TU Delft Delft University of Technology

Convergence Analysis of Domain Decomposition Methods for the Helmholtz Equation

by

Boaz Varkevisser

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on Thursday July 17, 2025 at 13:00 AM.

Student number:	5666562	
Project duration:	March 4, 2025 – July 7, 2025	
Thesis committee:	Dr. ir. mr. V. Dwarka,	TU Delft, supervisor
	Prof. dr. H. Schuttelaars,	TU Delft

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

TU Delft

Contents

Laymen's Summary

The Helmholtz equation is a famous equation in the world of physics with which the behavior of waves can be described. The equation is frequently used when studying seismic waves from earthquakes or electromagnetic waves of an MRI. It is also infamous for the difficulties that arise when trying to solve this equation with a computer. A great deal of research has been conducted over the last decades in order to find and improve numerical solution methods. As of today, no method is known that is feasible for general Helmholtz problems. This thesis investigates the root cause of problems when trying to solve the Helmholtz equation with domain decomposition methods. To this end, we work with the most basic form of the Helmholtz equation, which we will first construct. We then perform mathematical analysis to gain some insights into the root cause of problems. Numerical experiments are also conducted in order to investigate the behavior of the mathematical models.

Summary

The Helmholtz equation is used in the field of physics to model the behavior of waves in a 1D, 2D or 3D medium. The equation has gained much infamy within the community of numerical mathematics. Researchers have yet to find a numerical solution model that can give a feasible solution for the general Helmholtz problem. Difficulties are caused by wave number dependence of the numerical schemes that are used to model the Helmholtz equation. Large wave numbers bring forth a pollution error within the numerical solution, which means that the deviation from the real solution is increased. In the end, this causes current numerical methods to be computationally expensive for large wave numbers.

In this thesis, we are interested in the behavior of solutions when using domain decomposition methods with the generalized minimum residual (GMRES) solver. Eigenvalues of (preconditioned) systems rush to zero for large wave numbers, which causes an increase in the number of iterations that are needed for GMRES to obtain an accurate solution. Our main goal is to investigate the root cause of the appearance of near-zero eigenvalues when using domain decomposition methods. The domain decomposition is among the more popular methods, along with CSLP and deflation. For the latter methods, the cause of problems has been found, while for domain decomposition, this is still unknown.

We first build a simple numerical model for the Helmholtz equation, based on a finite difference scheme. The model is then approximated with a preconditioner based on domain decomposition and a coarse space. The simple model allows for theoretical analysis on eigenvalue behavior. The analysis mainly revolves around the computation of close-to-zero eigenvalues, but will also include a short investigation on the field of values and angles between subspaces. Numerical experiments will be performed to further investigate the behavior of the eigenvalues.

1 Introduction

Over the last decades, the Helmholtz equation has become infamous in the world of numerical mathematics. Applied in fields such as seismology, acoustics and electromagnetism, the Helmholtz equation is used to study the behavior of wave phenomena in a 1D, 2D or 3D domain. A great deal of research has been conducted in order to obtain reliable numerical solutions for this equation. A number of different solution methods have been constructed, but there is still some difficulty in finding one that is satisfactory for the general Helmholtz problem [5]. The challenge in constructing efficient numerical solvers lies in obtaining wave number independence in order to avoid the so-called pollution error. This error denotes the difference between the exact and the numerical solution. It turns out that the numerical solution to the Helmholtz equation heavily relies on the wave number, and trouble arises when this number grows. The bigger the wave number, the more the pollution error grows and the lower the convergence speed of numerical solvers becomes. In other words, computing solutions becomes increasingly expensive when the wave number grows.

Research into obtaining wave number independence predominantly revolves around the application of preconditioners. These are used in order to obtain more favorable linear system of equations that arises from numerically modeling the Helmholtz equation. Some of the most notable preconditioners are the deflation preconditioner, the complex shifted Laplacian preconditioner and the domain decomposition preconditioner (also called Schwarz preconditioner). This thesis treats the two-level Additive Schwarz (AS2) preconditioner. This involves a deconstruction of the domain into subdomains, as well as a coarsening of the domain. The (preconditioned) systems are often solved with the generalized minimum residual (GMRES) method. Even though preconditioners help improve the speed at which GMRES finds an accurate solution, problems still occur when the wave number increases. Eigenvalues of the preconditioners start to move towards the origin, which has a negative effect on the convergence of GMRES [6].

It is currently not known what causes the undesired eigenvalue behavior when working with domain decomposition methods. However, for the deflation method, the root cause of problems was recently recently discovered to revolve around matrix alignment [6]. This leads us to believe that for domain decomposition, the problem is either of the same nature, or inherent to the method. We suspect that the problem lies within the subdomains in which a domain is decomposed. The main question that we will try to answer is therefore: *How do subdomains in domain decomposition methods influence the rate of convergence of the GMRES solver for the Helmholtz equation?* To this end we will also treat the following questions:

- What information do eigenvalues of coefficient matrices give about GMRES convergence?
- How does the size/number of subdomains influence GMRES convergence
- How does a coarse space relate to the subdomains in terms of matrix alignment?

The behavior of the eigenvalues of the preconditioners and the coefficient matrices that arise from the linear systems are of main interest. We use a simple 1D Helmholtz model with Dirichlet boundary conditions for which the most mathematical theory is available. We also analyze the behavior of the GMRES solver with respect to the wave number when using domain decomposition preconditioners. Specifically, we analyze scalability. A numerical solver is called 'numerically scalable' if GMRES convergence does not depend on the size of the grid that is used to approximate the domain. Moreover, a numerical solver is called 'parallel

scalable' if GMRES convergence does not depend on the number of subdomains. All the analyses involve the use of domain decomposition preconditioners and coarse spaces based on interpolation methods.

This thesis is structured as follows. In Chapter 2 we will define and set up the numerical model for the Helmholtz equation, and the problems that arise are further discussed. Some different approaches to solving the Helmholtz equation will also be mentioned. In the third chapter, the focus is shifted towards the analysis of the simple model. We investigate the near-zero eigenvalues of the coefficient matrices and the AS2 preconditioner. Here, we expect to find a relation between the near-zero eigenvalues and the wave number, or bounds for the near-zero eigenvalues depending on the wave number. Finally, in Chapter 4, numerical results are provided to support the analysis from Chapter 3 and to investigate GMRES convergence.

2

Numerical Models

In this chapter we will discuss the model that will be used for this research, as well as the numerical methods. We will start with the 1D Helmholtz problem with Dirichlet boundary conditions.

$$\begin{cases} -\frac{d^2u(x)}{dx^2} - k^2 u(x) = f(x), & x \in \Omega = [0, L] \\ u(x) = 0, & x \in \partial\Omega \end{cases} \tag{2.1}$$

Here, $f(x)$ denotes some source function, and $k \in \mathbb{N}\backslash\{0\}$ is the wave number. This model will be used throughout this work and we will construct a numerical model using finite differences.

Oftentimes, Sommerfeld radiation conditions (also known as absorbing boundary conditions) are used to allow appropriate wave behavior on the domain Ω [1–3]. For theoretical analysis, however, these boundary conditions come with a downside, which will be made clear in the next section. When using Sommerfeld radiation conditions, the model becomes as follows.

$$\begin{cases} -\frac{d^2u(x)}{dx^2} - k^2 u(x) = f(x), & x \in \Omega = [0, L] \\ \frac{\partial u(x)}{\partial \mathbf{n}} - iku(x) = 0, & x \in \partial\Omega \end{cases} \tag{2.2}$$

with $\mathbf{n}$ being a unit normal vector of Ω and $i = \sqrt{-1}$.

2.1. The Finite Difference Method

We turn back to model (2.1). For the sake of simplicity, we use domain $\Omega = [0, 1]$. The domain is represented by a grid of $n+2$ equidistant nodes. These nodes are given by $x_i = ih$, where $h = \frac{1}{n+1}$. With the finite difference method, the mathematical model is rewritten for each $i \in \{1, 2, ..., n\}$ as

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} - k^2 u_i = f_i \tag{2.3}$$

where u_i is the solution approximated in node x_i, and $f_i = f(x_i)$. It follows from the boundary condition that $u_0 = u_{n+1} = 0$. Hence the solutions will be approximated on grid $G = \{x_1, ..., x_n\}$. Now, from (2.3) and setting $\mathbf{u} = (u_1, u_2, ..., u_n)$ and $\mathbf{f} = (f_1, f_2, ...f_n)$, we obtain the following system of linear equations:

$$A\mathbf{u} = \mathbf{f} \tag{2.4}$$

where we have

$$A = \frac{1}{h^2}\begin{pmatrix} 2-k^2h^2 & -1 & 0 & 0 & \cdots & 0 \\ -1 & 2-k^2h^2 & -1 & 0 & & \\ 0 & -1 & 2-k^2h^2 & -1 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ & & & & & \\ 0 & \cdots & \cdots & 0 & -1 & 2-k^2h^2 \end{pmatrix}$$

When solving 2.4 numerically, the pollution effect plays a significant role for large wave numbers [4, 15]. In order to decrease the pollution error, a general rule of thumb is to take 10 grid points per wavelength, i.e. taking $kh = \frac{2\pi}{10} \approx 0.628$. For higher wave numbers, more grid points may even be necessary [1]. This rule will be mostly used for the analysis in Chapter 3 and 4. If one were to completely avoid the pollution error, the condition $k^3h^2 = 1$, i.e. $h = k^{-\frac{3}{2}}$ has to be employed. But this quickly causes computations to be very expensive, since the number of grid points then grows exponentially when k increases. Note that by taking $kh = 0.628$ we also do not fully have scalability, but we use it as an approximation for numerical testing. To gain a more scalable model, preconditioners are used, which will be discussed in the next section.

It is common practice to use Krylov iterative methods to solve (2.4). However, since A becomes indefinite for large values of k [6], only a limited number of Krylov methods are viable. It is known that the GMRES method can still work well if A is non-normal, yet diagonalizable [1, 6]. A is indeed diagonalizable since A is real-symmetric. For this reason, the GMRES method is also used for this study. In [12], it is also suggested that the MINRES solver works well for Hermitian indefinite matrices, but we do not consider this method in this thesis.

Due to the indefiniteness of A it is often difficult to analyze the convergence behavior of numerical solvers [2, 3, 6]. For Hermitian matrices, however, theoretical analysis is available to investigate close-to-zero eigenvalues, on which the convergence speed of GMRES depends. And for matrix A specifically a closed form for the eigenvalues and eigenvectors is known [1]. The closed-form expressions are given in the next chapter. Close-to-zero eigenvalues can therefore be more easily investigated, which makes (2.1) a nice model to work with as a basis for this research.

Coming back to the Sommerfeld radiation conditions, an explicit expression for the linear system of the 2D model can be found in [1]. Here, it can be seen that the coefficient matrix is complex symmetric, but not Hermitian. Besides, no closed-form expressions for the eigenvalues are known for the coefficient matrix. No useful theory is available for these kinds of systems, which is why we do not consider model (2.2) for the analysis.

2.2. Preconditioners

The next step is to introduce preconditioners. A preconditioner is a matrix M that is similar to matrix A from (2.4), but with nicer properties concerning the eigenvalues. These are used to increase the convergence speed of numerical solvers by clustering the eigenvalues more favorably [1]. The main goal here with using preconditioners is to obtain eigenvalues that lie further away from zero to increase convergence of the GMRES solver. Typically, one does not compute M, but rather M^{-1}. This means that $M^{-1}A \approx I$ and subsequently, the system

$$M^{-1}A\mathbf{u} = M^{-1}\mathbf{f} \tag{2.5}$$

is solved using GMRES. In this study, only the two-level Schwarz domain decomposition preconditioners based on a Bézier coarse space are investigated. However, we will also discuss some other preconditioners that are commonly used.

2.2.1. Domain Decomposition

With domain decomposition, one divides the domain Ω into N smaller subdomains Ω_i that contain subgrid G_i. The goal is to reduce the overall problem to smaller subproblems in order to reduce computation times. This is done by defining the Helmholtz problem for each subdomain using prolongation matrices R_i (also known as extension matrices) and restriction matrices R_i^T. Hence we have that

$$R_i^T = \begin{pmatrix} 0 & \cdots & 1 & 0 & 0 & \cdots & & \cdots & 0 \\ 0 & \cdots & 0 & 1 & 0 & \cdots & & \cdots & 0 \\ \vdots & & & \ddots & \ddots & \ddots & & & \\ 0 & \cdots & & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \tag{2.6}$$

is an $m \times n$ matrix with $a_{1,j} = a_{2,j+1} = \ldots = a_{m,j+m} = 1$ and all other coefficients zero. For each i the matrix $A_i := R_i^T A R_i$ is then the coefficient matrix corresponding to subdomain i, and we will refer to these matrices as subdomain matrices. Note that for our model, each subdomain matrix is simply the $m \times m$ version of A.

The domain decomposition preconditioner allows for parallel computing, since the subdomain problems can be solved independently [1]. Therefore, large scale Helmholtz problems can be solved more efficiently, especially when using more subdomains. This advantage makes domain decomposition methods popular in numerical research.

In a lot of research, overlap of the subdomains is used in order to obtain better convergence. This means that some nodes might be contained in multiple subdomains. The reason why overlap is often considered is to account for wave transmission through different subdomains. The overlap needs to be paired with suitable transmission conditions to achieve good convergence of the solver [2, 6, 10]. However, for the two-level additive Schwarz preconditioners, it has been shown that overlap has in general a detrimental effect for the convergence when using large wave numbers [1, 3]. In particular in [1], experiments have been conducted with variations of the two-level additive Schwarz preconditioner in a 2D domain using zero, one, two and four nodes in the overlapping regions. The overlap almost eventually always resulted in an increase of the number of iterations needed for convergence of the GMRES solver. For just one specific case an overlap of one node showed an only slightly better performance. For this reason we will always use zero overlap when performing analyses. Hence the dimension m of A_i is always chosen to be a divisor of n.

We will now look at the standard Additive Schwarz (AS1) preconditioner, which is defined as

$$M_{AS1}^{-1} = \sum_{i=1}^{m} R_i A_i^{-1} R_i^T \tag{2.7}$$

In other words, the AS1 preconditioner is the sum of all the subproblems. This preconditioner is an example of a one-level preconditioner, for which it is known that GMRES convergence depends on the number of subdomains [3, 7] and is therefore not scalable. In order to obtain further scalability, a coarse space is often added to also account for global information transmission between the subdomains. This results in the so-called two-level additive Schwarz (AS2) preconditioners. In this thesis, the coarse spaces are constructed with interpolation. Two other prominent types of coarse spaces are the (eneralised Eigenproblems in the Overlap) and the DtN (Dirichlet-to-Neumann) coarse spaces. These are described in detail in [2], where both coarse spaces show adequate results.

2.2.2. Bézier Coarse Spaces

This section discusses the construction of coarse spaces using second order Bézier interpolation from [1]. A coarse space essentially removes certain grid nodes by using larger grid steps. We will use steps of $2h$. Larger steps are also possible to create different coarse spaces. Bézier coarse spaces have shown to give good GMRES convergence [1, 6], which is why they will be used for this research.

Let G_{2h} be the set of grid nodes in the coarse space, let u_h denote a solution in a grid node of G and let let u_{2h} denote a solution in a grid node of G_{2h}. If we use second-order Bézier interpolation, the Bézier prolongation operator I_{2h}^h is defined in the following way

$$[u_h]_i = I_{2h}^h [u_{2h}]_i = \begin{cases} \frac{1}{8}\left([u_{2h}]_{(i-1)/2} + 6[u_{2h}]_{(i+1)/2} + [u_{2h}]_{(i+3)/2}\right), & i \text{ is odd} \\ \frac{1}{2}\left([u_{2h}]_{i/2} + [u_{2h}]_{(i+2)/2}\right), & i \text{ is even} \end{cases} \tag{2.8}$$

for $i = 1, \ldots, n$. The restriction operator I_h^{2h} is defined such that

$$[u_{2h}]_j := I_h^{2h}[u_h]_j = \frac{1}{8}\left([u_h]_{2j-3} + 4[u_h]_{2j-2} + 6[u_h]_{2j-1} + 4[u_h]_{2j} + [u_h]_{2j+1}\right) \tag{2.9}$$

for $j = 1,...,|G_{2h}|$. The next step is to define the matrices corresponding to I_{2h}^{h} and I_{h}^{2h}. By letting Z and Z^T be the matrices corresponding to I_{2h}^{h} and I_{h}^{2h} respectively, we obtain

$$Z^T = \frac{1}{8}\begin{pmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & & \vdots \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & & \vdots \\ \vdots & & & & \ddots & \ddots & \ddots & & \vdots \\ 0 & \cdots & & \cdots & 0 & 1 & 4 & 6 & 4 \end{pmatrix} \tag{2.10}$$

an $\frac{n}{2} \times n$ matrix, and Z is simply the transpose of Z^T. Setting $A_{2h} = Z^T A Z$, the term $M_{2h}^{-1} := Z A_{2h}^{-1} Z^T$, which can be seen as a "coarse space preconditioner", is added to (2.7) to obtain the two-level Additive Schwarz (AS2) preconditioner:

$$M_{AS2}^{-1} = M_{2h}^{-1} + M_{AS1}^{-1} = Z A_{2h}^{-1} Z^T + \sum_{i=1}^{m} R_i A_i^{-1} R_i^T \tag{2.11}$$

In Chapter 4 we will compare the Bézier coarse space with the coarse space constructed with standard linear interpolation. For linear interpolation, the prolongation operator L_{2h}^{h} is such that

$$L_{2h}^{h}[u_{2h}]_i = \begin{cases} [u_{2h}]_{(i+1)/2}, & i \text{ is odd} \\ \frac{1}{2}\left([u_{2h}]_{i/2} + [u_{2h}]_{(i+2)/2}\right), & i \text{ is even} \end{cases} \tag{2.12}$$

and the restriction operator I_{h}^{2h} is defined such that

$$I_{h}^{2h}[u_h]_j = \frac{1}{2}\left([u_h]_{2j-2} + 2[u_h]_{2j-1} + [u_h]_{2j}\right) \tag{2.13}$$

The Bézier interpolation yields a higher-order approximation scheme and is therefore expected to yield better convergence compared to linear interpolation. Before turning to the analysis, we will discuss a couple more preconditioners that are commonly used.

2.2.3. Example

We will work out an example in order to illustrate the methodology. Consider a line from 0 to 1 with 10 grid nodes as shown in Figure 2.1. Then $n = 8$ and $h = \frac{1}{9}$. With the finite difference method, we obtain a linear system of equations for nodes x_1 up to and including x_8 (the nodes $x_0 = 0$ and $x_9 = 1$ are not included because of the boundary conditions).

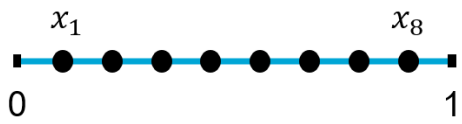


Figure 2.1: Example of a 1D domain with 10 grid nodes ($x_0 = 0$ and $x_9 = 1$).

The domain decomposition is chosen as in Figure 2.2, where the grid is divided into four subdomains, each containing 2 grid nodes. For each subdomain, the corresponding matrix is $A_i = \frac{1}{9}\begin{pmatrix} 2-\frac{k^2}{81} & -1 \\ -1 & 2-\frac{k^2}{81} \end{pmatrix}$. For the coarse space, the nodes with odd index are chosen. These are colored in green. The coarse space matrix A_{2h} is then computed as indicated in previous subsection.

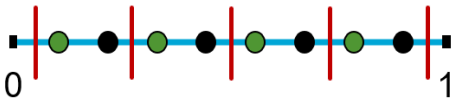


Figure 2.2: Example of a domain decomposition of a 1D domain. The green nodes represent the nodes chosen for the coarse space.

2.3. The Complex Shifted Laplacian Preconditioner

The complex shifted Laplacian preconditioner (CSLP) introduces a complex shift in the Helmholtz model. The preconditioner is defined as

$$M_{CSLP} = -\Delta_h - (\beta_1 + \beta_2 i)k^2 I_h \tag{2.14}$$

with Δ_h denoting the Laplacian operator and $i = \sqrt{-1}$. With the CSLP, eigenvalues are clustered in a circle around 1, which is generally favorable for GMRES convergence. Unfortunately, the CSLP also suffers from the effect of eigenvalues tending to zero for large k. Examples of eigenvalue clustering can be found in Figure 2.3 For not too large values of k, the CSLP does work rather well since the number of iterations scales only linearly. The preconditioner is therefore still regularly used. The CSLP needs to be inverted exactly in order to obtain the best results. In practice, this is too computationally costly and approximate inversions are computed, usually using multigrid iterations. But then the complex shift becomes large ($\mathcal{O}(k^2)$) which is also not favorable for GMRES convergence. For the most optimal results, $\beta_1 = 1$ and $\beta_2 = 0.5$ are chosen to simultaneously have few eigenvalues close to zero and to efficiently compute inversions. [1, 6, 11, 17]

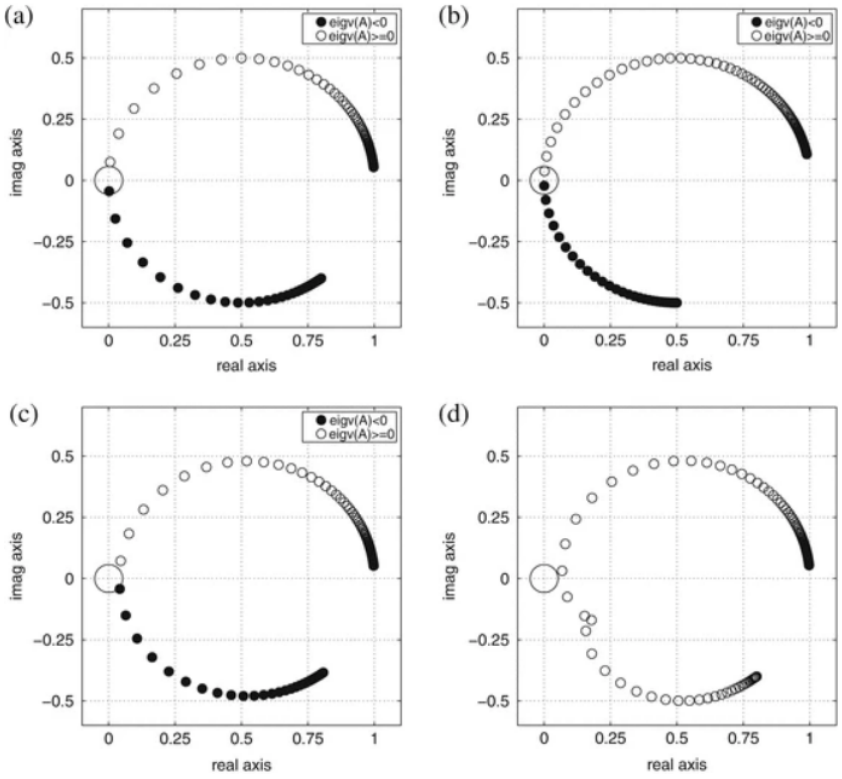


Figure 2.3: Eigenvalues of the CSLP preconditioner for different values of β_1 and β_2, $k = 1000$. [16]

2.4. The Deflation Preconditioner

In [6], the deflation preconditioner has been extensively studied. With deflation one aims to remove the smallest eigenvalues by deflating them to zero. If all the near zero eigenvalues could be eliminated this way, this would of course be an immediate remedy to the problem surrounding convergence. Slow convergence is

still an issue for large wave numbers, and it was concluded in [6] that this is mainly due to an approximation error for the transferring of grid functions. The deflation preconditioner is formulated as

$$P_D = I - P = I - AZ(Z^T AZ)^{-1} Z^T \tag{2.15}$$

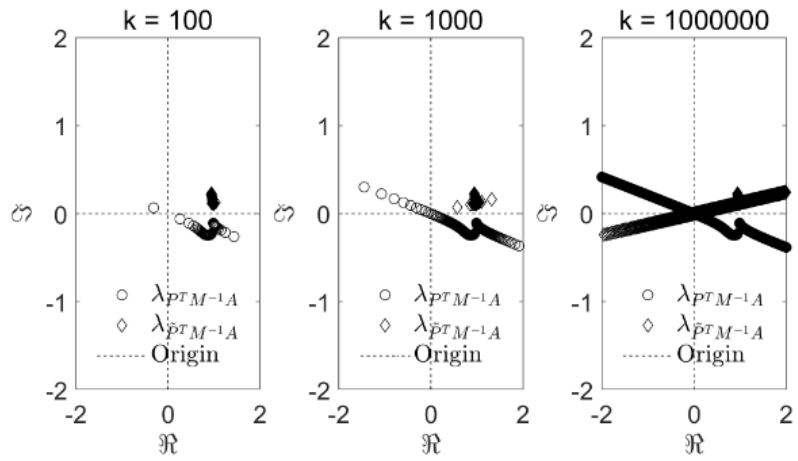
where Z and Z^T are the prolongation and restriction matrix respectively of some interpolation operators, like we saw with the Bézier coarse space. The system that is solved then reads as

$$P_D A\mathbf{u} = P_D\mathbf{f} \tag{2.16}$$

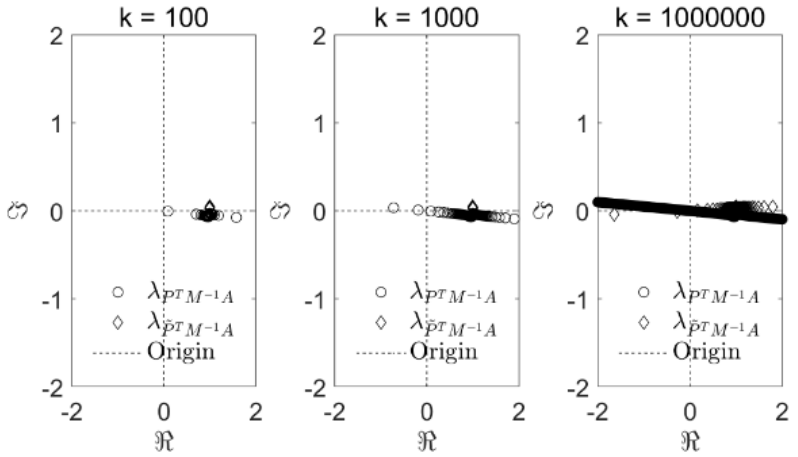
To remove the correct eigenvalues, the columns of Z need to be chosen appropriately. A natural choice would be to let each column represent an eigenvector of the eigenvalues that are removed. This, however, is computationally inefficient when dealing with large matrices. Eigenvectors are therefore approximated. Another option is to choose Z to be a linear or Bézier interpolation operator. In [6], these operators were used along with CSLP to build the DEF-based preconditioner (using linear interpolation) and the APD preconditioner (using Bézier interpolation). When combining CSLP and deflation, the preconditioned system reads as

$$M_{CSLP}^{-1} P_D A\mathbf{u} = M_{CSLP}^{-1} P_D\mathbf{f} \tag{2.17}$$

For the Bézier operator, this showed promising results, as can be seen in Figure 2.4. eigenvalues lie much more clustered around one and this is a significant improvement compared to Figure 2.3. The DEF-based preconditioner, however, does still yield a large number of close-to-zero eigenvalues for large k. In Chapter 4, we will also compare results between Bézier and linear interpolation, were the worse performance of linear interpolation will also be evident.[1, 6]



(a) $kh = 0.625$



(b) $kh = 0.3125$

Figure 2.4: Eigenvalues of the DEF-based preconditioner (circle marker) and the APD preconditioner (diamond marker), for different values of k and kh. [6]

3

Eigenvalue Analysis

3.1. Spectral Analysis

Returning to our initial problem (2.1), we mentioned that the eigenvalues and eigenvectors of the coefficient matrix A have been directly computed. From [1], the eigenvalues are given by

$$\lambda_l(A) = \frac{1}{h^2}(2 - 2\cos(l\pi h)) - k^2, \quad l = 1, ..., n \tag{3.1}$$

and the eigenvectors are

$$\phi_l = \sin(l\pi \mathbf{x}) \tag{3.2}$$

where $\mathbf{x}$ is the vector of grid nodes of G. The eigenvalues are increasing in l, so $\lambda_1 \leq \lambda_2 \leq \ldots \lambda_n$. From [6] we know that matrix A is indefinite whenever $k > \pi$. We now turn to the analysis of the AS2 preconditioner. As mentioned before, we want to investigate the close-to-zero eigenvalues. We therefore look at the "smallest absolute eigenvalue" (SAE). We say that $|\lambda_i|$ is the SAE of a matrix if $|\lambda_i| \leq |\lambda_j|$ for all j. In order to answer the first sub-question from the introduction, the SAEs of A, A_i and A_{2h} are investigated.

First, a result for the coarse space matrix A_{2h}. The following lemma is an adaptation of the analysis done in [6] (Lemma 4.1 and Corollary 4.2), where eigenvalue analysis was done for the two-level deflation preconditioner.

Lemma 3.1 *Assume $Z^T Z$ is an $\frac{n}{2} \times \frac{n}{2}$ matrix. If $l_{\min}$ is the index of the eigenvalue of A closest to zero and if ϕ_i^{2h} is the ith eigenvector of the coarse space matrix A_{2h}, then there exists a constant C_h, depending on h, such that*

$$A_{2h}\phi_{l_{\min}}^{2h} = C_h \lambda_{l_{\min}}(A)\phi_{l_{\min}}^{2h} \quad \text{and} \quad \lim_{h\to 0} C_h = \lambda_{l_{\min}}(Z^T Z) \tag{3.3}$$

□

From this result it is concluded that the SAE of A_{2h} is equal to the product of the SAE of A and $Z^T Z$ as $h \to 0$. Therefore we can compute an estimate for the eigenvalue of A_{2h} that is closest to zero. As stated before, we will use 10 points per wavelength ($kh = \frac{2\pi}{10}$).

Corollary 3.2 *Let $R(x)$ be the rounding function that rounds x to the nearest integer. Then the SAE of A_{2h} has a value of at least $\frac{3}{16h^2}\left|2 - 2\cos(\pi h R(\frac{0.203}{h})) - \left(\frac{2\pi}{10}\right)^2\right|$ when $kh = \frac{2\pi}{10}$.*

Proof. We first give a lower bound for $\lambda_{l_{\min}}(Z^T Z)$, with Z^T as in (2.10). We obtain

$$Z^T Z = \frac{1}{64}\begin{pmatrix} 53 & 28 & 1 & 0 & 0 & \cdots & 0 \\ 28 & 70 & 28 & 1 & 0 & & \vdots \\ 1 & 28 & 70 & 28 & 1 & & \vdots \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 & 28 & 70 & 28 \\ 0 & \cdots & \cdots & 0 & 1 & 28 & 69 \end{pmatrix}$$

is a symmetric $\frac{n}{2} \times \frac{n}{2}$ matrix. Using Gershgorin's circle theorem, one derives that the eigenvalues $\lambda_l(Z^T Z)$ lie in the interval $[\frac{3}{16}, 2]$. Hence $\mu_{l_{\min}} \geq \frac{3}{16}$.

Secondly, we compute $\lambda_{l_{\min}}(A)$. When using $kh = \frac{2\pi}{10}$ we obtain from (3.1):

$$\lambda_{l_{\min}}(A) = \frac{1}{h^2}\left(2 - 2\cos(l_{\min}\pi h) - k^2 h^2\right) \tag{3.4}$$

$$= \frac{1}{h^2}\left(2 - 2\cos(l_{\min}\pi h) - \left(\frac{2\pi}{10}\right)^2\right) \tag{3.5}$$

What remains is to compute $l_{\min}$. Using (3.1) we derive

$$\lambda_l(A) \approx 0 \iff \frac{1}{h^2}\left(2 - 2\cos(l\pi h) - \left(\frac{2\pi}{10}\right)^2\right) \approx 0 \tag{3.6}$$

$$\iff \cos(l\pi h) \approx 1 - 2\left(\frac{\pi}{10}\right)^2 \tag{3.7}$$

$$\iff l\pi h \approx 0.639 \tag{3.8}$$

$$\iff l \approx \frac{0.203}{h} \tag{3.9}$$

In other words, the SAE of A has index $R(\frac{0.203}{h})$ (this number becomes smaller when more grid points per wavelength are used). Applying Lemma 3.1 now yields the desired result. □

The following tables back up the results from Corollary 3.2. In Table 3.1 the validity of the index of the SAE is shown. In Table 3.2 the lower bound of the SAEs from Corollary 3.2 is compared to the actual SAEs.

n	$l_{\min}$	Estimated Index
250	51	51
500	102	102
1000	203	203
2000	407	407
4000	813	813

Table 3.1: (Estimated) indices of the SAE of A when using 10 points per wavelength.

n	SAE	Lower bound
250	257	11.4
500	1001	26.0
1000	1024	247
2000	4199	132
4000	2895	44.4

Table 3.2: SAEs of A_{2h} when using 10 points per wavelength, along with the lower bound as computed from Corollary 3.2.

Although Table 3.1 indicates the validity of Corollary 3.2, the lower bounds do not give a good approximation of the SAEs. The erratic behavior of the results does indicate that the SAE will never be too far from zero. This is not unexpected, since for greater n, more indices are available to minimize the SAE.

Next, we analyze the SAEs of the subdomain matrices. Lemma 3.1 may not be of help here since $R_i^T R_i$ might not be an $\frac{n}{2} \times \frac{n}{2}$ matrix. In fact, we already saw that A_i is just a smaller version of A. We can thus use (3.1) to obtain

$$\lambda_p(A_i) = \frac{1}{h^2}\left(2 - 2\cos\left(\frac{p\pi}{m+1}\right)\right) - k^2, \quad p = 1, 2, ..., m \tag{3.10}$$

If $p_{\min}$ is the index of the SAE of A_i, the it follows from the techniques used in Corollary 3.2 that

$$p_{\min} = 0.203(m+1), \tag{3.11}$$

$$\lambda_{p_{\min}}(A_i) \approx \frac{1}{h^2}\left(2 - 2\cos\left(\frac{\pi}{m+1} R[0.203(m+1)]\right) - \left(\frac{2\pi}{10}\right)^2\right) \tag{3.12}$$

If we compare to

$$\lambda_{l_{\min}}(A) = \frac{1}{h^2}\left(2 - 2\cos\left(\pi h R\left(\frac{0.203}{h}\right)\right) - \left(\frac{2\pi}{10}\right)^2\right) \tag{3.13}$$

$$= \frac{1}{h^2}\left(2 - 2\cos\left(\frac{\pi}{n+1} R\left[0.203(n+1)\right]\right) - \left(\frac{2\pi}{10}\right)^2\right) \tag{3.14}$$

we can see that there is only a slight change in the cosine term. One would expect the SAEs of A and the subdomain matrices A_i to be roughly the same for n and m large enough.

The analysis so far has not yet given us much information into the actual behavior of the eigenvalues with respect to k. We therefore turn to the preconditioner matrices in search for more clues.

3.2. Field of Values

An approach for analyzing the eigenvalues of the M_{AS2}^{-1} preconditioner is with the field of values (FOV). Usually, the FOV is only employed whenever eigenvalue analysis is difficult, such as when using Sommerfeld radiation conditions. Still, the FOV might yield some insights. The FOV of a matrix A is defined as

$$\mathscr{F}(A) = \left\{x^T A x \mid x \in \mathbb{C}^n \setminus \{0\},\ x^T x = 1\right\} \tag{3.15}$$

From [9] and (2.11) it follows that the eigenvalues of M_{AS2}^{-1} are contained in the sum of the field of values of M_{2h}^{-1} and M_{AS1}^{-1}, i.e. we have

$$\lambda_i(M_{AS2}^{-1}) \in \mathscr{F}(M_{2h}^{-1}) + \mathscr{F}(M_{AS1}^{-1}) \tag{3.16}$$

where the plus sign means that we take sums of an arbitrary element from $\mathscr{F}(M_{2h}^{-1})$ and from $\mathscr{F}(M_{AS1}^{-1})$. Since the eigenvalues of a matrix are contained in the FOV of that matrix, it might be useful to analyze the eigenvalues of M_{2h}^{-1} and M_{AS1}^{-1}. We know that M_{AS1}^{-1} is a block diagonal matrix with blocks A_i^{-1}, so the eigenvalues of M_{AS}^{-1} are exactly the eigenvalues of A_i^{-1}. As for M_{2h}^{-1}, numerical tests are needed to gain insight into the eigenvalues.

3.3. Principal Angles between Matrices

In order to investigate the relation between the subdomains and the coarse space, we look at the alignment of the coarse matrix A_{2h} with the subdomain matrices A_i. This is done by investigating the principal angles between these matrices. Angles between subspaces have mostly been used for convergence analysis of Krylov subspace methods. One can namely compute how similar the Krylov subspace is to the desired invariant subspace [13]. Angles of 90° indicate a gap between the subspaces, while angles of 0 indicate similarity, which is desired. In [6], it was shown that slow convergence was related to improper alignment between A and the

used coarse space. We suspect that a dissimilarity between the coarse space and the subdomains may also contribute to slower convergence. For the angles between two real orthonormal matrices A and B, we use the following formula from [14]:

$$\cos(\tilde{\theta}) = \tilde{\sigma} \tag{3.17}$$

Here, $\tilde{\theta}$ is a vector containing the angles between the column vectors of A and B, and $\tilde{\sigma}$ is the vector containing the singular values of $A^T B$. The smallest singular value determines the degree of alignment. A singular value close to zero corresponds to an angle close to 90° and thus bad alignment. It is required that both A and B have full column rank. This is generally not the case for A_{2h} and A_i. We resolve this problem by adding a sufficient amount of zero-rows to A_i to obtain matrix $\mathscr{A}_i$. Orthonormality is achieved by applying QR-orthonormalization. This, along with further analysis, is done numerically.

It is clear that numerical tests are needed to see the behavior of SAEs with respect to k and to find a correlation of the SAEs with GMRES convergence. Even though eigenvalue analysis is very applicable for our research model, it might not be the best approach for drawing any conclusions, for now. A more rigorous study is needed, which is not within the scope of this thesis.

4

Numerical Analysis

4.1. Close-to-Zero Eigenvalues

In order to gain more insight into the behavior of the SAEs, we run some tests with different values of k for different domain decompositions[1]. The following tables yield results on the SAEs of A, A_i and A_{2h} for Bézier interpolation and linear interpolation. Values of k and n are always chosen such that the 10 points per wavelength rule is roughly satisfied and such that numerical implementation is convenient. The results are later compared with the results from GMRES in section 4.2. In Table 4.1 we vary the dimension m of the subdomain matrices, while in Table 4.2, we vary the number of subdomains s. Note that $s = \frac{n}{m}$. We want to see whether patterns arise in the SAE behavior in order to answer sub-question 2 from the introduction.

k	n	$m = 2$	$m = 5$	$m = 10$	$m = 25$	$m = 50$	$m = 125$
157	250	38352 (1)	7768 (1)	4646 (2)	2345 (5)	1489 (10)	913 (26)
314	500	152405 (1)	31340 (1)	18904 (2)	9734 (5)	6326 (10)	3244 (26)
628	1000	607617 (1)	125899 (1)	76256 (2)	39643 (5)	26042 (10)	12162 (26)
1256	2000	2426465 (1)	504667 (1)	306294 (2)	159990 (5)	105641 (10)	47027 (26)
2512	4000	9697856 (1)	2020813 (1)	1227716 (2)	642795 (5)	425503 (10)	184862 (26)

Table 4.1: SAEs of A_i for different subdomain sizes when $kh \approx 0.628$. The numbers between parentheses denote the index of the eigenvalue.

k	n	$s = 2$	$s = 5$	$s = 10$	$s = 25$	$s = 50$	$s = 125$
157	250	913 (26)	1489 (10)	2345 (5)	4646 (2)	7768 (1)	38352 (1)
314	500	253 (51)	4550 (20)	6326 (10)	11368 (4)	18905 (2)	2722 (1)
628	1000	1745 (102)	3201 (41)	18157 (21)	29514 (8)	46166 (4)	74463 (2)
1256	2000	6531 (203)	18516 (81)	11217 (41)	84461 (16)	119514 (8)	378075 (3)
2512	4000	6749 (407)	12588 (163)	72312 (82)	101277 (33)	340827 (16)	539581 (7)

Table 4.2: SAEs of A_i, along with their index, for different numbers of subdomains when $kh \approx 0.628$.

k	n	A	A_{2h}
157	250	162 (51)	257 (51)
314	500	624 (102)	1001 (102)
628	1000	1240 (203)	1024 (203)
1256	2000	2477 (407)	4199 (406)
2512	4000	5189 (813)	2895 (813)

Table 4.3: SAEs of A and A_{2h}, along with their index, when $kh \approx 0.628$.

[1] All tests have been performed with Python. The used code is provided in Appendix A.

Table 4.1 shows a clear pattern within both the SAEs and their indices. When taking m constant, the SAEs increase by a factor of 4 when A becomes four times larger (i.e. when n is doubled), but decrease when A_i becomes becomes larger. The factor of 4 is in line with the expression from (3.12), since the term $\frac{1}{h^2}$ is multiplied by roughly 4 when n is doubled. The SAEs of A do not follow this pattern. This is not entirely surprising, since the cosine term in (3.14) also changes besides the $\frac{1}{h^2}$ term. For each value of m, the index of the SAE is the same regardless of the value of k and n. The index is always roughly one fifth of the corresponding value of m, which is in line with (3.11). The behavior of the indices as seen in Table 4.2 is also as expected. In each row, n is doubled, which means that m is also doubled since $m = \frac{n}{s}$. We see in (3.11) that $p_{\min}$ is then also roughly doubled. The behavior of the indices of A is explained analogously.

Note that there is a significant difference between the SAEs of A and the SAEs of A_i, even when the subdomains are large (for example for $k = 1256$ and $s = 2$). We conclude that the slight change that was seen between (3.12) and (3.14) can still contribute significantly to how close the cosine term is to $1 - 2\left(\frac{\pi}{10}\right)^2$. All SAEs seem to show a steady increase when k increases, with only a few exceptions. Therefore, we expect that GMRES would show slower convergence, since eigenvalues of the inverse matrices will be small.

The following tables show the effect of the subdomain size and the number of subdomains on the eigenvalues of $M_{AS2}^{-1}A$.

k	n	$m = 2$	$m = 5$	$m = 10$	$m = 25$	$m = 50$	$m = 125$
157	250	0.3577	0.6045	0.5923	0.5934	0.6086	0.6576
314	500	0.3532	0.6066	0.5973	0.5986	0.6148	0.6659
628	1000	0.3510	0.6082	0.6001	0.6017	0.6182	0.6683
1256	2000	0.3499	0.3573	0.3462	0.3452	0.3490	0.3676
2512	4000	0.3493	0.3564	0.3455	0.3445	0.3483	0.3667

Table 4.4: SAEs of $M_{AS2}^{-1}A$ for different subdomain sizes when $kh \approx 0.628$.

k	n	$s = 2$	$s = 5$	$s = 10$	$s = 25$	$s = 50$	$s = 125$
157	250	0.6576	0.6086	0.5934	0.5923	0.6045	0.3577
314	500	0.6370	0.6485	0.6148	0.5986	0.5973	0.0060
628	1000	1.000	0.4302	0.6518	0.6119	0.6015	0.0172
1256	2000	0.4358	0.3945	0.1920	0.3555	0.3473	0.3532
2512	4000	1.000	0.4908	0.3946	0.1313	0.3548	0.1223

Table 4.5: SAEs of $M_{AS2}^{-1}A$ for different numbers of subdomains when $kh \approx 0.628$.

In Table 4.4, the SAEs seem to increase in the first three rows, but then decrease whenever k is increased. Noteworthy is the sudden jump in the SAEs from $k = 628$ to $k = 1256$. From literature, it is expected that the SAEs would further decrease to zero when k is increased even further. However, the decrease of the SAEs seems steady for the most part, so we expect the iterations of GMRES to scale nicely along with n. Secondly, the SAEs show the same behavior regardless of the number of subdomains. They are always roughly the same, with only slightly better performance for $m = 5$ and $m = 125$. The only exception is for $m = 2$, where no jump is seen. When looking at Table 4.5, less consistencies are found. This erratic behavior is also somewhat found in Table 4.2, but there seems to be no correlation. Compare, for instance the results for $k = 1256$ and $s = 10$, and for $k = 314$ and $s = 125$. There we see for both tables a significant decrease of the SAE. However, for $k = 2512$ and $s = 5$, Table 4.2 shows an increase, whereas Table 4.5 shows a decrease. The notion that higher SAEs for the base coefficient matrices would result in lower convergence speed might therefore not be true. Finally, the column for $s = 2$ shows surprisingly high SAE, with a value of approximately 1 twice. But when using a lot of subdomains ($s = 125$), the values are much closer to zero. This may indicate that using a low amount of subdomains is favorable for convergence.

For a quick comparison, we also provide a table with SAEs when using a linear interpolation. We only look at different values of m to demonstrate. It can be clearly seen that SAEs are generally much lower, which is as expected.

k	n	$m=2$	$m=5$	$m=10$	$m=25$	$m=50$	$m=125$
157	250	0.0429	0.0164	0.0008	0.0361	0.0938	0.3464
314	500	0.0420	0.0146	0.0002	0.0372	0.0125	0.3722
628	1000	0.0224	0.0079	0.0001	0.0004	0.0131	0.1487
1256	2000	0.0107	0.0036	0.0004	0.0026	0.0136	0.1182
2512	4000	0.0057	0.0019	0.0006	0.0037	0.0032	0.0503

Table 4.6: SAEs of $M_{AS2}^{-1}A$ when using a linear interpolation coarse space and when $kh \approx 0.628$.

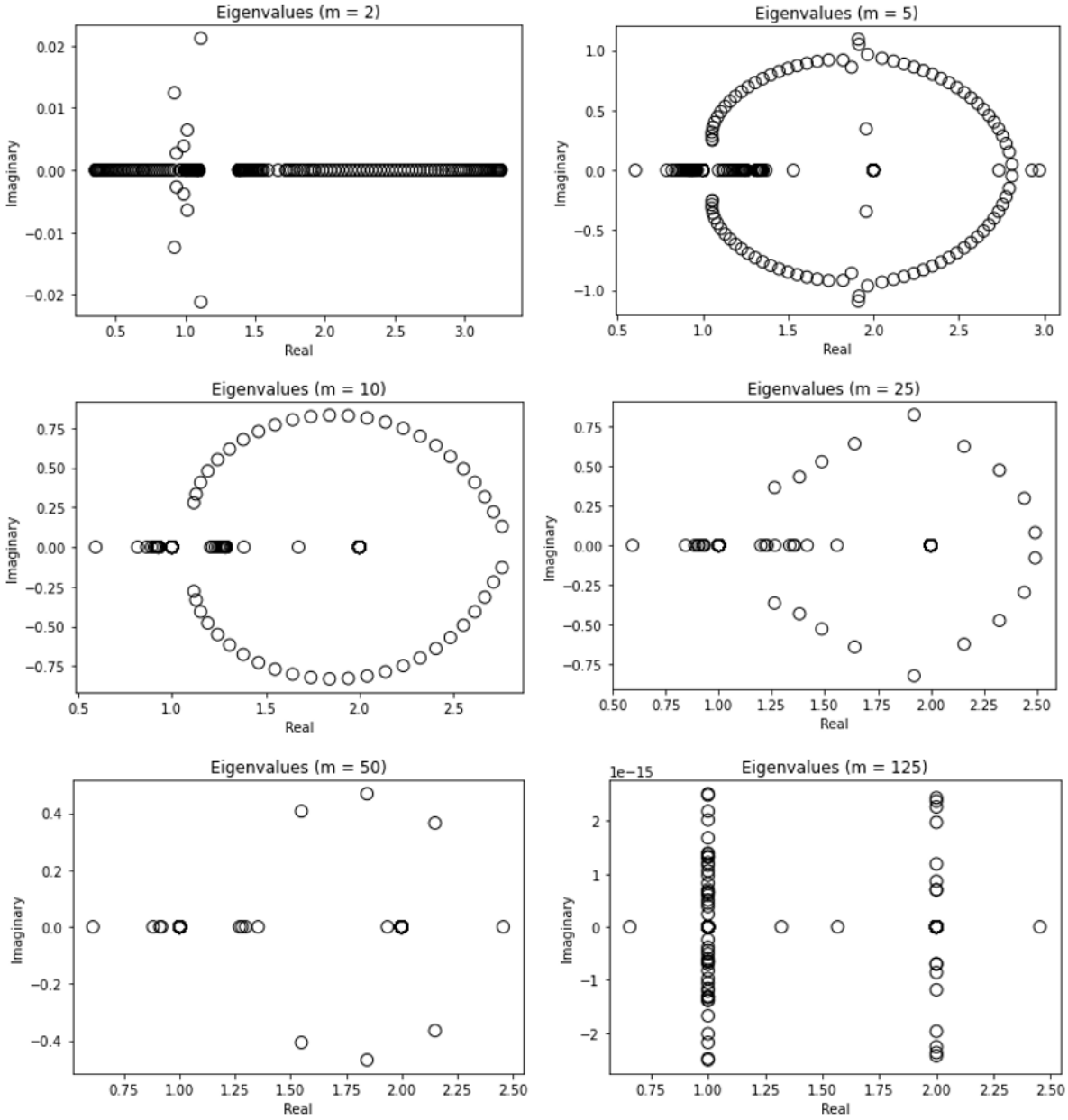


Figure 4.1: Eigenvalues of $M_{AS2}^{-1}A$ for $k = 157$, for different subdomain sizes.

In Figure 4.1, some plots are shown in which all the eigenvalues of $M_{AS2}^{-1}A$ are given for different values of m. These plots were generated for $k = 157$ to gain a further idea of the behavior of the eigenvalues. The plots indicate that the bigger the subdomain matrices, the less eigenvalues are clustered near zero. Especially with just two subdomains ($m = 125$), most eigenvalues are neatly clustered around 1 and 2, and it appears

that only one eigenvalue is close to zero. This further indicates that using a small number of subdomains is favorable.

We must reiterate that a larger number of subdomains is desired when working with domain decomposition methods. Figure 4.1 indicates that using the most amount of subdomains ($m = 2$) may not be beneficial for convergence. For slightly more subdomains, there only seems to be one eigenvalues that is very close to zero. Deflation could be used to remove this eigenvalue, which may significantly improve GMRES convergence.

4.2. GMRES convergence

We will now look into the actual number of iterations (NOI) needed for GMRES to converge with these two preconditioned systems. We allow the solver to perform at most 100 iterations. The results are compared to the previous tables.

k	n	$m = 2$	$m = 5$	$m = 10$	$m = 25$	$m = 50$	$m = 125$
157	250	27/98	31/100	26/100	22/100	14/18	6/5
314	500	28/100	30/100	29/100	25/100	20/100	13/14
628	1000	30/100	31/100	30/100	28/100	26/100	20/100
1256	2000	31/100	35/100	34/100	32/100	29/100	32/100
2512	4000	34/100	40/100	40/100	37/100	35/100	38/100

Table 4.7: NOI for different subdomain sizes when using Bézier interpolation (first number) and when using linear interpolation (second number). Here, $kh \approx 0.628$.

k	n	$s = 2$	$s = 5$	$s = 10$	$s = 25$	$s = 50$	$s = 125$
157	250	6/5	14/18	22/100	26/100	31/100	27/98
314	500	6/6	14/18	20/100	25/100	29/100	100/100
628	1000	6/6	17/18	24/100	25/100	28/100	100/100
1256	2000	6/6	16/18	79/100	29/100	30/100	26/100
2512	4000	6/6	17/18	34/100	56/100	36/100	56/100

Table 4.8: NOI for different numbers of subdomains when using Bézier interpolation (first number) and when using linear interpolation (second number). Here, $kh \approx 0.628$.

As expected, the NOI generally grows when k grows, and the Bézier interpolation vastly outperforms the linear interpolation in the general case. But overall, Table 4.4 shows a decent, almost linear scaling of the NOI along with n when k increases. Only when using a small number of subdomains, both Bézier and linear interpolation perform very well, which was already suspected from Table 4.5 and Figure 4.1. In fact, the NOI seem to be stable for $s = 2$. In other cases, the NOI almost immediately go to 100 when using linear interpolation, which means that the GMRES solver does not find a good solution in an appropriate time. The jump that we encountered in Table 4.4 is also somewhat visible in Table 4.7. Some oddities are also noticeable when comparing the NOI to Table 4.4. The NOI seem to increase faster for $m = 5$ and $m = 10$, whereas we saw in Table 4.4 that GMRES should perform slightly better due to higher SAEs. In Table 4.7, from $k = 1256$, an increase in the NOI is suddenly observed between the colums of $m = 50$ to $m = 125$. This is unexpected, since we saw higher SAE in the column of $m = 125$ in Table 4.4. However, there does seem to be an overall smoothing in the NOI, in the sense that the NOIs seem to depend less on m as k increases and when m is relatively small. This is a favorable result concerning parallel scalability. Table 4.5 does indicate a good correspondence of low SAE to high NOI. The outliers found in Table 4.10 for $s = 10$ and $s = 125$ all correspond to a very low SAE in Table 4.2. Discrepancies are found for $s = 2$, where the NOI does not seem to be influenced by the SAE, and for $s = 5$ and $k = 1256$, where both tables show a comparatively lower value.

When comparing to the SAEs from Table 4.1, 4.2 and 4.12, we see that the increase of SAEs corresponds to the increase of the NOI when varying m. But more inconsistencies are found with results for the number of subdomains. For example, when $k = 314$, the SAE is relatively small for $s = 2$ and $s = 125$. However, the NOI are vastly different. We do not find a clear pattern here.

To check the validity of the results, the following table shows the norm difference of the solution $\tilde{x}$ from GMRES and the exact solution x obtained from a direct numerical solver. As a demonstration, we only check

for three values of m. The results are in line with Table 4.7: Bézier interpolation yields solutions that are close from the real solution, whereas linear interpolation generally does not.

		$m=5$		$m=25$		$m=125$	
k	n	Bez	Lin	Bez	Lin	Bez	Lin
157	250	5.2e-06	15.5	2.9e-06	20.0	1.2e-12	6.7e-12
314	500	1.6e-05	40.2	1.9e-05	133	8.6e-06	1.9e-11
628	1000	1.3e-04	100	2.0e-04	318	3.1e-04	9.9e-02
1256	2000	5.2e-04	269	1.3e-04	1902	2.9e-04	1636
2512	4000	1.7e-03	1271	5.3e-04	4805	5.2e-04	8090

Table 4.9: Norm difference between the solution from GMRES and the exact solution from a direct solver when using the Bézier coarse space (Bez) and when using the linear interpolation coarse space (Lin). Here, $kh \approx 0.628$.

As a final comparison, we give some results for GMRES convergence when using approximately 20 points per wavelength ($kh = \frac{2\pi}{20}$). Since we are interested in a large number of subdomains, we only compare results for values of m. The NOIs are indeed significantly lower and even show more overall stability. The NOIs do differ more for the different values of m, which is less favorable for parallel scalability. Nonetheless, we can conclude that using many subdomains also yields good convergence, especially when $m = 2$.

k	n	$m=2$	$m=5$	$m=10$	$m=25$	$m=50$	$m=125$
157	500	15/33	18/35	27/71	18/38	19/100	11/14
314	1000	15/49	18/56	28/100	19/59	24/100	16/49
628	2000	15/100	18/100	28/100	19/100	24/100	18/100
1256	4000	15/100	18/100	28/100	19/100	25/100	19/100

Table 4.10: NOI for different subdomain sizes when using Bézier interpolation (first number) and when using linear interpolation (second number). Here, $kh \approx 0.314$.

Although a lower number of subdomains seems favorable for GMRES convergence, Table 4.4 is of main interest. As indicated earlier, a larger number of subdomains is desired for efficient computing. This is especially the case when working in a 2D or 3D setting.

4.3. FOV and Angles

The next table shows the range of eigenvalues of M_C^{-1}. It is clear that the eigenvalues lie very close to zero, especially when k is increased. When considering the FOV and (3.15), one might then ask how significantly the coarse space actually contributes to the convergence of GMRES in the theoretical sense. For now, this is merely a conjecture, since we do not know to what extent the eigenvalues are representative for the whole FOV.

k	n	Range
157	250	[-0.00135, 0.00514]
314	500	[-0.00089, 0.00132]
628	1000	[-0.00129, 0.00033]
1256	2000	[-0.00032, 0.00023]

Table 4.11: Smallest and largest eigenvalues of M_C^{-1} when $kh \approx 0.628$.

Our final results are on the angles between the coarse space and the subdomains. Since all subdomain matrices are the same, we only consider the first subdomain. Thus, the smallest singular value of $\mathscr{A}_i^T A_{2h}$ is investigated for different numbers/sizes of subdomains and coarse spaces. It was quickly found that the smallest singular value was consistently found to be 1, up to around 13 decimals. It follows from (3.17) that the largest principal angle is almost 0°, which indicates a very good alignment between the coarse space and the subdomains. This leads us to conclude that the matrix alignment does not have a negative impact on GMRES convergence.

k	n	A	A_{2h}
157	250	162	257
314	500	624	1001
628	1000	1240	1024
1256	2000	2477	4199
2512	4000	5189	2895

Table 4.12: SAEs of A and A_{2h}, along with their index, when $kh \approx 0.628$.

5

Conclusion & Discussion

The analysis from this research has confirmed a number of results from literature. Firstly, eigenvalues close to zero generally correspond to slower GMRES convergence, whereas eigenvalues away from zero yield faster convergence. Secondly, Bézier interpolation also yields much better convergence than linear interpolation. And finally, using a stricter rule of thumb for kh decreases the NOI significantly. Nonetheless, there are also some discoveries which cannot yet fully be explained with mathematical theory. Behavior of eigenvalues seems to follow a linear pattern for a large number of subdomains, but lose this pattern for a small number of subdomains. Therefore, it is not entirely clear to what extent eigenvalues of the subdomain matrices are an indicator for GMRES convergence. In Chapter 3, we provided some relations for the SAEs, but did not yield clear answers.

The question still remains what exactly happens within the subdomains, in a mathematical sense, that causes the near-zero eigenvalues. Where for the deflation method the root cause of problems could be pinpointed, this is still vague for domain decomposition. It might not be as related to the eigenvalues of the subdomains, nor the alignment between the coarse space and the subdomains as expected. However, we do suspect that matrix alignment could play a larger role when overlap is introduced. This could be topic for further research. Still, we do suspect that the SAEs of the subdomain matrices play an important role. The large SAEs of the subdomain matrices that were found

Given that a smaller amount of subdomains yields better results, another question that follows is: to what extent are domain decomposition methods actually helpful when solving Helmholtz problems? A more rigorous mathematical analysis on the behavior of eigenvalues within the subdomains and the correlation with GMRES convergence is again necessary. Even the use of a coarse space might be taken into question when looking at the FOV, although the difference in results for Bézier and linear interpolation does indicate that the coarse space contributes significantly to the improvement of GMRES convergence. Further research on SAE behavior, matrix alignment and the importance of the FOV is required. Moreover, it might be useful to also investigate the inclusion of deflation to determine the influence of the SAEs.

Appendix A: Python Code

```
import numpy as np
import scipy as sp
from numpy import linalg as LA
from scipy.linalg import block_diag
from scipy.sparse.linalg import gmres
import math
import matplotlib.pyplot as plt
```

```
n = 250
k = 157
k_ppw = (2*np.pi/10)*(n+1) #use to accurately account for 10 points per wavelength
m = 125            #use for varying subdomain size
subs = int(n/m)
#subs = 5          #use for varying number of subdomains
#m = int(n/subs)
h = 1/(n+1)
hm = 1/(m+1)
c = int(n/2)       #number of coarse grid nodes
print('k =', k, ', n =', n, ', subs =', subs, ', m =', m)
```

```
##### Exact Eigenvalues #####

EigvA = (1/(h**2))*(2-2*math.cos(np.pi*h*round(math.acos(1-0.5*((k*h)**2))/(np.pi*h))))-k**2
print('SAE of A:', EigvA)
EigvAi = (1/(h**2))*(2-2*math.cos(np.pi*hm*round(math.acos(1-0.5*((k_ppw*h)**2))/(np.pi*hm))))-k_ppw**
print('Eigenvalue of Ai:', EigvAi)
sae_A2h = (3/(16*(h**2)))*(2-2*math.cos(np.pi*h*round(math.acos(1-2*((np.pi/10)**2))/(np.pi*h)))-(2*np
print('Lower bound for sae_A2h =', abs(sae_A2h))
```

```
##### Building matrices #####

# construct rhs #
f = np.zeros(n)
nhalf = round(n/2);
f[nhalf] = 1/h;
```

```
# Constructing coefficient matrix #
Alist = [[0 for col in range(n)] for row in range(n)] #setting up nxn zero matrix
for i in range(n): #building A
    for j in range(n):
        if j == i:
            Alist[i][j] = (2/(h**2))-k**2
        elif j == i+1:
            Alist[i][j] = -1/(h**2)
        elif j == i-1:
            Alist[i][j] = -1/(h**2)
        else:
            Alist[i][j] = 0
A = np.array(Alist)


# Constructing subdomain matrix #
Ailist = [[0 for col in range(m)] for row in range(m)] #setting up mxm submatrix
for i in range(m): #building A_i
    for j in range(m):
        if j == i:
            Ailist[i][j] = (2/(h**2))-k**2
        elif j == i+1:
            Ailist[i][j] = -1/(h**2)
        elif j == i-1:
            Ailist[i][j] = -1/(h**2)
        else:
            Ailist[i][j] = 0
Ai = np.array(Ailist)
Ai_inv = LA.inv(Ai)


# Constructing bezier coarse space matrix #
ZTlist = [[0 for col in range(n)] for row in range(c)] #setting up (n/2)xn matrix
for i in range(c): #building ZT
    for j in range(n):
        if j == 2*i:
            ZTlist[i][j] = 6/8
        elif j == 2*i+1:
            ZTlist[i][j] = 4/8
        elif j == 2*i-1:
            ZTlist[i][j] = 4/8
        elif j == 2*i+2:
            ZTlist[i][j] = 1/8
        elif j == 2*i-2:
            ZTlist[i][j] = 1/8
        else:
            ZTlist[i][j] = 0
ZT = np.array(ZTlist)
Z = ZT.transpose()
A2h = ZT@A@Z
A2h_inv = LA.inv(A2h)


# Constructing linear coarse space
LTlist = [[0 for col in range(n)] for row in range(c)]
for i in range(c): #building Z
```

```
    for j in range(n):
        if j == 2*i:
            LTlist[i][j] = 1
        elif j == 2*i+1:
            LTlist[i][j] = 1/2
        elif j == 2*i-1:
            LTlist[i][j] = 1/2
        else:
            LTlist[i][j] = 0
LT = np.array(LTlist)
L = LT.transpose()
AL2h = LT@A@L
AL2h_inv = LA.inv(AL2h)


# Preconditioners #
M_AS1 = block_diag(*[Ai_inv for i in range(subs)])
M_bezcoarse = Z@A2h_inv@ZT
M_lincoarse = L@AL2h_inv@LT
M_AS2_bez = np.add(M_bezcoarse,M_AS1)
M_AS2_lin = np.add(M_lincoarse,M_AS1)
MinvA_bez = M_AS2_bez@A
MinvA_lin = M_AS2_lin@A
MAf_bez = MinvA_bez@f
MAf_lin = MinvA_lin@f




##### Principal Angles #####

# Angle between Ai and A2h #
Q_Ai, R_Ai = np.linalg.qr(Ai) #qr-orthonormalization
Q_A2h, R_A2h = np.linalg.qr(A2h)
print(Q_A2h)
temp = Q_Ai
zeros = np.zeros(m)
for i in range(c-m): #making number of columns of Ai equal to A2h
    temp = np.r_[Q_Ai, [zeros]] #by adding zero-columns
    Q_Ai = temp
S = Q_Ai.transpose()@Q_A2h
sigma = np.linalg.svd(S, compute_uv=False)
print(min(sigma)) #smallest singular value corresponds to largest angle




##### Computing SAEs numerically #####

# Matrix A #

EV = sorted(LA.eigvals(A))
evMin = min(EV) #lowest eigenvalue (optional)
```

```
evMax = max(EV) #largest eigenvalue (optional)
evAbsmin = abs(EV[0])
lMin = 0

for i in range(len(EV)):
    if abs(EV[i]) < abs(evAbsmin): #finding smallest absolute eigenvalue (SAE) of A
        evAbsmin = abs(EV[i])
        lMin = i #tracking index of SAE

#print('evMin =', evMin)
#print('evMax =', evMax)
print('evAbsmin =', evAbsmin, ', index = ', lMin+1)


# Ai #

EVi = sorted(LA.eigvals(Ai))
eviMin = min(EVi)
eviMax = max(EVi)
eviAbsmin = abs(EVi[0])
liMin = 0

for i in range(len(EVi)):
    if abs(EVi[i]) < abs(eviAbsmin):
        eviAbsmin = abs(EVi[i])
        liMin = i

#print('eviMin =', eviMin)
#print('eviMax =', eviMax)
print('eviAbsmin =', eviAbsmin, ', index = ', liMin+1)


# Coarse matrix #

EV2h = sorted(LA.eigvals(A2h))
ev2hMin = min(EV2h)
ev2hMax = max(EV2h)
ev2hAbsmin = abs(EV2h[0])
l2hMin = 0

for i in range(len(EV2h)):
    if abs(EV2h[i]) < abs(ev2hAbsmin):
        ev2hAbsmin = abs(EV2h[i])
        l2hMin = i

#print('ev2hMin =', ev2hMin)
#print('ev2hMax =', ev2hMax)
print('ev2hAbsmin =', ev2hAbsmin, ', index = ', l2hMin+1)


# Linear interpolation matrix #

EVL2h = sorted(LA.eigvals(AL2h))
evL2hMin = min(EVL2h)
evL2hMax = max(EVL2h)
evL2hAbsmin = abs(EVL2h[0])
```

```
for i in range(len(EVL2h)):
    if abs(EVL2h[i]) < abs(evL2hAbsmin):
        evL2hAbsmin = abs(EVL2h[i])

#print('evL2hMin =', evL2hMin)
#print('evL2hMax =', evL2hMax)
print('evL2hAbsmin =', evL2hAbsmin)


# AS2 with bezier interpolation #

EVMAb = sorted(LA.eigvals(MinvA_bez))
evmabMin = min(EVMAb)
evmabMax = max(EVMAb)
evmabAbsmin = abs(EVMAb[0])

for i in range(len(EVMAb)):
    if abs(EVMAb[i]) < abs(evmabAbsmin):
        evmabAbsmin = abs(EVMAb[i])

#print('evmabMin =', evmabMin)
#print('evmabMax =', evmabMax)
print('evmabAbsmin =', evmabAbsmin)


# AS2 with linear interpolation #

EVMAl = sorted(LA.eigvals(MinvA_lin))
evmalMin = min(EVMAl)
evmalMax = max(EVMAl)
evmalAbsmin = abs(EVMAl[0])

for i in range(len(EVMAl)):
    if abs(EVMAl[i]) < abs(evmalAbsmin):
        evmalAbsmin = abs(EVMAl[i])

#print('evmalMin =', evmalMin)
#print('evmalMax =', evmalMax)
print('evmalAbsmin =', evmalAbsmin)


# Bezier 'coarse preconditioner' #

EVm2h = sorted(LA.eigvals(M_bezcoarse))
evm2hMin = min(EVm2h)
evm2hMax = max(EVm2h)
evm2hAbsmin = abs(EVm2h[0])

for i in range(len(EVm2h)):
    if abs(EVm2h[i]) < abs(evm2hAbsmin):
        evm2hAbsmin = abs(EVm2h[i])

#print('evm2hMin =', evm2hMin)
#print('evm2hMax =', evm2hMax)
print('evm2hAbsmin =', evm2hAbsmin)
```

```
# Eigenvalue plot of preconditioned system #
x = [ele.real for ele in EVMAb]
y = [ele.imag for ele in EVMAb]
plt.scatter(x,y,s=80, facecolors='none', edgecolors='black')
plt.title('Eigenvalues (m = 125)')
plt.xlabel('Real')
plt.ylabel('Imaginary')
plt.show()




##### GMRES #####

### Gmres iterations ###

rtol = 1e-8
maxits = 100

# Counter to track number of iterations
bezIteration_counter = {"count": 0}
linIteration_counter = {"count": 0}

# Define a callback function that increments the counter
def bezIteration_callback(xk):
    bezIteration_counter["count"] += 1
def linIteration_callback(xk):
    linIteration_counter["count"] += 1

# Call gmres with the callback
bezgmres = sp.sparse.linalg.gmres(
    MinvA_bez, MAf_bez, tol=rtol, maxiter=maxits, callback=bezIteration_callback)
lingmres = sp.sparse.linalg.gmres(
    MinvA_lin, MAf_lin, tol=rtol, maxiter=maxits, callback=linIteration_callback)

# Output the number of iterations used
print(f"Number of GMRES iterations with bezier: {bezIteration_counter['count']}")
print(f"Number of GMRES iterations with linear interpolation: {linIteration_counter['count']}")


### Solution comparison ###

bezSolv = np.linalg.solve(MinvA_bez,MAf_bez)
bezdiff = bezSolv - bezgmres[0]
beznorm = np.linalg.norm(bezdiff)
print('Norm difference for bezier solution: ', beznorm)
linSolv = np.linalg.solve(MinvA_lin,MAf_lin)
lindiff = linSolv - lingmres[0]
linnorm = np.linalg.norm(lindiff)
print('Norm difference for linear interpolation solution: ', linnorm)
```

Bibliography

[1] E. Sieburgh. (2022). *Domain Decomposition Helmholtz Solvers*. Delft University of Technology.

[2] N. Bootland, V. Dolean, P. Jolivet, P. Tournier. (2021). *A comparison of coarse spaces for Helmholtz problems in the high frequency regime*. Computers & Mathematics with Applications, Vol. 98, pp. 239-253.

[3] O. Ernst, M. Gander. (2012). *Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods*. in Numerical Analysis of Multiscale Problems, Springer, Berlin, pp. 325–363.

[4] I. Babushka, S. Sauter. (1997). *Is the pollution effect of the FEM avoidable for the Helmholtz equation considering high wave numbers?* SIAM Journal on Numerical Analysis, Vol. 34, No. 6, pp. 2392–2423.

[5] C. Ma, C. Alber, R. Scheichl. (2023). *Wavenumber Explicit Convergence of a Multiscale Generalized Finite Element Method for Heterogeneous Helmholtz Problems*. SIAM Journal on Numerical Analysis, Vol. 61, No 3.

[6] V. Dwarka, C. Vuik. (2020). *Scalable convergence using two-level deflation preconditioning for the helmholtz equation*. SIAM Journal on Scientific Computing, Vol. 42, No. 2, pp. A901–A928.

[7] L. Conen. (2015). *Domain decomposition preconditioning for the Helmholtz equation : a coarse space based on local Dirichlet-to-Neumann maps*. Università della Svizzera italiana.

[8] D. P. Williamson. (2016). ORIE 6334 Spectral Graph Theory, Lecture 4. Available: `https://people.orie.cornell.edu/dpw/orie6334/Fall2016/lecture4.pdf`

[9] H. Wielandt. (1972). *On the Eigenvalues of A + B and AB*. Journal of Research of the National Bureau of Standards, Vol. 77B, Nos. 1 & 2.

[10] S. Gong, M. Gander, I. Graham, D. Lafontaine, E. Spence. (2022). *Convergence of parallel overlapping domain decomposition methods for the Helmholtz equation*. Numerische Mathematik, Vol. 152, pp. 259–306.

[11] J. Chen, V. Dwarka, C. Vuik. (2024). *Matrix-Free Parallel Preconditioned Iterative Solvers for the 2D Helmholtz Equation Discretized with Finite Differences*. Mathematics in Industry, Vol. 43, pp. 61-68

[12] J. Liesen, P. Tichy. (2005). *Convergence analysis of Krylov subspace methods*. Technical University of Berlin.

[13] C. Beattie, M. Embree, J. Rossi. (2004). *Convergence of Restarted Krylov Subspaces to Invariant Subspaces*. SIAM Journal on Matrix Analysis and Applications, Vol. 25, No. 4, pp. 1074–1109.

[14] P. Zhu, A. V. Knyazev. (2012). *Principal angles between subspaces and their tangents*. University of Colorado, Department of Mathematical and Statistical Sciences.

[15] K. Wang, Y. S. Wong. (2014). *Pollution-free finite difference schemes for non-homogeneous Helmholtz equation*. International Journal of Numerical Analysis and Modeling, Vol. 11, No. 4, pp. 87–815.

[16] D. Lahaye, C. Vuik. (2017). *How to Choose the Shift in the Shifted Laplace Preconditioner for the Helmholtz Equation Combined with Deflation*. In: Modern Solvers for Helmholtz Problems, pp. 85-112.

[17] M. B. van Gijzen, Y. A. Erlangga, C. Vuik. (2007). *Spectral Analysis of the Discrete Helmholtz Operator Preconditioned with a Shifted Laplacian*. SIAM Journal on Scientific Computing, Vol. 29, No. 5, pp. 1942–1958