations. *Numerical Mathematics: Theory, Methods and Applications*, 9(02):239–261, 2016.

[71] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, Cambridge, 1987.

[72] D. Kinderlehrer and G. Stampacchia. *An introduction to variational inequalities and their applications*, volume 88 of *Pure and Applied Mathematics*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], New York, 1980.

[73] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities. I. *Numer. Math.*, 69(2):167–184, 1994.

[74] M. Larin and A. Reusken. A comparative study of efficient iterative solvers for generalized Stokes equations. *Numerical Linear Algebra with Applications*, 15(November 2007):13–34, 2008.

[75] Y. Lee, J. Wu, J. Xu, and L. Zikatanov. Robust subspace correction methods for nearly singular systems. *Mathematical Models and Methods in Applied Sciences*, 17(11):1937–1963, 2007.

[76] S. P. MacLachlan and L. N. Olson. Theoretical bounds for algebraic multigrid performance: review and analysis. *Numerical Linear Algebra with Applications*, 21(2):194–220, 2014.

[77] D. S. Malkus and T. J. Hughes. Mixed finite element methods—reduced and selective integration techniques: a unification of concepts. *Computer Methods in Applied Mechanics and Engineering*, 15(1):63–81, 1978.

[78] J. Mandel. Multigrid convergence for nonsymmetric, indefinite variational problems and one smoothing step. *Appl. Math. Comput.*, 19(1-4):201–216, 1986. Second Copper Mountain conference on multigrid methods (Copper Mountain, Colo., 1985).

[79] T. A. Manteuffel, S. Münzenmaier, J. Ruge, and B. S. Southworth. Nonsymmetric reduction-based algebraic multigrid. *SIAM J. Sci. Comput.*, 41:S242–S268, 2019.

[80] T. A. Manteuffel, J. Ruge, and B. S. Southworth. Nonsymmetric algebraic multigrid based on local approximate ideal restriction (ℓAIR). *SIAM J. Sci. Comput.*, 40:A4105–A4130, 2018.

[81] K.-A. Mardal and R. Winther. Uniform preconditioners for the time dependent stokes problem. *Numerische Mathematik*, 98(2):305–327, 2004.

[82] K.-A. Mardal and R. Winther. Erratum: Uniform preconditioners for the time dependent stokes problem. *Numerische Mathematik*, 103(1):171–172, 2006.

[83] K.-A. Mardal and R. Winther. Preconditioning discretizations of systems of partial differential equations. *Numerical Linear Algebra with Applications*, 18(1):1–40, Jan 2011.

[84] S. F. McCormick. Multigrid methods for variational problems: general theory for the V-cycle. *SIAM Journal on Numerical Analysis*, 22(4):634–643, 1985.

[85] S. F. McCormick. *Multigrid methods*. SIAM, 1987.

[86] A. C. Muresan and Y. Notay. Analysis of aggregation-based multigrid. *SIAM Journal on Scientific Computing*, 30(2):1082–1103, 2008.

[87] A. Napov and Y. Notay. Comparison of bounds for V-cycle multigrid. *Appl. Numer. Math*, 60(3):176–192, 2010.

[88] A. Napov and Y. Notay. When does two-grid optimality carry over to the V-cycle? *Numerical linear algebra with applications*, 17(2-3):273–290, 2010.

[89] A. Napov and Y. Notay. Algebraic analysis of aggregation-based multigrid. *Numerical Linear Algebra with Applications*, 18(3):539–564, 2011.

[90] J. Nečas. Sur une méthode pour résoudre les équations aux dérivées partielles du type elliptique, voisine de la variationnelle. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 16(4):305–326, 1962.

[91] S. Nepomnyaschikh. Decomposition and fictitious domains methods for elliptic boundary value problems. In *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 62–72. Philadelphia: SIAM, 1992.

[92] R. Nicolaides. On the ℓ^2 convergence of an algorithm for solving finite element equations. *Mathematics of Computation*, 31(140):892–906, 1977.

[93] Y. Notay. Convergence analysis of perturbed two-grid and multigrid methods. *SIAM journal on numerical analysis*, 45(3):1035–1044, 2007.

[94] Y. Notay. Algebraic analysis of two-grid methods: The nonsymmetric case. *Numerical Linear Algebra with Applications*, 17(1):73–96, jan 2010.

[95] Y. Notay. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, 37(6):123–146, 2010.

[96] Y. Notay. Algebraic theory of two-grid methods. *Numerical Mathematics: Theory, Methods and Applications*, 8(2):168–198, 2015.

[97] Y. Notay and P. S. Vassilevski. Recursive Krylov-based multigrid cycles. *Numerical Linear Algebra with Applications*, 15(July 2007):473–487, 2008.

[98] P. Oswald. On discrete norm estimates related to multilevel preconditioners in the finite element method. In *Constructive Theory of Functions, Proc. Int. Conf. Varna*, pages 203–214, 1991.

[99] S. Pissanetzky. *Sparse matrix technology.* Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1984.

[100] S. Rippa. Minimal roughness property of the Delaunay triangulation. *Comput. Aided Geom. Design*, 7:489–497, 1990.

[101] C. Rodrigo, F. J. Gaspar, and L. T. Zikatanov. On the validity of the local Fourier analysis. *arXiv:1710.00408*, 2017.

[102] J. W. Ruge and K. Stüben. Algebraic multigrid. *in Multigrid Methods, Frontiers Appl. Math.,* SIAM, Philadelphia, 3:73–130, 1987.

[103] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Number 4. SIAM, second edition, 2003.

[104] K. Stüben. An introduction to algebraic multigrid. In *Multigrid by U. Trottenberg, C. Oosterlee, and A. Schüller*, pages 413–532. 2001.

[105] K. Stüben and U. Trottenberg. Multigrid methods: Fundamental algorithms, model problem analysis and applications. 1982.

[106] D. B. Szyld. The many proofs of an identity on the norm of oblique projections. *Numerical Algorithms*, 42(3-4):309–323, 2006.

[107] X.-C. Tai. Rate of convergence for some constraint decomposition methods for nonlinear variational inequalities. *Numerische Mathematik*, 93:755–786, 2003.

[108] X.-C. Tai and J. Xu. Global and uniform convergence of subspace correction methods for some convex optimization problems. *Mathematics of Computation*, 71(237):105–124, 2002.

[109] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[110] U. Trottenberg, C. W. Oosterlee, and A. Schuller. *Multigrid*. Elsevier, 2000.

[111] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, Sep 1996.

[112] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65:138–158, 1986.

[113] P. Vaněk, M. Brezina, J. Mandel, et al. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, 2001.

[114] P. S. Vassilevski. *Multilevel Block Factorization Preconditioners*. 2008.

[115] F. Wang and J. Xu. A crosswind block iterative method for convection-dominated problems. *SIAM Journal on Scientific Computing*, 21(2):620–645, 1999.

[116] O. B. Widlund. Some schwarz methods for symmetric and nonsymmetric elliptic problems. In *Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations*, number 55, page 19. SIAM Philadelphia, PA, 1992.

[117] R. Wienands and W. Joppich. *Practical Fourier analysis for multigrid methods*. CRC press, 2004.

[118] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34:581–613, 1992.

[119] J. Xu. A new class of iterative methods for nonselfadjoint or indefinite problems. *SIAM journal on numerical analysis*, 29(2):303–319, 1992.

[120] J. Xu. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 56:215–235, 1996.

[121] J. Xu and L. Zikatanov. A monotone finite element scheme for convection-diffusion equations. *Mathematics of Computation*, 68(228):1429–1446, 1999.

[122] J. Xu and L. Zikatanov. The method of alternating projections and the method of subspace corrections in Hilbert space. *Journal of The American Mathematical Society*, 15:573–597, 2002.

[123] J. Xu and L. Zikatanov. Some observations on Babuška and Brezzi theories. *Numerische Mathematik*, 94(1):195–202, mar 2003.

[124] J. Xu and L. Zikatanov. Algebraic multigrid methods. *Acta Numer.*, 26:591–721, 2017.

[125] J. Xu and L. T. Zikatanov. Algebraic multigrid methods. *ArXiv e-prints*, Nov. 2016.

[126] X. Xu. *Algebraic Theory of Multigrid Methods*. PhD thesis, University of Chinese Academy of Sciences, 2019.

[127] X. Xu and C.-S. Zhang. On the ideal interpolation operator in algebraic multigrid methods. *SIAM J. Numer. Anal.*, 56:1693–1710, 2018.

[128] K. Yoshida. *Functional Analysis*. Springer-Verlag, 1971.

[129] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49(4):379–412, 1986.

[130] H. Yserentant. Two preconditioners based on the multi-level splitting of finite element spaces. *Numerische Mathematik*, 58(1):163–184, 1990.

[131] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, 2(1993):285–326, 1993.

[132] X. Zhang. Multilevel schwarz methods. *Numerische Mathematik*, 63(1):521–539, 1992.

[133] L. T. Zikatanov. Two-sided bounds on the convergence rate of two-level methods. *Numer. Linear Algebra Appl.*, 15(5):439–454, 2008.

Index