Sharpened CG Iteration Bound for High-contrast Heterogeneous Scalar Elliptic PDEs

Going beyond condition number

WI5005: Thesis Project

Philip Soliman

TU Delft
Delft University of Technology

Sharpened CG Iteration Bound for High-contrast Heterogeneous Scalar Elliptic PDEs

Going beyond condition number

by

Philip Soliman

To obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on T.B.A.

Student number:	4945255	
Project duration:	December 2024 – September 2025	
Thesis committee:	Prof. H. Schuttelaars,	TU Delft, responsible supervisor
	Dr. A. Heinlein,	TU Delft, daily supervisor
	F. Cumaru,	TU Delft, daily co-supervisor
Faculty:	Faculty of Electrical Engineering, Mathematics and Computer Science	
Department:	Delft Institute of Applied Mathematics (DIAM)	

Published on Thursday 28th August, 2025 .

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Generative AI was used to assist in the writing of this thesis. The author is responsible for the content and accuracy of the work.

TU Delft

Contents

Nomenclature

Symbols

Table 1: Symbols related to the heterogeneous elliptic problem.

Symbol	Description
Ω	Bounded domain in $\mathbb{R}^d$ with Lipschitz boundary.
$\partial\Omega$	Boundary of Ω.
$\mathbb{R}^d$	d-dimensional Euclidean space.
$\mathcal{C}$	Scalar coefficient field in the elliptic problem, assumed to lie in $L^\infty(\Omega)$.
$\mathcal{C}_{\min}$	Lower bound of the scalar coefficient $\mathcal{C}$.
$\mathcal{C}_{\max}$	Upper bound of the scalar coefficient $\mathcal{C}$.
u	Exact solution of the elliptic problem.
f	Source term belonging to $L^2(\Omega)$.
u_D	Dirichlet boundary data.
u_h	Finite element approximation of u.
V_h	Finite-dimensional subspace of $H_0^1(\Omega)$.
V	Solution space $\{u \in H^1(\Omega) \mid u_{\delta\Omega} = u_D\}$.
$V_{h,0}$	Subspace $V_h \cap H_0^1(\Omega)$ with homogeneous boundary conditions.
$H_0^1(\Omega)$	Sobolev space of functions with zero trace on the boundary.
$\{\phi_i\}_{i=1}^n$	Basis functions spanning V_h.
$\mathcal{T}$	Triangulation of the domain Ω.
$\mathcal{N}$	Set of degrees of freedom (DOFs).
n	Number of degrees of freedom, $n = \lvert\mathcal{N}\rvert$.
$a(u_h, v_h)$	Bilinear form $\int_\Omega \mathcal{C}\nabla u_h \cdot \nabla v_h \, dx$.
(f, v_h)	Linear form $\int_\Omega f v_h \, dx$.
A	Stiffness matrix derived from the Galerkin method.
$\mathbf{b}$	Load vector in the resulting linear system.
$\mathbf{u}$	Solution vector with components u_i.
v_h	Test function in V_h.
$\lVert\cdot\rVert_A$	A-norm defined by $\lVert x\rVert_A = \sqrt{x^T A x}$.

Table 2: Symbols related to the Conjugate Gradient (CG) method.

Symbol	Description
$\mathbf{u}_0$	Initial guess for the solution.
$\mathbf{u}^*$	Exact solution of the linear system.
$\mathbf{u}_m$	Approximate solution at the m^{th} iteration.
$\mathbf{r}_0$	Initial residual defined as $\mathbf{b} - A\mathbf{u}_0$.
$\mathbf{r}_j$	Residual vector at the j^{th} iteration.
$\mathbf{p}_j$	Search direction at iteration j.
α_j	Step size computed at iteration j.
β_j	Coefficient used to update the search direction in iteration j.
$\mathcal{K}_m(A, \mathbf{r}_0)$	Krylov subspace spanned by $\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}$.
$\mathcal{K}_m$	Shorthand for Krylov subspace $\mathcal{K}_m(A, \mathbf{r}_0)$.
$\mathcal{L}$	Constraint subspace in projection methods.

Symbol	Description
V	Matrix whose columns span the subspace $\mathcal{K}$.
$\mathbf{c}$	Correction vector in projection methods.
H	Hessenberg matrix defined as $H = V^T AV$.
T_m	Tridiagonal Hessenberg matrix arising from the Lanczos process.
$\mathbf{v}_j$	Lanczos vectors forming orthonormal basis for Krylov subspace.
δ_j	Diagonal entries of T_m.
η_j	Off-diagonal entries of T_m.
$\mathbf{e}_m$	Error at iteration m, defined as $\mathbf{u}^* - \mathbf{u}_m$.
$\mathcal{P}_{m-1}$	Space of polynomials of degree at most $m-1$.
$r_m(x)$	Residual polynomial of degree m with $r_m(0) = 1$.
$q_{m-1}(x)$	Solution polynomial of degree $m-1$ in CG.
μ	Grade of a vector with respect to a matrix.
λ_i	Eigenvalues of A.
$\lambda_{\min}$	Smallest eigenvalue of A.
$\lambda_{\max}$	Largest eigenvalue of A.
$\kappa(A)$	Condition number of A, defined as $\lambda_{\max}/\lambda_{\min}$.
ξ_i	Components of the initial error in the eigenvector basis of A.
$\sigma(A)$	Spectrum (set of eigenvalues) of A.
C_m	Chebyshev polynomial of degree m.
$\hat{C}_m$	Real-valued transformed Chebyshev polynomial adapted to interval $[a, b]$.
ρ_0	Initial residual in the eigenvector basis of A.
Q	Orthonormal eigenbasis matrix in diagonalization $A = QDQ^T$.
D	Diagonal matrix of eigenvalues in diagonalization.
ϵ	Relative error tolerance.
ϵ_r	Relative residual tolerance.
ϵ_b	Residual tolerance relative to right-hand side.
$\tilde{\epsilon}$	Absolute error tolerance.
m	Number of CG iterations required for convergence.
m_1	Classical CG iteration bound based on condition number.
f	CG convergence rate $f = \dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$.

Table 3: Symbols related to Schwarz preconditioners.

Symbol	Description
Ω_i	Subdomains obtained from partitioning Ω.
R_i	Restriction operator for subdomain Ω_i.
D_i	Diagonal matrix representing the partition of unity (weights) for Ω_i.
N_{sub}	Number of subdomains.
M	Preconditioner matrix.
M_{ASM}^{-1}	Additive Schwarz preconditioner defined by $M_{\text{ASM}} = \sum_{i=1}^{N_{sub}} R_i^T (R_i A R_i^T)^{-1} R_i$.
M_{RAS}^{-1}	Restrictive additive Schwarz preconditioner defined by $M_{\text{RAS}} = \sum_{i=1}^{N_{sub}} R_i^T D_i (R_i A R_i^T)^{-1} R_i$.
A_i	Local operator on subdomain Ω_i, defined as $A_i = R_i A R_i^T$.
R_0	Restriction operator associated with the coarse space.
A_0	Coarse operator defined as $A_0 = R_0^T A R_0$.
P_j	Local projection operator associated with subdomain Ω_j.
P_0	Projection operator for the coarse space.

Symbol	Description
P_{ad}	Sum of the projection operators, $P_{ad} = \sum_{j=1}^{N_{sub}} P_j$, used in the two-level method.
$\kappa(P_{ad})$	Condition number of the preconditioned system, given by $\frac{\lambda_{\max}}{\lambda_{\min}}$ of P_{ad}.
C_0	Constant in stability decomposition for coarse space analysis.
k_0	Maximum number of subdomains that overlap at any point.
$\mathcal{P}_j$	A-symmetric projection operator for subdomain Ω_j.
E_j	Extension operator from subdomain Ω_j to global domain.

Table 4: Symbols related to eigenspectra and CG convergence bounds chapter 4 (Methodology).

Symbol	Description
$\sigma_1(A)$	Two-cluster eigenspectrum of A, defined as the union of two intervals $[a, b] \cup [c, d]$.
$\sigma_2(A)$	Eigenspectrum comprising a main cluster and a tail of eigenvalues, with the tail denoted by N_{tail}.
$[a, b]$	Interval containing the first cluster of eigenvalues.
$[c, d]$	Interval containing the second cluster of eigenvalues.
N_{tail}	Number of eigenvalues in the tail cluster (when the spectrum has one cluster plus a tail).
m_2	Sharpened CG iteration bound for two-cluster eigenspectrum.
p	Degree parameter in two-cluster CG bound.
κ_l	Left cluster condition number $\kappa_l = b/a$.
κ_r	Right cluster condition number $\kappa_r = d/c$.
$\hat{r}_p^{(i)}(x)$	Residual polynomial function for the left cluster, defined piecewise for the two-cluster ($i = 1$) and tail-cluster ($i = 2$) spectrum
$\hat{r}_{\bar{m}-p}(x)$	Residual polynomial function based on Chebyshev polynomials, corresponding to the complementary polynomial degree $\bar{m} - p$.
$C_p^{(i)}$	Cluster-specific Chebyshev polynomial of degree p, adapted to the eigenvalue distribution of the i^{th} cluster.
η_1	Upper bound for $\max_{x\in[a,b]} \lvert\hat{r}_p^{(1)}(x)\rvert$, given by $2\left(\frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}\right)^p$.
η_2	Upper bound for $\max_{x\in[a,b]} \lvert\hat{r}_p^{(2)}(x)\rvert$, expressed as $\left(\frac{b}{a}-1\right)^p$ when $p = N_{\text{tail}}$.
$\bar{m}$	Total degree of the composite residual polynomial $r_{\bar{m}}$, formed as the product $\hat{r}_p^{(i)}(x)\hat{r}_{\bar{m}-p}(x)$.
P	Performance ratio defined as $P = m_1/m_2$.
P_{uniform}	Performance ratio for uniform eigenspectrum case.
$q(\kappa_l, \kappa_r)$	Function appearing in uniform spectrum performance analysis.
$T_\kappa(\kappa_l, \kappa_r)$	Threshold function for determining when $m_2 < m_1$.
$W_{-1}(x)$	Lambert W function (principal branch -1).
$z_1^{(i,j)}, z_2^{(j)}$	Chebyshev coordinates in the frame of reference of the i^{th} cluster.
$\zeta_1^{(i,j)}, \zeta_2^{(j)}$	Transformed Chebyshev coordinates for multi-cluster analysis.
k	Positive integer parameter in the Chebyshev inequality, corresponding to the degree in the bound estimate.
p_i	Chebyshev degree associated with the i^{th} eigenvalue cluster, determining the contraction factor of that cluster's contribution to the residual.
κ_i	Condition number of the i^{th} eigenvalue cluster, defined as the ratio of the largest to smallest eigenvalue within the cluster.

Symbol	Description
f_i	Convergence factor (or spectral measure) for the i^{th} cluster.
Λ_t	Set of all eigenvalues residing in tail clusters.
$r_t(x)$	Residual polynomial for tail clusters: $r_t(x) = \prod_{\lambda \in \Lambda_t} \left(1 - \frac{x}{\lambda}\right)$.
$m_{N_{\text{cluster}}}$	Multi-cluster CG iteration bound.
$m_{N_{\text{tail-cluster}}}$	Multi-tail-cluster CG iteration bound.
K^*	Sorted partition indices for eigenspectrum clustering.
k^*	Split index that maximizes the ratio of consecutive eigenvalues.
I_t	Set of tail cluster start indices.

Table 5: Symbols related to implementation and numerical experiments.

Symbol	Description
Q_h	Fine quadrilateral mesh with mesh size h.
Q_H	Coarse quadrilateral mesh with mesh size H.
h	Fine mesh size, where $h = H/2^r$.
H	Coarse mesh size.
r	Refinement parameter such that $h = H/2^r$.
$\mathcal{Q}$	Set of mesh pairs $\{(Q_h, Q_H)\}$ used in experiments.
v_i^h	Internal fine mesh vertex.
q_j	Quadrilateral element in mesh.
$\mathcal{C}_{\text{const}}$	Constant coefficient function $\mathcal{C}_{\text{const}} \equiv 1$.
$\mathcal{C}_{\text{3layer, vert}}$	High-contrast coefficient function with three-layer vertical structure.
$\mathcal{C}_{\text{edge slabs, around vertices}}$	High-contrast coefficient function with edge slabs around vertices.
$\mathcal{M}^{-1}$	Set of preconditioners $\{M^{-1}_{\text{2-OAS-GDSW}}, M^{-1}_{\text{2-OAS-RGDSW}}, M^{-1}_{\text{2-OAS-AMS}}\}$.
$M^{-1}_{\text{2-OAS-GDSW}}$	Two-level overlapping additive Schwarz preconditioner with GDSW coarse space.
$M^{-1}_{\text{2-OAS-RGDSW}}$	Two-level overlapping additive Schwarz preconditioner with RGDSW coarse space.
$M^{-1}_{\text{2-OAS-AMS}}$	Two-level overlapping additive Schwarz preconditioner with AMS coarse space.
N_{iter}	Number of iterations for early estimation experiments.
f_{iter}	Fraction parameter for determining N_{iter}.
N_{update}	Update frequency for eigenvalue convergence detection.
τ_{extremal}	Tolerance parameter for extremal eigenvalue convergence.
m_{estimate}	Heuristic estimate defined as $m_{\text{estimate}} = \frac{1}{2}(m_{N_{\text{cluster}}} + m_{N_{\text{tail-cluster}}})$.

Abbreviations

Table 6: List of abbreviations and their full meanings.

Abbreviation	Full Meaning
CG	Conjugate gradient
PCG	Preconditioned conjugate gradient
SPD	Symmetric positive definite
FEM	Finite element method
ASM	Additive Schwarz method
RAS	Restrictive additive Schwarz
ORAS	Optimized restrictive additive Schwarz
2-OAS	Two-level overlapping additive Schwarz
MsFEM	Multiscale finite element method
ACMS	Approximate component mode synthesis
GDSW	Generalized Dryja-Smith-Widlund

Abbreviation	Full Meaning
RGDSW	Robust generalized Dryja-Smith-Widlund
AMS	Algebraic multiscale solver
DtN	Dirichlet-to-Neumann
DBC	Dirichlet boundary condition
NBC	Neumann boundary condition
PDE	Partial differential equation
DOF(s)	Degree(s) of freedom
LU	Lower-Upper decomposition
GMRES	Generalized minimal residual method

1 Introduction

In this thesis we focus on a simple scalar diffusion problem. Let $\Omega \subset \mathbb{R}^d$ be a bounded domain with Lipschitz boundary $\partial\Omega$, $\mathcal{C} \in L^\infty(\Omega)$ be scalar field defined on Ω such that $0 < \mathcal{C}_{\min} \leq \mathcal{C}(x) \leq \mathcal{C}_{\max} < \infty$ for all $x \in \Omega$ and $f \in L^2(\Omega)$ be a source term, then we define

Problem 1.1: High-contrast scalar elliptic problem: strong formulation

Let $u_D \in H^2(\partial\Omega)$ be a Dirichlet boundary condition. Find $u \in H^2(\Omega)$ such that

$$\begin{aligned} -\nabla \cdot (\mathcal{C}\nabla u) &= f \quad \text{in } \Omega, \\ u &= u_D \quad \text{on } \partial\Omega. \end{aligned} \tag{1.1}$$

In table 1.1 we give an overview of some real-world applications that can be modeled using Problem 1.1. These applications span a wide range of fields, including electronics, hydrogeology, biomechanics, materials science, and energy.

Table 1.1: Overview of Real-World High-Contrast Elliptic Problems

Application	Domain	High-Contrast Scenario
Thermal management of PCBs	Electronics / Thermal Eng.	Copper traces (high conductivity) in an epoxy substrate (low conductivity).
Subsurface fluid flow	Hydrogeology / Petroleum Eng.	High-permeability sandstone channels adjacent to low-permeability shale.
Electrical impedance tomography	Biomechanics / Medical Imaging	Varying electrical conductivities of biological tissues (muscle, fat, bone).
Analysis of composite materials	Materials Sci. / Structural Mech.	Stiff carbon fibers embedded in a compliant polymer matrix.
Transport in fuel cells	Energy / Electrochemistry	High gas diffusivity in open pores vs. low diffusivity in the solid material of the porous structure.

The first step in solving Problem 1.1 is to reformulate the problem in a way that reduces the regularity constraint on the solution $u \in H^2(\Omega)$ and not requiring to $\nabla \cdot (\mathcal{C}\nabla u)$ to exist pointwise. This leads to the

weak formulation of the problem, which is obtained by multiplying equation (1.1) with a test function $v \in H_0^1(\Omega)$ and integrating over Ω. The weak formulation is given by

Problem 1.2: High-contrast scalar elliptic problem: weak formulation

Let $u_D \in H^1(\partial\Omega)$ be a Dirichlet boundary condition. Find $u \in V = \{u \in H^1(\Omega) | u_{\delta\Omega} = u_D\}$ such that $\forall v \in H_0^1(\Omega)$

$$\int_\Omega \mathcal{C}\nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx. \tag{1.2}$$

To solve this problem numerically, we need to discretize the domain Ω and the solution space V. To that end we consider a triangulation $\mathcal{T}$ of the domain Ω with the $\mathcal{N}$ the set of degrees of freedom (DOFs). Then, we pick a finite dimensional subspace of V, V_h spanned by a set of basis functions ϕ_i defined locally on each of the elements $\tau \in \mathcal{T}$

$$V_h = \text{span}\{\phi_i\}_{i=1}^n,$$

where $n = |\mathcal{N}|$. This leads to the discretized weak formulation

Problem 1.3: High-contrast scalar elliptic problem: discretized weak formulation

Let $u_D \in H^1(\partial\Omega)$ be a Dirichlet boundary condition. Find $u_h \in V_h$ such that $\forall v_h \in V_{h,0} = V_h \cap H_0^1(\Omega)$

$$a(u_h, v_h) = \int_\Omega \mathcal{C}\nabla u_h \cdot \nabla v_h \, dx = \int_\Omega f v_h \, dx = (f, v_h), \tag{1.3}$$

where $a(u_h, v_h)$ is the bilinear form and (f, v_h) is the linear form. Equation (1.3) gives rise to the following system of equations

$$A\mathbf{u} = \mathbf{b}, \quad A_{ij} = a(\phi_i, \phi_j), \; b_i = (f, \phi_i) \quad \forall i, j \in \mathcal{N},$$

where $A \in \mathbb{R}^{n \times n}$ is the (by construction) symmetric stiffness matrix, $\mathbf{u} \in \mathbb{R}^n$ the solution vector with components u_i and $\mathbf{b} \in \mathbb{R}^n$ the load vector. The load vector $\mathbf{b}$ is constructed from the source term f and the boundary conditions. The approximate solution u_h is constructed from the basis functions ϕ_i as

$$u_h = \sum_{i \in \mathcal{N}} u_i \phi_i.$$

Note that the bilinear form a in Problem 1.3 is coercive, meaning that for all $0 \neq w \in H_0^1(\Omega)$ we have

$$a(w, w) = \int_\Omega \mathcal{C}\nabla w \cdot \nabla w \, dx = \int_\Omega \mathcal{C}|\nabla w|^2 \, dx = \mathcal{C}_{\text{min}} \|\nabla w\|_{L_2(\Omega)}^2 > \frac{\mathcal{C}_{\text{min}}}{C_p^2} \|\nabla w\|_{H_0^1(\Omega)}^2 \geq 0,$$

since $\mathcal{C}_{\text{min}} > 0$ and with C_p the Poincaré constant. Moreover, for

$$w = \sum_{i \in \mathcal{N}} w_i \phi_i \quad \text{with} \quad \mathbf{w} = (w_1, w_2, \ldots, w_n)^T \neq \mathbf{0}$$

we have

$$\mathbf{w}^T A \mathbf{w} = a(w, w) > 0.$$

It follows that A is positive definite, making it symmetric positive definite (SPD). Additionally, through the coercive property of a in combination with the Lax-Milgram theorem we get that both the continuous and discrete weak formulations Problems 1.2 and 1.3 are well-posed and have unique solutions.

Apart from possibly complex domains Ω, a major obstacle in solving the linear system $A\mathbf{u} = \mathbf{b}$ from Problem 1.3, comes from its high-contrast coefficient $\mathcal{C}$, which requires a broad range of element sizes $|\tau|$ in the triangulation $\mathcal{T}$ to fully resolve. As a result, the number of DOFs $n = |\mathcal{N}|$ can be very large, leading to a system matrix A that is large and sparse. This makes direct methods like Gaussian

elimination, LU- or Cholesky decomposition impractical, as they require storing the entire matrix in memory and generally have complexity $\mathcal{O}(n^3)$.

Though A is large and sparse, it *is* SPD. Therefore, the linear system $A\mathbf{u} = \mathbf{b}$ can be solved using the Conjugate Gradient method (CG). CG requires only the ability to compute matrix-vector products with A (complexity $\mathcal{O}(n)$ for sparse matrices) and does not require storing the entire matrix. Being an iterative method, CG produces a sequence of approximations $\mathbf{u}_i,\ i = 1, \ldots, m$ to the solution $\mathbf{u}$ and stops when some convergence criterion depending on a desired tolerance ϵ is met. This means that CG's complexity is given by $\mathcal{O}(mn)$.

Hence, the number of iterations m required for convergence is a crucial factor in the performance of CG. The key subject of section 2.1.3 is to analyze the convergence of CG and how it depends on the properties of the system matrix A. In particular, we will show that m is related to the distribution of the eigenvalues of A. For instance, in exact arithmetic m is bounded by the number of distinct eigenvalues of A, say k, from which follows that CG's complexity is given by $\mathcal{O}(kn)$. In general, we can derive, using a well-known bound on CG's convergence rate discussed in Theorem 2.7, an explicit expression for CG's complexity

$$\mathcal{O}\left(\sqrt{\kappa(A)}\log\left(\frac{2}{\epsilon}\right)n\right), \tag{1.4}$$

where $\kappa(A)$ is the condition number of A. Comparing equation (1.4) with the complexity of direct methods, we see that CG is much more efficient for large sparse SPD matrices like A.

However, the difficulty of allowing for a high-contrast coefficient $\mathcal{C}$ and, for instance, complex domains resides in that the condition number $\kappa(A)$ can be very large, which in turn increases CG's iteration count and complexity. Accounting for the high-contrast coefficient $\mathcal{C}$ concerns the construction of robust *coarse spaces*, some examples of which are given in chapter 3. On the other hand, handling of complex domains can be done using *domain decomposition methods* and this is the topic of section 2.2.

The particular implementation of domain decomposition method and coarse space results in a preconditioner matrix $M \in \mathbb{R}^{n\times n}$, which is used to transform the system $A\mathbf{u} = \mathbf{b}$ into a new system $M^{-1}A\mathbf{u} = M^{-1}\mathbf{b}$ with a (hopefully) smaller condition number. The preconditioned CG method (PCG), described in section 2.1.5, is then used to solve the transformed system. Consequently, using the equivalent of the CG complexity bound equation (1.4) for PCG, we can determine the performance of PCG with the preconditioner M as

$$\mathcal{O}\left(\sqrt{\kappa(M^{-1}A)}\log\left(\frac{2}{\epsilon}\right)n\right). \tag{1.5}$$

This thesis seeks to review the applicability of the bound in equation (1.5) to problems like Problem 1.1. The bound relies on an overestimation of the actual number of iterations m required for convergence and overstates the role of the condition number $\kappa(M^{-1}A)$ in determining the convergence rate of CG. We will see in section 2.1.4 that the actual number of iterations m is much smaller than the classical bound that equation (1.5) relies on and that the condition number $\kappa(M^{-1}A)$ is not the only factor influencing the convergence rate of CG.

The main research question in this work is as follows:

Research Question. How can we sharpen the CG iteration bound for Schwarz-preconditioned high-contrast heterogeneous scalar-elliptic problems beyond the classical condition number-based bound?

For instance, in [1], a two-level Schwarz preconditioner with either one of the AMS and GDSW coarse spaces significantly outperforms the same preconditioner with RGDSW coarse space, despite all three preconditioned systems having similar condition numbers. The key differences appear in their spectral gap and cluster width, highlighting the need for further investigation into spectral properties beyond the condition number.

Subsidiary Questions. To answer the main research question, we address the following subsidiary questions:

- **Q1** Given a certain eigenspectrum, can we construct a bound that is sharper than the classical condition number-based bound?
- **Q2** How does the number of CG iterations necessary for convergence of high-contrast heterogeneous problems depend on spectral characteristics other than the condition number?

Q3 Can we obtain a priori estimates for the number of iterations necessary for convergence of a PCG method with Schwarz-like preconditioners?

This research has great practical importance for the selection of the most efficient preconditioner for a given high-contrast problem. The condition number does not suffice to differentiate between preconditioners, as outlined in [1]. Thus, having the ability to differentiate preconditioners based on spectral characteristics from, for instance, their approximate eigenspectra would improve the selection process.

On top of that, having sharper bounds for (P)CG methods allows for more efficient allocation of computational resources, as it allows for a more accurate estimate of the number of iterations necessary for convergence. This is particularly important in high-performance computing environments, where the cost of each iteration can be significant.

As mentioned above, the computational complexity of (P)CG methods is given in equations (1.4) and (1.5). If we can find a bound that scales better with the other spectral characteristics, we can possibly show that (P)CG methods with Schwarz-like preconditioners are applicable to a wider range of high-contrast problems than previously thought. This would open up new avenues for research and applications in the field of numerical analysis and scientific computing.

Finding sharper bounds than the classical condition number-based bound is a non-trivial task. The main challenge lies in the fact that the condition number is a measure of the worst-case scenario, while we are interested in the average-case behavior of the (P)CG method. This requires a more nuanced understanding of the eigenspectrum and its impact on the convergence of the method.

Assuming we have some expression for a sharper bound, the following challenge then lies in obtaining a priori estimates for the spectrum of the preconditioned system. The literature does provide condition number estimates for various Schwarz preconditioners. For instance, in the simple cases of the additive Schwarz preconditioner with either a **ASM type I coarse space** or **ASM type II coarse space** an a priori estimate for the condition number is given by equation (B.4) in combination with either equation (B.6) or equation (B.7), respectively. The same can be said for the MsFEM and ACMS preconditioners, as is seen in section 3.2.

However, these estimates are not always sharp, and they do not provide information about the spectral gap or cluster width of the preconditioned system. So, the challenge remains; how can we obtain a priori estimates of the spectral characteristics necessary for the sharp bound?

Fortunately, we can always obtain a posteriori estimates for the spectrum of the preconditioned system, as is done in chapter 6. However, this requires the computation of the eigenspectrum of the preconditioned system, which can be computationally expensive. This is where the use of iterative methods such as (P)CG comes in handy, as they allow us to compute the eigenspectrum during the solution of the linear system.

This thesis is organized as follows. In chapter 2 (Mathematical background), the mathematical background of the CG method and Schwarz methods is introduced. chapter 3 (Related Work) reviews related work on coarse spaces and improved CG iteration bounds. In chapter 4 (Methodology), we expand on the available literature and ultimately derive two novel algorithms for a sharpened CG iteration bounds designed for high-contrast problems. The performance and practical applicability of the sharpened CG iteration bounds are then evaluated in chapter 6 (Results). Finally, chapter 7 (Conclusion) summarizes the key insights and discusses directions for future research.

2

Mathematical background

In section 2.1 (Conjugate gradient method) we discuss the classification of the CG method as both an error projection and a Krylov subspace method. We then derive the convergence rate of CG in section 2.1.3 (Convergence of CG), as well as the influence of the eigenvalues of the system matrix A on that rate in section 2.1.4 (Influence of eigenvalue distribution on CG convergence). The latter is important, since it shows that the condition number $\kappa(A)$ is not the only factor influencing the convergence rate of CG. The first part ends with a brief review of the PCG method in section 2.1.5 (Preconditioned CG). Then, the second part of this chapter section 2.2 (Schwarz methods) concerns the Schwarz methods, which are a class of domain decomposition methods. Schwarz methods can be used to construct preconditioners for use in PCG, even though these were originally devised as fixed point iteration methods for solving PDEs on complex domains.

2.1. Conjugate gradient method

We seek to solve the linear system of equations $A\mathbf{u} = \mathbf{b}$, where A is a symmetric positive definite (SPD) matrix. These properties of A make the CG method particularly suitable for solving the system, as motivated in chapter 1 (Introduction).

2.1.1. Projection methods

To begin to understand the CG method, we need to introduce the class of *projection methods*, which given some initial guess $\mathbf{u}_0$ find an approximation $\mathbf{u}_{\text{new}}$ to the solution of the linear system $A\mathbf{u} = \mathbf{b}$ in a *constrained subspace* $\mathcal{L} \subset \mathbb{R}^n$ using a *correction vector* $\mathbf{c}$ in another subspace $\mathcal{K} \subset \mathbb{R}^n$. That is, projection methods solve the following problem [21, Equation 5.3]

$$\text{Find } \mathbf{u}_{\text{new}} = \mathbf{u}_0 + \mathbf{c} \in \mathbf{u}_0 + \mathcal{K} \text{ such that } \mathbf{r}_{\text{new}} = A\mathbf{u}_{\text{new}} - \mathbf{b} = \mathbf{r}_0 - A\mathbf{c} \perp \mathcal{L}.$$

In doing so, projection methods perform a *projection step*, visualized in figure 2.1.

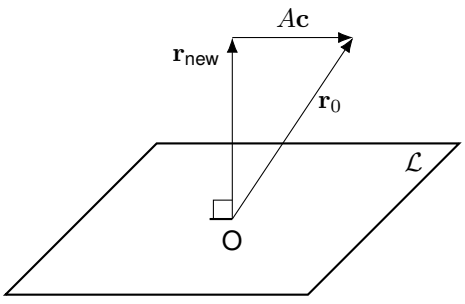

Figure 2.1: Visualization of projection method, based on [21, Figure 5.1]. The projection method finds the solution $\mathbf{u}_{\text{new}}$ in the affine subspace $\mathbf{u}_0 + \mathcal{K}$, such that the new residual $\mathbf{r}_{\text{new}}$ is orthogonal to the constraint subspace $\mathcal{L}$.

Error projection methods

The subclass of *error projection methods* defined by Definition 2.1 sets $\mathcal{K} = \mathcal{L}$.

Definition 2.1: Error projection method

To perform an error projection step, find $\mathbf{u}_{\text{new}} = \mathbf{u}_0 + \mathbf{c} \in \mathbf{u}_0 + \mathcal{K}$ such that

$$(\mathbf{r}_0 - A\mathbf{c}, w) = 0 \quad \forall w \in \mathcal{K}, \tag{2.1}$$

where $(\cdot, \cdot)$ is an inner product on $\mathcal{K}$. Let $V = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m]$ be a matrix whose columns span $\mathcal{K}$, then one error projection step is given by

$$\begin{aligned} \mathbf{u}_{\text{new}} &= \mathbf{u}_0 + V\mathbf{v} \\ V^T A V \mathbf{v} &= V^T \mathbf{r}_0, \end{aligned}$$

Error projection methods owe their name to the following central Theorem 2.1.

Theorem 2.1: Error minimization in A-norm

Given a linear system $A\mathbf{u} = \mathbf{b}$ with A SPD and exact solution $\mathbf{u}^*$. Define the errors $\mathbf{e}_0 = \mathbf{u}^* - \mathbf{u}_0$ and $\mathbf{e}_{\text{new}} = \mathbf{u}^* - \mathbf{u}_{\text{new}}$. Then, an error projection step minimizes the A-norm of the error in the affine subspace $\mathbf{u}_0 + \mathcal{K}$.

Proof. We have

$$\begin{aligned} A\mathbf{e}_{\text{new}} &= A(\mathbf{e}_0 - \mathbf{c}) \\ &= A(\mathbf{u}^* - \mathbf{u}_0 - \mathbf{c}) \\ &= \mathbf{r}_0 - A\mathbf{c} \end{aligned}$$

Hence, the orthogonality condition in equation (2.1) can be written as

$$(A\mathbf{e}_{\text{new}}, w) = (\mathbf{e}_0 - \mathbf{c}, w)_A = 0, \quad \forall w \in \mathcal{K}.$$

In other words, the correction vector $\mathbf{c}$ is the A-orthogonal projection of the error $\mathbf{e}_0$ onto $\mathcal{K}$. Therefore, there exists a projection operator $P_{\mathcal{K}}^A$ such that $\mathbf{c} = P_{\mathcal{K}}^A \mathbf{e}_0$ and we can write

$$\mathbf{e}_{\text{new}} = (I - P_{\mathcal{K}}^A)\mathbf{e}_0.$$

Moreover, we have using symmetry and positive definiteness of A that we can define the A-norm $\|\cdot\|_A$. Then, using the A-orthogonality between the error $\mathbf{e}_{\text{new}}$ and the correction $P_{\mathcal{K}}^A \mathbf{e}_0$, we get

$$\begin{aligned} \|\mathbf{e}_0\|_A^2 &= \|\mathbf{e}_{\text{new}}\|_A^2 + \|P_{\mathcal{K}}^A \mathbf{e}_0\|_A^2 \\ &\geq \|\mathbf{e}_{\text{new}}\|_A^2, \end{aligned}$$

which shows that the new error is smaller than the previous error in the A-norm. To show that the error projection step minimizes the A-norm of the error in the affine subspace $\mathbf{u}_0 + \mathcal{K}$, we let $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{y} \in \mathbf{u}_0 + \mathcal{K}$ be arbitrary, then using that $P_{\mathcal{K}}^A \mathbf{x} - \mathbf{y} \in \mathbf{u}_0 + \mathcal{K}$ we get

$$\begin{aligned} \|\mathbf{x} - \mathbf{y}\|_A^2 &= \|\mathbf{x} - P_{\mathcal{K}}^A \mathbf{x} + P_{\mathcal{K}}^A \mathbf{x} - \mathbf{y}\|_A^2 \\ &= \|\mathbf{x} - P_{\mathcal{K}}^A \mathbf{x}\|_A^2 + \|P_{\mathcal{K}}^A \mathbf{x} - \mathbf{y}\|_A^2 \\ &\geq \|\mathbf{x} - P_{\mathcal{K}}^A \mathbf{x}\|_A^2, \end{aligned}$$

which yields that [21, Theorem 1.38]

$$\min_{\mathbf{y} \in \mathbf{u}_0 + \mathcal{K}} \|\mathbf{x} - \mathbf{y}\|_A = \|\mathbf{x} - P_{\mathcal{K}}^A \mathbf{x}\|_A. \tag{2.2}$$

Now, substituting $\mathbf{x} = \mathbf{e}_0$ and $\mathbf{y} = \mathbf{c}$ into equation (2.2), we again find $\mathbf{c} = P_{\mathcal{K}}^A \mathbf{e}_0$, giving the desired result. □

General algorithm for error projection methods

By performing multiple (error) projection steps we obtain a sequence of approximations $\{\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_m\}$ to exact the solution $\mathbf{u}^*$ of the linear system $A\mathbf{u} = \mathbf{b}$. Theorem 2.1 ensures that each approximate solution $\mathbf{u}_j$ is closer to the exact solution $\mathbf{u}^*$ than the previous one $\mathbf{u}_{j-1}$. This idea forms the basis for a general error projection method and results in algorithm 1

Algorithm 1 Prototype error projection method [21, Algorithm 5.1]

Set $\mathbf{u} = \mathbf{u}_0$
while $\mathbf{u}$ is not converged **do**
  Choose basis V of $\mathcal{K} = \mathcal{L}$
  $\mathbf{r} = \mathbf{b} - A\mathbf{u}$
  $\mathbf{c} = (V^T AV)^{-1} V^T \mathbf{r}$
  $\mathbf{u} = \mathbf{u} + V\mathbf{c}$
end while

Projection methods differ in their choice of the spaces $\mathcal{K}$ and $\mathcal{L}$, as well as in how they obtain the basis V of $\mathcal{K}$ and the so-called *Hessenberg matrix* defined in Definition 2.2.

Definition 2.2: Hessenberg matrix

The Hessenberg matrix H is defined as the matrix $H = V^T AV$, where V is a matrix whose columns span the subspace $\mathcal{K}$.

Krylov subspace methods

Krylov subspace methods form yet another subclass of projection methods and are defined by their choice of the space $\mathcal{K}$. Namely,

$$\mathcal{K}_m(A_0, \mathbf{r}_0) = \text{span}\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}, \tag{2.3}$$

or $\mathcal{K}_m$ as a shorthand.

The dimension of $\mathcal{K}_m(A_0, \mathbf{r}_0)$ is related to the grade of $\mathbf{r}_0$ with respect to A_0, which is defined in Definition 2.3.

Definition 2.3: Grade of a vector

The grade of a vector $\mathbf{v}$ with respect to a matrix A is the lowest degree of the polynomial q such that $q(A)\mathbf{v} = 0$.

Consequently,

Theorem 2.2: Dimension of Krylov subspace

The Krylov subspace $\mathcal{K}_m$ is of dimension m if and only if the grade μ of $\mathbf{v}$ with respect to A is not less than m [21, proposition 6.2],

$$\dim(\mathcal{K}_m) = m \iff \mu \geq m,$$

such that

$$\dim(\mathcal{K}_m) = \min\{m, \text{grade}(\mathbf{v})\}. \tag{2.4}$$

A key property of the Krylov subspace methods is their ability to represent product of polynomials of A by some vector $\mathbf{v} \in \mathbb{R}^n$ in terms of section of A in $\mathcal{K}_m$, see Definitions 2.4 and 2.5.

Definition 2.4: Restriction of an operator

The action of a matrix A can be thought of as a mapping

$$\mathbb{R}^n \to \mathbb{R}^n : \mathbf{v} \mapsto A\mathbf{v}$$

Thus the domain and codomain of A are $\mathbb{R}^n$. Let $X \subset \mathbb{R}^n$, we can consider the map

$$X \to \mathbb{R}^n : \mathbf{v} \mapsto A\mathbf{v}$$

instead. The only difference from A is that the domain is X. This map is defined as the restriction $A_{|_X}$ of A to X.

Definition 2.5: Section of an operator

Let Q be a projector onto the subspace X. Then the section of the operator A onto X is defined as $QA_{|_X}$.

Finally, we can state the polynomial representation property of the Krylov subspace methods.

Theorem 2.3: Polynomial representation

Let Q_m be any projector onto $\mathcal{K}_m$ and let A_m be the section of A to $\mathcal{K}_m$, that is, $A_m = Q_m A_{|\mathcal{K}_m}$. Then for any polynomial q of degree not exceeding $m-1$ [21, proposition 6.3],

$$q(A)\mathbf{v} = q\left(A_m\right)\mathbf{v}$$

and for any polynomial of degree $\leq m$,

$$Q_m q(A)\mathbf{v} = q\left(A_m\right)\mathbf{v}$$

Theorem 2.3 shows that, for given restriction and projection matrices Q_m and P_m to $\mathcal{K}_m$ and a matrix of the form $H_m = Q_m^T A P_m$, $q(H_m)$ represents the action of $q(A)$ on $\mathbf{v}$. Choosing $Q_m = P_m = V_m$, we obtain the Hessenberg matrix from Definition 2.2 and equation (A.2) resulting from algorithm A.2. Additionally, from the Cayley-Hamilton theorem, we know that $m < n$. Therefore, the dimension of $\mathcal{K}_m$ is smaller than n. It follows that Krylov subspaces are able to efficiently represent the product of polynomials of A by $\mathbf{v}$ with the smaller dimensional matrix H_m. This is a crucial property for the CG method as well, since the solution it generates contains a product of a polynomial of A and the initial residual $\mathbf{r}_0$, see Theorem 2.4. Moreover, the Theorem 2.3 ensures that CG converges in a finite amount of steps, which technically makes it an exact solution method instead of an iterative one.

2.1.2. CG algorithm

The CG method exists in the intersection of error projection methods and Krylov subspace methods, as it is a projection method with the choice $\mathcal{L} = \mathcal{K} = \mathcal{K}_m$. We can derive the CG method starting from Arnoldi's method, see algorithm A.1. Arnoldi's method is much like algorithm 1 in that it uses a Gram-Schmidt orthogonalization procedure to obtain the basis V of the Krylov subspace $\mathcal{K}_m$ and a projection step to update the solution. Where Arnoldi's method is applicable to non-symmetric matrices by performing a full orthogonalization step, CG leverages the symmetry of A by doing a partial orthogonalization step. The latter is possible, since the CG method only needs to maintain the orthogonality of the residuals $\mathbf{r}_j$ with respect to the previous residuals $\mathbf{r}_{j-1}, r_{j-2}$ and not with respect to all previous residuals $\mathbf{r}_{j-3}, \ldots, \mathbf{r}_0$. The full derivation is discussed in the Appendix, section A and results in algorithm 2.

Algorithm 2 Conjugate Gradient Method

$\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$, $\mathbf{p}_0 = \mathbf{r}_0$, $\beta_0 = 0$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\alpha_j = (\mathbf{r}_j, \mathbf{r}_j)/(A\mathbf{p}_j, \mathbf{p}_j)$
  $\mathbf{u}_{j+1} = \mathbf{u}_j + \alpha_j \mathbf{p}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j A\mathbf{p}_j$
  $\beta_j = (\mathbf{r}_{j+1}, \mathbf{r}_{j+1})/(\mathbf{r}_j, \mathbf{r}_j)$
  $\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j$
end for

A crucial property of the approximate solution that algorithm 2 produces is given in Theorem 2.4

Theorem 2.4: CG approximate solution

The approximate solution at the m^{th} iteration is given by

$$\mathbf{u}_m = \mathbf{u}_0 + \sum_{i=0}^{m-1} c_i A^i \mathbf{r}_0 = \mathbf{u}_0 + q_{m-1}(A)\mathbf{r}_0, \tag{2.5}$$

where $q_{m-1}(A)$ is the solution polynomial of degree $m-1$ in A.

Proof. The CG method is a projection method with the choice $\mathcal{L} = \mathcal{K} = \mathcal{K}_m$. Hence, the approximate solution $\mathbf{u}_m$ is an element of the affine Krylov subspace $\mathbf{u}_0 + \mathcal{K}_m(A, \mathbf{r}_0)$, see Definition 2.1. The result follows from the fact that the Krylov subspace $\mathcal{K}_m(A, \mathbf{r}_0)$ is spanned by the vectors $\{\mathbf{r}_0, A\mathbf{r}_0, A^2\mathbf{r}_0, \ldots, A^{m-1}\mathbf{r}_0\}$ and that the approximate solution $\mathbf{u}_m$ is a linear combination of these vectors. The coefficients of this linear combination are given by the *CG solution coefficients* c_i. □

2.1.3. Convergence of CG

We derive a general bound for the error of the CG method in

Theorem 2.5: CG general error bound

Suppose we apply the CG method to the linear system $A\mathbf{u} = \mathbf{b}$ with A SPD and the exact solution $\mathbf{u}^*$. Then, the error of the m^{th} iterate $\mathbf{e}_m = \mathbf{u}^* - \mathbf{u}_m$ is bounded as

$$||\mathbf{e}_m||_A < \min_{r \in \mathcal{P}_{m-1}, r(0)=1} \max_{\lambda \in \sigma(A)} |r(\lambda)| ||\mathbf{e}_0||_A. \tag{2.6}$$

Proof. Combining the results of Theorem 2.1 and Theorem 2.4 we get that the error of the m^{th} iterate of the CG algorithm $\mathbf{e}_m = \mathbf{u}^* - \mathbf{u}_m$ satisfies

$$\|\mathbf{e}_m\|_A = \|(I - Aq_{m-1}(A))\mathbf{e}_0\|_A = \min_{q \in \mathcal{P}_{m-1}} \|(I - Aq(A))\mathbf{e}_0\|_A = \min_{r \in \mathcal{P}_m, r(0)=1} \|r(A)\mathbf{e}_0\|_A, \tag{2.7}$$

where $r_m(A) = I - Aq_{m-1}(A)$ is the *residual polynomial*. The right-hand side of equation (2.7) can be further bounded by letting λ_i, ξ_i be the eigenvalues of A and the components of $\mathbf{e}_0$ in the eigenbasis of A, respectively. Then

$$||r(A)\mathbf{e}_0||_A = \sqrt{\sum_{i=1}^{n} |r(\lambda_i)|^2 |\xi_i|^2} \leq \max_{\lambda \in \sigma(A)} |r(\lambda)| ||\mathbf{e}_0||_A,$$

which gives the desired result. □

Convergence criteria

We say that the CG method *converges* to a user-defined, absolute tolerance $\tilde{\epsilon}$ if the error of the m^{th} iterate $\mathbf{e}_m$ satisfies

$$\|\mathbf{e}_m\|_A \leq \tilde{\epsilon}.$$

Theorem 2.5 allows us to define a criterion based on a *relative tolerance* ϵ, see Definition 2.6

Definition 2.6: Convergence criterion

The CG method is said to have *converged* to a user-defined, relative tolerance ϵ if

$$\frac{\|\mathbf{e}_m\|_A}{\|\mathbf{e}_0\|_A} \leq \epsilon,$$

which according to Theorem 2.5 is satisfied when

$$\min_{r \in \mathcal{P}_{m-1}, r(0)=1} \max_{\lambda \in \sigma(A)} |r(\lambda)| \leq \epsilon. \tag{2.8}$$

However, the criterion in Definition 2.6 is not useful in practice, since it involves the usually unknown error $\mathbf{e}$. Luckily, Theorem 2.6 shows how we can relate the ratio of A-norms of the error to a similar ratio of the 2-norms of the residuals.

Theorem 2.6: Residual convergence criterion

The CG method has converged to a user-defined, relative tolerance ϵ in the sense of Definition 2.6 if

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{r}_0\|_2} \leq \frac{\epsilon}{\sqrt{\kappa}}, \tag{2.9}$$

where $\kappa = \dfrac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$ is the condition number of A.

Proof. We have for $i = 0, \ldots, m$ that

$$\mathbf{e}_i = A^{-1}\mathbf{b} - \mathbf{u}_i = A^{-1}(\mathbf{b} - A\mathbf{u}_i) = A^{-1}\mathbf{r}_i.$$

Therefore, we can write

$$\frac{\|\mathbf{e}_m\|_A}{\|\mathbf{e}_0\|_A} = \frac{\mathbf{r}_0^T A^{-T}\mathbf{r}_m}{\mathbf{r}_0^T A^{-T}\mathbf{r}_0} = \frac{\|\mathbf{r}_m\|_{A^{-1}}}{\|\mathbf{r}_0\|_{A^{-1}}},$$

where the last equality follows as A^{-1} is SPD. Now, by Theorem D.1 and using that the eigenvalues of A^{-1} are the inverses of the eigenvalues of A, we can bound the A^{-1}-norm of the residuals as

$$\|\mathbf{r}_m\|_{A^{-1}} \leq \frac{1}{\sqrt{\lambda_{\text{min}}}}\|\mathbf{r}_m\|_2,$$

and

$$\|\mathbf{r}_0\|_{A^{-1}} \geq \frac{1}{\sqrt{\lambda_{\text{max}}}}\|\mathbf{r}_0\|_2.$$

Hence, we can write

$$\frac{\|\mathbf{e}_m\|_A}{\|\mathbf{e}_0\|_A} \leq \sqrt{\kappa}\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{r}_0\|_2}.$$

To conclude, if we perform the CG method until the convergence criterion equation (2.9) is satisfied, we have

$$\frac{\|\mathbf{e}_m\|_A}{\|\mathbf{e}_0\|_A} \leq \sqrt{\kappa}\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{r}_0\|_2} \leq \epsilon,$$

which gives the desired result. □

An important conclusion to draw from Theorem 2.6 is that if we set some relative tolerance for the residuals ϵ_r such that we stop iterating when

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{r}_0\|_2} \leq \epsilon_r, \tag{2.10}$$

then we get that the CG method has converged to a relative error tolerance ϵ given by

$$\frac{\|\mathbf{e}_m\|_A}{\|\mathbf{e}_0\|_A} \leq \epsilon = \frac{\epsilon_r}{\sqrt{\kappa}}.$$

This means that the CG method converges to a relative tolerance ϵ that is smaller than the relative tolerance of the residuals ϵ_r by a factor of $\sqrt{\kappa}$. On the one hand, this allows us to set a convergence criterion based on the residuals, which can actually be computed during CG iterations, as is the point of Theorem 2.6. On the other hand, the convergence criterion based on the residuals is also pessimistic by the same factor of $\sqrt{\kappa}$. In other words, the CG method performs more iterations to converge to a stricter tolerance than the user-defined tolerance ϵ_r.

Moreover, using Theorem D.1 we can also bound the absolute error tolerance $\tilde{\epsilon}$ of the CG method in terms of the initial residual. That is, suppose we set ϵ_r as a convergence criterion, then we get

$$\tilde{\epsilon} \leq \frac{\epsilon_r}{\sqrt{\kappa}}\|\mathbf{e}_0\|_A = \frac{\epsilon_r}{\sqrt{\kappa}}\|\mathbf{r}_0\|_{A^{-1}} \leq \frac{\epsilon_r}{\sqrt{\lambda_{\text{max}}}}\|\mathbf{r}_0\|_2. \tag{2.11}$$

From Theorem 2.6 and choosing $\mathbf{u}_0 = \mathbf{0} \in \mathbb{R}^n$, we can derive a commonly implemented convergence criterion

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{b}\|_2} \leq \epsilon_b. \tag{2.12}$$

In this case, the relative error tolerance ϵ achieved by the CG method in the sense of Definition 2.6 still depends on the initial guess. Suppose, we choose a 'good' initial guess such that $\|\mathbf{r}_0\|_2 \leq \|\mathbf{b}\|_2$, then

$$\frac{\|\mathbf{r}_m\|_2}{\|\mathbf{b}\|_2} \leq \frac{\|\mathbf{r}_m\|_2}{\|\mathbf{r}_0\|_2},$$

which means equation (2.12) is more optimistic than equation (2.10), requiring less CG iterations before it is satisfied. However, if we choose a 'bad' initial guess such that $\|\mathbf{r}_0\|_2 \geq \|\mathbf{b}\|_2$, then equation (2.12) is more pessimistic than equation (2.10) and will require more CG iterations before it is satisfied. For a good (bad) initial guess we therefore achieve a relative error tolerance larger (smaller) than $\frac{\epsilon_b}{\sqrt{\kappa}}$. In regard to the absolute error tolerance $\tilde{\epsilon}$, we can write using Theorem D.1 that

'bad' initial guess $\tilde{\epsilon} \geq \frac{\epsilon_b}{\sqrt{\kappa}}\|\mathbf{e}_0\|_A = \frac{\epsilon_b}{\sqrt{\kappa}}\|\mathbf{r}_0\|_{A^{-1}} \geq \frac{\sqrt{\lambda_{\text{min}}}}{\lambda_{\text{max}}}\epsilon_b\|\mathbf{r}_0\|_2 \geq \frac{\sqrt{\lambda_{\text{min}}}}{\lambda_{\text{max}}}\epsilon_b\|\mathbf{b}\|_2,$

'good' initial guess $\tilde{\epsilon} \leq \frac{\epsilon_b}{\sqrt{\kappa}}\|\mathbf{e}_0\|_A = \frac{\epsilon_b}{\sqrt{\kappa}}\|\mathbf{r}_0\|_{A^{-1}} \leq \frac{\lambda_{\text{min}}}{\sqrt{\lambda_{\text{max}}}}\epsilon_b\|\mathbf{r}_0\|_2 \leq \frac{\lambda_{\text{min}}}{\sqrt{\lambda_{\text{max}}}}\epsilon_b\|\mathbf{b}\|_2.$

In other words, when we use equation (2.12) as a convergence criterion, the absolute error tolerance we achieve satisfies

$$\frac{\sqrt{\lambda_{\text{min}}}}{\lambda_{\text{max}}}\epsilon_b\|\mathbf{b}\|_2 \leq \tilde{\epsilon} \leq \frac{\lambda_{\text{min}}}{\sqrt{\lambda_{\text{max}}}}\epsilon_b\|\mathbf{b}\|_2. \tag{2.13}$$

Comparing equation (2.11) and equation (2.13) we gain the insight that when

$$\|\mathbf{r}_0\|_2 < \frac{\epsilon_b}{\epsilon_r\sqrt{\kappa}}\|\mathbf{b}\|_2,$$

using convergence criterion equation (2.9) is guaranteed to give a more accurate result than equation (2.12). On the other hand, if $\|\mathbf{r}_0\|_2$ is larger, then it might be better to use equation (2.12) as a convergence criterion. In practice, criterion equation (2.12) is used, since its accuracy bound in equation (2.13) holds, independent of initial guesses.

Convergence rate

Next to convergence criteria based on residuals, we can also try to find a solution to the minimization problem in equation (2.9), which gives us an expected convergence rate. Under the assumption of a uniform distribution of the eigenvalues of A, we can further bound the error of the m^{th} iterate of the CG algorithm by a Chebyshev polynomial. This is done in Theorem 2.7 and is a direct consequence of Theorem C.1.

Theorem 2.7: Convergence rate of CG

Let the linear system $A\mathbf{u} = \mathbf{b}$ be as in Theorem 2.5 and let the eigenvalues of A be uniformly distributed in the interval $[\lambda_{\text{min}}, \lambda_{\text{max}}]$. Then the error of the m^{th} iterate of the CG algorithm satisfies

$$\|\mathbf{e}_m\|_A \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \|\mathbf{e}_0\|_A. \tag{2.14}$$

Proof. We use the general expression for CG's error from Theorem 2.5 in combination with the uniform distribution of eigenvalues in $[\lambda_{\text{min}}, \lambda_{\text{max}}]$ to write the error of the m^{th} iterate of the CG algorithm as

$$\|\mathbf{e}_m\|_A \leq \min_{r\in\mathcal{P}_{m-1}, r(0)=1} \max_{\lambda\in[\lambda_{\text{min}},\lambda_{\text{max}}]} |r(\lambda)|\|\mathbf{e}_0\|_A, \tag{2.15}$$

which by Theorem C.1 is solved by the real-valued scaled Chebyshev polynomial $\hat{C}_m$ from Definition C.2 with $[a, b] = [\lambda_{\text{min}}, \lambda_{\text{max}}]$ and $\gamma = 0$. We obtain

$$\|\mathbf{e}_m\|_A \leq \frac{1}{d_m(0)}\|\mathbf{e}_0\|_A = \frac{1}{C_m(\frac{\kappa+1}{\kappa-1})}\|\mathbf{e}_0\|_A, \tag{2.16}$$

where $\kappa = \dfrac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$ is the condition number of A. Using the approximation from equation (C.2) and setting $\tilde{z} = \dfrac{\kappa+1}{\kappa-1}$ we can write

$$\begin{aligned}
\frac{1}{d_m(0)} &= \frac{1}{C_m(\tilde{z})} \\
&\leq \frac{2}{\left(\tilde{z}+\sqrt{\tilde{z}^2-1}\right)^m} \\
&= 2\left(\tilde{z}-\sqrt{\tilde{z}^2-1}\right)^m \\
&= 2\left(\frac{\kappa+1-2\sqrt{\kappa}}{\kappa-1}\right)^m \\
&= 2\left(\frac{(\sqrt{\kappa}-1)^2}{(\sqrt{\kappa}-1)(\sqrt{\kappa}+1)}\right)^m \\
&= 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m.
\end{aligned}$$

Substituting this into equation (2.16) gives us the desired result. □

From Theorem 2.7 and Definition 2.6 we get that the CG method converges to a user-defined, relative tolerance ϵ if

$$\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \leq \frac{\epsilon}{2},$$

such that number of iterations m is bounded by

$$m \leq \left\lceil \log_f\left(\frac{\epsilon}{2}\right)\right\rceil = \text{int}\left[\log_f\left(\frac{\epsilon}{2}\right)+1\right], \tag{2.17}$$

where $f = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} < 1$ is the CG *convergence rate*[1]. Equation 2.17 could in principle be used as a stopping criterion, but it is not practical, since we generally do not know the condition number κ of A a priori.

The iteration bound given in equation (2.17) can be bounded further by using the expansion

$$\ln\left(\frac{z-1}{z+1}\right) = -2z + \mathcal{O}(z^3),$$

to write

$$\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^m \leq \exp\left(-\frac{2m}{\sqrt{\kappa}}\right) \leq \frac{\epsilon}{2},$$

which can be rearranged to give

$$m \leq \left\lceil \frac{\sqrt{\kappa}}{2}\ln\left(\frac{2}{\epsilon}\right)\right\rceil = \left\lfloor \frac{\sqrt{\kappa}}{2}\ln\left(\frac{2}{\epsilon}\right)+1\right\rfloor = m_1. \tag{2.18}$$

Equation 2.18 is the one used in the introduction to discuss CG's complexity.

At this point, it is necessary to note that even though the Chebyshev polynomial is optimal for the conditions of Theorem 2.7, it is not guaranteed that the eigenvalues of A are uniformly distributed. In fact, the eigenvalues of A are often clustered around some value. This means that the Chebyshev polynomial is not optimal in this case, and we can actually expect a better convergence rate than equation (2.14), as we will see in section 2.1.4. Even though the bounds from equations (2.14) and (2.17) are not sharp for non-uniform distributions, they are still correct as upper bounds.

For general distributions of eigenvalues we can still derive a useful property of the CG method. Instead of the Chebyshev polynomial $\hat{C}_m$ we pose a test polynomial r_{test} of degree m that satisfies the constraints of the minimization problem in equation (2.6). This test polynomial is given by

$$r_{\text{test}}(t) = \prod_{i=1}^{m} \frac{\lambda_i - t}{\lambda_i}.$$

Indeed, note that $r_{\text{test}} \in \mathcal{P}_m$, $r_{\text{test}}(0) = 1$ and $r_{\text{test}}(\lambda_i) = 0$ for $i = 1, 2, \ldots, m$. We obtain for $m = n = |\mathcal{N}|$ that

$$||\mathbf{e}_n||_A = ||\mathbf{e}_0||_A \max_{\lambda\in\sigma(A)} |r_{\text{test}}(\lambda)| = 0,$$

which implies that CG solves the linear system in n iterations. Furthermore, if there are only k distinct eigenvalues, then the CG iteration terminates in at most k iterations.

The strategy of posing a test polynomial r_{test} that satisfies the constraints of the minimization problem to come up with a bound for the error of the CG method is not limited to the Chebyshev polynomial from Theorem 2.7 or the test polynomial r_{test}. In fact, we can use any polynomial r of degree m that satisfies the constraints of the minimization problem. Evaluating the maximum value of this polynomial on the eigenspectrum of A would then result in an error bound. The problem is of course that we want to find a polynomial that gives a *sharp* error bound. In section 3.3 we discuss some recent literature using this strategy to achieve sharper bounds than equation (2.14) and in chapter 4 we apply the same strategy to find a bound for the error of the CG method for the case of clustered eigenvalues.

2.1.4. Influence of eigenvalue distribution on CG convergence

The bound on CG's error in Theorem 2.7 and the convergence rate in equation (2.17) are based on the assumption that the eigenvalues of A are uniformly distributed. However, this is not always the case. In this section we discuss the influence of the eigenvalue distribution on the convergence of the CG method. In particular, we will see that if the eigenvalues are clustered around some value, then we can expect a better convergence rate than equation (2.14).

We can write the diagonalization of A as $A = QDQ^T$ with Q being the orthonormal eigenbasis and D the diagonal matrix of eigenvalues, since A is symmetric. The residual polynomial from equation (2.7)

[1] Note that the $\log(f)$ is negative, flipping in the inequality.

can then be expressed as $r_m(A) = I - Aq_{m-1}(A) = Q(I - Dq_{m-1}(D))Q^T = Vr_m(D)V^T$. As seen in equation (2.7), the error of the m^{th} iterate of the CG algorithm is given by

$$||\mathbf{e}_m||_A^2 = ||r_m(A)\mathbf{e}_0||_A^2,$$

and

$$\begin{aligned} ||r_m(A)\mathbf{e}_0||_A^2 &= \mathbf{e}_0^T r_m(A)^T A r_m(A)\mathbf{e}_0 \\ &= \mathbf{e}_0^T Q r_m(D) Q^T Q D Q^T Q r_m(D) Q^T \mathbf{e}_0 \\ &= (Q^T\mathbf{e}_0)^T r_m(D) D r_m(D) Q^T \mathbf{e}_0. \end{aligned}$$

We also have

$$\begin{aligned} Q^T\mathbf{e}_0 &= Q^T A^{-1}\mathbf{r}_0 \\ &= Q^T Q D^{-1} Q^T \mathbf{r}_0 \\ &= D^{-1}\rho_0, \end{aligned}$$

where $\rho_0 = Q^T\mathbf{r}_0 \in \mathbb{R}^n$ is the initial residual vector in the eigenvector basis of A. Therefore,

$$\begin{aligned} ||r_m(A)\mathbf{e}_0||_A^2 &= \rho_0^T D^{-1} r_m(D) D r_m(D) D^{-1}\rho_0 \\ &= \rho_0^T r_m(D) D^{-1} r_m(D)\rho_0 \\ &= \sum_{i=1}^{n} \frac{r_m(\lambda_i)^2}{\lambda_i}\rho_{0,i}^2, \end{aligned}$$

which gives

$$||\mathbf{e}_m||_A^2 = \sum_{i=1}^{n} \frac{r_m(\lambda_i)^2}{\lambda_i}\rho_{0,i}^2. \tag{2.19}$$

From equation (2.19) we learn two important aspects about the convergence of the CG method. First, loosely stated, we see that smaller eigenvalues of A lead to larger errors. This is because the eigenvalues of A are in the denominator of equation (2.19). Consequently, the second aspect is that the error of the m^{th} iterate of the CG algorithm is given by a weighted sum, in which the weights are given by the components of the initial residual in the eigenbasis of A. As stated in section 2.1.3, a 'good' initial guess $\mathbf{u}_0$ leads to lower absolute error tolerance, i.e. faster convergence. Now, we learn that a good initial guess $\mathbf{u}_0$ is one that has small components in the eigenbasis of A corresponding to the smallest eigenvalues.

To obtain the residual polynomial r_m, we can use the recurrence relation between the Lanczos vectors $\mathbf{v}_j$ in Theorem A.3 and expressions for the Hessenberg matrix coefficients in equations (A.11) and (A.12). In particular,

$$\begin{aligned} \frac{1}{\eta_{j+1}}\mathbf{v}_{j+1} &= A\mathbf{v}_j - \delta_j\mathbf{v}_j - \eta_j\mathbf{v}_{j-1} \\ &= p_{j+1}(A)\mathbf{v}_1, \end{aligned}$$

where we define $p_{-1}(A) = 0, p_0(A) = I$. Note that it is correct to assume that a polynomial p_{j+1} exists such that the above holds, as the Lanczos vectors form a basis for $\mathcal{K}_m$. This gives

$$\begin{aligned} \eta_{j+1}p_{j+1}(A)\mathbf{v}_1 &= A\mathbf{v}_j - \delta_j\mathbf{v}_j - \eta_j\mathbf{v}_{j-1}, \\ &= \left(Ap_j(A) - \delta_j p_j(A) - \eta_j p_{j-1}(A)\right)\mathbf{v}_1, \end{aligned}$$

and thus we get the following recurrence relation between successive p_j

$$p_{j+1}(A) = \frac{1}{\eta_{j+1}}\left((A - \delta_j)p_j(A) - \eta_j p_j(A)\right). \tag{2.20}$$

Furthermore, we have the following relation between the polynomials r_j and p_j [20, Section 3.2]

$$r_j(A) = (I - Aq_{j-1}(A))\mathbf{r}_0 = \frac{p_j(A)}{p_j(0)}\mathbf{r}_0. \tag{2.21}$$

Using equations (2.20) and (2.21) we can construct the r_j for all iterations $j = 0, 1, \ldots, m$ from the initial residual $\mathbf{r}_0$. The result of this process is shown in figure 2.2 for a system with $n = 10$, $A = B + B^T + nI_n$, $B \in \mathbb{R}^{n\times n}$ and random load vector $\mathbf{b} \in \mathbb{R}^n$. The CG method is run until a relative error tolerance of $\epsilon = 10^{-6}$ is attained. This is ensured by calculating κ and using the criterion from Theorem 2.6.

The coefficients c_i of the solution polynomial q_j in equation (2.5) can also be calculated from r_j for $j = 0, 1, \ldots, m$. To this end we suppose a function that extracts the coefficients of a polynomial p exists.

Definition 2.7: Coefficient extraction function

Let $p(t) = \sum_{i=0}^{m} c_i t^i$ be a polynomial of degree m. Then, the function coeff$(p; i)$ extracts the i^{th} coefficient of p such that coeff$(p; i) = c_i$.

Now using equation (2.21), we can write the solution polynomial as

$$\begin{gathered} Aq_{m-1}(A) = I - r_m(A) \\ r_m(\mathbf{0}) = I \implies A\sum_{i=1}^{m-1} c_{i-1}A^i = -\sum_{i=1}^{m} \text{coeff}(r_m; i)A^i, \end{gathered}$$

which implies

$$c_i = -\text{coeff}(r_m; i+1), \quad i = 0, 1, \ldots, m-1. \tag{2.22}$$

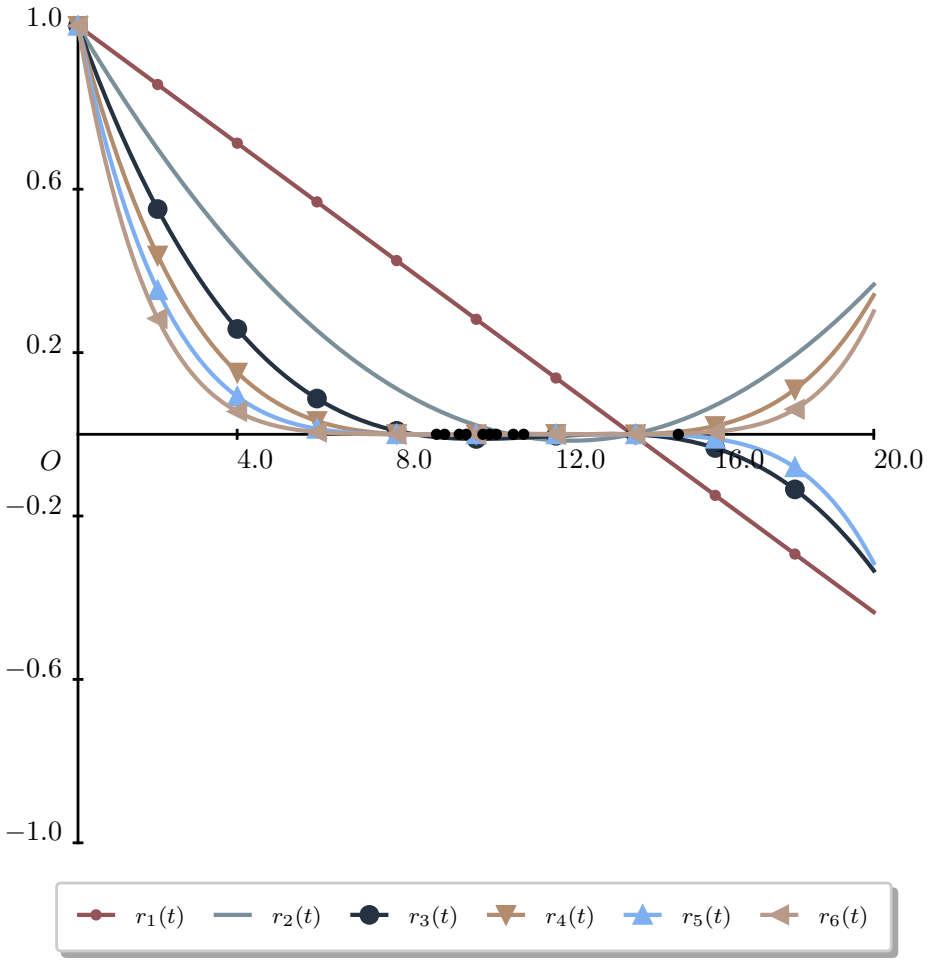

Figure 2.2: Residual polynomials resulting from successive CG iterations

The behavior of the residual polynomials is crucial for understanding the convergence properties of the CG method. In particular, the distribution of the eigenvalues of A significantly affects the convergence rate, as illustrated in figure 2.3.

We consider a system $A\mathbf{u}_{\text{test}} = \mathbf{b}_{\text{random}}$ with a known solution $\mathbf{u}_{\text{test}}$ and random load vector $\mathbf{b}_{\text{random}}$. The system size $n = 360$ is kept relatively small and the system matrix A is chosen to equal diagonal matrix D, making it numerically trivial to determine the exact solution $\mathbf{u}_{\text{test}} = D^{-1}\mathbf{b}_{\text{random}}$. For all plots the lowest and highest eigenvalue in figure 2.3 are $\lambda_{\text{min}} = 0.1$, $\lambda_{\text{max}} = 0.9$ such that $\kappa = 9$ and $f = \dfrac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} = \dfrac{1}{2}$. Furthermore, we use $\mathbf{u}_0 = \mathbf{0}$, set the relative error tolerance $\epsilon = 10^{-6}$ and use the stopping criterion from Definition 2.6 directly, as $\mathbf{e}_0 = \mathbf{u}_{\text{test}} - \mathbf{u}_0$ is known. Alternatively, we could have used the residual stopping criterion from Theorem 2.6, as κ is explicitly known in this case. This results in an overall iteration bound from equation (2.17) of

$$m \leq \left\lceil \log_f \left(\frac{10^{-6}}{2} \right) \right\rceil = 21.$$

Hence, the number of iterations required for convergence depends on the specific clustering of the eigenvalues, as pointed out for example in [17, Section 2.3].

Based on behavior exhibited in figure 2.3 and from Theorem 2.2, we can reason what the best and worst possible spectra for CG convergence are. That is, the best possible spectrum is one where eigenvalues are tightly clustered around distinct values, while the worst possible spectrum is one where the eigenvalues are evenly distributed across the whole range of the spectrum. This is illustrated in figure 2.4.

The first row in figure 2.4 shows an instance of the super-linear convergence that CG can exhibit, particularly when the eigenvalues are closely clustered. This is characteristic of CG is further touched upon in chapters 3 and 4.

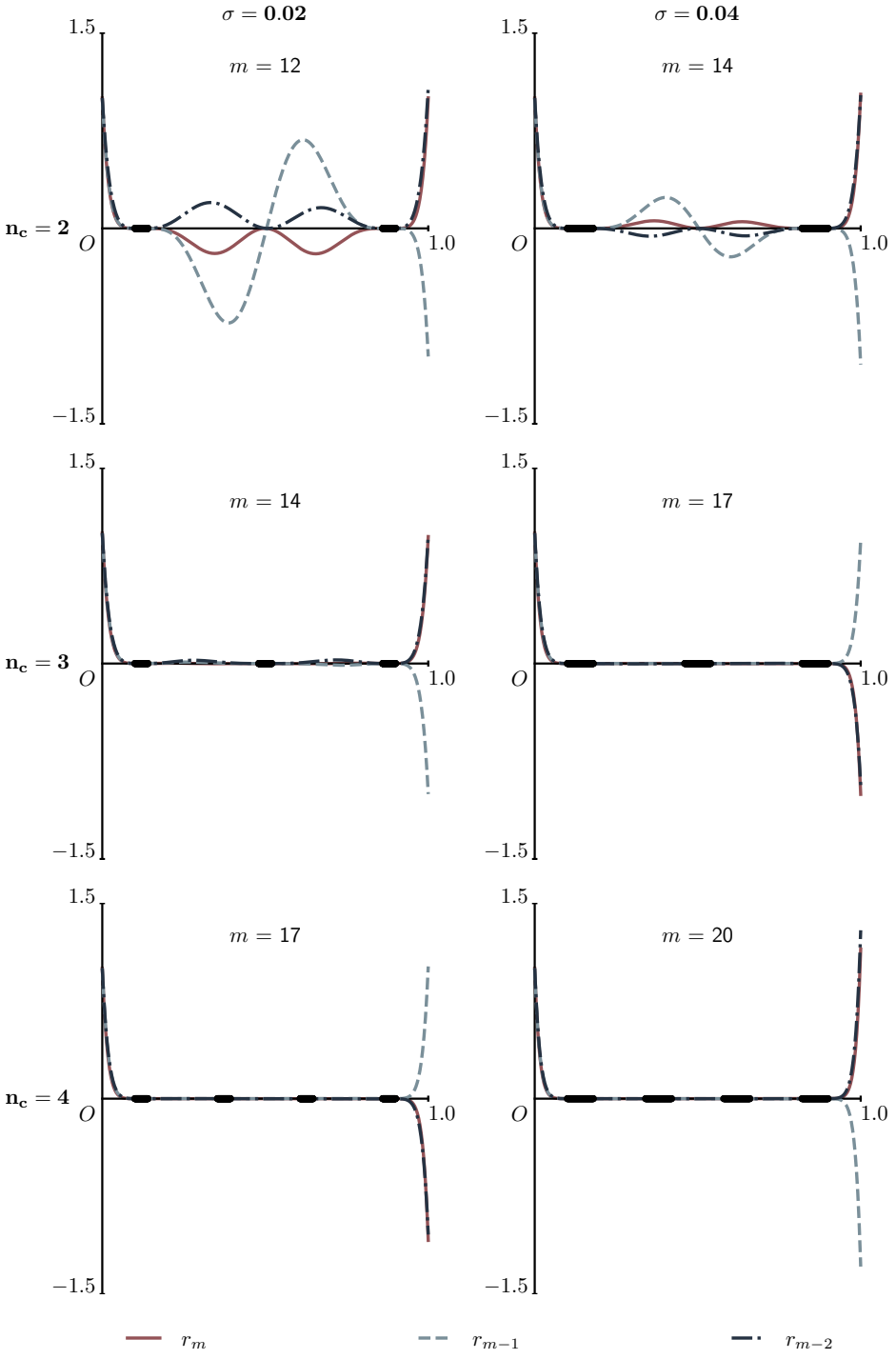

Figure 2.3: Plots of the last three CG residual polynomials for different eigenvalue distributions. n_c indicates the number of clusters and σ is the width of the cluster. The size of the system N and the condition number $\kappa(A)$ are kept constant. m indicates the number of iterations required for convergence.

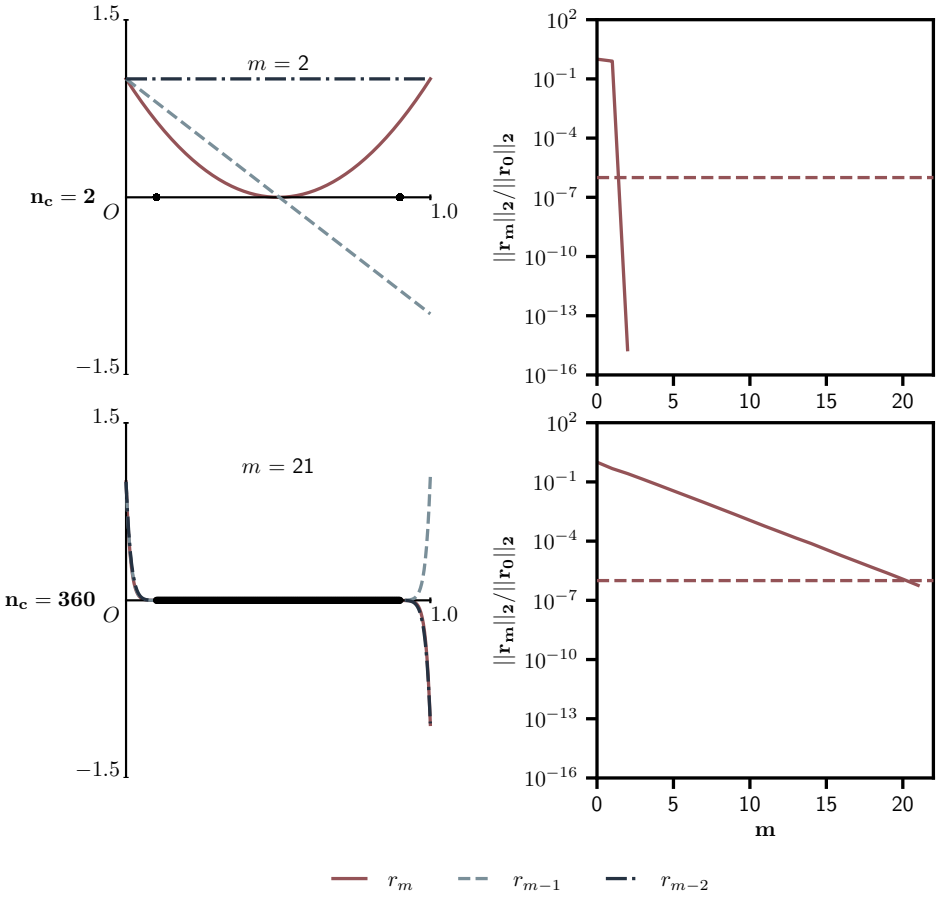

Figure 2.4: Best and worst possible spectra for CG convergence. The top row shows the best possible spectrum, where the eigenvalues are tightly clustered on two distinct values. The bottom row shows the worst possible spectrum, where the eigenvalues are evenly distributed across the whole range of the spectrum. The left column shows the eigenvalue distribution with the residual polynomials corresponding to the iteration. The right column shows the ratio of the last and first residual 2-norms versus the iteration number (solid line), as well as relative error tolerance (dashed line). Convergence is achieved at a relative error tolerance of 10^{-6}. Note that for both spectra $\frac{\|r_m\|_2}{\|r_0\|_2} \leq \frac{10^{-6}}{3}$, in accordance with Theorem 2.6.

2.1.5. Preconditioned CG

Suppose M is some SPD preconditioner, then variants of CG can be derived by applying M to the system of equations. The three main approaches are

PCG-1 left

$$M^{-1}A\mathbf{u} = M^{-1}\mathbf{b}$$

PCG-2 right

$$\begin{aligned} AM^{-1}\mathbf{x} &= M^{-1}\mathbf{b} \\ \mathbf{u} &= M^{-1}\mathbf{x}; \end{aligned}$$

PCG-3 symmetric or split

$$\begin{aligned} M &= LL^T \\ L^{-1}AL^{-T}\mathbf{x} &= L^{-1}\mathbf{b} \\ \mathbf{u} &= L^{-T}\mathbf{x}. \end{aligned}$$

Furthermore, all these variants are mathematically equivalent. Indeed, for the cases **PCG-type 1** and **PCG-type 2**, we can rewrite the CG algorithm using the $M-$ or $M^{-1}-$inner products, respectively. In either case the iterates are the same [21, Section 9.2]. For instance for the left preconditioned CG, we define $\mathbf{z}_j = M^{-1}\mathbf{r}_j$. Note that $M^{-1}A$ is self-adjoint with respect to the $M-$inner product, that is for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ we have

$$(M^{-1}A\mathbf{x}, \mathbf{y})_M = (A\mathbf{x}, \mathbf{y}) = (\mathbf{x}, A\mathbf{y}) = (\mathbf{x}, M^{-1}A\mathbf{y})_M.$$

We use this to get a new expression for α_j. To that end, we write

$$\begin{aligned} 0 &= (\mathbf{r}_{j+1}, \mathbf{r}_j)_M \\ &= (\mathbf{z}_{j+1}, \mathbf{r}_j) \\ &= (\mathbf{z}_j - \alpha_j M^{-1}A\mathbf{p}_j, M^{-1}\mathbf{r}_j)_M \\ &= (\mathbf{z}_j, M^{-1}\mathbf{r}_j)_M - \alpha_j (M^{-1}A\mathbf{p}_j, M^{-1}\mathbf{r}_j)_M \\ &= (\mathbf{z}_j, \mathbf{z}_j)_M - \alpha_j (M^{-1}A\mathbf{p}_j, \mathbf{z}_j)_M \end{aligned}$$

and therefore

$$\alpha_j = \frac{(\mathbf{z}_j, \mathbf{z}_j)_M}{(M^{-1}A\mathbf{p}_j, \mathbf{z}_j)_M}.$$

Using $\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j \mathbf{p}_j$ and A-orthogonality of the search directions $\mathbf{p}_j$ with respect to $M-$norm $(A\mathbf{p}_j, \mathbf{p}_k)_M = 0$, we can write

$$\alpha_j = \frac{(\mathbf{z}_j, \mathbf{z}_j)_M}{(M^{-1}A\mathbf{p}_j, \mathbf{p}_j)_M} = (\mathbf{r}_j, \mathbf{z}_j)/(A\mathbf{p}_j, \mathbf{p}_j).$$

Similarly, we can derive the equivalent expression for β_j as

$$\beta_j = \frac{(\mathbf{z}_{j+1}, \mathbf{z}_{j+1})_M}{(\mathbf{z}_j, \mathbf{z}_j)_M} = \frac{(\mathbf{r}_{j+1}, \mathbf{z}_{j+1})}{(\mathbf{r}_j, \mathbf{z}_j)}.$$

This gives the left preconditioned CG algorithm in 3.

Algorithm 3 Left preconditioned CG [21, Algorithm 9.1]

$\mathbf{r}_0 = b - A\mathbf{u}_0$, $z_0 = M^{-1}\mathbf{r}_0$, $p_0 = z_0$, $\beta_0 = 0$
for $j = 0, 1, 2, \ldots, m$ **do**
  $\alpha_j = (\mathbf{r}_j, \mathbf{z}_j)/(A\mathbf{p}_j, \mathbf{p}_j)$
  $\mathbf{u}_{j+1} = \mathbf{u}_j + \alpha_j \mathbf{p}_j$
  $\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j A\mathbf{p}_j$
  $\mathbf{z}_{j+1} = M^{-1}\mathbf{r}_{j+1}$
  $\beta_j = \dfrac{(\mathbf{r}_{j+1}, \mathbf{z}_{j+1})}{(\mathbf{r}_j, \mathbf{z}_j)}$
  $\mathbf{p}_{j+1} = \mathbf{z}_{j+1} + \beta_j \mathbf{p}_j$
end for

Furthermore it can be shown that the iterates of CG applied to the system with **PCG-type 3** results in identical iterates [21, Algorithm 9.2].

2.2. Schwarz methods

The content of this section is largely based on chapters 1, 2, 4 and 5 of [9] about Schwarz methods.

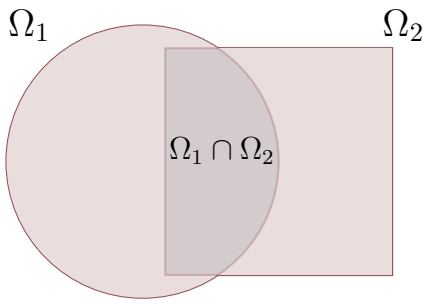

Figure 2.5: Keyhole domain Ω with two overlapping subdomains Ω_1 and Ω_2. The boundary of the keyhole domain is denoted by $\partial\Omega$ and the boundaries of the subdomains are denoted by $\partial\Omega_1$ and $\partial\Omega_2$. The overlapping region is denoted by $\Omega_1 \cap \Omega_2$.

The original Schwarz method was a way of proving that a Poisson problem on some complex domain Ω as in figure 2.5 has a solution.

$$\begin{cases} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{cases} \tag{2.23}$$

Existence of a solution is proved by splitting up the original complex domain in two (or more) simpler, possibly overlapping subdomains and alternatingly solving the Poisson problem on each of these subdomains. The idea is that with enough iterations, the solutions on the subdomains will converge to a solution on the original domain. The method is named after Hermann Schwarz, who first introduced the method in 1869 [22] and has since been extended to more general problems and is now a popular method for solving partial differential equations. Let the alternating Schwarz method be characterized as in Definition 2.8.

Definition 2.8: Alternating Schwarz algorithm

The alternating Schwarz algorithm is an iterative method based on alternately solving subproblems in domains Ω_1 and Ω_2. It updates $(u_1^n, u_2^n) \to \left(u_1^{n+1}, u_2^{n+1}\right)$ by

$$\begin{aligned} -\Delta\left(u_1^{n+1}\right) &= f && \text{in } \Omega_1, \\ u_1^{n+1} &= 0 && \text{on } \partial\Omega_1 \cap \partial\Omega, \\ u_1^{n+1} &= u_2^n && \text{on } \partial\Omega_1 \cap \overline{\Omega_2}; \end{aligned} \quad \text{then} \quad \begin{aligned} -\Delta\left(u_2^{n+1}\right) &= f && \text{in } \Omega_2, \\ u_2^{n+1} &= 0 && \text{on } \partial\Omega_2 \cap \partial\Omega, \\ u_2^{n+1} &= u_1^{n+1} && \text{on } \partial\Omega_2 \cap \overline{\Omega_1}. \end{aligned}$$

The original Schwarz algorithm is sequential and, thereby does not admit parallelization. However, a related algorithm can be parallelized. The Jacobi Schwarz method is a generalization of the original Schwarz method, where the subproblems are solved simultaneously and subsequently combined into a global solution [18]. In order to combine local solutions into one global solution, an extension operator E_i, $i = 1, 2$ is used. It is defined as

$$E_i(v) = v \text{ in } \Omega_i, \quad E_i(v) = 0 \text{ in } \Omega \backslash \Omega_i.$$

Instead of looking for local solutions directly, one can also solve for local corrections stemming from a global residual. This is the additive Schwarz method (ASM). It is defined in algorithm 4.

Algorithm 4 Additive Schwarz method [9, Algorithm 1.2]

Compute residual $r^n = f - \Delta u^n$.
For $i = 1, 2$ solve for a local correction v_i^n:

$$-\Delta v_i^n = r^n \text{ in } \Omega_i, \quad v_i^n = 0 \text{ on } \partial\Omega_i$$

Update the solution: $u^{n+1} = u^n + \sum_{i=1}^{2} E_i(v_i)^n$.

The restricted additive Schwarz method (RAS) is similar to ASM, but differs in the way local corrections are combined to form a global one. In the overlapping region of the domains it employs a weighted average of the local corrections. In particular, a partition of unity χ_i is used. It is defined as

$$\chi_i(x) = \begin{cases} 1, & x \in \Omega_i \setminus \Omega_{3-i}, \\ 0, & x \in \partial\Omega_i \setminus \partial\Omega \\ \alpha, & 0 \leq \alpha \leq 1, x \in \Omega_i \cap \Omega_{3-i} \end{cases}$$

such that for any function $w : \Omega \to \mathbb{R}$, it holds that

$$w = \sum_{i=1}^{2} E_i(\chi_i w_{\Omega_i}).$$

The RAS algorithm is defined in algorithm 5.

Algorithm 5 Restrictive additive Schwarz method [9, Algorithm 1.1]

Compute residual $r^n = f - \Delta u^n$.
For $i = 1, 2$ solve for a local correction v_i^n:

$$-\Delta v_i^n = r^n \text{ in } \Omega_i, \quad v_i^n = 0 \text{ on } \partial\Omega_i$$

Update the solution: $u^{n+1} = u^n + \sum_{i=1}^{2} E_i(\chi_i v_i^n)$.

2.2.1. Schwarz methods as preconditioners

This section is largely based on Section 1.3.2 of [9]. Let $\mathcal{N}_i$ be the set containing all degrees of freedom in the subdomain Ω_i and N_{sub} the number of subdomains such that $n_i = |\mathcal{N}_i|$ is the number of degrees of freedom in the subdomain Ω_i. The global set of degrees of freedom then satisfies

$$\mathcal{N} = \cup_{i=1}^{N_{\text{sub}}} \mathcal{N}_i.$$

Furthermore, let $R_i \in \mathbb{R}^{n_i \times n}$, R_i^T and D_i be the discrete versions of the restriction, extension and partition of unity operators, respectively. We have for $U \in \mathbb{R}^n$

$$U = \sum_{i=1}^{\mathcal{N}_{\text{sub}}} R_i^T D_i R_i U.$$

Note that D_i is a diagonal matrix where the entries are the values of the partition of unity function χ_i evaluated for each degree of freedom. Consider, for instance, a multidimensional FEM problem like Problem 1.3, and let $\mathcal{T}_i$ be a conforming triangulation of the subdomain Ω_i such that

$$\Omega_i = \cup_{\tau \in \mathcal{T}_i} \tau.$$

In this case

$$\mathcal{N}_i = \{k \in \mathcal{N} | \text{meas}(\text{supp}(\phi_k) \cap \Omega_i) > 0\},$$

and we can define

$$\mu_k = |\{j | 1 \leq j \leq N_{\text{sub}} \text{ and } k \in \mathcal{N}_j\}|.$$

Finally, this leads to

$$(D_i)_{kk} = \frac{1}{\mu_k}, \; k \in \mathcal{N}_i. \tag{2.24}$$

Originally the Schwarz method is a fixed point iteration [9, Definitions 1.12 and 1.13]

$$u^{n+1} = u^n + M^{-1} r^n, \; r^n = f - A u^n,$$

where M equals, but is not limited to, one of the following matrices;

$$M_{\text{ASM}}^{-1} = \sum_{i=1}^{N_{\text{sub}}} R_i^T A_i^{-1} R_i, , \tag{2.25a}$$

$$M_{\text{RAS}}^{-1} = \sum_{i=1}^{N_{\text{sub}}} R_i^T D_i A_i^{-1} R_i, \tag{2.25b}$$

where the *local operator* $A_i = R_i A R_i^T$ is the restriction of A to the subdomain Ω_i.

Although the original Schwarz method is not a preconditioner but a fixed-point iteration, the ASM and RAS methods define linear operators that can be applied as **PCG-type 1** and **PCG-type 2** preconditioners in Krylov subspace methods such as CG or GMRES, respectively. M_{ASM}^{-1} is SPD and suitable for CG. M_{RAS}^{-1} may not be symmetric in general and is typically used with Krylov methods like GMRES.

Optimized Schwarz methods and corresponding preconditioners can also be constructed by including more interface conditions (Robin or Neumann) in the subproblems. One such example is the Optimized Restrictive Additive Schwarz method (ORAS) discussed in [9, Chapter 2].

2.2.2. Two-level additive Schwarz method

From the discussion in section B.1 (Motivation) it is clear that the convergence of the Schwarz method not only depends on the extent of the overlap between various subdomains, but on the frequency components k in equation (B.2) as well. That is, low frequency modes experience slower convergence. To overcome this, we can perform a Galerkin projection of the error onto a *coarse space* that is spanned by these low frequency modes. By solving

$$\min_{\beta} ||A(\mathbf{u} + R_0^T \beta) - f||^2,$$

where R_0 is a matrix representing the coarse space. The solution to this problem is

$$\beta = (R_0 A R_0^T)^{-1} R_0 r,$$

where $r = f - A\mathbf{u}$ is the residual and the matrix $A_0 = R_0 A R_0^T$ is called the *coarse operator*.

The coarse space correction term can be added to the one-level Schwarz preconditioners equations (2.25a) and (2.25b) to get the following preconditioners

$$M_{\text{ASM,2}} = R_0^T A_0^{-1} R_0 + \sum_{i=1}^{N_{\text{sub}}} R_i^T A_i^{-1} R_i, \tag{2.26a}$$

$$M_{\text{RAS,2}} = R_0^T A_0^{-1} R_0 + \sum_{i=1}^{N_{\text{sub}}} R_i^T D_i A_i^{-1} R_i. \tag{2.26b}$$

Coarse spaces

The coarse space R_0 can be constructed in various ways. The classical way is called the Nicolaides space [9, Section 4.2], which uses the discrete partition of unity operators D_i as exemplified in equation (2.24) to get

$$R_0 = \sum_{i=1}^{N_{\text{sub}}} R_i^T D_i R_i. \tag{2.27}$$

Note that the coarse space in equation (2.27) has a block-diagonal form, which allows for efficient computation of the coarse operator.

Another way to construct R_0 is based on the analysis of the local errors in the Schwarz method. As seen in section B.1 the local error in any subdomain produced by the original (one-level) Schwarz method satisfies the homogeneous version of the original problem, i.e. right-hand side $f = 0$. At the interface the local error has a Dirichlet boundary condition that equals the error of the neighboring subdomain. Additionally, the convergence factor, e.g. ρ_{2D}, depends on the frequency of the modes in the local error. In particular, small frequencies appear to have slow convergence. The question thus becomes how to get rid of these small frequency modes in the local errors of all subdomains.

One possible answer is the so-called Dirichlet-to-Neumann map [9, Definition 5.1]

Definition 2.9: Dirichlet-to-Neumann map

For any function defined on the interface $u_{\partial\Omega_j} : \partial\Omega_j \mapsto \mathbb{R}$, we consider the Dirichlet-to-Neumann map

$$\mathrm{DtN}_{\Omega_j}\left(u_{\partial\Omega_j}\right) = \left.\frac{\partial v}{\partial \mathbf{n}_j}\right|_{\partial\Omega_j},$$

where $\partial\Omega_j := \partial\Omega_j \backslash \partial\Omega$ and v satisfies

$$\begin{aligned} -\nabla\cdot(\mathcal{C}\nabla v) &= 0 && \text{in } \Omega_j, \\ v &= u_{\partial\Omega_j} && \text{on } \partial\Omega_j, \\ v &= 0 && \text{on } \partial\Omega_j \cap \partial\Omega. \end{aligned} \tag{2.28}$$

The Dirichlet-to-Neumann map essentially solves for an error-like variable v that satisfies the Dirichlet local interface (or global boundary) conditions. DtN then maps the interface condition to the normal derivative of v on the interface, i.e. the Neumann condition. Now, as stated above and illustrated in [9, Figure 5.2], the low frequency modes of the error correspond to those modes that are nearly constant accross an interface, for which the Neumann condition is close to zero. So the problem of tackling slowly convergent modes in the error of the Schwarz method is equivalent to a problem of finding eigenpairs with small eigenvalue of the DtN operator. We can then use these eigenpairs to construct a coarse space.

Hence we aim to solve the eigenvalue problem

$$\mathrm{DtN}_{\Omega_j}(v) = \lambda v,$$

which can be reformulated in the variational form. To that end let $w \in H^1(\Omega_j)$ with $w_{\partial\Omega_j\cap\partial\Omega} \equiv 0$ be a test function. Multiply both sides of equation (2.28) by w, integrate over Ω_j and apply the divergence theorem to get

$$\int_{\Omega_j} \mathcal{C}\nabla v\cdot\nabla w - \int_{\partial\Omega_j} \mathcal{C}\frac{\partial v}{\partial \mathbf{n}_j} w = 0, \quad \forall w.$$

Then, using that $v = u_{\partial\Omega_j}$ on $\partial\Omega_j$ we get that $\dfrac{\partial v}{\partial \mathbf{n}_j} = \mathrm{DtN}(v) = \lambda v$ on $\partial\Omega_j$. This leads to the following variational eigenvalue problem

$$\text{Find } (v,\lambda) \text{ s.t. } \int_{\Omega_j} \mathcal{C}\nabla v\cdot\nabla w - \lambda\int_{\partial\Omega_j} \mathcal{C} v w = 0, \quad \forall w. \tag{2.29}$$

In section B.2 (Construction of two-level additive Schwarz preconditioner) the full construction of the coarse space R_0 is detailed, including the use of the Dirichlet-to-Neumann map to identify and eliminate low-frequency modes in the local errors. Then, in section B.3 (Convergence of two-level additive Schwarz system) the convergence properties of the two-level additive Schwarz method are discussed. By combining and simplifying the equations (B.4), (B.6) and (B.7), we obtain

$$\kappa(M^{-1}_{\mathsf{ASM,2,Nico}}A) \lesssim C\left(\frac{\mathcal{C}_{\mathsf{max}}}{\mathcal{C}_{\mathsf{min}}}\right)\left(C' + \frac{H}{\delta}\right). \tag{2.30a}$$

$$\kappa(M^{-1}_{\mathsf{ASM,2,DtN}}A) \lesssim C\left(\frac{\mathcal{C}_{\mathsf{max}}}{\mathcal{C}_{\mathsf{min}}}\right)\left(C' + \frac{1}{\lambda_c\delta}\right), \tag{2.30b}$$

where δ is the size of the overlap between subdomains, H the typical size of the subdomains, $\mathcal{C}_{\mathsf{max}}$ and $\mathcal{C}_{\mathsf{min}}$ are the maximum and minimum values of the coefficient function $\mathcal{C}$ in equation (2.28), respectively, C and C' are constants that depend on the geometry of the problem and λ_c is related to the cut-off eigenvalue for the eigenpairs of the Dirichlet-to-Neumann map operator. The first bound equation (2.30a) is for the Nicolaides coarse space, while the second bound equation (2.30b) is for the Dirichlet-to-Neumann coarse space.

Notice that the bounds equations (2.30a) and (2.30b) are both robust to the scaling in the fine-mesh size, that is they do not contain any terms depending on h. This is an important feature of the two-level Schwarz method.

Additionally, the bounds in equations (2.30a) and (2.30b) contain the contrast of the coefficient function $\mathcal{C}$, which plays a crucial role in the convergence behavior of the two-level additive Schwarz method, as explained in [13]. In high-contrast problems, where the ratio $\frac{\mathcal{C}_{\text{max}}}{\mathcal{C}_{\text{min}}}$ is large, the condition number of the preconditioned system can become very large, leading to a theoretically predicted slow convergence of the PCG method via equations (1.5) and (2.18).

Fortunately, the bound in equation (2.30b) contains the reciprocal of the cut-off eigenvalue λ_c, which can help mitigate the effects of high contrast, result in a lower condition number and faster convergence of PCG than for the Nicolaides coarse space. In this case, the two-level additive Schwarz method with the Dirichlet-to-Neumann coarse space is defined to be robust with respect to the contrast of the coefficient function $\mathcal{C}$, as the condition number does not depend on the contrast.

However, in most cases it is not possible to select λ_c such that the contrast fully cancels in the condition number bound. As a consequence, the preconditioned system maintains a large condition number and its spectrum contains a spectral gap. The convergence of these kinds of systems is the main concern of this thesis.

3

Related Work

3.1. The spectral gap arising in high-contrast problems

In Problem 1.1, high-contrast $\mathcal{C}(x)$ (e.g., 10^6 in conductive channels vs. 10^{-6} in barriers) means diffusion concentrates in high-permeability regions, while low-permeability zones resist it. This heterogeneity introduces modes (eigenvectors) corresponding to small eigenvalues on the order of the inverse of the high conductivity, i.e. $\mathcal{O}(10^{-6})$. In fact, the number of these eigenmodes in any (sub)domain is equal to the number of connected conductivity regions [12]. High-contrast $\mathcal{C}(x)$ thus splits the eigenspectrum of the stiffness matrix into two parts: a low-frequency and high-frequency part, resulting in a *spectral gap*.

3.2. Techniques for high-contrast problems

The small eigenvalues of the stiffness matrix A in Problem 1.1 lead to a large condition number κ, which in turn leads to slow convergence of the CG method through Theorem 2.7. Modern techniques effectively shrink the size of the spectral gap or try to (partly) remove the small eigenvalues from the spectrum with the aim of reducing κ. Below we discuss the use of multiscale solvers and or construct a coarse space that are *robust to high contrast*.

3.2.1. MsFEM

The multiscale finite element method (MsFEM), as presented in [16], constructs local basis functions spanning V_h and obtained from a homogeneous version of Problem 1.1. These basis functions are used to construct the stiffness matrix as in Problem 1.3. The resulting system is then solved using a multi-grid method. Though Hou and Wu show that MsFEM performs just as well as or better than traditional FEM depending on whether the computational grid can resolve the fine-scale features of $\mathcal{C}$, MsFEM still solves a global problem. MsFEM was originally designed as a discretization method. Its use as an upscaling method only became apparent later.

As we have seen section 2.2.2, we can also construct preconditioners for the purpose of dealing with high-contrast $\mathcal{C}$. That is, we construct the traditional FEM basis functions as in Problem 1.3, decompose the domain into subdomains Ω_i and define coarse and local operators A_0, A_i as in equations (2.26a) and (2.26b). The basis functions constructed using MsFEM can then be used for the construction of the coarse space R_0, akin to how the coarse space is constructed; based on the eigenmodes of the DtN in item **ASM type II coarse space**.

In [11, 13] a separation of scales with conforming fine and coarse triangulations, denoted as $\mathcal{T}_h$ and $\mathcal{T}_H$ is introduced and resulting coarse basis functions are required to satisfy five key assumptions (C1-C5) in [13, Section 2.2]. The coarse bases are constructed by homogeneous version of the system problem equation (1.1) on a coarse grid element $K \in \mathcal{T}_H$.

$$\begin{aligned} -\nabla \cdot (\mathcal{C}\nabla\phi^e_{c,i}) &= 0, && \forall \mathbf{x} \in K, \\ \phi^e_{c,i}(\mathbf{x}) &= \mu^e_i(\mathbf{x}), && \forall \mathbf{x} \in e,\ \forall e \in \partial K, \end{aligned} \tag{3.1}$$

where e is an edge of the coarse grid element K. The boundary conditions, $\mu^e_i(\mathbf{x})$ on edge $e \in K$, for this problem should satisfy conditions M1-M4 [13, Section 4] and are chosen to be either a linear

interpolation of the nodal values of $\mathcal{C}$, denoted as $\mathcal{C}^e$, or a harmonic extension thereof on the coarse grid element K. The latter satisfies

$$\mu_i^e(\mathbf{x}) = \left(\int_e \mathcal{C}^e ds\right)^{-1} \int_e^x \mathcal{C}^e ds, \quad \forall \mathbf{x} \in e, \tag{3.2}$$

and is similar to the oversampling method used in [16]. The restriction operator R_0 is then derived from these basis functions, as given in Equation 2.12 of [13]. The method introduces robustness indicators, $\pi(\alpha)$ and $\gamma(\alpha)$, to quantify the stability of the coarse space and its effectiveness in capturing fine-scale features. Similar to equations (B.4) and (B.7), we get [13, Theorem 3.9]

$$\kappa(M_{\mathsf{ASM},2}^{-1} A) \lessapprox \pi(\mathcal{C})\gamma(1) \max_i^{N_{\mathsf{sub}}} \left(1 + \frac{H_i}{\delta}\right) + \gamma(\mathcal{C}). \tag{3.3}$$

In particular, for linearly interpolated boundary conditions,

$$K(\eta) = \{\mathbf{x} \in K | \mathcal{C}(\mathbf{x}) \geq \eta\},$$

and arbitrary $\eta \geq 1$ the authors obtain [13, Theorem 4.3]

$$\gamma^{\mathsf{L}}(\mathcal{C}) \lessapprox \max_{K \in \mathcal{T}_H} \left\{ \eta \frac{H_K}{\epsilon(\eta, K)} \right\},$$

which does not grow unboundedly with the contrast of $\mathcal{C}$ as long as $\epsilon(\eta, K) = \mathsf{dist}(K(\eta), \delta K) > \frac{3h}{2}$, i.e. as long as $\mathcal{C}$ is *well-behaved* near the boundary of K. However, even for $\max_{\mathbf{x} \in K(\eta)} \to \infty$, i.e. badly behaved $\mathcal{C}$, the authors show that by the choosing oscillatory boundary condition from equation (3.2) for the construction of the coarse bases $\gamma^{\mathsf{Osc}}(\mathcal{C})$ remains bounded. Therefore, for an MsFEM coarse space spanned by coarse basis functions constructed from equation (3.1) with $\mu_i^e(\mathbf{x})$ as in equation (3.2), the condition number bound equation (3.3) is bounded and does not grow unboundedly with the contrast of $\mathcal{C}$. In other words, MsFEM is *robust* to high-contrast $\mathcal{C}$.

3.2.2. ACMS

The approximate component mode synthesis (ACMS) method, detailed in [14], is closely related to MsFEM in that it uses similar fine and coarse scale grids. The coarse problem is decomposed into two components: $u_c = u_I + u_\Gamma$, where u_I and u_Γ represent the interior and interface contribution, respectively. ACMS extends MsFEM by incorporating vertex-specific, edge-specific, and fixed-interface basis functions, where MsFEM corresponds solely to the vertex-specific functions. The vertex-specific basis functions are defined as harmonic extensions of trace values on the interface set Γ. Edge-specific basis functions, on the other hand, arise from an eigenvalue problem defined on an edge e, while fixed-interface basis functions correspond to eigenmodes of an eigenvalue problem within a coarse element T.

ACMS supports two types of coarse spaces, depending on whether Dirichlet (DBC) or Neumann (NBC) boundary conditions are applied. Under DBCs and similar to the coarse space constructed using the DtN in section 2.2.2, MsFEM basis functions are combined with edge-specific basis functions that match on a shared edge e_{ij} between subdomains Ω_i and Ω_j. These functions are constructed from the harmonic extension of eigenmodes defined on the edge e_{ij}, with a scaled bilinear form on the right-hand side. Only eigenmodes corresponding to eigenfrequencies below a set tolerance are retained. With NBCs, both MsFEM and edge-specific basis functions are modified. MsFEM functions remain defined on an edge e_{ij} and satisfy a Kronecker-delta vertex condition but are now obtained via a generalized eigenvalue problem on a slab of width kh, denoted η_{ij}^{kh}. The edge-specific functions are similarly defined through a generalized eigenvalue problem on the slab but without enforcing DBCs. Solving these eigenvalue problems can be made computationally efficient by employing mass matrix lumping techniques.

3.2.3. GDSW

The Generalized Dryja-Smith-Widlund (GDSW) method, introduced by [7] and like MsFEM and ACMS, partitions the computational domain into non-overlapping subdomains and further divides degrees

of freedom (DOFs) into interior and interface nodes. The only required input for the method is a coarse space G. G corresponds to the null space of the problem. In the case of linear elasticity, G spans the linearized rigid body modes, while for the diffusion problem the null space is a constant function. The restriction operators R_Γ and R_I project onto interface and interior DOFs, respectively, with subdomain-specific versions such as R_{Γ_j}.

By ordering the DOFs of into interface Γ and interior I nodes we can get [1, Equation 4, 5]

$$R_0 = \begin{pmatrix} R_I \\ R_\Gamma \end{pmatrix} = \begin{pmatrix} -A_{II}^{-1} A_{I\Gamma} \\ I_{\Gamma\Gamma} \end{pmatrix} R_\Gamma, \tag{3.4}$$

in which R_Γ and $R_I = -A_{II}^{-1} A_{I\Gamma} R_\Gamma$ are the restriction operators to the interface and interior nodes, respectively. Note that equation (3.4) is the discrete version of solving the problem in equation (3.1), known as *discrete harmonic extension*. The coarse solution on the interface set is subsequently defined as

$$u_{0,\Gamma} = \sum_{j=1}^{N_{\text{sub}}} R_{\Gamma_j}^T G_{\Gamma_j} q_j = R_\Gamma q,$$

where q represents the coarse space coefficients. The complete coarse solution is then given by

$$u_0 = R_\Gamma^T u_{0,\Gamma} + R_I^T u_{0,I},$$

RGDSW

The RGDSW method alters the GDSW preconditioner by reducing the coarse space dimension through a partitioning strategy based on nodal equivalence classes that associates each coarse mesh vertex with interface components formed by adjacent edges and faces, distributed among nearby vertices [8]. This reduction in the dimension of the coarse space is achieved without compromising the robustness of the condition number estimate, ensuring that the preconditioner's convergence properties are maintained [15].

3.2.4. AMS

The Algebraic Multiscale Solver (AMS) method, introduced in [19, 24] and further studied in [1] relies on domain decomposition into non-overlapping subdomains, followed by a further subdivision of interface nodes into edge, vertex, and face nodes (in 3D). The method eliminates lower diagonal blocks in the system matrix to facilitate efficient computation. Like (R)GDSW, AMS employs the energy minimization principle to obtain R_I, ensuring an optimal coarse space representation.

3.3. CG convergence in case of non-uniform spectra

The literature in the previous section 3.2 mostly aims to control the condition number of the preconditioned system $M_{\text{ASM}}^{-1} A$, which according to equations (2.14) and (2.17) controls the error and number of iterations in the CG method, respectively. However, we know from section 2.1.4 that the condition number is not the only factor influencing convergence. Excessive work has been done on the convergence rate of the CG method, especially in the context of non-uniform spectra. The following papers provide valuable insights into this topic.

First, in [2] a clever use of Chebyshev polynomials is demonstrated to obtain a sharpened CG iteration bound for the case of two clusters of eigenvalues, i.e. an eigenspectrum with a spectral gap. The authors pose a polynomial that satisfies the minimization problem in Theorem 2.7 and derive an error bound from the maximum of that polynomial on the eigenspectrum of A, a strategy described at the end of section 2.1.3. In sections 4.1 and 4.4 the arguments made in [2] are summarized and extended to the case of multiple clusters of eigenvalues, respectively.

Second, in [23] the convergence rate of CG method is investigated in *finite precision arithmetic* for the family of clusterpoint distributions with parameter $0 \leq \rho \leq 1$

$$\lambda_i = \lambda_{\text{min}} + \frac{i-1}{n-1}(\lambda_{\text{max}} - \lambda_{\text{min}})\rho^{n-i}, \quad i = 1, \ldots, n.$$

The authors show that the convergence rate strongly depends on the eigenvalue distribution of the matrix A. Their experiments reveal that in the case of inexact arithmetic there exists a critical value of ρ

at which the number of iterations required for convergence greatly exceeds the degrees of freedom n, even when the condition number is small $\kappa = 100$. Moreover, this critical value shifts with increasing precision, so that higher precision reduces the impact of rounding errors. The study further shows that the CG behavior is consistent across different algorithm variants (standard CG, SYMMQL, Jacobi acceleration, etc.). These results suggest that certain eigenvalue distributions make CG highly sensitive to numerical errors, and they underline the importance of preconditioners that can modify the eigenvalue distribution to improve CG robustness in practical applications.

Third, in [5] the authors provide a proof of CG's superlinear convergence. This proof is fundamentally different from the polynomial-based strategy that is used in the work of [2], in that this proof is based in the field of *logarithmic potential theory* (LPT). LPT is used to solve the minimization problem in Theorem 2.7 using a strategy that is akin to how charges distribute themselves over a conductor, the interested reader is referred to [10].

In their analysis of superlinear convergence, Beckermann and Kuijlaars use concepts from logarithmic potential theory. Their framework begins by modeling the eigenvalues of a family of matrices not as discrete points, but as a continuous distribution, or measure, σ on a compact set S. The core of their proof recasts the polynomial minimization problem from Theorem 2.5 into an equivalent problem in potential theory.

This involves finding a special probability measure, μ_t, that minimizes the logarithmic energy

$$I(\mu) = \int \int \log \frac{1}{|\lambda - \lambda'|} d\mu(\lambda') d\mu(\lambda)$$

under constraints related to the eigenvalue distribution σ and the normalized iteration count $t = m/n$. The solution to this energy minimization problem identifies a shrinking family of compact sets, $S(t)$, which represents the "effective" spectrum that the CG polynomial must handle at iteration m. As iterations proceed, t increases, and this effective spectral set $S(t)$ becomes smaller.

The resulting asymptotic convergence rate is then described using the Green's function, $g_S(t)$, associated with these shrinking sets [5, Equations 2.22-2.31]. The Green's function is a fundamental tool in potential theory that quantifies how quickly polynomials can decay away from a given set. It is defined in terms of the set's logarithmic capacity, cap$(S(t))$. The logarithmic capacity can be intuitively understood as a measure of the "size" of the set $S(t)$ from the perspective of polynomial approximation; a smaller capacity implies that it is easier to find a polynomial that is small across the entire set. As the effective spectrum $S(t)$ shrinks with the iteration count, its capacity decreases, leading to a sharper error bound and explaining the observed superlinear convergence.

In particular, Beckermann and Kuijlaars derive a new asymptotic error bound that is sharper than the standard estimate. This new bound is expressed via the integral [5, Equation 1.8]:

$$\frac{1}{m} \log \left(\min_{r \in \mathcal{P}_m,\ r(0)=1} \max_{\lambda \in \sigma(A)} |r(\lambda)| \right) \lesssim -\frac{1}{t} \int_0^t g_{S(\tau)}(0) d\tau, \tag{3.5}$$

where $t = \lim_{n \to \infty} \frac{m}{n}$.

As shown by equation (3.5) and Theorem 2.1 in [3], the error in equation (3.5) is bounded by a term that decreases more quickly as the number of iterations increases. Under additional separation conditions among eigenvalues (see Theorem 2.2), the bound is proven to be asymptotically sharp. Moreover, for matrices with equidistant eigenvalues, an explicit formula [5, Corollary 3.2 and Equation 3.11] confirms the improved bound and aligns with observed CG error curves. These findings help to explain why, in practice, CG converges faster than predicted by traditional condition number bounds.

In [3] the authors present a proof of the sharpness of the CG iteration bound in equation (3.5). The paper shows that one cannot beat the asymptotic error estimate bound obtained earlier. It analyzes a strategy in which zeros of the polynomial are set at all eigenvalues outside a chosen set S. The authors prove that any such polynomial cannot yield a better asymptotic error bound than the one given in equation (3.5). Under conditions that are natural for problems arising from discretized PDEs on the eigenvalue distribution, the bound is demonstrated to be sharp, and the discussion includes cases where equality in the bound is reached.

Finally, in [4] the authors extend the results of [5] to cases where the eigenvalue distribution is asymptotically uniform. They show that even when the asymptotic distribution equals an equilibrium distribution, the CG method can exhibit superlinear convergence. In this work, the superlinearity stems

from the particular choice of the right-hand side b. The authors present a family of examples based on the finite difference discretization of the one-dimensional Poisson problem, where they observe superlinear convergence according to the chosen right-hand sides.

4

Methodology

The methods described in this chapter are adapted from the ideas discussed in [2, Section 4]. Therein Axelsson presents a sharpened CG iteration bound for two characteristic eigenspectra $\sigma_{1,2}$. The eigenspectrum consisting of two disjoint clusters σ_1 and the *two-cluster* bound m_2 developed for it in section 4.1 are central to this thesis. Alongside the treatment of the two-cluster eigenspectrum is the analysis of the related *tail-cluster* eigenspectrum. Remarkably, the CG iteration bound for the tail-cluster eigenspectrum is also contained within the bound m_2. In section 4.2 the two-cluster bound is compared to its classical predecessor m_1 from equation (2.18), where a necessary condition is derived for which $m_2 < m_1$. Similarly, in section 4.3 the tail-cluster variant of m_2 is compared to m_1, resulting in another necessary condition for $m_2 < m_1$. The two-cluster bound is then generalized to a multiple-tail-cluster bound in section 4.4. Finally, in section 4.5 the conditions obtained in sections 4.2 and 4.3 and the generalized multiple-tail-cluster bound are combined with an eigenspectrum partitioning algorithm culminating in algorithms 9 and 11, yielding two novel and sharpened CG iteration bounds.

4.1. Two cluster case

On the eigenspectrum of A, consider two intervals $[a, b]$ and $[c, d]$ with $0 < a < b < c < d$ such that all eigenvalues of A are contained in the union of these two intervals. Additionally, we have $\kappa(A) = \frac{d}{a}$. We treat the following two cases simultaneously

$$\sigma_1(A) = [a, b] \bigcup [c, d] \tag{4.1}$$

$$\sigma_2(A) = [c, d] \bigcup_{\substack{i=1 \\ \lambda_i \in [a,b]}}^{N_{\text{tail}}} \lambda_i \tag{4.2}$$

where N_{tail} is the number of eigenvalues in the tail. The first case is a two-cluster eigenspectrum, while the second case has one cluster and a tail of eigenvalues. These characteristic eigenspectra are illustrated in figure 4.1.

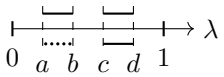

Figure 4.1: Eigenspectrum of A with two clusters (above) and a combination of a tail and cluster (below).

The CG error equation (2.6) suggests we look for a polynomial r_{m_2} of degree m_2 that satisfies the constraints of the minimization problem in order to find an iteration bound $m < m_2$. In other words, we do not solve the minimization problem directly, but we make a clever selection of the polynomial r_{m_2} that satisfies the constraints. As a consequence, the actual minimizing polynomial might require a lower degree m to satisfy the same relative error tolerance ϵ. Therefore, the degree m_2 we find is at least as large as the actual number of iterations required to achieve the desired relative error tolerance ϵ.

Axelsson suggests we use not one monolithic residual polynomial function, but a multiplication of two residual polynomial functions $\hat{r}_p^{(i)}(x)$ and $\hat{r}_{m_2-p}(x)$ for the two clusters. The superscript $^{(i)}$ corresponds to the two eigenspectra described above. The residual polynomial functions are constructed using the (transformed) Chebyshev polynomials from Definitions C.1 and C.2 with $\gamma = 0$ as follows

$$\hat{r}_p^{(i)}(x) \begin{cases} \hat{C}_p & \text{, if } i = 1 \\ \prod_{j=1}^{p} (1 - x/\lambda_j) & \text{, if } i = 2, p = N_{\text{tail}} \end{cases} \tag{4.3}$$

and

$$\hat{r}_{m_2-p}(x) = \hat{C}_{m_2-p}, \tag{4.4}$$

Indeed, the product $r_{m_2} = \hat{r}_p \hat{r}_{m_2-p} \in \mathcal{P}_{m_2}$. Hence, we can use the residual polynomial functions to bound the error at the m^{th} iterate. Now, we obtain the following intermediate bounds

$$\max_{\lambda\in[a,b]} |r_{m_2}(\lambda)| \leq \max_{\lambda\in[a,b]} |\hat{r}_p^{(i)}(\lambda)| \max_{\lambda\in[a,b]} |\hat{r}_{m_2-p}(\lambda)| \leq \max_{\lambda\in[a,b]} |\hat{r}_p^{(i)}(\lambda)|, \text{ and} \tag{4.5a}$$

$$\max_{\lambda\in[c,d]} |r_{m_2}(\lambda)| \leq \max_{\lambda\in[c,d]} |\hat{r}_p^{(i)}(\lambda)| \max_{\lambda\in[c,d]} |\hat{r}_{m_2-p}(\lambda)| \leq \max_{\lambda\in[c,d]} |\hat{r}_p(\lambda)| / C_{m_2-p}\left(\frac{d+c}{d-c}\right) \tag{4.5b}$$

where the equation (4.5a) follows from the fact that $|\hat{r}_{m_2-p}(x)| < 1 \; \forall x \in [a, b]$ and equation (4.5b) from

$$\left| C_{m_2-p}\left(\frac{d+c-2x}{d-c}\right) \right| < 1 \; \forall x \in [c, d].$$

Furthermore, using equation (C.2), we have

$$\frac{1}{C_k\left(\frac{z_1+z_2}{z_1-z_2}\right)} \leq 2\left(\frac{\sqrt{z_2}-\sqrt{z_1}}{\sqrt{z_2}+\sqrt{z_1}}\right)^k, \text{ for } z_1 > z_2 > 0 \text{ and } k \in \mathbb{N}^+, \tag{4.6}$$

and

$$\max_{\lambda\in[a,b]} |\hat{r}_p^{(i)}(\lambda)| \leq \begin{cases} 2\left(\frac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}\right)^p = \eta_1 & \text{, if } i = 1, \\ \left(\frac{b}{a}-1\right)^p = \eta_2 & \text{, if } i = 2, p = N_{\text{tail}}, \end{cases} \tag{4.7}$$

Note that if $i = 1$ we can determine p by requiring that the maximum of the residual polynomial function $\hat{r}_p^{(i)}$ in $[a, b]$ is equal to ϵ. This gives the following equation

$$p \leq \left\lfloor \frac{1}{2}\sqrt{\frac{b}{a}} \ln\frac{2}{\epsilon} + 1 \right\rfloor \tag{4.8}$$

Also note that for $i = 2$ $\hat{r}_p^{(2)}(\lambda) = 0 < \epsilon$ for all eigenvalues $\lambda \in [a, b]$.

Next, $\hat{r}_p^{(i)}$ in $[c, d]$ is bounded by its maximum value within $[a, b]$ multiplied by the polynomial that is the fastest growing polynomial in $\mathcal{P}_p$ outside- and bounded below 1 within $[a, b]$. This polynomial is again the (transformed, but unscaled) Chebyshev polynomial $C_p\left(\frac{2x-b-a}{b-a}\right)$. Therefore,

$$\max_{\lambda\in[c,d]} |\hat{r}_p^{(i)}(\lambda)| \leq \eta_i C_p\left(\frac{2d-b-a}{b+a}\right),$$

with η_i as defined in equation (4.7).

At this point we have ensured that $\max_{\lambda\in[a,b]} |r_{m_2}|$ is bounded by ϵ using equation (4.5a). So it remains to bound $\max_{\lambda\in[c,d]} |r_{m_2}|$ in equation (4.5b). Using above results we can write

$$\max_{\lambda\in[c,d]} |r_{m_2}(\lambda)| < \epsilon,$$

if we require that

$$\eta_i C_p\left(\frac{2d-b-a}{b-a}\right)/C_{m_2-p}\left(\frac{d+c}{d-c}\right) < \epsilon. \tag{4.9}$$

Using that for $x_1, x_2, x_3 \in \mathbb{R}^+$ with $x_1 > x_3$ and $z = \dfrac{x_1 - x_2}{x_3}$

$$\begin{aligned} C_p(z) &\leq \left(z + \sqrt{z^2-1}\right)^p \\ &= \left(\frac{x_1-x_2}{x_3} + \sqrt{\left[\frac{x_1-x_2}{x_3}\right]^2 - 1}\right)^p \\ &\leq \left(\frac{x_1}{x_3} + \sqrt{\left[\frac{x_1}{x_3}\right]^2 - 1}\right)^p \\ &\leq \left(\frac{2x_1}{x_3}\right)^p, \end{aligned}$$

and substituting $x_1 = 2d$, $x_2 = b + a$ and $x_3 = b - a$ we obtain the following inequality

$$\eta_i\left(\frac{4d}{b-a}\right)^p/C_{m_2-p}\left(\frac{d+c}{d-c}\right) < \epsilon. \tag{4.10}$$

Moreover,

$$\begin{aligned} \eta_i\left(\frac{4d}{b-a}\right)^p &= \begin{cases} 2\left(\dfrac{\sqrt{b}-\sqrt{a}}{\sqrt{b}+\sqrt{a}}\dfrac{4d}{b-a}\right)^p & \text{, if } i=1 \\ \left(\dfrac{b-a}{a}\dfrac{4d}{b-a}\right)^p & \text{, if } i=2, \end{cases} \\ &= \begin{cases} 2\left(\dfrac{4d}{b+2\sqrt{ab}+a}\right)^p & \text{, if } i=1 \\ \left(\dfrac{4d}{a}\right)^p & \text{, if } i=2, \end{cases} \\ &\leq 2\begin{cases} \left(\dfrac{4d}{b}\right)^p & \text{, if } i=1 \\ \left(\dfrac{4d}{a}\right)^p & \text{, if } i=2, \end{cases} \end{aligned}$$

We can therefore require that the bound in equation (4.10) is satisfied if we have

$$1/C_{m_2-p}\left(\frac{d+c}{d-c}\right) \leq \frac{\epsilon}{2\left(\frac{4d}{e_i}\right)^p},$$

where

$$e_i = \begin{cases} b & \text{, if } i=1 \\ a & \text{, if } i=2. \end{cases}$$

Again using equation (4.6) and solving for the degree $m_2 - p$ we obtain

$$m_2 - p \leq \frac{1}{2}\sqrt{\frac{d}{c}}\left(\ln\left(\frac{2}{\epsilon}\right) + p\ln\left(\frac{4d}{e_i}\right)\right),$$

which leads to the following bound for the number of iterations [2, Equation 4.4]

$$m_2 = \left\lfloor \frac{1}{2}\sqrt{\frac{d}{c}}\ln(2/\epsilon) + \left(1 + \frac{1}{2}\sqrt{\frac{d}{c}}\ln(4d/e_i)\right)p \right\rfloor, \tag{4.11}$$

where

$$1 \leq p \leq \min\left(\left\lfloor \frac{1}{2}\sqrt{\frac{b}{a}}\ln\frac{2}{\epsilon} + 1 \right\rfloor, N_{\text{tail}}\right).$$

4.2. Performance ratio of the two-cluster bound

In this section we assume that we are dealing with an eigenspectrum of the form $\sigma_1(A)$, i.e. we are only treating case 1. For this case, we compare the new bound in equation (4.11) to the (approximated) classical bound in equation (2.18). We will see that the new bound is not absolutely sharper than the classical one. However, we will derive an approximate, though accurate, criterion (see equation (4.22)) that allows us to discern under what conditions the new bound *is* sharper.

To that end, we restate equation (2.18) here for easy reference

$$m_1(\kappa) = \left\lfloor \frac{\sqrt{\kappa}}{2} \ln\left(\frac{2}{\epsilon}\right) + 1 \right\rfloor.$$

Note that ϵ is generally a predetermined constant. Therefore, we leave it out as an argument of m_1 and m_2. We also rewrite the bound from equation (4.11) in terms of the left and right cluster condition numbers $\kappa_l = \frac{b}{a} > 1$ and $\kappa_r = \frac{d}{b} > 1$ as

$$m_2(\kappa, \kappa_l, \kappa_r) = \left\lfloor \frac{\sqrt{\kappa_r}}{2} \ln(2/\epsilon) + \left(1 + \frac{\sqrt{\kappa_r}}{2} \ln\left(\frac{4\kappa}{\kappa_l}\right)\right) p \right\rfloor.$$

Then, substituting p gives

$$m_2(\kappa, \kappa_l, \kappa_r) = \left\lfloor 1 + \frac{\sqrt{\kappa_r}}{2} \ln\left(\frac{4\kappa}{\kappa_l}\right) + \frac{1}{2} \ln\left(\frac{2}{\epsilon}\right) \left(\sqrt{\kappa_l} + \sqrt{\kappa_r} + \frac{\sqrt{\kappa_l \kappa_r}}{2} \ln\left(\frac{4\kappa}{\kappa_l}\right)\right) \right\rfloor. \tag{4.12}$$

Next we introduce a measure of performance as the ratio of the number of iterations predicted by the classical bound to that predicted by the sharpened bound

$$P = \frac{m_1}{m_2}. \tag{4.13}$$

Consequently, the goal of this section is two-fold; determine the minimum value of P and the conditions on $\kappa, \kappa_l, \kappa_r$ for which $P > 1$.

4.2.1. Uniform spectrum performance

In order to see that the new bound is not absolutely sharper than the classical one, we determine the minimum value of P. We know that the product of two lower order Chebyshev polynomials is not optimal for a uniform eigenspectrum, cf. the proof of Chebyshev optimality outlined in Theorem C.1. Therefore, we assume that the minimum value of P is attained for the case of a uniform eigenspectrum, i.e. $a < b = c < d$, and we can set $\kappa = \kappa_l \kappa_r$, which yields

$$m_2(\kappa = \kappa_l\kappa_r, \kappa_l, \kappa_r) = \left\lfloor 1 + \frac{\sqrt{\kappa_r}}{2} \ln(4\kappa_r) + \underbrace{\frac{\sqrt{\kappa}}{2} \ln\left(\frac{2}{\epsilon}\right)}_{=m_1(\kappa_l\kappa_r)-1} \left(\underbrace{\frac{1}{\sqrt{\kappa_l}} + \frac{1}{\sqrt{\kappa_r}} + \frac{\ln(4\kappa_r)}{2}}_{:=q(\kappa_l,\kappa_r)} \right) \right\rfloor. \tag{4.14}$$

We recognize the classical CG bound as the factor in front of the last term in equation (4.14). Now, the performance ratio P in equation (4.13) satisfies

$$P(\kappa = \kappa_l\kappa_r, \kappa_l, \kappa_r) = P_{\text{uniform}}(\kappa_l, \kappa_r) = \left(\frac{1 + \frac{1}{2}\sqrt{\kappa_r} \ln(4\kappa_r)}{m_1(\kappa_l\kappa_r)} + \frac{m_1(\kappa_l\kappa_r) - 1}{m_1(\kappa_l\kappa_r)} q(\kappa_l, \kappa_r) \right)^{-1}, \tag{4.15}$$

which can be expanded as

$$P_{\text{uniform}}(\kappa_l, \kappa_r) = \frac{m_1(\kappa_l\kappa_r) - 1}{m_1(\kappa_l\kappa_r)} \left[q(\kappa_l, \kappa_r)^{-1} - \frac{1 + \frac{1}{2}\sqrt{\kappa_r} \ln(4\kappa_r)}{(m_1(\kappa_l\kappa_r) - 1) q(\kappa_l, \kappa_r)^2} + \mathcal{O}\left(\frac{1}{(m_1(\kappa_l\kappa_r) - 1)^2 q(\kappa_l, \kappa_r)^3} \right) \right].$$

Next, we require that $\kappa_r \gtrsim 5$, for which it holds that $\frac{1}{\sqrt{\kappa_l}} + \frac{1}{\sqrt{\kappa_r}} < \frac{\ln(4\kappa_r)}{2}$, $\forall \kappa_l \geq 1$. This requirement on κ_r ensures that we can expand $q(\kappa_l, \kappa_r)^{-1}$ as follows

$$q(\kappa_l, \kappa_r)^{-1} = \frac{1}{\ln(4\kappa_r)} - \frac{1}{\ln(4\kappa_r)^2} \left(\frac{1}{\sqrt{\kappa_l}} + \frac{1}{\sqrt{\kappa_r}} \right) + \mathcal{O}\left(\frac{1}{(\ln(4\kappa_r))^3} \left(\frac{1}{\sqrt{\kappa_l}} + \frac{1}{\sqrt{\kappa_r}} \right)^2 \right),$$

which gives the performance for a uniform eigenspectrum as

$$\begin{aligned} P_{\text{uniform}}(\kappa_l, \kappa_r) \approx \frac{1}{\ln(4\kappa_r)} \left(1 - \frac{1}{\sqrt{\kappa_l} \ln\left(\frac{2}{\epsilon}\right)}\right) \\ - \frac{1}{\ln(4\kappa_r)^2} \left(\frac{1}{\sqrt{\kappa_l}} + \frac{1}{\sqrt{\kappa_r}} + \frac{1}{\sqrt{\kappa_l \kappa_r} \ln\left(\frac{2}{\epsilon}\right)}\right) \\ + \mathcal{O}\left(\frac{1}{\ln(4\kappa_r)} \left[\frac{1}{\ln(4\kappa_r)^2} + \frac{1}{\kappa_l \kappa_r \ln\left(\frac{2}{\epsilon}\right)^2}\right]\right). \end{aligned} \tag{4.16}$$

The approximate equality in equation (4.16) stems from the fact that

$$\frac{m_1(\kappa_l \kappa_r) - 1}{m_1(\kappa_l \kappa_r)} \underset{\kappa_l \kappa_r \geq 5}{\approx} 1.$$

Equation 4.16 shows that the uniform (minimum) performance $P_{\text{uniform}}(\kappa_l, \kappa_r)$ tends in its leading order term to $1/\ln(4\kappa_r)$ as $\kappa_l \to \infty$. That is, let $P^{(i)}_{\text{uniform}}$ denote the i-th order expansion of equation (4.16), then

$$P_{\text{uniform}}(\kappa_l, \kappa_r) \lesssim \frac{1}{\ln(4\kappa_r)} = P^{(0)}_{\text{uniform}}(\kappa_r).$$

Only for small $\kappa_l = \dfrac{\kappa}{\kappa_r}$ does the first order term become significant. Additionally, equation (4.16) shows that for increasing κ_r we expect a decreasing minimum performance. We also find that the new bound $m_2(\kappa, \kappa_l, \kappa_r)$ is not absolutely sharper than the classical bound $m_1(\kappa)$. In fact, based on the terms in the expansion of equation (4.16) we can say that

$$P^{(1)}_{\text{uniform}}(\kappa_l, \kappa_r) \lesssim P_{\text{uniform}}(\kappa_l, \kappa_r) \lesssim P^{(0)}_{\text{uniform}}(\kappa_r) < 1, \tag{4.17}$$

4.2.2. Performance threshold

Now that we have established that the new bound is not absolutely sharper than the classical one, we can determine the conditions under which the new bound is sharper. We know that for $\kappa = \kappa_r \kappa_l$ the performance ratio P is given by equation (4.15). We now solve the inverse problem, i.e. for what κ is $P \geq 1$? This gives the following inequality

$$\frac{m_1}{m_2} \geq 1 \Rightarrow m_1(\kappa) \geq m_2(\kappa, \kappa_l, \kappa_r),$$

which can be rewritten as

$$\sqrt{\frac{\kappa}{\kappa_l \kappa_r}} \geq \frac{1}{2} \ln\left(\frac{4\kappa}{\kappa_l}\right) + \frac{1}{\sqrt{\kappa_r}} + \frac{1}{\sqrt{\kappa_l}} \left(1 + \log_{\frac{2}{\epsilon}}\left(\frac{4\kappa}{\kappa_l}\right)\right). \tag{4.18}$$

At this point we neglect the last term in equation (4.18). Doing so relaxes the constraint on the performance ratio as

$$P \geq 1 - \mathcal{O}\left(\sqrt{\frac{\kappa_r}{\kappa_l}} \log_{\frac{2}{\epsilon}}\left(\frac{4\kappa}{\kappa_l}\right)\right) = P_{\text{threshold}}, \tag{4.19}$$

which is a good approximation as long as $\kappa_l \geq \kappa_r \gg 1$. Next to simplifying equation (4.18), this reduces the number of variables. Indeed, by introducing a new parameter for the *spectral width* $s = \dfrac{\kappa}{\kappa_l}$, dropping the last term in equation (4.18) and setting $c_r = \dfrac{2}{\sqrt{\kappa_r}}$ we can write

$$c_r \sqrt{s} \geq \ln(4s) + c_r. \tag{4.20}$$

To solve the equality in equation (4.20) for s, we make use of the Lambert W function. The Lambert W function is defined as the solution to the equation $x = y\mathrm{e}^y$. For a real-valued argument $z \in [-1/e, 0)$, the equation has two real solutions, given by the branches $y = W_0(z) > -1$ and $y = W_{-1}(z) \leq -1$. In order

to solve equation (4.20), we transform the equality into the form $x = ye^y$ with $y = -\frac{c_r\sqrt{s}}{2} = -\sqrt{\frac{\kappa}{\kappa_l\kappa_r}} < -1$ and $x = -\frac{c_r}{4\exp\left(\frac{c_r}{2}\right)}$. Then, we can write

$$-\frac{c_r\sqrt{s}}{2} = y = W_{-1}(x) = W_{-1}\left(-\frac{c_r}{4\exp\left(\frac{c_r}{2}\right)}\right), \quad x \in \left(-\frac{1}{\mathrm{e}}, 0\right],$$

where W_{-1} is the first negative branch of the Lambert W function. The condition on x ensures that W_{-1} evaluates to a real number. Note that $\kappa_r \geq 1 \implies c_r \leq 1$. So the condition on x is indeed satisfied. Finally, after substituting $c_r = \frac{2}{\sqrt{\kappa_r}}$ we obtain an explicit expression for the spectral width s in terms of the right cluster condition number κ_r

$$s(\kappa, \kappa_l) \geq \kappa_r W_{-1}\left(-\frac{1}{2\sqrt{\kappa_r}\exp\left(\frac{1}{\sqrt{\kappa_r}}\right)}\right)^2,$$

or, in terms of the original condition numbers

$$\kappa \geq 4\kappa_l\kappa_r W_{-1}\left(-\frac{1}{2\sqrt{\kappa_r}\exp\left(\frac{1}{\sqrt{\kappa_r}}\right)}\right)^2 = T_\kappa(\kappa_l, \kappa_r). \tag{4.21}$$

The evaluation of the Lambert W function is not a trivial task and often requires numerical methods for accurate computation [6]. Luckily, there exists an expansion of $W_{-1}(x)$ for $x \to 0^-$ [6, Equation 4.19]. Let x be as above and set $L = \ln(-x)$ and $l = \ln(-L)$, then

$$\kappa \geq 4\kappa_l\kappa_r\left(L - l + \frac{l}{L}\right)^2 + \mathcal{O}\left(\frac{\kappa_l\kappa_r l^4}{L^4}\right) := T_\kappa^{(0)}(\kappa_l, \kappa_r) + \mathcal{O}\left(\frac{\kappa_l\kappa_r l^4}{L^4}\right). \tag{4.22}$$

Combining the results of this section with those of section 4.2.1 we can now say that as long as $\kappa_l\kappa_r \leq \kappa \leq T_\kappa(\kappa_l, \kappa_r)$, the performance ratio satisfies

$$P_{\text{uniform}}(\kappa_l, \kappa_r) \leq P(\kappa, \kappa_l, \kappa_r) \leq P_{\text{threshold}} \lesssim 1, \tag{4.23}$$

and, conversely, if $\kappa > T_\kappa(\kappa_l, \kappa_r)$, then

$$P(\kappa, \kappa_l, \kappa_r) > P_{\text{threshold}}$$

Figure 4.2 visualizes all the findings regarding the performance of the two-cluster bound m_2.

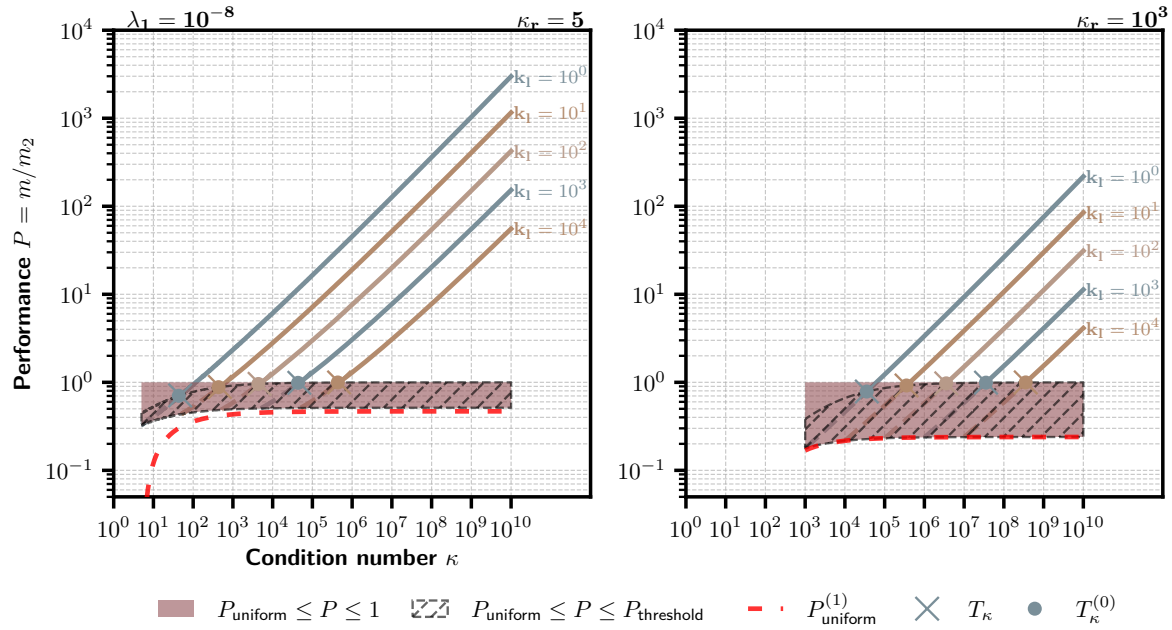

Figure 4.2: Performance ratio of the classical bound m_1 from equation (2.18) and the two-cluster bound m_2 from equation (4.11) as a function of the global condition number κ for right cluster condition number $\kappa_r = 5$ (**left**) and $\kappa_r = 10^3$ (**right**). All plots contain graphs for spectra with $\kappa_l = 1, 10, 10^2, 10^3, 10^4$. The plots also contain a red, shaded region for which $P_{\text{uniform}} < P < 1$, a diagonally hashed region resembling the performance bounds from equation (4.23), a red, dashed line resembling the first order expansion of the uniform performance ratio, $P^{(1)}_{\text{uniform}}$ from equation (4.16), and both the exact and approximate κ-threshold values for each κ_l graph from equations (4.21) and (4.22), respectively.

We can notice that the performance graphs for various left cluster widths κ_l grow with the square root of the global condition number, i.e. $P \approx \mathcal{O}(\sqrt{\kappa})$, as the slope of these lines is approximately $1/2$. This is to be expected from equation (4.12), since for constant κ_l, κ_r, and $\kappa > T_h(\kappa_l, \kappa_r) \gg 1$ we have that $P \sim \dfrac{\sqrt{\kappa}}{\ln(4\kappa)}$. The slope of the term $\dfrac{\sqrt{\kappa}}{\ln(4\kappa)}$ in a log-log plot satisfies

$$\frac{d\ln(P)}{d\ln(\kappa)} = \kappa \frac{d\ln(P)}{d\kappa} \sim \kappa \frac{d}{d\kappa}\left[\ln(\sqrt{\kappa}) - \ln(\ln(4\kappa))\right] = \frac{1}{2} - \frac{1}{\ln(4\kappa)} \underset{\kappa\to\infty}{\longrightarrow} \frac{1}{2}.$$

Next to this we can see that the first order expansion of the uniform performance ratio $P^{(1)}_{\text{uniform}}$ agrees well with the minimum performance bound as long as $\kappa = \kappa_l\kappa_r \gg 1$, which is in agreement with the error terms in equation (4.16) and the minimum performance bound from equation (4.23). Finally, we see that both $T_\kappa(\kappa_l, \kappa_r)$ and its zeroth order expansion $T^{(0)}_\kappa(\kappa_l, \kappa_r)$ from equations (4.21) and (4.22) are accurate approximations of the actual κ-threshold value at which $P = 1$, and even more so for larger values of κ, κ_r. We do see a deviation of both $T_\kappa(\kappa_l, \kappa_r)$ and $T^{(0)}_\kappa(\kappa_l, \kappa_r)$ for $\kappa_l \leq \kappa_r$. Again, this is in agreement with the error terms in equation (4.19).

4.3. Performance of the tail-cluster bound

The two-cluster bound from equation (4.11) is also derived for the spectrum σ_2, i.e. the case of a right cluster with a tail of eigenvalues to the left of it. In this case we can also derive a condition for which $m_2 < m_1$, or $P > 1$. The tail-cluster bound is given by equation (4.11) with $p = N_{\text{tail}} \leq \left\lfloor \dfrac{\sqrt{\kappa_l}}{2} \ln\dfrac{2}{\epsilon} + 1 \right\rfloor$. This gives

$$m_2(\kappa, \kappa_l, \kappa_r, p) = \left\lfloor \frac{\sqrt{\kappa_r}}{2} \ln\left(\frac{2}{\epsilon}\right) + p\left(1 + \frac{\sqrt{\kappa_r}}{2} \ln\left(\frac{4\kappa}{\kappa_l}\right)\right) \right\rfloor.$$

The performance ratio is now given by

$$P_{\text{tail-cluster}}(\kappa, \kappa_l, \kappa_r, p) = \frac{\sqrt{\frac{\kappa}{\kappa_r}} \ln\left(\frac{2}{\epsilon}\right)}{p \ln\left(\frac{4\kappa}{\kappa_l}\right)} - \mathcal{O}\left(\sqrt{\frac{\kappa}{\kappa_r}} \frac{\ln\left(\frac{2}{\epsilon}\right) + \frac{2p}{\sqrt{\kappa_r}}}{p^2 \ln\left(\frac{4\kappa}{\kappa_l}\right)^2}\right), \quad \text{for } p \geq \left\lceil \log_{\frac{4\kappa}{\kappa_l}}\left(\frac{2}{\epsilon}\right)\right\rceil. \tag{4.24}$$

Equation (4.24) shows that

$$P_{\text{tail-cluster}}(\kappa, \kappa_l, \kappa_r, p) \longrightarrow \frac{1}{\ln(4\kappa_r)} \approx P^{(0)}_{\text{uniform}}(\kappa_r) \text{ as } p \to \left\lfloor \frac{\sqrt{\kappa_l}}{2} \ln\frac{2}{\epsilon} + 1 \right\rfloor \text{ and } \kappa \to \kappa_l \kappa_r,$$

that is, the minimum performance of the tail-cluster bound reduces to the leading order term of the two-cluster bound's minimum performance from equation (4.16) as p approaches its maximum value. Another crucial aspect about $P_{\text{tail-cluster}}(\kappa, \kappa_l, \kappa_r, p)$ is recovered when we require its leading order term to be larger than 1, giving us the following inequality

$$p \leq \left\lfloor \sqrt{\frac{\kappa}{\kappa_r}} \log_{\frac{4\kappa}{\kappa_l}}\left(\frac{2}{\epsilon}\right) \right\rfloor. \tag{4.25}$$

Equation (4.25) can be interpreted as a *sparsity condition* on the tail cluster, i.e. the tail cluster must be sparse enough to ensure that the performance ratio is larger than 1.

Finally, we found that the tail-cluster bound is sharper than the classical bound if the following conditions on p hold

$$\left\lceil \log_{\frac{4\kappa}{\kappa_l}}\left(\frac{2}{\epsilon}\right) \right\rceil \leq p \leq \left\lfloor \sqrt{\frac{\kappa}{\kappa_r}} \log_{\frac{4\kappa}{\kappa_l}}\left(\frac{2}{\epsilon}\right) \right\rfloor. \tag{4.26}$$

The left-hand side inequality of equation (4.26) ensures that the expansion made in equation (4.24) is valid and can be interpreted as a minimum *density condition* on p. Notice that for $\epsilon = 10^{-8}$, $\kappa \approx 10^8$ and $\kappa_l \approx 10$ this left-hand side of equation (4.26) is approximately equal to 1, in which case we can neglect it as a condition on p as $p \geq 1$ is always satisfied. We will reconsider the validity of this simplification in chapter 6, but for now we say that the tail-cluster bound is sharper than the classical bound if equation (4.25) holds.

4.4. Generalization to multiple clusters

The technique outlined in section 4.1 starts at the left most cluster $[a, b]$, finds the Chebyshev degree $p_1 = p$ satisfying equation (4.8), moves to the neighboring cluster $[c, d]$ and finds the Chebyshev degree $p_2 = m_2 - p$ satisfying equation (4.9). Rewriting equation (4.9) gives the following equation for p_2:

$$\frac{1}{C_{p_2}\left(\frac{d+c}{d-c}\right)} \leq \frac{\epsilon}{\hat{C}^{(1)}_{p_1}(d)} = \epsilon_2, \tag{4.27}$$

where $C^{(1)}_{p_1}(x)$ is the (transformed) Chebyshev polynomial from Definition C.2 corresponding to the first cluster.

Suppose there is a third cluster to the right of $[c, d]$, i.e. $[e, f]$. We can repeat the process and find the Chebyshev degree p_3 satisfying a similar equation as equation (4.27) for the third cluster.

$$\frac{1}{C_{p_3}\left(\frac{f+e}{f-e}\right)} \leq \frac{\epsilon}{\hat{C}^{(1)}_{p_1}(f)\hat{C}^{(2)}_{p_2}(f)} = \epsilon_3,$$

This leads to the general equation for the Chebyshev degree p_i of the i^{th} cluster $[a_i, b_i]$

$$\frac{1}{C_{p_i}\left(\frac{b_i+a_i}{b_i-a_i}\right)} \leq \frac{\epsilon}{\prod_{j=1}^{i-1} \hat{C}^{(j)}_{p_j}(b_i)} = \epsilon_i. \tag{4.28}$$

An eigenvalue spectrum of this kind is visualized in figure 4.3.

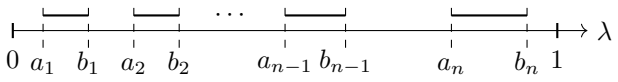

Figure 4.3: Example of an eigenvalue spectrum consisting of multiple clusters.

The Chebyshev polynomials C_p grow rapidly outside the interval $[-1, 1]$ [2, Section 4]. Therefore, it can be cumbersome for a computer to evaluate the product of the $\hat{C}_{p_j}^{(j)}(b_i)$ terms in the denominator of equation (4.28). Instead, we first apply equation (4.6) and introduce the cluster condition numbers $\kappa_i = \frac{b_i}{a_i}$, where i is the index of the cluster. We can then rewrite equation (4.28) using equation (C.2) as follows

$$p_i = \left\lceil \ln \frac{\epsilon_i}{2} / \ln \frac{\sqrt{\kappa_i} - 1}{\sqrt{\kappa_i} + 1} \right\rceil,$$

and

$$\ln \frac{\epsilon_i}{2} = \ln \frac{\epsilon}{2} - \sum_{j=1}^{i-1} \ln \hat{C}_{p_j}^{(j)}(b_i).$$

Let $z_1^{(i,j)} = \frac{b_j + a_j - 2b_i}{b_j - a_j}$ and $z_2^{(j)} = \frac{b_j + a_j}{b_j - a_j}$ then

$$\ln \hat{C}_{p_j}^{(j)}(b_i) = \ln C_{p_j}(z_1^{(i,j)}) - \ln C_{p_j}(z_2^{(j)}).$$

We have, using the approximations in equation (C.2)

$$\ln C_{p_j}(z_1^{(i,j)}) \approx p_j \ln \left| z_1^{(i,j)} - \sqrt{\left(z_1^{(i,j)}\right)^2 - 1} \right| - \ln 2, \tag{4.29}$$

and

$$\ln C_{p_j}(z_2^{(j)}) \approx p_j \ln \left[z_2^{(j)} + \sqrt{\left(z_2^{(j)}\right)^2 - 1} \right] - \ln 2, \tag{4.30}$$

both of which become more accurate approximations as $z, m \to \infty$. Introducing

$$\begin{aligned} \zeta_1^{(i,j)} &= z_1^{(i,j)} - \sqrt{\left(z_1^{(i,j)}\right)^2 - 1}, \\ \zeta_2^{(j)} &= z_2^{(j)} + \sqrt{\left(z_2^{(j)}\right)^2 - 1}, \text{ and} \\ f_i &= \frac{\sqrt{\kappa_i} - 1}{\sqrt{\kappa_i} + 1}, \end{aligned}$$

with κ_i the i^{th} cluster condition number, and substituting the inequalities 4.29 and 4.30 back into the bound for p_i gives

$$\begin{aligned} p_i &\leq \left\lceil \frac{\ln \frac{\epsilon}{2} - \sum_{j=1}^{i-1} p_j \left(\ln \zeta_1^{(i,j)} - \ln \zeta_2^{(j)} \right)}{\ln f_i} \right\rceil \\ &= \left\lceil \log_{f_i} \frac{\epsilon}{2} - \sum_{j=1}^{i-1} p_j \left(\log_{f_i} \zeta_1^{(i,j)} - \log_{f_i} \zeta_2^{(j)} \right) \right\rceil \\ &= \left\lceil \log_{f_i} \frac{\epsilon}{2} - \sum_{j=1}^{i-1} p_j \log_{f_i} \left(\frac{\zeta_1^{(i,j)}}{\zeta_2^{(j)}} \right) \right\rceil \end{aligned}$$

Note that in general $f_i < 1$ and $\zeta_1^{(i,j)} > \zeta_2^{(j)}$. Therefore, the term $\log_{f_i}\left(\frac{\zeta_1^{(i,j)}}{\zeta_2^{(j)}}\right) < 0$. So we multiply this term by -1 and obtain

$$p_i \leq \left\lceil \log_{f_i} \frac{\epsilon}{2} + \sum_{j=1}^{i-1} p_j \log_{f_i}\left(\frac{\zeta_2^{(j)}}{\zeta_1^{(i,j)}}\right) \right\rceil \tag{4.31}$$

Evidently, adding more clusters to the left of the interval $[a_i, b_i]$ increases the degree p_i of the Chebyshev polynomial. Next to this, equation (4.31) reduces to the classical CG iteration bound equation (2.17) for a single cluster when $i = N_{\text{clusters}} = 1$.

Equation 4.31 gives us a way to calculate the Chebyshev degree p_i of the i^{th} cluster $[a_i, b_i]$ in terms of the Chebyshev degrees of the previous clusters. To obtain a bound on the number of iterations for the CG method we sum the Chebyshev degrees of all the clusters

$$m_{N_{\text{clusters}}} = \sum_{i=1}^{N_{\text{clusters}}} p_i \tag{4.32}$$

4.4.1. Multiple tail clusters

We have not yet generalized to an eigenvalue spectrum consisting of multiple tail clusters. Moreover, the most general eigenvalue spectrum can consist of any combination of *regular* and tail clusters, as shown in figure 4.4.

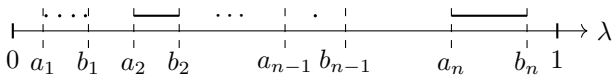

Figure 4.4: Example of the most general eigenvalue spectrum. In order from left to right; a tail cluster $[a_1, b_1]$, a regular cluster $[a_2, b_2]$, a tail cluster $[a_{n-1}, a_{n-1}]$ with a single eigenvalue and another regular cluster $[a_n, b_n]$.

Fortunately, there is a natural way to generalize the CG bound to this case. Let Λ_t be the set of all eigenvalues residing in tail clusters. Now consider a polynomial similar to equation (4.3) for the case $i = 2$, that is

$$r_t(x) = \prod_{\lambda \in \Lambda_t} \left(1 - \frac{x}{\lambda}\right). \tag{4.33}$$

The polynomial $r_t(x)$ is indeed the natural choice for polynomials of the tail clusters. To see this, suppose we were to consider separate polynomials $r_{t_j}(x)$ for each tail cluster j consisting of t_j tail eigenvalues $\lambda_j \in \Lambda_{t_j}$. Then we need $r_{t_j}(\lambda_j) = 0,\ \forall \lambda_j \in \Lambda_{t_j},\ \forall j$. The most natural choice for such a polynomial is

$$r_{t_j}(x) = \prod_{\lambda \in \Lambda_{t_j}} \left(1 - \frac{x}{\lambda}\right). \tag{4.34}$$

Now, consider the global residual polynomial $r(x)$ defined as

$$r(x) = \prod_j r_{t_j}(x) \prod_{i=1}^{N_{\text{clusters}}} \hat{C}_{p_i}^{(i)}(x) = \prod_{\lambda \in \Lambda_t} \left(1 - \frac{x}{\lambda}\right) \prod_{i=1}^{N_{\text{clusters}}} \hat{C}_{p_i}^{(i)}(x) = r_t(x) \prod_{i=1}^{N_{\text{clusters}}} \hat{C}_{p_i}^{(i)}(x), \tag{4.35}$$

where $\Lambda_t = \cup_j \Lambda_{t_j}$. Hence, we recover $r_t(x)$ from equation (4.33) and see that is a natural generalization.

The degree of r_t is equal to the number of eigenvalues in Λ_t, which we denote as N_t. Now N_t, unlike the degrees p_i of the Chebyshev polynomials $C_{p_i}^{(i)}$ corresponding to the clusters, is fixed. Therefore, the contribution of the tail clusters to the CG bound is simply N_t. However, r_t may still influence the degrees p_i through its contribution to the global residual polynomial $r(x)$ defined in equation (4.35). In particular, we rewrite equation (4.31) as

$$p_i \leq \left\lceil \log_{f_i} \frac{\epsilon}{2} + \sum_{j=1}^{i-1} p_j \log_{f_i} \frac{\zeta_2^{(j)}}{\zeta_1^{(i,j)}} - \sum_{\lambda \in \Lambda_t} \log_{f_i} \left|1 - \frac{h_i}{\lambda}\right| \right\rceil, \tag{4.36}$$

where

$$h_i = \arg \max_{x \in [a_i, b_i]} |r_t(x)|,$$

that is, we make the approximation that the maximum of the tail residual polynomial r_t on the i^{th} cluster is attained at one of that cluster's endpoints. For instance, in the case that the i^{th} cluster lies fully to the right of all tail eigenvalues, that is $a_i > \max\{\lambda : \lambda \in \Lambda_t\}$, we have $h_i = b_i$. Finally, we can combine equation (4.31) and equation (4.32) as

$$m_N = N_t + \sum_{i=1}^{N_{\text{clusters}}} p_i. \tag{4.37}$$

4.4.2. Algorithm for generalized CG bound

We summarize the techniques outlined in this section in algorithm 6. The algorithm takes a set of clusters, a possibly empty set of tail eigenvalues Λ_t and a relative error tolerance ϵ as input and returns the generalized multi-tail-cluster CG iteration bound m_N. The algorithm iterates over all clusters, calculating the Chebyshev degree p_i for each cluster based on the previous clusters' degrees and the relative error tolerance. It also treats tail eigenvalues as discussed in section 4.4.1. Finally, it sums up all the Chebyshev degrees and the contribution from the tail eigenvalues to obtain the multi-cluster CG bound.

Algorithm 6 $\text{GeneralizedCGIterationBound}(\text{clusters}, \Lambda_t, \epsilon)$

Input: Sorted set clusters $= \langle [a_1, b_1], [a_2, b_2], \ldots, [a_{N_{\text{clusters}}}, b_{N_{\text{clusters}}}] \rangle$, a set of tail eigenvalues Λ_t and relative error tolerance ϵ
Output: Generalized multi-tail-cluster CG iteration bound m_N
Initialize $N_t \leftarrow |\Lambda_t|$, $P \leftarrow \emptyset$
Construct the residual polynomial $\ln |r_t|(x) = \sum_{\lambda \in \Lambda_t} \ln \left| 1 - \frac{x}{\lambda} \right|$
for $[a_i, b_i] \in$ clusters **do**
    $f_i \leftarrow \frac{\sqrt{\kappa_i} - 1}{\sqrt{\kappa_i} + 1}$, where $\kappa_i = \frac{b_i}{a_i}$
    $\ln |r_t|_{\max} \leftarrow \max\{\ln |r_t|(x) : x \in [a_i, b_i]\}$
    $\epsilon_i \leftarrow \ln(\epsilon) - \ln |r_t|_{\max}$
    for $[a_j, b_j] \in \text{clusters}_{<i}$ **do** ▷ $\text{clusters}_{<i}$ = all clusters to the left of $[a_i, b_i]$
        $z_1 \leftarrow \frac{b_j + a_j - 2b_i}{b_j - a_j}$
        $z_2 \leftarrow \frac{b_j + a_j}{b_j - a_j}$
        $p_j \leftarrow P[j]$
        $\epsilon_i \leftarrow \epsilon_i - p_j \left(\ln \left(z_1 - \sqrt{z_1^2 - 1} \right) - \ln \left(z_2 + \sqrt{z_2^2 - 1} \right) \right)$
    end for
    $p_i \leftarrow \left\lceil \log_{f_i} \left(\frac{\epsilon_i}{2} \right) \right\rceil$
    $P \leftarrow P \cup \langle p_i \rangle$
end for
$m_N \leftarrow N_t + \sum_{i=1}^{N_{\text{clusters}}} P[i]$
return m_N

4.5. Algorithms for sharpened CG iteration bounds

In this section we combine the findings of sections 4.2 to 4.4 to derive two flexible algorithms for determining a sharpened CG bound, algorithms 9 and 11. The main goal of the algorithms in this section is to partition a given eigenspectrum in such a way that the resulting set of clusters and possibly tail eigenvalues can be fed into algorithm 6 to obtain CG iteration bounds $m_{N_{\text{cluster}}}$ and $m_{N_{\text{tail-cluster}}}$. Moreover, to ensure that $m_{N_{\text{cluster}}} < m_1$ and $m_{N_{\text{tail-cluster}}} < m_1$ the partitioning of the eigenspectrum is done such that the resulting clusters satisfy the performance conditions derived in sections 4.2.2 and 4.3, respectively.

4.5.1. Multi-cluster CG iteration bound

We start by exclusively partitioning a spectrum into clusters, that is we do not look for possible tail eigenvalues. Later in section 4.5.2 we will see how to extend the algorithms we derive to the incorporate clusters of tail eigenvalues.

The idea is to use a simple algorithm to split an eigenspectrum into two clusters. Then, we calculate the left and right cluster condition numbers κ_l and κ_r and check if the approximate threshold condition in equation (4.22) is satisfied. In fact, the algorithm recursively applies the previous two steps, stopping only when the threshold condition is not satisfied. The result is a list of indices at which to split the eigenspectrum, yielding an ordered set of clusters $\langle [a_i, b_i] \rangle_{i=1}^{N_{\text{cluster}}}$, where N_{cluster} is the number of clusters. Finally, we can use algorithm 6 to calculate the generalized CG iteration bound for a spectrum consisting of purely clusters, $m_{N_{\text{cluster}}}$.

To that end, suppose we are given a sorted set of eigenvalues $\sigma = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$ with $0 < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. We need to find an *optimal* way of splitting this set in two disjoint sets. In order to do so, we turn to figure 4.2, in which it is shown that the performance of the two-cluster bound m_2 relative to the classical bound m_1 for fixed κ increases both with smaller κ_l and κ_r. Therefore, we expect to get the greatest performance gain from splitting the spectrum between those eigenvalues that share the largest ratio. That is to say, we can always write

$$\kappa = \frac{\lambda_{k^*}}{\lambda_1} \frac{\lambda_{k^*+1}}{\lambda_{k^*}} \frac{\lambda_n}{\lambda_{k^*+1}} = \kappa_l \kappa_m \kappa_r,$$

where $\kappa_l = \dfrac{\lambda_{k^*}}{\lambda_1}$, $\kappa_m = \dfrac{\lambda_{k^*+1}}{\lambda_{k^*}}$ and $\kappa_r = \dfrac{\lambda_n}{\lambda_{k^*+1}}$. The split index k^* must be chosen such that the ratio $\dfrac{\lambda_{k^*+1}}{\lambda_{k^*}}$ is maximized and, simultaneously, κ_l, κ_r are minimized.

Therefore, we choose to split the eigenspectrum at the *largest logarithmic gap* between consecutive eigenvalues, resulting in algorithm 7.

Algorithm 7 $\mathrm{SplitEigenspectrum}(\sigma)$

1: **Input:** Sorted eigenvalues $\sigma = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$ with $\lambda_i > 0$
2: **Output:** Split index k^* such that clusters are $\langle \lambda_1, \ldots, \lambda_{k^*} \rangle$ and $\langle \lambda_{k^*+1}, \ldots, \lambda_n \rangle$
3: Initialize max_gap $\leftarrow 0$, $k^* \leftarrow 1$
4: **for** $i = 1$ to $n - 1$ **do**
5: Compute logarithmic gap: $g_i \leftarrow \ln(\lambda_{i+1}) - \ln(\lambda_i) = \ln\left(\dfrac{\lambda_{i+1}}{\lambda_i}\right)$
6: **if** $g_i >$ max_gap **then**
7: max_gap $\leftarrow g_i$
8: $k^* \leftarrow i$
9: **end if**
10: **end for**
11: **return** k^*

The logarithm of the ratio of consecutive eigenvalues is used to ensure that the split index k^* is chosen such that the ratio $\dfrac{\lambda_{k^*+1}}{\lambda_{k^*}}$ is maximized, as discussed above. The added advantage of using the logarithm instead of the ratio directly is that it prevents extremely large values in the inequality in line 6 of algorithm 7, possibly causing floating-point precision issues in the evaluation of said inequality.

Recursive application of algorithm 7 with stopping criterion based on the threshold condition in equation (4.22) leads to algorithm 8.

Algorithm 8 $\mathrm{PartitionEigenspectrum}(\sigma)$

1: **Input:** Sorted eigenvalues $\sigma = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$
2: **Output:** Sorted partition indices $K^* = \langle k_1^*, k_2^*, \ldots, k_{N_{\text{clusters}}-1}^*, n \rangle$
3: **if** $\sigma = \emptyset$ **then**
4: **return** $\emptyset$
5: **else if** $|\sigma| = n \leq 2$ **then**
6: **return** $\langle n \rangle$
7: **end if**
8: $\kappa \leftarrow \dfrac{\lambda_n}{\lambda_1}$
9: $k^* \leftarrow \mathrm{SplitEigenspectrum}(\sigma)$
10: $\kappa_l \leftarrow \dfrac{\lambda_{k^*}}{\lambda_1}$
11: $\kappa_r \leftarrow \dfrac{\lambda_n}{\lambda_{k^*+1}}$
12: **if** $\kappa > T^{(0)}(\kappa_l, \kappa_r)$ **then**
13: **return** $\mathrm{PartitionEigenspectrum}(\sigma_{\leq k^*}) \cup k^* + \mathrm{PartitionEigenspectrum}(\sigma_{>k^*})$
14: **else**
15: **return** $\langle n \rangle$ ▷ No further partitioning needed, return last index
16: **end if**

Note that algorithm 8 returns at a minimum a set containing only the last index $K^* = \langle n \rangle$, if the threshold condition is not satisfied for any split. Additionally, the value k^* is added (element-wise) to the result of the right-hand recursion in line 8 of algorithm 8. This is to ensure that the resulting set of indices K^* contains the correct, global indices of the eigenvalues in the spectrum, as the right-hand recursion only returns indices relative to the right-hand side of the split.

Finally, we can combine the partitioning from algorithm 8 with the Chebyshev degree calculation from algorithm 6 to obtain the sharpened CG bound algorithm 9. It is important to realize that partitioning done by algorithm 8 can also produce clusters consisting of a single eigenvalue, i.e. $[a_i, b_i] = [\lambda_i, \lambda_i]$. Clusters of this kind automatically satisfy the sparsity condition in equation (4.25) for any $\kappa \geq 1$, $\kappa_l, \kappa_r < \kappa$ and $\epsilon \leq \dfrac{1}{2}$. Therefore, these single eigenvalue clusters belong to the set of tail eigenvalues Λ_t. Even though we did not aim to find tail eigenvalues, we are forced to accept their existence in this case. Not merely because single eigenvalue clusters satisfy the sparsity condition, but also because the Chebyshev polynomial $\hat{C}_{p_i}^{(i)}$ corresponding to a single eigenvalue cluster is not well-defined.

Algorithm 9 $\mathrm{MultiClusterCGIterationBound}(\sigma, \epsilon)$

1: **Input:** Sorted eigenvalues $\sigma = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$, target relative error ϵ
2: **Output:** Sharpened CG iteration bound $m_{N_{\text{cluster}}} \leq m_1$
3: Initialize $\Lambda_t \leftarrow \emptyset$, clusters $\leftarrow \emptyset$, $k_a \leftarrow 1$
4: $K^* \leftarrow \mathrm{PartitionEigenspectrum}(\sigma)$
5: **for** $k_b \in K^*$ **do**
6: **if** $k_a = k_b$ **then**
7: $\Lambda_t \leftarrow \Lambda_t \cup \{\sigma[k_a]\}$ ▷ Single eigenvalue cluster, add to tail eigenvalues
8: **else**
9: clusters $\leftarrow$ clusters $\cup \langle [\sigma[k_a], \sigma[k_b]] \rangle$
10: **end if**
11: $k_a \leftarrow k_b + 1$
12: **end for**
13: $m_{N_{\text{cluster}}} \leftarrow \mathrm{GeneralizedCGIterationBound}(\text{clusters}, \Lambda_t, \epsilon)$
14: **return** $m_{N_{\text{cluster}}}$

Algorithm 9 first partitions the eigenspectrum into clusters using algorithm 8. Then, it constructs a list of clusters, leaving out single eigenvalue clusters, but adding them to the list of tail eigenvalues instead. Finally, the algorithm computes the sharpened CG bound using algorithm 6. In the case that

the eigenspectrum is never partitioned, i.e. the threshold condition is never satisfied, the algorithm returns the classical CG bound m_1 from equation (2.17).

4.5.2. Multi-tail-cluster CG iteration bound

In this section we adapt algorithm 8 to check additional conditions on $p = k^*$, that is

$$k^* < \left\lfloor \frac{\sqrt{\kappa_l}}{2} \ln \frac{2}{\epsilon} + 1 \right\rfloor$$

and the sparsity condition from equation (4.25). This results in algorithm 10. Similar to algorithm 8, the algorithm recursively partitions the eigenspectrum, returning the set of split indices K^*. In addition, algorithm 10 also updates a preinitialized set of tail eigenvalues Λ_t and a set of tail cluster start indices I_t. The tail cluster start indices are initialized to an offset value, which is incremented by the size of the left-hand partition at each recursive call. This way, the tail cluster start indices are always relative to the original eigenspectrum. The purpose of tracking the start indices of the tail clusters is to ensure that the tail eigenvalues can be identified and left out of the set of regular clusters upon calling the generalized multi-tail-cluster CG iteration bound in algorithm 11.

Algorithm 10 $\mathrm{PartitionEigenspectrumTails}(\sigma, \Lambda_t \leftarrow \emptyset, I_t \leftarrow \emptyset, \mathsf{offset} \leftarrow 1)$

Input: Sorted eigenvalues $\sigma = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$, preinitialized set of tail eigenvalues Λ_t, preinitialized set of tail cluster start indices I_t and an offset for the tail indices equal to 1
Output: Sorted partition indices $K^* = \langle k_1^*, k_2^*, \ldots, k_{N_{\mathsf{clusters}}-1}^*, n \rangle$, updated set of tail eigenvalues Λ_t and updated set of tail cluster start indices I_t
if $\sigma = \emptyset$ **then**
    return $\emptyset$
else if $|\sigma| = n = 1$ **then**
    $\Lambda_t \leftarrow \Lambda_t \cup \{\lambda_1\}$
    $I_t \leftarrow I_t \cup \{\mathsf{offset}\}$
    return $\langle 1 \rangle$
end if
$\kappa \leftarrow \dfrac{\lambda_n}{\lambda_1}$
$k^* \leftarrow \mathrm{SplitEigenspectrum}(\sigma)$
$\kappa_l \leftarrow \dfrac{\lambda_{k^*}}{\lambda_1}$
$\kappa_r \leftarrow \dfrac{\lambda_n}{\lambda_{k^*+1}}$
if $k^* < \left\lfloor \dfrac{\sqrt{\kappa_l}}{2} \ln \dfrac{2}{\epsilon} + 1 \right\rfloor$ and $k^* \leq \left\lfloor \sqrt{\dfrac{\kappa}{\kappa_r}} \log_{\frac{4\kappa}{\kappa_l}} \left(\dfrac{2}{\epsilon} \right) \right\rfloor$ **then**
    $\Lambda_t \leftarrow \Lambda_t \cup \sigma_{\leq k^*}$
    $I_t \leftarrow I_t \cup \{\mathsf{offset}\}$
    return $\langle k^* \rangle \cup k^* + \mathrm{PartitionEigenspectrumTails}(\sigma_{>k^*}, \Lambda_t, I_t, \mathsf{offset} + k^*)$
else if $\kappa > T^{(0)}(\kappa_l, \kappa_r)$ **then**
    return $\mathrm{PartitionEigenspectrumTails}(\sigma_{\leq k^*}, \Lambda_t, I_t, \mathsf{offset}) \cup \ldots$
        $k^* + \mathrm{PartitionEigenspectrumTails}(\sigma_{>k^*}, \Lambda_t, I_t, \mathsf{offset} + k^*)$
else
    return $\langle n \rangle$ ▷ No further partitioning needed, return last index
end if

We are now in a position to formulate the final algorithm of this chapter, the multi-tail-cluster CG iteration bound algorithm 11.

Algorithm 11 MultiTailClusterCGIterationBound(σ, ϵ)

Input: Sorted eigenvalues $\sigma = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$, target relative error ϵ
Output: Multi-tail-cluster CG iteration bound $m_{N_{\text{tail-cluster}}} \leq m_1$
Initialize $\Lambda_t \leftarrow \emptyset$, $I_t \leftarrow \emptyset$, clusters $\leftarrow \emptyset$, $k_a \leftarrow 1$
$K^* \leftarrow \text{PartitionEigenspectrumTails}(\sigma, \Lambda_t, I_t)$
for $k_b \in K^*$ **do**
    if $k_a \notin I_t$ **then** ▷ Cluster is **not** a tail cluster
        clusters $\leftarrow$ clusters $\cup \langle [\sigma[k_a], \sigma[k_b]] \rangle$
    end if
    $k_a \leftarrow k_b + 1$
end for
$m_{N_{\text{tail-cluster}}} \leftarrow \text{GeneralizedCGIterationBound}(\text{clusters}, \Lambda_t, \epsilon)$
return $m_{N_{\text{tail-cluster}}}$

5 Implementation

In this chapter we discuss the implementation of Problem 1.3 and the PCG method used to solve it, together with the relevant preconditioners from sections 2.2 and 3.2. We narrow the scope of the family of problems described by Problem 1.3 to three specific instances of the coefficient function $\mathcal{C}$. All implementations in this chapter, as well as the algorithms described in sections 4.4.2 and 4.5 are available in the online *High-Contrast Multi-Scale FEM* repository. For specific references to scripts or classes in the repository, we refer to the repository's top-level `README.md` file.

5.1. Implementation of the elliptic problem

We consider a square domain $\Omega = [0, 1]^2$ and introduce two conforming quadrilateral meshes Q_h and Q_H with fine and coarse mesh sizes h and H, respectively, where $h = H/2^r$ and $r \in \mathbb{N}$ a positive integer. We fix $r = 4$ and limit our study to the set of meshes

$$\mathcal{Q} = \{(Q_h, Q_H) \mid H \in \{1/4, 1/8, 1/16, 1/32, 1/64\}\}. \tag{5.1}$$

The *fine* and *coarse* meshes Q_h, Q_H for $H = 1/4$ are shown in figure 5.1.

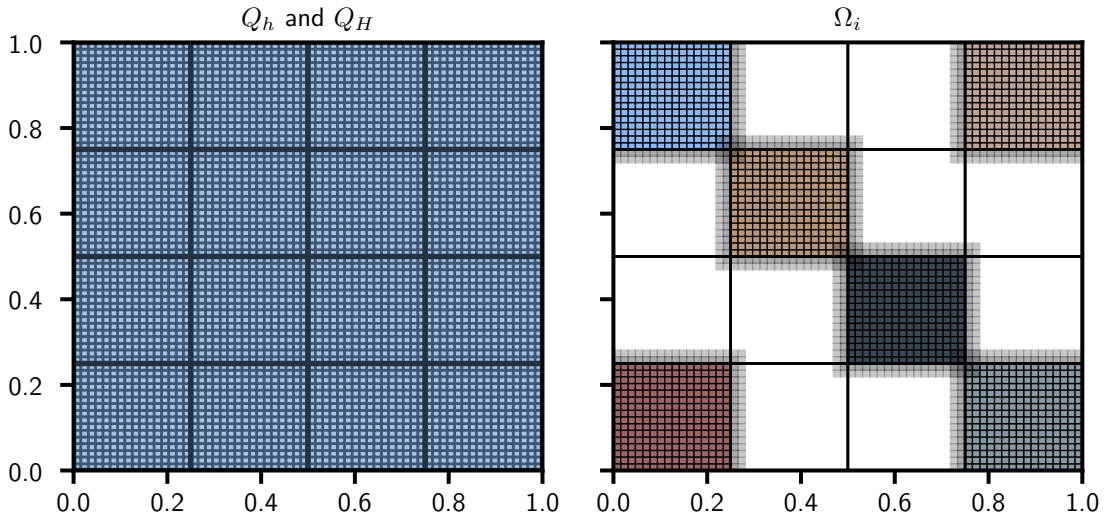

Figure 5.1: Plot of the conforming fine and coarse meshes Q_h and Q_H for $H = 1/4$ (**left**) and some of the overlapping subdomains Ω_i (**right**).

Next, we construct a FE space V_h from first order Legendre polynomials ϕ_i associated to each internal fine mesh vertex v_i^h in Q_h and locally defined on all quadrilateral elements q_j sharing v_i^h. That is

$$\text{supp}(\phi_i) = \bigcup_{j : v_i^h \in q_j, q_j \in Q_h} q_j, \quad \forall i \in \{1, \dots, n\}.$$

This defines the stiffness matrix A and load vector $\mathbf{b}$ as specified in Problem 1.3.

We construct for each fine mesh Q_h three coefficient functions $\mathcal{C}_{\text{const}} \equiv 1$, $\mathcal{C}_{\text{3layer, vert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$, the latter two of which are high-contrast coefficient functions with a periodic structure, as shown in figure 5.2.

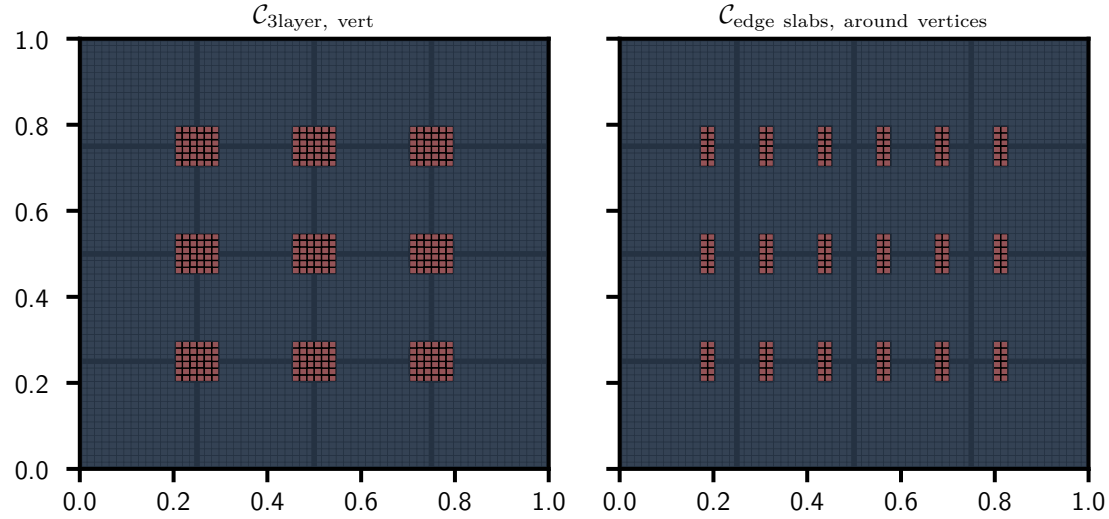

Figure 5.2: Plot of the coefficient function $\mathcal{C}_{\text{3layer, vert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$ defined on the fine mesh Q_h for $H = 1/4$. The contrast is $\frac{\mathcal{C}_{\text{max}}}{\mathcal{C}_{\text{min}}} = 10^8$ for both coefficient functions.

The coefficient functions $\mathcal{C}_{\text{3layer, vert}}$, $\mathcal{C}_{\text{edge slabs, around vertices}}$ are centered on or around those fine mesh vertices that lie on two coarse mesh edges $e_i^H \in Q_H$. The periodic structure of these coefficient functions is replicated when more subdomains are added. That is, the number of inclusions is kept proportional to the number of subdomains.

Finally, it is important to note that the meshes $\mathcal{M}$, finite element space V_h and coefficient functions in figure 5.2 are the same as the ones in [1].

5.2. Implementation of the PCG method

We implement a **PCG-type 1** method to solve the linear system arising from the discretization of the elliptic problem. To that end we first decompose the fine mesh Q_h into overlapping subdomains Ω_i to be used for the alternating Schwarz method, as visualized in figure 5.1. Then, we construct preconditioners with a general form similar to equation (2.26a)

$$M^{-1} = R_0^T A_0^{-1} R_0 + \sum_{i=1}^{N_{\text{sub}}} R_i^T A_i^{-1} R_i,$$

where R_i is the *restriction operator* to and $A_i = R_i^T A R_i$ is the *local operator* on Ω_i $\forall i \geq 1$. Also, $N_{\text{sub}} = (1/H)^2$, or simply the number of coarse mesh elements in Q_H. Similarly, the *coarse restriction* operator R_0 and corresponding *coarse operator* $A_0 = R_0^T A R_0$ are constructed as discussed in section 3.2. We construct R_0 for the GDSW, RGDSW, and AMS coarse spaces, resulting in the following two-level overlapping Schwarz preconditioners

$$\mathcal{M}^{-1} = \left\{ M^{-1}_{\text{2-OAS-GDSW}}, M^{-1}_{\text{2-OAS-RGDSW}}, M^{-1}_{\text{2-OAS-AMS}} \right\}. \tag{5.2}$$

6 Results

In this chapter we present and discuss two experiments with the sharpened CG iteration bounds from algorithms 9 and 11 applied to the approximate eigenspectra calculated as a by-product of the PCG method applied to Problem 1.3. The first experiment in section 6.1 tests the absolute sharpness of the new bounds as compared to the actual number of CG iterations required for convergence given a residual error tolerance ϵ_r. Where the first experiment presumes a full spectrum to be available at the time of calculating the new bounds, this is usually not the case in practice. Therefore, the second experiment in section 6.2 investigates the capability of the new bounds to predict the number of CG iterations required for convergence, but based on approximate spectra obtained in the first N_{iter} iterations. Each experiment is conducted on the same set of meshes $\mathcal{Q}$, using the preconditioners from $\mathcal{M}^{-1}$ and for at least the two coefficient functions $\mathcal{C}_{\text{3layer, vert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$ as described in chapter 5.

6.1. Sharpness of bounds

Here we run the PCG method from algorithm 3 until we achieve convergence in the sense of Theorem 2.6 with $\epsilon_r = 10^{-8}$. Every iteration we store the CG coefficients α, β such that at convergence we can construct the Lanczos matrix T_m using equations (A.2), (A.11) and (A.12). Then, we calculate all eigenvalues of T_m, also known as *Ritz* values. The eigenvalue spectrum $\sigma(T_m)$ at convergence of the PCG method is a *good* approximation of the spectrum $\sigma(M^{-1}A)$. Therefore, we can use $\sigma(T_m)$ as input for the multi-cluster and multi-tail-cluster CG iteration bounds from algorithms 9 and 11 and study how well the bounds match the actual number of CG iterations required for convergence m.

Before we present the iteration bounds, we show in figures 6.1 and 6.2 the output of $\mathrm{PartitionEigenspectrum}$ and $\mathrm{PartitionEigenspectrumTails}$ from algorithms 8 and 10 corresponding to the CG iteration bounds from algorithms 9 and 11 respectively. We observe that where the multi-cluster partitioning algorithm 8 consistently splits the eigenspectra $\sigma(M^{-1}A)\ \forall M^{-1} \in \mathcal{M}^{-1}$ into two clusters, the multi-tail-cluster partitioning algorithm 10 partitions the eigenspectrum into either one or two tail clusters and/or one regular cluster, depending on the coarse mesh size H.

For instance, consider the partitioning of the eigenspectrum $\sigma(M^{-1}_{\text{2-OAS-AMS}}A)$ for the mesh $Q_{1/4}$ and coefficient function $\mathcal{C}_{\text{3layer, vert}}$ among figures 6.1 and 6.2. $\mathrm{PartitionEigenspectrum}$ does not split $\sigma(M^{-1}_{\text{2-OAS-AMS}}A)$ and simply returns the extremal eigenvalues. In contrast, $\mathrm{PartitionEigenspectrumTails}$ finds two tail clusters, resulting in an output that consists exclusively of tail eigenvalues. In regard to the CG iteration bounds, this means that $m_{N_{\text{cluster}}} = m_1$ and $m_{N_{\text{tail-cluster}}} = m$. That is, the multi-cluster bound is equal to the classical bound, while the multi-tail-cluster bound is equal to the actual number of CG iterations required for convergence.

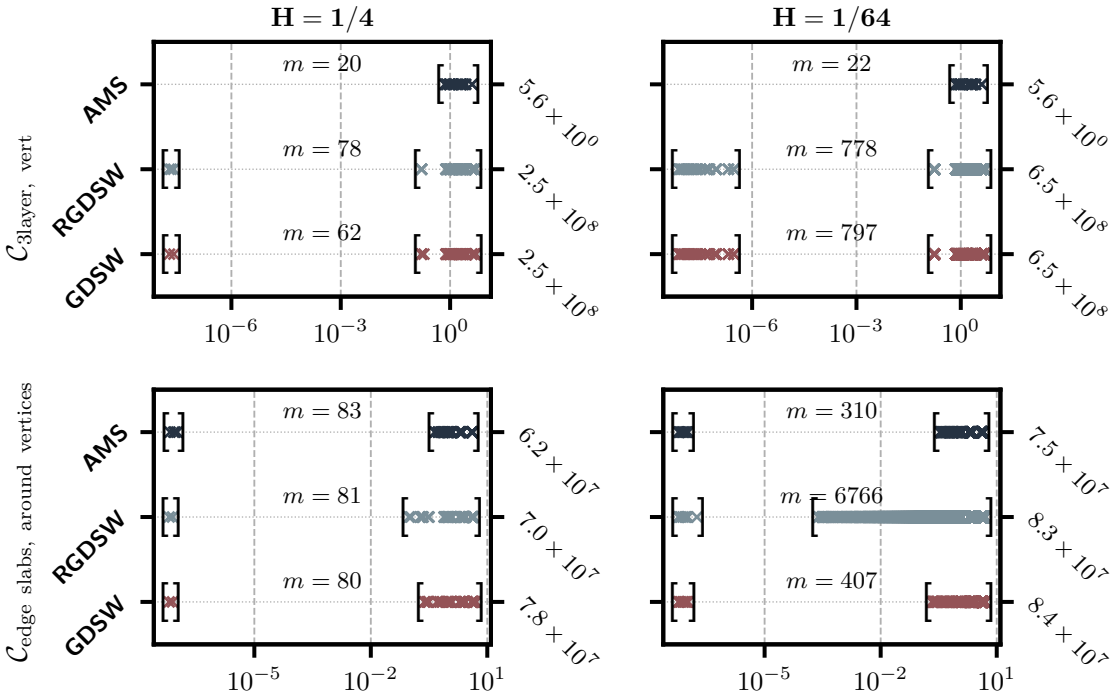

Figure 6.1: Partition of the eigenspectrum $\sigma(T_m) \approx \sigma(M^{-1}A)$ for $M^{-1} \in \mathcal{M}$ into clusters using the multi-cluster partitioning algorithm 8. The left and right columns correspond to coarse meshes $Q_{1/4}$ and $Q_{1/64}$ and the first and second row correspond to the coefficient functions $\mathcal{C}_{\text{3layer, vert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$ respectively. Every spectrum's condition number is plotted to the right of it as well as the number of PCG iterations m required for convergence. Eigenvalues are shown as crosses and clusters are indicated using square brackets.

Another interesting observation can be made when comparing the output of $\mathrm{PartitionEigenspectrumTails}$ given of the eigenspectrum $\sigma(M^{-1}_{\text{2-OAS-RGDSW}}A)$ for coefficient function $\mathcal{C}_{\text{3layer, vert}}$ among meshes $Q_{1/4}$ and $Q_{1/64}$ in figure 6.2. For the mesh $Q_{1/4}$, $\mathrm{PartitionEigenspectrumTails}$ finds two tail clusters and one regular cluster. For the finest mesh $Q_{1/64}$, however, it finds only one tail cluster and one regular cluster. This indicates that the second tail cluster and regular cluster are merged into one regular cluster as the mesh size H decreases. The term 'merged' is used here to mean that for $Q_{1/64}$ neither one of the splitting conditions in algorithm 10 is satisfied, i.e., the second cluster is not sufficiently separated from the third cluster by equation (4.22), nor do the number of eigenvalues in the cluster satisfy the sparsity condition from equation (4.25).

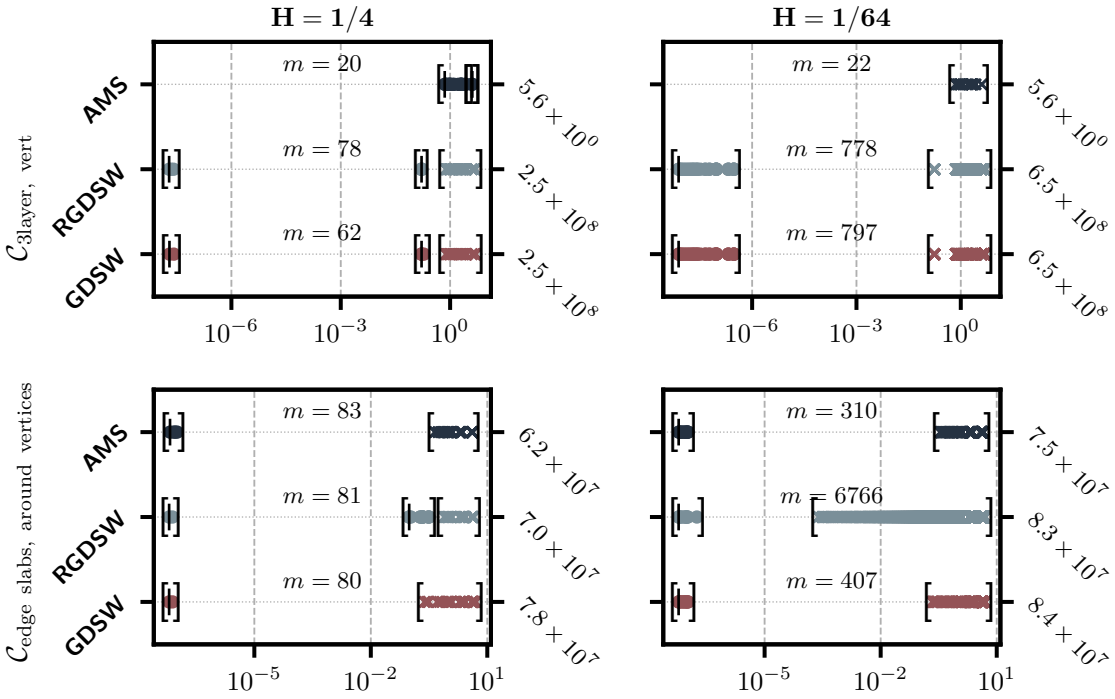

Figure 6.2: Partition of the eigenspectrum $\sigma(T_m) \approx \sigma(M^{-1}A)$ for $M^{-1} \in \mathcal{M}$ into tail clusters using the tail-cluster partitioning algorithm algorithm 10. The left and right columns correspond coarse meshes $Q_{1/4}$ and $Q_{1/64}$ and the first and second row correspond to the coefficient functions $\mathcal{C}_{\text{3layer, vert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$ respectively. Every spectrum's condition number is plotted to the right of it as well as the number of PCG iterations m required for convergence. Eigenvalues are shown either as crosses or as circles, depending on whether they form a regular or tail cluster, respectively. Additionally, starting indices of tail clusters, I_t, are indicated using vertical bars '|'

In figures 6.3 to 6.5 we present the CG iteration bounds $m_{N_{\text{cluster}}}, m_{N_{\text{tail-cluster}}}$ produced by $\mathrm{MultiClusterCGIterationBound}$ and $\mathrm{MultiTailClusteCGIterationBound}$ from algorithms 9 and 11, respectively. As mentioned in section 4.5, the output of $\mathrm{PartitionEigenspectrum}$ and $\mathrm{PartitionEigenspectrumTails}$ is specifically designed such that bounds $m_{N_{\text{cluster}}}, m_{N_{\text{tail-cluster}}} < m_1$. That is, no special effort is made in chapter 4 to make these bounds sharp with respect to the actual number of CG iterations m required for convergence. Fortunately, for all meshes, coefficient functions and preconditioners both bounds are either of the same order $m_{N_{\text{cluster}}}, m_{N_{\text{tail-cluster}}} = \mathcal{O}(m)$ or one order higher $m_{N_{\text{cluster}}}, m_{N_{\text{tail-cluster}}} = \mathcal{O}(10m)$.

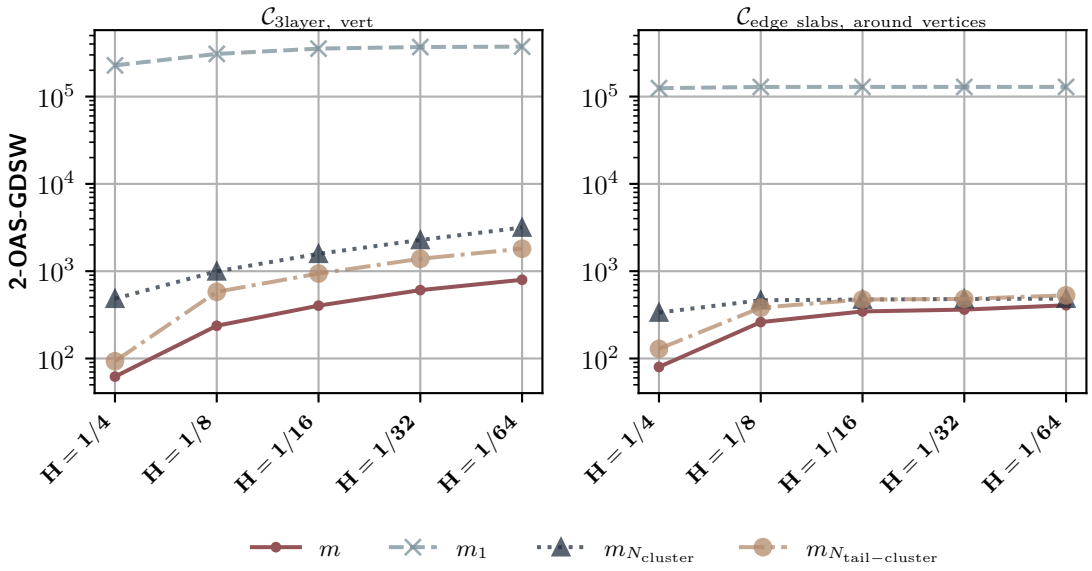

Figure 6.3: Plot of the number of CG iterations m (solid, red line with small circle markers)required to achieve convergence of the solution to Problem 1.3 in the sense of criterion equation (2.10) with $\epsilon_r = 10^{-8}$. The left and right columns corresponds to the $\mathcal{C}_{\text{3layer, vert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$ coefficient functions. Also shown are the corresponding classical m_1 (dashed, light-gray line with cross markers), multi-cluster $m_{N_{\text{cluster}}}$ (dotted, dark-blue line with triangle markers), and multi-tail-cluster $m_{N_{\text{tail-cluster}}}$ (dash-dotted, gold line with big circle markers) bounds for the CG method applied to the eigenspectra obtained from the Lanczos matrix convergence (Ritz values).

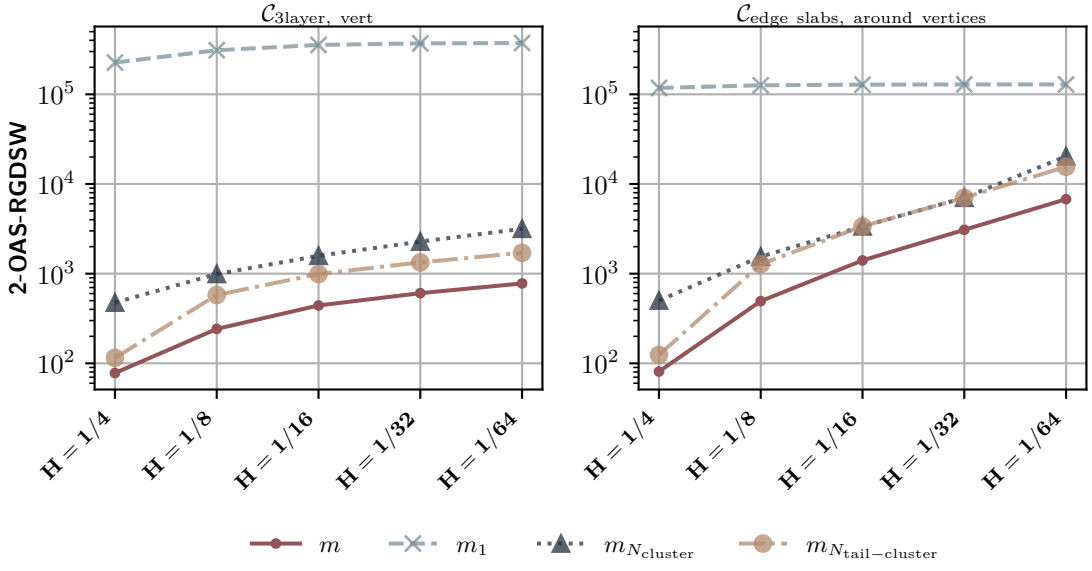

Figure 6.4: Similar to figure 6.3, but now for RGDSW coarse space.

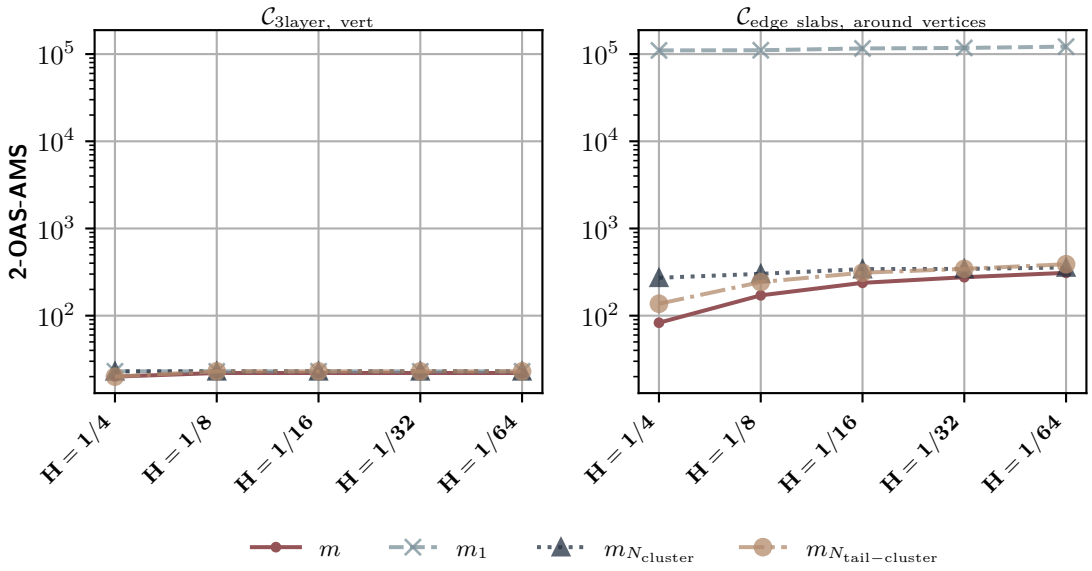

Figure 6.5: Similar to figure 6.3, but now for AMS coarse space.

Additionally, it appears $m_{N_{\text{tail-cluster}}} \lesssim m_{N_{\text{cluster}}}$. Even though this was not a design goal of the multi-tail-cluster CG iteration bound, it is a desirable property. The reason for the sharper multi-tail-cluster bound stems from the more rigorous partitioning done by $\mathrm{PartitionEigenspectrumTails}$ compared to $\mathrm{PartitionEigenspectrum}$, as shown in figure 6.2 and the related discussion above.

6.2. Early estimation of CG iteration bounds

We consider now a more practical setting where the eigenspectrum $\sigma(M^{-1}A)$ is not available at the time of calculating the CG iteration bounds. Instead, we assume that we only have access to an approximate eigenspectrum $\sigma(T_i)$ obtained from the Lanczos matrix T_i after i iterations of the PCG method. The goal is to investigate how well the multi-cluster and multi-tail-cluster CG iteration bounds can predict the number of CG iterations m required for convergence based on these approximate spectra.

6.2.1. Ritz value migration and convergence of bounds

To this end, we run the PCG method from algorithm 3 for N_{iter} iterations, where N_{iter} is a parameter that we can choose. Every iteration i, we calculate the multi-cluster and multi-tail-cluster CG iteration bounds $m_{N_{\text{cluster}}}(\sigma(T_i))$ and $m_{N_{\text{tail-cluster}}}(\sigma(T_i))$ using the approximate eigenspectrum $\sigma(T_i)$. In figures 6.6 and 6.7 we show the convergence of the multi-cluster and multi-tail-cluster CG iteration bounds $m_{N_{\text{cluster}}}(\sigma(T_i))$ and $m_{N_{\text{tail-cluster}}}(\sigma(T_i))$ as a function of the number of iterations i with $N_{\text{iter}} = 300$.

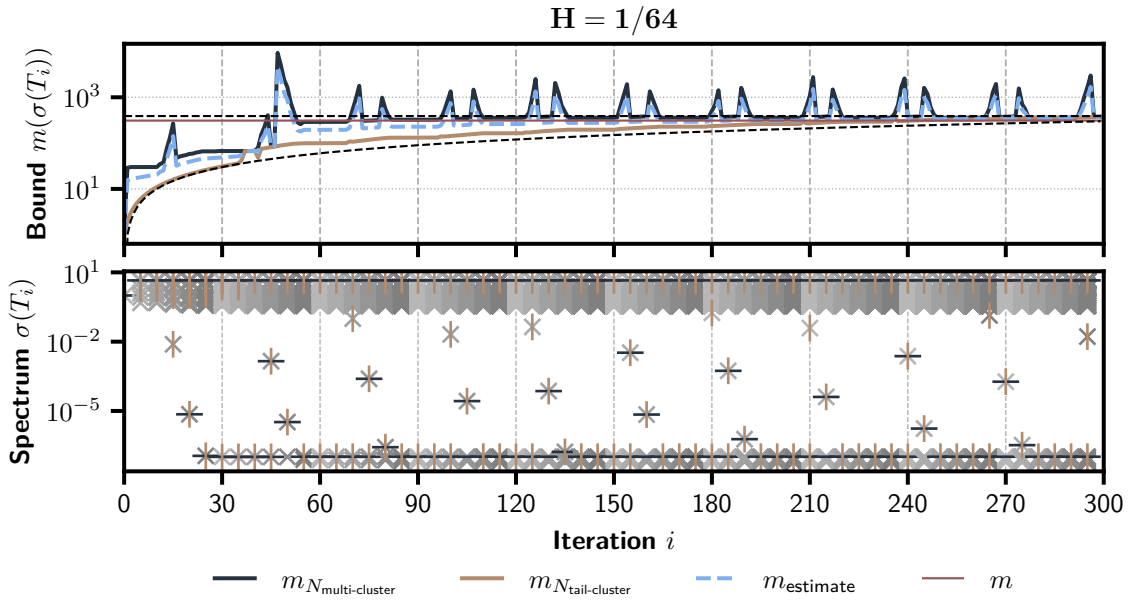

Figure 6.6: Plots of the multi-cluster CG iteration bounds $m_{N_{\text{cluster}}}$, $m_{N_{\text{tail-cluster}}}$ and $m_{\text{estimate}} = \frac{1}{2}(m_{N_{\text{cluster}}} + m_{N_{\text{tail-cluster}}})$ for the first $N_{\text{iter}} = 300$, $Q_{1/64}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$ (top) together with the approximate spectra $\sigma(T_i) \approx \sigma(M_{\text{2-OAS-AMS}}^{-1} A)$ that they are obtained from (bottom). Additionally, dark-blue dashes and gold bars indicate the partitions obtained from $\mathrm{PartitionEigenspectrum}$ and $\mathrm{PartitionEigenspectrumTails}$ respectively. In the top figure the number of CG iterations m required for convergence is indicated by a solid, black, horizontal line. The dashed, black curve is the identity plot of i. Finally, the number of CG iterations calculated by the multi-tail-cluster bound is indicated by a dashed, black, horizontal line.

Most notably, we observe that in figure 6.6 the multi-cluster bound converges to its final value within about 60 iterations, whereas the multi-tail-cluster bound needs almost all m iterations to converge. We can understand this difference by first looking at the spectra $\sigma(T_i)$ in the bottom of figure 6.6. We notice that the first cluster eigenvalue λ_1 splits off from the right cluster, forms a single eigenvalue cluster and settles in its final position all before 30 iterations. In the next 30 iterations, the second eigenvalue λ_2 separates from the right cluster, migrates towards the left cluster and joins it. In the process, several things change in the output of $\mathrm{PartitionEigenspectrum}$. First, the right cluster expands, as a consequence of λ_2 migrating away, yet still remaining a member. Second, in its migration λ_2 eventually satisfies the threshold condition from equation (4.22) and is removed from the right cluster, resulting in a rapid decrease of the right cluster size and the temporary formation of a third, single eigenvalue cluster consisting of only λ_2. What happened to the right cluster, happens in reverse order to the left cluster as λ_2 joins the left cluster, resulting in a rapid increase of the left cluster size, followed by a slow decrease of the left cluster size as λ_2 migrates towards the left cluster's extremal eigenvalue λ_1. This results in the characteristic two-peak shape of the multi-cluster bound in figure 6.6. We deem the process of the migration of an eigenvalue from the right to the left cluster and the characteristic two-peak shape in the plot of $m_{N_{\text{cluster}}}$ as *Ritz value migration*.

We are now in a position to understand the more rapid convergence of the multi-cluster bound compared to the multi-tail-cluster bound. That is, the multi-cluster bound needs at a minimum two clusters that approximate the final two clusters of the spectrum. This happens after at least two Ritz eigenvalue migrations have occurred. In contrast, the tail cluster bound never considers the left group of tail eigenvalues as a regular cluster. Consequently, the effect of the left tail eigenvalues on the multi-tail-cluster bound only becomes apparent once the left tail has gathered a *sufficient* number of tail eigenvalues. Moreover, the characteristic two-peak shape corresponding to the Ritz value migration is not present in the multi-tail-cluster bound. This can be attributed to the more rigorous splitting of $\mathrm{PartitionEigenspectrumTails}$, preventing the growth and shrinkage of the right cluster and, conversely, of the left cluster. Instead, the multi-tail-cluster bound is more stable and converges to its final value in a more gradual manner, yet does so more slowly than the multi-cluster bound.

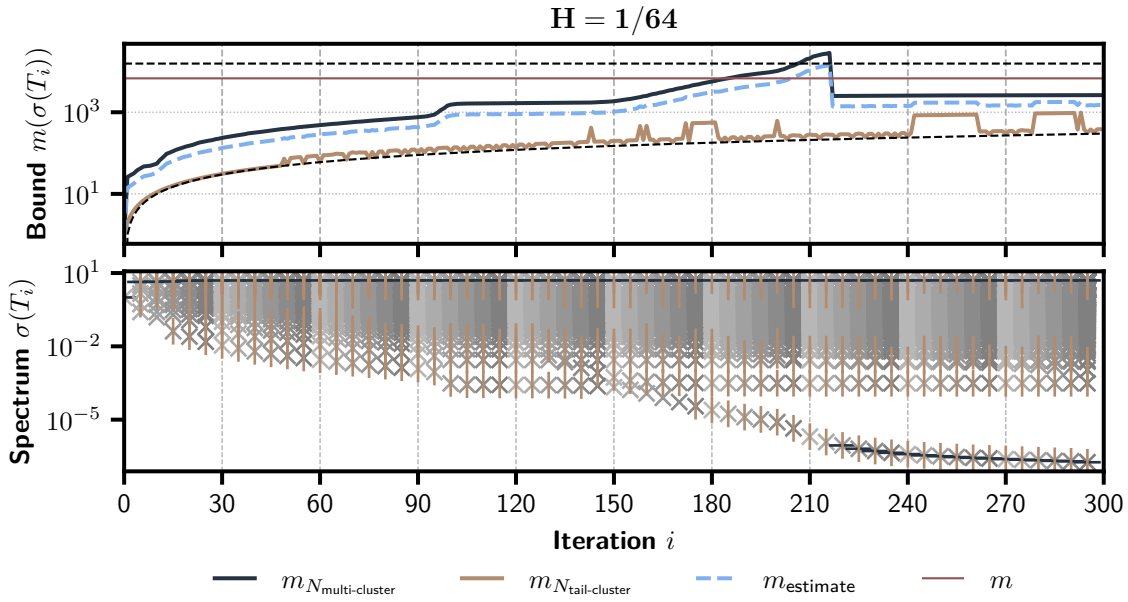

Figure 6.7: Similar to figure 6.6, but now for the RGDSW coarse space.

In figure 6.7 we observe a similar behavior of the multi-cluster and multi-tail-cluster bounds as in figure 6.6. However, the Ritz value migration occurs at a slower rate resulting in only one, single eigenvalue left cluster being formed within the first 300 iterations. Effectively, this means that both the multi-cluster and tail-cluster bounds underestimate the number of CG iterations m required for convergence.

6.2.2. Comparison of bounds

Here we utilize the results from section 6.2.1 to generate early CG iteration bounds $m_{N_{\text{cluster}}}(\sigma(T_i))$ and $m_{N_{\text{tail-cluster}}}(\sigma(T_i))$ for $i = 1, \dots, N_{\text{iter}}$, where N_{iter} satisfies

$$N_{\text{iter}} = \min\{300, \lfloor f_{\text{iter}} m \rfloor\}, \quad f_{\text{iter}} \in (0, 1],$$

where m is the final number of CG iterations required for convergence. Additionally, we use a simple algorithm that can detect temporary convergence of the extremal eigenvalues in the set of clusters that results from $\mathrm{PartitionEigenspectrum}$. That is let K^* be as in section 4.5, then we check after some specified update frequency N_{update} whether the extremal eigenvalues of the clusters in K^* have not changed for the last N_{update} iterations

$$\forall k \in K^* \quad \frac{\lambda_{k,i}}{\lambda_{k,i-N_{\text{update}}}} < 1 + \tau_{\text{extremal}}, \quad i = N_{\text{update}}, 2N_{\text{update}}, \dots, N_{\text{iter}}, \tag{6.1}$$

where τ_{extremal} is some tolerance parameter. If this condition is satisfied, we assume that the extremal eigenvalues of the clusters have converged, and we can use the current clusters to calculate the CG iteration bounds $m_{N_{\text{cluster}}}(\sigma(T_i))$ and $m_{N_{\text{tail-cluster}}}(\sigma(T_i))$.

The tables 6.1 to 6.3 show the results of this process for $f_{\text{iter}} = 0.6$, $N_{\text{update}} = 5$ and $\tau_{\text{extremal}} = 0.1$ for the coefficient function $\mathcal{C}_{\text{const}}$, $\mathcal{C}_{\text{3lvert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$, respectively.

Table 6.1: PCG iteration bounds m_1, $m_{N_{\text{cluster}}}$, $m_{N_{\text{tail-cluster}}}$, m_{estimate} for solving the model diffusion problem with coefficient function $\mathcal{C}_{\text{const}}$. Bounds are based on approximate spectra (Ritz values) obtained during the initial PCG iterations and are shown for meshes $\mathbf{H = 1/4}$, $\mathbf{H = 1/8}$, $\mathbf{H = 1/16}$, $\mathbf{H = 1/32}$, $\mathbf{H = 1/64}$ and 2-OAS preconditioners with GDSW, RGDSW, AMS coarse spaces. The i column shows the iteration at which the bounds are obtained. The color of each cell indicates whether the bound is larger (blue) or smaller (red) than the number of iterations required for convergence m. The shade of the cell is proportional to the absolute difference between m and the bound.

		m	m_1	$m_{N_{\text{cluster}}}$	$m_{N_{\text{tail-cluster}}}$	m_{estimate}	i
$\mathbf{H = 1/4}$	GDSW	23	34	34	11	23	11
	RGDSW	21	40	40	11	26	11
	AMS	18	23	23	11	17	11
$\mathbf{H = 1/8}$	GDSW	30	37	37	16	27	16
	RGDSW	32	43	43	11	27	11
	AMS	21	23	23	11	17	11
$\mathbf{H = 1/16}$	GDSW	33	37	37	16	27	16
	RGDSW	38	45	45	16	31	16
	AMS	22	24	24	16	20	16
$\mathbf{H = 1/32}$	GDSW	35	37	37	16	27	16
	RGDSW	42	44	44	16	30	16
	AMS	22	24	24	16	20	16
$\mathbf{H = 1/64}$	GDSW	36	37	37	16	27	16
	RGDSW	45	44	44	16	30	16
	AMS	23	29	29	11	20	11

Table 6.2: PCG iteration bounds m_1, $m_{N_{\text{cluster}}}$, $m_{N_{\text{tail-cluster}}}$, m_{estimate} for solving the model diffusion problem with coefficient function $\mathcal{C}_{3\text{layer, vert}}$. Bounds are based on approximate spectra (Ritz values) obtained during the initial PCG iterations and are shown for meshes $\mathbf{H = 1/4}$, $\mathbf{H = 1/8}$, $\mathbf{H = 1/16}$, $\mathbf{H = 1/32}$, $\mathbf{H = 1/64}$ and 2-OAS preconditioners with GDSW, RGDSW, AMS coarse spaces. The i column shows the iteration at which the bounds are obtained. The color of each cell indicates whether the bound is larger (blue) or smaller (red) than the number of iterations required for convergence m. The shade of the cell is proportional to the absolute difference between m and the bound.

		m	m_1	$m_{N_{\text{cluster}}}$	$m_{N_{\text{tail-cluster}}}$	m_{estimate}	i
$\mathbf{H = 1/4}$	GDSW	62	225,419	111	26	69	26
	RGDSW	78	224,028	109	26	68	26
	AMS	20	23	23	11	17	11
$\mathbf{H = 1/8}$	GDSW	237	300,788	641	91	366	51
	RGDSW	242	302,967	638	86	362	51
	AMS	22	23	23	11	17	11
$\mathbf{H = 1/16}$	GDSW	403	350,992	949	148	549	96
	RGDSW	442	351,891	948	147	548	96
	AMS	22	23	23	11	17	11
$\mathbf{H = 1/32}$	GDSW	607	367,021	2,228	231	1,230	161
	RGDSW	606	368,220	2,277	290	1,284	216
	AMS	22	23	23	11	17	11
$\mathbf{H = 1/64}$	GDSW	797	359,087	1,764	85	925	46
	RGDSW	778	359,976	1,763	85	924	51
	AMS	22	23	23	11	17	11

Table 6.3: PCG iteration bounds m_1, $m_{N_{\text{cluster}}}$, $m_{N_{\text{tail-cluster}}}$, m_{estimate} for solving the model diffusion problem with coefficient function $\mathcal{C}_{\text{edge slabs, around vertices}}$. Bounds are based on approximate spectra (Ritz values) obtained during the initial PCG iterations and are shown for meshes $\mathbf{H = 1/4}$, $\mathbf{H = 1/8}$, $\mathbf{H = 1/16}$, $\mathbf{H = 1/32}$, $\mathbf{H = 1/64}$ and 2-OAS preconditioners with GDSW, RGDSW, AMS coarse spaces. The i column shows the iteration at which the bounds are obtained. The color of each cell indicates whether the bound is larger (blue) or smaller (red) than the number of iterations required for convergence m. The shade of the cell is proportional to the absolute difference between m and the bound.

		m	m_1	$m_{N_{\text{cluster}}}$	$m_{N_{\text{tail-cluster}}}$	m_{estimate}	i
$\mathbf{H = 1/4}$	GDSW	80	124,727	88	36	62	36
	RGDSW	81	117,699	133	36	85	36
	AMS	83	98,421	191	84	138	41
$\mathbf{H = 1/8}$	GDSW	261	127,648	383	173	278	96
	RGDSW	494	124,745	1,274	583	929	186
	AMS	171	110,677	302	123	213	71
$\mathbf{H = 1/16}$	GDSW	346	123,872	306	134	220	76
	RGDSW	1,406	122,195	2,185	979	1,582	241
	AMS	238	110,969	310	126	218	81
$\mathbf{H = 1/32}$	GDSW	363	122,634	310	136	223	76
	RGDSW	3,082	109,007	1,415	920	1,168	271
	AMS	276	110,970	312	127	220	86
$\mathbf{H = 1/64}$	GDSW	407	127,897	485	268	377	186
	RGDSW	6,766	69,645	2,612	956	1,784	291
	AMS	310	114,629	324	132	228	91

In table 6.1 we can see that the classical bound m_1 and the multi-cluster bound $m_{N_{\text{cluster}}}$ are equal to each other. That is, no additional partitioning is performed by $\mathrm{PartitionEigenspectrum}$ and the multi-cluster bound reduces to the classical bound, as discussed in section 4.5. On the other hand, the multi-tail-cluster bound $m_{N_{\text{tail-cluster}}}$ is equal to the actual number of CG iterations i at the time of its calculation. This is a consequence of the fact that the eigenspectrum $\sigma(T_i)$ for $i = 11, 16$ as in table 6.1, is so sparse that $\mathrm{PartitionEigenspectrum}$ does not find any clusters. This is similar to the case discussed in figure 6.2 for the mesh $Q_{1/4}$ and coefficient function $\mathcal{C}_{\text{3layer, vert}}$. Moving to the high contrast coefficient functions $\mathcal{C}_{\text{3lvert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$ in tables 6.2 and 6.3, we can confirm that the multi-cluster bound performs better than the classical bound m_1 and differs from m by factor of 2 to 4, that is

$$km \sim m_{N_{\text{cluster}}} < m_1, \ k < 4. \tag{6.2}$$

Expectedly, the multi-tail-cluster bound $m_{N_{\text{tail-cluster}}}$ is too sharp to be useful in early estimation of the number of CG iterations m required for convergence. However, it may still serve a purpose as part of m_{estimate}, see figures 6.6 and 6.7. Even though m_{estimate} is merely a heuristic, it does seem to perform well as an early estimate of the number of CG iterations m required for convergence, as evidenced by both tables 6.2 and 6.3

We note that for the coefficient function $\mathcal{C}_{\text{edge slabs, around vertices}}$ in table 6.3 both bounds $m_{N_{\text{cluster}}}$, $m_{N_{\text{tail-cluster}}}$ and the heuristic m_{estimate} underestimate the number of CG iterations for the preconditioner $M_{\text{2-OAS-RGDSW}}$ on the meshes $Q_{1/32}$ and $Q_{1/64}$. This happens for the same reasons as discussed in section 6.2.1, i.e., the Ritz value migration is not yet complete, and the left cluster has not fully formed yet. For the bounds to still be useful in practice, we can simply increase the number of iterations N_{iter} to allow for more Ritz value migrations to occur. Or, if this is computationally too expensive, we can still use the bounds as indications of the number of CG iterations m required for convergence.

Finally, considering all tables 6.1 and 6.3 at once, we can say that the multi-cluster bound $m_{N_{\text{cluster}}}$ gives the most robust early upper bound on the number of CG iterations m required for convergence, assuming that one does sufficient iterations such that several Ritz values have migrated to the left cluster(s). The tail-cluster bound gives a sharper bound that is only accurate for a more developed set of Ritz eigenvalues, that is closer to the actual spectrum of A, as in figures 6.3 to 6.5. Be that as it may, within the first $N_{\text{iter}} = 300$ both bounds are able to distinguish between the robust preconditioners GDSW and AMS on the one side and the non-robust RGDSW on the other.

7 Conclusion

This thesis has stressed that the classical condition number-based Conjugate Gradient (CG) iteration bound, m_1, from equation (2.18), does not fully capture the convergence behavior in high-contrast heterogeneous elliptic problems, particularly when two-level Schwarz preconditioners are employed. The spectra of the preconditioned systems, $\sigma(M^{-1}A)$ for all $M^{-1} \in \mathcal{M}^{-1}$ as defined in equation (5.2), were found in [1] to not only possess a condition number like that of equation (3.3) but also to exhibit the spectral gap discussed in section 3.1. The presence of this spectral gap undermines the assumption of a uniformly distributed eigenspectrum, a key premise in the derivation of the classical CG iteration bound. This observation forms the primary motivation for this thesis: the classical bound is too coarse for high-contrast problems, necessitating the development of sharper, more descriptive bounds.

7.1. Development of Sharpened Iteration Bounds

A review of the relevant literature in section 3.3 revealed that a two-cluster sharpened CG iteration bound was derived in [2, Section 4], expressed in terms of the four extremal eigenvalues of the two clusters within a spectrum. In section 4.1, this bound was re-derived. The necessary conditions for which this two-cluster bound, m_2, is sharper than the classical bound, m_1, were established in sections 4.2 and 4.3 through equations (4.22) and (4.25). This process identified the three key spectral characteristics sought by **Q2**: the left- and right-cluster condition numbers, κ_l and κ_r, and the spectral width, $s = \dfrac{\kappa}{\kappa_l}$.

Building on the work of Axelsson, we developed two novel sharpened CG iteration bounds in section 4.5. By combining the aforementioned necessary conditions based on κ_l, κ_r and s ensuring $m_2 < m_1$ with a recursive partitioning algorithm and a generalized multi-cluster version of the bound from [2], we introduced a multi-cluster bound, $m_{N_{\text{cluster}}}$ (algorithm 9), and a tail-cluster bound, $m_{N_{\text{tail-cluster}}}$ (algorithm 11). The only difference between these two bounds is that the tail-cluster bound treats smaller, sparse clusters in a spectrum differently by considering each eigenvalue in that cluster individually, rather than as part of a collective cluster. This distinction allows the tail-cluster bound to capture more detailed spectral information.

7.2. Numerical Validation and Performance

The sharpness of these new bounds was investigated in section 6.1 by applying them to approximate eigenspectra obtained from the Lanczos matrix. These spectra resulted from a converged PCG process with relative residual error stopping-criterion of $\epsilon_r = 10^{-8}$ applied to Problem 1.3 for three two-level Schwarz preconditioners constructed with GDSW, RGDSW, or AMS coarse spaces. The bounds were evaluated for two specific high-contrast scalar coefficient functions, $\mathcal{C}_{\text{3layer, vert}}$ and $\mathcal{C}_{\text{edge slabs, around vertices}}$, as shown in figure 5.2.

In all tested scenarios, the tail-cluster bound, $m_{N_{\text{tail-cluster}}}$, consistently outperformed the multi-cluster bound, $m_{N_{\text{cluster}}}$, which in turn was significantly sharper than the classical condition number-based bound, m_1. Specifically, we found that both new bounds are either of the same order as m, that is $m_{N_{\text{cluster}}}, m_{N_{\text{tail-cluster}}} = \mathcal{O}(m)$, or one order higher, $m_{N_{\text{cluster}}}, m_{N_{\text{tail-cluster}}} = \mathcal{O}(10m)$. This result positively answers **Q1** and demonstrates the superior accuracy of the newly developed bounds. Furthermore, both

$m_{N_{\text{cluster}}}$ and $m_{N_{\text{tail-cluster}}}$ provide valuable information about the convergence behavior and can distinguish between the robustness of different preconditioners more accurately than the classical bound.

7.3. Challenges in Practical Estimation and Future Directions

Despite their sharpness, we showed in section 6.2 that the practical application of $m_{N_{\text{cluster}}}$ and $m_{N_{\text{tail-cluster}}}$ for *a priori* iteration estimation faces challenges. The bounds require more detailed spectral information than is typically available or computationally feasible to obtain from the initial PCG iterations. The core issue is that the Ritz values may not converge quickly enough to the true eigenvalues of A, particularly the internal ones defining cluster boundaries, to provide an accurate estimate of the full spectrum in the early PCG iterations. Consequently, the answer to **Q3** is that the utility of these bounds for early estimation depends on the specific coefficient function and preconditioner used.

In fact, the classical bound m_1 suffers from slowly converging Ritz values as well. Though to a lesser extent than the new bounds. This is because m_1 only relies on the extremal eigenvalues of the spectrum. In contrast, $m_{N_{\text{cluster}}}$ is most accurate when the extremal eigenvalues of *each cluster* are well approximated by the Ritz values, which is not always the case. Similarly, $m_{\text{tail-cluster}}$ requires the extremal eigenvalues of each cluster as well as a set of specific tail eigenvalues to be well approximated by the Ritz values. This is a more stringent requirement, which explains why $m_{N_{\text{tail-cluster}}}$ is not always an upper bound for m when only a few Ritz values are available.

For most combinations of coefficient functions and meshes tested with the GDSW and AMS coarse spaces, we observed the following relationship within the first 300 PCG iterations:

$$m_{\text{tail-cluster}}(\sigma(T_i)) \lesssim m \lesssim m_{N_{\text{cluster}}}(\sigma(T_i)) \text{ for } i \leq N_{\text{iter}} = 300,$$

where m is the actual number of iterations and $\sigma(T_i)$ is the Ritz spectrum at iteration i. This suggests we can leverage the tail-cluster bound to obtain a more accurate estimate of the number of iterations required for convergence. The average of the multi-cluster and tail-cluster bounds at iteration i denoted as m_{estimate} is such a heuristic. Though m_{estimate} is arbitrarily constructed and there is no guarantee that it is a valid upper bound, it provides good estimates of m. However, even the performance of $m_{N_{\text{cluster}}}$ as an upper bound and m_{estimate} as a heuristic deteriorates when the RGDSW coarse space is used to solve Problem 1.3 with $\mathcal{C}_{\text{edge slabs, around vertices}}$, where they can underestimate the true iteration count.

In conclusion, the main goal of this thesis was to sharpen the CG iteration bound for Schwarz-preconditioned high-contrast heterogeneous scalar-elliptic problems beyond the classical condition number-based bound. The derived multi-cluster and tail-cluster bounds offer a more nuanced and accurate picture of convergence behavior than the classical condition number-based bound, able to distinguish between preconditioners effectively.

Future research should focus on two main areas. First, applying the new bounds to a wider range of problems, including those with more complex high-contrast coefficients, finer mesh discretizations, different preconditioners, and other types of PDEs. Second, and more fundamentally, research into the *a priori* estimation of the key spectral characteristics (κ_l, κ_r, s) is crucial to circumvent the dependency of the new bounds on the slowly converging Ritz values, which would unlock their full potential for predictive performance analysis.

Appendix

A. Derivation of the CG Method

A.1. Arnoldi's method for linear systems

Arnoldi's method for linear systems $A\mathbf{u} = \mathbf{b}$, where A is a general (possibly non-symmetric) stiffness matrix, is just an instantiation of algorithm 1. It uses a Gramm-Schmidt orthogonalization procedure to simultaneously obtain the basis V of $\mathcal{K}$ and the Hessenberg matrix, see Definition 2.2. Assuming without loss of generality that V has dimension m, we set $V = V_m$ and let $\mathbf{v}_1 = \mathbf{r}_0/||\mathbf{r}_0||_2$ and $\beta = ||\mathbf{r}_0||_2$, then by Definition 2.1 we have

$$V_m^T A V_m = H_m \text{ and } V_m^T \mathbf{r}_0 = V_m^T \beta \mathbf{v}_1 = \beta e_1 \implies \begin{array}{c} \mathbf{u}_m = \mathbf{u}_0 + V_m \mathbf{c}, \\ H_m \mathbf{c} = \beta e_1. \end{array}$$

Substituting this into the template for the error projection methods given in algorithm 1 gives algorithm A.1.

Algorithm A.1 Arnoldi's method for linear systems (FOM) [21, Algorithm 6.4]

Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{u}_0$, $\beta = ||\mathbf{r}_0||_2$ and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
Define $H_m = \{0\}$
Define $V_1 = \{\mathbf{v}_1\}$
for $j = 1, 2, \ldots, m$ **do**
  $\mathbf{w}_j = A\mathbf{v}_j$
  for $i = 1, 2, \ldots, j$ **do**
    $h_{ij} = (\mathbf{w}_j, \mathbf{v}_i)$ and store h_{ij} in H_m
    $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$
  end for
  $h_{j+1,j} = ||\mathbf{w}_j||_2$
  if $h_{j+1,j} = 0$ **then**
    $m = j$
    break
  end if
  $\mathbf{v}_{j+1} = \mathbf{w}_j/h_{j+1,j}$ and store $\mathbf{v}_{j+1}$ into V_{j+1}
end for
Solve $H_m\mathbf{c} = \beta e_1$ for $\mathbf{c}$
$\mathbf{u}_m = \mathbf{u}_0 + V_m\mathbf{c}$

Note that a stopping criterion can be derived from the residual vector $\mathbf{r}_m = \mathbf{b} - A\mathbf{u}_m$. Theorem A.1 gives a way of calculating the size of the residual vector.

Theorem A.1: Arnoldi residual [21, Proposition 6.7]

The residual vector $\mathbf{r}_m = \mathbf{b} - A\mathbf{u}_m$ satisfies

$$\|r_m\|_2 = h_{m+1,m}|\mathbf{e}_m^T \mathbf{c}|, \tag{A.1}$$

Proof. We have

$$\begin{aligned} \mathbf{r}_m &= \mathbf{b} - A\mathbf{u}_m \\ &= \mathbf{r}_0 - AV_m\mathbf{c} \\ &= \beta v_1 - V_m H_m \mathbf{c} - h_{m+1,m}\mathbf{e}_m^T \mathbf{c}\mathbf{v}_{m+1} \\ &= -h_{m+1,m}\mathbf{e}_m^T \mathbf{c}\mathbf{v}_{m+1}. \end{aligned}$$

The result follows by taking the 2-norm of both sides of the equality and using the fact that $||\mathbf{v}_{m+1}||_2 = 1$. □

A.2. Lanczos' Algorithm

In the special case where A is symmetric, the Arnoldi method can be simplified to the Lanczos algorithm. In particular, for symmetric A, the Hessenberg matrix H_m is tridiagonal

$$H_m = T_m = \begin{pmatrix} \delta_1 & \eta_2 & 0 & \dots & 0 \\ \eta_2 & \delta_3 & \eta_3 & \dots & 0 \\ 0 & \eta_3 & \delta_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \eta_m \\ 0 & 0 & 0 & \eta_m & \delta_m \end{pmatrix}, \tag{A.2}$$

where we redefined H_m to be the tridiagonal matrix T_m, also called the *Ritz matrix*. The tridiagonality of T_m allows us to reduce the Gramm-Schmidt orthogonalization procedure in the inner for-loop in algorithm A.1 to just two vector subtractions and an inner product, resulting in algorithm A.1

Algorithm A.2 Lanczos algorithm for linear systems [21, Algorithm 6.16]

Compute $\mathbf{r}_0 = b - A\mathbf{u}_0$, $\beta = ||\mathbf{r}_0||_2$, $\mathbf{v}_0 = 0$ and $\mathbf{v}_1 = \mathbf{r}_0/\beta$
$V_1 = \{\mathbf{v}_1\}$
for $j = 1, 2, \dots, m$ **do**
  $\mathbf{w}_j = A\mathbf{v}_j - \eta_j \mathbf{v}_{j-1}$
  $\delta_j = (\mathbf{w}_j, \mathbf{v}_j)$
  $\mathbf{w}_j = \mathbf{w}_j - \delta_j \mathbf{v}_j$
  $\eta_{j+1} = ||\mathbf{w}_j||_2$
  if $\eta_{j+1} = 0$ **then**
    $m = j$
    Break
  end if
  $\mathbf{v}_{j+1} = \mathbf{w}_j/\eta_{j+1}$ and store $\mathbf{v}_{j+1}$ into V_{j+1}
end for
Solve the tridiagonal system $T_m\mathbf{c} = \beta\mathbf{e}_1$ for $\mathbf{c}$
$\mathbf{u}_m = \mathbf{u}_0 + V_m\mathbf{c}$

A.3. D-Lanczos

A downside of algorithm A.2 in particular and projections methods like algorithm 1 in general is their reliance on an arbitrary choice of dimension m. The methods run until the basis V_m is constructed and subsequently construct H_m to determine the correction $\mathbf{c}$. This is not ideal, since the resulting solution $\mathbf{u}_m$ may not be close enough to the true solution $\mathbf{u}$. That is, it is not guaranteed that the residual vector $\mathbf{r}_m$ is 'small enough'. Alternately, it might happen m is chosen too large, and the method is unnecessarily expensive. In the specific case of the Arnoldi method Theorem A.1 may be used to determine the residual before calculating $\mathbf{c}$. Though this saves some computational time, it still requires the construction of the basis V_m and the tridiagonal matrix T_m, as well as a restart of the algorithm. This is not ideal, since the construction of V_m and T_m is expensive.

To address the issue of arbitrary m, we construct a version of algorithm A.2 that allows us to incrementally update the solution $\mathbf{u}_m$ and the residual vector $\mathbf{r}_m$. This way, we can stop the algorithm when the residual vector is smaller than some predefined threshold, like $\mathbf{r}_m < \epsilon$.

To that end, we start by performing a LU-factorisation of T_m given by

$$T_m = L_m U_m = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ \tilde{\eta}_2 & 1 & 0 & \dots & 0 \\ 0 & \tilde{\eta}_3 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \tilde{\eta}_m & 1 \end{pmatrix} \times \begin{pmatrix} \tilde{\delta}_1 & \eta_2 & 0 & \dots & 0 \\ 0 & \tilde{\delta}_2 & \eta_3 & \dots & 0 \\ 0 & 0 & \tilde{\delta}_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \eta_m \\ 0 & 0 & 0 & \dots & \tilde{\delta}_m \end{pmatrix} \tag{A.3}$$

Then, the approximate solution is given by

$$\begin{aligned}\mathbf{u}_m &= \mathbf{u}_0 + V_m\mathbf{c}\\ &= \mathbf{u}_0 + V_m U_m^{-1} L_m^{-1}\beta\mathbf{e}_1\\ &= \mathbf{u}_0 + V_m U_m^{-1}(L_m^{-1}\beta\mathbf{e}_1)\\ &= \mathbf{u}_0 + P_m\mathbf{z}_m,\end{aligned}$$

where $P_m = V_m U_m^{-1}$ and $\mathbf{z}_m = L_m^{-1}\beta\mathbf{e}_1$. Considering the definition of U_m in equation (A.3), we have that the m^{th} column of P_m is given by

$$\mathbf{p}_m = \frac{1}{\tilde{\delta}_m}\left[\mathbf{v}_m - \eta_m\mathbf{p}_{m-1}\right]. \tag{A.4}$$

Furthermore, from the LU factorization of T_m we have that

$$\begin{aligned}\tilde{\eta}_m &= \frac{\eta_m}{\tilde{\delta}_{m-1}},\\ \tilde{\delta}_m &= \delta_m - \tilde{\eta}_m\eta_m,\ m > 1.\end{aligned}$$

Now the solution can be incrementally updated by realizing that

$$\mathbf{z}_m = \begin{pmatrix}\mathbf{z}_{m-1}\\ \zeta_m\end{pmatrix} = \begin{pmatrix}\mathbf{z}_{m-2}\\ \zeta_{m-1}\\ \zeta_m\end{pmatrix},$$

and

$$L_m = \begin{pmatrix}L_{m-1} & \mathbf{0}_{m-1}\\ \mathbf{0}_{m-2}^T & \tilde{\eta}_m & 1\end{pmatrix}.$$

Then,

$$L_m\mathbf{z}_m = \begin{pmatrix}L_{m-1}\mathbf{z}_{m-1}\\ \tilde{\eta}_m\zeta_{m-1} + \zeta_m\end{pmatrix} = \begin{pmatrix}\beta\mathbf{e}_1\\ 0\end{pmatrix},$$

where the last equality follows from definition of $\mathbf{z}_m$. Consequently, we have that

$$\zeta_m = -\tilde{\eta}_m\zeta_{m-1}.$$

Finally, we obtain

$$\begin{aligned}u_m &= \mathbf{u}_0 + P_m\mathbf{z}_m\\ &= \mathbf{u}_0 + [P_{m-1}\mathbf{p}_m]\begin{pmatrix}\mathbf{z}_{m-1}\\ \zeta_m\end{pmatrix}\\ &= \mathbf{u}_0 + P_{m-1}\mathbf{z}_{m-1} + \mathbf{p}_m\zeta_m\\ &= \mathbf{u}_{m-1} + \mathbf{p}_m\zeta_m.\end{aligned}$$

Putting it all together, we obtain algorithm A.3.

Algorithm A.3 D-Lanczos [21, Algorithm 6.17]

$\mathbf{r}_0 = b - A\mathbf{u}_0$, $\beta = ||\mathbf{r}_0||_2$, $\mathbf{v}_1 = \mathbf{r}_0/\beta$
$\tilde{\eta}_1 = \beta_1 = 0$, $\mathbf{p}_0 = 0$
for $m = 1, 2, \ldots, m$ until convergence **do**
  $w = A\mathbf{v}_m - \beta_m \mathbf{v}_{m-1}$
  $\delta_m = (w, \mathbf{v}_m)$
  if $m > 1$ **then**
    $\tilde{\eta}_m = \frac{\beta_m}{\tilde{\delta}_{m-1}}$
    $\zeta_m = -\tilde{\eta}_m \zeta_{m-1}$
  end if
  $\tilde{\delta}_m = \delta_m - \tilde{\eta}_m \beta_m$
  $\mathbf{p}_m = \frac{1}{\tilde{\delta}_m}[\mathbf{v}_m - \beta_m \mathbf{p}_{m-1}]$
  $\mathbf{u}_m = \mathbf{u}_{m-1} + \mathbf{p}_m \zeta_m$
  if $\|\mathbf{r}_{m+1}\|_2 < \epsilon$ **then**
    break
  end if
  $\mathbf{w} = \mathbf{w} - \delta_m \mathbf{v}_m$
  $\beta_{m+1} = ||\mathbf{w}||_2$
  $\mathbf{v}_{m+1} = \mathbf{w}/\beta_{m+1}$
end for

Some core properties of algorithm A.3 are described in Theorem A.2

Theorem A.2: A-orthogonality of p_m

The vectors p_m produced in algorithm algorithm A.3 are A-orthogonal to each other.

Proof. We have

$$\begin{aligned} P_m^T A P_m &= U_m^{-T} V_m^T A V_m U_m^{-1} \\ &= U_m^{-T} T_m U_m^{-1} \\ &= U_m^{-T} L_m, \end{aligned}$$

where U_m^{-T} and L_m are both lower diagonal matrices. Their product must be symmetric, since $P_m^T A P_m$ is symmetric (due to the symmetry of A). The result follows from the fact that $U_m^{-T} L_m$ must be a diagonal matrix □

Theorem A.3: Lanczos recurrence relation

The Lanczos vectors are related through the Lanczos recurrence relation

$$\eta_{j+1}(A)\mathbf{v}_{j+1} = A\mathbf{v}_j - \delta_j \mathbf{v}_j - \eta_j \mathbf{v}_{j-1}. \tag{A.5}$$

Proof. This follows directly from the definition of T_m in equation (A.2) and the definition of the Hessenberg matrix in Definition 2.2. □

A.4. Derivation of CG

From general properties of error projection methods and observations made in the in algorithm A.3, we can derive the CG method. We start by constraining subsequent residuals r_j to be orthogonal. This follows from choosing subspaces $\mathcal{K} = \mathcal{L}$, as in the Arnoldi process. Again, the space $\mathcal{K}$ is spanned by the vectors $\mathbf{v}_m$. Thus setting $\mathbf{v}_1 = \mathbf{r}_0/\|\mathbf{r}_0\|_2$, automatically means subsequent residuals will be orthogonal to each other. Then, as suggested by Theorem A.2, we also require that the vectors p_j are A-orthogonal to each other. From this point on, we use the term *search direction* to refer to the vectors

p_j. Next to this we also introduce the CG variables α_j and β_j, which are the step size and the search direction update, respectively. This results in the following update equations

$$\mathbf{u}_{j+1} = \mathbf{u}_j + \alpha_j \mathbf{p}_j, \tag{A.6}$$

and, thereby,

$$\mathbf{r}_{j+1} = \mathbf{r}_j - \alpha_j A\mathbf{p}_j. \tag{A.7}$$

If the residuals are to be orthogonal, then

$$(\mathbf{r}_{j+1}, \mathbf{r}_j) = 0 \implies (\mathbf{r}_j - \alpha_j A\mathbf{p}_j, \mathbf{r}_j) = 0 \implies \alpha_j = \frac{(\mathbf{r}_j, \mathbf{r}_j)}{(A\mathbf{p}_j, \mathbf{r}_j)}.$$

Now, using the relation between $\mathbf{r}_m$ and $\mathbf{v}_{m+1}$ found in the proof of Theorem A.1 and equation (A.4), we can write the next search direction as a linear combination of the previous search direction and the next residual

$$\mathbf{p}_{j+1} = \mathbf{r}_{j+1} + \beta_j \mathbf{p}_j. \tag{A.8}$$

Substituting equation (A.8), we obtain

$$(A\mathbf{p}_{j+1}, \mathbf{r}_j) = (A\mathbf{p}_j, \mathbf{p}_j - \beta_{j-1}\mathbf{p}_{j-1}) = (A\mathbf{p}_j, \mathbf{p}_j),$$

since p_j is A-orthogonal to p_{j-1}. This allows us to write

$$\alpha_j = \frac{(\mathbf{r}_j, \mathbf{r}_j)}{(A\mathbf{p}_j, \mathbf{p}_j)}. \tag{A.9}$$

Additionally, taking the inner product with $A\mathbf{p}_j$ on both sides of equation (A.8) gives

$$\beta_j = \frac{(\mathbf{r}_{j+1}, A\mathbf{p}_j)}{(\mathbf{p}_j, A\mathbf{p}_j)}.$$

Now, rewriting equation (A.7) gives

$$A\mathbf{p}_j = \frac{1}{\alpha_j}(\mathbf{r}_j - \mathbf{r}_{j+1}),$$

which we substitute into the equation for β_j to obtain

$$\beta_j = \frac{1}{\alpha_j}\frac{(\mathbf{r}_{j+1}, (\mathbf{r}_{j+1} - \mathbf{r}_j))}{(A\mathbf{p}_j, \mathbf{r}_j)} = \frac{(\mathbf{r}_{j+1}, \mathbf{r}_{j+1})}{((\mathbf{r}_j - \mathbf{r}_{j-1}), \mathbf{r}_j)} = \frac{(\mathbf{r}_{j+1}, \mathbf{r}_{j+1})}{(\mathbf{r}_j, \mathbf{r}_j)}. \tag{A.10}$$

Finally, equations (A.6) to (A.10) comprise one iteration of the CG method, as shown in algorithm 2.

A.5. CG relation to Lanczos

There exist relations between the entries of T_m δ_j, η_j and the CG coefficients α_j, β_j. Namely, we have

$$\delta_{j+1} = \begin{cases} \dfrac{1}{\alpha_j} + \dfrac{\beta_{j-1}}{\alpha_{j-1}} & j > 0, \\ \dfrac{1}{\alpha_0} & j = 0, \end{cases} \tag{A.11}$$

and

$$\eta_{j+1} = \frac{\sqrt{\beta_{j-1}}}{\alpha_{j-1}}. \tag{A.12}$$

Here we have used the definition of T_m and the fact that the residuals are multiples of the Lanczos vectors $\mathbf{r}_j = \text{scalar} \times \mathbf{v}_j$ [21, Equation 6.103].

B. Blueprint for the Two-level Schwarz preconditioner

In this section we discuss the general idea behind as well as the construction of robust coarse spaces to be used in conjunction with the One-level Schwarz preconditioner, ultimately leading to a Two-level Schwarz preconditioner. To that end, we first motivate the need for an additional level or coarse space in section B.1 (Motivation) by studying the convergence factor of the original Schwarz method. In section B.2 (Construction of two-level additive Schwarz preconditioner) we construct the Schwarz preconditioner with both a Nicolaides coarse space from equation (2.27) and one based on the Dirichlet-to-Neumann map from Definition 2.9. The latter coarse space is constructed using the eigenfunctions corresponding to the smallest m_j eigenvalues resulting from a local eigenproblem in each subdomain Ω_j defined in equation (2.29). Finally, in section B.3 (Convergence of two-level additive Schwarz system) bounds for the two full preconditioned systems' condition numbers are provided. All of this can be found in [9, Sections 5.1-5.5].

B.1. Motivation

First we consider the convergence of the original Schwarz method stated in definition 2.8 for two simple one- and two-dimensional domains Ω. This motivates the construction of the coarse space.

1D case

Let $L > 0$ and the domain $\Omega = (0, L)$. The domain is split into two subdomains $\Omega_1 = (0, L_1)$ and $\Omega_2 = (l_2, L)$ such that $l_2 \leq L_1$. Instead of solving for $u_{1,2}$ directly, we solve for the error $e^n_{1,2} = u^n_{1,2} - u_{|\Omega_i}$, which by linearity of the Poisson problem as well as the original Schwarz algorithm satisfies

$$\begin{aligned} -\frac{e_1^{n+1}}{dx^2} &= 0 \text{ in } (0, L_1), \\ e_1^{n+1}(0) &= 0, \\ e_1^{n+1}(L_1) &= e_2^n(L_1); \end{aligned} \qquad \text{and} \qquad \begin{aligned} -\frac{e_2^{n+1}}{dx^2} &= 0 \text{ in } (l_2, L), \\ e_2^{n+1}(l_2) &= e_1^{n+1}(l_2), \\ e_2^{n+1}(L) &= 0. \end{aligned}$$

The solution to the error problem is

$$e_1^{n+1}(x) = \frac{x}{L_1} e_2^n(L_1), \quad e_2^{n+1}(x) = \frac{L - x}{L - l_2} e_1^{n+1}(l_2).$$

These functions increase linearly from the boundary of the domain to the boundary of the overlapping region. The error at $x = L_1$ is updated as

$$e_2^{n+1}(L_1) = \frac{1 - \delta/(L - l_2)}{1 + \delta/l_2} e_2^n(L_1),$$

where $\delta = L_1 - l_2 > 0$ is the overlap. The error is reduced by a factor of

$$\rho_{1\mathsf{D}} = \frac{1 - \delta/(L - l_2)}{1 + \delta/l_2}, \tag{B.1}$$

which indicates the convergence becomes quicker as the overlap increases [9, Section 1.5.1].

2D case

In the 2D case two half planes are considered $\Omega_1 = (-\infty, \delta) \times \mathbb{R}$ and $\Omega_2 = (\delta, \infty) \times \mathbb{R}$. Following the example of [9, Section 1.5.2] the problem is

$$\begin{aligned} -(\eta - \Delta)u &= f \text{ in } \mathbb{R}^2, \\ u &\text{ bounded at infinity}, \end{aligned}$$

where $\eta > 0$ is a constant. Proceeding in similar fashion as the one-dimensional case, the error $e^{n+1}_{1,2}$ can be solved for in the two subdomains. This is done via a partial Fourier transform of the problem in the y-direction yielding an ODE for the transformed error $\hat{e}^{n+1}_{1,2}$ with the added Fourier constant k, which can be solved explicitly with the ansatz

$$\hat{e}_{1,2}^{n+1}(x, k) = \gamma_1(k) e^{\lambda_+(k)x} + \gamma_2(k) e^{\lambda_-(k)x},$$

where $\lambda_\pm(k) = \pm\sqrt{k^2+\eta}$. By using the interface conditions $\hat{e}_1^{n+1}(0,k) = \hat{e}_2^{n+1}(0,k)$ we get

$$\gamma_i^{n+1}(k) = \rho(k;\eta,\delta)^2\gamma_i^{n-1}(k),$$

such that the convergence factor is [9, Equation 1.36]

$$\rho_{\text{2D}}(k;\eta,\delta) = e^{-\delta\sqrt{\eta+k^2}} \tag{B.2}$$

which indicates that the convergence is quicker as the overlap increases as before. Next to this, it also shows that the convergence is quicker for higher k.

B.2. Construction of two-level additive Schwarz preconditioner

Here we present the construction of a full two-level Schwarz preconditioner. We partition Ω into N_{sub} subdomains Ω_j, which overlap each other by one or several layers of elements in the triangulation $\mathcal{T}$. We make the following general assumptions.

D1 For every degree of freedom $k \in \mathcal{N}$, there is a subdomain Ω_j such that ϕ_k has support in Ω_j [9, Lemma 5.3].

D2 The maximum number of subdomains a mesh element can belong to is given by

$$k_0 = \max_{\tau\in\mathcal{T}} \left(|\{j|1 \le j \le N_{\text{sub}} \text{ and } \tau \subset \Omega_j\}|\right).$$

D3 The minimum number of colors needed to color all subdomains so that no two adjacent subdomains have the same color is given by

$$N_c \ge k_0$$

D4 The minimum overlap for any subdomain Ω_j with any of its neighboring subdomains is given by

$$\delta_j = \inf_{x\in\Omega_j\setminus\cup_{i\neq j}\bar{\Omega}_i} \text{dist}(x, \partial\Omega_j \setminus \partial\Omega).$$

D5 The partition of unity functions $\{\chi_j\}_{j=1}^{N_{\text{sub}}} \subset V_h$ are such that

D5.a $\chi_j(x) \in [0,1], \quad \forall x \in \bar{\Omega}, j = 1,\dots,N_{\text{sub}}$,

D5.b $\text{supp}(\chi_j) \subset \bar{\Omega}_j$,

D5.c $\sum_{j=1}^{N_{\text{sub}}} \chi_j(x) = 1, \quad \forall x \in \bar{\Omega}$,

D5.d $\|\nabla\chi_j(x)\| \le \frac{C_\chi}{\delta_j}$,

and are given by

$$\chi_j(x) = I_h\left(\frac{d_j(x)}{\sum_{j=1}^{N_{\text{sub}}} d_j(x)}\right),$$

where

$$d_j(x) = \begin{cases} \text{dist}(x,\partial\Omega_j), & x \in \Omega_j, \\ 0, & x \in \Omega\setminus\Omega_j. \end{cases}$$

D6 The overlap region for any subdomain is given by

$$\Omega_j^\delta = \{x \in \Omega_j | \chi_j < 1\}.$$

From item **D1** it follows that the extension operator $E_j : V_{h,0}(\Omega_j) \to V_h$ can defined by

$$V_h = \sum_{j=1}^{N_{\text{sub}}} E_j V_{h,0}(\Omega_j).$$

Note that using the extension operator we can show that all the local bilinear forms are positive definite as

$$a_{\Omega_j}(v, w) = a(E_j v, E_j w) \geq \alpha \|E_j v\|_a^2, \quad \forall v, w \in V_{h,0}(\Omega_j),$$

and a is positive definite.

Finally, we define the a-symmetric projection operators $\tilde{\mathcal{P}}_j : V_{h,0} \to V_h$ and $\mathcal{P}_j : V_h \to V_h$ defined by

$$\begin{aligned} a_{\Omega_j}(\tilde{\mathcal{P}}_j u, v_j) &= a(u, E_j v_j) \quad \forall v_j \in V_{h,0}, \\ \mathcal{P} &= E_j \tilde{\mathcal{P}}_j. \end{aligned}$$

Then their matrix counterparts are given by

$$\begin{aligned} \tilde{P}_j &= A_j^{-1} R_j^T A, \\ P_j &= R_j^T A_j^{-1} R_j^T A, \end{aligned}$$

where $A_j = R_j A R_j^T$ and its inverse is obtained using an exact solver. From this we can construct the two-level additive Schwarz preconditioned system as

$$M_{\text{ASM,2}}^{-1} A = \sum_{j=1}^{N_{\text{sub}}} P_j. \tag{B.3}$$

B.3. Convergence of two-level additive Schwarz system

In the following, we denote

$$\mathcal{P}_{\text{ad}} = \sum_{j=1}^{N_{\text{sub}}} \mathcal{P}_j,$$

and correspondingly,

$$P_{\text{ad}} = \sum_{j=1}^{N_{\text{sub}}} P_j.$$

In the context of this thesis the two-level additive Schwarz method is used in combination with a Krylov subspace method, in which case convergence rate depends on the entire spectrum of eigenvalues, as discussed in section 2.1.4. However, an upperbound for the convergence rate can be derived from the condition number of P_{ad} using Theorem 2.7.

Using the fact that P_{ad} is symmetric (see [9, Lemma 5.8]) with respect to the a-norm, we can write

$$\kappa(P_{\text{ad}}) = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}},$$

where

$$\lambda_{\text{max}} = \sup_{v \in V_h} \frac{a(\mathcal{P}_{\text{ad}})}{a(v, v)}, \quad \lambda_{\text{min}} = \inf_{v \in V_h} \frac{a(\mathcal{P}_{\text{ad}})}{a(v, v)}.$$

Additionally, we can employ the a-orthogonality of the projection operators to get

$$\frac{a(\mathcal{P}_j u, u)}{\|u\|_a^2} = \frac{a(\mathcal{P}_j u, \mathcal{P}_j u)}{\|u\|_a^2} \leq 1.$$

Going further, we can pose that the projection operators defined by the sum of projection operators $\mathcal{P}_j$ of like-colored subdomains are a-orthogonal to each other, since the partition of unity functions χ_j are zero on their shared interfaces (see item **D3**). To that end, define

$$\mathcal{P}_{\Theta_i} = \sum_{j \in \Theta_i} \mathcal{P}_j,$$

where Θ_i is the set of indices of subdomains with color i and $i = 1, \dots, N_c$. Then, we can write [9, Lemma 5.9]

$$\begin{aligned}\lambda_{\max}(\mathcal{P}_{\text{ad}}) &= \sup_{v \in V_h} \sum_{i=1}^{N_c} \frac{a(\mathcal{P}_{\Theta_i} v, v)}{a(v,v)} \\ &\leq \sum_{i=1}^{N_c} \sup_{v \in V_h} \frac{a(\mathcal{P}_{\Theta_i} v, v)}{a(v,v)} \\ &\leq N_c + 1,\end{aligned}$$

where the extra one comes from the coarse projection operator $\mathcal{P}_0$. Note that this bound can be made sharper by using item **D2** to get $\lambda_{\max}(\mathcal{P}_{\Theta_i}) \leq k_0 + 1$.

Next, we define a stable decomposition [9, Definition 5.10].

Definition B.1: C_0-stable decomposition (uniform)

A function $v \in V_h$ is said to admit a C_0-stable decomposition if there exists a uniform constant $C_0 > 0$ such that

$$\sum_{j=0}^{N_{\text{sub}}} \|E_j v\|_a^2 \leq C_0^2 \|v\|_a^2$$

It can be shown that the minimum eigenvalue satisfies

$$\lambda_{\min}(\mathcal{P}_{\text{ad}}) \geq C_0^{-2},$$

provided that every $v \in V_h$ admits a C_0-stable decomposition in the sense of Definition B.1 [9, Theorem 5.11]. Note that through the equivalence of norms, we have

$$\mathcal{C}_{\min} \|\nabla v\|^2 \leq \|v\|_a^2 \leq \mathcal{C}_{\max} \|\nabla v\|^2.$$

Therefore, we can conclude that every $v \in V_h$ admits a C_0-stable decomposition if and only if

$$\sum_{j=0}^{N_{\text{sub}}} \|E_j \nabla v\|^2 \leq \frac{\mathcal{C}_{\max}}{\mathcal{C}_{\min}} C_0^2 \|\nabla v\|^2.$$

Finally, we can write the condition number of the two-level additive Schwarz preconditioner as

$$\kappa(P_{\text{ad}}) \leq \frac{\mathcal{C}_{\max}}{\mathcal{C}_{\min}} (N_c + 1) C_0^2. \tag{B.4}$$

The constant C_0 depends on the projection operator Π_j onto the chosen coarse space V_0 for each subdomain and is fully derived in [9, Sections 5.5-5.7]. We present the main results below.

I. **Nicolaides coarse space** The projection operator is defined as

$$\Pi_j^{\text{Nico}} u = \begin{cases} \left(\dfrac{1}{|\Omega_j|} \displaystyle\int_{\Omega_j} u \right) \mathbf{1}_{\Omega_j}, & \delta\Omega_j \cap \delta\Omega = \emptyset, \\ 0, & \text{otherwise}, \end{cases} \tag{B.5}$$

which are simply the averages of the function over the subdomain Ω_j and gives rise to the following basis functions in $V_{h,0}$

$$\Phi_j^{\text{Nico}} = I_h(\chi_j \mathbf{1}_{\Omega_j}).$$

Then,

$$V_0 = \text{span}\{\Phi_j^{\text{Nico}}\}_{j=1}^{N_{\text{sub}}},$$

and

$$\dim V_0 = \text{the number of floating subdomains},$$

that is the number of subdomains that are not connected to the boundary of the domain Ω. In this case [9, Theorem 5.16],

$$C_{0,\text{Nico}}^2 = \left(8 + 8C_\chi^2 \max_{j=1}^{N_{\text{sub}}} \left[C_P^2 + C_{\text{tr}}^{-1} \frac{H_j}{\delta_j}\right] k_0 C_{I_h}(k_0+1) + 1\right), \tag{B.6}$$

where H_j is the diameter of the subdomain Ω_j, C_χ the partition of unity constant from item **D5**, C_p the Poincaré constant following from [9, Lemma 5.18], C_{tr} the trace constant and C_{I_h} the stable interpolation constant.

II. **Local eigenfunctions coarse space** The projection operator is defined as

$$\Pi_j^{\text{spec}} u = \sum_{k=1}^{m_j} a_{\Omega_j}(u, v_k^{(j)}) v_k^{(j)},$$

where $v_k^{(j)}$ is the k^{th} eigenfunction resulting from the eigenproblem in equation (2.29). The basis functions in $V_{h,0}$ are then given by

$$\Phi_{j,k}^{\text{spec}} = I_h(\chi_j v_k^{(j)}),$$

resulting in the coarse space

$$V_0 = \text{span}\{\Phi_{j,k}^{\text{spec}}\}_{j=1,k=1}^{N_{\text{sub}},m_j},$$

with dimension

$$\dim V_0 = \sum_{j=1}^{N_{\text{sub}}} m_j.$$

In this case [9, Theorem 5.17]

$$C_{0,\text{DtN}}^2 = \left(8 + 8C_\chi^2 \max_{j=1}^{N_{\text{sub}}} \left[C_P^2 + C_{\text{tr}}^{-1} \frac{1}{\delta_j \lambda_{m_j+1}}\right] k_0 C_{I_h}(k_0+1) + 1\right). \tag{B.7}$$

C. Chebyshev approximation

This section contains the definition of Chebyshev polynomials of the first kind, some of their properties and their application to a minimization problem akin to the one described in Theorem 2.7.

C.1. Chebyshev polynomials

This section introduces Chebyshev polynomials and some of their properties. First, their definition in Definition C.1.

Definition C.1: Chebyshev polynomial

The m^{th} degree Chebyshev polynomial of the first kind is denoted as C_m, for $z \in \mathbb{C}$

$$C_m(z) = \begin{cases} \cos(m \cos^{-1}(z)), & |z| \leq 1, \\ \cosh(m \cosh^{-1}(z)), & |z| > 1, \end{cases},$$

as well as through the recurrence relation

$$C_m(z) = 2zC_{m-1}(z) - C_{m-2}(z), \quad m \geq 2,$$

with initial conditions

$$C_0(z) = 1, \quad C_1(z) = z.$$

For $|z| > 1$ we can also write

$$C_m(z) = \frac{1}{2}\left(\left(z + \sqrt{z^2-1}\right)^m + \left(z - \sqrt{z^2-1}\right)^m\right), \tag{C.1}$$

which may be approximated as

$$C_m(z) \approx \begin{cases} \frac{1}{2}\left(z+\sqrt{z^2-1}\right)^m, & \Re\{z\} > 1, \\ \frac{1}{2}\left(z-\sqrt{z^2-1}\right)^m, & \Re\{z\} < -1. \end{cases} \tag{C.2}$$

The extreme points of C_m are given by

$$z_k = \cos\left(\frac{k\pi}{m}\right), \quad k = 0, 1, \ldots, m. \tag{C.3}$$

Indeed, substituting z_k into C_m gives

$$C_m(z_k) = \cos(k\pi) = \cos(k\pi) = (-1)^k, \quad k = 0, 1, \ldots, m. \tag{C.4}$$

For the optimality proof we need to introduce the real-valued, transformed Chebyshev polynomial in Definition C.2.

Definition C.2: Real Transformed Chebyshev polynomial

The transformed Chebyshev polynomial of the first kind is denoted as $\hat{C}_m$, for $x \in \mathbb{R}$ is obtained through an affine change of variables T from the $[a, b] \subset \mathbb{R}$ to the interval $[-1, 1]$ as

$$t \in [a, b] \implies z = T(t) = \frac{2t-(a+b)}{b-a} \in [-1, 1],$$

and a subsequent scaling with the factor $C_m(T(\gamma))$ for $\gamma \in \mathbb{R}$ outside the interval $[a, b]$ as

$$\hat{C}_m(t) = \frac{C_m(T(t))}{C_m(T(\gamma))} = \frac{C_m\left(\frac{2t-(a+b)}{b-a}\right)}{C_m\left(\frac{2\gamma-(a+b)}{b-a}\right)}$$

Lastly, by equation (C.4) we get for $t_k = T^{-1}(z_k)$

$$\hat{C}_m(t_k) = \frac{(-1)^k}{C_m(T(\gamma))} = \frac{(-1)^k}{d_m(\gamma)}, \tag{C.5}$$

where $d_m(\gamma) = C_m(T(\gamma))$.

C.2. Chebyshev optimality

We now show that $\hat{C}_m$ from Definition C.2 is the solution of the following minimization problem

Theorem C.1: Min-max polynomial

The real-valued polynomial $p_m(t)$ of degree m such that for $\gamma \in \mathbb{R}$ outside the interval $[a, b]$ the following holds

$$\min_{p \in \mathcal{P}_m, p(\gamma)=1} \max_{t \in [a,b]} |p(t)|,$$

is given by the Chebyshev polynomial $\hat{C}_m$. Furthermore, we have

$$\min_{p \in \mathcal{P}_m, p(\gamma)=1} \max_{t \in [a,b]} |p_m(t)| = \frac{1}{d_m(\gamma)},$$

where $d_m(\gamma)$ is as in equation (C.5).

Proof. We proof this by contradiction. First, note that by equation (C.5) we have

$$\max_{t \in [a,b]} |\hat{C}_m(t)| = \max_{k=0,1,\ldots,m} |\hat{C}_m(t_k)| = \frac{1}{d_m},$$

Assume that there exists a polynomial $w_m(t)$ of degree m such that

$$\max_{t\in[a,b]} |w_m(t)| < \frac{1}{d_m}.$$

Without loss of generality we can assume w_m, just like $\hat{C}_m$, is a monic polynomial, i.e. $w_m(t) = t^m + \dots$. We now define the difference polynomial

$$f_m(t) = \hat{C}_m(t) - w_m(t) \in \mathcal{P}_{m-1},$$

where the inclusion in the $m-1$ degree polynomials follows from the fact that f_m is the difference of two monic polynomials of degree m.

Consider the values of f_m at the extreme points $t_k = T^{-1}(z_k)$ of $\hat{C}_m$ with z_k as in equation (C.3) and T as in Definition C.2. We distinguish between even and odd k. By equation (C.5) and the assumption on w_m we then obtain

even k: $f_m(t_k) = \dfrac{1}{d_m} - w_m(t_k) > 0.$

odd k: $f_m(t_k) = -\dfrac{1}{d_m} - w_m(t_k) < 0.$

From this we gather that $f_m(t_k)$ has alternating signs at the extreme points t_k of $\hat{C}_m$.

Now, the sequence $(z_k)_{k=0}^m$ is decreasing, and thus the sequence $(t_k)_{k=0}^m$ is also decreasing. This means that we can construct m distinct intervals $I_k = [t_{k+1}, t_k]$ such that f_m switches sign in each interval. By the intermediate value theorem, we know that f_m must have at least one root in each interval I_k. This leads to the conclusion that f_m has at least m roots in the interval $[a, b]$.

However, by the fundamental theorem of algebra f_m, a polynomial of degree $m-1$, can have at most $m-1$ distinct roots. The only possibility is for $f_m \equiv 0$, but then we have

$$\max_{t\in[a,b]} |w_m(t)| = \max_{t\in[a,b]} |\hat{C}_m(t)| = \frac{1}{d_m},$$

which contradicts our main assumption that w_m is a polynomial such that $\max_{t\in[a,b]} |w_m(t)| < \dfrac{1}{d_m}$. Thus, we conclude that the Chebyshev polynomial $\hat{C}_m$ is indeed the solution to the minimization problem. □

D. Rayleigh quotient

Definition D.1: T

e Rayleigh quotient of a matrix A and a vector $\mathbf{u}$ is defined as

$$R(A, \mathbf{u}) = \frac{\mathbf{u}^T A \mathbf{u}}{\mathbf{u}^T \mathbf{u}}. \tag{D.1}$$

Theorem D.1: Rayleigh quotient bound

Suppose A is symmetric. Then the Rayleigh quotient $R(A, \mathbf{u})$ is bounded by the smallest and largest eigenvalue of A, i.e.

$$\lambda_{\min} \leq R(A, \mathbf{u}) \leq \lambda_{\max}.$$

Proof. A has diagonalization $A = Q\Lambda Q^T$, where Q is an orthonormal eigenbasis and Λ is the diagonal with real and positive eigenvalues λ_i of A. The Rayleigh quotient satisfies with $\mathbf{v} = Q\mathbf{u}$

$$R(A, \mathbf{u}) = \frac{\mathbf{u}^T A \mathbf{u}}{\mathbf{u}^T \mathbf{u}} = \frac{\mathbf{v}^T \Lambda \mathbf{v}}{\mathbf{v}^T \mathbf{v}} = \sum_{i=1}^{n} \frac{\lambda_i v_i^2}{\|v\|_2},$$

which is a convex combination of the eigenvalues λ_i of A. Thus, we have

$$\lambda_{\min} \leq R(A, \mathbf{u}) \leq \lambda_{\max}.$$

□

Bibliography

[1] Filipe A. C. S. Alves, Alexander Heinlein, and Hadi Hajibeygi. *A computational study of algebraic coarse spaces for two-level overlapping additive Schwarz preconditioners*. 2024. arXiv: 2408.08187 [math.NA]. URL (cit. on pp. 3, 4, 27, 46, 56).

[2] O. Axelsson. "A class of iterative methods for finite element equations". In: *Computer Methods in Applied Mechanics and Engineering* 9.2 (1976), pp. 123–137. ISSN: 0045-7825. DOI: https://doi.org/10.1016/0045-7825(76)90056-6. URL (cit. on pp. 27, 28, 30–32, 38, 56).

[3] B. Beckermann and A. B. J. Kuijlaars. "On The Sharpness of an Asymptotic Error Estimate for Conjugate Gradients". In: *BIT Numerical Mathematics* 41.5 (2001), pp. 856–867 (cit. on p. 28).

[4] Bernhard Beckermann and Arno B. J. Kuijlaars. "Superlinear CG Convergence for Special Right-Hand Sides". In: *Electronic Transactions on Numerical Analysis* 14 (2002), pp. 1–19. ISSN: 1068-9613. URL (cit. on p. 28).

[5] Bernhard Beckermann and Arno B. J. Kuijlaars. "Superlinear Convergence of Conjugate Gradients". In: *SIAM Journal on Numerical Analysis* 39.1 (2001), pp. 300–329. DOI: 10.1137/S0036142999363188. eprint: https://doi.org/10.1137/S0036142999363188. URL (cit. on p. 28).

[6] Robert Corless et al. "On the Lambert W Function". In: *Advances in Computational Mathematics* 5 (Jan. 1996), pp. 329–359. DOI: 10.1007/BF02124750 (cit. on p. 35).

[7] Clark R. Dohrmann, Axel Klawonn, and Olof B. Widlund. "A family of energy minimizing coarse spaces for overlapping Schwarz preconditioners". In: *Domain Decomposition Methods in Science and Engineering XVII*. Vol. 60. Lecture Notes in Computational Science and Engineering. St. Wolfgang / Strobl, Austria, 2008, pp. 247–254 (cit. on p. 26).

[8] Clark R. Dohrmann and Olof B. Widlund. "On the Design of Small Coarse Spaces for Domain Decomposition Algorithms". In: *SIAM Journal on Scientific Computing* 39.4 (2017), A1466–A1488. DOI: 10.1137/17M1114272. eprint: https://doi.org/10.1137/17M1114272. URL (cit. on p. 27).

[9] Victorita Dolean, Pierre Jolivet, and Frédéric Nataf. *An Introduction to Domain Decomposition Methods*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2015. DOI: 10.1137/1.9781611974065. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611974065. URL (cit. on pp. 19–23, 63–67).

[10] Tobin A. Driscoll, Kim-Chuan Toh, and Lloyd N. Trefethen. "From Potential Theory to Matrix Iterations in Six Steps". In: *SIAM Review* 40.3 (1998), pp. 547–578. DOI: 10.1137/S0036144596305582. eprint: https://doi.org/10.1137/S0036144596305582. URL (cit. on p. 28).

[11] Yalchin Efendiev, Juan Galvis, and Xiao-Hui Wu. "Multiscale finite element methods for high-contrast problems using local spectral basis functions". In: *Journal of Computational Physics* 230.4 (2011), pp. 937–955. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2010.09.026. URL (cit. on p. 25).

[12] Juan Galvis and Yalchin Efendiev. "Domain Decomposition Preconditioners for Multiscale Flows in High-Contrast Media". In: *Multiscale Modeling & Simulation* 8.4 (2010), pp. 1461–1483. DOI: 10.1137/090751190. eprint: https://doi.org/10.1137/090751190. URL (cit. on p. 25).

[13] I. G. Graham, P. O. Lechner, and R. Scheichl. "Domain decomposition for multiscale PDEs". In: *Numerische Mathematik* 106 (2007), pp. 589–626. DOI: 10.1007/s00211-007-0074-1. URL (cit. on pp. 24–26).

[14] Alexander Heinlein. "Multiscale coarse spaces for overlapping Schwarz methods based on the ACMS space in 2D". en. In: (2018). Ed. by Lothar Reichel (Hg.) Ronny Ramlau, pp. 156–182. ISSN: 1068-9613. URL (cit. on p. 26).

[15] Alexander Heinlein et al. "Adaptive GDSW Coarse Spaces of Reduced Dimension for Overlapping Schwarz Methods". In: *SIAM Journal on Scientific Computing* 44.3 (2022), A1176–A1204. DOI: 10.1137/20M1364540. eprint: https://doi.org/10.1137/20M1364540. URL (cit. on p. 27).

[16] Thomas Y. Hou and Xiao-Hui Wu. “A Multiscale Finite Element Method for Elliptic Problems in Composite Materials and Porous Media”. In: *J. Comput. Phys.* 134.1 (June 1997), pp. 169–189. ISSN: 0021-9991. DOI: 10.1006/jcph.1997.5682. URL (cit. on pp. 25, 26).

[17] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Society for Industrial and Applied Mathematics, 1995. DOI: 10.1137/1.9781611970944. eprint: https://epubs.siam.org/doi/pdf/10.1137/1.9781611970944. URL (cit. on p. 16).

[18] P. L. Lions. “On the Schwarz Alternating Method III: A Variant for Nonoverlapping Subdomains”. In: *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*. Ed. by Tony F. Chan et al. Society for Industrial and Applied Mathematics, 1990. Chap. 11, pp. 202–223. ISBN: 9780898712537. URL (cit. on p. 20).

[19] Ivan Lunati and Seong H. Lee. “An Operator Formulation of the Multiscale Finite-Volume Method with Correction Function”. In: *Multiscale Modeling & Simulation* 8.1 (2009), pp. 96–109. DOI: 10.1137/080742117. eprint: https://doi.org/10.1137/080742117. URL (cit. on p. 27).

[20] Gérard Meurant and Zdeněk Strakoš. “The Lanczos and conjugate gradient algorithms in finite precision arithmetic”. In: *Acta Numerica* 15 (2006), pp. 471–542. DOI: 10.1017/S096249290626001X (cit. on p. 15).

[21] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. Second. Society for Industrial and Applied Mathematics, 2003. DOI: 10.1137/1.9780898718003. eprint: https://epubs.siam.org/doi/book/10.1137/1.9780898718003. URL (cit. on pp. 5–8, 19, 58, 59, 61, 62).

[22] H.A. Schwarz. In: *Journal für die reine und angewandte Mathematik* 1869.70 (1869), pp. 105–120. DOI: doi:10.1515/crll.1869.70.105. URL (cit. on p. 20).

[23] Z. Strakoš. “On the real convergence rate of the conjugate gradient method”. In: *Linear Algebra and its Applications* 154-156 (1991), pp. 535–549. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(91)90393-B. URL (cit. on p. 27).

[24] Y. Wang, H. Hajibeygi, and H.A. Tchelepi. “Algebraic multiscale solver for flow in heterogeneous porous media”. English. In: *Journal of Computational Physics* 259.February (2014). Harvest, pp. 284–303. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2013.11.024 (cit. on p. 27).