Multilevel Iterative Methods

Chen-Song Zhang

Version 1.0, November 25, 2024

Abstract

Over the last few decades, significant progress has been made in developing efficient iterative solvers for large-scale linear systems generated from partial differential equations (PDEs). Of particular interest is the class of multilevel iterative solvers/preconditioners, which has gained widespread attention for its practical and theoretical effectiveness. In this lecture note, we focus on the analysis and algorithms of multilevel iterative methods, including geometric and algebraic multigrid methods for discrete problems arising from PDEs. While the primary focus is on the simple Poisson's equation, we also discuss a few more complicated applications of multilevel iterative methods.

This lecture note was originally developed for one semester-long course at the Academy of Mathematics and Systems Science, Beijing. It draws on Prof. Jinchao Xu's short courses at Peking University in 2013 and at the Academy of Mathematics and Systems Science (AMSS) in 2016, as well as Prof. Ludmil Zikatanov's summer school lectures at AMSS in 2015. The discussion on solving singular and nearly-singular problems are based on Prof. Youngju Lee's lectures at AMSS in 2023. Special thanks go to Dr. Xuefeng Xu, Dr. Huilan Zeng, Dr. Wenjuan Liu, Mr. Bin Dai, ChatGPT, and Claude for proofreading 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 — June 20, 2022
- Version 0.9: Dec 30, 2022 — May 18, 2023
- Version 1.0: July 28, 2024 — November 25, 2024

Contents

Part I

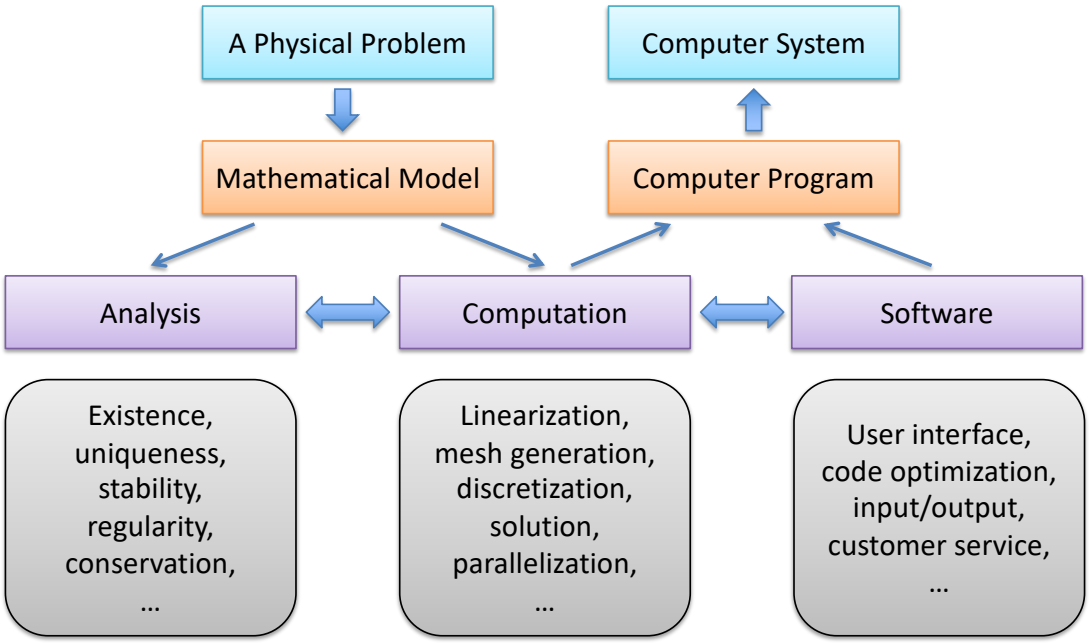
General Theory of Multilevel Iterative Methods

Chapter 1

Introduction

ch:intro

Computer simulations have become an indispensable tool in modern engineering and science. Many complex problems in these fields can be formulated as partial differential equations (PDEs), which require extensive numerical simulations. Solving the discretized systems underlying these equations can be quite costly, occupying a significant portion of the overall computational time. Several fast-solution techniques have been developed to address this issue, such as adaptive mesh refinement (AMR), domain decomposition (DD) methods, and multigrid (MG) methods – all of which involve multilevel ideas.



ig:simulation

Figure 1.1: Numerical simulation of a physical problem.

The above diagram provides a straightforward representation of how physical problems are resolved through numerical simulation, necessitating a comprehensive model, mathematical and

numerical analyses, scientific computing, and software engineering. Successfully simulating complex physical phenomena requires expertise in numerous scientific areas, making it difficult for a single individual to manage all aspects effectively. Therefore, close collaborations among various experts from different areas are crucial in today's era.

Effective linear solvers play a key role in many application areas in scientific computing. Various algorithms for solving linear systems exist, but this lecture concentrates on understanding the algorithmic and theoretical aspects of multilevel iterative techniques - specifically, the 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 design goals of linear solver that we desire in practice, including:

- Applicability — The method can be applied to the problem of interest.
- 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.
- User-friendliness — The method can be implemented or used easily.

Here, our intention is not to provide rigorous definitions of these characteristics as they will be discussed in further detail later on. Sometimes, these features can be conflicting, necessitating a careful balance in practice. Various solution methods are available, ranging from direct solvers to iterative solvers. Throughout this lecture, we will examine several popular multilevel iterative techniques, including overlapping domain decomposition methods with coarse space corrections, two-grid methods, geometric multigrid methods, and algebraic multigrid methods. Our primary focus will be on studying the convergence theories of these methods utilizing 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 solving the Poisson's equation with the Dirichlet boundary condition

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

eqn:Poisson

You will see this equation repeatedly throughout this lecture, as it is the first in a series of model equations. It will serve as the basis for many of the subsequent model equations in this lecture.

Remark 1.1 (Diffusion equation in various applications)**.** Due to their universal applicability, the Poisson's equation, and its extension the diffusion equation, play a fundamental role in physics, chemistry, and engineering, governing phenomena such as Fick's law for chemical concentration, Fourier's law for temperature, Ohm's law for electrostatic potential, and Darcy's law for porous media flow. □

1.1.1 Derivation and classical solution ⋆

The concept of diffusion, a fundamental process underlying the movement of particles, is key to understanding a wide range of phenomena in physics, chemistry, biology, sociology, economics, and finance. Diffusion occurs as a result of the random movement of particles, which leads to the net flow of particles from regions of high concentration (or high chemical potential) to regions of low concentration (or low chemical potential). This is also referred to as the movement of a substance down a concentration gradient. By understanding diffusion, we can better understand how particles move and interact with one another, enabling us to predict and control a broad range of phenomena in numerous fields of study.

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 a 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.,

$$\vec{F} = -\mu(x)\nabla u. \tag{1.2}$$

eqn:flux

where $\mu(x)$ is referred to as the diffusivity coefficient (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

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

This can be easily seen by applying the Divergence Theorem, i.e.

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

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

$$\frac{\partial}{\partial t}u = \nabla \cdot (\mu \nabla u). \tag{1.5}$$ eqn:heat1

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

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

The steady-state solution of equation (1.6) satisfies the well-known Poisson's equation

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

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

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

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

We have the *fundamental solution* of the Laplace equation

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

where γ_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 the classical textbook by Evans [88].

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

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

The first identity is usually referred to as the maximum principle and, on the other hand, the second is the so-called strong maximum principle.

Using the maximum principle, we can immediately 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 $u \in C^2(\Omega) \bigcap C(\overline{\Omega})$.

1.1.2 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

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

where $\alpha = [\alpha_1, \ldots, \alpha_d]$ is a multi-index, $\alpha_1, \ldots, \alpha_d \in \mathbb{N}$, and $|\alpha| := \alpha_1 + \cdots + \alpha_d$.

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

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

and, for $p = \infty$,

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

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

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

rem:scaling **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

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

This property is particularly useful in the scaling argument (or sometimes referred to as the homogeneity argument) for finite element error estimates. □

If $p = 2$, the spaces $W_2^k(\Omega)$ are Hilbert spaces and we denote them as $H^k(\Omega)$ for convenience. 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 $W_2^k(\Omega)$-norm. We denote the completion of $C_0^\infty(\Omega)$ in $H^k(\Omega)$ by $H_0^k(\Omega)$.

In some occasions, we will also use the fractional Sobolev space $H_0^{k+\sigma}(\Omega)$ where $0 < \sigma < 1$. It is defined as the completion of $C_0^\infty(\Omega)$ in the fraction norm:

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

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.

rop:embedding **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.

op:embeddingC **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 $W_p^m(\Omega) \subset C^{0,\mu}(\Omega) \subset C^0(\Omega)$.

A function f is in $C^{0,\mu}(\Omega)$ if and only if there exists a nonnegative constants C, such that

$$|f(x) - f(y)| \leqslant C\|x - y\|^\mu$$

for all x and y in Ω.

eg:embedding **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 find that

$$H^1(\Omega) \subset C^0(\Omega), \text{ if } d = 1; \text{ and } H^1(\Omega) \not\subset C^0(\Omega), \text{ if } d > 1.$$

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 $H^1(\Omega)$.

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

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

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

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

The above inequality 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.

prop:trace **Proposition 1.12** (Trace theorem)**.** There exists a unique linear operator

$$\text{trace} : H^1(\Omega) \mapsto L^2(\partial\Omega),$$

such that $\text{trace}(v) = v$ on $\partial\Omega$, if $v \in C^0(\overline{\Omega}) \bigcap H^1(\Omega)$, and

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

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

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

1.1.3 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 integral form. In the one-dimensional case, it is easy to see that

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

Now the question is how we can formulate the integral formulation in a "weaker" sense, namely the weak form—We apply integration by parts.

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

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

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

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

g:WeakPoisson **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 $\mathscr{U} = \mathscr{V} = H_0^1(\Omega)$. □

1.1.4 Well-posedness of the weak problem ⋆

ssc:bt

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 [148].

:banach_necas **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:

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

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

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

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

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

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

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). □

thm:necas **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

$$\sup_{v\in\mathscr{V}} \frac{a[w,v]}{\|v\|_{\mathscr{V}}} \geqslant \alpha\|w\|_{\mathscr{U}}, \quad \forall\, w \in \mathscr{U}; \tag{1.20}$$ eqn:inf_sup_1

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

$$\inf_{w\in\mathscr{U}} \sup_{v\in\mathscr{V}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} > 0 \qquad \text{and} \qquad \inf_{v\in\mathscr{V}} \sup_{w\in\mathscr{U}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} > 0. \tag{1.21}$$ eqn:inf_sup_2

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

$$\inf_{w\in\mathscr{U}} \sup_{v\in\mathscr{V}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} = \inf_{v\in\mathscr{V}} \sup_{w\in\mathscr{U}} \frac{a[w,v]}{\|w\|_{\mathscr{U}}\|v\|_{\mathscr{V}}} = \alpha. \tag{1.22}$$ eqn:inf_sup_3

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,

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

where $\mathcal{A}^T$ 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)**.** Suppose that $a[\cdot,\cdot] : \mathscr{U} \times \mathscr{V} \mapsto \mathbb{R}$ is a continuous bilinear form. 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 any of the above 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 $\alpha = \|\mathcal{A}^{-1}\|^{-1}_{\mathscr{L}(\mathscr{V};\mathscr{U})} \geqslant C^{-1}$. □

1.1.5 A simple model problem

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

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

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

The coercivity implies 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 is 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 $\|u\|_{\mathscr{V}} \leqslant \alpha^{-1}\|f\|_{\mathscr{V}'}$.

rem:energynorm **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

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

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

rem:well-posed **Remark 1.23** (Condition of Poisson's equation)**.** 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

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

We define the "condition number" of the Laplace operator

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

which is bounded. □

Despite the observation (Remark 1.23) that $-\Delta$ is "well-conditioned", our experience in numerical solution of the Poisson's equation may result in ill-conditioned problems, contradicting it. In practice, numerous challenges can arise while utilizing numerical methods to solve the equation, including the selection of appropriate numerical algorithms. The problem arises due to the fact that we are working with the mapping from $\mathscr{V}$ to $\mathscr{V}'$. In numerical methods, we usually consider

$$-\Delta : L^2(\Omega) \mapsto L^2(\Omega)$$

instead, then we lost boundedness and the condition number will blow up. More general theory has been developed in the seminar work by Babuška [9].

1.2 Discretization methods

scretizations

The process of discretization involves transforming continuous functions, models or equations into their discrete equivalents. Such a conversion is typically carried out as a preliminary step towards achieving computational efficiency, allowing for numerical assessment, and enabling implementation on modern computer systems.

Let $\Omega \subset \mathbb{R}^d$ 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

$$A\vec{u} = \vec{f}. \tag{1.24}$$ eqn:linear0

1.2.1 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.

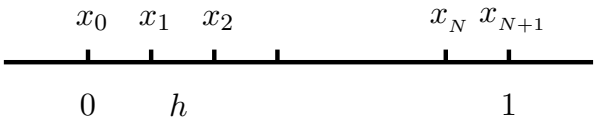


Figure 1.2: Uniform mesh in 1D.

fig:mesh1d

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

$$\frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{N-1} \\ u_N \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) \end{pmatrix}. \tag{1.25}$$ eqn:1DFDA

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 1.24 (An alternative form of the linear system)**.** Sometimes, it is 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. □

LaplaceEvalue **Remark 1.25** (Eigenvalues of Laplace operator)**.** Let us now consider the eigenvalue problem for the one-dimensional Laplace operator with homogeneous Dirichlet boundary conditions, i.e., $-u''(x) = \lambda u(x)$ for $x \in (0, 1)$ and $u(0) = u(1) = 0$. It is well-known that the eigenvalues and the corresponding eigenfunctions of the above equation are

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

We notice that large eigenvalues (larger k) correspond to eigenfunctions of highe frequency. □

Similar results can be expected for discrete problems. In fact, if A is symmetric, then all eigenvalues of A are real and there exists an orthonormal basis of $\mathbb{R}^N$ consisting of eigenvectors.

rem:FDevalue **Remark 1.26** (Eigenvalues of 1D FD problem)**.** For simplicity, we now assume $h \equiv 1$; if it is not the case, we can scale the original problem. It is well-known (see HW 1.3) that the eigenvalues of $A := \text{tridiag}(-1, 2, -1)$ are

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

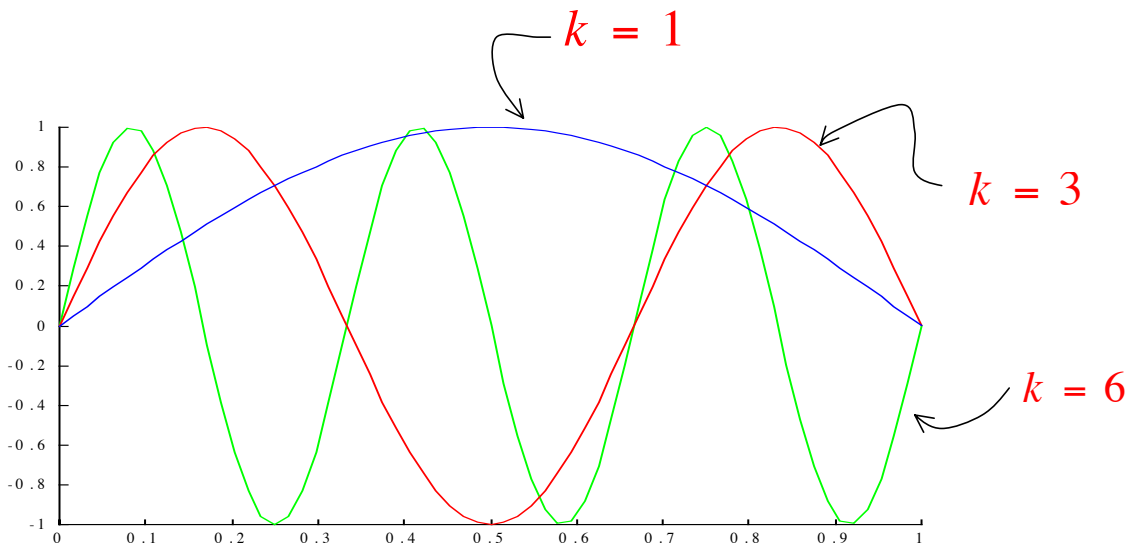
and the corresponding eigenvectors are

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

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. □



ig:eigenfct1D

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

From Figure 1.3, we can 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.

In Remark 1.26, we immediately notice that λ_N is close to 4 but λ_1 is close to 0. That is to say, the discrete operator is ill-conditioned for large N.

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

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

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

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

rem:FDevalue2D **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

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

□

rem:ill **Remark 1.28** (Discrete Poisson's equation is ill-conditioned)**.** Remark 1.23 demonstrates that Poisson's equation has a bounded condition number. Nonetheless, in the event that the mesh size h used in the finite difference (FD) discrete problems is small, the coefficient matrices A turn out to be ill-conditioned; see Remarks 1.26 and 1.27. Consequently, this can lead to issues with several iterative methods where the convergence rates rely on the spectrum of A. Similar behavior also happens in the finite element (FE) setting; see Remark 1.31. □

Remark 1.29 (Ordering)**.** The structure of coefficient matrix A depends on how the degrees of freedom (DOFs) are arranged. The ordering of DOFs not only influences the efficiency of smoothers and parallelization techniques but also impacts their smoothing properties. In certain linear solvers, such as LU factorization methods, determining the minimum bandwidth ordering is imperative. However, this process is NP-hard, making it particularly challenging to achieve. □

Remark 1.30 (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

$$A\vec{u} = \vec{f}, \tag{1.26}$$ eqn:FEM1D

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

rem:ill-FEM **Remark 1.31** (Eigenvalues of finite element system)**.** If we use the uniform mesh in Figure 1.2, then we have (see HW 1.4) that

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

This method yields similar algebraic system as in (1.25) but with different scaling. Upon solving this finite-dimensional problem, we are able to 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. □

1.2.2 High-frequency and locality

Eigenvalues and eigenfunctions, which are fundamental concepts in mathematical analysis, are typically associated with the fundamental modes of vibration or oscillation. As the frequency increases, the amplitude of the oscillation decreases asymptotically. Additionally, the smoother the function, the faster the decay rate. Weyl's law demonstrates that the asymptotic behavior of the eigenvalues in relation to the domain is highly dependent on it.

An important observation comes from the analysis to the local problem

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

Using Remark 1.25, we can obtain the eigenfunctions of this localized problem:

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

Suppose that u is the solution to the corresponding Poisson's equation on the whole domain Ω. Define the error $e := u - u_\delta$ in B_δ (difference between the solutions to the Poisson problem on the original domain and the localized ball). Apparently, e is harmonic in B_δ.

A commonly-used trick is to construct a cut-off function $\theta \in C_0^\infty(B_\delta)$, such that it satisfies the following conditions:

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

Then we have

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

The first and last inequalities immediately imply that

$$\Big(\int_{B_{\delta/2}} |e'(x)|^2\,dx\Big)^{\frac{1}{2}} \leqslant \Big(\int_{B_\delta} \theta^2(x)|e'(x)|^2\,dx\Big)^{\frac{1}{2}} \leqslant \frac{2C}{\delta}\Big(\int_{B_\delta} |e|^2\,dx\Big)^{\frac{1}{2}}. \tag{1.27}$$ eqn:Harnack

You may compare the above inequality (1.27) with the Poincaré inequality given in Proposition 1.11.

Now we plug in the eigenfunctions $u_{\delta,k}$ $(k = 1, 2, \ldots)$ to the above inequality. We observe that only some of the error components with

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

can be allowed. This suggests that the error function e primarily consists of *low-frequency* elements, with the oscillating components in the distance δ captured precisely.

em:GeoHighFreq **Remark 1.32** (Geometric high-frequencies)**.** This simple observation suggests that the local solution u_δ for the model problems provides an accurate approximation of the high-frequency part of u. Inspired by this idea and building on (1.27), we can define *geometric high-frequency* functions u_k as those with a high ratio of $\|\nabla u_k\|_{0,\Omega}$ to $\|u_k\|_{0,\Omega}$. It is important to note that singularities, which are a special form of high-frequency behavior, can also be resolved using local mesh refinement techniques. This method is effective precisely because high-frequency behavior is typically localized, and can be approximated accurately using finer meshes in the relevant areas. In subsequent chapters, we will delve into this issue in more detail, exploring its geometric and algebraic implications. □

1.2.3 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, for $x \in (x_{i-1}, x_i)$, that

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

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 (rough, nonsmooth, or singular)? We now assume that u is in $W^1_1(0,1)$. In view of the first inequality 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

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

m:EqDistError **Remark 1.33** (Equidistribution of error)**.** When solving differential equations, evenly distributed mesh spacing across the computational domain may not be the most effective method if the solution is not smooth. In such cases, it's better to pursue a mesh that distributes error evenly. This adaptive mesh approach is dependent on the solution itself and can dynamically adjust based on its changes to optimize the accuracy. This idea is called *equidistribution of error*, where mesh points are concentrated in areas with large errors and spread out in regions of lesser error. Developing an equidistributed error mesh is challenging and requires a nonlinear approximation procedure, which involves solving the differential equation numerically and then using the solution output to improve the mesh. This process of refinement continues until the error is uniformly distributed across the mesh. □

For more comprehensive information on this subject, refer to the work of DeVore [76], which explores equidistributed error meshes within the context of finite element methods. The paper provides a broader framework for establishing equidistributed error meshes and suggests multiple examples that demonstrate their potential effectiveness. In conclusion, equidistribution of error

presents a favorable alternative to the typical mesh spacing approach when dealing with problems where the solution is not smooth. Though obtaining this mesh requires greater computational efforts, its benefits are substantial enough to warrant further investigation by researchers and analysts seeking to improve differential equation solving.

1.3 Simple iterative solvers

sec:simple

Numerous methods exist for tackling the linear algebraic equations derived from finite difference, finite element, and other discretizations applied to Poisson's equation, including sparse direct solvers, fast Fourier transform techniques, and iterative solutions. However, for the purposes of this lecture, we will solely focus on discussing iterative solvers.

1.3.1 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:R **Example 1.34** (Richardson method)**.** The simplest iterative method for solving $A\vec{u} = \vec{f}$ might be the Richardson method

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

eqn:richardso

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

example:J **Example 1.35** (Weighted Jacobi method)**.** The weighted or damped Jacobi method can be written as

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

eqn:Jacobi

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:GS **Example 1.36** (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

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

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

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

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 fundamental iterative methods have been addressed in common textbooks on numerical analysis and can be formulated via the classical splitting approach. In the following, we utilize a modified version to achieve a better perspective.

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

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

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

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

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 $\rho(E) < 1$.

1.3.2 An observation on smoothing effect

Numerous basic iterative methods display diverse convergence rates for short and long wavelength error components, which suggests that these distinct scales require varying treatment. To delve deeper into this phenomena, we consider $\lambda_{\max}$ and $\lambda_{\min}$, which stand for the largest and smallest eigenvalues of A, and $\vec{\xi}^{\max}$ and $\vec{\xi}^{\min}$, their corresponding eigenvectors. One intriguing observation made by several experts is that when we employ the weighted Jacobi method (1.30) with a weight ω of $2/3$ to solve the equation $A\vec{u} = \vec{0}$, initialized with $\vec{\xi}^{\max}$, the convergence occurs rapidly. Conversely, if the same equation is solved using the weighted Jacobi iteration, but with a distinct initial guess of $\vec{\xi}^{\min}$, the convergence rate is much slower. See Figure 1.4 for a demonstration.

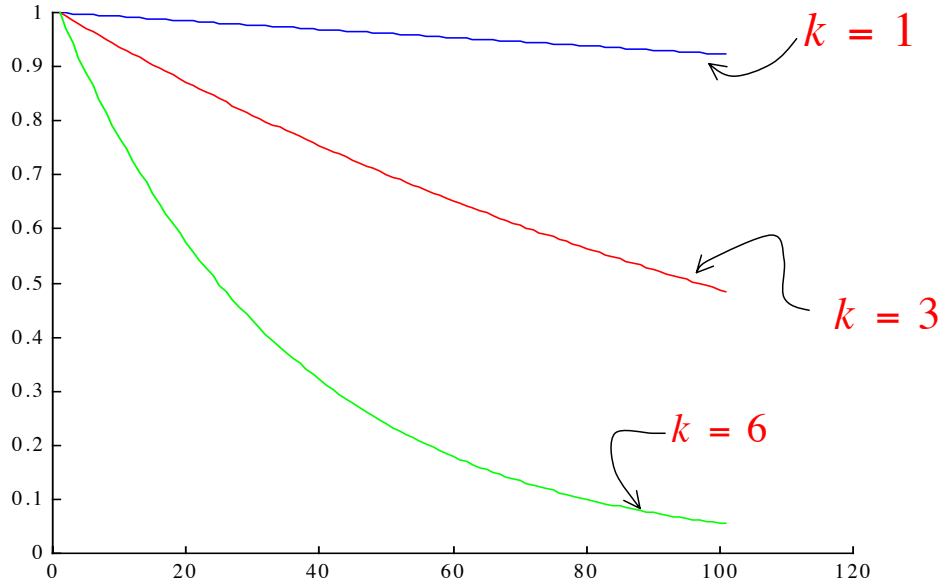


fig:errorJ

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

It's worth noting that the difference in results between the first problem (with $\vec{\xi}^{\max}$) and the second problem (with $\vec{\xi}^{\min}$) is mainly due to the nature of the errors involved. Specifically, the error in the first problem is oscillatory and high-frequency, while that in the second problem is smooth and low-frequency. This observation leads one to speculate that the weighted Jacobi method might be effective at quickly dampening the high-frequency component of the error but more slowly addressing the low-frequency component. For more details, see Remark 1.32.

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

1D are

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

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

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

By examining the above equation, it becomes apparent that the eigenvalues $|\lambda_k(E)|$ are less than or equal to $\frac{1}{3}$ for larger values of k ($\frac{N}{2} \leqslant k \leqslant N$), implying that the weighted Jacobi method is likely to converge quickly for these larger values of 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

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

We can observe from (1.35) that the rate of convergence is fast for high-frequency error components, indicated by larger values of k, and sluggish for low-frequency components, represented by smaller values of k.

1.3.3 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 weighted 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 that

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

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 also apply the same analysis to the Jacobi method as well and the Fourier coefficient in front of the highest frequency becomes:

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

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

sec:GMG1D

In this section, we present a straightforward motive and a preview of the widely used multigrid method, an example of multilevel iterative methods. The insights obtained herein will be advantageous for our subsequent discussions. For a brief introduction to the multigrid methods, the tutorial by Briggs et al. [65] is a well-known resource.

Consider solving the linear system arising from the *finite difference* scheme (1.25) for the Poisson's equation in 1D, namely

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

1.4.1 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.

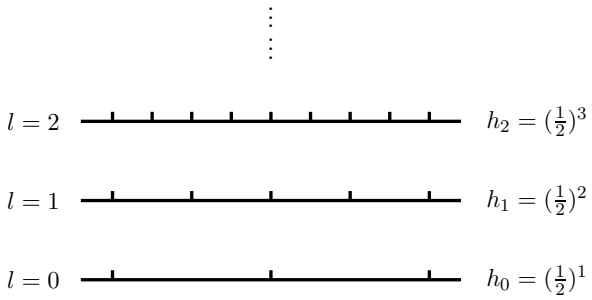


fig:hiergrid

Figure 1.5: Hierarchical grids for 1D multigrid method.

1.4.2 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

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

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

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

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.

1.4.3 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.,

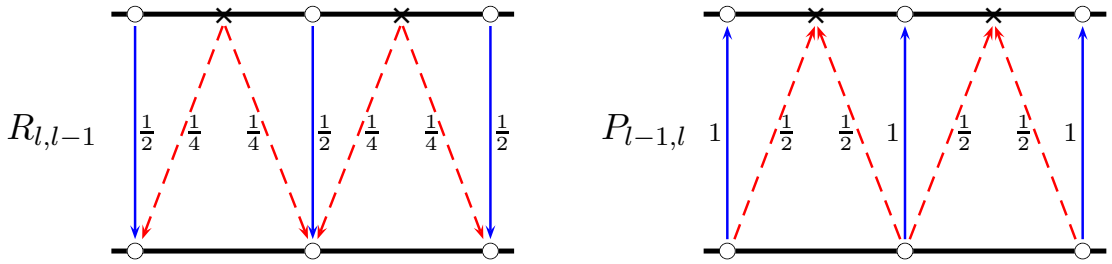


Figure 1.6: Transfer operators between two consecutive levels (Left: restriction operator; right: prolongation operator).

ig:transfer1d

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

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

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

1.4.4 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).

alg:1DGMG **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 1.38 (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

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

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 1.39 (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. □

#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

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

Algorithm 1.1 represents one iteration of the multigrid method, and we can use it to iterate until our approximation is deemed "satisfactory". We often use the relative residual $|\vec{r}|_0/|\vec{f}|_0$ as a stopping criterion, typically set to be less than 10^{-6}. This algorithm can be easily implemented (see HW 1.6). Table 1.1 shows the numerical results of Algorithm 1.1 applied to the 1D Poisson's

equation with three G-S iterations as smoother. Unlike classical Jacobi and G-S methods, this multigrid method converges uniformly with respect to h. This is a highly desirable feature of multilevel iterative methods, which we will explore further in this lecture.

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 presented in this lecture were computed using the Fast Auxiliary Space Preconditioning (FASP) package. This package offers C source files that can be used to build a library of efficient iterative solvers and preconditioners for solving large-scale linear systems of equations. The FASP basic library (i.e. faspsolver) consists of various modern and effective iterative solvers that are commonly used in a wide range of applications, from simple examples of discretized scalar partial differential equations (PDEs) to complex numerical simulations of multicomponent physical systems. The FASP package provides C99 (ISO/IEC 9899:1999)-compatible code, making it highly versatile and accessible to users across multiple platforms.

The main components of the FASP basic library are:

- Basic linear iterative methods;
- Standard Krylov subspace methods;
- Geometric and algebraic multigrid methods;
- Incomplete factorization methods.

The FASP distribution also includes several examples for solving simple benchmark problems.

The fundamental (kernel) distribution of FASP is open-source and is licensed under the GNU Lesser General Public License or LGPL. However, other distributions of the software may have different licensing arrangements (for more information, you can reach out to the development team). The latest version of FASP can be directly downloaded from:

https://github.com/FaspDevTeam/faspsolver

To compile, you need a C99 compiler (and a F90 compiler if you need to compile Fortran examples). By default, we use GNU gcc/gfortan, respectively. Configuring and building the FASP library and test suite requires CMake 2.8.12 or higher http://www.cmake.org/.

The command to configure is:

```
> mkdir Build; cd Build; cmake ..
```

After successfully configing the environment, just run:

```
> make  # to compile the FASP lib only; do not install
```

To install the FASP library and executables, run:

```
> make install  # to compile and install the FASP lib
```

Note: The default prefix is the FASP source directory.

For further details on the usage and implementation of FASP, we recommend consulting the user's guide and reference manual of the package[1]. However, keep in mind that FASP is undergoing heavy development, and the code may have changed since the guide's last update. As such, we advise users to exercise caution when employing this guide.

1.6 Homework problems

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

:Eigenvectors **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:Evalue1DFD **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 $\text{tridiag}(b, a, c) \in \mathbb{R}^{N\times N}$?

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

[1] The guide is available online at http://www.multigrid.org/fasp and can also be found in the "faspsolver/-doc/" directory.

hw:FDFEHeat **HW 1.5.** Derive 1D FD and FE discretizations for the heat equation (1.6) using the backward Euler method for time discretization.

hw:GMGcode **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.

w:JacobiEigen **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.

Chapter 2

Iterative Solvers and Preconditioners

ch:iterative

The term iterative method encompasses various numerical techniques that involve successive approximations, denoted by $\{u^{(m)}\}$, to find the exact solution u to a given problem. This chapter examines two types of iterative methods: (1) stationary, which involves the same operations for each iteration, and (2) nonstationary, which includes iteration-dependent operations. Stationary

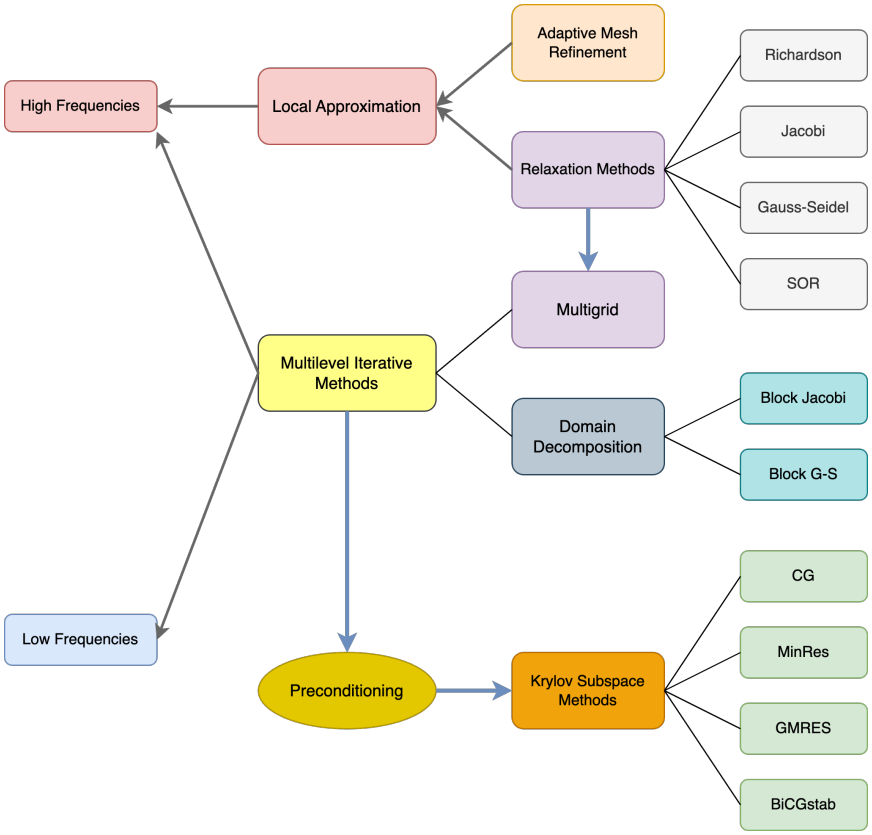


Figure 2.1: Iterative and preconditioning methods discussed in this note.

fig:Chapter2

methods are generally easy to understand and implement, but they tend to be less effective than

nonstationary methods. Conversely, nonstationary methods are a more recent development and tend to have more complex analyses associated with them. In Figure 2.1, we give a bird's eye view of the methods that will be discussed.

2.1 Stationary linear iterative methods

sec:stationary

This section focuses on stationary iterative methods, two typical examples being the Jacobi and Gauss-Seidel methods. We will explore why these methods are generally considered inefficient, despite still being 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

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

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

$$A\vec{u} = \vec{f}. \tag{2.2}$$ eqn:linear2

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

Remark 2.1 (More general setting)**.** In fact, iterative methods can be approached in a broader setting. For instance, consider a finite-dimensional Hilbert space, V, its dual, V', and a linear operator $\mathcal{A} : V \mapsto V'$, as well as $f \in V'$. It's worth noting that a significant portion of this lecture can be easily extended to such a context. □

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

alg:iter **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

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

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

2.1.1 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 [212] for details. □

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

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

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

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

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

m:nonsingular **Remark 2.3** (Non-singularity)**.** If $\mathcal{N}(\mathcal{A}) = \{0\}$, then $\mathcal{A}$ is *injective* or one-to-one. Apparently, $\mathcal{A} : V \mapsto \mathcal{R}(\mathcal{A})$ is *surjective* or onto. If we consider a symmetric operator $\mathcal{A} : \mathcal{N}(\mathcal{A})^\perp \mapsto \mathcal{R}(\mathcal{A})$, 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 linear operator 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,

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

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

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 (have been used in Chapter 1):

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

And all these definitions are consistent for symmetric positive definite problems, where the condition number does not depend on the choice of norm.

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

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

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

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

Symmetry is a concept that is relative to the inner product that underlies it. In this chapter, we consistently refer to the $(\cdot,\cdot)$-inner product when discussing symmetry. It is worth noting that, by definition, $(\mathcal{B}\mathcal{A})^* = \mathcal{B}^T\mathcal{A}$; see HW 2.2 for verification of this equation.

If $\mathcal{B}^T = \mathcal{B}$, it does not necessarily follow that $(\mathcal{B}\mathcal{A})^T = \mathcal{B}\mathcal{A}$. However, there is a key identity that holds:

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

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.,

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

□

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

prop:rho **Proposition 2.5** (Spectral radius and norm)**.** Suppose V is a 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:

lem:sym **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 $\rho(\mathcal{B}) = \|\mathcal{B}\|_{\mathcal{A}}$.

2.1.2 Convergence of stationary iterative methods

Next, we examine the convergence analysis of the stationary iterative method (2.3). A method is only considered to be convergent if $u^{(m)}$ converges to u for every initial guess $u^{(0)}$. It is worth emphasizing that each iteration (2.3) depends only on the previous approximation u^{old}, without utilizing any data from previous iterations. Essentially, each iteration performs the same operations repeatedly.

It is easy to see that

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

where $\mathcal{I}: V \mapsto V$ is the identity operator and the operator $\mathcal{E} := \mathcal{I} - \mathcal{B}\mathcal{A}$ is called the *error propagation operator* (or, sometimes, error reduction operator)[2]. Hence we can get the following simple convergence theorem.

thm:rho **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.

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

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

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

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

$$\rho(\mathcal{I} - \mathcal{BA}) = \max\Big(\lambda_{\max}(\mathcal{BA}) - 1,\, 1 - \lambda_{\min}(\mathcal{BA})\Big). \tag{2.10}$$ eqn:rho-condn

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

Although the convergence result is straightforward, applying it can be challenging. Furthermore, it doesn't provide any direct insight into how quickly the algorithm converges, assuming it does at all. The subsequent example will further elaborate on this point.

Remark 2.8 (Asymptotic convergence behavior)**.** An iterative method converges for any initial guess if and only if the spectral radius of the iteration matrix $\rho(E)$ is less than one. However, it is crucial to understand that the spectral radius of E only determines 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. However, it is not clear how quickly the error diminishes during the initial stages of iteration. For instance, consider the following example where it takes all iterations up to the last one for the error to finally decrease, highlighting the uncertainty regarding how swiftly the error diminishes in the early stages of the iteration process. □

Example 2.9 (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.

chardsonSpeed

Remark 2.10 (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 1.34. 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 could be very slow if A is ill-conditioned. □

2.1.3 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:

alg:symiter

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

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

eqn:Bsym

lem:decay **Lemma 2.11** (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 2.12 (Effects 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 2.11 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,

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

Furthermore, we have obtained the following identity:

$$\rho(\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}) = \|\mathcal{I} - \overline{\mathcal{B}}\mathcal{A}\|_{\mathcal{A}} = \sup_{v \in V\setminus\{0\}} \frac{\big((\mathcal{I} - \overline{\mathcal{B}}\mathcal{A})v, v\big)_{\mathcal{A}}}{\|v\|_{\mathcal{A}}^2}. \tag{2.15}$$ eqn:rate1

The above two equations immediately give

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

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 2.14). □

We can also easily obtain a contraction property of the symmetrized iteration. In Lemma 2.11, we have already seen that

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

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

thm:symconv **Theorem 2.13** (Convergence of Symmetrized Algorithm)**.** The symmetrized iteration in 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. □

Ex:ConvBbar **Example 2.14** (Convergence of $\mathcal{B}$ and $\overline{\mathcal{B}}$ iterators)**.** Note that even if $\overline{\mathcal{B}}$ is not SPD, i.e., $\overline{\mathcal{B}}$ does not give a convergent method, the method defined by $\mathcal{B}$ could still converge. For example, 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. □

We notice that

$$(\overline{\mathcal{B}}\mathcal{A}v, v)_{\mathcal{A}} = \big((\mathcal{B} + \mathcal{B}^T - \mathcal{B}^T\mathcal{A}\mathcal{B})\mathcal{A}v, v\big)_{\mathcal{A}} = 2(\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}} - (\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}}, \quad \forall v \in V.$$

In fact, it is easy to check that the above convergence condition is equivalent to that there exists a positive constant α such that

$$2(\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}} - (\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \geqslant \alpha(v, v)_{\mathcal{A}}, \quad v \in V. \tag{2.16}$$ eqn:Bcond2

cor:SPDConv1 **Corollary 2.15** (Convergence conditions)**.** Suppose the following two assumptions hold:

(A1) There exists $\omega \in (0, 2)$ such that $(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \leqslant \omega(\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}}, \quad \forall v \in V$;

(A2) There exists $\beta > 0$ such that $(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \geqslant \beta(v, v)_{\mathcal{A}}, \quad \forall v \in V$.

Then the iterative method defined by $\mathcal{B}$ converges.

Proof. From assumptions (A1) and (A2), we have

$$2(\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}} - (\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \geqslant \Big(\frac{2}{\omega} - 1\Big)(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \geqslant \frac{\beta(2-\omega)}{\omega}(v, v)_{\mathcal{A}}, \quad \forall v \in V.$$

Then (2.16) holds with $\alpha = \beta(2-\omega)/\omega$.

On the other hand, assume that (2.16) hold. For any $v \in V$, using boundedness of the operators, we have

$$2(\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}} - (\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \geqslant \alpha(v,v)_{\mathcal{A}} \geqslant \frac{\alpha}{C}(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}},$$

which gives (A1) with $\omega = \frac{2}{1+\alpha/C}$. Furthezrmore, since $\mathcal{A}$ is positive semidefinite,

$$(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}}^{1/2}(v,v)_{\mathcal{A}}^{1/2} \geqslant (\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}} \geqslant \frac{\alpha}{2}(v,v)_{\mathcal{A}},$$

which give (A2) with $\beta = \alpha^2/4$. □

2.1.4 Convergence rate of stationary iterative methods

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 2.11 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 known as the contraction factor of the iterative method.

From now on, we will consider only symmetric positive definite (SPD) iterates $\mathcal{B}$. Nevertheless, even for non-symmetric iterates, one could examine their symmetrized counterpart.

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

thm:rate **Theorem 2.16** (Convergence rate of linear stationary method)**.** 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

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

Consequently, the second equality of the theorem is proved. □

ex:dJacobi **Example 2.17** (Convergence of 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 weighted 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 arising from discretizing the Poisson's equation, we should use a *damping* factor $0 < \omega < 1$. □

2.1.5 Generalized GS method ⋆

Similar to the weighted Jacobi method (see Example 2.17), we define the weighted GS 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 GS method converges if $0 < \omega < 2$. In fact, $\omega = 1$ yields the standard GS 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 GS methods, we assume that there is an invertible smoother or a local relaxation method S for the equation $A\vec{u} = \vec{f}$, like the weighted Jacobi smoother $S = \omega D^{-1}$ $(0 < \omega < 1)$. We can define a generalized or modified GS method:

$$B := \left(S^{-1} + L\right)^{-1}. \tag{2.17}$$ eqn:MGS

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

We can analyze the convergence rate of this generalized GS method. Since $K = B^{-T} + B^{-1} - A$ is a symmetric matrix and we can write (2.13) as $\overline{B} = B^T K B$. If B is the iteration matrix defined by (2.17), 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:

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

Now we apply Theorem 2.16 and get the following identity for the convergence rate:

thm:rateGS **Corollary 2.18** (Convergence rate of generalized GS)**.** If $K = S^{-T} + S^{-1} - D$ is SPD, then the generalized GS 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 fundamental result will serve as the foundation for our subsequent analysis of subspace correction methods.

ex:FDGS **Example 2.19** (Solving 1D Poisson's equation using GS)**.** We apply the GS method to the 1D FD/FE system (1.25) for the Poisson's equation. 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 generalized GS method. Corollary 2.18 shows that the convergence rate of the GS 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 the eigenvalues of this discrete coefficient matrix A of FD are known (see Remark 1.26), we can estimate the denominator by

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

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. □

2.1.6 Solving semidefinite problems ⋆

:semidefinite

In this section, we analyze the linear stationary iterative method for the symmetric and positive semidefinite problems; for details, we refer to [195]. Let $\mathcal{A} : V \mapsto V$ be a symmetric and positive semidefinite operator and V be a finite dimensional Hilbert space with inner product $(\cdot,\cdot)$. Let $\mathcal{Q} : V \mapsto \mathcal{R}(\mathcal{A})$ be the orthogonal projection under the inner product $(\cdot,\cdot)$. Let $|v|_{\mathcal{A}} := (v,v)_{\mathcal{A}}^{1/2}$ for any $v \in V$. Consider the general iterative method

$$u^{\text{new}} = u^{\text{old}} + \mathcal{B}(f - \mathcal{A}u^{\text{old}}), \tag{2.18}$$

eqn:SingularB

where $\mathcal{B}$ is a linear operator from V to V and it might be singular. The Moore–Penros inverse is denoted by $\mathcal{B}^\dagger$. In case $\mathcal{N}(\mathcal{B}) = \mathcal{N}(\mathcal{A})$ and $\mathcal{R}(\mathcal{B}) = \mathcal{R}(\mathcal{A})$, then $\mathcal{B}^\dagger : V \mapsto V$ is a zero extension of $\mathcal{B}^{-1} : \mathcal{R}(\mathcal{A}) \mapsto \mathcal{R}(\mathcal{A})$, i.e.

$$\begin{aligned}
&\mathcal{B}^\dagger c = 0, \quad \forall c \in \mathcal{N}(\mathcal{A}),\\
&\mathcal{B}^\dagger v = B^{-1} v, \quad \forall v \in \mathcal{R}(\mathcal{A}).
\end{aligned}$$

Similar to Lemma 2.11, we have

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

for any $v \in V$. Following the same lines as Theorem 2.13 and Corollary 2.15, we have the following convergence results.

:SingularConv

Theorem 2.20 (Convergence for SPSD problems)**.** The iteration (2.18) is convergent if and only if $\mathcal{Q}\overline{\mathcal{B}}\,\mathcal{Q}$ is SPD on $\mathcal{R}(\mathcal{A})$.

SingularConv1

Corollary 2.21 (Convergence conditions for SPSD problems)**.** Suppose the following two assumptions hold:

(A1) There exists $\omega \in (0,2)$ such that $(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \leqslant \omega(\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}}, \quad \forall v \in V$;

(A2) There exists $\beta > 0$ such that $(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \geqslant \beta(v,v)_{\mathcal{A}}, \quad \forall v \in V$.

Then the iterative method (2.18) converges.

Under the assumptions (A1) and (A2), we find that

$$\begin{aligned}
|v|_{\mathcal{A}}^2 - |(\mathcal{I} - \mathcal{B}\mathcal{A})v|_{\mathcal{A}}^2 &= 2(\mathcal{B}\mathcal{A}v, v)_{\mathcal{A}} - (\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}}\\
&\geqslant \Big(\frac{2}{\omega} - 1\Big)(\mathcal{B}\mathcal{A}v, \mathcal{B}\mathcal{A}v)_{\mathcal{A}} \geqslant \frac{\beta(2-\omega)}{\omega}|v|_{\mathcal{A}}^2.
\end{aligned}$$

This immediately implies that

$$|\mathcal{I}-\mathcal{B}\mathcal{A}|_{\mathcal{A}}^2 \leqslant 1-\frac{\beta(2-\omega)}{\omega}<1.$$

In fact, we can obtain the following general characterization of the convergence rate for the semidefinite problems.

:SingularRate **Theorem 2.22** (Convergence rate for Semidefinite Problems)**.** Under the assumptions (A1) and (A2), the iteration method (2.18) satisfies that

$$|\mathcal{I}-\mathcal{B}\mathcal{A}|_{\mathcal{A}}^2 = 1-\frac{1}{c_1}, \quad \text{with } c_1 := \sup_{v\in\mathcal{R}(\mathcal{A}),|v|_{\mathcal{A}}=1}\inf_{c\in\mathcal{N}(\mathcal{A})}(v+c,v+c)_{(\mathcal{Q}\overline{\mathcal{B}}\mathcal{Q})^\dagger}.$$

Proof. We have

$$\begin{aligned}|(\mathcal{I}-\mathcal{B}\mathcal{A})v|_{\mathcal{A}}^2 &= \sup_{v\in\mathcal{R}(\mathcal{A})}\frac{\big((\mathcal{I}-\mathcal{B}\mathcal{A})v,(\mathcal{I}-\mathcal{B}\mathcal{A})v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}}\\ &= 1-\inf_{v\in\mathcal{R}(\mathcal{A})}\frac{\big(\overline{\mathcal{B}}\mathcal{A}v,v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}} = 1-\inf_{v\in\mathcal{R}(\mathcal{A})}\frac{\big(\mathcal{Q}\overline{\mathcal{B}}\mathcal{Q}v,v\big)}{(\mathcal{A}^\dagger v,v)}.\end{aligned}$$

Both $\mathcal{Q}\overline{\mathcal{B}}\mathcal{Q}$ and $\mathcal{A}$ are symmetric and positive definite on $\mathcal{R}(\mathcal{A})$. So $(\mathcal{Q}\overline{\mathcal{B}}\mathcal{Q})^\dagger v\in\mathcal{R}(\mathcal{A})$ and we have

$$c_1 = \sup_{v\in\mathcal{R}(\mathcal{A})}\frac{\big((\mathcal{Q}\overline{\mathcal{B}}\mathcal{Q})^\dagger v,v\big)}{(v,v)_{\mathcal{A}}} = \sup_{v\in\mathcal{R}(\mathcal{A})}\inf_{c\in\mathcal{N}(\mathcal{A})}\frac{\big((\mathcal{Q}\overline{\mathcal{B}}\mathcal{Q})^\dagger(v+c),v+c\big)}{(v,v)_{\mathcal{A}}}.$$

Hence the result. □

Example 2.23 (GS for semidefinite problems)**.** Consider the SPSD problem $A\vec{u}=\vec{f}$. Suppose the GS method can be applied, i.e.

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

then $B=(D+L)^{-1}$. In this case, $\overline{B}=B+B^T-B^TAB$ and it is easy to check that

$$\overline{B}^{-1} = (D+L)D^{-1}(D+L^T) =: S, \quad \text{on } \mathcal{R}(A).$$

From Theorem 2.22, we have

$$c_1 := \sup_{v\in\mathcal{R}(A)}\frac{\big((\mathcal{Q}\overline{B}\mathcal{Q})^\dagger v,v\big)}{(v,v)_A} = \sup_{v\in\mathcal{R}(A)}\frac{\big((\mathcal{Q}S^{-1}\mathcal{Q})^\dagger v,v\big)}{(v,v)_A}.$$

Let $w:=(\mathcal{Q}S^{-1}\mathcal{Q})^\dagger v$. Then $w\in\mathcal{R}(A)$ and

$$w = S(v+c(v)), \quad \text{with } c(v) := S^{-1}w-v.$$

By the definition, we can find $c(v) \in \mathcal{N}(A)$ and

$$c_1 = \sup_{v \in \mathcal{R}(A)} \frac{\big(S(v + c(v)), v + c(v)\big)}{(v, v)_A}.$$

Let $\xi := \arg\inf_{c \in \mathcal{N}(A)} \big(S(v + c), v + c\big)$. Then ξ uniquely satisfies that $\big(S(v + \xi), c\big) = 0$, for all $c \in \mathcal{N}(A)$. Apparently, $\xi = c(v)$ also satisfies this equation. So $\xi = c(v)$ and

$$c_1 = \sup_{v \in \mathcal{R}(A)} \frac{\big(S(v + c(v)), v + c(v)\big)}{(v, v)_A} = \sup_{v \in \mathcal{R}(A)} \inf_{c \in \mathcal{N}(A)} \frac{\big(S(v + c), v + c\big)}{(v, v)_A}.$$

This gives the convergence estimate for the GS method for the semidefinite problems. □

rem:NSingular **Remark 2.24** (GS for a nearly-singular system)**.** Consider a non-singular linear system with the coefficient matrix

$$A_\epsilon := A_s + \epsilon I = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix} + \epsilon \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and solve $A_\epsilon \vec{u} = \vec{f}$ using the GS method. We notice that A_s is singular. In fact, it is symmetric positive semi-definite (from the Poisson problem with Neumann boundary condition). On the other hand, A_ϵ is non-singular for $\epsilon > 0$. In fact, with the convergence results in Theorem 4.10

ϵ	1	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	0
Number of iterations	14	94	823	7,427	66,556	588,770	2

tab:NSingular Table 2.1: Number of iterations until the energy norm $\|\vec{u} - \vec{u}^{(k)}\|_{A_\epsilon} \leqslant 10^{-6}$.

and some simple calculations, we can derive that the convergence rate of the GS method is determined by

$$c_0 = c_1 - 1 = \sup_{\vec{v} \in \mathbb{R}^3} \frac{(1 + \epsilon)^{-1} v_2^2 + (2 + \epsilon)^{-1} v_3^2}{(A_\epsilon \vec{v}, \vec{v})}$$

Choosing a vector $\vec{v} = (v_1, v_2, v_3)^T = (1, 1, 1)^T$ from the null space of A_s and plugging into the above equality, we find that

$$c_0 \geqslant 1 + \frac{3 + 2\epsilon}{3\epsilon(1 + \epsilon)(2 + \epsilon)} > \frac{1}{\epsilon(1 + \epsilon)(2 + \epsilon)} \to \infty, \quad \text{as } \epsilon \to 0.$$

So we can expect the convergence of GS will deteriorate as ϵ becomes small and the linear system becomes close to singular. The numerical results are listed in Table 2.1 for various $\epsilon \geqslant 0$. □

2.1.7 An example: gradient descent method

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

$$\min_{u\in V} \mathcal{F}(u) := \frac{1}{2}(\mathcal{A}u, u) - (f, u). \tag{2.19}$$ eqn:minform

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 2.3), 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

$$\alpha_{\text{opt}} = \frac{(f - \mathcal{A}u^{\text{old}}, p)}{(\mathcal{A}p, p)} = \frac{(r^{\text{old}}, p)}{(\mathcal{A}p, p)}, \quad \text{with } r^{\text{old}} = f - \mathcal{A}u^{\text{old}}. \tag{2.20}$$ eqn:optimalst

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 2.25 (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. □

The SD method is both straightforward to implement and computationally inexpensive, with each step only necessitating one matrix-vector multiplication and two inner products. Unfortunately, this method typically converges rather slowly. The subsequent algorithm description of the SD method illustrates this point:

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
```

ex:relax **Example 2.26** (Line-search and the GS 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 step:

$$u_i^{\text{new}} = u_i^{(i-1)} + \frac{f_i - \sum_{j=1}^N a_{i,j}\, u_j^{(i-1)}}{a_{i,i}} = \frac{1}{a_{i,i}}\Big(f_i - \sum_{j<i} a_{i,j}u_j^{\text{new}} - \sum_{j>i} a_{i,j}u_j^{\text{old}}\Big). \tag{2.21}$$ eqn:GSiter

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

ex:GS-Schwarz **Remark 2.27** (The GS method and Schwarz method)**.** Based on (2.21), we can write the GS error propagation matrix in a different form

$$I - BA = (I - I_N a_{N,N}^{-1} I_N^T A)\cdots(I - I_1 a_{1,1}^{-1} I_1^T A) = (I - \Pi_N)\cdots(I - \Pi_1), \tag{2.22}$$ eqn:GSoper

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 GS will be further discussed later in the framework of Schwarz method and subspace correction method. □

Theorem 2.28 (Convergence rate of steepest descent method)**.** If we apply the exact line-search with 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

$$\big\|u - u^{(m)}\big\|_{\mathcal{A}} \leqslant \left(\frac{\kappa(\mathcal{A}) - 1}{\kappa(\mathcal{A}) + 1}\right)^m \big\|u - u^{(0)}\big\|_{\mathcal{A}}. \tag{2.23}$$ eqn:rateSD

Proof. The exact line-search stepsize is easy to obtain by 1D quadratic programming. At the m-th iteration, the energy functional 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-hand 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 [150], 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. □

The Kantorovich inequality is a fundamental tool that is used in the study of the convergence properties of descent methods. It provides an upper bound on the difference between two functions that are close to each other, in terms of their derivatives. This inequality is often used in the analysis of gradient descent methods, which are commonly used in optimization to find the minimum of a function. By providing a bound on the difference between the objective function and its approximation, the Kantorovich inequality helps to establish the rate at which the algorithm converges to the optimum.

2.2 Krylov subspace methods

sec:KSM

Nonstationary iterative methods are generally favored as standalone solvers in practical applications. One prominent class of nonstationary methods is the Krylov subspace method (KSM) [103]. This section mainly follows the discussions in [204]. Suppose $\mathcal{A} : V \mapsto V$ is an invertible operator. According to the Cayley–Hamilton theorem (HW 2.4), there exists a $\mathcal{P}_{N-1}$ polynomial $q_{N-1}(\lambda)$ of degree not exceeding $N-1$ such that $\mathcal{A}^{-1} = q_{N-1}(\mathcal{A})$. As a result, the solution u of the linear system can be expressed as $u = q_{N-1}(\mathcal{A}) f$.

Krylov subspace methods formulate iterative approximations to u in

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

The Krylov subspace methods are powerful techniques for solving large linear systems, especially when the coefficient matrix is too large to be explicitly stored. They work by constructing a sequence of increasingly accurate approximations to the solution using the subspaces spanned by the successive matrix-vector products. The Conjugate Gradient (CG) method, a specific Krylov subspace method, is particularly suitable for symmetric positive definite matrices. It efficiently converges to the solution using only matrix-vector multiplications, avoiding the need to store the entire matrix. Other Krylov methods, like GMRES (Generalized Minimal Residual) and BiCGstab (Biconjugate Gradient Stabilized), are suitable for nonsymmetric matrices.

Finding a suitable approximation for u in $\mathcal{K}_m(\mathcal{A}, f)$ efficiently is essential for practical applications. However, the definition of a "good" approximation varies depending on the specific problem and desired properties of the solution. Therefore, various approaches have been proposed to define the optimal approximation. The selection of criterion and method involves a trade-off between accuracy, efficiency, stability, and so on.

2.2.1 Arnoldi method

We now assume that the linear operator $\mathcal{A}$ is repeatedly applied to a vector $v \in V$:

$$v^{(0)} = v, \quad v^{(1)} = \mathcal{A}v^{(0)} = \mathcal{A}v, \quad \dots \quad v^{(m)} = \mathcal{A}v^{(m-1)} = \mathcal{A}^m v, \quad \dots$$

and the sequence $\{v^{(0)}, v^{(1)}, \dots, v^{(m)}, \dots\}$ is the so-called *Krylov sequence*. As we mentioned earlier, the *Krylov subspace* is defined

$$\mathcal{K}_m(\mathcal{A}, v) := \operatorname{span}\{v, \mathcal{A}v, \mathcal{A}^2 v, \dots, \mathcal{A}^{m-1} v\} = \operatorname{span}\{v^{(0)}, v^{(1)}, v^{(2)}, \dots, v^{(m-1)}\}.$$

Apparently, we have $\mathcal{K}_m(\mathcal{A}, v) = \{p(\mathcal{A})v \,:\, p \in \mathcal{P}_{m-1}\}$; see HW 2.5. The corresponding matrix form

$$K_m(\mathcal{A}, v) := \left[v, \mathcal{A}v, \mathcal{A}^2 v, \dots, \mathcal{A}^{m-1} v\right] = \left[v^{(0)}, v^{(1)}, v^{(2)}, \dots, v^{(m-1)}\right].$$

is referred to as the *Krylov matrix*.

Remark 2.29 (Instability of the Krylov matrix)**.** It is clear that if we normalize each vector $v^{(m)}$, the Krylov sequence above corresponds to the Power iteration, which converges to the eigenvector associated with the largest eigenvalue of $\mathcal{A}$. However, this may cause linear dependence among the vectors in the Krylov sequence, resulting in numerical instability during the iteration process. We also note that, if (λ, v) is an eigen-pair of $\mathcal{A}$, then

$$\mathcal{K}_m(\mathcal{A}, v) = \mathcal{K}_1(\mathcal{A}, v), \quad m = 2, 3, \dots$$

This implies that the Krylov subspace will not expand any longer. □

Special care must be taken to mitigate numerical instabilities when constructing the Krylov subspace to ensure the reliability and accuracy of the results.

Remark 2.30 (Characteristic polynomial)**.** The characteristic polynomial of $\mathcal{A}$ can be written as

$$|\lambda\mathcal{I} - \mathcal{A}| := \lambda^N + \alpha_{N-1}\lambda^{N-1} + \cdots + \alpha_1\lambda + \alpha_0.$$

From the Cayley–Hamilton theorem, we can obtain that

$$\mathcal{A}^N + \alpha_{N-1}\mathcal{A}^{N-1} + \cdots + \alpha_1\mathcal{A} + \alpha_0 = 0$$

and, hence,

$$v^{(N)} + \alpha_{N-1}v^{(N-1)} + \cdots + \alpha_1 v^{(1)} + \alpha_0 v^{(0)} = 0.$$

□

Suppose that the Krylov matrix has full column-rank. We apply the QR factorization to the Krylov matrix such that

$$K_m(\mathcal{A}, v) = Q_m R_m,$$

where $Q_m \in \mathbb{R}^{N\times m}$ and $Q_m^T Q_m = I$, $R_m \in \mathbb{R}^{m\times m}$ is a nonsingular upper triangular matrix. We notice that

$$\mathcal{K}_m(\mathcal{A}, v) = \mathcal{R}(Q_m) = \text{span}\{q_1, q_2, \ldots, q_m\},$$

i.e., the columns of Q_m form an orthonormal basis of $\mathcal{K}_m(\mathcal{A}, v)$.

Theorem 2.31 (Arnoldi decomposition)**.** Given $A \in \mathbb{R}^{N\times N}$ and $0 \neq v \in \mathbb{R}^N$. If $K_{m+1}(A, v)$ has full column rank and Q_{m+1} is the Q-factor of the QR-factorization of $K_{m+1}(A, v)$, then there exists an irreducible upper Hessenberg matrix $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ such that

$$AQ_m = Q_{m+1}\bar{H}_m. \tag{2.24}$$ eqn:Arnoldi

Proof. Since $K_m(A, v) = Q_m R_m$, we have

$$\left[Av, A^2v, \ldots, A^m v\right] = AK_m(A, v) = AQ_m R_m.$$

On the other hand,

$$[v, AK_m(A, v)] = K_{m+1}(A, v) = Q_{m+1}R_{m+1}.$$

By comparing the last m columns of the above two matrices, we obtain that

$$AQ_m R_m = Q_{m+1}\tilde{H}_m,$$

where $\tilde{H}_m := R_{m+1}(:, 2 : m + 1) \in \mathbb{R}^{(m+1)\times m}$ is the last m columns of R_{m+1}. Apparently, $\tilde{H}_m$ is irreducible and upper Hessenberg since R_{m+1} is nonsingular and upper triangular. Hence we can obtain (2.24) with $\bar{H}_m := \tilde{H}_m R_m^{-1}$ which is also irreducible and upper Hessenberg. □

Remark 2.32 (Practical form of the Arnoldi decomposition)**.** The irreducible upper Hessenberg matrix $\bar{H}_m \in \mathbb{R}^{(m+1)\times m}$ can be rewritten as

$$\bar{H}_m = \begin{bmatrix} H_m \\ \beta_m e_m^T \end{bmatrix}.$$

This gives us a more useful form of the *Arnoldi decomposition* (2.24), i.e.,

$$AQ_m = Q_m H_m + \beta_m q_{m+1} e_m^T, \tag{2.25}$$ eqn:Arnoldi1

which provides an iterative scheme to construct Q_{m+1}. By multiplying Q_m^T on both sides, we can obtain that

$$H_m = Q_m^T A Q_m. \tag{2.26}$$ eqn:RayleighQ

□

Using the Arnoldi decomposition (2.25), we find that the last columns on both sides are

$$Aq_m = Q_m h_m + \beta_m q_{m+1}.$$

And by looking at the right-hand side of (2.26), we get $h_m = Q_m^T A q_m$. With the help of this relation, we can obtain the next iteration

$$\beta_m q_{m+1} = Aq_m - Q_m h_m,$$

where

$$\beta_m := \|Aq_m - Q_m h_m\|_0$$

and

$$q_{m+1} := \frac{Aq_m - Q_m h_m}{\beta_m}.$$

This is actually the Gram-Schmidt (GS) orthogonalization. To improve stability, people usually employ a modified procedure (MGS, which uses updated vectors w in Line 4), which gives the well-known Arnoldi method[3].

Listing 2.2: Arnoldi method with MGS

```
1 %% Given a normalized vector q_1 with ||q_1||_0 = 1 and the iteration number m;
2 for j = 1, 2, ..., m
3     w ← A q_j;
4     h_ij ← (w, q_i), w ← w − h_ij q_i,   i = 1, 2, ..., j;
5     h_{j+1,j} = ||w||_0;
6     if h_{j+1,j} = 0
```

[3]Another approach is to use the Householder transformation to enhance stability, although it typically involves a higher computational cost.

```
        m ← j, break;
    end
    q_{j+1} ← w/h_{j+1,j};
end
```

2.2.2 Lanczos method

The Lanczos method is an iterative numerical method for solving large symmetric matrices. The Lanczos method works by finding an orthonormal basis for the Krylov subspace generated by successive multiplication of a starting vector with the given matrix. To do this, the method repeatedly computes the matrix-vector product of the matrix and the current Krylov basis vector, and then orthogonalizes the resulting vector with respect to the previous vectors in the Krylov subspace using the Gram-Schmidt process.

If $A \in \mathbb{R}^{N\times N}$ is symmetric, the left-hand side of (2.26) simplifies to a tridiagonal matrix

$$H_m = Q_m^T A Q_m = T_m := \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \beta_{m-1} & \alpha_m \end{bmatrix}$$

Correspondingly, we have the so-called *Lanczos decomposition*:

$$AQ_m = Q_m T_m + \beta_m q_{m+1} e_m^T. \tag{2.27}$$ eqn:Lanczos1

Hence we have

$$\begin{aligned} Aq_1 &= \alpha_1 q_1 + \beta_1 q_2 \\ Aq_j &= \beta_{j-1} q_{j-1} + \alpha_j q_j + \beta_j q_{j+1}, \quad j = 2, 3, \ldots, m \end{aligned}$$

According to the orthogonality of the columns of Q_m, this procedure can be written as the following Lanczos method.

Listing 2.3: Lanczos method

```
%% Given a normalized vector q_1 with ‖q_1‖_0 = 1 and max iteration number m;
β_0 ← 0, q_0 ← 0;
for j = 1, 2, ..., m
    w ← A q_j;
    α_j ← (w, q_j);
    w ← w − α_j q_j − β_{j−1} q_{j−1};
    β_j = ‖w‖_0;
    if β_j = 0
        m ← j, break;
```

```
    end
    q_{j+1} ← w/β_j;
end
```

Remark 2.33 (Lanczos procedure for non-symmetric problems)**.** For non-symmetric matrices, the Lanczos method can provide a pair of biorthogonal bases for two subspaces:

$$\mathcal{K}_m(\mathcal{A}, v) := \text{span}\{v, \mathcal{A}v, \ldots, \mathcal{A}^{m-1}v\}$$

and

$$\mathcal{K}_m(\mathcal{A}^T, w) := \text{span}\{w, \mathcal{A}w, \ldots, \mathcal{A}^{m-1}w\}.$$

In this case,

$$T_m = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \delta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{m-1} \\ & & \delta_{m-1} & \alpha_m \end{bmatrix}.$$

In this procedure, we need to store two sets of basis vectors, unlike in Arnoldi's algorithm, which requires only one. A more significant issue with the Lanczos algorithm for non-symmetric matrices is the potential risk of breakdown. This occurs when $(v_j, w_j) \approx 0$, causing the algorithm to halt without identifying an appropriate affine space that contains the true solution. Therefore, it requires a different approach for non-symmetric problems. □

2.2.3 Conjugate gradient method

We consider an SPD linear operator $\mathcal{A}$. Now we apply a descent direction method with search direction $p^{(m)}$, i.e.

$$u^{(m+1)} = u^{(m)} + \alpha_m p^{(m)}. \tag{2.28}$$ eqn:descent

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

$$\alpha_m := \frac{\left(r^{(m)}, p^{(m)}\right)}{\left(p^{(m)}, p^{(m)}\right)_{\mathcal{A}}}. \tag{2.29}$$ eqn:alpha

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 the new search direction to be $\mathcal{A}$-conjugate to the previous direction:

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

By simple calculations, we obtian the weight

$$\beta_m := -\frac{\left(\mathcal{A}r^{(m+1)}, p^{(m)}\right)}{\left(\mathcal{A}p^{(m)}, p^{(m)}\right)}. \tag{2.30}$$ eqn:beta

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

lem:cg1 **Lemma 2.34** (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. It guarantees that we can apply a short recurrence iteration procedure while keep all directions are orthogonal to each other; see HW 2.7.

lem:cg2 **Lemma 2.35** (Stepsizes for CG)**.** For the conjugate gradient method, we have the 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 trivial transformations, but it is essential for implementation, which is described as follows:

Listing 2.4: 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
```

We can summarize the above CG iterative procedure as the following equations:

$$\left.\begin{aligned} & r^{(0)} = f - \mathcal{A}u^{(0)}, \quad p^{(0)} = r^{(0)}, \quad \rho_0 = (r^{(0)}, r^{(0)}); \\ & \sigma_m = (p^{(m)}, \mathcal{A}p^{(m)}), \quad \alpha_m = \rho_m/\sigma_m; \\ & u^{(m+1)} = u^{(m)} + \alpha_m p^{(m)}; \\ & r^{(m+1)} = r^{(m)} - \alpha_m \mathcal{A}p^{(m)}; \\ & \rho_{m+1} = (r^{(m+1)}, r^{(m+1)}), \quad \beta_{m+1} = \rho_{m+1}/\rho_m; \\ & p^{(m+1)} = r^{(m+1)} + \beta_{m+1} p^{(m)}; \end{aligned}\right\} \quad m = 0, 1, 2, \ldots \tag{2.31}$$

method:CG

Remark 2.36 (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. □

Remark 2.37 (From Lanczos to CG)**.** If we want to find the best approximation of u in the Krylov subspace $\mathcal{K}_m(\mathcal{A}, f)$ with respect to the $\mathcal{A}$-norm (assuming initial guess $u^{(0)} = 0$), it is equivalent to find $u^{(m)} \in \mathcal{K}_m(\mathcal{A}, f)$ such that

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

see HW 2.6. The Lanczos method starting from $q_1 := f/\|f\|_0$ is associated with the Q-factor Q_m whose columns is an orthonormal basis of $\mathcal{K}_m(\mathcal{A}, f)$. That is equivalent to find $y_m \in \mathbb{R}^m$ and $u^{(m)} = Q_m y_m$ such that

$$Q_m^T(f - \mathcal{A}u^{(m)}) = Q_m^T(f - \mathcal{A}Q_m y_m) = 0.$$

Notice that $T_m = Q_m^T \mathcal{A} Q_m$. It is then equivalent to solve a symmetric tridiagonal linear system

$$T_m\, y_m = \|f\|_0 e_1.$$

Since T_m is SPD, there exists a LU decomposition, which can be computed without pivoting. Then we can write the approximate solution

$$u^{(m)} = Q_m U_m^{-1} L_m^{-1} \|f\|_0 e_1,$$

in which L_m has unit diagonal entries. In turn, we need to solve, using substitution, the equations:

$$L_m\, y_m = \|f\|_0 e_1, \quad P_m\, U_m = Q_m, \quad u^{(m)} = P_m\, y_m.$$

This will give rise to the conjugate gradient method as well [172]. □

rem:CG-poly **Remark 2.38** (Polynomial form of CG)**.** It is straightforward to check that, for the CG method,

$$\mathcal{K}_{m+1}(\mathcal{A}, r^{(0)}) = \operatorname{span}\left\{r^{(0)}, r^{(1)}, \ldots, r^{(m)}\right\} = \operatorname{span}\left\{p^{(0)}, p^{(1)}, \ldots, p^{(m)}\right\}$$

and $p^{(0)} = r^{(0)}$. By examining the CG method (2.31), we find that there exist two polynomials $\phi_m, \psi_m \in \mathcal{P}_m$, such that

$$r^{(m)} = \phi_m(\mathcal{A})\, r^{(0)}, \quad p^{(m)} = \psi_m(\mathcal{A})\, p^{(0)} = \psi_m(\mathcal{A})\, r^{(0)}.$$

From the above equation, we can see that, to get a convergent method, $\phi_m(\mathcal{A})$ must be a contraction in some sense. By setting $\phi_0 = 1$ and $\psi_{-1} = 0$, we have the iterations

$$\phi_{m+1} = \phi_m - \alpha_m\, t\, \psi_m, \quad \psi_m = \phi_m + \beta_m\, \psi_{m-1}.$$

Define a bilinear form $[\phi \cdot \psi] := \left(\phi(\mathcal{A}) r^{(0)}, \psi(\mathcal{A})\, r^{(0)}\right) = {r^{(0)}}^T \phi(\mathcal{A})\, \psi(\mathcal{A})\, r^{(0)}$. Using these notation, we have a polynomial form of the CG method:

$$\left.\begin{aligned} &\phi_0 = 1, \quad \psi_{-1} = 0, \quad \rho_{-1} = 1; \\ &\rho_m = [\phi_m \cdot \phi_m], \quad \beta_m = \rho_m/\rho_{m-1}; \\ &\psi_m = \phi_m + \beta_m \psi_{m-1}; \\ &\sigma_m = [\psi_m \cdot t\psi_m], \quad \alpha_m = \rho_m/\sigma_m; \\ &\phi_{m+1} = \phi_m - \alpha_m t \psi_m; \end{aligned}\right\} \quad m = 0, 1, 2, \ldots \tag{2.32}$$

method:CG-poly

Based on Lemma 2.34, we notice that $[\phi_i \cdot \phi_j] = \rho_i \delta_{ij}$ and $[\psi_i \cdot t\psi_j] = \sigma_i \delta_{ij}$. □

The CG method has been proven to converge significantly faster than the steepest descent method in practice, as demonstrated by the following theorem.

Theorem 2.39 (Convergence rate of CG)**.** The convergence rate of the CG iteration satisfies the following estimate:

$$\left\|u - u^{(m)}\right\|_{\mathcal{A}} \leqslant 2\left(\frac{\sqrt{\kappa(\mathcal{A})} - 1}{\sqrt{\kappa(\mathcal{A})} + 1}\right)^m \left\|u - u^{(0)}\right\|_{\mathcal{A}}. \tag{2.33}$$

eqn:rateCG

Proof. We only give a sketch of proof here. From Lemma 2.34, 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

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

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 2.8 as a guideline to complete the proof. □

If the eigenvalues of $\mathcal{A}$ are distributed uniformly in the range $[\lambda_{\min},\lambda_{\max}]$, the upper bound stated in (2.33) is proven to be optimal. Notably, a few isolated poor eigenvalues barely affect the asymptotic convergence of the CG method. However, in such cases, the bound (2.33) is no longer sharp, and the rate of asymptotic convergence can instead be approximated by the effective condition number [7, 8].

rem:EffCondNum **Remark 2.40** (Effective condition number)**.** 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

$$\frac{\|u-u^{(m)}\|_{\mathcal{A}}}{\|u-u^{(0)}\|_{\mathcal{A}}} \leqslant 2C\left(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\right)^{m-m_0}, \qquad m\geqslant m_0 \tag{2.34}$$ eqn:rateCG2

where the *effective condition number* is defined as

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

We can use (2.34) to estimate the rate of convergence of the CG subspace methods instead. In this estimate, the constant

$$C := \max_{\lambda\in\sigma_{\text{eff}}(\mathcal{A})}\prod_{\mu\in\sigma_{\text{iso}}(\mathcal{A})}\left|1-\frac{\lambda}{\mu}\right| \leqslant \big(\kappa(A)-1\big)^{m_0}.$$

In particular, $C\leqslant 1$ if σ_{iso} contains only isolated large eigenvalues. □

rem:NCG **Remark 2.41** (Nonlinear conjugate gradient)**.** By considering minimization of a more general functional $\mathcal{F}(\cdot)$ over a quadratic functional, steepest descent directions based search directions can still be used. The Flecther-Reeves formula [95] and related methods can be utilized for parameter β, while a line search algorithm can be applied for stepsize α. □

2.2.4 Some variants of CG method ⋆

In large-scale parallel computation, we often need to reduce or hide communication costs associated with the inner products needed in the CG method. Here we introduce two variants

of the CG method, namely the pipelined CG method (toward communication-hiding methods) and the s-step CG method (communication-avoiding). The general idea is to reduce cost of communications and synchronisations by avoiding communications or overlapping communications with computation. These kinds of techniques are very important for communication-bounded algorithms.

Listing 2.5: Pipelined conjugate gradient method

```
%% Given an initial guess u and a tolerance ε;
r ← f − A u,  p ← r;
s ← A p,  w ← A r,  z ← A w;
α ← (r, r)/(p, s);
while ‖r‖ > ε
    ũ ← u + α p;
    r̃ ← r − α s;
    w̃ ← w − α z;
    z̃ ← A w̃;  %% SpMV, asymc
    β ← (r̃, r̃)/(r, r);
    α ← (r̃, r̃) / ((w, r) − (β/α)(r̃, r̃));
    p̃ ← r̃ + β p;
    s̃ ← w̃ + β s;
    z̃ ← z̃ + β z;  %% Results of SpMV needed here!
    Update:  u ← ũ,  r ← r̃,  p ← p̃,  w ← w̃,  s ← s̃,  z ← z̃;
end
```

Listing 2.6: s-Step conjugate gradient method

```
%% Given an initial guess u_1 and a tolerance ε;
r_1 ← f − A u_1;
for k = 0:MaxIter
    for j = 1:s
        w_{sk+j} ← A r_{sk+j};  %% P2P communication
        μ_{sk+j} ← (r_{sk+j}, r_{sk+j});  %% Test for convergence; All_reduce
        ν_{sk+j} ← (w_{sk+j}, r_{sk+j});
        γ_{sk+j} ← μ_{sk+j}/ν_{sk+j};
        if sk + j == 1
            ρ_{sk+j} ← 1;
        else
            ξ_1 ← γ_{sk+j}/γ_{sk+j−1};
            ξ_2 ← μ_{sk+j}/μ_{sk+j−1};
            ρ_{sk+j} ← (1 − ξ_1 ξ_2/ρ_{sk+j−1})^{−1};
        end
        x_{sk+j+1} ← ρ_{sk+j}(x_{sk+j} + γ_{sk+j} r_{sk+j}) + (1 − ρ_{sk+j}) x_{sk+j−1};
        r_{sk+j+1} ← ρ_{sk+j}(r_{sk+j} − γ_{sk+j} w_{sk+j}) + (1 − ρ_{sk+j}) r_{sk+j−1};
    end
end
```

2.2.5 Minimal residual methods

We first examine the scenario where $\mathcal{A} : V \mapsto V$ is an indefinite and symmetric isomorphism. In this case, a technique akin to the previously discussed Lanczos method can be employed. For simplicity, we assume a zero initial guess $u^{(0)} = 0$, but note that it is not a necessary condition. This Krylov subspace method is characterized by

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

In this case, the Lanczos decomposition (2.27) reads

$$\mathcal{A}Q_m = Q_{m+1}\bar{T}_m,$$

where $Q_m \in \mathbb{R}^{N \times m}$ is the Q-factor of the Krylov matrix (i.e. $Q_m e_1 = f/\|f\|_0$) and

$$\bar{T}_m := \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m-2} & \alpha_{m-1} & \beta_{m-1} \\ & & & \beta_{m-1} & \alpha_m \\ & & & & \beta_m \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}.$$

Assume that $\beta_0 := \|f\|_0$ [4] and $u^{(m)} = Q_m y_m$ with $y_m \in \mathbb{R}^m$. We have

$$\|f - \mathcal{A}u^{(m)}\|_0 = \|f - \mathcal{A}Q_m y_m\|_0 = \|\beta_0 Q_{m+1} e_1 - Q_{m+1}\bar{T}_m y_m\|_0 = \|\beta_0 e_1 - \bar{T}_m y_m\|_0.$$

We need to find $y_m \in \mathbb{R}^m$ such that

$$\|\beta_0 e_1 - \bar{T}_m y_m\|_0 = \min_{y \in \mathbb{R}^m} \|\beta_0 e_1 - \bar{T}_m y\|_0.$$

We can apply the Givens transforms on the upper Hessenberg matrix $\bar{T}_m$ to obtain an upper triangular matrix

$$G_m G_{m-1} \cdots G_2 G_1 \bar{T}_m = \begin{bmatrix} R_m \\ 0 \end{bmatrix}, \quad \text{and} \quad R_m = \begin{bmatrix} \gamma_1 & \delta_1 & \varepsilon_1 & & \\ & \gamma_2 & \delta_2 & \ddots & \\ & & \ddots & \ddots & \varepsilon_{m-2} \\ & & & \gamma_{m-1} & \delta_{m-1} \\ & & & & \gamma_m \end{bmatrix} \in \mathbb{R}^{m \times m}.$$

[4] Note that we are using the zero initial guess.

Here the Givens transforms have the following form

$$G_i := \begin{bmatrix} I_{i-1} & 0 & 0 & 0 \\ 0 & c_i & s_i & 0 \\ 0 & -s_i & c_i & 0 \\ 0 & 0 & 0 & I_{m-i} \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+1)} \tag{2.35}$$ eqn:Givens

and $G := G_m G_{m-1} \cdots G_2 G_1$. In the Lanczos method, $\beta_i \neq 0$; this implies $\gamma_i \neq 0$ and hence R_m is nonsingular.

By the above definition, we have

$$\beta_0 G e_1 = G(\beta_0 e_1) =: \begin{bmatrix} \zeta_m \\ \rho_m \end{bmatrix}, \quad \zeta_m := \begin{bmatrix} z_1 \\ \vdots \\ z_m \end{bmatrix}.$$

It is easy to check that

$$\begin{aligned} z_1 &= \beta_0 c_1, \\ z_i &= (-1)^{i-1} \beta_0 s_1 s_2 \cdots s_{i-1} c_i, \quad i = 2, 3, \ldots, m \\ \rho_m &= (-1)^m \beta_0 s_1 s_2 \cdots s_m. \end{aligned} \tag{2.36}$$ eqn:tau_rho

We then have

$$\|\bar{T}_m y_m - \beta_0 e_1\|_0^2 = \|G\bar{T}_m y - \beta_0 G e_1\|_0^2 = \left\| \begin{bmatrix} R_m \\ 0 \end{bmatrix} y - \begin{bmatrix} \zeta_m \\ \rho_m \end{bmatrix} \right\|_0^2 = \|R_m y - \zeta_m\|_0^2 + \rho_m^2.$$

This means that $y_m = R_m^{-1}\zeta_m$ and

$$u^{(m)} = Q_m y_m = Q_m R_m^{-1} \zeta_m$$

gives the so-called minimum residual (MINRES) method. Apparently, this form of the MINRES method is not efficient because it is expensive to store Q_m. The practical implement of MINRES is based on the following observations.

Suppose that Q_m is known. Let $P_m := Q_m R_m^{-1}$ and hence $P_m R_m = Q_m$. Then it is easy to check that

$$\begin{aligned} \gamma_1 p_1 &= q_1, \\ \delta_1 p_1 + \gamma_2 p_2 &= q_2, \\ \varepsilon_{j-2} p_{j-2} + \delta_{j-1} p_{j-1} + \gamma_j p_j &= q_j, \quad j = 3, 4, \ldots, m \end{aligned}$$

and it can be solved as

$$\begin{aligned} p_1 &= q_1/\gamma_1, \\ p_2 &= \big(q_2 - \delta_1\, p_1\big)/\gamma_2, \\ p_j &= \big(q_j - \delta_{j-1}\, p_{j-1} - \varepsilon_{j-2}\, p_{j-2}\big)/\gamma_j, \quad j = 3, 4, \ldots, m \end{aligned}$$

We can also easily see that

$$\bar{T}_{m+1} = \begin{bmatrix} \bar{T}_m & \bar{t}_{m+1} \\ 0 & \beta_{m+1} \end{bmatrix} \in \mathbb{R}^{(m+2)\times(m+1)} \quad \text{with} \quad \bar{t}_{m+1} := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \beta_m \\ \alpha_{m+1} \end{bmatrix} \in \mathbb{R}^{m+1}.$$

And using the Givens transformation, we wish to obtain

$$R_{m+1} = \begin{bmatrix} R_m & r_{m+1} \\ 0 & \gamma_{m+1} \end{bmatrix} \in \mathbb{R}^{(m+1)\times(m+1)} \quad \text{with} \quad r_{m+1} := \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \varepsilon_{m-1} \\ \delta_m \end{bmatrix} \in \mathbb{R}^{m}.$$

By some calculations, we get

$$\begin{aligned} \varepsilon_{m-1} &= s_{m-1}\beta_m, \\ \hat{\beta}_m &= c_{m-1}\beta_m, \\ \delta_m &= c_m\hat{\beta}_m + s_m\alpha_{m+1}, \\ \hat{\alpha}_{m+1} &= -s_m\hat{\beta}_m + c_m\alpha_{m+1}, \\ \begin{bmatrix} \gamma_{m+1} \\ 0 \end{bmatrix} &= \begin{bmatrix} c_{m+1} & s_{m+1} \\ -s_{m+1} & c_{m+1} \end{bmatrix} \begin{bmatrix} \hat{\alpha}_{m+1} \\ \beta_{m+1} \end{bmatrix}. \end{aligned}$$

Furthermore, from (2.36), we have

$$z_{m+1} = \rho_m c_{m+1}, \quad \rho_{m+1} = -\rho_m s_{m+1}.$$

And we have already showed that

$$p_{m+1} = \big(q_{m+1} - \delta_m\, p_m - \varepsilon_{m-1}\, p_{m-1}\big)/\gamma_{m+1}$$

and, in turn,

$$u^{(m+1)} = P_{m+1}\zeta_{m+1} = \begin{bmatrix} P_m & p_{m+1} \end{bmatrix} \begin{bmatrix} \zeta_m \\ z_{m+1} \end{bmatrix} = u^{(m)} + z_{m+1}\, p_{m+1}.$$

This is the practical MINRES method which only requires to store five vectors $u^{(m)}$, p_m, p_{m-1}, q_{m+1}, and q_m in order to compute the new iteration $u^{(m+1)}$.

Remark 2.42 (Convergence of minimum residual method)**.** We can derive analytically that (see, for example, [103])

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

In this case, the following crude convergence estimate holds

$$\left\|r^{(m)}\right\|_0 = \left\|\mathcal{A}(u-u^{(m)})\right\|_0 \leqslant 2\left(\frac{\kappa(\mathcal{A})-1}{\kappa(\mathcal{A})+1}\right)^m \left\|\mathcal{A}(u-u^{(0)})\right\|_0 = 2\left(\frac{\kappa(\mathcal{A})-1}{\kappa(\mathcal{A})+1}\right)^m \left\|r^{(0)}\right\|_0. \quad (2.37)$$

eqn:rateMINRES

If all the eigenvalues are positive, we can get sharp convergence estimate using Chebyshev polynomials. It is more challenging to get a general yet sharp estimate for indefinite problems; see [171] for more details. □

On the other hand, if the linear operator $\mathcal{A} : V \mapsto V$ is not symmetric, then we need to apply the Arnoldi method (2.24) instead:

$$\mathcal{A}Q_m = Q_m H_m + \beta_m q_{m+1} e_m^T = Q_{m+1}\bar{H}_m.$$

Here $Q_{m+1} := [Q_m, q_{m+1}] \in \mathbb{R}^{N\times(m+1)}$ satisfies $Q_{m+1}^T Q_{m+1} = I_{m+1}$. The upper Hessenberg matrix

$$\bar{H}_m = \begin{bmatrix} H_m \\ \beta_m e_m^T \end{bmatrix} \in \mathbb{R}^{(m+1)\times m}.$$

Like before, we assume that $\beta_0 := \|f\|_0$ and $u^{(m)} = Q_m y_m$ with $y_m \in \mathbb{R}^m$. We have

$$\|f - \mathcal{A}u^{(m)}\|_0 = \|f - \mathcal{A}Q_m y_m\|_0 = \|\beta_0 Q_{m+1}e_1 - Q_{m+1}\bar{H}_m y_m\|_0 = \|\beta_0 e_1 - \bar{H}_m y_m\|_0.$$

Then we apply the Givens transformation to $\bar{H}_m$ to solve this least squares problem. This gives rise to the well-known GMRES method.

The residual norms in the GMRES method are guaranteed to be monotonically non-increasing, and the method terminates with the exact solution in at most N iterations, assuming exact arithmetic. However, unlike the MINRES method, GMRES does not offer an efficient way to avoid saving $O(mN)$ vectors, making it impractical to allow very large m. In practice, the iterative process is restarted with a relatively small m, leading to the restarted GMRES(m) method. The computational complexity of GMRES is approximately $O(m^2N)$, meaning both memory and computational costs rise significantly as m increases. To prevent these costs from becoming prohibitively high, it is necessary to periodically terminate and restart the iterations, which is the basis of the restarted GMRES approach.

Remark 2.43 (Generalized minimum residual method)**.** For general linear system, we can apply the Krylov subspace method: Find $u^{(m)} \in u^{(0)} + \mathcal{K}_m(r^{(0)})$, such that $f - \mathcal{A}u^{(m)} \perp \mathcal{L}_m$, where the m-dimension subspace $\mathcal{L}_m \subset V$ is given. In GMRES method [172], for example, $\mathcal{L}_m = \mathcal{A}\mathcal{K}_m$, in which $\mathcal{K}_m$ is the m-th Krylov subspace. In each iteration of GMRES, we minimize the residual norm over all vectors in $u^{(0)} + \mathcal{K}_m$. □

Listing 2.7: Generalized minimum residual method

```
%% Given an initial guess u and the maximal iteration number m;
r ← f − Au, β ← ‖r‖, v_1 ← r/β;
for j = 1, 2, ..., m
    w_j ← Av_j;
    h_ij ← (w_j, v_i), w_j ← w_j − h_ij v_i,   i = 1, 2, ..., j;
    h_{j+1,j} = ‖w_j‖_0;
    if h_{j+1,j} = 0
        m ← j, break;
    end
    v_{j+1} ← w_j/h_{j+1,j};
end
H̄_m ← {h_ij}_{1≤i≤m+1, 1≤j≤m};
y_m ← argmin_y ‖βe_1 − H̄_m y‖_0;
Update: u ← u + v^(m) y_m;
```

2.2.6 Biconjugate gradient methods

We have expressed the CG method in the form of a polynomial iteration in Remark 2.38. If $\mathcal{A}$ is not symmetric, we can revise the bilinear form as

$$[\phi \cdot \psi] := \left(\phi(\mathcal{A}^T)\tilde{r}^{(0)},\, \psi(\mathcal{A})\, r^{(0)}\right),$$

where $r^{(0)}$ and $\tilde{r}^{(0)}$ are two initial vectors. Similar to the CG method, we can employ an iterative scheme (the BiConjugate Gradient or BCG method):

$$\left.\begin{aligned} &\phi_0 = 1, \quad \psi_{-1} = 0, \quad \rho_{-1} = 1;\\ &\rho_m = [\phi_m \cdot \phi_m], \quad \beta_m = \rho_m/\rho_{m-1};\\ &\psi_m = \phi_m + \beta_m \psi_{m-1};\\ &\sigma_m = [\psi_m \cdot t\psi_m], \quad \alpha_m = \rho_m/\sigma_m;\\ &\phi_{m+1} = \phi_m - \alpha_m t\psi_m; \end{aligned}\right\} \quad m = 0, 1, 2, \ldots \tag{2.38}$$

method:BCG-po

This will then give

$$[\phi_i \cdot \phi_j] = \rho_i \delta_{ij}, \quad [\psi_i \cdot t\psi_j] = \sigma_i \delta_{ij}.$$

Furthermore, the BCG method results in

$$r^{(m)} = \phi_m(\mathcal{A})r^{(0)}, \quad \tilde{r}^{(m)} = \phi_m(\mathcal{A}^T)\tilde{r}^{(0)}, \quad q^{(m)} = \psi_m(\mathcal{A})r^{(0)}, \quad \tilde{q}^{(m)} = \psi_m(\mathcal{A}^T)\tilde{r}^{(0)}$$

and

$$u^{(m+1)} = u^{(m)} + \alpha_m q^{(m)}, \quad m = 0, 1, 2, \ldots$$

We notice that

$$r^{(m)} \perp \mathcal{K}_m(\mathcal{A}^T, \tilde{r}^{(0)}).$$

We also notice that (see HW 2.9):

$$r^{(m)}_{\text{BCG}} = \phi_m(\mathcal{A})r^{(0)} \perp \mathcal{K}_m(\mathcal{A}^T, \tilde{r}^{(0)}), \quad q^{(m)}_{\text{BCG}} = \psi_m(\mathcal{A})r^{(0)} \perp \mathcal{A}^T\mathcal{K}_m(\mathcal{A}^T, \tilde{r}^{(0)}). \tag{2.39}$$ eqn:BCGorth

In the BCG method, we must compute the adjoint operator $\mathcal{A}^T$. To avoid this, the Conjugate Gradient Square (CGS) method was proposed. It is based on the following observations on the iterative procedure (2.38):

$$\rho_m = [\phi_m \cdot \phi_m] = [\phi_0 \cdot \phi_m^2],$$
$$\sigma_m = [\psi_m \cdot t\psi_m] = [\psi_0 \cdot t\psi_m^2].$$

So we can define

$$r^{(m)} = \phi_m^2(\mathcal{A})\, r^{(0)}, \quad q^{(m)} = \psi_m^2(\mathcal{A})\, r^{(0)}. \tag{2.40}$$ eqn:CGS-resid

In order to compute these vectors, we can utilize (2.38) to obtain that

$$\psi_m^2 = (\phi_m + \beta_m\psi_{m-1})^2 = \phi_m^2 + 2\beta_m\phi_m\psi_{m-1} + \beta_m^2\psi_{m-1}^2,$$
$$\phi_{m+1}^2 = (\phi_m - \alpha_m t\psi_m)^2 = \phi_m^2 - 2\alpha_m t\phi_m\psi_m + \alpha_m^2 t^2\psi_m^2,$$

and

$$\phi_{m+1}\psi_m = (\phi_m - \alpha_m t\psi_m)\psi_m = \phi_m\psi_m - \alpha_m t\psi_m^2,$$
$$\phi_m\psi_m = \phi_m(\phi_m + \beta_m\psi_{m-1}) = \phi_m^2 + \beta_m\phi_m\psi_{m-1}.$$

By defining

$$p^{(m)} = \phi_m(\mathcal{A})\psi_{m-1}(\mathcal{A})\, r^{(0)}, \quad w^{(m)} = \phi_m(\mathcal{A})\psi_m(\mathcal{A})\, r^{(0)},$$

we arrive at the following iterative scheme

$$\left.\begin{aligned}
w^{(m)} &= r^{(m)} + \beta_m p^{(m)};\\
q^{(m)} &= r^{(m)} + 2\beta_m p^{(m)} + \beta_m^2 q^{(m-1)} = w^{(m)} + \beta_m\left(p^{(m)} + \beta_m q^{(m-1)}\right);\\
p^{(m+1)} &= w^{(m)} - \alpha_m\mathcal{A}q^{(m)};\\
r^{(m+1)} &= r^{(m)} - 2\alpha_m\mathcal{A}w^{(m)} + \alpha_m^2\mathcal{A}^2q^{(m)} = r^{(m)} - \alpha_m\mathcal{A}\left(w^{(m)} + p^{(m+1)}\right);\\
u^{(m+1)} &= u^{(m)} + \alpha_m\left(w^{(m)} + q^{(m)}\right);
\end{aligned}\right\} \quad m = 0, 1, \ldots \tag{2.41}$$ method:CGS-po

Remark 2.44 (CGS and BCG)**.** In the above method, we do not need to compute $\mathcal{A}^T$ any more. Theoretically speaking, we should have

$$r_{\mathrm{BCG}}^{(m)} = \phi_m(\mathcal{A})r^{(0)}, \quad r_{\mathrm{CGS}}^{(m)} = \phi_m^2(\mathcal{A})r^{(0)} = \phi_m(\mathcal{A})r_{\mathrm{BCG}}^{(m)},$$

which means CGS should converge twice as fast as BCG. However, CGS usually generates oscillating residual $\|r^{(m)}\|_0$ in practice. □

Next we will introduce the BiCGstab method, which is much more popular in practice. Consider a method who generates residuals in the following form:

$$r^{(m)} = \tilde{\phi}_m(\mathcal{A})r_{\mathrm{BCG}}^{(m)} = \tilde{\phi}_m(\mathcal{A})\phi_m(\mathcal{A})r^{(0)},$$

with a polynomial $\tilde{\phi}_m \neq \phi_m$. Usually, it can be given as

$$\tilde{\phi}_0 = 1, \quad \tilde{\phi}_{m+1}(t) = (1 - \omega_{m+1}t)\,\tilde{\phi}_m(t).$$

In this iterative procedure, we need to determine the coefficients $\{\omega_m\}_{m=1,2,\ldots}$

$$\tilde{\phi}_{m+1}\phi_{m+1} = (1 - \omega_{m+1}t)\tilde{\phi}_m\left(\phi_m - \alpha_m t\psi_m\right) = (1 - \omega_{m+1}t)\left(\tilde{\phi}_m\phi_m - \alpha_m t\tilde{\phi}_m\psi_m\right),$$

$$\tilde{\phi}_m\psi_m = \tilde{\phi}_m\left(\phi_m + \beta_m\psi_{m-1}\right) = \tilde{\phi}_m\phi_m + \beta_m(1 - \omega_m t)\tilde{\phi}_{m-1}\psi_{m-1}.$$

Motivated by (2.40), we define

$$r^{(m)} = \tilde{\phi}_m(\mathcal{A})\phi_m(\mathcal{A})\,r^{(0)}, \quad p^{(m)} = \tilde{\phi}_m(\mathcal{A})\psi_m(\mathcal{A})\,r^{(0)}. \tag{2.42}$$ eqn:BiCGstab-

In this setting, we can immediately derive that

$$\begin{aligned} p^{(m)} &= r^{(m)} + \beta_m(\mathcal{I} - \omega_m\mathcal{A})\,p^{(m-1)}, \\ r^{(m+1)} &= (\mathcal{I} - \omega_{m+1}\mathcal{A})\left(r^{(m)} - \alpha_m\mathcal{A}p^{(m)}\right) = (\mathcal{I} - \omega_{m+1}\mathcal{A})s^{(m)}, \end{aligned}$$

with $s^{(m)} := r^{(m)} - \alpha_m\mathcal{A}p^{(m)}$. We retain the freedom to choose a suitable coefficient ω_{m+1}, also known as the stabilization parameter, by minimizing

$$\|r^{(m+1)}\|_0 = \min_{\omega}\|(\mathcal{I} - \omega\mathcal{A})s^{(m)}\|_0.$$

This immediately gives

$$\omega_{m+1} := \frac{\left(s^{(m)}\right)^T\mathcal{A}\,s^{(m)}}{\left(s^{(m)}\right)^T\mathcal{A}^T\mathcal{A}\,s^{(m)}}.$$

Now, in order to make the iteration work, we only need to compute α_m and β_m. We have

$$\alpha_m = \frac{\rho_m}{\sigma_m}, \quad \beta_m = \frac{\rho_m}{\rho_{m-1}}, \quad \rho_m = [\phi_m \cdot \phi_m], \quad \sigma_m = [\psi_m \cdot t\psi_m].$$

We have the orthogonality property (2.39). For any $\psi \in \mathcal{P}_{m-1}$, the bilinear form

$$[\phi_m \cdot \psi] = \left(\psi(\mathcal{A}^T)\tilde{r}^{(0)}\right)^T \left(\phi_m(\mathcal{A})r^{(0)}\right) \equiv 0, \tag{2.43}$$ eqn:BiCGstab1

$$[\psi_m \cdot t\psi] = \left(\psi(\mathcal{A}^T)\tilde{r}^{(0)}\right)^T \mathcal{A}\left(\psi_m(\mathcal{A})r^{(0)}\right) \equiv 0. \tag{2.44}$$ eqn:BiCGstab2

Suppose that the leading term coefficients of ϕ_m and $\tilde{\phi}_m$ are ξ_m and η_m, respectively. $\xi_0 = \eta_0 = 1$. Using the iteration procedure, we get

$$\xi_{m+1} = -\alpha_m \xi_m, \quad \eta_{m+1} = -\omega_m \eta_m,$$

and ψ_m has the same leading term coefficient as ϕ_m. Hence, $\psi_m - \frac{\xi_m}{\eta_m}\tilde{\phi}_m$ and $\phi_m - \frac{\xi_m}{\eta_m}\tilde{\phi}_m$ are both $(m-1)$-polynomial in $\mathcal{P}_{m-1}$. From (2.43) and (2.44), we can easily see that

$$\rho_m = [\phi_m \cdot \phi_m] = [\phi_m \cdot \frac{\xi_m}{\eta_m}\tilde{\phi}_m], \quad \sigma_m = [\psi_m \cdot t\psi_m] = [\frac{\xi_m}{\eta_m}\tilde{\phi}_m \cdot t\psi_m]$$

We summarize the above calculations in the following lemma:

lem:bcgs2 **Lemma 2.45** (Stepsizes for BiCGstab)**.** For the BiCGstab method, we have the following identities:

$$\alpha_m = \frac{\left(\tilde{r}^{(0)}, r^{(m)}\right)}{\left(\tilde{r}^{(0)}, \mathcal{A}p^{(m)}\right)} \qquad \text{and} \qquad \beta_m = \frac{\alpha_{m-1}}{\omega_m}\frac{\left(\tilde{r}^{(0)}, r^{(m)}\right)}{\left(\tilde{r}^{(0)}, r^{(m-1)}\right)}.$$

The BiCGstab algorithm is given in the following listing. Compared to the BCG and CGS methods, BiCGstab is known for its superior stability. It avoids computing the transpose operator $\mathcal{A}^T$, is based on short recurrence, and only requires two matrix-vector inner products in each iteration.

Listing 2.8: BiCGstab method

```
%% Given an initial guess u and a tolerance ε;
r ← f − Au,  p ← r;
Choose a fixed vector r̃_0 such that ρ ← (r̃_0, r) ≠ 0
while ‖r‖ > ε
    w ← Ap,  σ ← (r̃_0, w),  α ← ρ/σ;
    s ← r − α w,  v ← As,  ω ← (s, v)/(v, v);
    u ← u + α p + ω s;
    r ← s − ω v;
    ρ̃ ← (r̃_0, r),  β ← (α ρ̃)/(ω ρ);
    p ← r + β (p − ω w);
    Update: ρ ← ρ̃;
end
```

2.2.7 Generalizing KSM to Hilbert spaces

The Krylov subspace methods (KSMs) can also be extended to linear operators $\mathcal{A} : \mathscr{V} \mapsto \mathscr{V}$, where $\mathscr{V}$ is a separable Hilbert space[5]. It is important to note that the convergence estimates (2.33) and (2.37) do not depend on the finite dimensionality N. Whereas the KSM framework is deeply understood for finite-dimensional problems, it is instead less explored and lacks a systematic study in the infinite-dimensional case.

In view of Remark 1.23, we assume

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

and the inf-sup condition (1.16) holds

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

Hence the condition number $\kappa(\mathcal{A}) \leqslant C_a/\alpha$ is bounded. This discussion follows the work by Mardal and Winther [136] closely.

Theorem 2.46 (Convergence of KSM in Hilbert spaces)**.** Let $\mathcal{A} : \mathscr{V} \mapsto \mathscr{V}$ be a symmetric isomorphism. The minimum residual method satisfies the following estimate:

$$\left\|\mathcal{A}(u - u^{(m)})\right\| \leqslant 2\delta^m \left\|\mathcal{A}(u - u^{(0)})\right\|, \tag{2.45}$$

where $0 < \delta < 1$ only depends on $\kappa(\mathcal{A})$. Moreover, if $\mathcal{A}$ is positive-definite, then the conjugate gradient method satisfies that

$$\left\|u - u^{(m)}\right\|_{\mathcal{A}} \leqslant 2\delta^m \left\|u - u^{(0)}\right\|_{\mathcal{A}}, \tag{2.46}$$

where $\delta = (\sqrt{\kappa(\mathcal{A})} - 1)/(\sqrt{\kappa(\mathcal{A})} + 1)$.

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 $\mathscr{W} = \mathscr{V}'$. 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[6]:

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

[5] Note that $\mathscr{V}$ here might not be finite dimensional.

A symmetric isomorphism $\mathcal{A} \in \mathscr{L}(\mathscr{V}; \mathscr{V}')$ satisfies that

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

where $\langle \cdot, \cdot \rangle$ is the duality pair for $\mathscr{V}'$ and $\mathscr{V}$. 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 preconditioned system

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

and $\mathcal{B}\mathcal{A}$ 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. From the convergence estimate (2.45), we have

$$\langle \mathcal{A}(u - u^{(m)}), \mathcal{B}\mathcal{A}(u - u^{(m)})\rangle^{1/2} \leqslant 2\delta^m \langle \mathcal{A}(u - u^{(0)}), \mathcal{B}\mathcal{A}(u - u^{(0)})\rangle^{1/2},$$

with δ depends on $\kappa(\mathcal{B}\mathcal{A})$ only.

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

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

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

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

and the inf-sup condition (1.16) gives

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

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

Example 2.47 (Poisson solver as a preconditioner)**.** As an example, we consider a second-order elliptic operator $\mathcal{A} : H_0^1(\Omega) \mapsto H^{-1}(\Omega)$ defined by

$$\langle \mathcal{A}u, v\rangle = a[u, v] := \int_\Omega (\mu(x)\nabla u) \cdot \nabla v dx.$$

Assume that $\mu(x) \in \mathbb{R}^{d\times d}$ is a symmetric and uniformly positive definite

$$c|\xi|^2 \leqslant \xi^T \mu(x)\xi \leqslant C|\xi|^2, \quad x \in \Omega,\ \xi \in \mathbb{R}^d.$$

We need to define

$$\big(\mathcal{B}f, v\big)_{H_0^1(\Omega)} := \big(\nabla(\mathcal{B}f), \nabla v\big)_{0,\Omega} = \langle f, v\rangle.$$

In this sense, we can choose $\mathcal{B} = (-\Delta)^{-1}$ as a preconditioner and the condition number of $\mathcal{BA} : H_0^1(\Omega) \mapsto H_0^1(\Omega)$ is bounded, i.e.

$$\kappa(\mathcal{BA}) = \|\mathcal{BA}\|_{\mathscr{L}(H_0^1(\Omega);H_0^1(\Omega))}\|(\mathcal{BA})^{-1}\|_{\mathscr{L}(H_0^1(\Omega);H_0^1(\Omega))} \leqslant \frac{C}{c}.$$

We note that other inner products can be used, which will yield different preconditioners. As long as the continuity condition and the inf-sup condition hold, the preconditioned system is well-conditioned. □

Now we summarize the above discussions 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 Preconditioning techniques

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:

$\mathcal{BA}u = \mathcal{B}f$		Left preconditioning
$\mathcal{AB}v = f$	$u = \mathcal{B}v$	Right preconditioning
$\mathcal{B}_L\mathcal{AB}_Rv = \mathcal{B}_Lf$	$u = \mathcal{B}_Rv$	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 $\kappa(\mathcal{BA}) \ll \kappa(\mathcal{A})$, if we apply a Krylov subspace method to solve a preconditioned linear system.

2.3.1 Construction of preconditioners

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

lem:condnum **Lemma 2.48** (Estimation of condition number)**.** If μ_0 and μ_1 are positive constants satisfying

$$\mu_0(\mathcal{A}u,u) \leqslant (\mathcal{B}^{-1}u,u) \leqslant \mu_1(\mathcal{A}u,u), \quad \forall u \in V, \tag{2.47}$$ eqn:condnum1

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, $\sigma\big((\mathcal{B}\mathcal{A})^{-1}\big) \subseteq \big[\mu_0,\mu_1\big]$. □

Using equivalent conditions found in (2.47) can often provide more convenient ways to analyze the condition number, as demonstrated in the following lemma and remark. The proof of this lemma has been left as an exercise for the reader and can be found in HW 2.10.

equiv-condnum **Lemma 2.49** (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 (2.47) are equivalent to

$$\mu_0(\mathcal{B}u,u) \leqslant (\mathcal{A}^{-1}u,u) \leqslant \mu_1(\mathcal{B}u,u), \quad \forall u \in V, \tag{2.48}$$ eqn:condnum2

or

$$\mu_1^{-1}(\mathcal{A}u,u) \leqslant (\mathcal{A}\mathcal{B}\mathcal{A}u,u) \leqslant \mu_0^{-1}(\mathcal{A}u,u), \quad \forall u \in V, \tag{2.49}$$ eqn:condnum3

or

$$\mu_1^{-1}(\mathcal{B}u,u) \leqslant (\mathcal{B}\mathcal{A}\mathcal{B}u,u) \leqslant \mu_0^{-1}(\mathcal{B}u,u), \quad \forall u \in V. \tag{2.50}$$ eqn:condnum4

rem:AmABA **Remark 2.50** (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:

$$-\alpha(\mathcal{A}u,u) \leqslant \big(\mathcal{A}(\mathcal{I}-\mathcal{B}\mathcal{A})u,u\big) \leqslant \delta(\mathcal{A}u,u), \quad \forall u \in V \tag{2.51}$$ eqn:condnum5

and

$$(1+\alpha)^{-1}(\mathcal{A}u,u) \leqslant (\mathcal{B}^{-1}u,u) \leqslant (1-\delta)^{-1}(\mathcal{A}u,u), \quad \forall u \in V. \tag{2.52}$$ eqn:condnum6

Apparently, (2.51) can also be written as

$$-\alpha(u,u)_{\mathcal{A}} \leqslant \big((\mathcal{I}-\mathcal{B}\mathcal{A})u,u\big)_{\mathcal{A}} \leqslant \delta(u,u)_{\mathcal{A}}, \quad \forall u \in V. \tag{2.53}$$ eqn:condnum7

This equivalent relation will be useful later on. □

2.3.2 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. $(\cdot,\cdot)_{\mathcal{A}}$ in §2.2.7. 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 this new inner product.

lem:pcg2 **Lemma 2.51** (Stepsizes of PCG)**.** For the preconditioned conjugate gradient method, we have the following identities:

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

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 2.9: 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
```

2.3.3 Precondition 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, let $\rho_0 := \rho(\mathcal{I} - \mathcal{B}\mathcal{A}) = \|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}$. The method is not only converging but also a contraction, i.e., $\|u - u^{(m)}\|_{\mathcal{A}} \leqslant \rho_0^m \|u - u^{(0)}\|_{\mathcal{A}} \to 0$ as $m \to +\infty$. Similar argument as Theorem 2.13 shows that $\mathcal{B}$ must be SPD. Furthermore, by definition of $\|\cdot\|_{\mathcal{A}}$, we have

$$\Big(\big(\mathcal{A} - 2\mathcal{A}\mathcal{B}\mathcal{A} + \mathcal{A}\mathcal{B}\mathcal{A}\mathcal{B}\mathcal{A}\big)u, u\Big) \leqslant \rho_0^2 (u,u)_{\mathcal{A}}.$$

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

$$\begin{aligned}\left((\mathcal{I} - \mathcal{A}^{1/2}\mathcal{B}\mathcal{A}^{1/2})^2 v, v\right) \leqslant \rho_0^2(v,v) \quad &\Longrightarrow \quad \left|((\mathcal{I} - \mathcal{A}^{1/2}\mathcal{B}\mathcal{A}^{1/2})v, v)\right| \leqslant \rho_0(v,v) \\ &\Longrightarrow \quad \left|((\mathcal{A} - \mathcal{A}\mathcal{B}\mathcal{A})u, u)\right| \leqslant \rho_0(\mathcal{A}u, u), \quad \forall u \in V.\end{aligned}$$

Hence Remark 2.50 shows (see HW 2.12) that the condition number is uniformly bounded, i.e.,

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant \frac{1+\rho_0}{1-\rho_0}.$$

In fact, the above estimate can also be easily obtained from $\rho(\mathcal{I} - \mathcal{B}\mathcal{A}) = \rho_0 < 1$.

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 (2.33) is equal to

$$\delta_{\mathrm{CG}} := \frac{\sqrt{\kappa(\mathcal{B}\mathcal{A})} - 1}{\sqrt{\kappa(\mathcal{B}\mathcal{A})} + 1} \leqslant \frac{\sqrt{\frac{1+\rho_0}{1-\rho_0}} - 1}{\sqrt{\frac{1+\rho_0}{1-\rho_0}} + 1} = \frac{1 - \sqrt{1-\rho_0^2}}{\rho_0} < \rho_0. \tag{2.54}$$

eqn:IterPreco

The last inequality holds true when $0 < \rho_0 < 1$.

The discussion above suggests that a preconditioner can be found for any convergent stationary linear iterative method, and its convergence can be accelerated by PCG. However, this comes at an additional cost of applying the preconditioners. Preconditioning is crucial for practical problems, and KSMs are sometimes referred to as accelerators. It is desirable to have an effective preconditioner which satisfy most, if not all, of the following properties:

- The preconditioned linear systems have improved "condition" of the linear system.
- The spectral condition number of $\mathcal{B}\mathcal{A}$ should be better 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.

To summarize, we present numerical results (Figure 2.2 and Figure 2.3) comparing standard and preconditioned Krylov subspace methods for solving the finite difference system that arises from the 2D Poisson's equation on a unit square domain with homogeneous Dirichlet boundary conditions. The problem size for this study is 16384×16384.

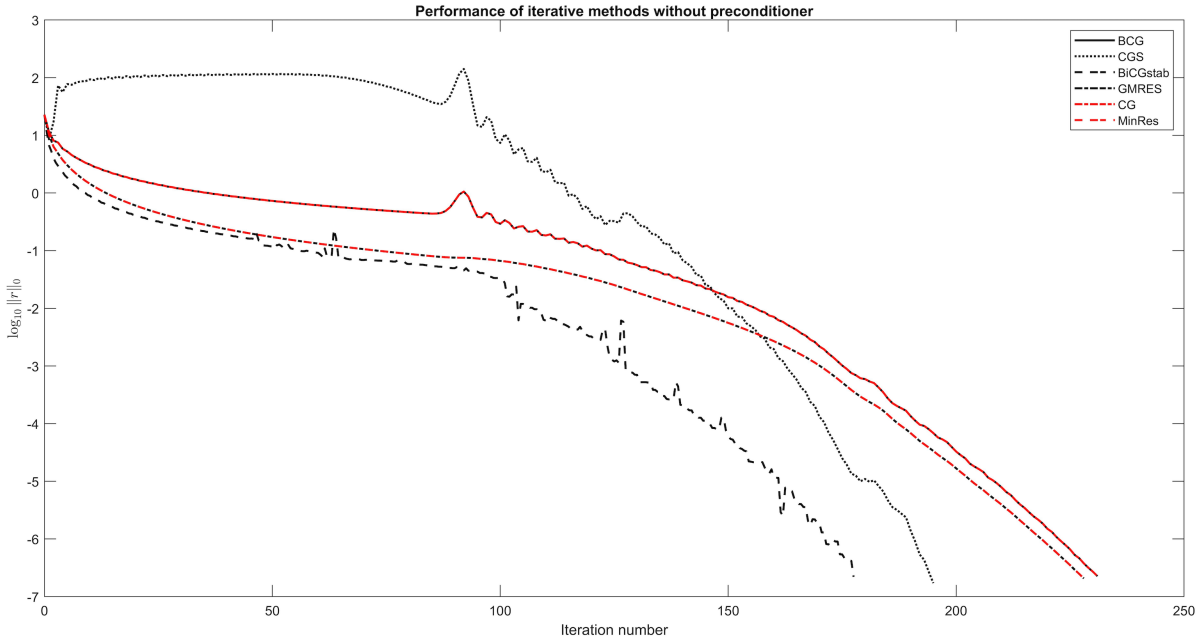


fig:KSM1

Figure 2.2: Solving FD equation using Krylov subspace methods without preconditioning.

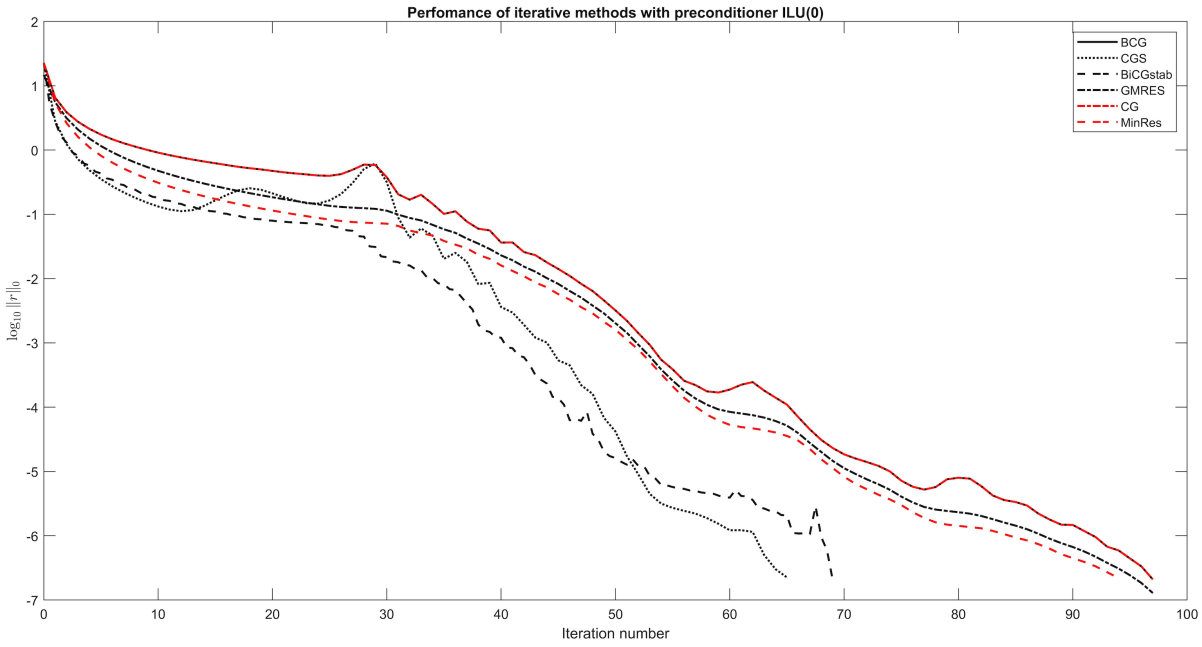


fig:KSM2

Figure 2.3: Solving FD equation using Krylov subspace methods with ILU(0) preconditioning.

2.3.4 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

$$\frac{\|u - u^{(m)}\|}{\|u\|} \leqslant \kappa(\mathcal{A}) \frac{\|r^{(m)}\|}{\|f\|}. \tag{2.55}$$ eqn:ResErr

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 (2.55).

We notice that the right-hand side of (2.55) 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 2.49, $\left(r^{(m)}, r^{(m)}\right)_{\mathcal{B}}^{\frac{1}{2}}$ is equivalent to $\left(e^{(m)}, e^{(m)}\right)_{\mathcal{A}}^{\frac{1}{2}}$, if $\mathcal{B}$ 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 the left 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 $\|r\|$. Sometimes it might not be a good idea to use $r_{\mathcal{B}}$ instead of r.

2.4 Domain decomposition methods

sec:DDM

Domain decomposition methods (DDMs) are commonly used in numerical methods for partial differential equations (PDEs) as they employ efficient "divide and conquer" techniques to iteratively solve sub-problems defined on smaller subdomains. DDMs provide a convenient framework for solving and preconditioning heterogeneous or multiphysics problems, regardless of the discretization method used (e.g., FD and FE). These methods offer significant benefits for solving problems on parallel computers, particularly when it comes to their algebraic solution. Although there are two ways of subdividing the computational domain, overlapping and non-overlapping, overlapping domain decomposition methods are the focus of our discussion.

2.4.1 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.

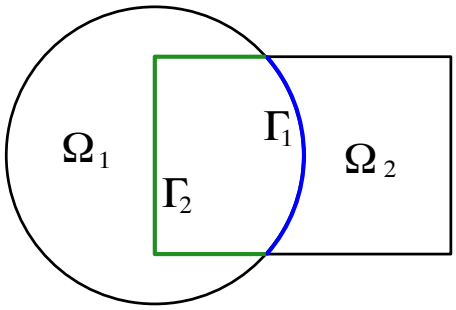


fig:ddm

Figure 2.4: 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

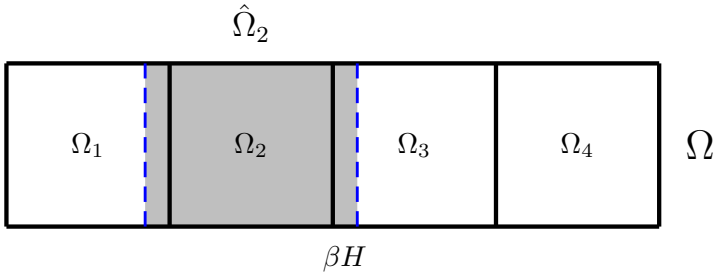
$$u^{(m+1)}(x) := \begin{cases} u_2^{(m+1)}, & \text{if } x \in \Omega_2; \\ u_1^{(m+1)}, & \text{if } x \in \Omega \backslash \Omega_2. \end{cases}$$

2.4.2 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 2.5. Of course, we can use more general partitioning strategies as well.



g:overlapping

Figure 2.5: 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

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

Let $\vec{v}_i \in \mathbb{R}^{N_i}$. The injection matrix (natural embedding) $I_i \in \mathbb{R}^{N \times N_i}$ is defined as

$$(I_i \vec{v}_i)_k = \begin{cases} (\vec{v}_i)_k, & \text{if } k \in \hat{G}_i; \\ 0, & \text{if } k \in G \backslash \hat{G}_i. \end{cases} \tag{2.56}$$

eqn:restricti

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

$$B_{\text{as}} := \sum_{i=1}^{n} B_i = \sum_{i=1}^{n} I_i A_i^{-1} I_i^T, \tag{2.57}$$

eqn:ASM

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

$$I - B_{\mathrm{ms}}A := (I - B_nA)\cdots(I - B_1A) = \prod_{i=n}^{1}(I - B_iA). \tag{2.58}$$ eqn:MSM

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

2.4.3 Convergence of overlapping DDMs ⋆

Domain decomposition methods, particularly the ASM version, are typically employed as preconditioners for parallel computing. Convergence has been analyzed in [82, 83], and we will showcase results for the additive version.

Theorem 2.52 (Effect of DD preconditioner)**.** The condition number of AS domain decomposition method is independent of the mesh size h and satisfies

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

where H is size of domain partitions, βH characterizes size of the overlaps, and C is a constant independent of mesh sizes.

The DD preconditioner (2.57) 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.

$$B_{\mathrm{as},2} := I_0A_0^{-1}I_0^T + \sum_{i=1}^{n} I_iA_i^{-1}I_i^T,$$

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

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

where the constant is independent of mesh sizes.

The theorem above demonstrates that a suitable coarse-level correction can eliminate mesh size dependence. In the subsequent chapters, we will create and evaluate two-level and multilevel iterative methods.

2.5 Homework problems

hw:BtBstar **HW 2.1.** Show the identity (2.7).

w:A-transpose **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:SDalpha **HW 2.3.** Show the optimal stepsize (2.20) for general descent direction method.

hw:CHThm **HW 2.4.** Let $A \in \mathbb{R}^{N\times N}$ and $q(\lambda) := |\lambda I - A|$ be the characteristic polynomial of A. Show the Cayley–Hamilton theorem, i.e., every square matrix A satisfies its own characteristic polynomial $q(A) = 0$.

hw:Krylov **HW 2.5.** Given $A \in \mathbb{R}^{N\times N}$ and $0 \neq v \in \mathbb{R}^N$. Then the Krylov subspace satisfies the following properties:

1. $\mathcal{K}_m(A, v) \subset \mathcal{K}_{m+1}(A, v)$;
2. $A\mathcal{K}_m(A, v) \subset \mathcal{K}_{m+1}(A, v)$;
3. $\mathcal{K}_m(A, v) = \mathcal{K}_m(\alpha A, v) = \mathcal{K}_m(A, \alpha v)$, for any $0 \neq \alpha \in \mathbb{R}$;
4. $\mathcal{K}_m(A, v) = \mathcal{K}_m(A - \alpha I, v)$, for any $\alpha \in \mathbb{R}$;
5. $\mathcal{K}_m(Q^{-1}AQ, Q^{-1}v) = Q^{-1}\mathcal{K}_m(A, v)$, for any nonsingular matrix $Q \in \mathbb{R}^{N\times N}$;
6. $\mathcal{K}_m(A, v) = \{p(A)v : p \in \mathcal{P}_{m-1}\}$, where $\mathcal{P}_{m-1}$ is the real polynomial of degree less than m.

hw:CG-Krylov **HW 2.6.** Suppose that $\mathcal{A}$ is SPD and $\mathcal{A}u = f$. Show that the following conditions are equivalent to each other:

1. Vector $u_m \in \mathcal{K}_m(\mathcal{A}, f)$ satisfies that $\|u_m - u\|_{\mathcal{A}} = \min\{\|v - u\|_{\mathcal{A}} : v \in \mathcal{K}_m(\mathcal{A}, f)\}$;
2. Vector $u_m \in \mathcal{K}_m(\mathcal{A}, f)$ satisfies that $\|f - \mathcal{A}u_m\|_{\mathcal{A}^{-1}} = \min\{\|f - \mathcal{A}v\|_{\mathcal{A}^{-1}} : v \in \mathcal{K}_m(\mathcal{A}, f)\}$;
3. Vector $u_m \in \mathcal{K}_m(\mathcal{A}, f)$ satisfies that $v^T(f - \mathcal{A}u_m) = 0$, for any $v \in \mathcal{K}_m(\mathcal{A}, f)$.

gateDirections **HW 2.7.** Prove Lemmas 2.34 and 2.35.

hw:Cheb **HW 2.8.** 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

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

and $\left|T_n(x)\right| \leqslant 1$ 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

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

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

hw:BCG **HW 2.9.** In the BCG method, we have the following properties:

1. $\big(\tilde{r}^{(i)}, r^{(j)}\big) = \rho_i \delta_{ij}$;
2. $\big(\tilde{q}^{(i)}, \mathcal{A}q^{(j)}\big) = \sigma_i \delta_{ij}$;
3. $\mathcal{K}_m(\mathcal{A}, r^{(0)}) = \operatorname{span}\{q^{(0)}, q^{(1)}, \ldots, q^{(m-1)}\} = \operatorname{span}\{r^{(0)}, r^{(1)}, \ldots, r^{(m-1)}\}$;
4. $\mathcal{K}_m(\mathcal{A}^T, \tilde{r}^{(0)}) = \operatorname{span}\{\tilde{q}^{(0)}, \tilde{q}^{(1)}, \ldots, \tilde{q}^{(m-1)}\} = \operatorname{span}\{\tilde{r}^{(0)}, \tilde{r}^{(1)}, \ldots, \tilde{r}^{(m-1)}\}$.

equiv-condnum **HW 2.10.** Prove Lemma 2.49.

quiv-condnum2 **HW 2.11.** Show that (2.51) and (2.52) are equivalent to each other.

hw:condBA **HW 2.12.** 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

Two-grid Methods

ch:two-grid

In the preceding chapter, we explored a variety of iterative solvers and preconditioners for solving the linear algebraic system presented in equation (2.1). However, as the meshsize h approaches zero, the convergence rate of these methods tends to deteriorate, with the exception of the two-level overlapping domain decomposition method that incorporates coarse-grid correction. This observation serves as the foundation for our subsequent discussions on multilevel iterative methods. In this chapter, we will delve into the two-grid (or more generally, two-level) method for solving the discrete Poisson's equation:

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

3.1 Finite element methods

sec:FEM

In Chapter 1, we briefly explored the finite element approximation for this model problem. The Finite Element Method (FEM) is a type of Galerkin method that leverages piecewise polynomial spaces for the approximate test and trial function spaces. For more information on the standard FEM, readers are encouraged to refer to [70, 116, 37, 59]. These sources provide a detailed discussion of FEM construction and error analysis. From this point forward, our discussions will primarily focus on finite element discretizations.

Throughout this chapter, we will utilize the standard notations for Sobolev spaces. That is, $H^k(\Omega)$ represents the classical Sobolev space of scalar functions on a bounded domain $\Omega \subset \mathbb{R}^d$, with derivatives up to order k that are square integrable. The full norm of this space is denoted by $\|\cdot\|_k$, which is accompanied by the corresponding semi-norm $|\cdot|_k$. Additionally, 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 as they relate to the subdomain of Ω.

We now take a little detour and say a few more words about the finite element discretizations; see [59] for more details. The linear operator $\mathcal{A} : \mathscr{V} \mapsto \mathscr{V}'$ is defined by

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

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.,

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

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

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

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

3.1.1 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:

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

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

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

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.

rem:GO **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.,

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

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*. □

lem:Cea **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.

prop:optimal **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 (3.1) 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 [201]. □

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

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

□

3.1.2 Finite element ⋆

The Finite Element Method (FEM) has a rich history of practical use and is widely applied to various problems in physics and engineering. It has proven to be successful in numerous fields, including structural mechanics. After decades of extensive development, the classical (conforming) finite element method has become a well-understood and flourishing area in scientific computation. One of the most attractive features of FEM is its ability to handle complex geometries, boundaries, and operators with relative ease.

def:fe **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*.

f:nodal_basis **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

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

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.

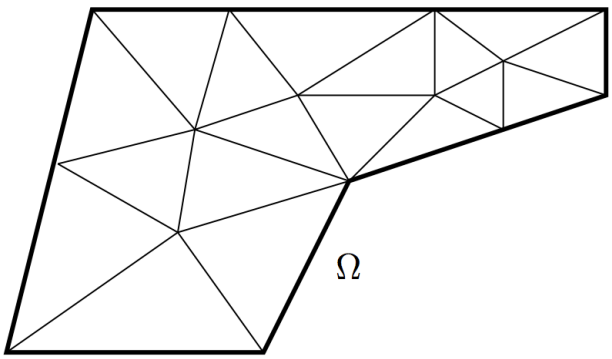


fig:domain

Figure 3.1: A polygonal domain Ω with conforming partition.

rem:VEF **Remark 3.10** (Number of geometric entities)**.** Let us now briefly discuss the relationships among the numbers of vertices, edges, faces, and elements in a triangular or tetrahedral partition. We 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 *half-edge*, which is defined as a pair consisting of an edge and a face that 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 $\#\mathsf{V} - \#\mathsf{E} + \#\mathsf{F} = 2$. Hence, we can obtain that $\#\mathsf{F} \approx 2\,\#\mathsf{V}$ and $\#\mathsf{E} \approx 3\,\#\mathsf{V}$. In 3D, we have the following asymptotic relations: $\#\mathsf{F} \approx 12\,\#\mathsf{V}$, $\#\mathsf{E} \approx 7\,\#\mathsf{V}$, 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.,

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

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

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

or, equivalently denoted in an operator form,

$$\mathcal{A}_h u_h = f_h \quad \text{in } V_h. \tag{3.6}$$ eq:ell_s

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

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

We are going to discuss this notation in §3.2.

rem:sim **Remark 3.11** (A very useful notation)**.** In our discussion, we adopt the notations introduced by Xu [196]. The symbol $a \lesssim b$ signifies that there exists a constant C that is independent of the mesh size h, satisfying $a \leqslant Cb$. Similarly, the notations "$\gtrsim$" and "$\cong$" can be defined. These notations are significant since, in our future discourse, we aim to devise solvers and preconditioners that result in convergence rates that are independent of the mesh size h. □

3.1.3 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.

prop:interp **Proposition 3.12** (Interpolation error)**.** Let $\mathcal{M}_h$ be a uniform mesh and V_h be a C^α ($\alpha \geqslant 0$) finite element space on $\mathcal{M}_h$. The interpolant $\mathcal{J}_h : W_p^m(\Omega) \mapsto V_h$ satisfies

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

prop:inverse **Proposition 3.13** (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

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

Using Proposition 3.13, 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.

nfty-estimate **Proposition 3.14** (Discrete Sobolev inequality [50])**.** 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 $C_3(h) = h^{-\frac{1}{2}}$.

prop:L2proj **Proposition 3.15** (Weighted estimate for L^2 projection [50])**.** 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

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

rem:SimEst **Remark 3.16** (Simultaneous estimate)**.** From the above weighted L^2-estimate, we can easily show the so-called simultaneous estimate

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

□

rem:rho-FE **Remark 3.17** (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 $\kappa(\mathcal{A}_h) \cong h^{-2}$. □

3.1.4 Error analysis ⋆

We now briefly introduce standard error estimates for the continuous linear finite element; see [70, 59] 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.,

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

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

thm:H1error **Theorem 3.18** (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 $\|u - u_h\|_{1,\Omega} \lesssim h\|f\|_{0,\Omega}$.

By applying the well-known duality argument, we have the L^2 error estimate; see [59, Theorem 5.4.8] for example.

thm:L2error **Theorem 3.19** (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 3.20 (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)

$$\|\!|f-\mathcal{A}u_h|\!\|_*\lesssim|\!|\!|u-u_h|\!|\!|\lesssim\|\!|f-\mathcal{A}u_h|\!\|_*\,. \tag{3.9}$$ eqn:err-res-e

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 Algebraic representations

ec:matrixform

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 $\mathcal{A}_hu_h=f_h$ 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.

3.2.1 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

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

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

$$\vec{v}:=\begin{pmatrix}(v,\phi_1)\\ (v,\phi_2)\\ \vdots\\ (v,\phi_N)\end{pmatrix}\qquad\text{and}\qquad\vec{v}=M\underline{v}, \tag{3.11}$$ eqn:v-matrix2

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

$$(\underline{u}, \vec{v}) \equiv (\underline{u}, \vec{v})_{l^2} = (\underline{u}, \vec{v})_0 = \underline{u}^T M \underline{v} = (u, v)_V.$$

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 a 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 $\sum_{i=1}^{N'} \left(\underline{\mathcal{A}}\right)_{i,j} \psi_i = \mathcal{A}\phi_j \ \ (j = 1, \dots, N)$, i.e.,

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

On the other hand, the dual representation (the *stiffness matrix*) corresponding to $\mathcal{A} : V \mapsto V$ 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.

lem:mat-form **Lemma 3.21** (Algebraic 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. $(u, v) = (M\underline{u}, \underline{v})$.

ex:identity **Example 3.22** (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 useful equality

$$\underline{\mathcal{A}}^{-1} = \underline{\mathcal{A}^{-1}}. \tag{3.13}$$ eqn:MatFormIn

That is to say, the inverse matrix of the primal form of $\mathcal{A}$ is the primal form of the inverse of operator $\mathcal{A}$. □

Example 3.23 (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 $\phi_i = \vec{e}_i := (0, \dots, 1, \dots, 0)^T \in \mathbb{R}^N$, 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 $\hat{\mathcal{A}} = \underline{\mathcal{A}}$. □

3.2.2 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 $\{\phi_i\}_{i=1,\dots,N}$ 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 $\underline{u} = (u_i)_{i=1}^N \in \mathbb{R}^N$ be the vector of coefficients of u_h, namely $\underline{u_h}$. Let $\vec{f} = (f_i)_{i=1}^N := \{\langle f, \phi_i\rangle\}_{i=1}^N$. 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.

rem:SpecMass **Remark 3.24** (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

$$(M\underline{v}, \underline{v}) = \sum_{i,j} \underline{v}_i\, \underline{v}_j\, (\phi_i, \phi_j) = (v, v) = \int_\Omega v^2(x)\, dx \cong h^d \sum_i \underline{v}_i^2 \cong h^d(\underline{v}, \underline{v}). \tag{3.14}$$ eqn:spec-M

It is consistent with the well-known facts that the mass matrix is SPD and well-conditioned, i.e.,

$$h^d\|\xi\|_0^2 \lesssim \xi^T M\, \xi \lesssim h^d\|\xi\|_0^2, \quad \forall\, \xi \in \mathbb{R}^N.$$

□

rem:SpecStiff **Remark 3.25** (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 3.17,

$$h^d\|\xi\|_0^2 \lesssim \xi^T A\, \xi \lesssim h^{d-2}\|\xi\|_0^2, \quad \forall\, \xi \in \mathbb{R}^N.$$

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. □

3.2.3 Algebraic 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:

$$\underline{u}^{\text{new}} = \underline{u}^{\text{old}} + \omega\Big(\vec{f} - \hat{\mathcal{A}}\,\underline{u}^{\text{old}}\Big). \tag{3.15}$$ eqn:Richardso

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

$$\mathcal{B}_\omega v := \omega \sum_{i=1}^{N} (v, \phi_i)\phi_i, \quad \forall v \in V \quad \Longleftrightarrow \quad \underline{\mathcal{B}_\omega} = \omega M. \tag{3.16}$$ eqn:Richardso

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

$$\underline{u}^{\text{new}} = \underline{u}^{\text{old}} + \underline{\mathcal{S}}(M^{-1}\vec{f} - M^{-1}\hat{\mathcal{A}}\,\underline{u}^{\text{old}}) = \underline{u}^{\text{old}} + \underline{\mathcal{S}}M^{-1}(\vec{f} - \hat{\mathcal{A}}\,\underline{u}^{\text{old}}). \tag{3.17}$$ eqn:MatS

The above equality indicates that, we shall define a smoother in the matrix form as

$$S := \underline{\mathcal{S}}M^{-1}, \quad \text{i.e.,} \quad \underline{\mathcal{S}} = SM. \tag{3.18}$$ eqn:MatRepSmo

Example 3.26 (Matrix form of the Richardson iteration)**.** If we consider the above Richardson method (3.16) 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 (3.15). □

We will now discuss another important concept in our future analysis: the matrix form of symmetrization. Let $w := \mathcal{S}^T u$. Then we have

$$\vec{w} = \Big((\mathcal{S}^T u, \phi_i)\Big)_{i=1}^N = \Big(\sum_j \underline{u}_j(\mathcal{S}^T\phi_j, \phi_i)\Big)_{i=1}^N = \Big(\sum_j \underline{u}_j(\phi_j, \mathcal{S}\phi_i)\Big)_{i=1}^N = (\hat{\mathcal{S}})^T\underline{u}.$$

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

$$\underline{\mathcal{S}^T} = M^{-1}\big(M\underline{\mathcal{S}}\big)^T = M^{-1}\underline{\mathcal{S}}^T M = S^T M. \tag{3.19}$$ eqn:MatST

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 (3.13) in Example 3.22, it is easy to see that

$$\underline{\mathcal{S}^{-1}} = \big(\underline{\mathcal{S}}\big)^{-1} = \big(SM\big)^{-1} = M^{-1}S^{-1}. \tag{3.20}$$ eqn:MatSinv

Using the definition of symmetrized operator (2.13) and (3.18)–(3.20), we can define the matrix form of the symmetrization

$$\begin{aligned} \overline{S} \;:=\; \overline{\underline{\mathcal{S}}}M^{-1} &= \underline{\mathcal{S}^T(\mathcal{S}^{-T}+\mathcal{S}^{-1}-\mathcal{A})\mathcal{S}}\,M^{-1} \\ &= S^T M\big(M^{-1}S^{-T}+M^{-1}S^{-1}-M^{-1}\hat{\mathcal{A}}\big)SMM^{-1} \\ &= S^T\big(S^{-T}+S^{-1}-A\big)S, \end{aligned} \tag{3.21}$$

which is formally consistent with the definition of symmetrization (2.13).

3.3 Smoothers and smoothing effect

sec:smoother

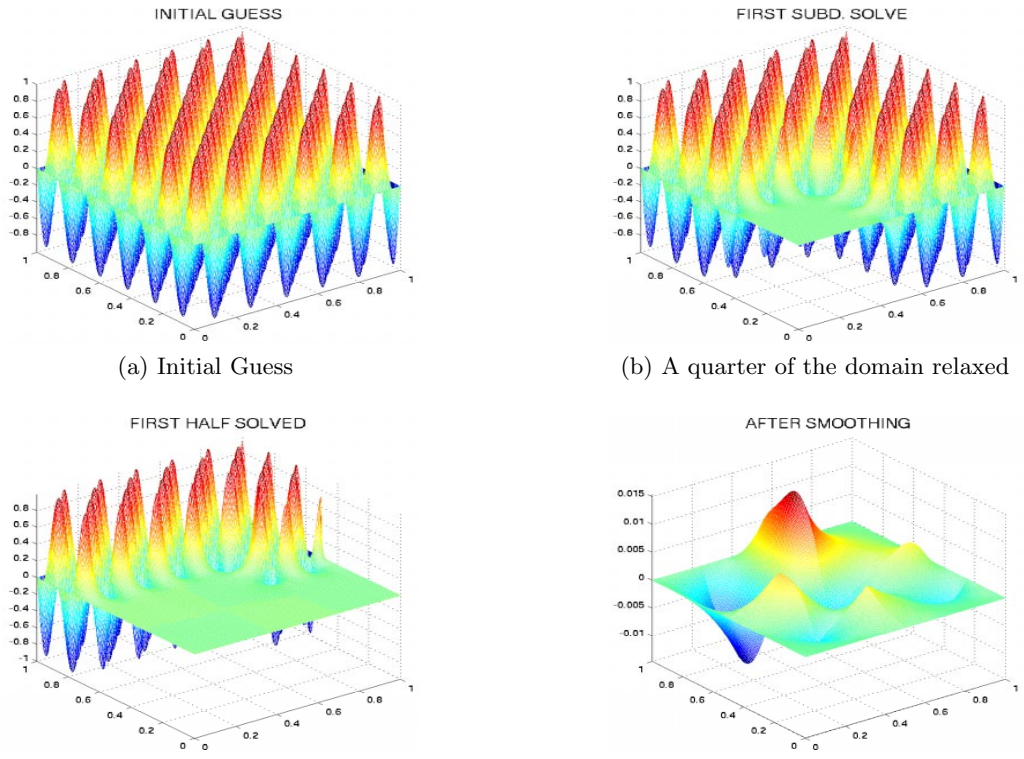
The stationary linear iterative methods discussed so far, including the weighted Jacobi and Gauss-Seidel methods, are commonly referred to as "local relaxation" techniques. This term reflects the fact that these methods address the residual vector locally, typically correcting one variable at a time, as shown in Example 2.26. Although not particularly efficient as standalone solvers, they are highly effective at reducing high-frequency error components (see §1.3) and remain essential in modern multilevel iterative techniques. Other methods, such as SOR and incomplete factorizations, yield similar results. In this section, we will explore the smoothing effects of these methods through various analytical approaches.

3.3.1 A numerical example

The weighted Jacobi and Gauss-Seidel methods are frequently referred to as "local relaxations," as they are particularly useful for addressing error components that are localized in nature. Consequently, it is not unexpected that both of these methods can effectively damp out

non-smooth components. However, given that relatively smoother error components tend to be more globally related, both methods are comparatively inefficient at addressing them.

It has been observed that basic stationary linear iterative schemes tend to converge quickly in the initial stages, followed by a slowdown after a few steps (see Figure 1.4 for the convergence behavior of the weighted Jacobi method). It is noteworthy that these methods not only converge rapidly in the first few steps, but also exhibit the ability to smooth out the error function rather swiftly. In other words, the error function becomes a much smoother function after only a few iterations. This property of an iterative scheme is known as the "smoothing property," and any iterative scheme possessing this trait is referred to as a "smoother."



(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.

basiciteration

In Figure 3.2, we see a visual representation of the multiplicative overlapping domain decomposition method using four subdomains. After a single iteration, this method is capable of smoothing out the high frequency portion and retaining the lower frequency portion. Basic linear relaxation schemes, such as the Richardson, Jacobi, and Gauss-Seidel iterations, are inherently limited in their scope as local methods. As a result, they are only able to capture the high frequency (local) portion of the error, and do not perform well on low frequency (global) components.

3.3.2 Local Fourier analysis ⋆

Local Fourier analysis (LFA), also known as local mode analysis, is a powerful technique for comprehending and forecasting the convergence rate of geometric multigrid (GMG) methods. Classical LFA uses error expansion in terms of the eigenvectors of a discrete differential operator without considering boundary conditions, followed by study of the behavior of the multigrid error propagation operator when acting on these components. LFA has been proven to yield an accurate convergence rate for GMG on model problems with periodic boundary conditions on rectangular domains. Further discussions can be found in [54, 185, 193]. More recently, LFA has been demonstrated to be applicable to more practical scenarios, such as those involving Dirichlet boundary conditions. Specifically, if a problem is compatible to a periodic boundary problem, LFA can provide an accurate convergence rate for multigrid schemes, as was proven in [167].

LFA has been developed for geometric multigrid algorithms for a wide range of problems, including those with non-constant or non-linear coefficients. The technique can be applied to various discretization methods, such as finite difference or finite volume methods, provided that the resulting discrete problems can be represented in a stencil form. However, its applicability to finite element methods is limited, owing to the fact that their grids are typically unstructured. In this context, we will use LFA to analyze simple smoothers, with the aim of providing readers with a glimpse of this powerful tool. For more comprehensive discussions, readers may refer to the practical guide on LFA written by Wienands and Joppich [193].

In order to analyze the local behavior of iterative methods, we consider the 2D Poisson's equation; see §1.2. We begin with the weighted (damped) Jacobi method as an example. Using the local Fourier analysis, we have the following observations:

1. The standard five-point 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 weighted 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. Define a grid function (essential in the LFA):

$$e_{i,j} = \sum_{\theta\in\Theta_n} \alpha_\theta e^{\sqrt{-1}(i\theta_1 + j\theta_2)}$$

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 $e^{\sqrt{-1}(i\theta_1+j\theta_2)}$ with $\theta := (\theta_1,\theta_2)$:

$$\lambda(\theta) := \frac{\alpha_\theta^{\text{new}}}{\alpha_\theta^{\text{old}}} = 1-\omega\Big(1-\frac{\cos(\theta_1)+\cos(\theta_2)}{2}\Big) \leqslant 1,$$

when $0<\omega\leqslant 1$. 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

$$\bar\rho := \sup_\theta\Big\{\big|\lambda(\theta)\big|\ :\ \theta\in[-\pi,\pi)^2\setminus[-\pi/2,\pi/2)^2\Big\}.$$

By plugging in the end points, we get the the smoothing factor for the weighted Jacobi method is

$$\bar\rho_{\text{WJ}} := \max\Big\{\Big|1-2\omega\Big|,\ \Big|1-\frac{1}{2}\omega\Big|\Big\}.$$

Remark 3.27 (Optimal damping factor for smoothing)**.** We notice that, if $\omega = 1$ (the Jacobi method), then $\bar\rho_{\text{WJ}} = 1$. Apparently, the "best" weight that minimizes the smoothing factor is $\omega = 4/5$, which leads to $\bar\rho_{\text{WJ}} = 3/5$. □

:GeoHighFreq2 **Remark 3.28** (What is high-frequency error)**.** In the preceding local Fourier analysis, we defined the high-frequency component to correspond to $\frac{\pi}{2}\leqslant|\theta_k|\leqslant\pi$. However, as noted in Remark 1.32, high-frequency components can be accurately approximated by looking at local behavior, while low-frequency components should be well represented on coarser grids; this will be further discussed in Remark 3.33. As such, this definition is not universal and must be adjusted to correspond with the coarsening algorithm under consideration. Semi-coarsening or red-black coarsening, for instance, will lead to distinct definitions of high-frequency, as detailed in [185]. Later, we will explore how to define this concept from an algebraic perspective. □

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

$$u_{i,j}^{\text{new}} = \frac{1}{4}\big(u_{i-1,j}^{\text{new}} + u_{i+1,j}^{\text{old}} + u_{i,j-1}^{\text{new}} + u_{i,j+1}^{\text{old}}\big) + \frac{1}{4}h^2 f_{i,j},\quad i,j = 1,\dots,n.$$

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 3.29 (Anisotropic problems and smoothing effect)**.** It should be observed that the aforementioned analysis only applies to problems that feature uniform partition and isotropic coefficients. For anisotropic problems, it is important to note that the G-S method (as well as other point relaxation methods) does not provide as effective a smoothing factor as in the isotropic case. In fact, the smoothing factor will converge to 1 as the ratio between small and large coefficients approaches 0 (refer to Chapter 6 for more detailed discussion). □

em:GSordering **Remark 3.30** (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 $\bar{\rho}_{\text{RB}}$ is $\frac{1}{4}$ [179, 185]. This means the smoothing effect of the red-black ordering for G-S is better. □

3.3.3 Smoothing effect

Considering the Richardson method (3.16), 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 (3.14) implies that

$$(\mathcal{B}_\omega v, v) \cong \omega h^d (v, v). \tag{3.22}$$ eqn:Richardso

For the finite element discretization of the Poisson's equation, we choose the weight of the Richardson iteration to be $\omega = h^{2-d}$, i.e.,

$$\mathcal{S}_{\text{R}} v := \mathcal{B}_\omega v = h^{2-d} \sum_{i=1}^N (v,\phi_i)\phi_i, \quad \forall v \in V. \tag{3.23}$$ eqn:Richardso

In view of (3.22) and using the fact that the spectral radius of the FE operator is $\rho(\mathcal{A}) \cong h^{-2}$ (see Remark 3.17), we find

$$(\mathcal{S}_{\mathrm{R}} v, v) \cong h^2 (v, v) \cong \frac{1}{\rho(\mathcal{A})}(v, v).$$

Roughly speaking, $\mathcal{S}_{\mathrm{R}}$ behaves like $\mathcal{A}^{-1}$ in the high-frequency regime. This is a natural property we will ask for from a smoother later on:

$$(\mathcal{S} v, v) \cong \frac{1}{\rho(\mathcal{A})}(v, v). \tag{3.24}$$ eqn:smoothing

In fact, such conditions are only needed in the range of $\mathcal{S}$.

Apparently, the weighted 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 (3.18), we have the Jacobi smoother

$$(\mathcal{S}_{\mathrm{J}} v, v) = (M \underline{\mathcal{S}_{\mathrm{J}} v}, \underline{v}) = (M D^{-1} M \underline{v}, \underline{v}) \cong h^{d+2}(\underline{v}, \underline{v}) \cong h^2 (v, v) \cong \frac{1}{\rho(\mathcal{A})}(v, v).$$

Next, we shall show an interesting fact that the G-S method behaves in a similar way as in the Jacobi method.

:GS-algebraic **Lemma 3.31** (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.

:SGS-operator **Lemma 3.32** (Smoothing property of SGS)**.** Let $\mathcal{S} : V \mapsto V$ be the symmetrized G-S (SGS) iterator. Then we have

$$\big(\mathcal{S} v, v\big) \cong h^2 (v, v) \cong \frac{1}{\rho(\mathcal{A})}(v, v). \tag{3.25}$$ eqn:smoother-

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

$$(\mathcal{S}v, v) = (M\underline{\mathcal{S}v}, \underline{v}) = (M\underline{\mathcal{S}}\,\underline{v}, \underline{v}) = \left\|D^{\frac{1}{2}}(D+L)^{-1}M\underline{v}\right\|_0^2.$$

Hence to show the lemma is equivalent to prove that

$$\left\|D^{\frac{1}{2}}(D+L)^{-1}M\underline{v}\right\|_0^2 \cong h^2(M\underline{v}, \underline{v}).$$

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

$$h^{d-2}(D\xi, \xi) \cong h^{2(d-2)}\|\xi\|_0^2 \cong \left\|(D+L)\xi\right\|_0^2 = (M\underline{v}, M\underline{v}), \quad \forall\, \xi \in \mathbb{R}^N,$$

which can be seen from Lemma 3.31. □

3.3.4 Smoother as preconditioner ⋆

From the property (3.24), which is satisfied by the aforementioned popular smoothers, we can easily see that

$$\rho_{\mathcal{A}}^{-1}(v, v) \lesssim (\mathcal{S}v, v) \lesssim \rho_{\mathcal{A}}^{-1}(v, v), \tag{3.26}$$ eqn:smoothing

where $\rho_{\mathcal{A}} := \rho(\mathcal{A})$. In this note, we refer to the smoothing property as the phenomenon whereby the smoother $\mathcal{S}$ behaves similarly to $\mathcal{A}^{-1}$ in the high frequency regime. While other conditions or assumptions for smoothers have been explored and discussed in the literature, readers who are interested in learning more about general smoothers that are defined as additive and multiplicative Schwarz methods are encouraged to refer to the paper by Bramble and Pasciak [44], as well as the references cited within it.

Based on this property, we can establish a lower bound for the minimal eigenvalue, namely $\rho_{\mathcal{A}}^{-1} \lesssim \lambda_{\min}(\mathcal{S})$. When the smoother is symmetric, this property also suggests that the smoother satisfies the Symmetric Positive Definite (SPD) criteria, making its symmetrized form, $\overline{\mathcal{S}}$, a viable preconditioner candidate. With regards to Remark 3.17, which asserts that $\|v\|_0^2 \lesssim (v, v)_{\mathcal{A}} \leqslant \rho_{\mathcal{A}}\|v\|_0^2$, a simple manipulation of the terms leads us to the following conclusion:

$$\rho_{\mathcal{A}}^{-1}(v, v)_{\mathcal{A}} \lesssim \rho_{\mathcal{A}}^{-1}(\mathcal{A}v, \mathcal{A}v) \lesssim (\mathcal{S}\mathcal{A}v, v)_{\mathcal{A}} \lesssim \rho_{\mathcal{A}}^{-1}(\mathcal{A}v, \mathcal{A}v) \leqslant (v, v)_{\mathcal{A}}. \tag{3.27}$$ eqn:smoothing

Thanks to Lemmas 2.48 and 2.49, we can deduce from (3.27) that

$$\kappa(\mathcal{S}\mathcal{A}) \lesssim \rho(\mathcal{A}) \cong \kappa(\mathcal{A}).$$

This suggests that while these smoothers can be useful as preconditioners, they may not necessarily improve the condition number of the linear system. Therefore, it becomes clear that designing an effective preconditioner requires more than just a good smoother—a topic we will explore further in the following sections of this note.

3.4 Two-grid methods

rid-algorithm

From the analysis in §3.3, we have found that local relaxation methods (smoothers) can damp the oscillatory components of the error rather quickly. To address the less efficient treatment of the smooth components by local relaxation methods, coarser levels can be introduced, as motivated by the two-level DD method in §2.4. After a few smoothing steps, the resulting problem can be approximated on a coarser grid and continued with a "coarse version" of the problem. This approach allows high frequency parts of the error to be resolved with relaxation schemes, while the low frequency part is addressed by the coarse levels. To begin, we investigate a much simpler case: the two-grid method, before delving into multilevel methods.

Firstly, we make an observation that heuristically explains why a coarse-grid solution can provide a good approximation for smooth functions. Specifically, smooth functions can be accurately represented on the coarse-grid, which is the final piece of the puzzle that motivates multilevel iterative methods. We provide only a brief outline of the proof here and leave the complete proof to the readers (see HW 3.5).

rseCorrection

Remark 3.33 (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

$$\begin{aligned} \|u_h - u_H\|_0^2 = a[w, u_h - u_H] &= a[w - w_H, u_h - u_H] \\ &\leqslant |\!|\!| w - w_H |\!|\!| \, |\!|\!| u_h - u_H |\!|\!| \lesssim H|w|_2 \, |\!|\!| u_h - u_H |\!|\!| \,. \end{aligned}$$

Hence the following inequality holds

$$\|u_h - u_H\|_0 \lesssim H \, |\!|\!| u_h - u_H |\!|\!| \lesssim H \, |\!|\!| u_h |\!|\!| \,. \tag{3.28}$$

eqn:SmoothPar

That is to say, if u_h is relatively smooth (small first derivatives), then u_h can be well approximated by u_H. □

We summarize the motivations of multilevel iterative methods discussed so far:

- Solution of local problems can be used to approximate high frequency components of the global solution; see Remark 1.32.
- A high resolution mesh can be effective in capturing local features; see Remark 1.33.
- Local relaxation methods are effective in reducing high frequency errors but not for low frequency errors; smoother alone is not good enough; see Figures 1.4 and 3.2.
- Coarse-space problems can provide good approximations to the fine-space problems if the solution is smooth; see Remark 3.33.
- Multilevel iterative methods can be used as preconditioners for Krylov subspace methods; see the inequalities in (2.54).

3.4.1 General two-grid methods

Let V_h be fine-grid finite element space and V_H be the coarse-grid space (usually it is a subspace of V_h.) The two-grid method for equation (3.2) can be described as

alg:atwo-grid **Algorithm 3.1** (General two-grid method)**.** Given an initial guess $u^{(0)} \in V_h$.

(i) **Pre-smoothing**: Apply a few smoothing 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 smoothing 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 3.34 (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

$$\mathcal{A}_c = \mathcal{I}_c^T \mathcal{A} \mathcal{I}_c = \mathcal{Q}_c \mathcal{A} \mathcal{I}_c$$

is defined by the Galerkin relation. □

Suppose that $\mathcal{S}$ is a smoother and $\mathcal{B}_c$ is a solver or approximate solver for the coarse-grid problem.

alg:two-grid **Algorithm 3.2** (Two-grid 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**: $u^{(2)} = u^{(1)} + (\mathcal{I}_c \mathcal{B}_c \mathcal{I}_c^T)(f - \mathcal{A}u^{(1)})$;

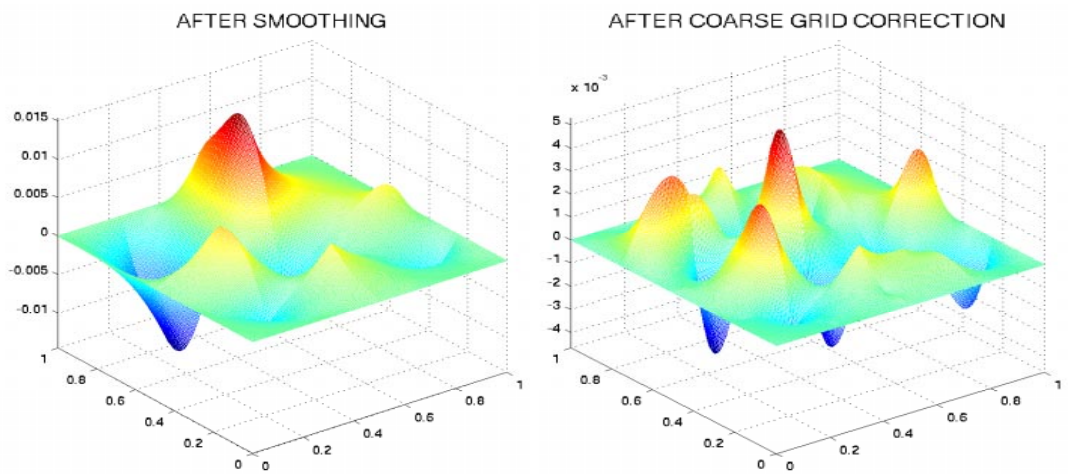
(iii) **Post-smoothing**: $u^{(3)} = u^{(2)} + \mathcal{S}^T(f - \mathcal{A}u^{(2)})$.

If $\mathcal{B}_c = \mathcal{A}_c^{-1}$, the algorithm is called an exact two-grid (TG) method. On the contrary, if $\mathcal{B}_c \approx \mathcal{A}_c^{-1}$, the algorithm is an inexact two-grid (ITG) method.

An interesting observation about this algorithm is its striking similarity to the multigrid algorithm discussed in Chapter 1. In particular, it consists of two core processes: smoothing steps and coarse-grid correction (CGC). When implemented in a complimentary fashion, these processes can lead to highly effective algorithms with superior performance. Specifically, choosing appropriate $\mathcal{S}$, V_c, and $\mathcal{B}_c$ can enhance the efficiency of the algorithm in solving the given equation. The two-grid method was developed with the intention of capturing high-frequency components of error on the fine-grid and delegating the low-frequency components to the coarser grid. The effect of coarse-grid correction is illustrated in Figure 3.3; it's worth noting that the two images depicted there are in different scales.

3.4.2 Convergence analysis of two-grid method

In this section, we will estimate the convergence rate of two-grid methods. To begin, we present a few simple lemmas. The first lemma concerns the norm of oblique projections, also known as Kato's lemma, which has been proven and reproduced in several different fields. For further details, please refer to the paper by Szyld [181].



rsecorrection Figure 3.3: After coarse-grid correction, global low frequency is replaced by local high frequency.

lem:Kato **Lemma 3.35** (Norm of oblique projections)**.** If Π is a continuous projection on 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. 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 $\|w\|^2 = \|x\|^2 + \|y\|^2 + 2\,(x, y) = 1$ 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.

lem:TGiter **Lemma 3.36** (Iterator of two-grid method)**.** The two-grid method has a corresponding iterator $\mathcal{B}_{\text{TG}} : V' \to V$ defined as

$$\mathcal{B}_{\text{TG}} = \overline{\mathcal{S}} + \left(\mathcal{I} - \mathcal{S}^T \mathcal{A}\right) \mathcal{I}_c \mathcal{B}_c \mathcal{I}_c^T \left(\mathcal{I} - \mathcal{A}\mathcal{S}\right), \tag{3.29}$$ B_Inv

where $\overline{\mathcal{S}} = \mathcal{S}^T + \mathcal{S} - \mathcal{S}^T \mathcal{A}\mathcal{S}$ is the symmetrization of the smoother $\mathcal{S}$.

lem:TGerror **Lemma 3.37** (Error propagation of two-grid method)**.** The error propagation operator $\mathcal{E}_{\rm TG} = \mathcal{I} - \mathcal{B}_{\rm TG}\mathcal{A}$ for the two-grid method is

$$\mathcal{E}_{\rm TG} = (\mathcal{I} - \mathcal{S}^T\mathcal{A})(\mathcal{I} - \mathcal{B}_c\mathcal{A}_c\Pi_c)(\mathcal{I} - \mathcal{S}\mathcal{A}), \tag{3.30}$$ E_op

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

$$\mathcal{E}_{\rm TG} = (\mathcal{I} - \mathcal{S}^T\mathcal{A})(\mathcal{I} - \Pi_c)(\mathcal{I} - \mathcal{S}\mathcal{A}). \tag{3.31}$$ E_op1

The explicit formula for the projection operator Π_c in (3.31) can be written as

$$\Pi_c = \mathcal{I}_c\mathcal{A}_c^{-1}\mathcal{I}_c^T\mathcal{A}.$$

Upon examining equation (3.31), we recognize the crucial role played by reducing the norms of both the coarse-level correction operator $\mathcal{I} - \Pi_c$ and the error reduction operator $\mathcal{I} - \mathcal{S}\mathcal{A}$ in achieving optimal performance. Furthermore, by taking into account Lemma 3.35, it is evident that analyzing the behavior of $\|\Pi_c\|_{\mathcal{A}}$ is equivalent to assessing the exact coarse-level correction operator $\|\mathcal{I} - \Pi_c\|_{\mathcal{A}}$.

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 (3.31).

We now present a theorem which gives the convergence rate of a simplified two-grid method (Algorithm 3.3) in terms of approximability of the coarser space V_c.

alg:two-grid2 **Algorithm 3.3** (Simplified two-grid 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**: $u^{(2)} = u^{(1)} + \mathcal{S}(f - \mathcal{A}u^{(1)})$.

For simplicity, we assume that $\mathcal{S} = \overline{\mathcal{S}}$ is SPD. In the two-grid method analysis below, we need the following notation

$$\mathcal{T} = \mathcal{T}_{\overline{\mathcal{S}}} := \overline{\mathcal{S}}\mathcal{A} \,:\, V \mapsto V. \tag{3.32}$$ eqn:SA

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 two-grid method is given by the following theorem. One can compare this result with the convergence rate of the smoother or iterator, $\overline{\mathcal{S}}$, given by Theorem 2.16:

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

The following theorem can be derived from the XZ identity, as stated in Theorem 4.15. Here, for the sake of completeness, we present a direct proof originally given in [202].

d-convergence **Theorem 3.38** (Convergence rate of the two-grid method)**.** The convergence rate of the two-grid method, Algorithm 3.3, with the exact coarse-level solver is given by

$$\|\mathcal{E}_{\mathrm{TG}}\|^2_{\mathcal{A}} = 1 - \frac{1}{c_1(V_c)}, \tag{3.33}$$ eq:two-grid-c

where

$$c_1(V_c) := \sup_{v\in V} \frac{\|(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v\|^2_{\overline{\mathcal{S}}^{-1}}}{\|v\|^2_{\mathcal{A}}} = \sup_{v\in V} \inf_{v_c\in V_c} \frac{\|v - v_c\|^2_{\overline{\mathcal{S}}^{-1}}}{\|v\|^2_{\mathcal{A}}}. \tag{3.34}$$ KVc

Sketch of the proof. (1) It follows from (3.30) that the simplified TG method (Algorithm 3.3) has the following error propagation operator

$$\mathcal{E}_{\mathrm{TG}} = (\mathcal{I} - \overline{\mathcal{S}}\mathcal{A})(\mathcal{I} - \Pi_c).$$

Hence, we can immediately see that

$$\|\mathcal{E}_{\mathrm{TG}}\|^2_{\mathcal{A}} = \sup_{v\in V} \frac{\|(\mathcal{I} - \overline{\mathcal{S}}\mathcal{A})(\mathcal{I} - \Pi_c)v\|^2_{\mathcal{A}}}{\|v\|^2_{\mathcal{A}}} = \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\|(\mathcal{I} - \overline{\mathcal{S}}\mathcal{A})v\|^2_{\mathcal{A}}}{\|v\|^2_{\mathcal{A}}}.$$

Using the definition of $(\cdot,\cdot)_{\mathcal{A}}$-projection Π_c, we can show that

$$\|\mathcal{E}_{\mathrm{TG}}\|^2_{\mathcal{A}} = \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big((\mathcal{I} - \mathcal{T})v, v\big)_{\mathcal{A}}}{\|v\|^2_{\mathcal{A}}} = 1 - \inf_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big(\mathcal{T}v, v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}} = 1 - \inf_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big((\mathcal{I} - \Pi_c)\mathcal{T}v, v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}}.$$

(2) Define

$$\mathcal{X} := (\mathcal{I} - \Pi_c)\mathcal{T} \,:\, V_c^{\perp_{\mathcal{A}}} \mapsto V_c^{\perp_{\mathcal{A}}} \tag{3.35}$$ eqn:Xdef

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 $\big(\Pi_c\mathcal{T}^{-1}(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})u, v\big)_{\mathcal{A}} = \big(\mathcal{T}^{-1}(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})u, v\big)_{\mathcal{A}} = \big((\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})u, v\big)_{\overline{\mathcal{S}}^{-1}} = 0$ for any $u \in V_c^{\perp_{\mathcal{A}}}$ and $v \in V_c$, we have $\Pi_c\mathcal{Z} = 0$, which implies that $\mathcal{Z} : V_c^{\perp_{\mathcal{A}}} \mapsto V_c^{\perp_{\mathcal{A}}}$. Furthermore, by the definition of projections, we get

$$\mathcal{X}\mathcal{Z} = (\mathcal{I} - \Pi_c)(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}}) = \mathcal{I} - \Pi_c = \mathcal{I} \quad \text{on } V_c^{\perp_{\mathcal{A}}}.$$

(3) Consequently $\lambda_{\min}(\mathcal{X}) = \lambda_{\max}(\mathcal{Z})^{-1}$. Finally,

$$\begin{aligned}\lambda_{\max}(\mathcal{Z}) &= \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{(\mathcal{T}^{-1}(\mathcal{I}-\mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v,v)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}} = \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\big((\mathcal{I}-\mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v,v\big)_{\overline{\mathcal{S}}^{-1}}}{(v,v)_{\mathcal{A}}} \\ &= \sup_{v\in V_c^{\perp_{\mathcal{A}}}} \frac{\|(\mathcal{I}-\mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v\|^2_{\overline{\mathcal{S}}^{-1}}}{(v,v)_{\mathcal{A}}} = \sup_{v\in V} \frac{\|(\mathcal{I}-\mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v\|^2_{\overline{\mathcal{S}}^{-1}}}{\|v\|^2_{\mathcal{A}}} =: c_1(V_c).\end{aligned}$$

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 $v\in V$ (using the same argument in the very beginning of this proof). □

Theorem 3.38 estimates the convergence rate of a two-grid method in terms of $c_1(V_c)$. A smaller bound on $c_1(V_c)$ indicates a faster convergence rate for a given method. Specifically, the two-grid method achieves uniform convergence if $c_1(V_c)$ is uniformly bounded with respect to mesh size. However, one potential issue when applying Theorem 3.38 is that it can be challenging 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

$$C_L\|v\|^2_{\mathcal{D}} \leqslant \|v\|^2_{\overline{\mathcal{S}}^{-1}} \leqslant C_U\|v\|^2_{\mathcal{D}}, \quad \forall\, v\in V.$$

Similar to the definition of $c_1(V_c)$, we can introduce the quantity

$$c_1(V_c,\mathcal{D}) = \sup_{v\in V}\frac{\|(\mathcal{I}-\mathcal{Q}_{\mathcal{D}})v\|^2_{\mathcal{D}}}{\|v\|^2_{\mathcal{A}}} = \sup_{v\in V}\inf_{v_c\in V_c}\frac{\|v-v_c\|^2_{\mathcal{D}}}{\|v\|^2_{\mathcal{A}}},$$

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:

-convergence2 **Theorem 3.39** (An estimate of convergence rate of TG)**.** The convergence rate of the two-grid method (3.29) with exact coarse-level solver is given by

$$1-\frac{1}{C_L\, c_1(V_c,\mathcal{D})} \leqslant \|\mathcal{E}_{\mathrm{TG}}\|^2_{\mathcal{A}} \leqslant 1-\frac{1}{C_U\, c_1(V_c,\mathcal{D})} \leqslant 1-\frac{1}{C_U C}, \tag{3.36}$$ eq:two-grid-c

where C is an upper bound of $c_1(V_c,\mathcal{D})$, i.e.,

$$\inf_{v_c\in V_c}\|v-v_c\|^2_{\mathcal{D}} \leqslant C\|v\|^2_{\mathcal{A}}, \quad \forall\, v\in V. \tag{3.37}$$ KVc1

It is important to note that we have only presented 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 two-grid method by using the convergence factor of the exact two-grid method, as outlined in [152, 205].

3.4.3 Optimal coarse space

Next, we will discuss how to choose the coarse space to optimize the convergence rate, which is a helpful concept when developing algebraic multigrid methods (AMGs). The space that spans the eigenvectors of $\overline{\mathcal{S}}\mathcal{A}$ corresponding to small eigenvalues is the "best" coarse space as it minimizes the convergence rate. The following theorem characterizes the optimal choice of coarse space V_c with a fixed smoother $\mathcal{S}$:

-grid-optimal

Theorem 3.40 (Optimal coarse space)**.** Given a smoother $\mathcal{S}$, the best coarse space of dimension N_c is given by

$$V_c^{\text{opt}} := \underset{\dim V_c = N_c}{\operatorname{argmin}} \|\mathcal{E}_{\text{TG}}(V_c)\|_{\mathcal{A}} = \operatorname{span}\left\{\xi_k\right\}_{k=1}^{N_c}, \tag{3.38}$$

optVH

where $\left\{\xi_k\right\}_{k=1}^{N_c}$ are the eigenfunctions corresponding to the N_c 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 3.38, 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 [71], 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 (3.38). □

Remark 3.41 (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 eigenpairs of $\overline{\mathcal{S}}\mathcal{A}$ are expensive to compute, the practical value of Theorem 3.40 is limited. But it will provide useful guidance in the design algebraic multilevel methods in §7.1.

3.5 Algebraic representation of two-grid methods

:MatrixFromTG

Prior to implementation, it is essential to comprehend the matrix representation of an algorithm in order to tackle inquiries arising from it, such as those noted in Section 1.4. In this regard, we will outline the matrix representation of the two-grid method within the finite element domain.

3.5.1 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 $P = \underline{\mathcal{I}_c}$.

PreserveConst

Remark 3.42 (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.,

$$P\, \mathbf{1}_{N_c} = \mathbf{1}_N.$$

The prolongation preserves the smooth vectors (actually, $\mathbf{1}_N$ is in the null space of $\hat{\mathcal{A}}$ in this case). □

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

$$\underline{\mathcal{Q}_c v} = \underline{v_c} = M_c^{-1}\vec{v}_c = M_c^{-1}\underline{\mathcal{I}_c}^T M \underline{v} \quad \Longrightarrow \quad \underline{\mathcal{I}_c^T} = \underline{\mathcal{Q}_c} = M_c^{-1}\underline{\mathcal{I}_c}^T M = M_c^{-1}P^T M. \tag{3.39}$$ eqn:MatQ

3.5.2 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

$$\underline{\mathcal{A}_c} = \underline{\mathcal{Q}_c}\,\underline{\mathcal{A}}\,\underline{\mathcal{I}_c} \quad \Longrightarrow \quad \hat{\mathcal{A}}_c = M_c\underline{\mathcal{A}_c} = M_c\underline{\mathcal{Q}_c}\,\underline{\mathcal{A}}\,\underline{\mathcal{I}_c} = P^T M \underline{\mathcal{A}} P = P^T \hat{\mathcal{A}} P. \tag{3.40}$$ eqn:Ac-PtAP

Then the coarse stiffness matrix satisfies

$$\hat{\mathcal{A}}_c = P^T \hat{\mathcal{A}} P. \tag{3.41}$$ eqn:PtAP

Therefore, the algebraic form (3.41) 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 3.43 (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 [65] for example) states $R = \alpha P^T$. This difference comes from the scaling effect caused by different meshsizes. 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

$$\hat{\mathcal{A}}_c = P^T \hat{\mathcal{A}} P \quad \Longrightarrow \quad A_c = \left(\frac{h}{H} P^T\right) AP =: RAP.$$

This remark explains how we can obtain such the constant α in general. □

3.5.3 Two-grid iterator in matrix form

From (3.29), we have that the two-grid 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

$$\underline{\mathcal{B}_{\mathrm{TG}}} = \underline{\overline{\mathcal{S}}} + \big(\underline{\mathcal{I}} - \underline{\mathcal{S}^T\mathcal{A}}\big)\underline{\mathcal{I}_c\mathcal{A}_c^{-1}\mathcal{I}_c^T}\big(\underline{\mathcal{I}} - \underline{\mathcal{A}\mathcal{S}}\big).$$

So we define

$$B_{\mathrm{TG}} := \underline{\mathcal{B}_{\mathrm{TG}}}M^{-1} = \underline{\overline{\mathcal{S}}}M^{-1} + \big(\underline{\mathcal{I}} - \underline{\mathcal{S}^T\mathcal{A}}\big)\underline{\mathcal{I}_c\mathcal{A}_c^{-1}\mathcal{I}_c^T}\big(\underline{\mathcal{I}} - \underline{\mathcal{A}\mathcal{S}}\big)M^{-1}.$$

Using the matrix form the symmetrization, inversion, and transpose derived earlier, we can easily get

$$B_{\mathrm{TG}} = \overline{S} + (I - S^TA)PA_c^{-1}P^T(I - AS) = \overline{S} + (I - S^TA)P\big(P^TAP\big)^{-1}P^T(I - AS).$$

Now we are ready to introduce the matrix representation of the two-grid method for solving the linear system $A\underline{u} = \vec{f}$. We describe the two-grid method as a preconditioner action $B_{\mathrm{TG}}(\cdot)$. For the sake of simplicity and consistency, here we abuse the notation and still use the general vector notation $\vec{v}$ to denote the primal form of v. 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 two-grid method

```
%% Given any vector r⃗;
Pre-smoothing: v⃗ ← Sr⃗;
Coarse-grid correction: w⃗ ← v⃗ + P(P^TAP)^{-1}P^T(r⃗ − Av⃗);
Post-smoothing: B_TG r⃗ ← w⃗ + S^T(r⃗ − Aw⃗);
```

Similarly, from (3.31), we have matrix form of the iteration matrix

$$\begin{aligned} E_{\mathrm{TG}} = \underline{\mathcal{E}_{\mathrm{TG}}} &= (I - S^TA)(I - PA_c^{-1}P^TA)(I - SA) \\ &= (I - S^TA)(I - \Pi_c)(I - SA), \end{aligned} \tag{3.42}$$

eqn:ETG_matri

where $\Pi_c := \underline{\mathit{\Pi}_c} = PA_c^{-1}P^TA$ is the matrix form of the coarse-level correction; see HW 3.8.

In [91], an algebraic analysis of the two-grid 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 $\tilde{\Pi}_c := A^{\frac{1}{2}}\Pi_cA^{-\frac{1}{2}} = A^{\frac{1}{2}}PA_c^{-1}P^TA^{\frac{1}{2}}$. This algebraic form is explicit and might be easier to understand compared with Theorem 3.38.

3.6 Homework problems

hw:ApostBound **HW 3.1.** Show the a posteriori error bounds (3.9).

hw:matrix-rep **HW 3.2.** Prove the statements in Lemma 3.21.

hw:Richardson **HW 3.3.** Show the operator form and matrix form (3.16) of the Richardson method.

hw:TG **HW 3.4.** Prove Lemma 3.36 and Lemma 3.37.

rseCorrection **HW 3.5.** Give a complete proof of Remark 3.33.

hw:MGTG **HW 3.6.** Write the 1D multigrid method in §1.4 as a two-grid method (Algorithm 3.2) called recursively and modify your implementation in this way.

hw:ETG **HW 3.7.** Give the detailed proof of Theorem 3.38. 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 (3.35) is self-adjoint with respect to $(\cdot,\cdot)_{\mathcal{A}}$-inner product.

ETGmatrixform **HW 3.8.** Derive the primal matrix representation of Π_c and $\mathcal{E}_{\mathrm{TG}}$ respectively.

Chapter 4

Subspace Correction Methods

ch:subspace

In the previous chapters, we have introduced several iterative solvers for the linear equation

$$\mathcal{A}u = f, \tag{4.1}$$

eqn:original

where $\mathcal{A} : V \mapsto V$ is SPD. A linear stationary iterative method can be written as

$$u^{\text{new}} = u^{\text{old}} + \mathcal{B}(f - \mathcal{A}u^{\text{old}}). \tag{4.2}$$

eqn:original-

In Chapter 2, we have seen that: If $\mathcal{B}$ is an SPD operator, the above iterative method (4.2) converges. Additionally, $\mathcal{B}$ can be effectively used as a preconditioner for Krylov subspace methods, such as the Preconditioned Conjugate Gradient (PCG) method.

In this chapter, we introduce a theoretical framework for analyzing linear iterative methods and preconditioners through the lens of space decomposition and subspace corrections. This unified framework serves as a foundation for establishing convergence theory for a range of methods, including the multigrid method, domain decomposition method, and the two-grid method discussed in earlier chapters.

4.1 Successive and parallel subspace corrections

Suppose we have a subspace decomposition of the solution space, where the overall solution can be represented as a sum of components from these subspaces:

$$V = \sum_{j=1}^{J} V_j \qquad \text{and} \qquad V_j \subset V \ \ (j = 1, \ldots, J).$$

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. This type of

decomposition is often utilized in iterative methods, allowing for the solution to be computed or approximated by solving smaller problems within each subspace. The interactions between these subspaces, and how corrections are made across them, form the basis for understanding the efficiency and convergence of methods such as multigrid or domain decomposition.

4.1.1 Abstract framework for subspace corrections

We begin by defining a few linear operators that have been introduced in various contexts throughout the previous chapters. These operators play a central role in the analysis and formulation of the iterative methods discussed, and understanding their properties will be crucial for the development of the theoretical framework presented in this chapter.

f:subproblems **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:

m:projections **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

$$\hat{\mathcal{A}}_c \underline{u_c} = P^T \hat{\mathcal{A}} \, \underline{u} \quad \Longrightarrow \quad \underline{u_c} = \hat{\mathcal{A}}_c^{-1} P^T \hat{\mathcal{A}} \, \underline{u}.$$

Hence, we obtain

$$\underline{\Pi_c} \, \underline{u} = \underline{\Pi_c u} = \underline{u_c} = \hat{\mathcal{A}}_c^{-1} P^T \hat{\mathcal{A}} \, \underline{u}$$

and 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 the above equation with the matrix form of the L^2-projection in (3.39), i.e.

$$\underline{\mathcal{Q}_c} = M_c^{-1} P^T M,$$

which has been derived in the previous chapter. □

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:

$$\mathcal{A}e = r \quad \Longrightarrow \quad \mathcal{Q}_j \mathcal{A} e = \mathcal{Q}_j r \quad \Longrightarrow \quad \mathcal{A}_j \Pi_j e = \mathcal{Q}_j r \quad \Longrightarrow \quad \mathcal{A}_j e_j = r_j,$$

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; see §2.4. We first describe the idea of subspace correction in the following abstract "algorithm"[1], which is just a generalization of Algorithm 2.1:

alg:msc **Algorithm 4.1** (Method of subspace corrections)**.** $u^{\text{new}} = MSC(u^{\text{old}})$

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

[1] Note that this procedure is not a real algorithm due to the fact that it does not specify how to combine the corrections $\tilde{e}_j$'s from different subspaces.

(ii) **Solve error equation on** V_j: $\mathcal{A}_j e_j = r_j$ by $e_j \approx \tilde{e}_j = \mathcal{S}_j r_j$

(iii) **Apply correction**: $u^{\text{new}} = u^{\text{old}} + \tilde{e}_j$

Notice that, instead of constructing an iterator for the whole system, Algorithm 4.1 only considers one subproblem (on the subspace V_j) at a time.

rem:subsolvers **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 find

$$u_j = \operatorname*{argmin}_{v \in V_j} \|u - v\|_{\mathcal{A}}.$$

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. □

4.1.2 SSC and PSC methods

Algorithm 4.1 did not specify how to combine the corrections $\tilde{e}_j$'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 (2.58) and PSC can be viewed as the additive Schwarz method (2.57). We now give descriptions of the SSC and PSC algorithms.

alg:ssc **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) $u^{\text{new}} = v$

ted-subsolvers **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 $\mathcal{S}_j$. □

alg:psc **Algorithm 4.3** (Parallel subspace corrections)**.** $u^{\text{new}} = PSC(u^{\text{old}})$

(i) $r^{\text{old}} = f - \mathcal{A}u^{\text{old}}$

(ii) $u^{\text{new}} = u^{\text{old}} + \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j r^{\text{old}}$

From the above algorithms (Algorithm 4.2 and 4.3), it is immediately clear why they are named as SSC and PSC, respectively.

Define operators

$$\mathcal{T}_j = \mathcal{T}_{\mathcal{S}_j} := \mathcal{S}_j \mathcal{Q}_j \mathcal{A} = \mathcal{S}_j \mathcal{A}_j \Pi_j : V \mapsto V_j.$$

Apparently, as in (3.32), 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 assume that all the subspace solvers $\mathcal{S}_j$ 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 subspace solvers are exact on each subspace, then we have $\mathcal{T}_j = \Pi_j : V \mapsto V_j$.

- For the SSC method, the iterative error satisfy:

$$u - u^{\text{new}} = (\mathcal{I} - \mathcal{B}_{\text{SSC}}\mathcal{A})(u - u^{\text{old}}) = (\mathcal{I} - \mathcal{T}_J) \cdots (\mathcal{I} - \mathcal{T}_2)(\mathcal{I} - \mathcal{T}_1)(u - u^{\text{old}}). \tag{4.6}$$ eqn:SSC

  If $J = N$, $V_j = \text{span}\{\phi_j\}$ and $\mathcal{S}_j = \mathcal{A}_j^{-1}$ $(j = 1, \ldots, N)$, then the corresponding SSC method (4.6) is exactly the GS method; see (2.22).

- For the PSC method, the iterator (or, more often, the preconditioner) satisfies

$$\mathcal{B}_{\text{PSC}} = \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j = \sum_{j=1}^{J} \mathcal{I}_j \mathcal{S}_j \mathcal{Q}_j \qquad \text{and} \qquad \mathcal{B}_{\text{PSC}}\mathcal{A} = \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j \mathcal{A} = \sum_{j=1}^{J} \mathcal{T}_j. \tag{4.7}$$ eqn:PSC

  If all $\mathcal{S}_j$'s $(j = 1, \ldots, J)$ are SPD, then the preconditioner $\mathcal{B}_{\text{PSC}}$ is also SPD; see HW 4.2. If each subspace $V_j = \text{span}\{\phi_j\}$ $(j = 1, \ldots, N)$, 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 discussed any multilevel structures within the above methods. To incorporate multilevel iterative methods into the subspace correction framework, we will need to introduce multilevel subspace decompositions. Before we do that, we first consider the general convergence analysis of MSC.

4.2 Expanded systems and block solvers

In this section, we discuss an expanded system of $\mathcal{A}u = f$ (namely the equation (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.

4.2.1 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 have

$$\mathbf{\Pi u} = (\mathcal{I}_1, \ldots, \mathcal{I}_J) \begin{pmatrix} u_1 \\ \vdots \\ u_J \end{pmatrix} = \sum_{j=1}^{J} \mathcal{I}_j u_j = \sum_{j=1}^{J} u_j.$$

So we have

$$\mathbf{\Pi}^T = \begin{pmatrix} \mathcal{I}_1^T \\ \vdots \\ \mathcal{I}_J^T \end{pmatrix} = \begin{pmatrix} \mathcal{Q}_1 \\ \vdots \\ \mathcal{Q}_J \end{pmatrix}.$$

It should be noted that $\mathbf{\Pi}\,\mathbf{\Pi}^T \neq \mathcal{I}$, in general.

Define an operator matrix $\mathbf{A} : \mathbf{V} \mapsto \mathbf{V}$ 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

$$\mathbf{A} := \mathbf{\Pi}^T \mathcal{A}\, \mathbf{\Pi} = \begin{pmatrix} \mathcal{A}_{1,1} & \cdots & \mathcal{A}_{1,J} \\ \vdots & \ddots & \vdots \\ \mathcal{A}_{J,1} & \cdots & \mathcal{A}_{J,J} \end{pmatrix} = \Big(\mathbf{A}_{i,j}\Big)_{J \times J}.$$

Given any right-hand side function $f \in V$, we define

$$\mathbf{f} := \mathbf{\Pi}^T f = \begin{pmatrix} \mathcal{I}_1^T f \\ \vdots \\ \mathcal{I}_J^T f \end{pmatrix} = \begin{pmatrix} \mathcal{Q}_1 f \\ \vdots \\ \mathcal{Q}_J f \end{pmatrix} \in \mathbf{V}.$$

In this setting, we can consider the following problem: Find $\mathbf{u} \in \mathbf{V}$, such that

$$\mathbf{A}\mathbf{u} = \mathbf{f}. \tag{4.8}$$ eqn:expanded

This system is called the *expanded equation* of the original linear equation (4.1).

We will see how the solutions 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 $\mathbf{B} : \mathbf{V} \mapsto \mathbf{V}$

$$|\mathbf{B}|_{\mathbf{A}} := \sup_{|\mathbf{v}|_{\mathbf{A}} \neq 0} \frac{|\mathbf{B}\mathbf{v}|_{\mathbf{A}}}{|\mathbf{v}|_{\mathbf{A}}}.$$

4.2.2 Block solvers for expanded systems

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

$$\mathbf{u}^{\text{new}} = \mathbf{u}^{\text{old}} + \mathbf{B}(\mathbf{f} - \mathbf{A}\mathbf{u}^{\text{old}}), \tag{4.9}$$ eqn:iterEx

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 (2.17), we can generalize the block Jacobi and GS methods. Suppose that there is a non-singular block diagonal smoother (or relaxation operator) $\mathbf{S} : \mathbf{V} \mapsto \mathbf{V}$, i.e.,

$$\mathbf{S} = \text{diag}(\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_J), \quad \text{with } \mathcal{S}_j : V_j \mapsto V_j, \quad j = 1, 2, \ldots, J.$$

We define generalized block Jacobi method by $\mathbf{B} = \mathbf{S}$ and the generalized block Gauss–Seidel method by $\mathbf{B} = (\mathbf{S}^{-1} + \mathbf{L})^{-1}$.

m:equivalence **Theorem 4.7** (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 behavior, 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

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

The contraction factor can be written

$$\begin{aligned}
\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2 &= \sup_{v\neq 0}\frac{\|(\mathcal{I}-\mathcal{B}\mathcal{A})v\|_{\mathcal{A}}^2}{\|v\|_{\mathcal{A}}^2} = \sup_{v\neq 0}\frac{(v,v)_{\mathcal{A}} - \big((\mathcal{B}^T + \mathcal{B} - \mathcal{B}^T\mathcal{A}\mathcal{B})\mathcal{A}v, v\big)_{\mathcal{A}}}{(v,v)_{\mathcal{A}}} \\
&= \sup_{\mathbf{\Pi}\mathbf{v}\neq 0}\frac{(\mathbf{\Pi}\mathbf{v},\mathbf{\Pi}\mathbf{v})_{\mathcal{A}} - \big((\mathcal{B}^T + \mathcal{B} - \mathcal{B}^T\mathcal{A}\mathcal{B})\mathcal{A}\mathbf{\Pi}\mathbf{v},\mathbf{\Pi}\mathbf{v}\big)_{\mathcal{A}}}{(\mathbf{\Pi}\mathbf{v},\mathbf{\Pi}\mathbf{v})_{\mathcal{A}}} \\
&= \sup_{|\mathbf{v}|_{\mathbf{A}}\neq 0}\frac{(\mathbf{v},\mathbf{v})_{\mathbf{A}} - \big((\mathbf{B}^T + \mathbf{B} - \mathbf{B}^T\mathbf{A}\mathbf{B})\mathbf{A}\mathbf{v},\mathbf{v}\big)_{\mathbf{A}}}{|\mathbf{v}|_{\mathbf{A}}^2} \\
&= |\mathbf{I} - \mathbf{B}\mathbf{A}|_{\mathbf{A}}^2.
\end{aligned}$$

Hence we get the desired result. □

ex:Jacobi-psc **Example 4.8** (Block Jacobi method and PSC)**.** We now apply the block Jacobi method for the expanded system (4.8), i.e.,

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

We notice that $\mathbf{D}^{-1}\mathbf{A} = \mathbf{D}^{-1}\mathbf{\Pi}^T\mathcal{A}\mathbf{\Pi}$, which is spectrally equivalent to $\mathbf{\Pi}\mathbf{D}^{-1}\mathbf{\Pi}^T\mathcal{A}$ because $\sigma(\mathcal{B}\mathcal{A})\backslash\{0\} = \sigma(\mathcal{A}\mathcal{B})\backslash\{0\}$. In fact, from Theorem 4.7, we can see that the above iterative method is equivalent to

$$u^{\text{new}} = u^{\text{old}} + \mathbf{\Pi}\mathbf{D}^{-1}\mathbf{\Pi}^T(f - \mathcal{A}u^{\text{old}}) = u^{\text{old}} + \sum_{j=1}^{J}\mathcal{I}_j\mathcal{A}_j^{-1}\mathcal{I}_j^T\left(f - \mathcal{A}u^{\text{old}}\right).$$

We immediately recognize that this is the PSC method (or the additive Schwarz method) with exact subspace solvers. □

ex:GS-ssc **Example 4.9** (Block GS method and SSC)**.** Similar to the above example, we find that the block GS method is just the SSC method (or the multiplicative Schwarz method) for the original problem. We now apply the block GS method for the expanded system (4.8), i.e.,

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

We can rewrite this method as

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

Hence we have

$$\mathbf{D}\mathbf{u}^{\text{new}} = \mathbf{D}\mathbf{u}^{\text{old}} + \mathbf{f} - \mathbf{L}\mathbf{u}^{\text{new}} - (\mathbf{D}+\mathbf{U})\mathbf{u}^{\text{old}};$$

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 GS 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 GS method. From the above equation, we notice that the block GS method is just the SSC method with exact subspace solvers $\mathcal{S}_j = \mathcal{A}_j^{-1}$ for the original linear equation (4.1). □

4.2.3 Convergence analysis of block solvers

Motived by the weighted Jacobi and GS methods, we assume that there is an invertible diagonal smoother or local relaxation method $\mathbf{S}$ for solving $\mathbf{A}\mathbf{u} = \mathbf{f}$. Similar to the generalized GS method presented in §2.1, we define a generalized or modified block GS method:

$$\mathbf{B} := \big(\mathbf{S}^{-1} + \mathbf{L}\big)^{-1}. \tag{4.10}$$ eqn:blockMGS

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

$$\left(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}\right) = \left(\mathbf{B}^{-1}\mathbf{K}^{-1}\mathbf{B}^{-T}\mathbf{v}, \mathbf{v}\right) = \left(\left(\mathbf{S}^{-1} + \mathbf{L}\right)\mathbf{K}^{-1}\left(\mathbf{S}^{-T} + \mathbf{U}\right)\mathbf{v},\, \mathbf{v}\right), \quad \forall \mathbf{v} \in \mathbf{V} \tag{4.11}$$ eqn:invBbar1

From the definition of $\mathbf{K}$, it is clear that $\mathbf{K}$ is diagonal and

$$\begin{aligned}\mathbf{K} &= (\mathbf{S}^{-T} + \mathbf{U}) + (\mathbf{S}^{-1} + \mathbf{L}) - (\mathbf{D} + \mathbf{L} + \mathbf{U}) \\ &= \mathbf{S}^{-T} + \mathbf{S}^{-1} - \mathbf{D} = \mathbf{S}^{-T}\left(\mathbf{S}^T + \mathbf{S} - \mathbf{S}^T\mathbf{D}\mathbf{S}\right)\mathbf{S}^{-1}.\end{aligned}$$

Hence, its inverse matrix is also diagonal and

$$\mathbf{K}^{-1} = \mathbf{S}(\mathbf{S}^T + \mathbf{S} - \mathbf{S}^T\mathbf{D}\mathbf{S})^{-1}\mathbf{S}^T. \tag{4.12}$$ eqn:invK

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:

$$\left(\overline{\mathbf{B}}^{-1}\mathbf{v}, \mathbf{v}\right) = (\mathbf{A}\mathbf{v}, \mathbf{v}) + \left(\mathbf{K}^{-1}(\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v},\, (\mathbf{D} + \mathbf{U} - \mathbf{S}^{-1})\mathbf{v}\right), \quad \forall \mathbf{v} \in \mathbf{V}. \tag{4.13}$$ eqn:invBbar0

Now we apply the estimates discussed in §2.1 (i.e., the general convergence rate estimate for SPSD problems[2]) and get the following convergence result:

thm:MBGS **Theorem 4.10** (Convergence rate of generalized block GS)**.** If $\mathbf{K} := \mathbf{S}^{-T} + \mathbf{S}^{-1} - \mathbf{D}$ is SPD, then the generalized block GS 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

sec:XZ

In the previous section, we have found that the SSC method for the original equation is equivalent to the block GS method for the expanded equation using the subspaces $\{V_j\}_{j=1}^J$. Now we try to analyze the convergence rate of the block GS 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 [68].

[2] In order to estimate the convergence rate of stationary iterative methods for 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 non-singular.

4.3.1 A technical lemma

Suppose $V = \sum_{j=1}^{J} V_j$. It is clear that $\mathbf{\Pi} : \mathbf{V} \mapsto V$ is surjective and $\mathbf{\Pi u} = \sum_{j=1}^{J} \mathcal{I}_j \mathbf{u}_j$. We have the following simple but useful lemma:

lemma:equiv **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

$$(\mathcal{B}^{-1}v, v) = \inf_{\substack{\mathbf{v}\in\mathbf{V}\\ \mathbf{\Pi v}=v}} (\mathbf{B}^{-1}\mathbf{v}, \mathbf{v}), \quad \forall v \in V.$$

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

$$0 = (\mathcal{B}v, v) = (\mathbf{B\Pi}^T v, \mathbf{\Pi}^T v) \quad \Longrightarrow \quad \mathbf{\Pi}^T v = 0 \quad \Longrightarrow \quad v \in \mathrm{null}(\mathbf{\Pi}^T) = \mathrm{range}(\mathbf{\Pi})^{\perp}.$$

Since $\mathbf{\Pi}$ is surjective (onto), we have $v = 0$. This proves the iterator $\mathcal{B}$ is SPD.

Define $\mathbf{v}_* := \mathbf{B\Pi}^T\mathcal{B}^{-1}v$. It is easy to see that

$$\mathbf{\Pi v}_* = \mathbf{\Pi B\Pi}^T\mathcal{B}^{-1}v = \mathcal{B}\mathcal{B}^{-1}v = v, \quad \forall v \in V,$$

and

$$(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{w}) = (\mathbf{\Pi}^T\mathcal{B}^{-1}v, \mathbf{w}) = (\mathcal{B}^{-1}v, \mathbf{\Pi w}), \quad \forall \mathbf{w} \in \mathbf{V}.$$

If $\mathbf{w} \in \mathrm{null}(\mathbf{\Pi})$, 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 $\mathbf{w} \in \mathrm{null}(\mathbf{\Pi})$. 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

$$\begin{aligned}\inf_{\substack{\mathbf{v}\in\mathbf{V}\\ \mathbf{\Pi v}=v}} (\mathbf{B}^{-1}\mathbf{v}, \mathbf{v}) &= \big(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{v}_*\big) + \inf_{\mathbf{w}\in\mathrm{null}(\mathbf{\Pi})} \big(\mathbf{B}^{-1}\mathbf{w}, \mathbf{w}\big) \\ &= \big(\mathbf{B}^{-1}\mathbf{v}_*, \mathbf{v}_*\big) = \big(\mathbf{\Pi}^T\mathcal{B}^{-1}v, \mathbf{B\Pi}^T\mathcal{B}^{-1}v\big) = \big(\mathcal{B}^{-1}v, v\big).\end{aligned}$$

Hence the result. □

We can then derive the following expression for the inverse of the PSC preconditioner (see Example 4.8), which can be found in [192, 197, 106, 202].

em:InversePSC **Lemma 4.12.** Assume that all $\mathcal{S}_j$'s are SPD. Then

$$(\mathcal{B}_{\mathrm{PSC}}^{-1}v, v) = \inf_{\sum_j v_j = v} \sum_{j=1}^{J} (\mathcal{S}_j^{-1}v_j, v_j), \quad \forall\, v \in V.$$

Apparently, if $\mathcal{S}_j = \mathcal{A}_j^{-1}$, the above identity in Lemma 4.12 reads

$$(\mathcal{B}_{\mathrm{PSC}}^{-1} v, v) = \inf_{\sum_j v_j = v} \sum_{j=1}^{J} (v_j, v_j)_{\mathcal{A}_j}, \quad \forall\, v \in V.$$

Remark 4.13 (Minimizer for the expanded system)**.** From the above proof, we observe that

$$\mathbf{v}_* = \mathbf{B}\mathbf{\Pi}^T \mathcal{B}^{-1} v$$

is the minimizer of $\inf_{\substack{\mathbf{v}\in\mathbf{V}\\ \mathbf{\Pi}\mathbf{v}=v}} (\mathbf{B}^{-1}\mathbf{v}, \mathbf{v})$. □

Remark 4.14 (Auxiliary space method)**.** The above lemma on relation between the expanded problem and the original problem can also be extended to the following *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 on the auxiliary space method in §4.5. □

4.3.2 The XZ identity

We now present the XZ identity originally proved by Xu and Zikatanov [200] which gives the exact convergence rate of the SSC method.

:xzidentityc0 **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

$$\|\mathcal{I} - \mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2 = 1 - \frac{1}{1 + c_0} = 1 - \frac{1}{c_1}, \tag{4.14}$$ eq:xzidentity

where

$$c_0 := \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \|\mathcal{S}_j^T \mathbf{w}_j\|_{\mathcal{S}_j^{-1}}^2 \tag{4.15}$$ eq:xzc0

and

$$c_1 := \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \left\| \overline{\mathcal{S}}_j \mathcal{S}_j^{-1} \mathbf{v}_j + \mathcal{S}_j^T \mathbf{w}_j \right\|_{\overline{\mathcal{S}}_j^{-1}}^2. \tag{4.16}$$ eq:xzc1

Proof. (1) 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 calculations, we get

$$\begin{aligned}(\mathbf{D}+\mathbf{U})\,\mathbf{v} &= \Big(\sum_{j\geqslant 1}\mathcal{Q}_1\mathcal{A}\mathcal{Q}_j^T\mathbf{v}_j,\ \sum_{j\geqslant 2}\mathcal{Q}_2\mathcal{A}\mathcal{Q}_j^T\mathbf{v}_j,\ \cdots\Big)^T\\ &= \Big(\sum_{j\geqslant 1}\mathcal{A}_1\mathit{\Pi}_1\mathcal{I}_j\mathbf{v}_j,\ \sum_{j\geqslant 2}\mathcal{A}_2\mathit{\Pi}_2\mathcal{I}_j\mathbf{v}_j,\ \cdots\Big)^T\\ &= \Big(\mathcal{A}_1\mathit{\Pi}_1\sum_{j\geqslant 1}\mathbf{v}_j,\ \mathcal{A}_2\mathit{\Pi}_2\sum_{j\geqslant 2}\mathbf{v}_j,\ \cdots\Big)^T.\end{aligned}$$

Hence we can denote

$$\big(\mathbf{D}+\mathbf{U}-\mathbf{S}^{-1}\big)\,\mathbf{v}=(\mathbf{w}_1,\mathbf{w}_2,\ldots,\mathbf{w}_J)^T,\quad \text{with}\ \ \mathbf{w}_j:=\mathcal{A}_j\mathit{\Pi}_j\sum_{i\geqslant j}\mathbf{v}_i-\mathcal{S}_j^{-1}\mathbf{v}_j.$$

Due to (4.12) and the fact that $\mathbf{K}$ is diagonal, we have

$$\Big(\mathbf{K}^{-1}\big(\mathbf{D}+\mathbf{U}-\mathbf{S}^{-1}\big)\mathbf{v},\ \big(\mathbf{D}+\mathbf{U}-\mathbf{S}^{-1}\big)\mathbf{v}\Big)=\sum_{j=1}^{J}\Big(\mathcal{S}_j\overline{\mathcal{S}}_j^{-1}\mathcal{S}_j^T\mathbf{w}_j,\mathbf{w}_j\Big)=\sum_{j=1}^{J}\big\|\mathcal{S}_j^T\mathbf{w}_j\big\|^2_{\overline{\mathcal{S}}_j^{-1}},$$

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

$$\sup_{\|v\|_{\mathcal{A}}=1}\inf_{\mathbf{\Pi v}=v}\Big(\overline{\mathbf{B}}^{-1}\mathbf{v},\mathbf{v}\Big)=1+\sup_{\|v\|_{\mathcal{A}}=1}\inf_{\mathbf{\Pi v}=v}\sum_{j=1}^{J}\big\|\mathcal{S}_j^T\mathbf{w}_j\big\|^2_{\overline{\mathcal{S}}_j^{-1}}.$$

By applying Theorem 2.16 and Lemma 4.11, we know

$$\|\mathcal{I}-\mathcal{B}\mathcal{A}\|^2_{\mathcal{A}}=1-\Big(\sup_{\|v\|_{\mathcal{A}}=1}\big(\overline{\mathcal{B}}^{-1}v,v\big)\Big)^{-1}=1-\Big(\sup_{\|v\|_{\mathcal{A}}=1}\inf_{\mathbf{\Pi v}=v}\big(\overline{\mathbf{B}}^{-1}\mathbf{v},\mathbf{v}\big)\Big)^{-1}.\tag{4.17}$$ eqn:c1Bbar

This gives the desired estimate for the constant c_0.

(2) On the other hand, from (4.11), we have

$$\begin{aligned}\Big(\overline{\mathbf{B}}^{-1}\mathbf{v},\mathbf{v}\Big) &= \Big(\mathbf{K}^{-1}\big(\mathbf{S}^{-T}+\mathbf{U}\big)\mathbf{v},\ \big(\mathbf{S}^{-T}+\mathbf{U}\big)\mathbf{v}\Big)\\ &= \sum_{j=1}^{J}\Big\|\big(\mathcal{S}_j^{-1}+\mathcal{S}_j^{-T}-\mathcal{A}_j\big)^{-\frac{1}{2}}\big(\mathcal{S}_j^{-T}\mathbf{v}_j+\sum_{i>j}\mathcal{Q}_j\mathcal{A}\mathcal{I}_i\mathbf{v}_i\big)\Big\|^2.\end{aligned}\tag{4.18}$$

We notice that

$$\begin{aligned}\mathcal{S}_j^{-T}\mathbf{v}_j+\sum_{i>j}\mathcal{Q}_j\mathcal{A}\mathcal{I}_i\mathbf{v}_i &= \mathcal{S}_j^{-T}\mathbf{v}_j+\mathcal{A}_j\mathit{\Pi}_j\sum_{i>j}\mathbf{v}_i=\big(\mathcal{S}_j^{-T}+\mathcal{S}_j^{-1}-\mathcal{A}_j\big)\mathbf{v}_j+\mathbf{w}_j\\ &= \mathcal{S}_j^{-T}\overline{\mathcal{S}}_j\mathcal{S}_j^{-1}\mathbf{v}_j+\mathbf{w}_j=\mathcal{S}_j^{-T}\Big(\overline{\mathcal{S}}_j\mathcal{S}_j^{-1}\mathbf{v}_j+\mathcal{S}_j^T\mathbf{w}_j\Big).\end{aligned}$$

Plugging this into the previous identity, we get

$$\Big(\overline{\mathbf{B}}^{-1}\mathbf{v},\mathbf{v}\Big)=\sum_{j=1}^{J}\Big\|\overline{\mathcal{S}}_j\mathcal{S}_j^{-1}\mathbf{v}_j+\mathcal{S}_j^T\mathbf{w}_j\Big\|^2_{\overline{\mathcal{S}}_j^{-1}}.$$

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 $\mathcal{T}_{\overline{\mathcal{S}}_j} := \overline{\mathcal{S}}_j\mathcal{A}_j = \mathcal{T}_j + \mathcal{T}_j^* - \mathcal{T}_j^*\mathcal{T}_j : V_j \mapsto V_j$ and we can rewrite the above estimate (4.16) in a slightly different form. Notice that, in (4.18),

$$\mathcal{S}_j^{-T}\mathbf{v}_j + \sum_{i>j}\mathcal{Q}_j\mathcal{A}\mathcal{I}_i\mathbf{v}_i = \mathcal{A}_j\big(\mathcal{S}_j^T\mathcal{A}_j\big)^{-1}\mathbf{v}_j + \mathcal{A}_j\Pi_j\sum_{i>j}\mathbf{v}_i = \mathcal{A}_j\Big[\big(\mathcal{T}_j^*\big)^{-1}\mathbf{v}_j + \Pi_j\sum_{i>j}\mathbf{v}_i\Big]$$

and

$$\big(\mathcal{S}_j^{-1} + \mathcal{S}_j^{-T} - \mathcal{A}_j\big)^{-1}\mathcal{A}_j = \big(\mathcal{T}_j^{-1} + (\mathcal{T}_j^*)^{-1} - \mathcal{I}_j\big)^{-1} = \mathcal{T}_j\mathcal{T}_{\overline{\mathcal{S}}_j}^{-1}\mathcal{T}_j^*.$$

Thus we have

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1}\inf_{\sum_j \mathbf{v}_j = v}\sum_{j=1}^{J}\Big\|\mathcal{T}_{\overline{\mathcal{S}}_j}^{-\frac{1}{2}}\Big(\mathbf{v}_j + \mathcal{T}_j^*\Pi_j\sum_{i>j}\mathbf{v}_i\Big)\Big\|_{\mathcal{A}_j}^2. \tag{4.19}$$ eq:xzc1T

□

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

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1}\big\|\mathcal{T}_{\overline{\mathcal{S}}}^{-\frac{1}{2}}v\big\|_{\mathcal{A}}^2 = \sup_{\|v\|_{\mathcal{A}}=1}\big((\overline{\mathcal{S}}\mathcal{A})^{-1}v, v\big)_{\mathcal{A}} = \sup_{\|v\|_{\mathcal{A}}=1}\big(\overline{\mathcal{S}}^{-1}v, v\big),$$

which is exactly the convergence rate given in Theorem 2.16. □

Example 4.18 (Two-grid method)**.** Theorem 3.38 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

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1}\big\|\mathcal{T}_{\overline{\mathcal{S}}}^{-\frac{1}{2}}(\mathcal{I} - \mathcal{Q}_{\overline{\mathcal{S}}^{-1}})v\big\|_{\mathcal{A}}^2,$$

which coincides with (3.34) in Theorem 3.38. For a complete proof of this result, we refer to Zikatanov [219]. □

When we solve each subspace problem exactly, the XZ identity is substantially simpler since $\mathcal{T}_j = \Pi_j : V \mapsto V_j$ in this case. This special case of the XZ identity is given in the following corollary.

cor:XZ-exact **Corollary 4.19** (SSC with exact subspace solvers)**.** If an exact subspace solver $\mathcal{S}_j = \mathcal{A}_j^{-1}$ is used for each subspace, then we have, in (4.14), that

$$c_0 = \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \Big\| \mathit{\Pi}_j \sum_{i>j} \mathbf{v}_i \Big\|_{\mathcal{A}_j}^2 \tag{4.20}$$ eq:c0p

and

$$c_1 = \sup_{\|v\|_{\mathcal{A}}=1} \inf_{\sum_j \mathbf{v}_j = v} \sum_{j=1}^{J} \Big\| \mathit{\Pi}_j \sum_{i\geqslant j} \mathbf{v}_i \Big\|_{\mathcal{A}_j}^2 . \tag{4.21}$$ eq:c1p

Remark 4.20 (Alternating projection method)**.** Provided that $\mathit{\Pi}_j : V \mapsto V_j$ $(j = 1, 2, \ldots, J)$, define $\Theta_j := \mathcal{I} - \mathit{\Pi}_j : V \mapsto V_j^{\perp} =: U_j$. Now we can define a projection

$$\Theta_0 : V \mapsto U_0, \qquad U_0 := \bigcap_{j=1}^{J} U_j.$$

We notice that $\Theta_j \Theta_0 = \Theta_0$. From the XZ identity with exact subspace solvers, we have

$$\Big\| \Pi_{j=J,\ldots,1} \Theta_j \Big\|_{\mathcal{A}}^2 = \Big\| \Pi_{j=J,\ldots,1} (\mathcal{I} - \mathit{\Pi}_j) \Big\|_{\mathcal{A}}^2 = \frac{c_0}{1+c_0}.$$

We immediately see that

$$\Big\| \Pi_{j=J,\ldots,1} \Theta_j (\mathcal{I} - \Theta_0) v \Big\|_{\mathcal{A}}^2 \leqslant \frac{c_0}{1+c_0} \Big\| (\mathcal{I} - \Theta_0) v \Big\|_{\mathcal{A}}^2.$$

Hence,

$$\Big\| \left(\Pi_{j=J,\ldots,1} \Theta_j - \Theta_0 \right) v \Big\|_{\mathcal{A}}^2 \leqslant \frac{c_0}{1+c_0} \|v\|_{\mathcal{A}}^2.$$

Besides, $(\Pi_{j=J,\ldots,1} \Theta_j - \Theta_0)^k = (\Pi_{j=J,\ldots,1} \Theta_j)^k - \Theta_0$. We obtain that $\lim_{k\to\infty} (\Pi_{j=J,\ldots,1} \Theta_j)^k = \Theta_0$, which proves convergence of the method of alternating projections by von Neumann. □

4.4 Convergence analysis of PSC

sec:ConvPSC

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.

4.4.1 Relating PSC to SSC

The following theorem shows the relation between the PSC and SSC methods.

thm:PSCvsSSC **Theorem 4.21** (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

$$\begin{aligned}\|v\|_{\mathcal{A}}^2 &= \sum_{k,j=1}^{J} (v_k, v_j)_{\mathcal{A}} = \sum_{k=1}^{J} \Big((v_k, v_k)_{\mathcal{A}} + 2 \sum_{j>k}^{J} (v_k, v_j)_{\mathcal{A}} \Big) \\ &= 2 \sum_{k=1}^{J} \sum_{j \geqslant k}^{J} (v_k, v_j)_{\mathcal{A}} - \sum_{k=1}^{J} (v_k, v_k)_{\mathcal{A}}.\end{aligned}$$

Hence, since Π_k is an $\mathcal{A}$-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 4.12, Corollary 4.19, and (4.17), it gives the first inequality. The second one is also easy; see HW 4.5. □

Remark 4.22 (From sequential method to parallel method)**.** This theorem shows that, if the SSC method works well as an iterative method, then the PSC method based on the same space decomposition should also work, as a preconditioner. □

4.4.2 Condition number of PSC

Next, we give a direct analysis of the condition number of the PSC method. In order to obtain estimates on the condition number of the preconditioned systems, we first give the following assumptions:

Assump:MSC **Assumption 4.23** (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

$$\sum_{j=1}^{J} \big(\mathcal{S}_j^{-1} v_j, v_j \big) \leqslant K_1 (\mathcal{A}v, v); \tag{4.22}$$ assump:MSC1

2. For any $u, v \in V$,

$$\sum_{(i,j)} \big(\mathcal{T}_i u, \mathcal{T}_j v \big)_{\mathcal{A}} \leqslant K_2 \Big(\sum_{i=1}^{J} (\mathcal{T}_i u, u)_{\mathcal{A}} \Big)^{\frac{1}{2}} \Big(\sum_{j=1}^{J} (\mathcal{T}_j v, v)_{\mathcal{A}} \Big)^{\frac{1}{2}}. \tag{4.23}$$ assump:MSC2

thm:PSC **Theorem 4.24** (Condition number of PSC)**.** If the above Assumption 4.23 holds true, the PSC method (4.7) satisfies

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant K_1 K_2.$$

Proof. (1) For any $v \in V$, suppose that $v = \sum_{j=1}^{J} v_j$ is a decomposition that satisfies the first condition of Assumption 4.23. 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 2.48 and 2.49 yield the desired estimate. □

According to Theorem 4.24, 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 be obtained for SSC as well.

Remark 4.25 (Similar estimate for SSC)**.** In fact, with the same assumptions (Assumption 4.23), we can also show that the SSC method also converges with

$$\|\mathcal{I}-\mathcal{B}\mathcal{A}\|_{\mathcal{A}}^2 \leqslant 1-\frac{2-\omega_1}{K_1(1+K_2)^2} \quad \text{and} \quad \omega_1 := \max_j \rho(\mathcal{S}_j\mathcal{A}_j) = \max_j \rho(\mathcal{T}_j). \tag{4.24}$$ eqn:omega1

Because a sharp result has been given in §4.3, we will just leave the proof to the readers (cf., for example, [196]). □

4.4.3 Estimates of K_1 and K_2 ⋆

Assumption 4.23 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.

lem:K1 **Lemma 4.26** (Estimates of K_1)**.** Assume that, for any $v \in V$, there exists 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

$$K_1 \leqslant C_1/\omega_0, \quad \text{where } \omega_0 := \min_{j=1,\dots,J} \{\lambda_{\min}(\mathcal{S}_j \mathcal{A}_j)\};$$

(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

$$K_1 \leqslant \hat{C}_1/\hat{\omega}_0, \quad \text{where } \hat{\omega}_0 := \min_{j=1,\dots,J} \{\rho_j \lambda_{\min}(\mathcal{S}_j)\}.$$

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

$$(\mathcal{S}_j^{1/2} \mathcal{A}_j \mathcal{S}_j^{1/2} \mathcal{S}_j^{-1/2} v_j, \mathcal{S}_j^{-1/2} v_j) \geqslant \omega_0 (\mathcal{S}_j^{-1} v_j, v_j), \quad j = 1, \dots, J.$$

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 $K_1 \leqslant C_1/\omega_0$.

(ii) Similar to the previous part, from the definition of $\hat{\omega}_0$, we have

$$\rho_j (v_j, v_j) = \rho_j (\mathcal{S}_j \mathcal{S}_j^{-1/2} v_j, \mathcal{S}_j^{-1/2} v_j) \geqslant \hat{\omega}_0 (\mathcal{S}_j^{-1} v_j, v_j), \quad j = 1, \dots, J.$$

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 $K_1 \leqslant \hat{C}_1/\hat{\omega}_0$. □

We introduce a nonnegative symmetric matrix

$$\Sigma = \big(\sigma_{i,j}\big) \in \mathbb{R}^{J\times J}, \tag{4.25}$$ eqn:Sigma

where each entry $\sigma_{i,j}$ is the smallest constant such that

$$\big(\mathcal{T}_i u, \mathcal{T}_j v\big)_{\mathcal{A}} \leqslant \omega_1 \sigma_{i,j} \big(\mathcal{T}_i u, u\big)_{\mathcal{A}}^{\frac{1}{2}} \big(\mathcal{T}_j v, v\big)_{\mathcal{A}}^{\frac{1}{2}}, \quad \forall\, u, v \in V. \tag{4.26}$$ eqn:Strengthe

It is clear that $0 \leqslant \sigma_{i,j} \leqslant 1$. ω_1 has been defined in (4.24).

lem:K2 **Lemma 4.27** (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 (4.25), 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 preconditioning ⋆

sec:auxspace

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 assumptions:

- Firstly, Π is stable
$$\|\Pi\tilde{v}\|_{\mathcal{A}} \leqslant C \|\tilde{v}\|_{\tilde{\mathcal{A}}}, \qquad \forall\, \tilde{v} \in \tilde{V}.$$
- Secondly, for any $v \in V$, there exists $\tilde{v} \in \tilde{V}$ such that $\Pi\tilde{v} = v$ and
$$c\|\tilde{v}\|_{\tilde{\mathcal{A}}} \leqslant \|v\|_{\mathcal{A}}.$$

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

$$\kappa(\mathcal{B}\mathcal{A}) \leqslant \left(\frac{C}{c}\right)^2 \kappa(\tilde{\mathcal{B}}\tilde{\mathcal{A}}).$$

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 [149, 198].

The fictitious domain method is a large class of methods which is usually employed 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 [102] for details.

4.6 Homework problems

HW 4.1. Prove the statements in Remark 4.5.

hw:B-spd **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:bGS **HW 4.3.** Show that the block GS method for the expanded system is just the SSC method for the original problem.

hw:thm42 **HW 4.4.** Prove Theorem 4.10.

hw:PSCvsSSC **HW 4.5.** Prove Theorem 4.21. What is the constant c_*?

Part II

Examples of Multilevel Iterative Methods

Chapter 5

Subspace Correction Preconditioning

ch:examples

In Chapter 4, we discussed linear stationary iterative methods in the method of subspace corrections (MSC) framework. In this chapter, we provide examples of multilevel methods and analyze their convergence within the subspace corrections framework.

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.

5.1.1 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 2.5. Suppose we have a finite-dimensional coarse space $V_0 \subset \mathscr{V}$ on a quasi-uniform mesh of meshsize $H = \operatorname{diam}(\Omega_j)$. We notice that $\hat{\Omega}_j$ contains and is slightly bigger than Ω_j. This results in an overlapping domain decomposition. We shall the maximal number of overlaps is independent of the meshsize H.

Apparently, this yields 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 sub-domain as well as on the coarse space gives an *abstract multiplicative Schwarz* method[1].

We first define a partition of unity function $\theta_j \in C^1(\overline{\Omega})$ $(j = 1, \ldots, J)$ such that

[1] It is an abstract algorithm because we did not discretize each sub-domain problems.

(1) $0 \leqslant \theta_j \leqslant 1$ and $\sum_{j=1}^{J} \theta_j = 1$;

(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

$$v_0 \in V_0 \quad \text{and} \quad v_j := \theta_j (v - v_0) \in \mathscr{V}_j, \ \ j = 1, \ldots, J.$$

5.1.2 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 3.15 or the so-called simultaneous estimate in Remark 3.16, 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 XZ identity (Corollary 4.19), we get the following uniform convergence result.

prop:ConvDDM **Proposition 5.1** (Uniform convergence of two-level DDM)**.** The abstract domain decomposition method with coarse space corrections converges uniformly.

We leave the full proof to the interested readers; see HW 5.16.

:DDM-onelevel **Remark 5.2** (DDM without coarse space)**.** This analysis demonstrates the importance of the coarse space V_0. In fact, a similar proof shows the convergence rate depends on H^{-2} if we do not apply the coarse space corrections. □

5.2 HB preconditioner

sec:HB

In the previous section, we have seen a two-level domain decomposition method in the setting of subspace corrections. Now we investigate an example with multiple levels.

5.2.1 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^l 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

$$V_l := \big\{ v \in \mathscr{V} \,:\, v|_\tau \in \mathcal{P}_1(\tau),\ \forall\, \tau \in \mathcal{M}_l \big\}. \tag{5.1}$$ eqn:SpaceV

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

$$W_l := \big\{ v \in V_l \,:\, v(x) = 0,\ \forall\, x \in \mathring{G}(\mathcal{M}_{l-1}) \big\} \tag{5.2}$$ eqn:SpaceW

and obtain a multilevel space decomposition

$$V = W_0 \oplus W_1 \oplus \cdots \oplus W_L. \tag{5.3}$$ eqn:DecompW

Let $\mathcal{J}_l : V \mapsto V_l$ be the canonical interpolation operator and define $\mathcal{J}_{-1} := 0$. It is easy to see that

$$W_l = (\mathcal{I} - \mathcal{J}_{l-1})V_l = (\mathcal{J}_l - \mathcal{J}_{l-1})V, \quad l = 0, \ldots, L. \tag{5.4}$$ eqn:V-DecompW

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

$$\{\psi_{l,i}(x) : i = 1, \ldots, m_l, \ l = 0, \ldots, L\} \tag{5.5}$$ eqn:HB

is the so-called *hierarchical basis*.

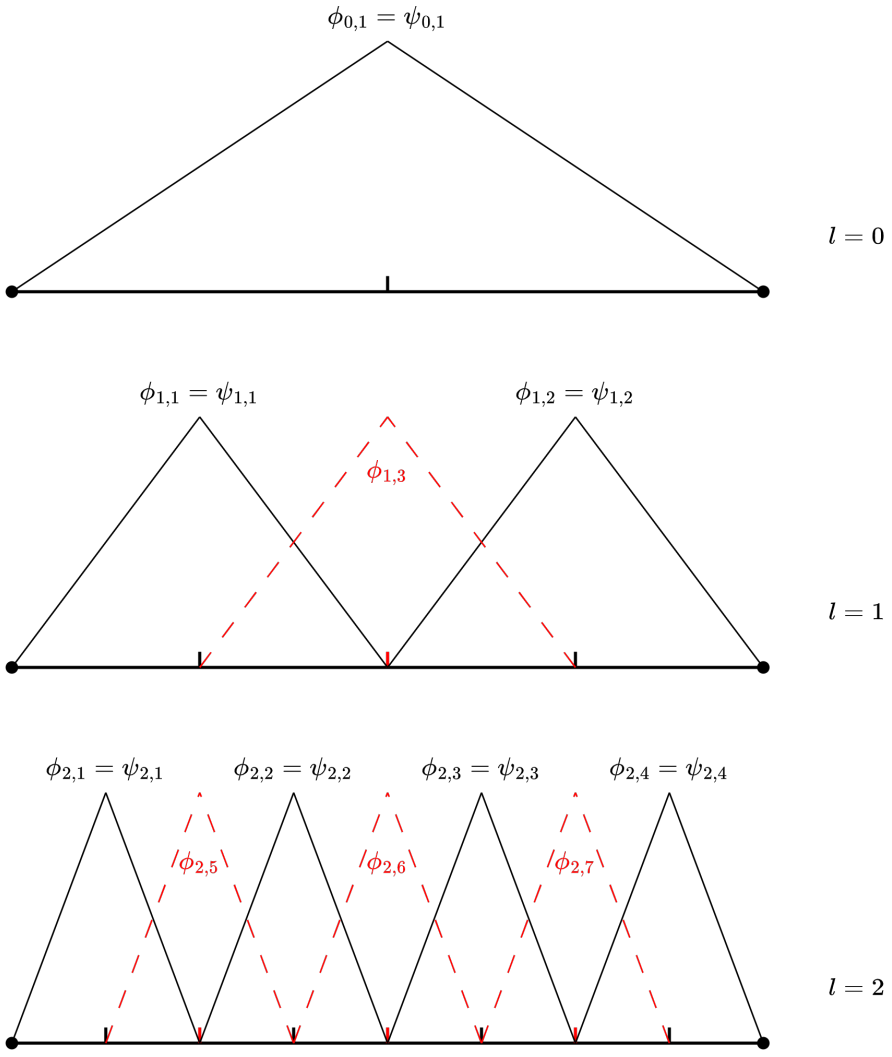


Figure 5.1: Regular and hierarchical basis functions in 1D. fig:MGbasis

Notice that the decomposition (5.3) is a direct sum and there is no redundancy in this decomposition at all.

5.2.2 Telescope expansions

According to (5.4), we can decompose a function $v \in V$ as

$$v = \sum_{l=0}^{L} (\mathcal{J}_l - \mathcal{J}_{l-1}) v, \tag{5.6}$$ eqn:decomp1

which is in the telescope sum form. Apparently, there are other ways for similar decomposition.

Using notations in Definition 4.1, we have

$$\begin{cases} \mathcal{A}_l : V_l \mapsto V_l & (\mathcal{A}_l u_l, v_l) = a[u_l, v_l], \quad \forall\, u_l, v_l \in V_l; \\ \mathcal{Q}_l : L^2 \mapsto V_l & (\mathcal{Q}_l u, v_l) = (u, v_l), \qquad \forall\, v_l \in V_l; \\ \Pi_l : \mathscr{V} \mapsto V_l & (\Pi_l u, v_l) = a[u, v_l], \qquad \forall\, v_l \in V_l. \end{cases} \tag{5.7}$$

We introduce a new notation $i \wedge j := \min(i, j)$. It is trivial to see that

$$\mathcal{Q}_i \mathcal{Q}_j = \mathcal{Q}_{i \wedge j}, \quad \Pi_i \Pi_j = \Pi_{i \wedge j}, \tag{5.8}$$ eqn:proj1

and

$$(\mathcal{Q}_i - \mathcal{Q}_{i-1})(\mathcal{Q}_j - \mathcal{Q}_{j-1}) = (\Pi_i - \Pi_{i-1})(\Pi_j - \Pi_{j-1}) = 0, \quad \forall\, i \neq j. \tag{5.9}$$ eqn:proj2

If we define $\mathcal{Q}_{-1} = \Pi_{-1} = 0$, we have the following possible decompositions

$$v = \sum_{l=0}^{L} (\mathcal{Q}_l - \mathcal{Q}_{l-1}) v = \sum_{l=0}^{L} (\Pi_l - \Pi_{l-1}) v. \tag{5.10}$$ eqn:decomp2

5.2.3 Hierarchical basis preconditioner

We now use the Richardson iteration discussed in §3.3 as the subspace solver for each one-dimensional subspace, 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

$$\mathcal{B}_{\mathrm{HB}} r = \sum_{j=1}^{N} \mathcal{S}_j \mathcal{Q}_j r, \tag{5.11}$$ eqn:HBprecond

which results in

$$\mathcal{B}_{\mathrm{HB}} r = \sum_{l=0}^{L} \left(h_l^{2-d} \sum_{i=1}^{m_l} (r, \psi_{l,i}) \psi_{l,i} \right). \tag{5.12}$$ eqn:HBprecond

And this is the explicit form of the well-known *hierarchical basis* (HB) *preconditioner* proposed by Yserentant [213].

We now analyze this preconditioner in the framework of PSC in §4.4. In order to do that, we need a few important estimates.

:StableInterp **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 $c_3(l) = \gamma^{l-L}$.

Proof. Using the interpolation error estimate in Proposition 3.12, 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 $|\cdot|_{1,\tau}$ as in Remark 1.6, the discrete Sobolev inequality Proposition 3.14, and the Poincaré inequality Proposition 1.10, we can obtain, for any $v \in V$, that

$$\begin{aligned} |\mathcal{J}_l v|_{1,\tau} &= |\mathcal{J}_l v - v_\tau|_{1,\tau} \lesssim h^{\frac{d}{2}-1}\|\mathcal{J}_l v - v_\tau\|_{\infty,\tau} \\ &\leqslant h^{\frac{d}{2}-1}\|v - v_\tau\|_{\infty,\tau} \lesssim C_d \|v - v_\tau\|_{1,\tau} \lesssim C_d |v|_{1,\tau}. \end{aligned}$$

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 $w \in W_l$ for any $0 \leqslant l \leqslant L$, we have

$$c_d^{-1}(l) h_l^{-2}(w, w) \lesssim a[w, w].$$

Compare this with the general Poincaré inequality in Proposition 1.11. Furthermore, from the inverse inequality Proposition 3.13, we always have

$$a[w, w] = |w|_1^2 \lesssim h_l^{-2}\|w\|_0^2 = h_l^{-2}(w, w).$$

Hence the operator $\mathcal{A}_l^{\mathrm{HB}}$ is "well-conditioned" up to a constant $c_d(l)$; compare this property with the standard Lagrange finite element basis case in Remark 3.17. □

5.2.4 Strengthened Cauchy–Schwarz inequality

lem:SCS1 **Lemma 5.5** (Inner product between two levels)**.** If $i \leqslant j$, we have

$$a[u, v] \lesssim \gamma^{\frac{j-i}{2}} h_j^{-1}|u|_1 \|v\|_0, \quad \forall\, u \in V_i, v \in V_j.$$

Proof. We first restrict our attention to an element $\tau_i \in \mathcal{M}_i$. For $v \in V_j$, there is a unique function $v_1 \in V_j$, 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 $u \in V_i$ 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 $\mathrm{supp}(v_0) \subset \overline{T}$. 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

$$\|\nabla u\|_{0,T \bigcap \tau_i} = \frac{|T \bigcap \tau_i|^{1/2}}{|\tau_i|^{1/2}} \|\nabla u\|_{0,\tau_i} \lesssim \left(\frac{h_i^{d-1} h_j}{h_i^d}\right)^{1/2} \|\nabla u\|_{0,\tau_i} \lesssim \gamma^{\frac{j-i}{2}} |u|_{1,\tau_i}.$$

Combining the above two inequalities, we have

$$\int_{\tau_i} \nabla u \cdot \nabla v = \int_{\tau_i} \nabla u \cdot \nabla v_0 \lesssim \gamma^{\frac{j-i}{2}} h_j^{-1} |u|_{1,\tau_i} \|v\|_{0,\tau_i}, \quad \forall \tau_i \in \mathcal{M}_i.$$

By the Cauchy–Schwarz inequality, we obtain the estimate:

$$\begin{aligned} a[u,v] &= \sum_{\tau_i \in \mathcal{M}_i} \int_{\tau_i} \nabla u \cdot \nabla v \lesssim \gamma^{\frac{j-i}{2}} h_j^{-1} \sum_{\tau_i \in \mathcal{M}_i} |u|_{1,\tau_i} \|v\|_{0,\tau_i} \\ &\leqslant \gamma^{\frac{j-i}{2}} h_j^{-1} \Big(\sum_{\tau_i \in \mathcal{M}_i} |u|_{1,\tau_i}^2 \Big)^{1/2} \Big(\sum_{\tau_i \in \mathcal{M}_i} \|v\|_{0,\tau_i}^2 \Big)^{1/2} = \gamma^{\frac{j-i}{2}} h_j^{-1} |u|_1 \|v\|_0. \end{aligned}$$

Hence the result. □

lem:SCS2 **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

$$a[u_i, v_j] \lesssim \gamma^{\frac{|i-j|}{2}} \|u_i\|_{\mathcal{A}} \|v_j\|_{\mathcal{A}}.$$

Proof. If $j \geqslant i$, we have $v_j = (\mathcal{J}_j - \mathcal{J}_{j-1})v = v_j - \mathcal{J}_{j-1} v_j$. So $\|v_j\|_0 = \|v_j - \mathcal{J}_{j-1} v_j\|_0 \lesssim h_j \|v_j\|_{\mathcal{A}}$ follows from Proposition 3.12. If $i \geqslant j$, we can argue in a similar way. Hence the result follows directly from Lemma 5.5. □

If $v \in V$, Lemma 5.6 yields

$$\begin{aligned} |v|_1^2 &= \sum_{l,m} \big(\nabla(\mathcal{J}_l - \mathcal{J}_{l-1})v, \nabla(\mathcal{J}_m - \mathcal{J}_{m-1})v\big) \\ &\lesssim \sum_{l,m} \gamma^{\frac{|l-m|}{2}} \|(\mathcal{J}_l - \mathcal{J}_{l-1})v\|_1 \|(\mathcal{J}_m - \mathcal{J}_{m-1})v\|_1 \lesssim \sum_{l} \|(\mathcal{J}_l - \mathcal{J}_{l-1})v\|_1^2. \end{aligned}$$

On the right-hand side, we have the summation of components from all levels.

lem:SCS3 **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

$$(u_i, \mathcal{T}_j v)_{\mathcal{A}} \lesssim \gamma^{\frac{j-i}{2}} \|u_i\|_{\mathcal{A}} \|v\|_{\mathcal{A}}, \quad \forall\, u_i \in V_i, v \in V. \tag{5.13}$$ eqn:SCSIneq1

For $0 \leqslant i, j \leqslant L$, we have the strengthened Cauchy–Schwarz inequality

$$(\mathcal{T}_i u, \mathcal{T}_j v)_{\mathcal{A}} \lesssim \gamma^{\frac{|j-i|}{4}} (\mathcal{T}_i u, u)_{\mathcal{A}}^{\frac{1}{2}} (\mathcal{T}_j v, v)_{\mathcal{A}}^{\frac{1}{2}}, \quad \forall\, u, v \in V. \tag{5.14}$$ eqn:SCSIneq2

Proof. By applying Lemma 5.5, we get

$$(u_i, \mathcal{T}_j v)_{\mathcal{A}} = a[u_i, \mathcal{T}_j v] \lesssim \gamma^{\frac{j-i}{2}} h_j^{-1} \|u_i\|_{\mathcal{A}} \|\mathcal{T}_j v\|_0.$$

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 (5.13).

Consider the case when $j \geqslant i$. By the Cauchy–Schwarz inequality and the inequality (5.13), we get

$$(\mathcal{T}_i u, \mathcal{T}_j v)_{\mathcal{A}} \leqslant (\mathcal{T}_j\mathcal{T}_i u, \mathcal{T}_i u)_{\mathcal{A}}^{\frac{1}{2}} (\mathcal{T}_j v, v)_{\mathcal{A}}^{\frac{1}{2}} \lesssim \gamma^{\frac{j-i}{4}} \|\mathcal{T}_i u\|_{\mathcal{A}} (\mathcal{T}_j v, v)_{\mathcal{A}}^{\frac{1}{2}}.$$

Also observe that, a special case of the above inequality is $(\mathcal{T}_i u, \mathcal{T}_i u)_{\mathcal{A}} \lesssim \|\mathcal{T}_i u\|_{\mathcal{A}} (\mathcal{T}_i u, u)_{\mathcal{A}}^{\frac{1}{2}}$ and the second inequality (5.14) follows immediately. □

5.2.5 Convergence analysis of HB preconditioner

thm:rateHB **Theorem 5.8** (Convergence of HB preconditioner)**.** The multilevel PSC preconditioner $\mathcal{B}_{\mathrm{HB}}$ defined in (5.11) 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 $(\mathcal{J}_l - \mathcal{J}_{l-1})v \in W_l$ and $\mathcal{J}_{-1} = 0$. With careful calculations, the inverse estimate Proposition 3.13 and stability Lemma 5.3 ($\mathcal{J}_l = \Pi_l$ in 1D) yield

$$\sum_{l=0}^{L} \|v_l\|_{\mathcal{A}}^2 \lesssim \sum_{l=0}^{L} h_l^{-2} \|v_l\|_0^2 \lesssim C_d(h) \|v\|_{\mathcal{A}}^2. \tag{5.15}$$ HB:lowerbd

On the other hand, we know

$$\hat{\omega}_0 = \min_l \rho_l \lambda_{\min}(\mathcal{S}_l) \cong 1.$$

Therefore $K_1 \lesssim C_d(h)$ due to Lemma 4.26. The strengthened Cauchy–Schwarz inequality (5.14) and Lemma 4.27 give that $K_2 \lesssim 1$. The convergence result then follows directly from Theorem 4.24. □

This theorem shows the HB preconditioner converges rapidly when combined with some Krylov subspace method. However, the condition number still depends on the mesh size h, especially in 3D. We now briefly discuss how such dependence can be eliminated.

Define an operator $\mathcal{H} : V \mapsto V$ such that

$$(\mathcal{H}v, w) := \sum_{l=0}^{L} \sum_{x_i \in \mathring{G}(\mathcal{M}_l)\backslash\mathring{G}(\mathcal{M}_{l-1})} h_l^{d-2}\Big(\big(\mathcal{J}_l v - \mathcal{J}_{l-1}v\big)(x_i),\ \big(\mathcal{J}_l w - \mathcal{J}_{l-1}w\big)(x_i)\Big).$$

Hence we get

$$(\mathcal{H}v, v) = \sum_{l=0}^{L} \sum_{x_i \in \mathring{G}(\mathcal{M}_l)\backslash\mathring{G}(\mathcal{M}_{l-1})} h_l^{d-2}\Big|\big(\mathcal{J}_l v - \mathcal{J}_{l-1}v\big)(x_i)\Big|^2, \quad \forall\, v \in V.$$

In fact, this operator is the inverse of the HB preconditioner, i.e., $\mathcal{H} = \mathcal{B}_{\mathrm{HB}}^{-1}$; see [214]. In fact, in the proof of Theorem 5.8, we have shown the following norm equivalence:

$$\|v\|_{\mathcal{A}}^2 \lesssim (\mathcal{H}v, v) = \sum_{l=0}^{L} h_l^{-2}\|(\mathcal{J}_l - \mathcal{J}_{l-1})v\|_0^2 \lesssim C_d(h)\|v\|_{\mathcal{A}}^2. \tag{5.16}$$ eqn:HBYser

5.3 BPX preconditioner

sec:BPX

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

$$\begin{aligned}\|v\|_{\mathcal{A}}^2 &= \Big\|\sum_{l=0}^{L}(\Pi_l - \Pi_{l-1})v\Big\|_{\mathcal{A}}^2 = \sum_{0\leqslant i,j\leqslant L} a\big[(\Pi_i - \Pi_{i-1})v, (\Pi_j - \Pi_{j-1})v\big] \\ &= \sum_{l=0}^{L} a\big[(\Pi_l - \Pi_{l-1})v, (\Pi_l - \Pi_{l-1})v\big] = \sum_{l=0}^{L}\Big|(\Pi_l - \Pi_{l-1})v\Big|_1^2. \end{aligned} \tag{5.17}$$

Notice that this is corresponding to the telescope sum of the Ritz projections in (5.10). Motivated by the above norm equivalence (5.17) and relations (5.16), one can easily construct a "better" multilevel PSC method $\mathcal{B} = \sum_{j=1}^{J} \mathcal{S}_j \Pi_j$. However, Π_j is not "friendly" for computation in general

except for $d = 1$ in which $\Pi_j = \mathcal{J}_j$ is just the interpolation[2]. Now we explore the idea of telescope expansion using the L^2-projection (5.9) instead of the interpolation or the Ritz projection. And it turns out to give rise to the well-known BPX preconditioner.

In the previous section, along with the hierarchical basis decomposition, we have also obtained a natural multilevel space decomposition

$$V = \sum_{l=0}^{L} V_l = \sum_{l=0}^{L} \sum_{i=1}^{n_l} V_{l,i}, \tag{5.18}$$ eqn:MGdecomp

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 critical for optimal convergence rate.

Based on the multilevel space decomposition (5.18), we can construct multilevel subspace correction methods. Among them, the most prominent (multilevel) example of PSC methods is the BPX preconditioner [49], i.e.,

$$\mathcal{B}_{\mathrm{BPX}} = \sum_{j=1}^{J} \mathcal{S}_j \mathcal{Q}_j, \quad \text{with } J = \sum_{l=0}^{L} n_l, \tag{5.19}$$ eqn:Multileve

which is computationally more appealing and converges uniformly. We notice that the HB and BPX preconditioners are in the same form except on different space decompositions. They both belong to the class of so-called *multilevel nodal basis* preconditioners.

5.3.1 Norm equivalence

We will now show why the BPX preconditioner works better than the HB preconditioner. We note the HB preconditioner is not optimal for dimensions higher than one due to the worsened H^1-stability property of the interpolations. We would expect improved stability properties for L^2-projections.

lem:telescope **Lemma 5.9** (Terms in L^2-projection telescope sum)**.** For any $v \in V$, we have

$$\left|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right|_1 \cong h_l^{-1} \left\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right\|_0.$$

Proof. Using the inverse inequality, namely Proposition 3.13, we get

$$\left|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right|_1 \lesssim h_l^{-1} \left\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\right\|_0.$$

Applying the weighted L^2-estimate in Proposition 3.15 to the following trivial relation

$$(\mathcal{Q}_l - \mathcal{Q}_{l-1})v = (\mathcal{I} - \mathcal{Q}_{l-1})(\mathcal{Q}_l - \mathcal{Q}_{l-1})v,$$

we can obtain the lower bound. □

[2]Note that this is equivalent to the HB preconditioner in 1D.

lem:SCS4 **Lemma 5.10** (Strengthened Cauchy–Schwarz inequality)**.** 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

$$a[u_i, v_j] \lesssim \gamma^{\frac{|i-j|}{2}} \|u_i\|_{\mathcal{A}} \|v_j\|_{\mathcal{A}}.$$

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. □

em:norm-equiv **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 the L^2-projection from V to V_l, we have $\|\mathcal{Q}_l v\|_0 \leqslant \|v\|_0$, $\forall\, v \in L^2(\Omega)$. Furthermore, using Proposition 3.15, we obtain

$$\|\mathcal{Q}_l v\|_1 \leqslant \|v\|_1, \quad \forall\, v \in V.$$

By space interpolation, we have, for any $\sigma \in (0, \frac{1}{2})$, that

$$\|\mathcal{Q}_l v\|_\sigma \leqslant \|v\|_\sigma, \quad \forall\, v \in V.$$

Let $\alpha \in (\frac{1}{2}, 1)$. If $\mathit{\Pi}_l : V \mapsto V_l$ is the standard H^1-projection, the finite element theory gives

$$\|v - \mathit{\Pi}_l v\|_{1-\alpha} \lesssim h_l^\alpha \|v\|_1, \quad \forall\, v \in V. \tag{5.20}$$ eqn:GalerkinE

Let $v_i := (\mathit{\Pi}_i - \mathit{\Pi}_{i-1})v$. Note that $\rho_l = \rho(\mathcal{A}_l) \cong h_l^{-2}$. It is easy to show, with help from the inverse inequality (Proposition 3.13) and (5.20), 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 l, i, and j are all level indices and we can apply summation by parts.

We can show that $\sum_{i,j} \gamma^{\alpha|i-j|} \|v_i\|_1 \|v_j\|_1 \lesssim \sum_i \|v_i\|_1^2 \lesssim \|v\|_1^2$, 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 $\frac{1}{2} < \alpha < \frac{3}{2}$. □

5.3.2 Convergence analysis for BPX preconditioner

All subspaces in (5.18) 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:

$$\mathcal{S}_l v := \sum_{i=1}^{n_l} h_l^{2-d} (v, \phi_{l,i})\, \phi_{l,i} \quad (\approx \mathcal{S}_l^0 v). \tag{5.21}$$ eqn:Slv

This simplification helps us to reduce the cost of computation as well as implementation. Apparently, we have

$$(\mathcal{S}_l v, v) = h_l^{2-d} (\vec{v}, \vec{v}) = h_l^2 (v, v). \tag{5.22}$$ eqn:BPX-Smoot

We have seen that the Richardson method, the weighted Jacobi method, and the G-S method all satisfy similar conditions; see (3.24).

onRequirement **Remark 5.13** (Behavior of the smoother)**.** Note that the method (5.21) 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 corresponding PSC method yields the well-known *BPX preconditioner*

$$\mathcal{B}_{\mathrm{BPX}} = \sum_{l=0}^{L} \mathcal{S}_l \mathcal{Q}_l = \sum_{l=0}^{L} \mathcal{I}_l \mathcal{S}_l \mathcal{Q}_l = \sum_{l=0}^{L} \mathcal{I}_l \mathcal{S}_l \mathcal{I}_l^T \tag{5.23}$$ eqn:BPX

in operator form [49].

thm:BPX **Theorem 5.14** (Uniform convergence of BPX)**.** The BPX preconditioner (5.23) is uniformly convergent, i.e., $\kappa(\mathcal{B}_{\mathrm{BPX}}\mathcal{A}) \lesssim 1$.

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 (5.8) and (5.9), 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

$$(\tilde{\mathcal{A}}^{-1}v, v) = \sum_{l=0}^{L} h_l^2 (\mathcal{Q}_l v, v) - \sum_{l=0}^{L} h_l^2 (\mathcal{Q}_{l-1} v, v) = h_L^2 (\mathcal{Q}_L v, v) + \sum_{l=0}^{L-1} (1-\gamma^2) h_l^2 (\mathcal{Q}_l v, v).$$

On the other hand, we have

$$(\mathcal{B}_{\mathrm{BPX}} v, v) = \Big(\sum_{l=0}^{L} \mathcal{S}_l \mathcal{Q}_l v, v \Big) = \sum_{l=0}^{L} (\mathcal{S}_l \mathcal{Q}_l v, v) = \sum_{l=0}^{L} h_l^2 (\mathcal{Q}_l v, v).$$

Namely, $(\tilde{\mathcal{A}}^{-1}v, v) \cong (\mathcal{B}_{\mathrm{BPX}} v, v)$. That is to say, $(\mathcal{A}v, v) \cong (\tilde{\mathcal{A}}v, v) \cong (\mathcal{B}_{\mathrm{BPX}}^{-1} v, v)$. Hence it gives the uniform convergence result by Lemma 2.48. □

Remark 5.15 (Multilevel decomposition according to frequencies)**.** From the above analysis, we find that, for any $v \in V$,

$$\big|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\big|_1 \cong h_l^{-1} \big\|(\mathcal{Q}_l - \mathcal{Q}_{l-1})v\big\|_0 \quad \Longrightarrow \quad \|\nabla v_l\|_0 \cong \|h_l^{-1} v_l\|_0.$$

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 $\kappa(\tilde{\mathcal{A}}^{-1}\mathcal{A}) \lesssim 1$. □

5.3.3 Matrix representation of BPX

Using the matrix representation notations introduced in §3.2 and §3.5, the equations (3.16) and (3.39) in particular, we immediately obtain the matrix representation of the BPX method:

$$\underline{\mathcal{B}}\,\underline{u} = \underline{\mathcal{B}u} = \sum_{l=0}^{L} \underline{\mathcal{I}_l}\,\underline{\mathcal{S}_l}\,\underline{\mathcal{Q}_l}\,\underline{u} = \sum_{l=0}^{L} P_l \,(h_l^{2-d} M_l)\,(M_l^{-1} P_l^T M)\,\underline{u} = \sum_{l=0}^{L} h_l^{2-d} P_l\, P_l^T M\, \underline{u},$$

where P_l is the primal form of the natural embedding from V_l to V. In view of (3.17), we get the matrix form of the BPX preconditioner

$$B := \underline{\mathcal{B}} M^{-1} = \sum_{l=0}^{L} h_l^{2-d} P_l\, P_l^T. \tag{5.24}$$

eqn:MatBPX

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

hw:ConvDDM **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)?

hw:bpx-inv **Problem 5.17.** Let $\tilde{\mathcal{A}} := \sum_{l=0}^{L} h_l^{-2}(\mathcal{Q}_l - \mathcal{Q}_{l-1})$. Show that $\tilde{\mathcal{A}}^{-1} = \sum_{l=0}^{L} h_l^{2}(\mathcal{Q}_l - \mathcal{Q}_{l-1})$.

hw:BPX **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

ch:mg

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 [93] 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, but much cheaper to solve. This recursive process is repeated until a coarse-grid 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. [108, 51] noticed that this iterative procedure was considerably faster than standard relaxation methods and brought it to the attention of the western scientific community.

6.1 Geometric multigrid method

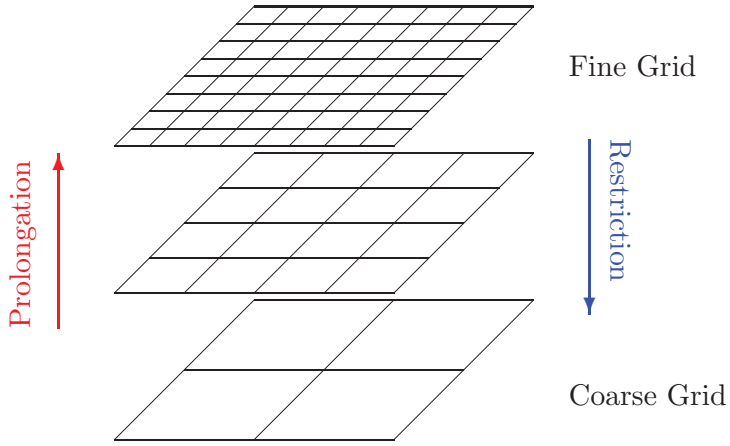
sec:GMG

The geometric multigrid (GMG) method is an optimal iterative solver for the linear algebraic systems (2.1) arising from some 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 approximated well on coarse spaces.

GMG establishes and exploits hierarchical structures. It exemplifies the *divide and conquer* approach, which has been applied in two-grid methods, as discussed in §3.4. Unfortunately, for large-scale problems, the coarse-grid problem might remain too large to be solved efficiently. This naturally leads to introducing more than two nested meshes.

- **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 approximate problems on coarse grids;
- **Prolongation**: Represent the correction computed on a coarser grid to a finer grid.



fig_multigrid

Figure 6.1: Pictorial illustration of a multigrid method with three grid levels.

6.1.1 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 as 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:

alg:V-cycle

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 $\mathcal{B}_l : V_l \mapsto V_l$, 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 $k = 1, 2, \ldots, m_1$, 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 $k = 1, 2, \ldots, m_2$, compute

$$v_l \leftarrow v_l + \mathcal{S}_l^T\big(r_l - \mathcal{A}_l v_l\big).$$

This algorithm is the so-called V(m_1,m_2)-cycle. The notation like V$(1,2)$ means the V-cycle multigrid with 1 pre-smoothing and 2 post-smoothing steps.

Remark 6.1 (Setup and solve phases)**.** Algorithm 6.1 gives a typical *solve* phase of multigrid methods. It relies on the hierarchical information (for example $\mathcal{A}_l$, $l = 0, 1, \ldots, L-1$) constructed by a procedure called the *setup* phase. Apparently, the setup phase only need to be called once and shared by the iterations in the solve phase. □

From the above algorithm, we can see this V-cycle multigrid method is just a generalization of Algorithm 3.2 (the abstract two-grid method). Clearly, the geometric multigrid method V(1,1)-cycle (using 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 (5.18). Furthermore, on each one-dimensional subspace $\tilde{V}_j$, the subspace problem is solved exactly.

According to Lemma 3.37, the error propagation 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 propagation operator for the MG V-cycle:

$$\mathcal{E}_L = \mathcal{I} - \mathcal{B}_L\mathcal{A}_L\mathit{\Pi}_L = \big(\mathcal{I} - \mathcal{S}_L^T\mathcal{A}_L\big)\cdots\big(\mathcal{I} - \mathcal{S}_1^T\mathcal{A}_1\big)\big(\mathcal{I} - \mathit{\Pi}_0\big)\big(\mathcal{I} - \mathcal{S}_1\mathcal{A}_1\big)\cdots\big(\mathcal{I} - \mathcal{S}_L\mathcal{A}_L\big).$$

6.1.2 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 (3.39), in turn, give the inter-grid transformations:

$$\hat{\mathcal{A}}_l = M_l\, \underline{\mathcal{A}_l} = M_l\, \underline{\mathcal{I}_l^T}\, \underline{\mathcal{A}}\, \underline{\mathcal{I}_l} = M_l\, \underline{\mathcal{I}_l^T}\, M^{-1} \hat{\mathcal{A}}\, \underline{\mathcal{I}_l} = \underline{\mathcal{I}_l}^T \hat{\mathcal{A}}\, \underline{\mathcal{I}_l}, \quad 0 \leqslant l < L.$$

Hence we get the matrix form of the coarse-level operator

$$\hat{\mathcal{A}}_l = P_l^T \hat{\mathcal{A}}\, P_l \quad \text{or} \quad A_l = P_l^T A P_l, \quad 0 \leqslant l < L. \tag{6.1}$$ eqn:Algebraic

6.1.3 Anisotropic problems ⋆

For GMG, error smoothness is in the usual geometric sense. However, this is not trivial for problems on unstructured meshes or with complex coefficients. A representative example is the second-order elliptic problem

$$-\epsilon u_{xx} - u_{yy} = f(x, y), \quad \forall\, (x, y) \in \Omega, \tag{6.2}$$ eqn:aniso

where $\epsilon > 0$ is usually small.

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, 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 observation suggests that the standard G-S method barely have any smoothing effect on the anisotropic problem when ϵ is small.

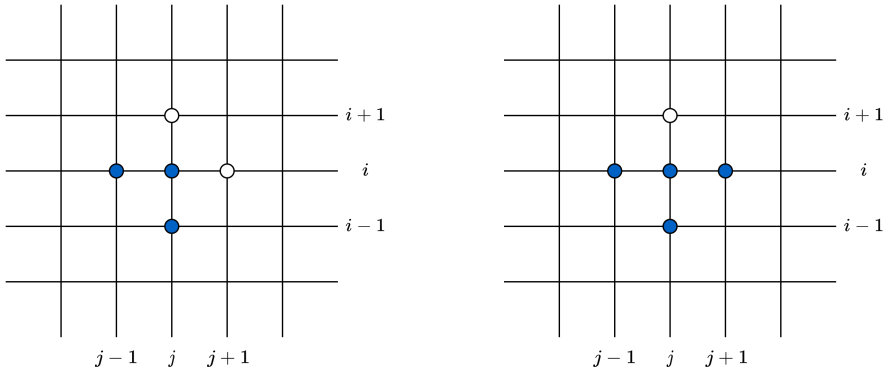


Figure 6.2: Standard (left) and line Gauss–Seidel (right) smoothers: Blue points have updated values and white points have old values.

fig_LGS

On the other hand, if we apply the line G-S smoother, things will be a lot different; see Figure 6.2. Suppose we apply the line smoother in natural ordering, 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 treatments like: (1) apply an line smoother (group all those y-variables corresponding to the same x-coordinate together), or (2) employ y-semi-coarsening (only coarse in the y-direction), or (3) construct 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.

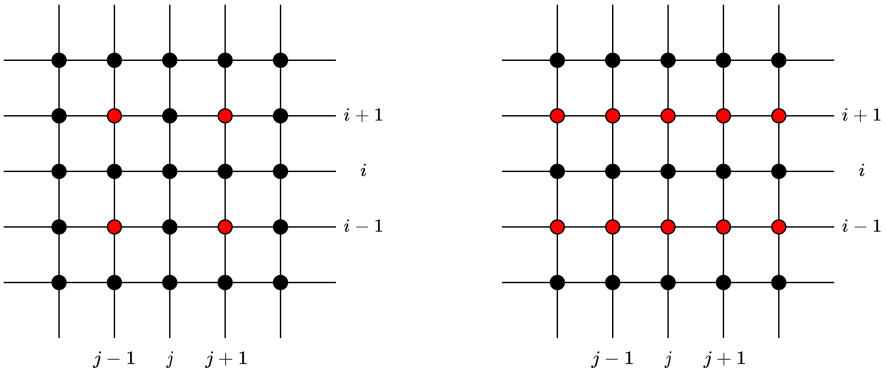


Figure 6.3: Examples of coarsening methods (Left: standard coarsening; Right: y-semi-coarsening): Red depicts coarse points and black depicts fine points.

ig_coarsening

This example illustrates a typical problem used by researchers to evaluate the robustness of multigrid methods as well as other iterative solvers. Other examples include problems with high-contrast coefficients, heterogeneous coefficients, anisotropic meshes, etc.

6.2 Convergence analysis of multigrid methods

sec:qualconv

In this section, we show the slash cycle or sawtooth cycle (i.e., /-cycle) converges uniformly (h-independently) using the XZ identity discussed previously. For simplicity, we will only discuss the proof in 1D here. Multidimensional cases and other MG methods can also be analyzed in the subspace correction framework, but the analysis is more technically involved; see [196].

6.2.1 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

$$v = \mathcal{J}_L v = \sum_{l=0}^{L} \big(\mathcal{J}_l - \mathcal{J}_{l-1}\big)v = \sum_{l=0}^{L} v_l. \tag{6.3}$$ eqn:HBdecomp

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

$$(\mathcal{I} - \mathcal{J}_l)v = \sum_{k=l+1}^{L} v_k = \sum_{k=l+1}^{L} \sum_{j=1}^{n_k} v_{k,j}$$

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.

rem:HB **Remark 6.2** (Relation with the HB preconditioner)**.** Note that several parts of the above analysis depend on the one-dimensional ($d = 1$) assumption for simplicity. In fact, the decomposition (6.3) used in this proof is the hierarchical basis (HB) decomposition discussed in §5.2. We have already seen that the HB method convergence rate is not actually optimal for multidimensional cases ($d > 1$). So the proof requires modification for higher dimensions. We will not explore the details of that approach here. Several alternative approaches in the literature prove the optimality of GMG methods; we briefly review them in the following subsection. □

6.2.2 Some historical remarks ⋆

The theoretical analysis in this note closely follows the argument of subspace corrections theory. We now briefly review the history of multigrid convergence theory. A comprehensive literature review is not possible here; interested readers should see the monographs [110, 139, 40, 65, 185, 189], survey papers [196, 215], and references therein for a more thorough treatment.

In the early 1960s, Fedorenko first introduced and analyzed the multigrid method for finite difference equations of the Poisson equation on a unit square [93, 94]. Bakhvalov extended the result to more complex cases with variable coefficients [10]. Nicolaides provided an analysis for finite element discretizations of second-order elliptic equations [151]. In the late 1970s, Hackbusch and Brandt made a major breakthrough, showing multigrid is highly efficient [108, 51]. Their seminal work popularized multigrid, motivating extensive research to develop a general convergence theory. The simplest case is a two-level hierarchy. Bank and Dupont developed a two-level hierarchical basis (HB) finite element method [13] and proved two-grid method convergence for finite elements [12]. Under certain conditions, their two-grid theory shows W-cycle (or more robust) multigrid with sufficient smoothing steps converges similarly to the two-grid method; see [12, 109, 110, 185]. However, this approach cannot prove the uniform convergence of V-cycle multigrid, which is more important in practice [109].

Hackbusch [109] and Braess and Hackbusch [38] first gave a general convergence theory for multigrid, including the V-cycle. The classical book by Hackbusch [110] 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* properties, namely,

$$(v_l, \mathcal{A}_l v_l) \lesssim (v_l, \mathcal{B}_l^{-1} v_l), \qquad \forall\, v_l \in V_l; \tag{6.4}$$ cond_smooth

$$(w_l, \mathcal{B}_l^{-1} w_l) \lesssim \kappa (w_l, \mathcal{A}_l w_l), \qquad \forall\, w_l \in W_l := \big\{ (\Pi_l - \Pi_{l-1}) v \,:\, v \in V \big\}. \tag{6.5}$$ cond_approx

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 contraction factor goes to zero as the number of smoothing steps increases. The approximation property (6.5) 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, [36, 16]. Bramble and Pasciak [42] introduced a *regularity and approximation* condition to show convergence of multigrid methods including the V-cycle for any positive m. Bank and Yserentant [16] 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 [213] and Bank, Dupont, and Yserentant [14] 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 [49] 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 [48, 47] 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 [84] 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 [158, 218, 196, 45, 35, 106].

Xu [196] 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 [215] reviewed the classical proof and the subspace correction proof for the convergence of multigrid methods. By combining the two convergence theories, Brenner [58] proved that contraction factor for some V-cycle methods decreases as number of smoothing steps increases without full elliptic regularity assumption. Moreover, Xu and Zikatanov [200] considered methods of subspace corrections in an abstract setting and showed that the contraction 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 rate 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 nor realistic predictions of the actual contraction factor in practice [185]. This statement seems confusing as the XZ identity gives an exact equality for the contraction 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 easy to obtain a quantitative convergence estimate. Algebraic convergence estimates can be applied to obtain reasonable quantitative convergence speed for multigrid methods; see [138, 145] 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* contraction factor of the multigrid algorithms (in the sense this contraction factor can be obtained by the worst case mode); see [53, 166].

6.3 Nested iterations

sec:cycles

As in Algorithm 6.1, the solve phase approximates corresponding problems by calling a two-grid algorithm recursively. There are different approaches for the solve phase; for example, the V-cycle method in §6.1. In this section, we discuss other popular methods for the solve phase.

6.3.1 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 $\mathcal{B}$ restricted 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 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:

$$u_c^{(0)} = 0, \quad u_c^{(k)} = u_c^{(k-1)} + \mathcal{B}_c\big(f_c - \mathcal{A}_c u_c^{(k-1)}\big), \quad k = 1, \ldots, \nu. \tag{6.6}$$

eqn:mgstep

The parameter ν is often called the *cycle index*.

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

$$\mathcal{B}_{c,\nu} f_c := \big(\mathcal{I} - \mathcal{E}_c^{\nu}\big)\big(\mathcal{I} - \mathcal{E}_c\big)^{-1}\mathcal{B}_c f_c = \big(\mathcal{I} - \mathcal{E}_c^{\nu}\big)\mathcal{A}_c^{-1} f_c. \tag{6.7}$$

eqn:nu-cycle

Motivated by (6.7), we introduce a polynomial $q_\nu(t) := (1-t)^\nu \in \mathcal{P}_\nu$ and let

$$\mathcal{B}_{c,\nu} := \Big(\mathcal{I} - q_\nu\big(\mathcal{B}_c\mathcal{A}_c\big)\Big)\mathcal{A}_c^{-1}.$$

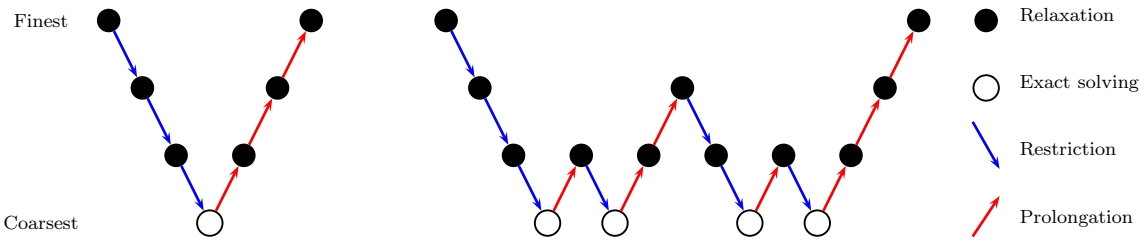


fig:VW-cycle

Figure 6.4: Multigrid V-cycle (left) and W-cycle (right).

alg:MG-cycle

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 $k = 1, 2, \ldots, m_1$, compute

$$v \leftarrow v + \mathcal{S}_l\big(r_l - \mathcal{A}_l v\big);$$

(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., $\mathcal{A}_{l-1}e_{l-1} = \mathcal{Q}_{l,l-1}\big(r_l - \mathcal{A}_l v\big)$ using the iteration: Set $e_{l-1} = 0$ initially. For $k = 1, \ldots, \nu$, compute

$$e_{l-1} \leftarrow e_{l-1} + \mathcal{B}_{l-1}\Big(\mathcal{Q}_{l,l-1}(r_l - \mathcal{A}_l v) - \mathcal{A}_{l-1}e_{l-1}\Big); \tag{6.8}$$ eqn:W-cycle

Update the solution with

$$v \leftarrow v + \mathcal{I}_{l-1,l}e_{l-1};$$

(3) **Post-smoothing**: For $k = 1, 2, \ldots, m_2$, 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.

From the previous discussion, we notice 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. 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 6.4. By calling the coarse correction twice as in (6.8), we can obtain $\mathcal{B}_{c,2}$, i.e. the W-cycle; see HW 6.4. 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 6.5 for an alternative scheme.

In V-cycle and W-cycle, the iterators on different coarse levels (except the coarsest level) are fixed to be 1 and 2, respectively. 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 [6] and references therein for details.

Remark 6.3 (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 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 [156]. □

ex:AMLI **Example 6.4** (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. □

[1] Here "algebraic" stands for the fact that certain inner polynomial iterations are used in the multilevel cycle.

6.3.2 Complexity of multigrid iterations

Now we turn our attention to the work estimate (i.e. the number of floating-point calculations) of nested iterations. For simplicity, we only consider the AMLI-cycle in Example 6.4 with $\mu_2 \equiv 1$. Denote the computational work needed by $\mathcal{B}_l$ is $\mathcal{W}_l$. 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

$$\begin{aligned}
\mathcal{W}_{(k+1)l_0} &= \mu_1 O\big(N_{(k+1)l_0}\big) + O\big(N_{kl_0+1} + \cdots + N_{kl_0+l_0}\big) + \mu_1 \mathcal{W}_{kl_0} \\
&= \mu_1 O\big(N_{(k+1)l_0}\big) + \mu_1 \mathcal{W}_{kl_0} \\
&= \mu_1 O\big(N_{(k+1)l_0}\big) + \mu_1^2 O(N_{kl_0}) + \mu_1^2 \mathcal{W}_{(k-1)l_0} \\
&= \cdots\cdots \\
&= O\Big(\sum_{j=2}^{k+1} \mu_1^{k+2-j} N_{jl_0}\Big) + \mu_1^k \mathcal{W}_{l_0} \\
&= O\Big(\sum_{j=1}^{k+1} \mu_1^{k+2-j} N_{jl_0}\Big) \\
&= O\big(N_{(k+1)l_0}\big) \sum_{j=1}^{k+1} \big(\mu_1 \gamma^{d\,l_0}\big)^j.
\end{aligned}$$

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.

rem:VV **Remark 6.5** (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. □

6.3.3 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 computational 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 using 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 better 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 6.5. 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.

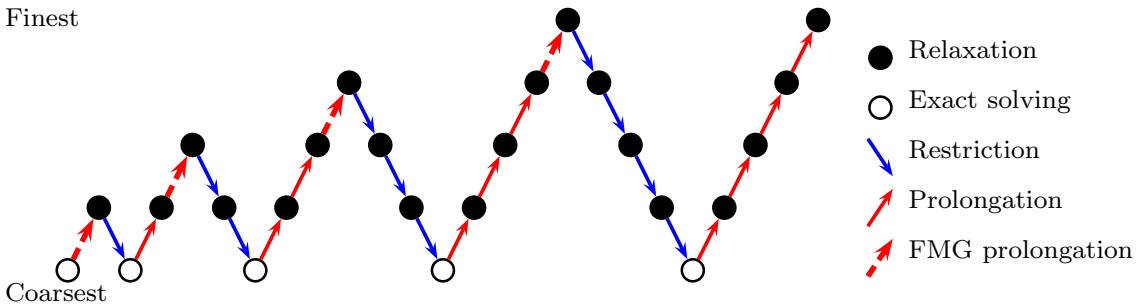


fig:F-cycle

Figure 6.5: 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
```

thm:FMG **Theorem 6.6** (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 $e_0 = 0$. On the l-th level $(0 < l \leqslant L)$, we have

$$\begin{aligned} \|\!|e_l|\!\| &\leqslant \delta^\nu \|\!|u_l - \tilde{u}_{l-1}|\!\| \leqslant \delta^\nu \Big(\|\!|u_l - u|\!\| + \|\!|u_{l-1} - u|\!\| + \|\!|u_{l-1} - \tilde{u}_{l-1}|\!\| \Big) \\ &\leqslant \delta^\nu \Big(C h_l |u|_2 + \|\!|e_{l-1}|\!\| \Big), \end{aligned}$$

where C is independent of meshsize h_l. 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 ν is large enough, then $\delta^\nu < \gamma$ and

$$\|\!|e_l|\!\| \leqslant \frac{C\delta^\nu h_l}{1-\gamma^{-1}\delta^\nu}|u|_2 \lesssim h_l|u|_2.$$

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.

6.4 Two-grid estimates for multigrid analysis

sec:quanconv

In this section, we introduce a simple tool for estimating convergence speed of the multigrid methods using the two-grid contraction 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.

6.4.1 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 [12, 109, 185]. 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 [152] 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 $\mathcal{B}_{\mathrm{eTG}}$:

$$\begin{aligned} \lambda_{\max}(\mathcal{B}_{\mathrm{TG}}\mathcal{A}) &\leqslant \lambda_{\max}(\mathcal{B}_{\mathrm{eTG}}\mathcal{A})\ \max\Big\{\lambda_{\max}(\mathcal{B}_c\mathcal{A}_c), 1\Big\}, \\ \lambda_{\min}(\mathcal{B}_{\mathrm{TG}}\mathcal{A}) &\geqslant \lambda_{\min}(\mathcal{B}_{\mathrm{eTG}}\mathcal{A})\ \min\Big\{\lambda_{\min}(\mathcal{B}_c\mathcal{A}_c), 1\Big\}. \end{aligned}$$

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 on $l = 2, 3, \ldots, L$, that

$$\rho_l^{\text{W-cycle}} \leqslant \frac{\sigma}{1-\sigma}.$$

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.

6.4.2 Limitations of two-grid theory for GMG ⋆

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 [143] 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 [146]. 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 [205] and [210].

6.4.3 LFA ladder

For a specific problem, it is recommended to perform quantitative analysis (more specifically, the LFA method) to determine the maximum possible convergence rate of multigrid algorithms. A general procedure for developing multigrid programs, known as the "LFA ladder", has been adopted in practice:

1. Choose an appropriate discretization scheme for the problem;
2. Find an effective smoother with a satisfactory contraction factor μ using the LFA method;
3. Select transfer operators and determine the two-grid LFA contraction factor σ;
4. Check whether the two-grid LFA contraction factor σ is close to μ;
5. Check whether the contraction factor of the multigrid program approximates σ;
6. Apply the full multigrid method;
7. Verify whether the expected discretization accuracy is achieved.

This procedure helps practitioners make development decisions and improve efficiency. We recommend readers see [185, 193] for more details on these techniques.

6.5 Implementation of multigrid methods

sec:implement

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 while it might not be an efficient way.

6.5.1 A sparse matrix data structure

First, we discuss how to represent a sparse matrix in practice. Evidently, we do not wish to store the zeros in a sparse matrix. There exist various ways to store a sparse matrix with optimal storage complexity. More importantly, the choice of storage format usually depends on the hardware architecture. A widely used general purpose data structure is the so-called

Compressed Sparse Row (CSR) format [172]. The CSR storage format of a sparse matrix A consists of three arrays, defined as follows:

1. A double array of non-zero entries corresponding to the column indices, *val*, of size nnz;
2. An integer array of row pointers, IA, of size $N + 1$;
3. An integer array of column indices, JA, 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 6.7 (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:

- *val* is of the same size as JA and

$$val = \left\| \; 1.0 \; \middle| \; 1.5 \; \middle| \; 1.2 \; \right\| \; 1.0 \; \middle| \; 7.0 \; \middle| \; 6.0 \; \middle| \; 1.0 \; \right\| \; 3.0 \; \middle| \; 6.0 \; \left\| \; 2.0 \; \middle| \; 5.0 \; \middle| \; 1.0 \; \right\|$$

- IA is of size 5 and

$$IA = \left\| \; 1 \; \middle| \; 4 \; \middle| \; 8 \; \middle| \; 10 \; \middle| \; 13 \; \right\|$$

- JA is of size $IA(5) - IA(1) = 12$

$$JA = \left\| \; 1 \; \middle| \; 2 \; \middle| \; 5 \; \right\| \; 2 \; \middle| \; 4 \; \middle| \; 3 \; \middle| \; 5 \; \right\| \; 1 \; \middle| \; 3 \; \left\| \; 3 \; \middle| \; 5 \; \middle| \; 1 \; \right\|$$

Note that the indices in JA need not be sorted in ascending order as seen in this example. □

Remark 6.8 (Some comments on CSR)**.** We provide a few comments on the CSR format:

1. There exist various data structures for storing sparse matrices, suited to specific problems, algorithms and hardware. Nevertheless, the CSR format remains widely used and accepted.

2. The format does not assume any ordering of entries within each row.

3. The format allows duplicating nonzero values. □

With a sparse matrix stored in the CSR format, the matrix-vector multiplication $v = Au$ can be performed in the following way:

Listing 6.2: Sparse matrix-vector multiplication

```
for i = 1, ..., N
    t ← 0;
    for k = IA(i), ..., IA(i + 1) − 1
        j ← JA(k);
        t ← t + val(k) ∗ u(j);
    end
    v(i) ← t;
end
```

Now we give a pseudo code of the GS method (1.31) 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

```
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 code is very similar to the previous matrix-vector multiplication and can be implemented very easily.

6.5.2 Assembling finite element matrix

Geometric multigrid methods are often implemented without assembling the global stiffness matrix (matrix-free). However, we implement GMG using a matrix-based approach. So we first discuss how to assemble the finite element matrix. Consider the mesh in Figure 6.6 which has both triangles and quadrilaterals.

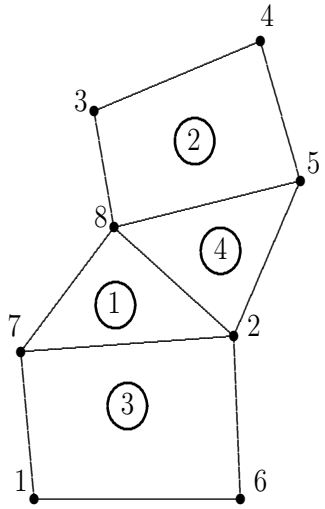


fig:FEmesh Figure 6.6: A mesh with 4 elements and 8 nodes

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, namely, IA and JA;
2. Loop through elements and compute the actual entries in A; This step (the actual assembly of the entries) is usually easier but has to be done case by case. We leave this step to the readers.

Here we only explain the first step in the following abstract algorithm:

alg:silly **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 6.6 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:

$$IE = \| 1 \,|\, 4 \,|\, 8 \,|\, 12 \,|\, 15 \,\|$$
$$JE = \| 7 \,|\, 8 \,|\, 2 \,\| 8 \,|\, 3 \,|\, 4 \,|\, 5 \,\| 1 \,|\, 6 \,|\, 7 \,|\, 2 \,\| 2 \,|\, 5 \,|\, 8 \,\|$$

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.

rem:transpose **Remark 6.9** (How to find transpose of a CSR matrix)**.** The nontrivial task is to perform transposition without using additional memory. An algorithm by F. Gustavson from the 1970s can achieve this, found in [160]. During transposition by Gustavson's algorithm, column indices within each row emerge in increasing order naturally. □

[2]Make sure that you do not add anything twice. This can be done using an additional indicator array.

6.5.3 Matrix form of transfer operators

As seen in (6.1), we can obtain coarse-level stiffness matrices using the algebraic Galerkin relation:

$$A_l = P_l^T A\, P_l, \quad 0 \leqslant l < L.$$

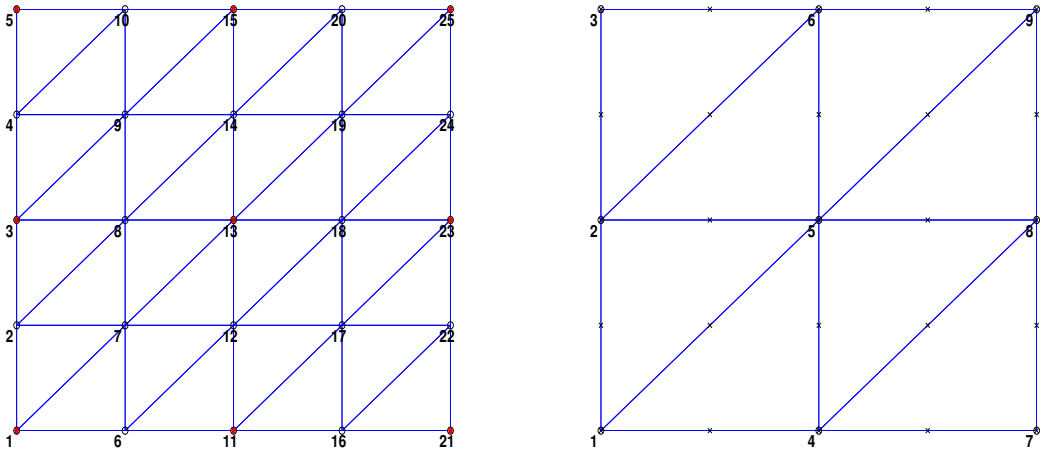
Since we can calculate the matrices A_l's level by level, we only need the prolongation matrix between two consecutive levels, $P_{l-1,l}$ for $0 < l \leqslant L$. In the operator form, it is trivial to define the prolongation, which is just the natural embedding operator. But for implementation, we 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 6.10 (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 the coarse mesh $\mathcal{M}_1$; see Table 6.1.

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



grid12

Figure 6.7: Fine and coarse meshes.

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

tab:A1

Table 6.1: The nonzero entries of the stiffness matrix A_1 on the fine grid.

numbering given in Figure 6.7, 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 a similar way.

Remark 6.11 (Improving efficiency of V-cycle implementation)**.** While Algorithm 6.1 is defined as a recursive call of the two-grid method, this often results in a costlier implementation. Several techniques can improve the efficiency (reducing wall time) of each V-cycle:

- Matrix-free implementation to avoid SpMV;
- Replace the recursive procedure by a loop over each level;
- Use red-black ordering for smoothers to obtain better parallelism;
- Combine smoothing sweeps to reduce cache-missing rate.

6.6 Homework problems

prob:GMGconv **HW 6.1.** Show the geometric multigrid V-cycle (Algorithm 6.1) is uniformly convergent in $\mathbb{R}^d$.

b:interp-proj **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 $\mathit{\Pi}_l : V \mapsto V_l$.

prob:decompH1 **HW 6.3.** Let $\Omega = (0,1)$ and $v \in V_h$ be a $\mathcal{P}_1$ Lagrange finite element function. Show that $|v|_1^2 = \sum_{l=l}^{L} |v_l|_1^2$.

prob:W-cycle **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 (6.8).

prob:FMG **HW 6.5.** Show the work estimate of the full multigrid method is $O(N)$.

Chapter 7

Algebraic Multigrid Methods

ch:amg

Consider the linear system of equations arising from the second-order elliptic equation with diffusion coefficient on structured or unstructured grids:

$$Au = f, \quad \text{where } A \in \mathbb{R}^{N\times N} \text{ is SPD and } u, f \in \mathbb{R}^N.$$

Problems with anisotropic coefficients on regular grids, or problems with isotropic coefficients but on anisotropic grids, can cause difficulties for geometric multigrid methods. While geometric multigrid (GMG) relies on the availability of robust smoothers, algebraic multigrid (AMG) takes a different approach by focusing on constructing a suitable coarse space [56, 57, 170]. AMG is an approach to generalize GMG and enhance its robustness. There are several situations where AMG can be applied but GMG cannot, for instance, problems on complex domains or irregular triangulations, problems with discontinuous coefficients, and purely algebraic problems.

7.1 From GMG to AMG

sec:AMG

How to make multigrid methods more robust in practice has been an important question since the early stages of the method's development. AMG is one of the many approaches to improving robustness. In this section, we first demonstrate some motivations for algebraic multigrid methods.

7.1.1 General procedure of multigrid methods

From our previous discussions, we observe that a typical multigrid (MG) algorithm contains two phases: the "*setup*" phase and the "*solve*" phase. The setup phase initializes a hierarchical structure, including coarse spaces, prolongations, restrictions, and coarse solvers for multilevel iterations. Note 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 be applied for equations on general domains equipped with unstructured grids. Algebraic multigrid (AMG) methods can be viewed as a generalization of GMG methods. See [203] and references therein for details.

We now explain how to perform the multigrid setup phase in a relatively general setting. Once complete, an appropriate multilevel nested iteration scheme should be chosen for the solve phase; see §6.3. It is immediately evident that we only need to discuss how to set up hierarchical information between two consecutive grids/levels for multigrid methods. We can summarize a general multigrid setup procedure in 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 $A_c = P^T AP$ and $f_c = P^T f$.

For GMG methods, Steps 2–4 are determined by the information of nested grids and the users only need to find an effective smoother S. For example, in §1.4, we have presented a 1D GMG method in a purely algebraic fashion. We have observed that:

(1) In GMG coarsening, the topologies of the graph representing the stiffness matrices on different levels are explicitly given from the geometric refinement procedure.

(2) For GMG, Prolongation and restriction usually depend only on the topological structure of the grids without using the grid coordinates.

(3) For GMG, smoothness of error is defined geometrically. In more general settings, a geometrically smooth error can be 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 algorithms usually employ simple relaxation processes (typically

point-wise relaxations) and then attempt to construct suitable operator-dependent interpolations using the algebraic information of A to treat the error components that cannot be reduced by the relaxation.

7.1.2 Sparse matrices and graphs ⋆

A sparse matrix can be represented as a graph. As the sparse matrices that we consider are mainly symmetric, we begin with undirected graphs. 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*. We always consider set of vertices being a subset 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 a graph G is a set of vertices in G that is nonadjacent (i.e. no two vertices are adjacent). A *maximal independent set* (mIS) or *maximal stable set* is an independent set such that adding any other vertex would introduce an adjacent pair. A graph can have many mIS's of varying sizes; among them, the largest mIS, or potentially several equally large mIS's, of a graph is called a *maximum independent set* or MIS. The MIS problem can be described as follows:

Given a graph $\mathsf{G} = (\mathsf{V},\, \mathsf{E})$, find an independent set in G of maximum cardinality. In the weighted case, each node $v \in \mathsf{V}$ has an associated non-negative weight $w(v)$ and the goal is to find a maximum weight independent set.

This problem is NP-hard and it is natural to ask for approximation algorithms.

Remark 7.1 (Minimal vertex cover)**.** A related notion is the so-called minimal vertex cover (mVC), which is a vertex cover of a graph that is not a proper subset of any other vertex cover. A *vertex cover* of $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ is a set $\mathsf{S} \subseteq \mathsf{V}$ such that all edges $(i, j) \in \mathsf{E}$ has at least one end point in S. A minimal vertex cover corresponds to the complement of maximal independent vertex set, so the numbers of minimal vertex covers and maximal independent vertex sets in a graph are identical.

A *path* from a vertex i to another vertex j is a sequence of edges

$$\big\{(i,\, k_1),\, (k_1, k_2),\, \ldots,\, (k_{\ell-2}, k_{\ell-1}),\, (k_{\ell-1}, j)\big\} \subseteq \mathsf{E}$$

and the number of edges ℓ is called the length of this path. A vertex j is *connected* to a vertex i if there is a path from j to i. The distance between j and i 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 to each other and is set to ∞ if they are not connected. An undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ is called *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

$$\mathsf{E} := \big\{(i,\, j) \,:\, a_{i,j} \neq 0\big\}.$$

As a general rule, sparse matrices do not provide any geometric information for the underlying graph except 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 $\mathsf{G}(A)$.

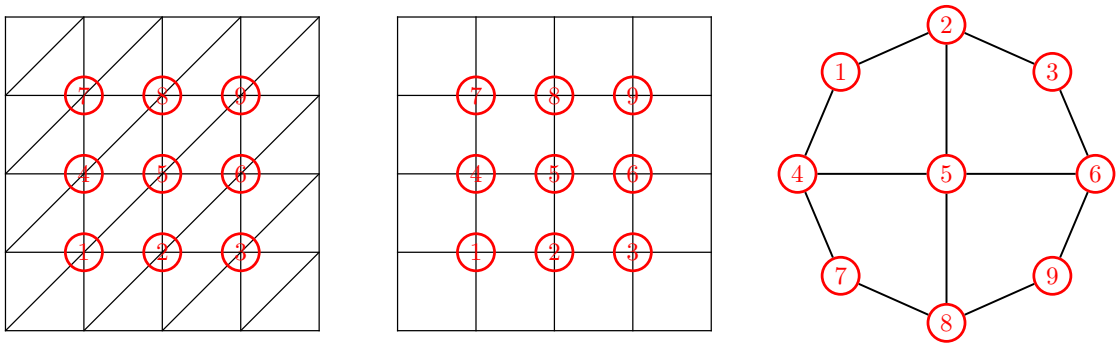


Figure 7.1: Finite element grid (left), finite difference grid (middle), and graph of their corresponding stiffness matrices (right).

fig:graph

7.1.3 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

$$a_{i,i} > 0, \quad a_{i,j} \leqslant 0 \ \ (i \neq j), \quad \text{and} \quad a_{j,j} \geqslant \sum_{i\neq j} |a_{i,j}|, \quad a_{j,j} > \sum_{i\neq j} |a_{i,j}| \ \text{ for at least one } j.$$

Apparently, the stiffness matrix A in equation (1.26) for the model problem is an *M-matrix*. The classical convergence theory for AMG was developed for the class of symmetric M-matrices (see [52, 170]). However, stiffness matrices from finite element discretizations are generally not M-matrices, even for the Poisson's equation. In fact, whether a stiffness matrix is an M-matrix depends on the specific mesh $\mathcal{M}$ used. In practice, many AMG algorithms use simple filtering schemes to construct an approximate M-matrix A_M from the stiffness matrix A. Xu

and Zikatanov [202] introduced the concept of "M-matrix relatives" to analyze such constructed M-matrices.

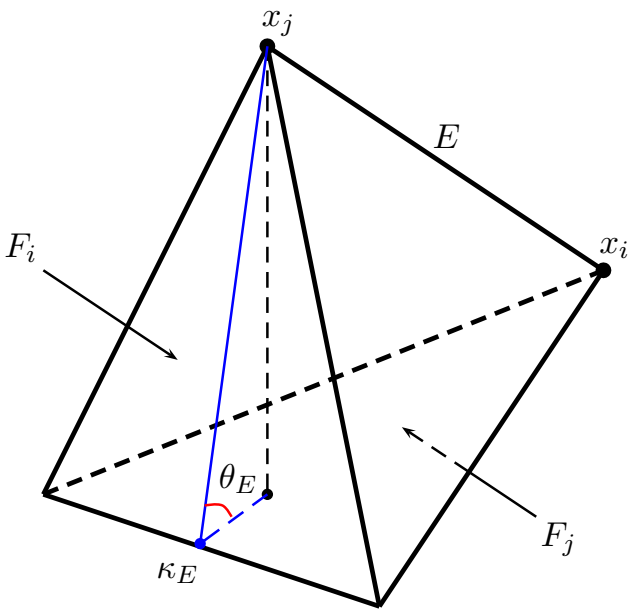


fig:simplex Figure 7.2: Definition of θ_E and κ_E in a simplex τ in 3D.

First, we introduce a few notations in Figure 7.2. In any given 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

$$\omega_E(\tau) := \frac{1}{d(d-1)} \big|\kappa_E(\tau)\big| \cot \theta_E(\tau). \tag{7.1}$$

eqn:OET

With these definitions, we then have the following result; see [199] for details.

prop:M-matrix **Proposition 7.2** (Condition for M-matrix)**.** The stiffness matrix for the Poisson's equation is an M-matrix if and only if

$$\sum_{\tau \supset E} \omega_E(\tau) \geqslant 0, \quad \forall E.$$

Remark 7.3 (Delaunay triangulation and M-matrix)**.** In $\mathbb{R}^2$, the above proposition simply means the sum of the angles opposite to any edge is less than or equal to π, implying the underlying triangulation must be Delaunay. Hence, the stiffness matrix for the Poisson's equation will be an M-matrix if the triangulation is Delaunay. This condition is nearly sharp[1]. □

The *Delaunay triangulation* is a geometric structure widely used by engineers for mesh generation. In two dimensions, it offers a notable advantage: among all possible triangulations

[1] The opposite direction is true with a few possible exceptions near the boundary.

of a given set of points, the Delaunay triangulation maximizes the minimum angle, thereby reducing the occurrence of narrow and sharp angles. This characteristic not only enhances the mesh's structural quality but also improves interpolation accuracy. Extensive research has been dedicated to Delaunay triangulations, resulting in highly efficient algorithms for constructing and updating these meshes; see for example [97].

In the context of a finite point set G, a triangle is Delaunay if its vertices are in G and its open circumdisk is empty—namely, it contains no point of G. Note that any number of points in G can lie on a Delaunay triangle's circumcircle without affecting its Delaunay property. Similarly, an edge is Delaunay if its vertices are in G and it has at least one empty open circumdisk. A Delaunay triangulation of G is a triangulation in which every triangle is Delaunay.

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 [165] for details. Let $\phi_{\mathcal{M}_h} \in V_h$ be 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

$$\left|\phi_{\mathcal{M}_h^D}\right|_1^2 \leqslant \left|\phi_{\mathcal{M}_h}\right|_1^2, \quad \forall\, v \in \mathbb{R}^N.$$

The above inequality means the Delaunay triangulation results in lower roughness of finite element functions among all possible triangulations on a fixed set of vertices.

7.1.4 Tarjan's algorithm ⋆

We have observed that ordering is very important for the performance of local relaxation methods like the Gauss–Seidel method. For example, in Remark 3.30, we have shown that the ordering is important using the local Fourier analysis. By far, we have not assigned any kind of ordering for the unknowns in the solution vector. In algebraic multigrid methods, we can 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 its "big entries" all in the lower triangle and this technique can enhance the performance of the Gauss–Seidel smoother.

Example 7.4 (Cross-wind ordering)**.** In Figure 7.3, the bold edges represent the cross-wind connections. If we number the blocks sequentially from left to right, with the degrees of freedom

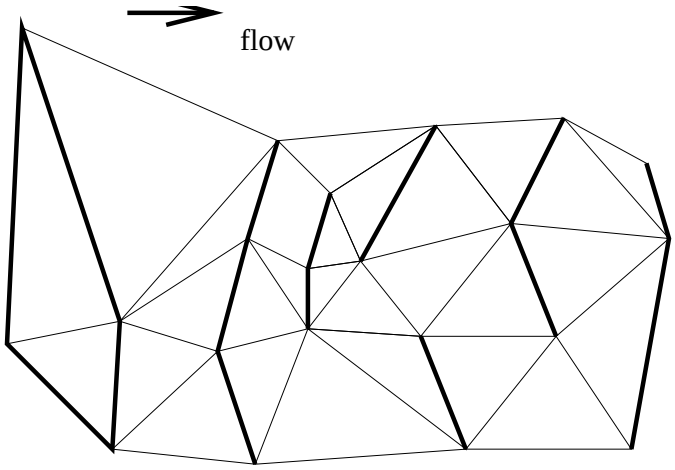


fig:block

Figure 7.3: A sample mesh along with a specified flow direction

within each block ordered arbitrarily, the stiffness matrix will have a lower triangular structure as shown in equation (7.2). Based on this idea, an algorithm known as the Cross-Wind-Block method has been proposed by Wang and Xu [190] for convection-dominated problems. □

The question now is of course how to find such an ordering. In this section, we present the Tarjan's algorithm [184] in the graph theory to find the "best" ordering for the Gauss–Seidel method. Tarjan's algorithm is used to find strongly connected components in a graph. A *strongly connected component* is a maximal subset of vertices such that there is a path between any two vertices in the subset. Tarjan's algorithm uses depth-first search on the digraph:

1. Drop some of the entries in the matrix A, which are non-essential. This will transform the graph corresponding to A to a digraph; see Remark 7.5.

2. Find the strongly connected components in this directed graph. Each one of these components correspond to a diagonal block in the stiffness matrix after permutation.

rem:digraph **Remark 7.5** (Preprocessing to a get directed graph)**.** We comment that sometimes the graph corresponding to A is undirected (for example, the finite element stiffness matrix of the Poisson's equation). However, if we are consider a non symmetric problem, situation could be very different. In any event, 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} if $|a_{ki}/a_{ik}| < \epsilon$. Then we can apply the Tarjan's algorithm for finding the strongly connected components in the digraph. □

After permutation according to the strongly connected component ordering obtained by the

Tarjan's algorithm, the matrix A will have the following structure:

$$A = \begin{bmatrix} \boxed{A_{11}} & \approx \epsilon & \approx \epsilon & \cdots & \cdots \\ \boxed{A_{21}} & \boxed{A_{22}} & \approx \epsilon & \cdots & \cdots \\ \boxed{A_{31}} & \boxed{A_{32}} & \boxed{A_{33}} & \cdots & \cdots \\ \cdots & \cdots & \cdots & \ddots & \cdots \\ \boxed{A_{K1}} & \boxed{A_{K2}} & \boxed{A_{K3}} & \cdots & \boxed{A_{KK}} \end{bmatrix}. \tag{7.2}$$ eqn:lower

Example 7.6 (Strongly connected component)**.** Imagine now that a graph represents a town, the edges are streets and the vertices are houses. You are walking along the streets and they are one way streets (directed). You may go and arrive at a house for the first time; other than that, there are two possible situations which may occur:

1. Either you go and 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.

It is obvious that, if we return at a place we have been before (encountering a cycle), we found a cycle that corresponds to a 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. □

A simplified algorithm is as follows:

alg:tarjan **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 7.7 (Finding strongly connected components of a graph)**.** The above algorithm is visualized in Figures 7.4, in which we consider a 2D flow problem. First we assume that we start

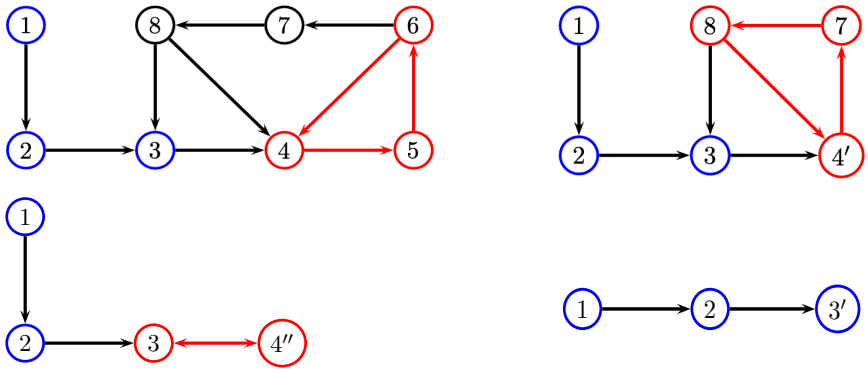


fig:gt2-5

Figure 7.4: Finding strongly connected components of a directed graph

from 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 continue with the new graph. 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. □

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 cost 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 corresponding 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 [107]. A good explanation and a lots of examples related to the Tarjan's algorithm can be found in [100].

7.2 Motivations of algebraic multigrid methods

onvergenceAMG

In §6.2, 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. [56, 57, 52] on the convergence analysis applicable to AMG methods, there have been a lot of discussions on the AMG convergence theory; see [170, 61, 178, 90, 91, 189] for example. The readers are referred to the recent survey paper by Xu and Zikatanov [202].

7.2.1 Algebraic convergence theory

In the purely algebraic setting, because the local 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 [129, 155, 202], 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 [131, 153, 133, 132]; for analysis based on aggregation-type AMG algorithms, we refer to [186, 188, 62, 143, 147, 60].

We have shown the exact contraction factor estimate of two-level methods in Theorem 3.38. Now we derive a convergence theory from an algebraic viewpoint. In particular, we wish to give conditions on the grid-transfer matrices, such that two-level AMG methods converge. We mainly follow the argument in a recent survey by MacLachlan and Olson [129]. Throughout this chapter, without loss of generality, we assume that:

Assumption 7.8 (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

$$\Pi_c = PA_c^{-1}P^TA = P(P^TAP)^{-1}P^TA \tag{7.3}$$ eqn:Pi_cForm

is a projection onto range(P). The error reduction matrix for the two-gird method in Algorithm 3.3 can then be written as

$$E_{\mathrm{TG}} := (I - SA)(I - \Pi_c). \tag{7.4}$$ eq:EtgMatrix

From Theorem 3.38, 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

$$\|E_{\mathrm{TG}}\|_A^2 = \sup_{e \neq 0} \frac{\|(I - SA)(I - \Pi_c)e\|_A^2}{\|e\|_A^2} = 1 - \delta^*, \tag{7.5}$$ eq:ErrorEtgMa

where δ^* yields a sharp and parameter-independent two-grid contraction factor. Of course, it is essential to pose conditions only on the prolongation P to ensure convergence, as the other components in (7.4) have already been given.

AMG:TG **Theorem 7.9** (Convergence rate of two-level algorithm)**.** If there exists $\delta > 0$ such that

$$\|(I - SA)e\|_A^2 \leqslant \|e\|_A^2 - \delta\|(I - \Pi_c)e\|_A^2, \quad \forall\, e \in V, \tag{7.6}$$ eq:TGassump1

then the V(0, 1) two-grid method satisfies that

$$\|E_{\mathrm{TG}}\|_A^2 = 1 - \hat{\delta}, \quad \text{with } \hat{\delta} := \inf_{(I-\Pi_c)e\neq 0} \frac{\|e\|_A^2 - \|(I - SA)e\|_A^2}{\|(I - \Pi_c)e\|_A^2} \geqslant \delta.$$

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 $(I - SA)(I - \Pi_c)e = 0$, 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

$$\frac{\|(I - SA)(I - \Pi_c)^2\hat{e}\|_A^2}{\|(I - \Pi_c)^2\hat{e}\|_A^2 + \|\Pi_c(I - \Pi_c)\hat{e}\|_A^2} = \frac{\|(I - SA)(I - \Pi_c)\hat{e}\|_A^2}{\|(I - \Pi_c)\hat{e}\|_A^2} \geqslant \frac{\|(I - SA)(I - \Pi_c)\hat{e}\|_A^2}{\|\hat{e}\|_A^2}.$$

From the last inequality, the contraction factor achieves the supremum when $\Pi_c\hat{e} = 0$. That is to say, from the definition (7.5),

$$\|E_{\mathrm{TG}}\|_A^2 = \sup_{(I-\Pi_c)e\neq 0} \frac{\|(I - SA)e\|_A^2}{\|(I - \Pi_c)e\|_A^2}.$$

Hence the result. □

Note that, if we further assume that the parameter $\hat{\delta}$ in Theorem 7.9 is bounded uniformly on all levels, we can also obtain a uniform bound for the V-cycle contraction factor by recursion [138]. This bound also gives reasonable estimates numerically [145].

Basically, the condition (7.6) 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, S is allowed to be ineffective for the error components that can be reduced by CGC efficiently. The components for which S is ineffective are considered *smooth* and they have to be in the range of the interpolation, range(P), roughly.

So it is reasonable to assume (7.6) in order to get an efficient TG algorithm. However, it is difficult to obtain such a $\hat{\delta}$ 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

$$\begin{aligned}\|E_{\mathrm{TG}}e\|_A^2 &\leqslant \big\|(I-\Pi_c)e\big\|_A^2 - \hat{\alpha}_g g\big((I-\Pi_c)e\big)\\ &\leqslant \big\|(I-\Pi_c)e\big\|_A^2 - \hat{\alpha}_g\hat{\beta}_g^{-1}\big\|(I-\Pi_c)e\big\|_A^2\\ &= \Big(1-\hat{\alpha}_g\hat{\beta}_g^{-1}\Big)\big\|(I-\Pi_c)e\big\|_A^2 \qquad (7.7)\\ &\leqslant \Big(1-\hat{\alpha}_g\hat{\beta}_g^{-1}\Big)\|e\|_A^2, \qquad (7.8)\end{aligned}$$

eq:EandCGC1

eq:EandCGC2

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:

$$\big\|(I-SA)e\big\|_A^2 \leqslant \|e\|_A^2 - \bar{\alpha}_g\, g(e), \quad \forall\, e\in V, \qquad (7.9)$$

A1_smooth

and

$$\exists\, \bar{\beta}_{g,\mathrm{s}},\ \text{such that } \big\|(I-\Pi_c)e\big\|_A^2 \leqslant \bar{\beta}_{g,\mathrm{s}}\, g(e), \quad \forall\, e\in V. \qquad (7.10)$$

A2_approx

The condition (7.9) is a smoothing property and the condition (7.10) is an approximation property. The condition (7.10) is oftentimes called the *strong approximation assumption*. In view of (7.7), we can further weaken this condition and assume the *weak approximation assumption*:

$$\exists\, \bar{\beta}_{g,\mathrm{w}},\ \text{such that } \big\|(I-\Pi_c)e\big\|_A^2 \leqslant \bar{\beta}_{g,\mathrm{w}}\, g\big((I-\Pi_c)e\big), \quad \forall\, e\in V. \qquad (7.11)$$

A2_wapprox

From the above analysis, we can easily deduce the following theorem.

AMG:TG2

Theorem 7.10 (Convergence estimate of two-level AMG)**.** If (7.9) and (7.10) (or its weaker version (7.11)) hold, then V(0, 1) two-grid method satisfies

$$\|E_{\mathrm{TG}}\|_A^2 \leqslant 1-\bar{\alpha}_g\,\bar{\beta}_g^{-1}.$$

Remark 7.11 (Strong and weak approximation properties)**.** The strong approximation assumption (7.10) can be used to show convergence of V-cycle AMG methods via a recursion [138, 170]. But the weak approximation assumption (7.11) is not sufficient for V-cycle to converge [52]. □

It is important to recognize that, even if we can provide simple conditions on coarsening to ensure the approximation assumptions necessary for a convergent two-level or multilevel AMG method, it remains unclear how to develop an algorithm that satisfies these assumptions using purely algebraic information. In fact, doing so in a strict sense is challenging. The coarsening process involves identifying coarse variables and constructing prolongation matrices, and these two steps are typically interdependent.

In the remainder of this chapter, we discuss more practical approaches for coarsening.

7.2.2 Interpolation operators

Now the question is how to choose such a function $g(e)$? Furthermore, how to apply Theorem 7.10 to enforce convergence conditions on the prolongation (or interpolation) matrix P to guarantee good AMG performance?

Apparently, if $g(e) := \|(I-\Pi_c)e\|_A^2$, we have $\hat{\alpha}_g = \hat{\delta}$ and $\hat{\beta}_g \equiv 1$. Another possible choice has been suggested by Ruge and Stüben [170]:

$$g(e) := \|e\|_{AD^{-1}A}^2.$$

In this case, by definition, the strong approximation assumption (7.10) can be written as

$$\inf_{e_c \in V_c} \|e - Pe_c\|_A^2 \leqslant \bar{\beta}_{\mathrm{s}} \|e\|_{AD^{-1}A}^2, \quad \forall\, e \in V. \tag{7.12}$$ A2_approx2

On the other hand, noticing (7.3) and let $e_c \in V_c$, we have

$$\begin{aligned}\|(I-\Pi_c)e\|_A^2 &= \Big((I-\Pi_c)e, (I-\Pi_c)e\Big)_A = \Big((I-\Pi_c)e, (I-\Pi_c)e - Pe_c\Big)_A \\ &\leqslant \|(I-\Pi_c)e\|_{AD^{-1}A} \|(I-\Pi_c)e - Pe_c\|_D\end{aligned}$$

If we assume, instead of (7.12), that

$$\inf_{e_c \in V_c} \|e - Pe_c\|_D^2 \leqslant \bar{\beta}_{\mathrm{w}} \|e\|_A^2, \quad \forall\, e \in V, \tag{7.13}$$ A2_wapprox2

then

$$\begin{aligned}\|(I-\Pi_c)e\|_A^2 &\leqslant \|(I-\Pi_c)e\|_{AD^{-1}A} \|(I-\Pi_c)e - Pe_c\|_D \\ &\leqslant \|(I-\Pi_c)e\|_{AD^{-1}A} \cdot \bar{\beta}_{\mathrm{w}}^{\frac{1}{2}} \|(I-\Pi_c)e\|_A,\end{aligned}$$

which yields the weak approximation property (7.11). In this way, we derived two alternative bounds, Eqs. (7.12) and (7.13), for the strong and weak approximation assumptions, respectively.

Using Eq. (7.13), we can obtain a convergence bound for the two-level method. As noted previously, the weak approximation property in Eq. (7.13) is typically insufficient to ensure a good interpolation P for a V-cycle. Additional conditions must be imposed 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 $Q = PT$, where $T \in \mathbb{R}^{N_c \times N}$ satisfies $TP = I_c$. If $T_{\mathrm{s}} := (P^T AP)^{-1} P^T A$, then it is easy to see that $Q_{\mathrm{s}} = PT_{\mathrm{s}} = \Pi_c$ is such an example. We can also give a simplified choice $T_{\mathrm{w}} := (P^T DP)^{-1} P^T D$. For any vector $0 \neq e \in V$, we can assume that

$$\inf_{e_c \in V_c} \frac{\|e - Pe_c\|_A^2}{\|e\|_{AD^{-1}A}^2} \leqslant \frac{\|e - Qe\|_A^2}{\|e\|_{AD^{-1}A}^2} \leqslant \bar{\beta}_{\mathrm{s}} \quad \text{or} \quad \inf_{e_c \in V_c} \frac{\|e - Pe_c\|_D^2}{\|e\|_A^2} \leqslant \frac{\|e - Qe\|_D^2}{\|e\|_A^2} \leqslant \bar{\beta}_{\mathrm{w}}, \tag{7.14}$$ cond:WeakAppr

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 well according to Theorem 7.10.

We notice that, in the above inequalities (the second one in particular), the measure

$$\mu_D(Q,e) := \frac{\|(I-Q)e\|_D^2}{\|e\|_A^2}, \quad \forall\, e \neq 0$$

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

$$\inf_{e_c \in V_c} \frac{\|e - Pe_c\|_X^2}{\|e\|_A^2} \leqslant \frac{\|e-Qe\|_X^2}{\|e\|_A^2} = \mu_X(Q,e) \leqslant \kappa. \tag{7.15}$$ eq:XZ_TGAMG_A

If we minimize

$$\sup_{e \neq 0} \mu_X(PT, e)$$

to find the "best possible" interpolation operator P, then it yields the so-called *ideal interpolation* [90, 209].

7.2.3 Algebraic smooth error

In Theorem 3.40, we have seen the following theoretical result: For any given smoother S, the best coarse space of dimension N_c is given by

$$V_c^{\text{opt}} := \text{span}\{\phi_k\}_{k=1}^{N_c}, \tag{7.16}$$ eqn:optVH

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 eigen-pairs 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 (3.27), 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 (7.9), it motivates the following definition of the algebraic smooth vector:

Definition 7.12 (Algebraic smoothness)**.** Let $\varepsilon \in (0,1)$ be a small parameter. If $e \in V$ satisfies

$$(\overline{S}Ae, e)_A \leqslant \varepsilon (e,e)_A, \quad \text{i.e. } (\overline{S}Ae, Ae) \leqslant \varepsilon \|e\|_A^2,$$

then e is algebraically ε-smooth (or the ε-algebraic low-frequency) with respect to A.

For the algebraic smooth error component e, by adding and subtraction and (2.13), we have

$$\Big((I-\overline{S}A)e,e\Big)_A \geqslant (1-\varepsilon)(e,e)_A \quad\Longrightarrow\quad \frac{\|(I-SA)e\|_A^2}{\|e\|_A^2} \geqslant 1-\varepsilon.$$

Evidently, the contraction factor for this error component e approaches 1 if ε is small. Essentially, this means that algebraically smooth error components are those that the smoother S or $\overline{S}$ cannot effectively damp. That is, an error that cannot be eliminated by the smoother constitutes a smooth error, as noted in Remark 1.32 for geometrically smooth errors.

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

$$\|e\|_A^2 \leqslant \varepsilon\|e\|_{\overline{S}^{-1}}^2, \tag{7.17}$$ eqn:algsmooth

which can be viewed as an alternative characterisation of algebraically smooth (low-frequency) vectors.

Inspired by the above definition of algebraic low-frequency components (7.17), we can define algebraic high-frequency components as follows:

Definition 7.13 (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 [202]:

Theorem 7.14 (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 $V = V_c + V_{\mathrm{hf}}$ is a stable decomposition, i.e., for any $v \in V$, there exist $v_c \in V_c$ and $v_{\mathrm{hf}} \in V_{\mathrm{hf}}$ such that $v = Pv_c + v_{\mathrm{hf}}$ and $\|v_{\mathrm{hf}}\|_A^2 \leqslant \beta\|v\|_A^2$. Then the resulting two-level AMG satisfies

$$\|E_{\mathrm{TG}}\|_A \leqslant 1-\zeta\beta^{-1}.$$

Proof. Since we have the following estimate

$$\inf_{w_c\in V_c}\|v-Pw_c\|_{\overline{S}^{-1}}^2 \leqslant \|v_{\mathrm{hf}}\|_{\overline{S}^{-1}}^2 \leqslant \frac{1}{\zeta}\|v_{\mathrm{hf}}\|_A^2 \leqslant \frac{\beta}{\zeta}\|v\|_A^2,$$

we can prove the theorem using two-level convergence theory. □

Remark 7.15 (Local adaptation of AMG)**.** In AMG methods, it is not important whether the smoother S smooths the error in a geometric sense. Rather, the key point is that the error after smoothing sweeps must be characterized algebraically to a certain degree, making it possible to construct coarse levels and define interpolations locally adapted to the given smoother's properties. □

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

$$(Au, v) = \sum_{\substack{(i,j)\in\mathsf{E}\\ i<j}} -a_{i,j}(u_i - u_j)(v_i - v_j). \tag{7.18}$$ eqn:Auv-Neuma

We can also easily derive the corresponding equality for the Dirichlet boundary condition or the mixed boundary condition:

$$(Au, v) = \sum_{\substack{(i,j)\in\mathsf{E}\\ i<j}} -a_{i,j}(u_i - u_j)(v_i - v_j), \qquad \text{if} \quad u_j = v_j = 0, \ \ \forall\, x_j \in \Gamma_D. \tag{7.19}$$ eqn:Auv-Diric

m:smootherror **Remark 7.16** (Smooth error and Classical AMG)**.** A simpler characterization of smooth error is used in methods like the Classical AMG. According to (7.18) and (7.17), the algebraically smooth error e satisfies that

$$\sum_{i<j} -a_{i,j}(e_i - e_j)^2 = (Ae, e) \leqslant \varepsilon \|e\|^2_{S^{-1}} \ll 1. \tag{7.20}$$ eqn:CAMG-smoo

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. □

7.2.4 Construction of coarse spaces

We now discuss a few guidelines on how to construct coarse spaces and prolongation matrices based on the AMG theory discussed above. In §6.1, we have proposed a general procedure of the multigrid setup phase, among which the coarsening algorithms (Step 2) 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.

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 3.39, 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 and (7.14) motivate that we can construct a coarse space V_c, 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., $\|v\|_A \leqslant \varepsilon^{1/2}\|v\|_{\overline{S}^{-1}}$ is small, then v can be approximated well in the coarse space V_c. This condition is sufficient for convergence of the two-grid method. Motivated by Lemma 3.32, we can further simplify it and just choose $D := \|A\|I$, for example.

Heuristically, the error becomes smooth after a few relaxation steps, and we can expect the coarse space to approximate a smooth vector v accurately if the coarse space is chosen appropriately. Motivated by Theorem 7.10 and Equation (7.14), we propose Assumption 7.17:

asmp2 **Assumption 7.17** (Weak approximability)**.** $\|(I - PR)v\|_D \leqslant \beta \|v\|_A, \quad \forall\, v \in V.$

In view of Remark 3.42, we assume that the prolongation operator preserves constants (Assumption 7.18). In fact, from the weak approximation property (Assumption 7.17) 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:

asmp1 **Assumption 7.18** (Constant preserving)**.** $P\mathbf{1}_{N_c} = \mathbf{1}_N$.

Unlike in the geometric setting, considering convergence alone is not meaningful in the algebraic multigrid context. This is because the computational complexity of each AMG cycle could be prohibitively high. Compare this with the GMG complexity discussed in §6.3. A uniformly convergent AMG method could be still very slow. Hence the complexity of the multilevel hierarchy for AMG requires special attention.

:OpComplexity **Remark 7.19** (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 $\operatorname{nnz}(\cdot)$ 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 large linear systems on parallel computers. Sometimes, we have to truncate the "insignificant" nonzero entries or specify sparsity patterns to reduce complexity [89]. □

7.3 Classical algebraic multigrid methods

:ClassicalAMG

The original AMG approach (the classical AMG) [56] was developed under the assumption that A is an M-matrix. 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.

7.3.1 General AMG setup phase

We have presented a general framework for implementing multigrid methods in Section 7.1. Here, we provide a general two-level setup procedure suitable for AMG methods (including both the classical AMG and aggregation-based AMG approaches). This setup algorithm can then be applied recursively to construct a multilevel hierarchy until the coarse grid size N_c becomes sufficiently small or the coarse grid matrix A_c becomes too dense to continue coarsening.

alg:AMGsetup

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 a 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, one might be able to perform a preprocessing step to obtain a more suitable matrix A_f. This step can be used as a way to introduce an auxiliary space method.

2. In classical AMG methods, we use the so-called C/F splitting, namely, split all N variables into two sets: N_c C-variables and N_f F-variables, i.e., $N = N_c + N_f$. On the other hand, aggregation-based AMG forms aggregates of fine-level variables (vertices) to cover the graph G.

3. As observed previously, forming an interpolation P that satisfies the weak approximation property is crucial for convergence. This task can be further divided into two stages: (1) Determining the sparsity pattern; (2) Assigning weights to P. Sometimes, we can "truncate" the matrix P by eliminating small entries, if P is not sparse enough.

4. For symmetric problems, the Galerkin relation naturally leads to assuming $R = P^T$. But for nonsymmetric problems, R may also need to be constructed.

5. In GMG, the coarse-level problems can 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. This can easily become the most time-consuming part of the setup phase. Implementing this part requires attention, especially for parallel efficiency.

6. Sometimes, A_c might not be sparse enough even after P is truncated when using the Galerkin relation. In this case, A_c need to be further modified to obtain a sparse approximation.

7.3.2 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 7.16, 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 we say that the variable i is *strongly negatively coupled* or *strongly n-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 coupled variables* by considering the positive coupling as well.

Remark 7.20 (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 can be used in the aggregation-based methods in the next section. □

Denote further

$$\mathsf{S}_j := \big\{ i \in \mathsf{N}_j : \ i \ \text{ strongly coupled to } \ j \big\} \quad \text{and} \quad \mathsf{S}_j^T := \big\{ i \in \mathsf{V} : \ j \in \mathsf{S}_i \big\}.$$

So S_j is the set of indices which *affects* j and S_j^T is the ones which are *affected* by j. After finding the strongly coupled variables, we can filter the coefficient matrix to obtain a filtered matrix A_{S} by removing all non-strongly coupled connections, namely, by dropping "insignificant" nonzeros entries from A.

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 7.23). A variable i is said *strongly coupled* to another variable j along a path of length ℓ if there exists a sequence of edges

$$\big\{ (i,\, k_1),\, (k_1, k_2),\, \ldots,\, (k_{\ell-2}, k_{\ell-1}),\, (k_{\ell-1}, j) \big\} \subseteq \mathsf{E}$$

such that $k_{l+1} \in \mathsf{S}_{k_l}$ for $l = 1, 2, \ldots, \ell - 2$. If there exist at least one path of length less than or equal to ℓ such that i is strongly coupled to j, then we say that i is ℓ-strongly coupled to j and denoted by $j \in \mathsf{S}_i^\ell$.

We note that, based on the nonzero pattern of A^ℓ or a filtered version 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 **??** for powers of the matrix A.

7.3.3 C/F splitting

The classical Ruge–Stüben method splits the set of vertices V into two non-intersecting sets: the fine variables F and the coarse variables C. All indices in F are affected by some index in

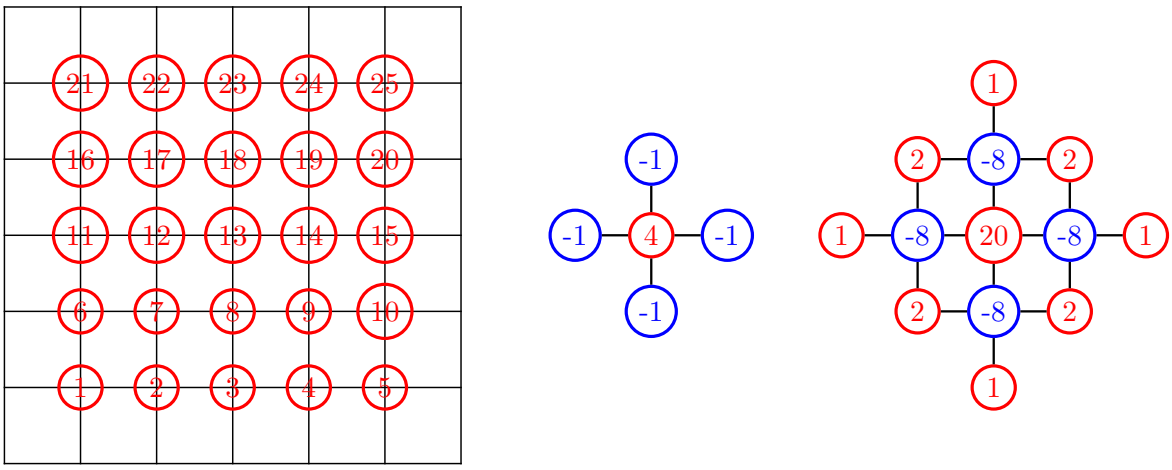


Figure 7.5: Finite difference grid (left), strong connections and weights in A of vertex 13 (middle), and 2-strong connections and weights in A^2 of 13 (right).

ngconnections

A =

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	-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

Figure 7.6: The stiffness matrix A for five-point stencil finite difference scheme on the mesh given in Figure 7.5 (left).

fig:A1

```
>> 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     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
     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0    -8     2     0     0     0    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     2     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).

fig:A2

C, while C contains as few entries as possible. Then, F is chosen as the set of indices of finer grid nodes, and C is chosen as the set of indices of coarse grid nodes. The indices of nodes are assigned as coarse or fine successively. Let U denote the set of indices of nodes not yet assigned. We summarize the algorithm in the following subroutine:

Listing 7.1: Classical C/F splitting method

```
1  U ← V,  C ← ∅,  F ← ∅;
2  while U ≠ ∅
3      λ_i ← 2|S_i^T ⋂ F| + |S_i^T ⋂ U|,  i ∈ U;
4      k ← arg max{λ_i, i ∈ U};
5      C ← C ⋃ {k},  U ← U\{k};
6      F ← F ⋃ S_k^T,  U ← U\S_k^T;
7  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 extremely important for the classical AMG because it usually yields coarsening ratios that are relatively close to 1.0.

- 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 part of the fine variables.

- 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 are strongly coupled to it.
- In the later stage, vertices that strongly coupled to more 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 7.21 (Isolated points)**.** Before starting the above algorithm, isolated points (like Dirichlet boundary points) are usually filtered out and simply defined as F-variables. Similarly, if a point has very strong diagonal dominance, it can also be safely considered isolated and moved to F. □

Remark 7.22 (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. □

em:Aggressive **Remark 7.23** (Aggressive coarsening)**.** In practice, the standard C/F splitting scheme given above usually results in high operator complexity (refer to Remark 7.19), which leads to high computational and storage demands; see Table 7.1. In such cases, we can apply the so-called

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}).

ab:coarsening

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-variables only, we don't need all entries of A_S^2). □

Example 7.24 (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 $u_{i,j-1}$ mainly. Thus if we apply the C/F procedure,

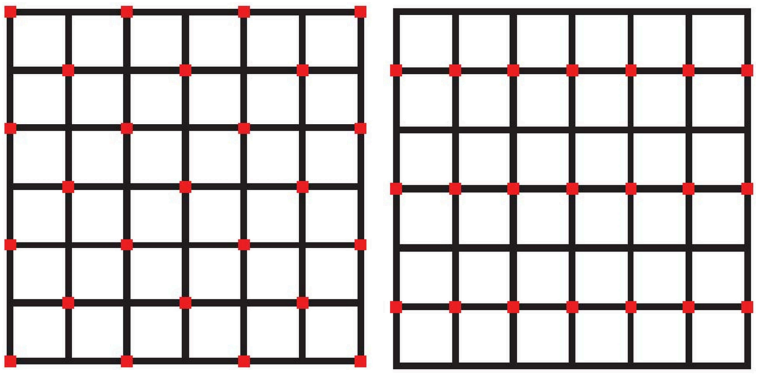


Figure 7.8: C/F splitting for the 2D elliptic problem with $\epsilon = 1$ (left) and $\epsilon \ll 1$ (right), where the red points correspond to C-variables and the black points correspond to F-variables.

ig:coarsening

the coarsening will take place indeed in one direction only (semi-coarsening); see Figure 7.8. □

7.3.4 Construction of prolongation

After obtaining a C/F splitting, upon 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

$$\begin{pmatrix} A_{\mathsf{C},\mathsf{C}} & A_{\mathsf{C},\mathsf{F}} \\ A_{\mathsf{F},\mathsf{C}} & A_{\mathsf{F},\mathsf{F}} \end{pmatrix} \begin{pmatrix} u_{\mathsf{C}} \\ u_{\mathsf{F}} \end{pmatrix} = \begin{pmatrix} f_{\mathsf{C}} \\ f_{\mathsf{F}} \end{pmatrix}$$

Let $e^H \in \mathbb{R}^{N_c}$ be a vector corresponding to the coarse variables. 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 geometric point of view, it is natural to expect

$$a(\phi_j^H, \phi_i^h) = 0, \quad j \in \mathsf{C},\ i \in \mathsf{F}. \tag{7.21}$$ eqn:CFconditi

In Chapter 3, we have observed that

$$(\phi_1^H, \ldots, \phi_{N_c}^H) = (\phi_1^h, \ldots, \phi_N^h)P.$$

It is then trivial to see that, we should have $(Pe^H)_j = e_j^H$, if $j \in \mathsf{C}$. Hence, it is natural to define that

$$P := \begin{pmatrix} I_{\mathsf{C}} \\ W \end{pmatrix},$$

where $I_{\mathsf{C}} \in \mathbb{R}^{N_c \times N_c}$ is the identity matrix and $W \in \mathbb{R}^{(N-N_c)\times N_c}$. In the matrix form, the condition (7.21) can be written as

$$\begin{pmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & I_{\mathsf{F}} \end{pmatrix} \begin{pmatrix} A_{\mathsf{C},\mathsf{C}} & A_{\mathsf{C},\mathsf{F}} \\ A_{\mathsf{F},\mathsf{C}} & A_{\mathsf{F},\mathsf{F}} \end{pmatrix} \begin{pmatrix} I_{\mathsf{C}} \\ W \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}.$$

That is to say,

$$A_{\mathsf{F},\mathsf{C}} + A_{\mathsf{F},\mathsf{F}} W = \mathbf{0}$$

or

$$W = -A_{\mathsf{F},\mathsf{F}}^{-1} A_{\mathsf{F},\mathsf{C}}.$$

It is easy to check that this prolongation matrix P satisfies Assumption 7.18 if the row-sum of A is zero. However, this prolongation is too expensive to compute in practice. There are many different ways to approximate W by a simpler sparse matrix.

1) Direct interpolation scheme

For the smooth error component $e^h \in \mathbb{R}^N$, we have

$$A_{\mathsf{F},\mathsf{F}}\, e_{\mathsf{F}}^h + A_{\mathsf{F},\mathsf{C}}\, e_{\mathsf{C}}^h \ll \mathbf{1} \quad \Longrightarrow \quad \sum_{j=1}^{N} a_{i,j} e_j^h \approx 0, \quad i \in \mathsf{F}.$$

Motived by the above observation, we can also assume that

$$a_{i,i} e_i^h + \sum_{j \in \mathsf{N}_i} a_{i,j} e_j^h = 0, \quad i \in \mathsf{F}. \tag{7.22}$$ eqn:GInterp

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. In this way, we obtain

$$a_{i,i}e_i^h + \sum_{j\in\mathsf{S}_i} a_{i,j}e_j^h = 0, \quad i \in \mathsf{F}. \tag{7.23}$$ eqn:SInterp

We approximate the above equation (7.22) with

$$a_{i,i}e_i^h + \alpha_i \sum_{j\in\mathsf{N}_i\bigcap\mathsf{C}} a_{i,j}e_j^h = 0, \quad \alpha_i = \frac{\sum_{k\in\mathsf{N}_i} a_{i,k}}{\sum_{k\in\mathsf{N}_i\bigcap\mathsf{C}} a_{i,k}}, \quad i \in \mathsf{F}.$$

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

$$e_i^h = \sum_{j\in\mathsf{N}_i\bigcap\mathsf{C}} w_{i,j}e_j^H \qquad \text{and} \qquad w_{i,j} = \frac{a_{i,j}}{\sum_{k\in\mathsf{N}_i\bigcap\mathsf{C}} a_{i,k}}. \tag{7.24}$$ eqn:direct-in

In this case, the matrix form is just $W = \big(\mathrm{diag}(A_{\mathsf{F},\mathsf{C}}\mathbf{1})\big)^{-1}A_{\mathsf{F},\mathsf{C}}$. It is straightforward to show that Assumption 7.18 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

$$a_{i,i}e_i^h + \alpha_i \sum_{j\in\mathsf{P}_i} a_{i,j}e_j^h = 0, \quad \alpha_i = \frac{\sum_{k\in\mathsf{N}_i} a_{i,k}}{\sum_{k\in\mathsf{P}_i} a_{i,k}}, \quad \forall\, i \in \mathsf{F}.$$

If the i-th row has zero row-sum, then this gives the well-known direct interpolation

$$e_i^h = \sum_{j\in\mathsf{P}_i} w_{i,j}e_j^H \qquad \text{and} \qquad w_{i,j} = \frac{a_{i,j}}{\sum_{k\in\mathsf{P}_i} a_{i,k}}. \tag{7.25}$$ eqn:direct-in

2) Standard interpolation scheme

In the equation (7.22), we can first eliminate all e_j^h for $j \in \mathsf{S}_i\bigcap\mathsf{F}$, using the j-th equation, by the approximation

$$e_j^h := -\frac{1}{a_{j,j}} \sum_{k\in\mathsf{N}_j} a_{j,k}e_k^h.$$

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

$$a_{i,i}e_i^h + \sum_{j\in \mathsf{P}_i} a_{i,j}e_j^H + \sum_{j\in \mathsf{S}_i\backslash \mathsf{P}_i} a_{i,j}e_j^h = 0, \quad i \in \mathsf{F}.$$

Therefore, in order to obtain an interpolation matrix W for P, 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

$$\begin{cases} e_i^h = e_i^H, & i \in \mathsf{C} \\ a_{i,i}e_i^h + \sum_{j\in \mathsf{P}_i} a_{i,j}e_j^H + \sum_{j\in \mathsf{S}_i\backslash \mathsf{P}_i} a_{i,j}\dfrac{\sum_{k\in \mathsf{P}_i} a_{i,k}e_k^h}{\sum_{k\in \mathsf{P}_i} a_{i,k}} = 0, & i \in \mathsf{F}. \end{cases} \tag{7.26}$$

intprol

This is the so-called Jacobi interpolation method.

Remark 7.25 (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, 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 7.26 (Initial guess of weights)**.** If the initial guess $W^{(0)}$ preserves constants, then we get

$$W - W^{(k)} = \left(I - D_{\mathsf{F},\mathsf{F}}^{-1}A_{\mathsf{F},\mathsf{F}}\right)^k (W - W^{(0)}).$$

Since both W and $W^{(0)}$ preserves constants, all improved weights $W^{(k)}$ also preserve constants by iteration. □

7.4 Aggregation-based algebraic multigrid methods

ggregationAMG

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 coarse variable.

7.4.1 Unsmoothed aggregation AMG

There are multiple sophisticated methods for forming aggregates. In principle, any combinatorial graph partitioning algorithm can be applied to create these aggregations. We begin by introducing a straightforward greedy algorithm for forming aggregates, which is based on the concept of a maximum independent set, as 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
```

After the above aggregation procedure, 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.

Whence aggregations of the fine variables are formed, it is easy to define the prolongation matrix, for $1 \leqslant i \leqslant N$ and $1 \leqslant j \leqslant N_c$, by

$$\big(P\big)_{i,j} = \begin{cases} 1, & \text{if } i \in \mathsf{C}_j; \\ 0, & \text{if } i \notin \mathsf{C}_j. \end{cases}$$

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

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \in \mathbb{R}^{N \times N_c}. \tag{7.27}$$ eqP

Figure 7.9 shows the aggregations 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 can be written as:

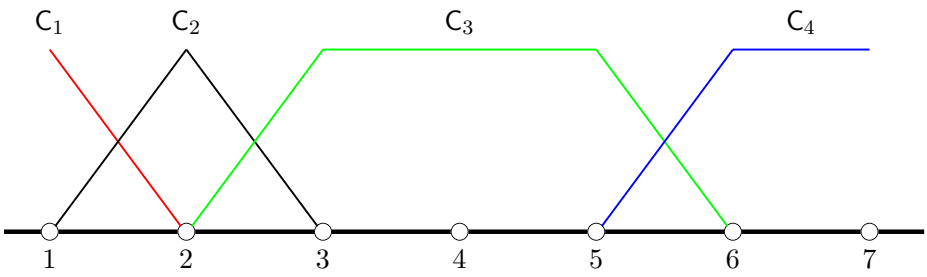


Figure 7.9: Aggregates and prolongation corresponding to (7.27).

ig:aggregates

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
```

7.4.2 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 §6.3. 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 $(A)_{k_i,j} = 0$ 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 $(A\mathbf{1}_i)_{k_i} = 0$.

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

$$P_{\text{SA}} := \big(\psi_1, \psi_2, \ldots, \psi_{N_c}\big).$$

Define $\mathbf{1}_c := (1, \ldots, 1)^T \in \mathbb{R}^{N_c}$. Hence we have $P_{\text{SA}}\mathbf{1}_c = \mathbf{1}$. Furthermore, the coarse level matrix $A_c = P_{\text{SA}}^T A P_{\text{SA}}$ satisfies that

$$A_c\,\mathbf{1}_c = (P_{\text{SA}}^T A P_{\text{SA}})\,\mathbf{1}_c = P_{\text{SA}}^T A\,\mathbf{1} = \mathbf{0}.$$

By applying this definition recursively, we can finish the AMG setup for the smoothed aggregation method.

Listing 7.4: Smoothed aggregation method

```
U ← V;
for i ∈ U
    S_i ← {j ∈ U : j is strongly coupled to i};
    construct a column of prolongation P based on S_i;
    U ← U\({i} ⋃ S_i);
end
Smooth the basis functions using the weighted Jacobi method P_SA = (I − ωD^{-1}A)P;
```

Aggregation method	SA [186]	UA [186]	Pairwise UA [154]
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}).

b:aggregation

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.

Part III

Applications of Multilevel Iterative Methods

Chapter 8

Fluid Problems

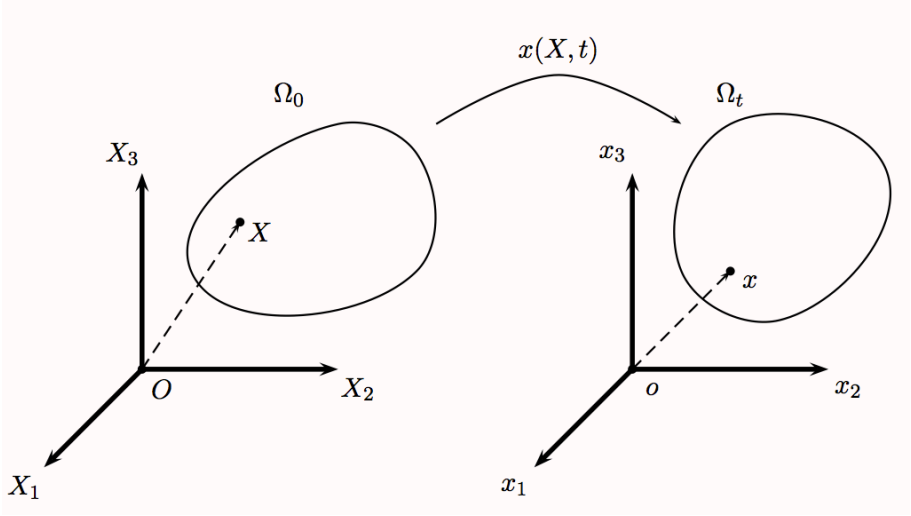
Computational fluid dynamics (CFD) is a branch of fluid mechanics that uses numerical analysis and algorithms to solve and analyze fluid problems. Computers are used to perform the calculations required to simulate liquids or/and gases with surfaces defined by boundary conditions. The fundamental basis 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.

8.1.1 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



configuration

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

$$\dot{x} = \frac{dx(X,t)}{dt} = \mathbf{u}(x,t) \quad \text{and} \quad x(X,0) = X. \tag{8.1}$$

eqn:flowmap

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

$$F := \frac{\partial x}{\partial X} \quad \text{and} \quad J := |F| = \det(F), \tag{8.2}$$

eqn:defgrad

respectively. F is also called the Jacobian matrix.

For any function $f(\cdot, t) : \Omega_t \mapsto \mathbb{R}$, we can easily derive that

$$\dot{f} := \frac{df(x,t)}{dt} = \nabla f \cdot \frac{dx}{dt} + f_t = f_t + \mathbf{u} \cdot \nabla f, \tag{8.3}$$

eqn:fdot

which is usually known as 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

$$\dot{J} = J \operatorname{tr}(F^{-1}\dot{F}). \tag{8.4}$$ eqn:JacobiFor

Hence we can immediately obtain

$$\dot{J} = J \operatorname{tr}(\frac{\partial X}{\partial x}\frac{\partial \dot{x}}{\partial X}) = J \operatorname{tr}(\nabla \mathbf{u}) = J(\nabla \cdot \mathbf{u}), \tag{8.5}$$ eqn:Jdot

which in turn gives an ODE for J, i.e.

$$\dot{J} = (\nabla \cdot \mathbf{u})J \quad \text{and} \quad J(0) = 1. \tag{8.6}$$ eqn:Jode

On the other hand, we can derive similar results for the deformation gradient F itself:

$$\dot{F} = \frac{d}{dt}(\frac{\partial x}{\partial X}) = \frac{\partial \dot{x}}{\partial X} = \frac{\partial \mathbf{u}}{\partial X} = \nabla \mathbf{u} F. \tag{8.7}$$ eqn:Fdot

We can easily immediately see that

$$F_t + \mathbf{u} \cdot \nabla F = \nabla \mathbf{u}\, F \quad \text{and} \quad F(0) = I. \tag{8.8}$$ eqn:F

8.1.2 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*.

lem:transport **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

$$\rho_t + \nabla\cdot(\rho\mathbf{u}) = 0 \quad \text{and} \quad \dot{\rho} + \rho\nabla\cdot\mathbf{u} = 0, \tag{8.10}$$ ean:fluidcon

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

$$\rho(x(X,t),t) = \frac{\rho_0(X)}{J}. \tag{8.11}$$ eqn:rhoratio

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:

$$\rho_t + (\mathbf{u}\cdot\nabla)\rho = 0 \quad \text{or} \quad \dot{\rho} = 0. \tag{8.12}$$ ean:fluidcon1

Together with $\rho(X,0) = \rho_0$ being a constant, we can get $\rho \equiv \rho_0$ for all time $t \in [0,T]$.

8.1.3 Balance of momentum

Now we consider the incompressible Newtonian fluids. Due to the Newton's Second Law, we have the balance of momentum

$$\frac{d}{dt}\int_{\Omega_t} \rho\mathbf{u}\,dx = \text{Force}(\Omega_t). \tag{8.13}$$ eqn:Newton2

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

$$\frac{d}{dt}\int_{\Omega_t}\rho\mathbf{u}\,dx=\int_{\Omega_t}\rho(\mathbf{u}_t+\mathbf{u}\cdot\nabla\mathbf{u})\,dx. \tag{8.14}$$

eqn:momentumL

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 (for example, the gravity), $\mathbf{T}$ is the traction tensor on the boundary of Ω_t, and $\mathbf{n}$ is the outer normal direction on the boundary $\partial\Omega_t$.

Remark 8.2 (Traction force)**.** The exact form of $\mathbf{T}$ depends on the underlying constitutive laws. For Newtonian fluids, the traction can be defined as

$$\mathbf{T}:=-pI+2\mu\boldsymbol{\varepsilon}(\mathbf{u}), \tag{8.15}$$

eqn:T

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

$$2\,\nabla\cdot\boldsymbol{\varepsilon}(\mathbf{u})=\Delta\mathbf{u}. \tag{8.16}$$

eqn:div_Lapla

This way we can get the momentum equation (balance of force) for incompressible Newtonian fluids:

$$\rho(\mathbf{u}_t+\mathbf{u}\cdot\nabla\mathbf{u})=-\nabla p+\mu\Delta\mathbf{u}. \tag{8.17}$$

eqn:momentum

If the density ρ is a constant, we further simplify the above equation (by modifying the definition of p and μ) to give

$$\mathbf{u}_t+\mathbf{u}\cdot\nabla\mathbf{u}=-\nabla p+\mu\Delta\mathbf{u}. \tag{8.18}$$

eqn:momentum1

8.1.4 Mathematical models

ssc:model

To summarize, we have derived the mathematical model for incompressible Newtonian fluids, i.e., the *Navier–Stokes* (NS) equations:

$$\begin{cases}
\rho(\mathbf{u}_t+\mathbf{u}\cdot\nabla\mathbf{u})-\mu\Delta\mathbf{u}+\nabla p=\mathbf{f}, & \Omega_t & \text{balance of momentum;}\\
\rho_t+\nabla\cdot(\rho\mathbf{u})=0, & \Omega_t & \text{conservation of mass;}\\
\nabla\cdot\mathbf{u}=0, & \Omega_t & \text{incompressibility;}\\
\mathbf{u}=0, & \partial\Omega_t & \text{no-slip boundary;}\\
\mathbf{u}|_{t=0}=\mathbf{u}_0, & \Omega_t & \text{initial condition.}
\end{cases} \tag{8.19}$$

eqn:NS

If we assume the density ρ is a constant, then we can write (8.19) as follows:

$$\begin{cases} \mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} - \mu\Delta\mathbf{u} + \nabla p = \mathbf{f}, & \Omega_t & \text{momentum equation;} \\ \nabla\cdot\mathbf{u} = 0, & \Omega_t & \text{continuity equation;} \\ \mathbf{u} = 0, & \partial\Omega_t & \text{no-slip boundary;} \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega_t & \text{initial condition.} \end{cases} \tag{8.20}$$

eqn:NS1

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:

$$\begin{cases} \rho(\mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u}) + \nabla p = 0, & \Omega_t & \text{balance of momentum;} \\ \rho_t + \nabla\cdot(\rho\mathbf{u}) = 0, & \Omega_t & \text{conservation of mass;} \\ \nabla\cdot\mathbf{u} = 0, & \Omega_t & \text{incompressibility;} \\ \mathbf{u}\cdot\mathbf{n} = 0, & \partial\Omega_t & \text{no-flow boundary;} \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega_t & \text{initial condition.} \end{cases} \tag{8.21}$$

eqn:Euler

If the density ρ is a constant, then we have the following simplified form:

$$\begin{cases} \mathbf{u}_t + \mathbf{u}\cdot\nabla\mathbf{u} + \nabla p = 0, & \Omega_t & \text{momentum equation;} \\ \nabla\cdot\mathbf{u} = 0, & \Omega_t & \text{continuity equation;} \\ \mathbf{u}\cdot\mathbf{n} = 0, & \partial\Omega_t & \text{no-flow boundary;} \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega_t & \text{initial condition.} \end{cases} \tag{8.22}$$

eqn:Euler1

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.

8.2.1 The time-dependent Stokes equation

On an open bounded set $\Omega \subset \mathbb{R}^d$, we consider

$$\begin{cases} \mathbf{u}_t - \mu\Delta\mathbf{u} + \nabla p = \mathbf{f}, & \Omega; \\ \nabla\cdot\mathbf{u} = 0, & \Omega; \\ \mathbf{u} = 0, & \partial\Omega; \\ \mathbf{u}|_{t=0} = \mathbf{u}_0, & \Omega. \end{cases} \tag{8.23}$$

eqn:Stokes

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

$$\left\{\begin{array}{rcll}(\mathcal{I}-\epsilon^2\Delta)\mathbf{u}+\nabla p &=& \mathbf{f}, & \Omega;\\ \nabla\cdot\mathbf{u} &=& 0, & \Omega;\\ \mathbf{u} &=& 0, & \partial\Omega.\end{array}\right. \tag{8.24}$$ eqn:Stokes1ti

We can further simplify the discussion and only consider the following steady-state Stokes equations, i.e.,

$$\left\{\begin{array}{rcll}-\Delta\mathbf{u}+\nabla p &=& \mathbf{f}, & \Omega;\\ \nabla\cdot\mathbf{u} &=& 0, & \Omega;\\ \mathbf{u} &=& 0, & \partial\Omega.\end{array}\right. \tag{8.25}$$ eqn:Stokes1

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

$$\left\{\begin{array}{rcll}2\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\, dx + (p, \nabla\cdot\mathbf{v}) &=& (\mathbf{f},\mathbf{v}), & \forall\, \mathbf{v} \in \mathscr{V};\\ (\nabla\cdot\mathbf{u},\, q) &=& 0, & \forall\, q \in \mathscr{Q}.\end{array}\right. \tag{8.26}$$ eqn:Stokes1w

The derivation is straightforward and hence leave to the readers; see HW 8.2.

Remark 8.3 (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. □

8.2.2 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

$$\left\{\begin{array}{rcll}a[u,v] + b[v,p] &=& \langle f, v\rangle, & \forall v \in \mathscr{V};\\ b[u,q] &=& \langle g, q\rangle, & \forall q \in \mathscr{Q}.\end{array}\right. \tag{8.27}$$ eqn:saddle

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

$$\mathscr{Z} := \text{null}(\mathcal{B}) = \left\{v \in \mathscr{V} \ :\ b[v, q] = 0,\ \forall q \in \mathscr{Q}\right\} \subset \mathscr{V}.$$

Because $b[\cdot, \cdot]$ is continuous, $\mathscr{Z}$ is closed. Hence we can give an orthogonal decomposition

$$\mathscr{V} = \mathscr{Z} \oplus \mathscr{Z}^\perp = \text{null}(\mathcal{B}) \oplus \text{null}(\mathcal{B})^\perp.$$

For any $u \in \mathscr{V}$, we have $u = u_0 + u_\perp$, with $u_0 \in \text{null}(\mathcal{B})$ and $u_\perp \in \text{null}(\mathcal{B})^\perp$.

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 §1.1.4, we can see that, if $\mathcal{B}$ is surjective, namely,

$$\inf_{q \in \mathscr{Q}} \sup_{v \in \mathscr{V}} \frac{b[v, q]}{\|v\|_{\mathscr{V}} \|q\|_{\mathscr{Q}}} = \beta > 0, \tag{8.28}$$ eqn:infsupB

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

$$\inf_{u \in \mathscr{Z}} \sup_{v \in \mathscr{Z}} \frac{a[u, v]}{\|u\|_{\mathscr{V}} \|v\|_{\mathscr{V}}} = \inf_{v \in \mathscr{Z}} \sup_{u \in \mathscr{Z}} \frac{a[u, v]}{\|u\|_{\mathscr{V}} \|v\|_{\mathscr{V}}} = \alpha > 0. \tag{8.29}$$ eqn:infsupA

With the conditions (8.29) and (8.28), we obtain a unique solution $u = u_0 + u_\perp$.

[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 $\text{null}(\mathcal{B})$. Due to $u_\perp - \tilde{u}_\perp \in \text{null}(\mathcal{B})^\perp$, we find $u_\perp - \tilde{u}_\perp = 0$.

We can find the solution for the pressure variable by solving

$$\mathcal{B}^T p = f - \mathcal{A}u. \tag{8.30}$$ eqn:Bp

For any $v \in \mathscr{Z} = \text{null}(\mathcal{B})$, 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 [63, Theorem 1.1]:

thm:Brezzi **Theorem 8.4** (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

$$\|u\|_{\mathscr{V}} + \|p\|_{\mathscr{Q}} \lesssim \|f\|_{\mathscr{V}'} + \|g\|_{\mathscr{Q}'}.$$

mixed-inf-sup **Remark 8.5** (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

$$\tilde{a}[(u,p),(v,q)] = \langle f, v\rangle + \langle g, q\rangle, \quad \forall (v,q) \in \mathscr{X}. \tag{8.31}$$ eqn:saddle2

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. □

8.2.3 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 8.6 (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 [99, III.3.1]. □

Remark 8.7 (Existence of solution)**.** It has been shown in the above lemma that range$(\mathcal{B}) = L^2(\Omega)/\mathbb{R} \cong \mathscr{Q}$. Or equivalently, we have null$(\mathcal{B}^T) \bigcap \mathscr{Q} = \{0\}$. □

Using the previous lemma and the Brezzi theorem, we can easily get the following result:

kes_wellposed **Theorem 8.8** (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

$$\|\mathbf{u}\|_1 + \|p\|_0 \lesssim \|\mathbf{f}\|_{-1}.$$

8.2.4 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

$$\text{Find} \ \ \mathbf{u} \in \mathscr{V}: \quad 2\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}) : \boldsymbol{\varepsilon}(\mathbf{v})\, dx + \gamma(\nabla \cdot \mathbf{u}, \nabla \cdot \mathbf{v}) = (\mathbf{f}, \mathbf{v}), \quad \forall\, \mathbf{v} \in \mathscr{V}. \tag{8.34}$$ eqn:Stokes2w

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 [130].

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

$$\begin{cases} 2\displaystyle\int_\Omega \boldsymbol{\varepsilon}(\mathbf{u}_h) : \boldsymbol{\varepsilon}(\mathbf{v}_h)\, dx - (p_h, \nabla \cdot \mathbf{v}_h) &= (\mathbf{f}, \mathbf{v}_h), & \forall\, \mathbf{v}_h \in V_h, \\ (\nabla \cdot \mathbf{u}_h, q_h) &= 0, & \forall\, q_h \in Q_h. \end{cases} \tag{8.35}$$ eqn:Stokes1d

[2] The computed velocity is vanishing or unnaturally small for big λ.

The existence of the discrete solution $(\mathbf{u}_h, p_h)$ is straightforward due to the conformity of the approximation spaces.

8.3.1 Well-posedness and convergence

Let $Z_h = \text{null}(\mathcal{B}_h)$ be the kernel of the discrete divergence operator. In fact, the coercivity of $a[\cdot,\cdot]$ yields that

$$\inf_{\mathbf{u}_h \in Z_h} \sup_{\mathbf{v}_h \in Z_h} \frac{a[\mathbf{u}_h, \mathbf{v}_h]}{\|\mathbf{u}_h\|_1 \|\mathbf{v}_h\|_1} = \alpha_h > 0. \tag{8.36}$$ eqn:inf_sup_A

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 $Z_h \not\subset \mathscr{Z}$, then there is a variational crime and we have following estimate:

$$\|\mathbf{u} - \mathbf{u}_h\|_{\mathscr{V}} \leqslant \left(1 + \frac{C_a}{\alpha_h}\right) \inf_{\mathbf{v} \in Z_h} \|\mathbf{u} - \mathbf{v}\|_{\mathscr{V}} + \frac{1}{\alpha_h} \sup_{\mathbf{w} \in Z_h \setminus \{0\}} \frac{\left|a[\mathbf{u} - \mathbf{u}_h, \mathbf{w}]\right|}{\|\mathbf{w}\|_{\mathscr{V}}}.$$

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

$$\left|a[\mathbf{u} - \mathbf{u}_h, \mathbf{w}]\right| \leqslant C_b \|\mathbf{w}\|_{\mathscr{V}} \|p - q\|_{\mathcal{Q}}.$$

We can then conclude with the following best approximation result:

Lemma 8.9 (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

$$\|\mathbf{u} - \mathbf{u}_h\|_{\mathscr{V}} \leqslant \left(1 + \frac{C_a}{\alpha_h}\right) \inf_{\mathbf{v} \in Z_h} \|\mathbf{u} - \mathbf{v}\|_{\mathscr{V}} + \frac{C_b}{\alpha_h} \inf_{q \in Q_h} \|p - q\|_{\mathscr{Q}}.$$

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 $\text{null}(\mathcal{B}_h^T)$ is not trivial, then $\text{range}(\mathcal{B}_h)$ 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:

$$\inf_{q_h \in Q_h} \sup_{\mathbf{v}_h \in V_h} \frac{b(\mathbf{v}_h, q_h)}{\|\mathbf{v}_h\|_1 \|q_h\|_0} = \beta_h > 0. \tag{8.37}$$ eqn:inf_sup_B

The condition $\text{null}(\mathcal{B}_h^T) = \{0\}$ is necessary for the inf-sup condition above. If $\text{null}(\mathcal{B}_h^T)$ is non-trivial, then the numerical solution p_h is not unique, namely, $p_h + s_h$ is also a solution when $s_h \in \text{null}(\mathcal{B}_h^T)$. In this case, we usually find the computed pressure is oscillatory and, hence, $\text{null}(\mathcal{B}_h^T)$ is often referred to as the space of spurious pressure modes.

Theorem 8.10 (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

$$\|\mathbf{u} - \mathbf{u}_h\|_{\mathscr{V}} + \|p - p_h\|_{\mathscr{Q}} \lesssim \inf_{\mathbf{v} \in Z_h} \|\mathbf{u} - \mathbf{v}\|_{\mathscr{V}} + \inf_{q \in Q_h} \|p - q\|_{\mathscr{Q}}.$$

8.3.2 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 8.11 (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

$$C_r := \dim Q_h / \dim V_h.$$

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 [32] 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 $k \geqslant 2$

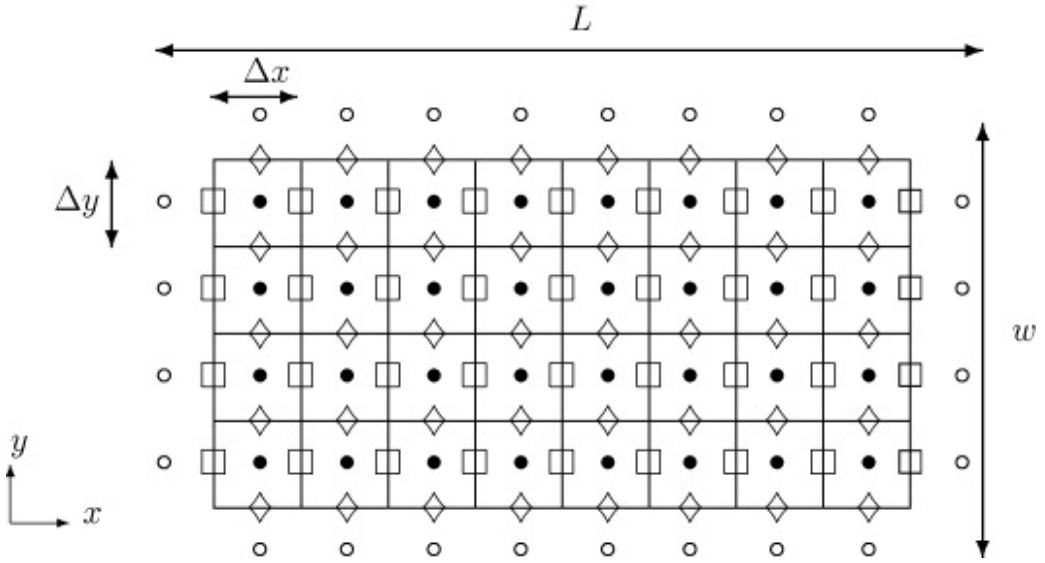


fig:mac

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.

8.3.3 Mixed methods for the Poisson's equation ⋆

:MixedPoisson

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

$$\begin{cases} \mathbf{u} - \nabla p &= \mathbf{f}, \quad \text{in } \Omega; \\ \nabla \cdot \mathbf{u} &= g, \quad \text{in } \Omega; \\ \mathbf{u} \cdot \mathbf{n} &= 0, \quad \text{on } \partial\Omega \end{cases} \tag{8.38}$$

eqn:PoissonMi

In this section, we use this model problem to further motivate 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

$$\begin{cases} (\mathbf{u}, \mathbf{v}) + (p, \nabla \cdot \mathbf{v}) &= \langle \mathbf{f}, \mathbf{v} \rangle, \quad \forall\, \mathbf{v} \in H_0(\mathrm{div}, \Omega); \\ (\nabla \cdot \mathbf{u}, q) &= \langle g, q \rangle, \quad \forall\, q \in L_0^2(\Omega). \end{cases} \tag{8.39}$$

eqn:PoissonMi

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

$$(\mathbf{u}, \mathbf{v})_{H_0(\mathrm{div},\Omega)} := (\mathbf{u}, \mathbf{v}) + (\nabla \cdot \mathbf{u}, \nabla \cdot \mathbf{v}). \tag{8.40}$$

eqn:Hdiv-inne

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

$$\tilde{\mathcal{A}}_0 = \begin{bmatrix} \mathcal{I} & -\mathrm{grad} \\ \mathrm{div} & 0 \end{bmatrix} : H_0(\mathrm{div}, \Omega) \times L_0^2(\Omega) \mapsto H_0(\mathrm{div}, \Omega)' \times L_0^2(\Omega)$$

is an isomorphism. The canonical preconditioner is a block diagonal isomorphism

$$\tilde{\mathcal{D}}_0^{(1)} = \begin{bmatrix} (\mathcal{I} - \mathrm{grad}\,\mathrm{div})^{-1} & 0 \\ 0 & \mathcal{I} \end{bmatrix} : H_0(\mathrm{div}, \Omega)' \times L_0^2(\Omega) \mapsto H_0(\mathrm{div}, \Omega) \times L_0^2(\Omega).$$

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

$$\left\{ \begin{array}{rcll} (\mathbf{u}, \mathbf{v}) - (\nabla p, \mathbf{v}) & = & \langle \mathbf{f}, \mathbf{v} \rangle, & \forall\, \mathbf{v} \in [L^2(\Omega)]^d; \\ -(\mathbf{u}, \nabla q) & = & \langle g, q \rangle, & \forall\, q \in H^1(\Omega) \bigcap L_0^2(\Omega). \end{array} \right. \tag{8.41}$$ eqn:PoissonMi

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

$$\tilde{\mathcal{D}}_0^{(2)} = \left[\begin{array}{cc} \mathcal{I} & 0 \\ 0 & (-\Delta)^{-1} \end{array} \right] : [L^2(\Omega)]^d \times \left(H^1(\Omega) \bigcap L_0^2(\Omega)\right)' \mapsto [L^2(\Omega)]^d \times \left(H^1(\Omega) \bigcap L_0^2(\Omega)\right).$$

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 Classical solvers for the Stokes equation

In this section, we discuss how to construct iterative solvers and preconditioners for the saddle-point problems, like the Stokes equation and the time-dependent Stokes equation. We notice that the corresponding operator $\tilde{\mathcal{A}}$ of the Stokes system

$$\tilde{\mathcal{A}} \left[\begin{array}{c} \mathbf{u} \\ p \end{array} \right] = \left[\begin{array}{c} \mathbf{f} \\ 0 \end{array} \right] \quad \text{with} \quad \tilde{\mathcal{A}} := \left[\begin{array}{cc} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{array} \right] = \left[\begin{array}{cc} -\Delta & \text{grad} \\ -\,\text{div} & 0 \end{array} \right] \tag{8.42}$$ eqn:StokesSys

is an isomorphism mapping from $[H_0^1(\Omega)]^d \times L_0^2(\Omega)$ onto $[H^{-1}(\Omega)]^d \times L_0^2(\Omega)$.

8.4.1 Uzawa method

One of the most well-known iterative method for solving (8.42) is the Uzawa method. Using the first equation, we can first update the velocity with the pressure fixed as the last iteration

$$\mathcal{A}\mathbf{u}^{\text{new}} = f - \mathcal{B}^T p^{\text{old}}.$$

Then we can eliminate the velocity from the second equation and obtain a Schur problem

$$\mathcal{B}\mathcal{A}^{-1}\mathcal{B}^T p = \mathcal{B}\mathcal{A}^{-1}\mathbf{f},$$

which can be simply solved using an iterative method like the Richardson method:

$$p^{\text{new}} = p^{\text{old}} + \omega(\mathcal{B}\mathcal{A}^{-1}\mathbf{f} - \mathcal{B}\mathcal{A}^{-1}\mathcal{B}^T p^{\text{old}}) = p^{\text{old}} + \omega\mathcal{B}\mathbf{u}^{\text{new}}.$$

The above method is called the Uzawa method and it is just the Richardson method 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.

rem:Schur **Remark 8.12** (Schur complement)**.** Since the $\mathcal{A}$ is SPD, the Schur complement $\mathcal{S} = \mathcal{B}\mathcal{A}^{-1}\mathcal{B}^T$ is symmetric and positive semi-definite. Moreover, if $\mathcal{B}$ has full rank, $\mathcal{S}$ is also SPD and we can also apply the CG method to solve the Schur complement equation. However, generally speaking, $\mathcal{S}^{-1}p$ cannot be computed efficiently with acceptable computational cost. Furthermore, since $\mathcal{S}$ is not explicitly given, it is also difficult to apply more powerful iterative method in this direction. Hence the Schur complement $\mathcal{S}$ should be approximated by some approximation $\hat{\mathcal{S}}$. There are many different ways based on approximation of the Schur complement; see the survey paper [19]. In the Stokes problem, the Schur complement $\mathcal{S} := \mathcal{B}\mathcal{A}^{-1}\mathcal{B}^T$ is an SPD and well-conditioned. So it is fine to solve it with such a simple iterative method. □

em:UzawaSpeed **Remark 8.13** (Convergence of Uzawa method)**.** As we have discussed in Remark 2.10, we can employ the optimal weight

$$\omega_{\text{opt}} = \frac{2}{\lambda_{\max}(\mathcal{S}) + \lambda_{\min}(\mathcal{S})}$$

in the Richardso method for the Schur problem. Hence the convergence rate for the pressure variable can be estimated by

$$\|p - p^{(k)}\| \leqslant \left(\frac{\kappa(\mathcal{S}) - 1}{\kappa(\mathcal{S}) + 1}\right)^k \|p - p^{(0)}\|.$$

In many cases, the method could be slow to converge. □

Next, we try to understand the above iterative method in the matrix factorization form.

factorization **Remark 8.14** (Block factorizations)**.** We can apply the following block factorizations to the matrix $\tilde{\mathcal{A}}$ to obtain that

$$\begin{aligned}\begin{bmatrix} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{bmatrix} &= \begin{bmatrix} \mathcal{I}_u & 0 \\ \mathcal{B}\mathcal{A}^{-1} & \mathcal{I}_p \end{bmatrix} \begin{bmatrix} \mathcal{A} & 0 \\ 0 & \mathcal{S} \end{bmatrix} \begin{bmatrix} \mathcal{I}_u & \mathcal{A}^{-1}\mathcal{B}^T \\ 0 & -\mathcal{I}_p \end{bmatrix} \begin{bmatrix} \\ \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{A} & 0 \\ \mathcal{B} & \mathcal{S} \end{bmatrix} \begin{bmatrix} \mathcal{I}_u & \mathcal{A}^{-1}\mathcal{B}^T \\ 0 & -\mathcal{I}_p \end{bmatrix} = \begin{bmatrix} \mathcal{I}_u & 0 \\ \mathcal{B}\mathcal{A}^{-1} & -\mathcal{I}_p \end{bmatrix} \begin{bmatrix} \mathcal{A} & \mathcal{B}^T \\ 0 & \mathcal{S} \end{bmatrix}.\end{aligned}$$

These factorizations are very useful for constructing solvers for the saddle-point systems. □

As the last decomposition in Remark 8.14, we can factorize the coefficient matrix $\tilde{\mathcal{A}}$ as

$$\begin{bmatrix} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{bmatrix} = \begin{bmatrix} \mathcal{I}_u & 0 \\ \mathcal{B}\mathcal{A}^{-1} & -\mathcal{I}_p \end{bmatrix} \begin{bmatrix} \mathcal{A} & \mathcal{B}^T \\ 0 & \mathcal{S} \end{bmatrix}.$$

This means the original linear system (8.42) can be rewritten as

$$\begin{bmatrix} \mathcal{A} & \mathcal{B}^T \\ 0 & \mathcal{S} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathcal{B}\mathcal{A}^{-1}\mathbf{f} \end{bmatrix},$$

which has an upper triangular coefficient matrix. We can apply the Richardson's iteration for the second equation in the above system, i.e.,

$$p^{\text{new}} = p^{\text{old}} + \omega\Big(\mathcal{B}\mathcal{A}^{-1}\mathbf{f} - \mathcal{S}p^{\text{old}}\Big) = p^{\text{old}} + \omega\Big(\mathcal{B}\mathcal{A}^{-1}\mathbf{f} - \mathcal{B}\mathcal{A}^{-1}\mathcal{B}^T p^{\text{old}}\Big).$$

Hence we can plug in the new pressure to the first equation to obtain an alternative direction method

$$\mathcal{A}u^{\text{new}} = \mathbf{f} - \mathcal{B}^T p^{\text{old}}, \quad p^{\text{new}} = p^{\text{old}} + \omega\mathcal{B}u^{\text{new}}. \tag{8.43}$$ eqn:uzawa

rem:InUzawa **Remark 8.15** (Inexact Uzawa method)**.** In each step of the Uzawa method (8.43), we need to solve an elliptic problem efficiently. This can usually be done by a fast Poisson solver. But in some cases, it is not always easy to construct a fast solver for this step (such a case will be soon given in the next subsection). We can also use an inexact method to reduce computational complexity in each step. For example, with $\mathcal{C}$ is a given approximation of $\mathcal{A}^{-1}$, we can simply define

$$u^{\text{new}} = u^{\text{old}} + \mathcal{C}\Big(\mathbf{f} - \mathcal{A}\mathbf{u}^{\text{old}} - \mathcal{B}^T p^{\text{old}}\Big), \quad p^{\text{new}} = p^{\text{old}} + \omega\mathcal{B}\mathbf{u}^{\text{new}}.$$

This method is usually referred to the inexact Uzawa method; for the convergence analysis, we refer to [46, 69]. □

8.4.2 Augmented Lagrangian method

One way to speedup the convergence of the Uzawa type methods is to apply the so-called Augmented Lagrangian method (cf., for example, [96]):

$$(\mathcal{A} + \epsilon^{-1}\mathcal{B}^T\mathcal{B})\mathbf{u}^{\text{new}} = \mathbf{f} - \mathcal{B}^T p^{\text{old}}, \quad p^{\text{new}} = p^{\text{old}} + \epsilon^{-1}\mathcal{B}\mathbf{u}^{\text{new}}. \tag{8.44}$$ eqn:AL

rem:UzawaAL **Remark 8.16** (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

$$\tilde{\mathcal{A}}_\epsilon \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ 0 \end{bmatrix}, \quad \text{where } \tilde{\mathcal{A}}_\epsilon := \begin{bmatrix} \mathcal{A} + \epsilon^{-1}\mathcal{B}^T\mathcal{B} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{bmatrix}. \tag{8.45}$$ eqn:dis-saddl

This modification makes sense because the new term in the first equation is actually zero according to the second equation. So mathematically, the modified problem is equivalent to the original problem. Furthermore, the damping factor ω is now chosen to be ϵ^{-1}. □

thm:AL **Theorem 8.17** (Convergence rate of Augmented Lagrangian method)**.** Let $(\mathbf{u}^{(0)}, p^{(0)})$ be a given initial guess and $(\mathbf{u}^{(m)}, p^{(m)})$ be the iterates obtained via the Augmented Lagrangian method (8.44). Then we have

$$\|p - p^{(m)}\|_0 \leqslant \Big(\frac{\epsilon}{\epsilon + \lambda_1}\Big)^m \|p - p^{(0)}\|_0,$$
$$\|\mathbf{u} - \mathbf{u}^{(m)}\|_{\mathcal{A}} \leqslant \sqrt{\epsilon}\|p - p^{(m-1)}\|_0 \leqslant \sqrt{\epsilon}\Big(\frac{\epsilon}{\epsilon + \lambda_1}\Big)^{m-1} \|p - p^{(0)}\|_0,$$

where λ_1 is the minimal eigenvalue of $\mathcal{S} = \mathcal{B}\mathcal{A}^{-1}\mathcal{B}^T$.

Sketch of proof. From (8.44) and (8.45), we have

$$(\mathcal{A} + \epsilon^{-1}\mathcal{B}^T\mathcal{B})\big(\mathbf{u} - \mathbf{u}^{(m)}\big) = -\mathcal{B}^T\big(p - p^{(m-1)}\big)$$

and

$$p - p^{(m)} = \Big(\mathcal{I} - \mathcal{B}(\epsilon\mathcal{A} + \mathcal{B}^T\mathcal{B})^{-1}\mathcal{B}^T\Big)\big(p - p^{(m-1)}\big).$$

By the Shermann–Morrison–Woodburry formula, we have

$$\mathcal{Z} := \mathcal{B}(\epsilon\mathcal{A} + \mathcal{B}^T\mathcal{B})^{-1}\mathcal{B}^T = \mathcal{S}_\epsilon - \mathcal{S}_\epsilon(\mathcal{I} + \mathcal{S}_\epsilon)^{-1}\mathcal{S}_\epsilon, \quad \mathcal{S}_\epsilon := \epsilon^{-1}\mathcal{B}\mathcal{A}^{-1}\mathcal{B}^T.$$

It is straightforward to verify that

$$\mathcal{I} - \mathcal{Z} = \mathcal{I} - \mathcal{B}(\epsilon\mathcal{A} + \mathcal{B}^T\mathcal{B})^{-1}\mathcal{B}^T = \mathcal{I} - \mathcal{S}_\epsilon + \mathcal{S}_\epsilon(\mathcal{I} + \mathcal{S}_\epsilon)^{-1}\mathcal{S}_\epsilon = (\mathcal{I} + \mathcal{S}_\epsilon)^{-1}.$$

The above equality shows $\rho(\mathcal{Z}) \leqslant 1$ and $p - p^{(m)} = (I + \mathcal{S}_\epsilon)^{-1}\big(p - p^{(m-1)}\big)$. So the first estimate follows immediately. The second estimate is obtained by observing

$$\begin{aligned}\big\|\mathbf{u} - \mathbf{u}^{(m)}\big\|_{\mathcal{A}}^2 &= \Big((\mathcal{A} + \epsilon^{-1}\mathcal{B}^T\mathcal{B} - \epsilon^{-1}\mathcal{B}^T\mathcal{B})\big(\mathbf{u} - \mathbf{u}^{(m)}\big), u - u^{(m)}\Big)\\ &\leqslant \epsilon\big(\mathcal{Z}(p - p^{(m-1)}), p - p^{(m-1)}\big)\end{aligned}$$

and then applying the first estimate. □

According to Theorem 8.17, it seems that we can make the method (8.44) converges 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 $\mathcal{A} + \epsilon^{-1}\mathcal{B}^T\mathcal{B}$, which is more difficult to solve and has been discussed in Remark 2.24; see [127] for a cure.

Remark 8.18 (Augmented Lagrangian preconditioner)**.** We can also apply the Augmented Lagrangian method as a preconditioner

$$\tilde{\mathcal{T}}_{\mathrm{AL}} := \begin{bmatrix} \mathcal{A} + \epsilon^{-1}\mathcal{B}^T\mathcal{B} & 0 \\ \mathcal{B} & \epsilon\mathcal{I} \end{bmatrix}^{-1}, \tag{8.46}$$ eqn:AL-precon

which is often referred to as the AL preconditioner [20].

Remark 8.19 (Stabilized finite element methods)**.** The augmented Lagrangian method is closely related to the grad-div stabilization [64] of the Stokes (or Navier–Stokes) problem:

$$\begin{cases} (I - \mu\Delta)\mathbf{u} - \epsilon^{-1}\nabla\nabla\cdot\mathbf{u} + \nabla p &= \mathbf{f}, \quad \Omega;\\ \nabla\cdot\mathbf{u} &= 0, \quad \Omega;\\ \mathbf{u} &= 0, \quad \partial\Omega. \end{cases} \tag{8.47}$$ eqn:Stokes2

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 (8.44).

8.4.3 Projection method

We obverse that

$$\begin{bmatrix} \mathcal{A} & \mathcal{B}^T \\ \mathcal{B} & 0 \end{bmatrix} \begin{bmatrix} \mathcal{I}_u & \mathcal{B}^T \\ 0 & -\mathcal{B}\mathcal{B}^T \end{bmatrix} = \begin{bmatrix} \mathcal{A} & \mathcal{A}\mathcal{B}^T - \mathcal{B}^T\mathcal{B}\mathcal{B}^T \\ \mathcal{B} & \mathcal{B}\mathcal{B}^T \end{bmatrix} =: \begin{bmatrix} \mathcal{A} & \mathcal{E} \\ \mathcal{B} & \mathcal{B}\mathcal{B}^T \end{bmatrix}.$$

For the Stokes problem, we know that $\mathcal{A} = -\Delta$, $\mathcal{B}^T = \operatorname{grad}$, $\mathcal{B} = -\operatorname{div}$. In this case,

$$\mathcal{E} = (-\Delta + \operatorname{grad}\operatorname{div})\operatorname{grad} = \operatorname{curl}\operatorname{curl}\operatorname{grad} = 0.$$

Although, in practice due to the boundary conditions, $\mathcal{E}$ might not be zero, we can expect this part should be small and can then be omitted. This way, we can construct an iterative method

$$\begin{bmatrix} \mathbf{u}^{\text{new}} \\ p^{\text{new}} \end{bmatrix} = \begin{bmatrix} \mathbf{u}^{\text{old}} \\ p^{\text{old}} \end{bmatrix} + \begin{bmatrix} \mathcal{I}_u & \mathcal{B}^T \\ 0 & -\mathcal{A}_p \end{bmatrix} \begin{bmatrix} \mathcal{A} & 0 \\ \mathcal{B} & \mathcal{A}_p \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{f} - \mathcal{A}\mathbf{u}^{\text{old}} - \mathcal{B}^T p^{\text{old}} \\ -\mathcal{B}\mathbf{u}^{\text{old}} \end{bmatrix}, \tag{8.48}$$

eqn:Projectio

where $\mathcal{A}_p := \mathcal{B}\mathcal{B}^T$ and it is the pressure Laplace operator for the Stokes problem.

In order to provide a more practical iterative procedure, we first solve the following two equations

$$\mathcal{A}\,\delta\mathbf{u} = \mathbf{f} - \mathcal{A}\mathbf{u}^{\text{old}} - \mathcal{B}^T p^{\text{old}}, \quad \mathcal{A}_p\,\delta p = -\mathcal{B}\mathbf{u}^{\text{old}} - \mathcal{B}\,\delta\mathbf{u} = -\mathcal{B}(\mathbf{u}^{\text{old}} + \delta\mathbf{u}).$$

In turn, the above iteration (8.48) can be written as

$$\begin{aligned} &\mathbf{u}^{(k+1/2)} = \mathbf{u}^{(k)} + \delta\mathbf{u}, \\ &\delta p = -\mathcal{A}_p^{-1}\mathcal{B}\mathbf{u}^{(k+1/2)}, \quad \mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1/2)} + \mathcal{B}^T\delta p, \\ &p^{(k+1)} = p^{(k)} + \mathcal{B}\mathbf{u}^{(k+1/2)}. \end{aligned}$$

Notice that

$$\mathcal{B}\mathbf{u}^{(k+1)} = \mathcal{B}\mathbf{u}^{(k+1/2)} + \mathcal{B}\mathcal{B}^T\delta p = \mathcal{B}\mathbf{u}^{(k+1/2)} - \mathcal{B}\mathcal{B}^T\mathcal{A}_p^{-1}\mathcal{B}\mathbf{u}^{(k+1/2)} = 0.$$

Hence the second equation in the saddle-point problem is satisfied exactly by the new iteration. For the Stokes equation, this means the velocity is divergence-free. This is reason why the method is called a projection method.

To summarize, we obtain a simple iterative method

$$\mathbf{u}^{(k+1/2)} = \mathcal{A}^{-1}(\mathbf{f} - \mathcal{B}^T p^{(k)}), \tag{8.49}$$

$$\mathbf{u}^{(k+1)} = \mathbf{u}^{(k+1/2)} - \mathcal{B}^T\mathcal{A}_p^{-1}\mathcal{B}\mathbf{u}^{(k+1/2)}, \tag{8.50}$$

$$p^{(k+1)} = p^{(k)} + \mathcal{B}\mathbf{u}^{(k+1/2)}. \tag{8.51}$$

In the above method, the second step is the so-called projection step and the other two steps coincide with the classical Uzawa method. Futhermore, we also find that the method can be constructed with a velocity Poisson solver and a pressure solver.

8.4.4 Block preconditioners

A natural and widely-used preconditioner for the Stokes system is the classical block diagonal preconditioner

$$\tilde{\mathcal{D}} = \begin{bmatrix} (-\Delta)^{-1} & 0 \\ 0 & \mathcal{I} \end{bmatrix}.$$

This observation immediately motivates the classical block diagonal preconditioner [43].

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 8.5, the stable discretizations can be characterized by a discrete inf-sup condition: There exists a constant α_0, independent of h, such that

$$\inf_{x \in X_h} \sup_{y \in X_h} \frac{\tilde{a}[x, y]}{\|x\|_{\mathscr{X}} \|y\|_{\mathscr{X}}} \geqslant \alpha_0 > 0. \tag{8.52}$$ eqn:dis-mixed

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

$$\tilde{\mathcal{D}}_h := \begin{bmatrix} (-\Delta_h)^{-1} & 0 \\ 0 & \mathcal{I}_h^{-1} \end{bmatrix}. \tag{8.53}$$ eqn:Stokes-pr

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 [136].

Above we have discussed how to construct canonical (natural) preconditioners based on the mapping property of the continuous Stokes equation. Now we 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.,

$$\tilde{A} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \quad \text{and} \quad \tilde{A} := \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}. \tag{8.54}$$ eqn:dis-saddl

We can apply the block preconditioners discussed in the previous subsection to solve this discrete problem; see the survey and numerical experiments by He and Vuik [112].

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. If we consider the above block diagonal method, the preconditioner can be written as

$$\tilde{D} := \begin{bmatrix} A^{-1} & 0 \\ 0 & M_p^{-1} \end{bmatrix}, \tag{8.55}$$ eqn:diag-prec

where M_p is the mass matrix corresponding to the pressure approximation space and, hence, it is well-conditioned; see Remark 3.24. It is easy to check that (8.55) is exactly the algebraic form of (8.53). Because both A and M_p are symmetric positive definite matrices, the preconditioner is well-defined. In fact, $\tilde{D}$ in (8.55) can be viewed as an approximation of $\mathrm{diag}(A^{-1}, S^{-1})$.

Similarly, we can use the block lower triangular matrix to construct a preconditioner

$$\tilde{T} := \begin{bmatrix} A & 0 \\ B & \hat{S} \end{bmatrix}^{-1}. \tag{8.56}$$ eqn:lowertrig

In particular, if we replace A by its diagonal part D in the LU decomposition of Remark 8.14, then we get the so-called SIMPLE preconditioner

$$\tilde{T}_{\mathrm{SIMPLE}} := \begin{bmatrix} I_u & D^{-1}B^T \\ 0 & -I_p \end{bmatrix}^{-1} \begin{bmatrix} A & 0 \\ B & BD^{-1}B^T \end{bmatrix}^{-1}. \tag{8.57}$$ eqn:simple-pr

The name of this block preconditioner comes from the widely-used SIMPLE method [159] in CFD. We now briefly review the iterative procedure of the SIMPLE method. Firstly, we can obtain an approximate velocity by solving

$$A\tilde{u} = f - B^T p^{\mathrm{old}}.$$

Apparently, this approximation did not consider the mass conservation equation. So we correct it by $u^{\mathrm{new}} = \tilde{u} + \delta u$ such that

$$A(\tilde{u} + \delta u) = f - B^T p^{\mathrm{new}}.$$

This means that the correction term satisfies

$$A\delta u = -B^T(p^{\mathrm{new}} - p^{\mathrm{old}}) =: -B^T\delta p.$$

By simply use the diagonal part to approximate A, we can obtain the relation $\delta u = -D^{-1}B^T\delta p$. This way, we are ready to derive a pressure equation from $B(\tilde{u} + \delta u) = 0$. It is easy to see that the corrections satisfy

$$BD^{-1}B^T\delta p = B\tilde{u} \tag{8.58}$$ eqn:SIMPLE1a

and

$$\delta u = -D^{-1}B^T\delta p. \tag{8.59}$$ eqn:SIMPLE1b

We notice that the last two correction terms coincide with the preconditioner (8.57).

The method can be also written in terms of the iteration variables directly. Namely,

$$BD^{-1}B^T p^{\text{new}} = BD^{-1}B^T p^{\text{old}} + B\tilde{u} = BD^{-1}(f + D\tilde{u} - A\tilde{u}) \tag{8.60}$$ eqn:SIMPLE2a

and

$$u^{\text{new}} = \tilde{u} + \delta u = \tilde{u} - D^{-1}B^T\delta p = D^{-1}(f + D\tilde{u} - A\tilde{u}) - D^{-1}B^T p^{\text{new}}. \tag{8.61}$$ eqn:SIMPLE2b

8.4.5 Solving 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 [136].

According to the classical theory of intersections and sums of Hilbert spaces [23], we can introduce the norms for $\mathscr{X}_1 \bigcap \mathscr{X}_2$ and $\mathscr{X}_1 + \mathscr{X}_2$ as

$$\|u\|_{\mathscr{X}_1 \bigcap \mathscr{X}_2} := \left(\|u\|^2_{\mathscr{X}_1} + \|u\|^2_{\mathscr{X}_2}\right)^{\frac{1}{2}}$$

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

$$\tilde{\mathcal{A}}_\epsilon := \begin{bmatrix} \mathcal{I} - \epsilon^2\Delta & -\operatorname{grad} \\ \operatorname{div} & 0 \end{bmatrix}$$

For this problem, we shall construct a preconditioner which is uniformly convergent with respect to both h and ϵ.

① In view of §8.3.3, we know that $\tilde{\mathcal{A}}_0$ is bounded from $H_0(\operatorname{div},\Omega) \times L_0^2(\Omega)$ into its dual space. Hence we consider the operator $\tilde{\mathcal{A}}_\epsilon$ on

$$\mathscr{X}_\epsilon := \Big(H_0(\operatorname{div},\Omega) \bigcap \epsilon[H_0^1(\Omega)]^d\Big) \times L_0^2(\Omega) \quad \text{and} \quad \mathscr{X}_\epsilon' := \Big(H_0(\operatorname{div},\Omega)' + \epsilon^{-1}[H^{-1}(\Omega)]^d\Big) \times L_0^2(\Omega).$$

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

$$\tilde{\mathcal{D}}_\epsilon^{(1)} = \begin{bmatrix} (\mathcal{I} - \operatorname{grad}\operatorname{div} - \epsilon^2\Delta)^{-1} & 0 \\ 0 & \mathcal{I} \end{bmatrix}.$$

② We have seen that $\tilde{\mathcal{A}}_0$ is also bounded on $[L^2(\Omega)]^d \times \big(H^1(\Omega) \bigcap L_0^2(\Omega)\big)$ into its dual space. Furthermore, in order to guarantee the inf-sup condition, the proper norm for the pressure unknown is [134, 135]:

$$\sup_{\mathbf{v}\in[H_0^1(\Omega)]^d} \frac{(q, \nabla\cdot\mathbf{v})}{\|\mathbf{v}\|_{L^2 \bigcap \epsilon H^1}} = \|\nabla q\|_{L^2 + \epsilon^{-1}H^{-1}} \sim \|q\|_{H^1 + \epsilon^{-1}L^2}.$$

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

$$\tilde{\mathcal{D}}_\epsilon^{(2)} = \begin{bmatrix} (\mathcal{I} - \epsilon^2\Delta)^{-1} & 0 \\ 0 & (-\Delta)^{-1} + \epsilon^2\mathcal{I} \end{bmatrix}.$$

Along this line, we can construct discrete block diagonal preconditioners for the time-dependent Stokes problem [87, 41]. 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:

$$\begin{cases} u_t - \Delta u &= f, \quad \Omega; \\ u &= 0, \quad \partial\Omega; \\ u|_{t=0} &= u_0, \quad \Omega. \end{cases} \tag{8.62}$$ eqn:heat2

We discretize the first equation in (8.62) 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

$$(\mathcal{I} - \epsilon^2\Delta)u_m = f'_m. \tag{8.63}$$ eqn:Reaction-

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. [136, 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 (5.23) in §5.3. In view of (5.22), on level l, we wish to have a smoother $\mathcal{S}_l$ behaves like

$$\big(\mathcal{S}_l v, v\big) = \frac{h_l^2}{h_l^2 + \epsilon^2}\,\big(v, v\big), \quad \forall\, v \in V_l.$$

This smoother then defines the corresponding BPX preconditioner for the semi-discrete problem (8.63). 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 Multigrid methods for the Stokes equation

Using a general multilevel iterative procedure, we can construct coupled geometric multigrid methods for the saddle-point problem (8.54) 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 §6.3. 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 [125]. 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.

8.5.1 Braess–Sarazin smoother

The Braess–Sarazin smoother was introduced in [39] and can be written as

$$\begin{bmatrix} u^{(m+1)} \\ p^{(m+1)} \end{bmatrix} = \begin{bmatrix} u^{(m)} \\ p^{(m)} \end{bmatrix} + \begin{bmatrix} \omega D & B^T \\ B & 0 \end{bmatrix}^{-1} \left(\begin{bmatrix} f \\ 0 \end{bmatrix} - \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u^{(m)} \\ p^{(m)} \end{bmatrix} \right), \tag{8.64}$$

eqn:Braess

where ω is a positive parameter. This method mimics the weighted Jacobi smoother for the Poisson's equation.

We need to solve, in each smoothing step, the following the linear system

$$\begin{bmatrix} \omega D & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \delta u^{(m)} \\ \delta p^{(m)} \end{bmatrix} = \begin{bmatrix} f - Au^{(m)} - B^T p^{(m)} \\ -Bu^{(m)} \end{bmatrix}.$$

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

$$\left(BD^{-1}B^T\right)\delta p^{(m)} = \omega B u^{(m)} + BD^{-1}\left(f - Au^{(m)} - B^T p^{(m)}\right).$$

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.

8.5.2 Vanka smoother

Next we introduce a smoother originally proposed by Vanka [187]. 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.,

$$I_i = \begin{bmatrix} I_{u,i} & 0 \\ 0 & I_{p,i} \end{bmatrix} \in \mathbb{R}^{(|\mathsf{S}_i|+1)\times(n+m)},$$

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):

$$I - \tilde{T}_{\mathrm{FVanka}}\tilde{A} = \prod_{i=1}^{m}\left(I - I_i^T\tilde{A}_i^{-1}I_i\tilde{A}\right), \tag{8.65}$$ eqn:FVanka

where

$$\tilde{A}_i = I_i\tilde{A}I_i^T = \begin{bmatrix} A_i & B_i^T \\ B_i & 0 \end{bmatrix} \in \mathbb{R}^{(|\mathsf{S}_i|+1)\times(|\mathsf{S}_i|+1)}.$$

We can also use a simplified version (i.e., the Diagonal Vanka smoother):

$$I - \tilde{T}_{\mathrm{DVanka}}\tilde{A} = \prod_{i=1}^{m}\left(I - I_i^T\tilde{D}_i^{-1}I_i\tilde{A}\right), \tag{8.66}$$ eqn:DVanka

where

$$\tilde{D}_i = \begin{bmatrix} D_i & B_i^T \\ B_i & 0 \end{bmatrix} \in \mathbb{R}^{(|\mathsf{S}_i|+1)\times(|\mathsf{S}_i|+1)}.$$

In this case, due to the special nonzero pattern of $\tilde{D}_i$, it can be solved very efficiently.

8.6 Homework problems

hw:div-eps **HW 8.1.** Show the equation (8.16). Hint: In $\mathbb{R}^2$, taking divergence of the symmetric gradient, we get

$$\begin{aligned}\nabla\cdot\boldsymbol{\varepsilon}(\mathbf{u}) &= \begin{bmatrix} \partial_1^2 u_1 + \frac{1}{2}\partial_2(\partial_2 u_1 + \partial_1 u_2) \\ \partial_2^2 u_2 + \frac{1}{2}\partial_1(\partial_1 u_2 + \partial_2 u_1) \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{2}(\partial_1^2 u_1 + \partial_2^2 u_1) + \frac{1}{2}\partial_1(\partial_1 u_1 + \partial_2 u_2) \\ \frac{1}{2}(\partial_1^2 u_2 + \partial_2^2 u_2) + \frac{1}{2}\partial_2(\partial_1 u_1 + \partial_2 u_2) \end{bmatrix} = \frac{1}{2}\Delta\mathbf{u} + \frac{1}{2}\nabla\nabla\cdot\mathbf{u}.\end{aligned}$$

w:Stokes_weak **HW 8.2.** Derive the weak form (8.26) of the Stokes equations (8.25).

hw:AL **HW 8.3.** Give the complete proof of Theorem 8.17.

Chapter 9

Optimization Problems

ch:optim

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

sec:vi

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 [120]. For a comprehensive discussion on numerical methods for VIs, we refer to Glowinski [101].

9.1.1 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

$$a[u, v-u] \geqslant f(v-u), \quad \forall\, v \in \mathcal{K}_\chi. \tag{9.1}$$

eqn:evi1

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

$$a[w, v-w] \geqslant f_0(v-w) := f(v-w) - a[\chi, v-w], \quad \forall\, v \in \mathcal{K}_0. \tag{9.2}$$

eqn:evi2

For problem (9.1), the *Lagrange multiplier* can be defined as σ_1 such that

$$\langle \sigma_1(u), \varphi \rangle := f(\varphi) - a[u, \varphi], \quad \forall\ \varphi \in H_0^1(\Omega). \tag{9.3}$$ eqn:sigma

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

$$\langle \sigma_1(u), v - u \rangle \leqslant 0, \quad \forall\ v \in \mathcal{K}_\chi, \tag{9.4}$$ eqn:nonpos

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,

$$\langle \sigma(v) - \sigma(u), v - u \rangle = -a[v - u, v - u] = -\left\| v - u \right\|^2, \quad \forall\ v, u \in H_0^1(\Omega). \tag{9.5}$$ eqn:mono

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

$$\begin{cases} \mathcal{A}u - f \geqslant 0 \\ u - \chi \geqslant 0 \\ \langle \mathcal{A}u - f, u - \chi \rangle = 0. \end{cases} \tag{9.6}$$ eqn:lcp

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 $v = \chi$. □

9.1.2 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:

$$a[u_h, v_h - u_h] \geqslant f_0(v_h - u_h), \quad \forall\ v_h \in \mathcal{K}_0 \bigcap V_h. \tag{9.7}$$ eqn:fem2

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

$$(\underline{v} - \underline{u})^T(A\underline{u} - \vec{f_0}) \geqslant 0, \quad \forall\ \underline{v} \geqslant 0, \tag{9.8}$$ eqn:dis2

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 [55]) 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

$$(\vec{f_0} - A\underline{v})_{+,i} = \begin{cases} (\vec{f_0} - A\underline{v})_i & \text{if } \underline{v}_i > 0 \\ \min\{(\vec{f_0} - A\underline{v})_i, 0\} & \text{if } \underline{v}_i = 0. \end{cases}$$

9.1.3 Error and residual

ssc: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

$$\mathcal{F}(v) - \mathcal{F}(u) = \frac{1}{2}\left|\!\left|\!\left| v - u \right|\!\right|\!\right|^2 - \langle \sigma, v - u\rangle, \quad \forall\ v \in \mathcal{K}_\chi. \tag{9.9}$$ eqn:Idiff

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

$$\begin{aligned} \mathcal{F}(u_h) - \mathcal{F}(u) &= \frac{1}{2}\left|\!\left|\!\left| u_h - u \right|\!\right|\!\right|^2 - \langle \sigma, u_h - u\rangle \\ \mathcal{F}(w_h) - \mathcal{F}(w) &= \frac{1}{2}\left|\!\left|\!\left| w_h - w \right|\!\right|\!\right|^2 - \langle \sigma, w_h - w\rangle. \end{aligned} \tag{9.10}$$ eqn:diff

It is easy to see that the variational inequality (9.2) can be written as the following quadratic minimization problem:

$$\min_{w\in\mathcal{K}_0} \frac{1}{2}a[w,w] - f_0(w). \tag{9.11}$$ eqn:min2

For finite element approximation, we compute the finite dimensional minimization problem

$$\min_{w_h\in V_h\bigcap\mathcal{K}_0} \frac{1}{2}a[w_h,w_h] - f_0(w_h). \tag{9.12}$$ eqn:femmin2

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.,

$$\min_{\hat{w}_h+e_h\in V_h\bigcap\mathcal{K}_0} \frac{1}{2}a[e_h,e_h] - \langle\sigma(\hat{w}_h),e_h\rangle\,. \tag{9.13}$$ eqn:ErrVI

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

$$u = \operatorname*{argmin}_{v\in\mathscr{V}} \mathcal{F}(v). \tag{9.14}$$ eqn:min

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*

$$\mathcal{G}(u) := \mathcal{F}'(u) = 0, \tag{9.15}$$ eqn:1st-cond

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 $\mathcal{A} = \mathcal{G}'$.

9.2.1 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

$$u = \operatorname*{argmin}_{v \in \mathbb{R}^N} \mathcal{F}(v) \tag{9.16}$$ eqn:dmin

or

$$\mathcal{G}(u) = 0. \tag{9.17}$$ eqn:d1st-cond

The idea of coarse-grid correction used in Algorithm **??** 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).

9.2.2 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

$$u^{(k+1)} = u^{(k)} - \big[\nabla^2 \mathcal{F}(u^{(k)})\big]^{-1} \nabla \mathcal{F}(u^{(k)}). \tag{9.18}$$ eqn:Newton

In the above iteration step, we need to solve a linear system, the Jacobian equation:

$$\mathcal{A}e^{(k)} := \big[\nabla^2 \mathcal{F}(u^{(k)})\big]e^{(k)} = -\nabla \mathcal{F}(u^{(k)}) =: r^{(k)}. \tag{9.19}$$ eqn:Jeqn

We can employ the methods discussed in the previous chapters to solve such equations.

Listing 9.1: Newton–Raphson method

```
1 Given an initial guess u ∈ 𝒱 and set r ← −∇ℱ(u);
2 while ‖r‖ > ε
3     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.

9.2.3 Full approximation scheme

For the nonlinear equation (9.15), the residual corresponding to an approximate solution v can be defined as

$$r := -\mathcal{G}(v) = \mathcal{G}(u) - \mathcal{G}(v) \tag{9.20}$$ eqn:nonlinear

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

$$\mathcal{G}_c\big(u_c^{(1)}\big) - \mathcal{G}_c\big(\mathcal{I}_c^T u^{(1)}\big) = r_c = \mathcal{I}_c^T r = -\mathcal{I}_c^T \mathcal{G}\big(u^{(1)}\big). \tag{9.21}$$ eqn:nonlinear

This means, on the coarse level, a problem similar to the original problem (with different right-hand side) should be solved

$$\mathcal{G}_c\big(u_c^{(1)}\big) = \mathcal{G}_c\big(\mathcal{I}_c^T u^{(1)}\big) - \mathcal{I}_c^T \mathcal{G}\big(u^{(1)}\big). \tag{9.22}$$ eqn:nonlinear

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

$$u^{(2)} = u^{(1)} + \mathcal{I}_c\big(u_c^{(1)} - \mathcal{I}_c^T u^{(1)}\big). \tag{9.23}$$ eqn:nonlinear

Apparently the above idea can be applied recursively as we discussed in §6.3. 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);
```

9.2.4 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 [183].

9.3 Constrained minimization

In this section, we consider multilevel solvers for constrained minimization problems

$$u = \operatorname*{argmin}_{v\in\mathcal{K}_0} \mathcal{F}(v) := \frac{1}{2}a[v,v] - f(v), \tag{9.24}$$ eqn:EVI2

which is equivalent to the variational inequality (9.2).

9.3.1 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 [55].

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

$$u^{(2)} = u^{(1)} + \mathcal{I}_c\big(u_c^{(1)} - \mathcal{I}_c^T u^{(1)}\big).$$

9.3.2 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

$$u = \underset{v\geqslant 0,\, v\in\mathbb{R}^N}{\operatorname{argmin}} \mathcal{F}(v) := \frac{1}{2} v^T A v - f^T v. \tag{9.25}$$ eqn:disEVI2

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

$$(A - U^{-1}\Sigma)\delta u = \mu U^{-1}\mathbf{1} + f - Au. \tag{9.26}$$ eqn:IPmethod

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 [15] for details.

9.3.3 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

$$(Q_l v_h, \phi_l) = (v_h, \phi_l), \quad \phi_l \in V_h^l, \tag{9.27}$$ eqn:L2proj

and the energy projections $\Pi_l : V_h^j \to V_h^l$ by

$$a[\Pi_l v_h, \phi_l] = a[v_h, \phi_l], \quad \phi_l \in V_h^l. \tag{9.28}$$ eqn:proj

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

$$\min_{e_{j-1}^{(0)} \in V_h^{j-1},\ w_j^{(0)} + e_{j-1}^{(0)} \bigcap \mathcal{K}_0} \frac{1}{2} a[e_{j-1}^{(0)}, e_{j-1}^{(0)}] - \langle \sigma(w_j^{(0)}), e_{j-1}^{(0)} \rangle. \tag{9.29}$$ eqn:corrction

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):

$$\min_{d_{j-1}^{(0)} \in V_h^{j-1} \bigcap \mathcal{K}_0} \frac{1}{2} a[d_{j-1}^{(0)}, d_{j-1}^{(0)}] - \langle \sigma(w_j^{(0)}), d_{j-1}^{(0)} \rangle. \tag{9.30}$$ eqn:newcorrct

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 [123]. 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

$$\min_{w_j^{(0)}+\alpha\phi_1 \in V_h^j \bigcap \mathcal{K}_0} \frac{1}{2} a[w_j^{(0)} + \alpha\phi_1, w_j^{(0)} + \alpha\phi_1] - f_0(w_j^{(0)} + \alpha\phi_1). \tag{9.31}$$ eqn:ssc

Similar to the discussion in the previous section, we need to solve a discrete problem

$$\min_{w_j^{(0)}+\alpha\phi_1 \in V_h^j \bigcap \mathcal{K}_0} \frac{1}{2} a[\phi_1, \phi_1]\alpha^2 - \langle \sigma(w_j^{(0)}), \phi_1 \rangle \alpha. \tag{9.32}$$ eqn:ssc2

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 $w_j^{(i-1)} + \alpha\phi_i \in V_h^j \bigcap \mathcal{K}_0$.

We refer to the paper by Tai [182] for details.

Chapter 10

Robustness and Adaptivity

ch:robust

The efficient and robust solution of linear algebraic systems is one of the main bottlenecks in large-scale numerical simulation. In this report, we review some old and new techniques for improving the robustness of iterative solvers for large-scale sparse linear equations. In particular, we will focus on methods based on machine learning to automatically select solver components in order to get better overall simulation performance. Deep learning techniques, which have gained popularity in many application areas of machine learning, can also be used to enhance this automatic selection procedure.

10.1 Robustness of linear solvers

sec:intro

Due to the fact that, for many applications, a significant portion of simulation time for transient problems is spent in linear solvers, a lot of efforts have been directed to the research on solution methods for linear systems, which result in plenty of solution algorithms and software packages [4]. Oftentimes practitioners without "proper" training might find themselves in a very difficult position to choose a good solver or its parameters from excessive number of options. More frustratingly, for a complex physical problem, there might not exist a universally best solver for all linear systems over the course of simulation. Actually, performance of linear solvers are largely affected by stage of evolution of physics, characteristics of discretization methods, requirement of accuracy, closeness to solution, limitation in computing resources, and so on.

10.1.1 Why robustness is important

Simulation-based scientific discovery and engineering design have been the main driving force for developing high-performance computers and algorithms. As we entered the multi-

petaflop[1] era, frequency of a single CPU core does not increase beyond certain critical value. On the other hand, the number of computing cores in supercomputers is growing exponentially, which results in higher and higher system complexity [3]. More processing cores competing for various levels of memory resources widens the speed gap between computations and memory accesses. It has become increasingly important for algorithms to be well-suited to the emerging parallel hardware architectures of extreme scale. This trend also affects how one would choose a solver for a particular problem. In this sense, complexity and different choices of hardware architectures pose more difficulties in the choosing solutions methods for domain scientists. Co-design of hardware, software, algorithm, and application is crucial to the success of scientific or engineering computing to achieve exascale[2] performance [142, 5, 77].

Direct methods are popular in practice due to their robustness for a large class of problems. Some specialized direct solvers, like fast Poisson solvers based on FFT, are very efficient and readily useable at different hardware and software platforms (see [98, 137] for example). But these methods can only be applied for specific equations or special discretizations, which restricts their applications in complicated engineering problems. Some direct methods, like general-purpose sparse direct solvers [173], can be employed as a black-box solver and be plugged into user-domain simulation programs easily. The sparse direct solvers are robust and effective for a large class of discrete problems. More importantly, they require little human intervention and are especially efficient for relatively small problems with around a million unknowns. Larger problems, on the other hand, tend to require a large amount of memory as well as computational time.

If applied successfully, direct methods can provide solution to nonsingular systems as accurate as floating accuracy and condition number allow; see [72, 73] for example. However, this also means that direct methods might give non-necessarily "high accuracy" for some applications. In fact, accuracy is rarely the only property we ask for linear algebraic solvers. Efficiency, scalability, cost-effectiveness, robustness, and reliability are important properties and we need to balance between them in practice. There is no universal criteria for choosing solution methods and it all depends on what we want to achieve and what cost we are willing to pay. For example, in order to accelerate simulation when solving Jacobian systems arsing from Newton linearization, we might use different types of iterative methods to different levels of accuracy in different stages of nonlinear iteration [24]. Direct methods usually fail to provide enough flexibility for users to tune and iterative methods are usually employed in such situations.

Ever increasing practical demand to solve very large linear systems and requirements on applicability suggest considering iterative methods as an alternative. Iterative methods can be used

[1] 10^{15} floating-point operations per second.
[2] 10^{18} floating-point operations per second.

as general-purpose or specialized solvers due to their cost-effectiveness and memory-efficiency; see, for example, [122]. The most successful example of iterative methods is the so-called Krylov subspace method (KSM), like the conjugate gradient (CG) method and the generalized minimal residual (GMRES) method [172]. Since properties of linear systems can be very different in a single simulation run, robustness of solution methods is an important, if not the biggest, concern. However, the biggest weakness of iterative methods is arguably robustness, which refers to the ability of iterative methods to resist perturbations or changes to the underlying physical equation. Hence, to design a solver package based on iterative methods as a replacement of the state-of-the-art direct solvers, the issue of robustness has to be resolved [176]. Indeed, a robust general iterative solver is still hard to obtain, especially in industrial applications, and might be the biggest dream for many engineers.

10.1.2 Robustness of linear solvers

Generally speaking, the robustness of a system can be viewed as the property of being strong and healthy in the constitution. Most iterative solution methods can converge in a timely manner for a variety of simple model problems, but slow down considerably or even fail to converge when applied to more complex cases. For solution methods of linear algebraic systems, the definition of robustness is twofold:

- First of all, the method should be breakdown-free and provide a reliable solution. This basically says that we would like the simulation to run without unexpected interruptions caused by the linear solver.
- Secondly, the robustness refers to the ability of solution methods to handle most problems arising from simulations in a highly efficient manner. That is to say, the performance of the solution method should be resistant to perturbations of physical as well as discretization parameters and it gives reasonably accurate solutions in reasonable turn-back time.

def:robust **Definition 10.1** (Robustness of a linear solver)**.** A linear solver $\mathcal{S}$ is robust for a class of problems $\mathcal{P}$ if and only if the following property

$$\max_{P_\alpha \in \mathcal{P}} \left\| \mathcal{S}(P_\alpha) \right\| \lesssim \varepsilon,$$

where $\|\cdot\|$ is a performance measure and ε is acceptance tolerance.

The performance measure and tolerance are usually problem-dependent and can be determined by the end users. Unfortunately, oftentimes we could not find a single solver good for all different problem parameters. Hence we can weaken the condition and call a class of solvers $\mathcal{S}$

is robust if

$$\operatorname*{argmin}_{S_\beta \in \mathcal{S}} \max_{P_\alpha \in \mathcal{P}} \left\| S_\beta(P_\alpha) \right\| \lesssim \varepsilon.$$

10.2 Robustness of Iterative Solvers

sec:pre

Consider the large-scale sparse system of linear algebraic equations arising from a partial differential equation. As we pointed out earlier, performance of solution methods could be affected by properties of the physical problem as well as its discretization methods, let alone parameters of the solution methods. Usually, these properties can be characterized by simple parameters, like temperature, diffusion coefficient, spatial mesh size, etc. There are also cases when we can not or would not characterize those properties as parameters for cost concerns, like heterogenous coefficients, unstructured grids, etc. These properties will eventually enter the coefficient matrices A and usually handled in a purely algebraic manner. There are several strategies to improve robustness of iterative solvers and we can categorise them into three types described in this section.

10.2.1 Constructing preconditioners not sensitive to parameters

Preconditioners based on incomplete factorization of A, like the Incomplete LU (ILU) methods, are undoubtedly among the most popular methods in engineering [172]. ILU methods are purely algebraic and are widely applied as subproblem solvers in the domain decomposition (DD) methods for parallel computing. A related type of methods is the so-called Approximate Inverse (AINV) preconditioners [21, 18] which is based the approximated factorizations of A^{-1}. Incomplete factorizations can fail for a general SPD matrix due to the so-called pivot breakdown and could be improved by shifting or modification of A. A more robust remedy for poorly conditioned linear systems is the breakdown-free versions of ILU [33, 22, 34, 162]. Sometimes, ILU methods might yield a relatively high complexity in order to obtain good convergence behavior, especially in 3D.

Algebraic multigrid (AMG) is another type of popular preconditioning technique [56, 57, 170] and it is, in some sense, more robust compared with GMG methods [169]. Problems with anisotropic coefficients on regular meshes, or problems with isotropic coefficients on anisotropic meshes, will cause troubles for geometric multigrid methods. While GMG essentially relies on the availability of robust smoothers, AMG takes a different approach by focusing on constructing suitable coarse space. Following the seminar work by Brandt et al. [56, 57, 52] on the convergence analysis applicable to AMG methods, there have been a lot of discussions on the AMG theory; see [170, 61, 178, 90, 91, 189] for example. The readers are referred to the recent survey papers on theoretical development [129, 155, 202] as well as applications and parallelization [206] of AMG

methods. For the development on non-symmetric problems, we refer to [131, 153, 133, 132]; for analysis based on aggregation-type AMG algorithms, we refer to [186, 188, 62, 143, 147, 60].

10.2.2 Combining iteration, precondition, and decoupling strategies

It is known that the ILU methods can be employed as smoothers to improve robustness of the multigrid methods [119, 191, 194]. A simple and transparent framework for combining preconditioners like ILU or additive Schwarz preconditioners with AMG or any another norm-convergent iterative method has been proposed by Hu et al. [115] for SPD problems. In such a combined preconditioner, the component provided by the norm-convergent iterative method need not be very effective when used alone. Such a framework has been applied to solve the systems arising from the porous media flow problem [115] and extended to nonsymmetric problems in the radiation diffusion problem [216, 217]. Another related strategy by combining different methods to construct a more powerful solver for nonsymmetric problems can be found in [74].

For systems of partial differential equations, besides iterative solvers and preconditioners and their combinations, it is sometimes beneficiary to apply extra decoupling steps to weaken the strength of coupling between different physical variables. By combining decoupling methods with appropriate solution methods, we can improve solver efficiency and robustness for many complex problems arising from discretization of coupled (nonlinear) PDE systems; for example, for semiconductor device simulation [11] and for compositional model in reservoir simulation [124, 161]. In particular, Qiao et al. [161] discussed the conditions when a preconditioner is suitable for a particular decoupling strategy.

For transient problems, it is very often that no single iterative solver, preconditioner, or decoupling is able to work very well for all linear systems arising during simulation. In order to minimize solver failure possibility and maximize robustness, a natural idea is to combine various solvers or preconditioners during simulation. One possible approach is the so-called poly-iterative method, which applies various solvers or preconditioners with similar structure simultaneously [17, 105]. An alternative approach is the composite method, which relies on a composition of multiple iterative solvers to improve reliability [29, 26, 28, 27]. Although these multi-method solvers are viable approaches to improve robustness in practice, especially for large-scale problems, they tend to have a non-negligible overhead (i.e., extra computational work during application); see a summary recently given by Sood [177] for more details.

10.2.3 Empolying an automated solver-selection procedure

Besides the methods mentioned above, an adaptive or automatic solver selection procedure can be constructed to assist users to choose free parameters (see Figure 10.1) based on the

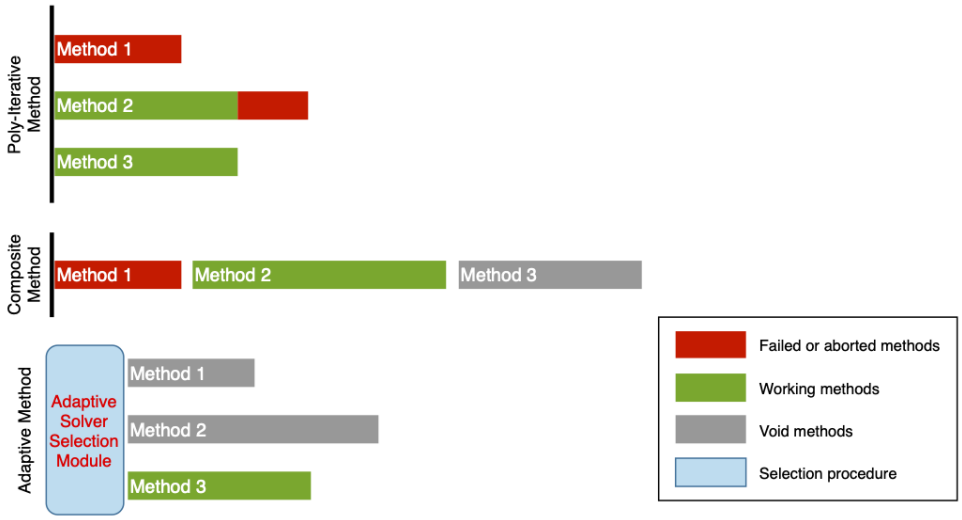


fig:combined

Figure 10.1: Different ways of combined multiple solvers to improve robustness

given feature parameters in order to get robust performance; see, for example, [163, 140, 207, 208]. The general algorithm selection problem in an abstract setting has been proposed by Rice [164]. Since then this problem has discussed by many research groups and a software package (ASLib) for benchmarking algorithm selection methods has been recently developed [31]. A software framework (SALSA) has been suggested for self-adapting linear algebra and linear solution algorithms [79, 80, 75, 78, 85]. One may use analytical or empirical information to design adaptive strategies for selecting solvers or their parameters for different problems; see Figure 10.2.

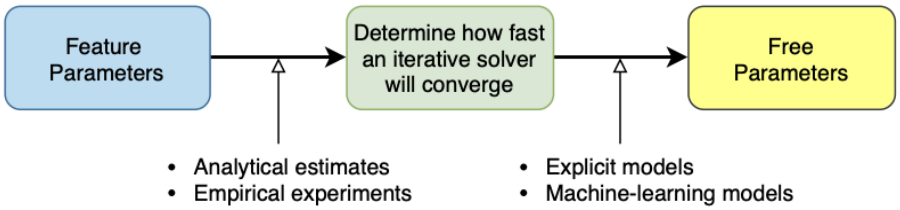


fig:adaptive

Figure 10.2: Typical procedure for constructing adaptive iterative solvers

This approach has been shown to be effective in many fields. In our experience, there are several key components to make this approach efficient for a particular problem:

1. choosing a general enough solution procedure which is efficient or even optimal for simple cases and can be adjusted for more difficult cases;

2. providing a small set of feature parameters which affect solver performance the most;

3. constructing a performance model based on analytical convergence factor estimates or empirical experiments which can predict how efficient the solver might be;

4. having an efficient procedure to train a performance model based on actually simulation runs, in case an analytical performance model is hard or not possible to be obtained.

10.3 Robustness of ILU Preconditioners

sec:robustilu

It is well-known that direct solvers have better robustness properties and can be usually applied as a blackbox solver. In this section, we briefly discuss the robustness of LU and ILU factorizations. They are not only important general-purpose solvers, but also can be combined with multilevel iterative methods in various ways.

10.3.1 LU factorization

LU factorization and Gaussian elimination (GE) are well-known methods for solving linear systems, especially for problems with general dense coefficient matrices. For simplicity, we do not consider numerical stability and pivoting here. For a nonsingular matrix $A \in \mathbb{R}^{N\times N}$, we need find a lower triangular matrix $L \in \mathbb{R}^{N\times N}$ and an upper triangular matrix $U \in \mathbb{R}^{N\times N}$ such that

$$A = LU.$$

Since we can require the main diagonal of L to an all-one vector, the two factors L and U can be saved in a compact way in the original matrix A. So we give the GE algorithm implemented as an in-place algorithm (the i-th row of A is overwritten by the i-th row of L and U).

Listing 10.1: Gaussian elimination – KIJ variant

```
for k = 1, 2, ..., N − 1
    for i = k + 1, ..., N
        a_ik ← a_ik/a_kk;
        for j = k + 1, ..., N
            a_ij ← a_ij − a_ik a_kj;
        end
    end
end
```

The following theorem first proved by Fan [92] is important to analyze the procedure of Gaussian elimination. This theorem makes sure the procedure can always continue and will not break down. This is an important robustness property for linear solvers.

thm:fan **Theorem 10.2** (Fan Theorem)**.** Let A be an M-matrix and A_1 be the matrix obtained from the first step of Gaussian elimination. Then A_1 is also an M-matrix. The $(N-1)\times(N-1)$ sub-matrix of A_1 by removing its first row and first column is also an M-matrix.

Although the above algorithm (the KIJ variant) is usually used to explain the Gaussian elimination algorithm, it is rarely employed in practice due to performance consideration, especially for sparse problems stored in certain data structures like CSR discussed in §6.5. Next we give a more useful (or efficient) implementation of the same algorithm, which is referred to as the the Gaussian – IKJ variant.

Listing 10.2: Gaussian elimination – IKJ variant

```
for i = 2, ..., N
    for k = 1, ..., i − 1
        a_ik ← a_ik/a_kk;
        for j = k + 1, ..., N
            a_ij ← a_ij − a_ik a_kj;
        end
    end
end
```

10.3.2 Incomplete LU factorization

Incomplete LU factorization can be viewed as an inaccurate LU factorization and provides an approximation to LU. In general, we need find a lower triangular matrix $L \in \mathbb{R}^{N\times N}$ and an upper triangular matrix $U \in \mathbb{R}^{N\times N}$ such that

$$R = LU - A, \quad \tilde{A} := A + R,$$

in which $R \in \mathbb{R}^{N\times N}$ has certain static or dynamic zero pattern. For simplicity, we only discuss methods with static zero pattern. Equivalently, we can specify, for the approximation matrix $\tilde{A}$, a fixed zero pattern

$$\mathcal{Z} := \big\{(i,j)\,|\,i \neq j, 1 \leqslant i,j \leqslant N\big\}.$$

Note that, with this definition, we require that the diagonal entries of $\tilde{A}$ to be nonzero. Whence the zero or nonzero pattern of the factorization is given, we can modify the LU methods to the corresponding ILU (with static zero pattern) methods. For example, ILU(0) is a method with $\mathcal{Z}$ which has the same zero pattern as the coefficient matrix A.

Listing 10.3: ILU with static zero pattern – KIJ variant

```
%% Given a zero pattern Z
for (i, j) ∈ Z, a_ij ← 0;
for k = 1, 2, ..., N − 1
```

```
    for i = k+1, ..., N  &&  (i,j) ∉ Z
        a_ik ← a_ik / a_kk;
        for j = k+1, ..., N  &&  (i,j) ∉ Z
            a_ij ← a_ij − a_ik a_kj;
        end
    end
end
```

Similar to the LU methods discussed previously, we can also construct an IKJ variant of the above ILU method.

Listing 10.4: ILU with static zero pattern – IKJ variant

```
%% Given a zero pattern Z
for (i,j) ∈ Z,  a_ij ← 0;
for i = 2, ..., N
    for k = 1, ..., i−1  &&  (i,j) ∉ Z
        a_ik ← a_ik / a_kk;
        for j = k+1, ..., N  &&  (i,j) ∉ Z
            a_ij ← a_ij − a_ik a_kj;
        end
    end
end
```

10.3.3 Robustness of ILU factorization

Definition 10.3 (Regular splitting)**.** Let A, M, N be three given matrices satisfying $A = M - N$. The pair of matrices $(M,\ N)$ is a regular splitting of A, if M is nonsingular and M^{-1} and N are nonnegative.

We now consider the factorization method in Algorithm 10.3. Here, we use the subscript k to denote the k-step of factorization. So $A_1 \in \mathbb{R}^{N\times N}$ is the matrix after first step of the Gauss elimination. Then

$$\tilde{A}_1 = A_1 + R_1,$$

where $\tilde{A}_1 \in \mathbb{R}^{N\times N}$ is the result of the first step of ILU. According to the definition of $\mathcal{Z}$, the dropped entries are nonpositive and R_1 is nonnegative.

By Theorem 10.2, we find that A_1 is an M-matrix. It can be proved that $\tilde{A}_1$ is also an M-matrix; see HW 10.2. In this sense, the algorithm will not break down and it can further produce

$$A_k = L_k \tilde{A}_{k-1}, \quad L_k := I - \frac{1}{a_{kk}^{(k)}} \begin{bmatrix} 0_k \\ A_{k+1:N,k} \end{bmatrix} e_k^T.$$

Hence, at the k-th step, we obtain that

$$\tilde{A}_k = A_k + R_k = L_k \tilde{A}_{k-1} + R_k.$$

Applying the above relation recursively, we get

$$\tilde{A}_{N-1} = \big(L_{N-1}L_{N-2}\cdots L_1\big)A + \big(L_{N-1}L_{N-2}\cdots L_2 R_1 + \cdots + L_{N-1}R_{N-2} + R_{N-1}\big).$$

Define

$$\begin{aligned} U &:= \tilde{A}_{N-1}, \\ L &:= \big(L_{N-1}L_{N-2}\cdots L_1\big)^{-1}, \\ S &:= L_{N-1}L_{N-2}\cdots L_2 R_1 + \cdots + L_{N-1}R_{N-2} + R_{N-1}. \end{aligned}$$

Notice that, at the k-th step, entries dropped only appear in the $(N-k)\times(N-k)$ lower sub-matrix of A_k. So the first k rows and columns of R_k are zero. As a result, we have

$$L_{N-1}L_{N-2}\cdots L_{k+1}R_k = L_{N-1}L_{N-2}\cdots L_1 R_k.$$

Then it is easy to see that

$$S = L_{N-1}L_{N-2}\cdots L_1(R_1 + R_2 + \cdots + R_N) = L^{-1}R,$$

where $R := R_1 + R_2 + \cdots + R_N$. This gives $LU = A + R$ and the result can be summarized in the following theorem.

thm:ilu **Theorem 10.4** (Robustness of ILU)**.** Let A be an M-matrix and Z be a given zero pattern. Then Algorithm 10.3 does not break down and produces an incomplete factorization

$$A = LU - R,$$

which is a regular splitting of A.

10.4 Workflow for Selecting Solvers

sec:select

Along with the development of machine learning and deep learning theories, data-driven algorithms have been used in a variety of ways for automatically selecting solvers and their parameters. Most traditional ML-based methods train a supervised classifier to predict appropriate parameters for unknown linear systems [113, 24, 86].

10.4.1 Automatic classifiers for linear solvers

The key point of these methods is very similar. Firstly, one has to create a database and split it into a training set and a test set (sometimes, a validation set as well). Every entry in the database consists of

- input parameters:
  - feature parameters, e.g., symmetry of the coefficient matrix, size of the coefficient matrix;
  - free parameters, e.g., type of preconditioner, number of computing nodes.
- output labels, e.g., whether the iterative method converges, how many iteration needed.

Secondly, using the training set to train the classifier. Finally, using the test set to verify the effectiveness of the trained classifier.

In [113], neural networks with a single hidden layer were used as a classifier to divide 260 matrixes which is randomly selected from Florida Sparse Matrix Collection, combining with 72 iterative methods, into two categories (convergent and non-convergent). In [24], alternation decision trees were applied to do the classification, but the output label is not a simple criteria to reflect convergence but compared with a specific baseline solver. More specifically, only if the iterative method is at least ρ time faster than the baseline method (the GMRES method with block ILU preconditioner), it will be labeled as 1, otherwise -1. In [86], the authors suggested that the multi-label classifiers outperform single-label classifiers in almost every simulation. And in the lighthouse project [141], a variety of classification methods like LibSVM, BayesNet, KNN, and so on were compared in terms of accuracy.

A common conclusion that can be drawn from the previous studies is that the number of input parameters is not proportional to the classification effectiveness. This suggests that reducing the number of parameters can decrease the overhead of computing attributes of matrix without significantly influencing the accuracy of classification. Bhowmick et al. [30] studied feature set reduction and ordering; and they demonstrated that the training time could be reduced by a factor of 125 on average.

10.4.2 General methodology

- The workflow contains two steps: the offline step trains a model for selecting "optimal" free parameters with human experts; the online step, on the other hand, select parameters automatically based on the trained model.

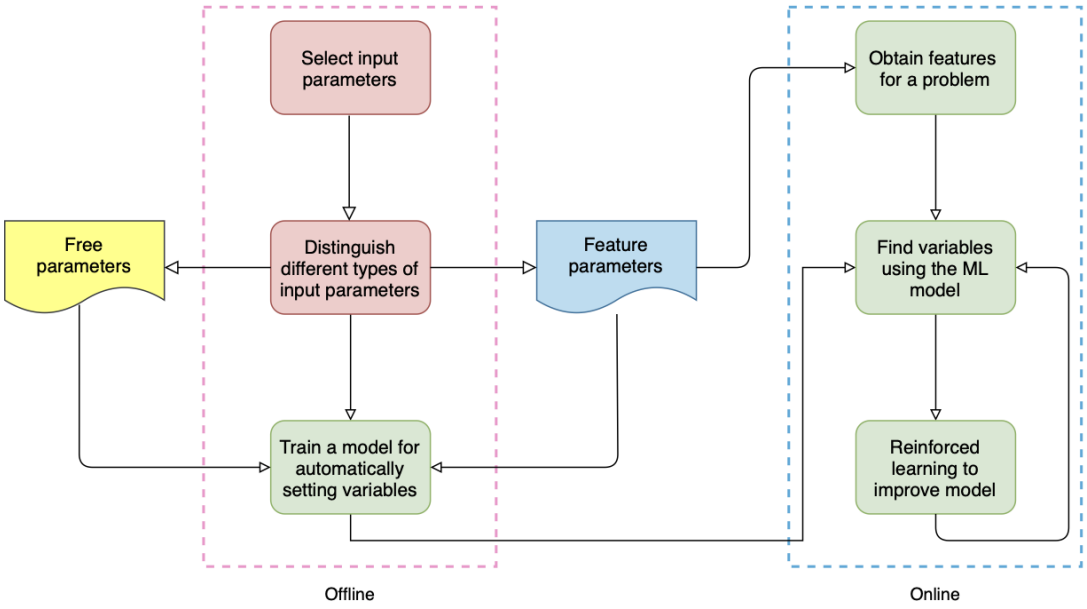


fig:workflow

Figure 10.3: Workflow for selecting solver parameters

- The solver and application experts should work together to choose an efficient solution method which provides enough generality for the simulation problem at hand. Experts should also choose a reasonable set of free parameters in order to improve performance of training.
- We focus more on reinforced learning and transfer learning to obtain a better model and prevent over-fitting.
- Which training algorithm to use is probably of secondary importance as the training is done offline.

I. Offline step

1. For a given simulation problem, we need to determine what input parameters (including both feature and free parameters) should be considered based on the given objective; This step is mainly by human experts.
2. We then select feature and free parameters to obtain an input set with small number of most important solver parameters; This step can be done by experts with help from machines.
3. Choose a machine learning model based on the size of input set as well as the amount of training data we have at hand; This step can be done by experts with help from machines.

4. Train the model; This step is done by machines.

Ⅱ. Online step

1. For a specific linear systems during a simulation run, we extract features requested by Step 2 of the office step.

2. Obtain free parameters based on the trained model.

3. Collect the result data to improve offline model later on.

10.5 Robustness of Multilevel Iterative Methods

sec:robustmg

A lot of effort has been devoted to improve robustness of iterative methods. One particularly powerful technique is combining various Krylov subspace methods with proper preconditioners. When combined with suitable preconditioners, Krylov subspace methods are efficient for linear systems arising from partial differential equations. For example, geometric multigrid (GMG) methods [111, 66, 185], although by themselves can be used as efficient solution methods, are usually applied as preconditioners for KSMs. These methods are uniformly efficient with respect to discretization scales and can be equipped with weighted smoothers (like weighted Jacobi and SOR methods [211]) to yield more robust convergence behavior for some partial differential equations. On the other hand, such a solver framework also introduces many parameters and an adaptive selecting procedure to choose solvers and parameters is critical for end users [17, 140, 26, 28, 81].

Machine learning (ML) techniques can naturally be applied to construct an adaptive or automatic procedure to choose good solver parameters in practice. There are several algorithms based on machine learning for classifying or selecting linear solvers [144]. Some focus on constructing models or selecting parameters for training [24, 113, 30, 25, 175]; some focus on learning algorithms to enhance performance [86, 141]; and, more recently, some focus on automatically constructing iterative methods using machine learning [104, 114, 128, 118]. We propose a two-step workflow for engineers to build adaptive linear solvers based on multilevel methods accelerated by KSMs:

- In the offline step, we determine a set of parameters based on simulation goals; select feature and variable parameters for a suitable machine learning model; and then train an initial model for selecting appropriate variable parameters.

- During the online step, we extract feature parameters based available problem information; obtain input parameters based on trained model and plug them into the solution procedure; and, finally, collect the resulting data for improving the offline model further.

10.5.1 Adaptive multilevel iterative solvers

Multilevel preconditioners like GMG and AMG might be, sometimes, too costly or, some other times, not robust with respect to physical parameters. Such solution strategy also introduce several parameters, like KSM method type, smoother type, smoother ordering, coarsening type, interpolation type, and so on. For a practical simulation problem, there could be many other parameters for describing underlying physics, mathematical model, discretization method, solution procedure, and available computing resource.

For the properties that can be parameterized, we can divide them into two groups: *feature parameters* and *free parameters*. *Feature parameters* or *features* include task characteristics (physical, discretization, solver, resource, etc) that are considered fixed for a specific simulation run. On the other hand, *free parameters* are the properties that can be adjusted by hand or automatically for solving different problems. Here the term,*free parameters*, is a general concept, which might be restart number of GMRES, type of preconditioner, number of smoothing steps in AMG, number of CPU cores to used, and so on.

Traditionally, when facing a particular problem, domain scientists need to choose a fixed preconditioner with an appropriate multigrid method using a-priori information. In some cases, local Fourier analysis (LFA) or local mode analysis can be applied to predict asymptotic convergence factor of multigrid algorithms [54, 185]. Moreover, approximations of convergence factor based on LFA can be obtained using automated procedures [193, 121, 117], it makes selecting a good multigrid method possible. Numerical software packages for carrying out LFA automatically [193, 2, 1] and optimizing multigrid parameters [157, 174, 67] are also available. However, it is usually difficult, if not impossible, to select the "best" method in advance; especially in the purely algebraic setting, as in most of the practical applications.

10.5.2 Constructing multigrid based on machine learning

Recently, more advanced deep network models are involved in the process of auto-tuning the iterative methods. The main feature of these methods is that advanced machine learning algorithms are applied to construct specific components of some iterative methods, like the prolongation and smoothing operators in multigrid methods [180, 174, 104, 128]. In [174], evolutionary algorithms were employed to choose the faster solver by adjusting the type of smoother, the number of smoothing steps, as well as the relaxation factor in each coarse level.

Both [128] and [104] focused on the prolongation operator in AMG—The first one used graph convolution neural networks to learn the weight coefficients in prolongation matrix P and the second one used ResNet considered previously in [180].

One particularly useful trick demonstrated in these papers is that we do not have to use the same type of data in the training set and test set. In fact, due to generality of neural networks, we can use the data with certain properties that is easy to calculate (for example small in size) in the training set, while using the data from difficult problem in test set. For example, the error propagation matrix of a two-level AMG with prolongation P and smoother S can be given by

$$E = (I - S^T A)[I - P(P^T AP)^{-1} P^T A](I - SA).$$

In general, the spectral radius of E is not easy to estimate for large-scale systems and we can construct, in the training set, matrices A with special structures that are relatively easy to calculate the spectral radius of the corresponding matrix E.

Furthermore, deep neural networks are also used as optimization techniques. Based on the existed methods, researchers utilize neural networks to optimize the parameters in those methods, in order to achieving better performance, and result in new methods which is different from the methods mentioned above. In [118], steps in GMG are considered as an analogy of layers in deep neural networks. Since the prolongation matrix P, restriction matrix R and the damping coefficient ω is differentiable in each step, therefore backpropagation approach in neural networks can be used to optimize P, R and ω. A method called DMG (Deep MultiGrid method) is derived after training, but one shortcoming of the method is for every new matrix, the whole process of training has to be re-run, which may be impractical. In [114], a variant of Jacobi iterative method generated by CNN(Convolutional Neural Network) [126] or U-Net [168] is illustrated using 2-D Poisson equation.

10.6 Homework problems

hw:GE **HW 10.1.** Try to implement the KIJ and IKJ variants of the Gaussian elimination method and design numerical tests to compare their performance. Are there other implementation strategies? Please specify.

hw:Saad1.33 **HW 10.2.** Suppose that M and N are two matrices which satisfy that $M \leqslant N$ and $N(i,j) \leqslant 0$ for all $i \neq j$. If M is an M-matrix, then N is an M-matrix.

Bibliography

[1] aLFA, https://gitlab.com/nilskintscher/alfa.

[2] LFA Lab, https://github.com/hrittich/lfa-lab.

[3] HPC Top500 list, https://www.top500.org/lists/2019/11/, 2019.

[4] Freely available software for linear algebra, http://www.netlib.org/utk/people/jackdongarra/la-sw.html, Last updated in 2021.

[5] D. Abts, J. Thompson, and G. Schwoerer. Architectural support for mitigating DRAM soft errors in large-scale supercomputers. Technical report, 2006.

[6] O. Axelsson. A survey of algebraic multilevel iteration (AMLI) methods. *BIT Numerical Mathematics*, 43:863–879, 2003.

[7] O. Axelsson and G. Lindskog. On the eigenvalue distribution of a class of preconditioning methods. *Numerische Mathematik*, 48(5):479–498, 1986.

[8] O. Axelsson and G. Lindskog. On the rate of convergence of the preconditioned conjugate gradient method. *Numerische Mathematik*, 48(5):499–523, 1986.

[9] I. Babuška. Error-bounds for finite element method. *Numerische Mathematik*, 16(4):322–333, 1971.

[10] 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.

[11] R. E. Bank, T. F. Chan, W. M. Coughran Jr, and R. K. Smith. The alternate-block-factorization procedure for systems of partial differential equations. *BIT Numerical Mathematics*, 29(4):938–954, 1989.

[12] R. E. Bank and T. Dupont. An optimal order process for solving finite element equations. *Mathematics of Computation*, 36(153):35–51, 1981.

[13] R. E. Bank and T. F. Dupont. Analysis of a two-level scheme for solving finite element equations. Technical report, 1980.

[14] R. E. Bank, T. F. Dupont, and H. Yserentant. The hierarchical basis multigrid method. *Numerische Mathematik*, 52(4):427–458, 1988.

[15] 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.

[16] R. E. Bank and H. Yserentant. Multigrid convergence: A brief trip down memory lane. *Computing and Visualization in Science*, 13(4):147–152, 2010.

[17] R. Barrett, M. Berry, J. Dongarra, V. Eijkhout, and C. Romine. Algorithmic bombardment for the iterative solution of linear systems: a poly-iterative approach. *Journal of Computational and applied Mathematics*, 74(1-2):91–109, 1996.

[18] M. Benzi, J. K. Cullum, and M. Tuma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 22(4):1318–1332, 2000.

[19] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, May 2005.

[20] 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.

[21] M. Benzi and M. Tuma. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing*, 19(3):968–994, 1998.

[22] M. Benzi and M. Tuma. A robust incomplete factorization preconditioner for positive definite matrices. *Numerical Linear Algebra with Applications*, 10(5-6):385–400, 2003.

[23] J. Bergh and J. Lofstrom. *Interpolation spaces: an introduction*, volume 223. Springer Science & Business Media, 2012.

[24] S. Bhowmick, V. Eijkhout, Y. Freund, E. Fuentes, and D. Keyes. Application of machine learning to the selection of sparse linear solvers. *Int. J. High Perf. Comput. Appl*, 2006.

[25] S. Bhowmick, V. Eijkhout, Y. Freund, E. Fuentes, and D. Keyes. Application of alternating decision trees in selecting sparse linear solvers. In *Software Automatic Tuning*, pages 153–173. Springer, 2011.

[26] S. Bhowmick, L. McInnes, B. Norris, and P. Raghavan. The role of multi-method linear solvers in pde-based simulations. In *International Conference on Computational Science and Its Applications*, pages 828–839. Springer, 2003.

[27] S. Bhowmick, L. McInnes, B. Norris, and P. Raghavan. Robust algorithms and software for parallel pde-based simulations. In *Proceedings of the Advanced Simulation Technologies Conference, ASTC*, volume 4, pages 18–22, 2004.

[28] S. Bhowmick, P. Raghavan, L. McInnes, and B. Norris. Faster pde-based simulations using robust composite linear solvers. *Future Generation Computer Systems*, 20(3):373–387, 2004.

[29] S. Bhowmick, P. Raghavan, and K. Teranishi. A combinatorial scheme for developing efficient composite solvers. In *International Conference on Computational Science*, pages 325–334. Springer, 2002.

[30] S. Bhowmick, B. Toth, and P. Raghavan. Towards low-cost, high-accuracy classifiers for linear solver selection. In *International Conference on Computational Science*, pages 463–472. Springer, 2009.

[31] B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, et al. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016.

[32] 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.

[33] M. Bollhöfer. A robust ilu with pivoting based on monitoring the growth of the inverse factors. *Linear Algebra and its Applications*, 338(1-3):201–218, 2001.

[34] M. Bollhöfer. A robust and efficient ilu that incorporates the growth of the inverse triangular factors. *SIAM Journal on Scientific Computing*, 25(1):86–103, 2003.

[35] F. Bornemann and H. Yserentant. A basic norm equivalence for the theory of multilevel methods. *Numerische Mathematik*, 64(1):455–476, 1993.

[36] D. Braess. The contraction number of a multigrid method for solving the poisson equation. *Numerische Mathematik*, 37(3):387–404, 1981.

[37] 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.

[38] 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.

[39] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Applied Numerical Mathematics*, 23(1):3–19, feb 1997.

[40] J. Bramble. *Multigrid methods*. Pitman research notes in mathematics series. Longman Scientific & Technical, 1993.

[41] J. Bramble and J. Pasciak. Iterative techniques for time dependent Stokes problems. *Computers Math. Applic.*, 33:13–30, 1997.

[42] J. H. Bramble and J. E. Pasciak. New convergence estimates for multigrid algorithms. *Mathematics of computation*, 49(180):311–329, 1987.

[43] 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.

[44] J. H. Bramble and J. E. Pasciak. The analysis of smoothers for multigrid algorithms. *Mathematics of Computation*, 58(198):467–488, 1992.

[45] J. H. Bramble and J. E. Pasciak. New estimates for multilevel algorithms including the V-cycle. *Mathematics of computation*, 60(202):447–471, 1993.

[46] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev. Analysis of the inexact uzawa algorithm for saddle point problems. *SIAM Journal on Numerical Analysis*, 34(3):1072–1092, 1997.

[47] 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.

[48] 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.

[49] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Mathematics of Computation*, 55(191):1–22, Jul. 1990.

[50] J. H. Bramble and J. Xu. Some estimates for a weighted L^2 projection. *Mathematics of Computation*, 56:463–476, 1991.

[51] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[52] A. Brandt. Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, 19(1-4):23–56, July 1986.

[53] 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.

[54] A. Brandt. Multigrid guide. Technical report, 2011.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] M. Brezina. An improved convergence analysis of smoothed aggregation algebraic multigrid. *Numerical Linear Algebra with Applications*, 19:441–469, 2012.

[61] 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.

[62] M. Brezina, R. Falgout, S. MacLachlan, T. a. Manteuffel, S. Mccormick, and J. Ruge. Aggregation (αSA) Multigrid. *SIAM Review*, 47(2):317–346, 2005.

[63] 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.

[64] 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.

[65] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. Siam, 2000.

[66] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2000.

[67] J. Brown, Y. He, S. MacLachlan, M. Menickelly, and S. M. Wild. Tuning multigrid methods with robust optimization. *arXiv preprint arXiv:2001.00887*, 2020.

[68] L. Chen. Deriving the XZ identity from auxiliary space method. In *Domain Decomposition Methods in Science and Engineering XIX*, pages 309–316. Springer, 2011.

[69] X.-l. Cheng. On the nonlinear inexact uzawa algorithm for saddle-point problems. *SIAM Journal on Numerical Analysis*, 37(6):1930–1934, 2000.

[70] 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.

[71] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Number v. 1 in Methods of Mathematical Physics. Wiley, 1991.

[72] T. A. Davis. *Direct methods for sparse linear systems*, volume 2. SIAM, 2006.

[73] T. A. Davis, S. Rajamanickam, and W. M. Sid-Lakhdar. A survey of direct methods for sparse linear systems. *Acta Numerica*, 25:383–566, 2016.

[74] B. A. de Dios, A. T. Barker, and P. S. Vassilevski. A combined preconditioning strategy for nonsymmetric systems. *SIAM Journal on Scientific Computing*, 36(6):A2533–A2556, 2014.

[75] J. Demmel, J. Dongarra, V. Eijkhout, E. Fuentes, A. Petitet, R. Vuduc, R. C. Whaley, and K. Yelick. Self-adapting linear algebra algorithms and software. *Proceedings of the IEEE*, 93(2):293–312, 2005.

[76] R. A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51, nov 2008.

[77] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J.-C. Andre, D. Barkai, J.-Y. Berthou, T. Boku, B. Braunschweig, F. Cappello, B. Chapman, a. Choudhary, S. Dosanjh, T. Dunning, S. Fiore, a. Geist, B. Gropp, R. Harrison, M. Hereld, M. Heroux, a. Hoisie, K. Hotta, Y. Ishikawa, F. Johnson, S. Kale, R. Kenway, D. Keyes, B. Kramer, J. Labarta, a. Lichnewsky, T. Lippert, B. Lucas, B. Maccabe, S. Matsuoka, P. Messina, P. Michielse, B. Mohr, M. S. Mueller, W. E. Nagel, H. Nakashima, M. E. Papka, D. Reed, M. Sato, E. Seidel, J. Shalf, D. Skinner, M. Snir, T. Sterling, R. Stevens, F. Streitz, B. Sugar, S. Sumimoto, W. Tang, J. Taylor, R. Thakur, a. Trefethen, M. Valero, a. van der Steen, J. Vetter, P. Williams, R. Wisniewski, and K. Yelick. The International Exascale Software Project roadmap. *International Journal of High Performance Computing Applications*, 25(1):3–60, Jan. 2011.

[78] J. Dongarra, G. Bosilca, Z. Chen, V. Eijkhout, G. E. Fagg, E. Fuentes, J. Langou, P. Luszczek, J. Pjesivac-Grbovic, K. Seymour, et al. Self-adapting numerical software (sans) effort. *IBM Journal of Research and Development*, 50(2.3):223–238, 2006.

[79] J. Dongarra and V. Eijkhout. Self-adapting numerical software and automatic tuning of heuristics. In *International Conference on Computational Science*, pages 759–767. Springer, 2003.

[80] J. Dongarra and V. Eijkhout. Self-adapting numerical software for next generation applications. *The International Journal of High Performance Computing Applications*, 17(2):125–131, 2003.

[81] J. Dongarra, S. Tomov, P. Luszczek, J. Kurzak, M. Gates, I. Yamazaki, H. Anzt, A. Haidar, and A. Abdelfattah. With extreme computing, the rules have changed. *Computing in Science & Engineering*, 19(3):52–62, 2017.

[82] 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.

[83] 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.

[84] 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.

[85] V. Eijkhout, E. Fuentes, N. Ramakrishnan, P. Kang, S. Bhowmick, D. Keyes, and Y. Freund. A self-adapting system for linear solver selection. In *Proc. 1st int'l workshop on automatic performance tuning (iWAPT2006)*, pages 44–53, 2006.

[86] P. R. Eller, J. R. C. Cheng, and R. S. Maier. Dynamic linear solver selection for transient simulations using multi-label classifiers. In *Procedia Computer Science*, volume 9, pages 1523–1532. Elsevier B.V., 2012.

[87] 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.

[88] L. C. Evans. *Partial Differential Equations*. American Mathematical Society, 1998.

[89] R. D. Falgout and J. B. Schroder. Non-Galerkin coarse grids for algebraic multigrid. *SIAM J. Sci. Comput.*, 36:C309–C334, 2014.

[90] R. D. Falgout and P. S. Vassilevski. On generalizing the algebraic multigrid framework. *SIAM J. Numer. Anal.*, 42:1669–1693, 2004.

[91] 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.

[92] K. FAN. NOTE ON M-MATRICES . *The Quarterly Journal of Mathematics*, 11(1):43–49, 01 1960.

[93] R. P. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Mathematics and Mathematical Physics*, 1(4):1092–1096, 1962.

[94] R. P. Fedorenko. The speed of convergence of one iterative process. *USSR Computational Mathematics and Mathematical Physics*, 4(3):227–235, 1964.

[95] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7(2):149–154, 01 1964.

[96] M. Fortin and R. Glowinski. *Augmented Lagrangian methods: applications to the numerical solution of boundary-value problems*, volume 15. Elsevier, 2000.

[97] S. Fortune. Voronoi diagrams and delaunay triangulations. In *Handbook of discrete and computational geometry*, pages 705–721. Chapman and Hall/CRC, 2017.

[98] M. Frigo and S. G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[99] G. P. Galdi. *An introduction to the mathematical theory of the Navier-Stokes equations: Steady-state problems*. Springer Science & Business Media, 2011.

[100] A. Gibbons. *Algorithmic graph theory*. Cambridge university press, 1985.

[101] R. Glowinski. *Numerical methods for nonlinear variational problems*. Springer-Verlag, New York, 1984.

[102] 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.

[103] 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.

[104] D. Greenfeld, M. Galun, R. Kimmel, I. Yavneh, and R. Basri. Learning to Optimize Multigrid PDE Solvers. *arXiv preprint arXiv:1902.10248*, feb 2019.

[105] C. Greif, T. Rees, and D. B. Szyld. Gmres with multiple preconditioners. *SeMA Journal*, 74(2):213–231, 2017.

[106] M. Griebel and P. Oswald. On the abstract theory of additive and multiplicative Schwarz algorithms. *Numerische Mathematik*, 180:163–180, 1995.

[107] F. Gustavson. Finding the block lower triangular form of a sparse matrix. In *Sparse matrix computations*, pages 275–289. Elsevier, 1976.

[108] W. Hackbusch. Ein iteratives verfahren zur schnellen auflösung elliptischer randwertprobleme. Technical report, 1976.

[109] W. Hackbusch. Multi-grid convergence theory. In *Multigrid methods*, pages 177–219. Springer, 1982.

[110] W. Hackbusch. *Multi-grid methods and applications*. Springer, 1985.

[111] W. Hackbusch. *Multi-grid methods and applications*. Springer Verlag, 1985.

[112] 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.

[113] A. Holloway and T.-Y. Chen. Neural Networks for Predicting the Behavior of Preconditioned Iterative Solvers. Technical report, 2007.

[114] J.-T. Hsieh, S. Zhao, S. Eismann, L. Mirabella, and S. Ermon. Learning Neural PDE Solvers with Convergence Guarantees. jun 2019.

[115] X. Hu, S. Wu, X. Wu, J. Xu, C. Zhang, S. Zhang, and L. Zikatanov. Combined preconditioning with applications in reservoir simulation. *Multiscale Modeling & Simulation*, 11(2):507–521, 2013.

[116] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, Cambridge, 1987.

[117] K. Kahl and N. Kintscher. Automated local fourier analysis (alfa). *BIT Numerical Mathematics*, pages 1–36, 2020.

[118] A. Katrutsa, T. Daulbaev, and I. Oseledets. Black-box learning of multigrid parameters. *Journal of Computational and Applied Mathematics*, 368:112524, 2020.

[119] R. Kettler. Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods. In *Multigrid methods*, pages 502–534. Springer, 1982.

[120] 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.

[121] N. Kintscher. *Automated Local Fourier Analysis (aLFA) and geometric multigrid for graphene*. PhD thesis, Universität Wuppertal, Fakultät für Mathematik und Naturwissenschaften ..., 2019.

[122] D. A. Knoll and D. E. Keyes. Jacobian-free newton–krylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[123] R. Kornhuber. Monotone multigrid methods for elliptic variational inequalities. I. *Numer. Math.*, 69(2):167–184, 1994.

[124] S. Lacroix, Y. V. Vassilevski, and M. F. Wheeler. Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS). *Numerical linear algebra with applications*, 8(8):537–549, 2001.

[125] 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.

[126] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[127] 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.

[128] I. Luz, M. Galun, H. Maron, R. Basri, and I. Yavneh. Learning Algebraic Multigrid Using Graph Neural Networks. *arXiv preprint arXiv:2003.05744*, mar 2020.

[129] 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.

[130] 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.

[131] 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).

[132] T. A. Manteuffel, S. Münzenmaier, J. Ruge, and B. S. Southworth. Nonsymmetric reduction-based algebraic multigrid. *SIAM J. Sci. Comput.*, 41:S242–S268, 2019.

[133] 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.

[134] K.-A. Mardal and R. Winther. Uniform preconditioners for the time dependent Stokes problem. *Numerische Mathematik*, 98(2):305–327, 2004.

[135] K.-A. Mardal and R. Winther. Erratum: Uniform preconditioners for the time dependent Stokes problem. *Numerische Mathematik*, 103(1):171–172, 2006.

[136] 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.

[137] P.-G. Martinsson and V. Rokhlin. A fast direct solver for boundary integral equations in two dimensions. *Journal of Computational Physics*, 205(1):1–23, 2005.

[138] S. F. McCormick. Multigrid methods for variational problems: general theory for the V-cycle. *SIAM Journal on Numerical Analysis*, 22(4):634–643, 1985.

[139] S. F. McCormick. *Multigrid methods*. SIAM, 1987.

[140] L. McInnes, B. Norris, S. Bhowmick, and P. Raghavan. Adaptive sparse linear solvers for implicit cfd using newton-krylov algorithms. In *Proceedings of the Second MIT Conference on Computational Fluid and Solid Mechanics*, volume 2, pages 1024–1028, 2003.

[141] P. Motter, K. Sood, E. Jessup, and B. Norris. Lighthouse. In *Proceedings of the 3rd International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering - SE-HPCCSE '15*, pages 16–24, New York, New York, USA, 2015. ACM Press.

[142] S. Mukherjee, J. Emer, and S. K. Reinhardt. The soft error problem: An architectural perspective. In *Proc. 11th Int'l Symp. on High-Performance Computer Architecture (HPCA)*, 2005.

[143] A. C. Muresan and Y. Notay. Analysis of aggregation-based multigrid. *SIAM Journal on Scientific Computing*, 30(2):1082–1103, 2008.

[144] K. Naono, K. Teranishi, J. Cavazos, and R. Suda. *Software automatic tuning: from concepts to state-of-the-art results*. Springer Science & Business Media, 2010.

[145] A. Napov and Y. Notay. Comparison of bounds for V-cycle multigrid. *Appl. Numer. Math*, 60(3):176–192, 2010.

[146] 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.

[147] A. Napov and Y. Notay. Algebraic analysis of aggregation-based multigrid. *Numerical Linear Algebra with Applications*, 18(3):539–564, 2011.

[148] 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.

[149] 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.

[150] M. Newman. Kantorovich's inequality. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, page 33, 1960.

[151] R. Nicolaides. On the ℓ^2 convergence of an algorithm for solving finite element equations. *Mathematics of Computation*, 31(140):892–906, 1977.

[152] Y. Notay. Convergence analysis of perturbed two-grid and multigrid methods. *SIAM journal on numerical analysis*, 45(3):1035–1044, 2007.

[153] Y. Notay. Algebraic analysis of two-grid methods: The nonsymmetric case. *Numerical Linear Algebra with Applications*, 17(1):73–96, jan 2010.

[154] Y. Notay. An aggregation-based algebraic multigrid method. *Electronic transactions on numerical analysis*, 37(6):123–146, 2010.

[155] Y. Notay. Algebraic theory of two-grid methods. *Numerical Mathematics: Theory, Methods and Applications*, 8(2):168–198, 2015.

[156] Y. Notay and P. S. Vassilevski. Recursive Krylov-based multigrid cycles. *Numerical Linear Algebra with Applications*, 15(July 2007):473–487, 2008.

[157] C. W. Oosterlee and R. Wienands. A genetic search for optimal multigrid components within a fourier analysis setting. *SIAM Journal on Scientific Computing*, 24(3):924–944, 2003.

[158] 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.

[159] S. V. Patankar. *Numerical heat transfer and fluid flow*. Hemisphere Publishing Corporation, 1980.

[160] S. Pissanetzky. *Sparse matrix technology*. Academic Press Inc. [Harcourt Brace Jovanovich Publishers], London, 1984.

[161] C. Qiao, S. Wu, J. Xu, and C.-S. C.-S. Zhang. Analytical Decoupling Techniques for Fully Implicit Reservoir Simulation. *Journal of Computational Physics*, 336:664–681, 2017.

[162] A. Rafiei and M. Bollhöfer. Robust incomplete factorization for nonsymmetric matrices. *Numerische Mathematik*, 118(2):247–269, 2011.

[163] J. R. Rice. On the construction of polyalgorithms for automatic numerical analysis. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, pages 301–313, 1967.

[164] J. R. Rice. The algorithm selection problem. In *Advances in computers*, volume 15, pages 65–118. Elsevier, 1976.

[165] S. Rippa. Minimal roughness property of the Delaunay triangulation. *Comput. Aided Geom. Design*, 7:489–497, 1990.

[166] C. Rodrigo, F. J. Gaspar, and L. T. Zikatanov. On the validity of the local Fourier analysis. *arXiv:1710.00408*, 2017.

[167] C. Rodrigo, F. J. Gaspar, and L. T. Zikatanov. On the validity of the local fourier analysis. *Journal of Computational Mathematics*, 37(3):340–348, 2018.

[168] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[169] J. Ruge and K. Stüben. Algebraic multigrid. *Multigrid methods*, 3:73–130, 1987.

[170] J. W. Ruge and K. Stüben. Algebraic multigrid. *in Multigrid Methods, Frontiers Appl. Math.,* SIAM, Philadelphia, 3:73–130, 1987.

[171] T. Rusten, R. Winther, and A. A preconditioned iterative method for saddlepoint problems. *SIAM Journal on Matrix Analysis and Applications*, 13(3):887–904, 1992.

[172] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, second edition, 2003.

[173] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Generation Computer Systems*, 20(3):475–487, 2004.

[174] J. Schmitt, S. Kuckuk, and H. Köstler. Optimizing geometric multigrid methods with evolutionary computation. *arXiv preprint arXiv:1910.02749*, 2019.

[175] W. M. Sid-Lakhdar, M. M. Aznaveh, X. S. Li, and J. W. Demmel. Multitask and transfer learning for autotuning exascale applications. *arXiv preprint arXiv:1908.05792*, 2019.

[176] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. Cfd vision 2030 study: a path to revolutionary computational aerosciences. Technical report, 2014.

[177] K. Sood. *Iterative Solver Selection Techniques for Sparse Linear Systems*. PhD thesis, University of Oregon, 2019.

[178] K. Stüben. An introduction to algebraic multigrid. In *Multigrid by U. Trottenberg, C. Oosterlee, and A. Schüller*, pages 413–532. 2001.

[179] K. Stüben and U. Trottenberg. Multigrid methods: Fundamental algorithms, model problem analysis and applications. 1982.

[180] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[181] D. B. Szyld. The many proofs of an identity on the norm of oblique projections. *Numerical Algorithms*, 42(3-4):309–323, 2006.

[182] X.-C. Tai. Rate of convergence for some constraint decomposition methods for nonlinear variational inequalities. *Numerische Mathematik*, 93:755–786, 2003.

[183] 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.

[184] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[185] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, 2001.

[186] 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.

[187] S. P. Vanka. Block-implicit multigrid solution of Navier-Stokes equations in primitive variables. *Journal of Computational Physics*, 65:138–158, 1986.

[188] P. Vaněk, M. Brezina, J. Mandel, et al. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematik*, 88(3):559–579, 2001.

[189] P. S. Vassilevski. *Multilevel Block Factorization Preconditioners*. 2008.

[190] F. Wang and J. Xu. A crosswind block iterative method for convection-dominated problems. *SIAM Journal on Scientific Computing*, 21(2):620–645, 1999.

[191] P. Wesseling. Theoretical and practical aspects of a multigrid method. *SIAM Journal on Scientific and Statistical Computing*, 3(4):387–407, 1982.

[192] 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.

[193] R. Wienands and W. Joppich. *Practical Fourier analysis for multigrid methods*. CRC press, 2004.

[194] G. Wittum. On the robustness of ilu smoothing. *SIAM journal on scientific and statistical computing*, 10(4):699–717, 1989.

[195] J. Wu, Y. Lee, J. Xu, and L. Zikatanov. Convergence analysis on iterative methods for semidefinite systems. *Journal of Computational Mathematics*, 26:797–815, 11 2008.

[196] J. Xu. Iterative methods by space decomposition and subspace correction. *SIAM Review*, 34:581–613, 1992.

[197] J. Xu. A new class of iterative methods for nonselfadjoint or indefinite problems. *SIAM journal on numerical analysis*, 29(2):303–319, 1992.

[198] J. Xu. The auxiliary space method and optimal multigrid preconditioning techniques for unstructured grids. *Computing*, 56:215–235, 1996.

[199] J. Xu and L. Zikatanov. A monotone finite element scheme for convection-diffusion equations. *Mathematics of Computation*, 68(228):1429–1446, 1999.

[200] 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.

[201] J. Xu and L. Zikatanov. Some observations on Babuška and Brezzi theories. *Numerische Mathematik*, 94(1):195–202, mar 2003.

[202] J. Xu and L. Zikatanov. Algebraic multigrid methods. *Acta Numer.*, 26:591–721, 2017.

[203] J. Xu and L. T. Zikatanov. Algebraic multigrid methods. *ArXiv e-prints*, Nov. 2016.

[204] S. Xu and Q. Jiang. *Six Lectures on Matrix Computation*. Higher Education Press, 2011.

[205] X. Xu. *Algebraic Theory of Multigrid Methods*. PhD thesis, University of Chinese Academy of Sciences, 2019.

[206] X. Xu. Parallel algebraic multigrid methods: State-of-the art and challenges for extreme-scale applications. *Journal on Numerical Methods and Computer Applications*, 40(4):243–260, 2019.

[207] X. Xu, Z. Mo, and H. An. An adaptive amg preconditioning strategy for solving largescale sparse linear systems. *SCIENCE CHINA Information Sciences*, (010):1411–1420, 2016.

[208] X. Xu, Z. Mo, X. Yue, H. An, and S. Shu. αsetup-amg: an adaptive-setup-based parallel amg solver for sequence of sparse linear systems. *CCF Transactions on High Performance Computing*, online first, 2020.

[209] X. Xu and C.-S. Zhang. On the ideal interpolation operator in algebraic multigrid methods. *SIAM J. Numer. Anal.*, 56:1693–1710, 2018.

[210] X. Xu and C.-S. Zhang. Convergence analysis of inexact two-grid methods: A theoretical framework. *SIAM J Numer Anal*, 60:133–156, 2022.

[211] S. Yang and M. K. Gobbert. The optimal relaxation parameter for the sor method applied to the poisson equation in any space dimensions. *Applied Mathematics Letters*, 22(3):325–331, 2009.

[212] K. Yoshida. *Functional Analysis*. Springer-Verlag, 1971.

[213] H. Yserentant. On the multi-level splitting of finite element spaces. *Numerische Mathematik*, 49(4):379–412, 1986.

[214] H. Yserentant. Two preconditioners based on the multi-level splitting of finite element spaces. *Numerische Mathematik*, 58(1):163–184, 1990.

[215] H. Yserentant. Old and new convergence proofs for multigrid methods. *Acta Numerica*, 2(1993):285–326, 1993.

[216] X. Yue, S. Shu, X. Xu, and Z. Zhou. An adaptive combined preconditioner with applications in radiation diffusion equations. *Communications in Computational Physics*, 18(5):1313–1335, 2015.

[217] X. Yue, X. Xu, and S. Shu. Jasmin-based two-dimensional adaptive combined preconditioner for radiation diffusion equations in inertial fusion research. *East Asian Journal on Applied Mathematics*, 7(3):495–507, 2017.

[218] X. Zhang. Multilevel Schwarz methods. *Numerische Mathematik*, 63(1):521–539, 1992.

[219] L. T. Zikatanov. Two-sided bounds on the convergence rate of two-level methods. *Numer. Linear Algebra Appl.*, 15(5):439–454, 2008.

Index