Preconditioning for Mimetic Spectral Element Method

A Preliminary Study Applied on Elliptic Equations

Zeyu Liu

TU Delft

Preconditioning for Mimetic Spectral Element Method

A Preliminary Study Applied on Elliptic Equations

by

Zeyu Liu

to obtain the degree of Master of Science
at the Delft University of Technology,

Student number: 4520955

Thesis committee:	Associate Prof. dr. ir.	Marc I. Gerritsma,	TU Delft, supervisor
	Assistant Prof. dr.	Steven J. Hulsholf,	TU Delft
	Associate Prof. dr.	Martin B. van Gijzen,	TU Delft

TU Delft

This thesis is dedicated to my dearly beloved parents.

Contents

1 Introduction

1.1. Why Preconditioning?

Numerical methods are widely applied to solve problems modelled by partial differential equation (PDE). This numerical simulation process can be split to two steps: discretisation of PDE and solution of algebraic system of equations. The discretisation process replaces a continuous PDE with an algebraic linear system of equations acting on a set of discrete degrees of freedom (DoF). After discretisation, the linear system of equations needs to be solved by a linear equation solver. The solution will be used to reconstruct and approximate the PDE's solution. The two steps above do the most important job in a full cycle of a numerical simulation.

To give an example, let's look at Figure 1.1. Say we want to simulate an equilibrium state of a temperature field on a 2-dimensional domain, with the temperature distribution known on the boundary. This is a heat diffusion problem, modelled by a Poisson equation. To simulate this process, we first write out the mathematical expression of the governing law (usually conservation laws and constitutive relations) in the form of PDEs. From the continuous PDE, we arrange the unknowns in the domain of interest with a specific pattern on a mesh, and then replace the operators in the continuous PDEs by discrete operators that acts as a mapping similar to the operators in PDEs. In our example, the gradient operator ∇ corresponds to the matrix $\mathbf{B}$, the matrix $\mathbf{A}$ is a mass matrix, and the matrix $\mathbf{C}$ has a similar function as the divergence operator. Until now, the PDE is replaced by a discrete linear system of equations about a set of DoF arranged in the mesh. This is call discretisation. Now the rest of the work is done by the solver of linear equation systems. These two steps do the most important jobs in a numerical simulation process.

However, as the problem size grow, the numbers of DoF will be large. The linear equation system obtained from discretisation will not always be easy to solve, and this is affected by the type of solver we use, the condition of the left-hand-side matrix, the physical memory occupied by the matrix, etc.. Since problems of large size gives the major challenge, and iterative solvers are usually the only choice as the problem size gets large enough, the solvers that we work with will mostly be iterative ones. In order to make the iterative solution process fast and accurate, the large linear equation system will need to be preconditioned.

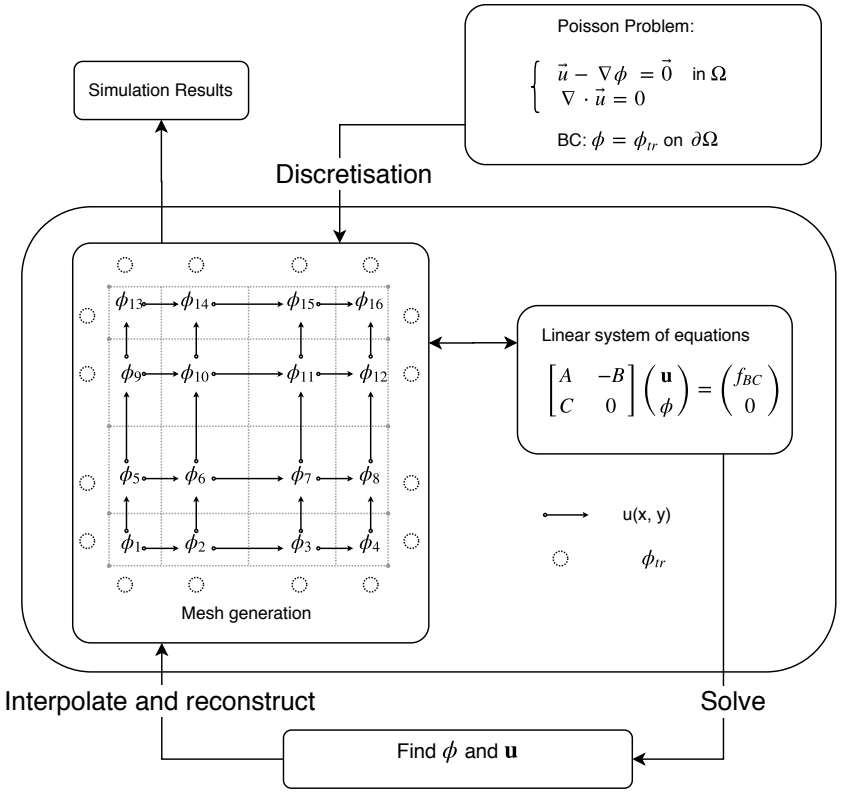

Figure 1.1: Example of a Numerical Simulation

1.2. Research Questions

This is the first time that preconditioning techniques are systematically studied for Mimetic Spectral Element Method (MSEM). Thus before asking research questions, a literature study was made to obtain a global picture about preconditioning and the discretisation method MSEM, which is partially reviewed in Chapter2. After the literature study, several research questions appeared. The main research interests are concentrated on the following 4 items:

1. What is the cause of the ill-conditioning of a PDE-related linear system of equations, or more specifically, MSEM-related systems.
2. How does the spectrum of the matrices change with respect to refinements and mesh distortion.
3. Find out efficient preconditioners and techniques to improve computation efficiency.
4. Is it possible to build a systematic framework for the future study on preconditioning techniques for MSEM.

These research questions will be answered as the report proceeds. But before we dive into the report, we give the layout of this thesis report.

1.3. Thesis Report Layout

In Chapter 2, a literature review will serve as both background knowledge and starting point for this thesis project. Chapter 3 will derive the discretisation formulations used in this thesis. In Chapter 4, a theoretical analysis on the condition numbers of the discrete operators is given. Chapter 5 will introduce preconditioning techniques suitable for *p*-version MSEM, and in Chapter 6, the techniques are extended to *hp*-version MSEM with hybrid formulation. Several results are presented for the hybrid *hp*-MSEM. In Chapter 7, we make a synthetic conclusion of this thesis, and recommendations will be given for the coming studies.

2
Literature Review

In this section, I kindly invite my readers to go through a brief literature review, concerning the Mimetic Spectral Element Method, and some preconditioning techniques used for solving FEM related saddle point problems. These background information will provide the context for the starting point of our research.

2.1. Mimetic Spectral Element Method

Mimetic Spectral Element Method is a discretisation method that combines the principle of mimetic discretisation and hp-FEM (Spectral Element Method). We will make a short introduction of mimetic discretisation and Spectral Element Method (SEM or *hp*-FEM). The purpose of this introduction is to guide my reader to the literatures that articulates these methods.

2.1.1. Mimetic Discretisation

After discretisation of a PDE, we see a "replica" of the PDE in the algebraic world. A desired discretisation schemes should provide a discrete model that is accurate, stable, and physically consistent. In [9], Bochev pointed out that for numerical schemes that clearly separate topology and metric, stability indicates if the topological structures in PDEs are well preserved, and consistency indicates if the approximations on the Physical quantities are proper, or in other words, the metric is properly used. In [10], Bochev and Hyman provided a common framework that abstracts discretisation methods that are compatible with the Physics behind PDEs, called mimetic discretisation.

In mimetic discretisation, the PDE operators acting on function spaces (of infinite dimension) are replaced by mimetic operators acting on discrete subspaces (of finite dimensions) of the infinite dimensional function spaces. The topological structure of the PDE is preserved by the discrete operators after discretisation. Thus the discretisation scheme mimics the topology of PDEs, with basic identities in vector calculus (like the Stokes theorem) satisfied.

The core tools used in the analysis of mimetic discretisation methods are differential geometry and algebraic topology. Differential geometry exposes the association between geometric objects and Physical quantities, thus further indicate the proper metric where the quantities are measured. The algebraic topology studies how discrete operators preserve the topological structure of PDEs on a given mesh. For an in-depth study of the theories about mimetic discretisation, I refer the reads to Kreeft's PhD paper [38]. In [38], an extensive discussion on mimetic spectral element method is made.

2.1.2. Spectral Element Method

Spectral Element Method (SEM), also known as the *hp*-FEM, belongs to the Finite Element Method (FEM) family. One important difference between SEM and FEM is that SEM use smooth basis functions to represent the function of interest. By using high order basis functions, the DoF number is increased, which is called a *p*-refinement. Spectral methods shows exponential convergence property while computing smooth quantities under *p*-refinements. If SEM was used just as an FEM, the normal increase of number of elements is called an *h*-refinement. For a domain with discontinuous quantities, *h*-refinement would be a better choice. For the basic information on SEM, refer to [21].

Traditional *hp*-FEM have great potentials in high precision simulation, however, the basis functions used in traditional *hp*-FEM do not fit into the framework of mimetic discretisation. Thus, followed by a similar methodology and principle in [10], Gerritsma, Palha, Jasper, and etc. introduced SEM to the mimetic family, building up the framework of Mimetic Spectral Element Method (see [43], [30]). In this paper, I will skip the review on their large amount of work but to mention the part important for our research. One of the most important development is on the basis function used in the new MSEM. The basis functions used in MSEM are of two types. One is the nodal basis function (see [21], Lagrangian basis polynomial), which is used in traditional *hp*-FEM, and another one is the edge basis function developed by Gerritsma in [29]. This new basis function made a bridge from traditional *hp*-FEM to mimetic discretisations. In the same paper, Gerritsma also showed the application of edge basis functions in spacial discretisation by combining them with Lagrange polynomial basis functions.

2.2. Preconditioning Techniques

From discretisation, we obtained a linear system of equations denoted by:

$$\mathbf{Ax} = \mathbf{b}, \tag{2.1}$$

where $\mathbf{A} \in \mathbf{R}^{n\times n}$ is a non-singular n-dimensional square matrix, and $\mathbf{x}, \mathbf{b} \in \mathbf{R}^n$. As mentioned before, if the linear equation system obtained from discretisation is easy to solve is determined by the condition of the left-hand-side matrix. We will see this in the coming subsection.

2.2.1. Condition Number and Preconditioning

The condition number of matrix $\mathbf{A}$ is defined by one of the following equivalent formulae (see [33]):

$$\kappa(\mathbf{A}) = \|\mathbf{A}\|\|\mathbf{A^{-1}}\|, \tag{2.2a}$$

$$\kappa(\mathbf{A}) = \lim_{\epsilon\to 0} \sup_{\|\mathbf{\Delta A}\|\leq\epsilon\|\mathbf{A}\|} \frac{\|(\mathbf{A}+\mathbf{\Delta A})^{-1} - \mathbf{A}^{-1}\|}{\epsilon\|\mathbf{A}^{-1}\|}, \tag{2.2b}$$

$$\kappa(\mathbf{A}) = \left(\min_{det(\mathbf{A}+\mathbf{\Delta A})=0} \frac{\|\mathbf{\Delta A}\|}{\|\mathbf{A}\|}\right)^{-1}. \tag{2.2c}$$

This definition for the matrix condition number above is not precise due to the freedom in choice of norms, but in my discussion, the vector induced 2-norm is used to measure the condition of the matrix. This is because it is invariant under orthogonal transformation and related to the vector 2-norm. Thus in this thesis, the condition number is the ratio of the largest absolute eigenvalue and the smallest absolute eigenvalue. From definition, the condition number of a matrix is always larger than 1.

After the definition, we point out that: the smaller the condition number is, the closer the eigenvalues are clustered, and the better the matrix is conditioned. A well conditioned matrix is preferred by the solver and I am going to demonstrate that.

There are two types of solvers for linear equation system, direct solver and iterative solver. The direct solvers use traditional techniques to eliminate entries in the system such as LU decomposition, Gauss elimination etc. (see [24]). Elimination process includes subtraction and division. When the smallest eigenvalue is small enough, the influence of the inexactness will be even amplified by the division, thus will produce a solution of very low precision (see [36]). A preconditioned system scales up small eigenvalues thus increase the precision. Direct solvers are advantageous when the system is not too large, they require a predictable amount of resource in terms of floating point operation times and storage (see [24]). However, as the size of the system grows large, the resource required by the direct solver scales up rapidly. Thus for large scale simulation, iterative solvers became a more popular choice. But for iterative solvers, the convergence speed is directly related to the eigenvalues of the matrix. For an ill-conditioned matrix, the eigenvalues are distinct from each other, then the convergence rate will differ greatly between each components. This will bring in two major issues. One is that the iterative

solver may not converge to the required residual level within an affordable iteration steps. Another issue is that even the solver minimised the residual to the machine precision, the smallest eigenvalue will scale up the uncertainty when error quantification is preformed.

From the discussion above, we can make the following statements:

1. Preconditioning may help direct solvers to obtain higher accuracy.
2. Preconditioning helps iterative solvers to reduce solution time and increase maximum accuracy.
3. Preconditioning study for iterative solvers deserve more effort, ill-conditioned problems are mostly taking place in large simulations.

2.2.2. Introduction of Preconditioners

In this subsection, a comprehensive demonstration about preconditioner will be given. And after that, we will list 3 types of configurations that a preconditioner could act on a linear equation system.

A comprehensive demonstration of a preconditioner could be described as follows. Suppose one has a matrix $\mathbf{M}$ easily available, invertible, and it nicely approximates the coefficient matrix $\mathbf{A}$ spectrally. Then one can premultiply $\mathbf{M}^{-1}$ to the system and make a new system looks like:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \tag{2.3}$$

The matrix $\mathbf{M}$ or $\mathbf{M}^{-1}$ is the preconditioner. After preconditioning, the new coefficient matrix $\mathbf{M}^{-1}\mathbf{A}$ will have a spectrum clustered near 1. Thus the new system is well conditioned.

The example given above is only one of the configurations that a preconditioner could act on a system. We give a wide class of preconditioner configurations in Fig. 2.1 (see [ke chen]).

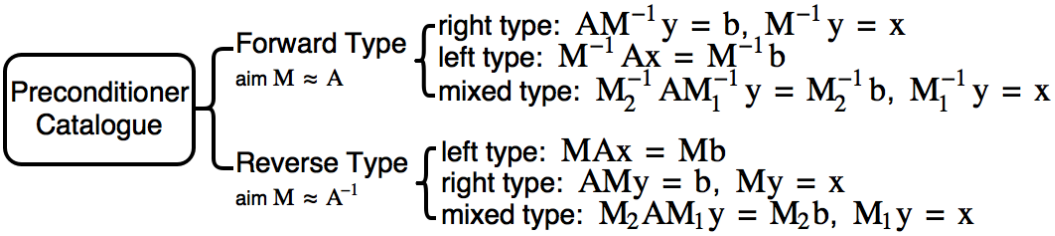

Figure 2.1: Preconditioner Catalogue

2.2.3. Approaches to Preconditioning

There are two ways to approach a preconditioning problem. One is to approach it as part of the source problem, i.e., to put preconditioning into the framework of the numerical simulation cycle. Another approach is to treat only the given matrix and find a well-conditioned system as an output. We call the first approach ”knowing the source”, and the second one ”knowing the matrix”.

The approach ”knowing the matrix” is good in terms of its generality once one has the knowledge about the matrices’ properties in terms of symmetry, definiteness and block structure etc.. This approach usually involves optimised program algorithms, advanced algebraic tools, and could be applied to problems arisen in optimisation, physics, statistics etc.. But it may not be the best choice of a specific type of problem. More specifically, PDE analysts would prefer seeing matrices derived from PDEs as a member of discrete operator families, and those discrete operators inherits many properties (e.g., symmetric/self-adjoint) from their continuous counterpart operators in PDE. This idea is illustrated in the review paper by Mardal and Wither (see [41]). This valuable point of view is also revealed in [31] by Günnel, pointing out that preconditioning is a correction for the unnatural choice of mappings on the underlying spaces. This is also the point made by Wathen in [46] : ”Whether or not a matrix system arises from PDEs, knowledge of the source problem can be helpful in designing a good preconditioner.” This awareness is indeed widely acknowledged by applied mathematicians. Benzi also

differentiates preconditioners into physical ones and algebraic ones. Last point to be made here is that, we differentiate preconditioners in this way not only for classification purpose, but also to optimally combine these two approaches for better preconditioning techniques. A good example could be that the multigrid method is originally a physical approach, but being generalised to algebraic multigrid that applies to a wider problem classes.

In the framework of my thesis study, I treat preconditioning as part of PDE solution process, i.e., "knowing the source". But the general techniques that are used while "knowing the matrix" will be combined into the big picture. Thus before looking at the preconditioning techniques for FEM-related problems, we give a review of some general techniques that are used to build preconditioners.

2.2.4. General Techniques of Preconditioning

General techniques are powerful tackling a certain type of matrix with specific properties, thus they will be helpful to precondition specific block in a system. But a single techniques is usually not enough to tune a large system in PDE-related linear systems. This subsection will introduce some popular preconditioning techniques.

Incomplete Factorisation This is a technique designed for large linear systems. Direct solvers live on matrix factorisation, where fill-in process is usually unavoidable. This process often destroys the sparsity of the original system. Before the invention of this method, sparsity-preserving pivoting algorithms are introduced (see[5]), however, considered too complex and expensive for large systems. As a concession, incomplete factorisation in equation (2.4) is applied, where $\mathbf{M_1}$ and $\mathbf{M_2}$ will inherit the sparsity while minimising the drop-out error. It is conflictive in general to achieve both at the most excellence, so a concept called "level of fill-in" is introduced by Gustafsson and Watts to make a balance in between (see [32] [48]). The "level of fill-in" is determined by a "drop tolerance", whose optimal choice is highly problem dependent. This leads to a proposal by Saad later on, a dual threshold strategy that dynamically drop out fill-ins by looking at drop tolerance (see [44]). To quote from Benzi after his test in [4] : "The resulting preconditioner is quite powerful. If it fails on a problem for a given choice of parameters (drop tolerance and numbers of fill-in), it will often succeed by taking a smaller value of drop tolerance or a larger number of fill-ins."

$$\mathbf{A} = \mathbf{M_1}\mathbf{M_2} + \mathbf{E} \tag{2.4}$$

This method has some difficulties in providing robustness and reliability for general and especially indefinite problems, but is widely used as a block preconditioner for SPD blocks, e.g., the mass matrix. Benzi made an in-depth review about many relevant issues concerning its solution existence, solution stability, ordering and parallel potential in [6] for further reference.

Sparse Approximate Inverse Since incomplete factorisation might destroy sparsity and involves advanced algorithms to achieve optimal performance, numerical analysts turned to an alternative, sparse approximate inverse. A comprehensive review on this method is given by Saad in [23]. This technique is devoted to finding a sparse matrix $\mathbf{M}$, an approximation of the inverse of coefficient matrix $\mathbf{A}$, as a preconditioner. This method preserves sparsity but is not preferred if a symmetric iterative solver such as Conjugate Gradient or Minimum Residual is used.

There are two main approaches of this method, one is to minimise the Frobenius norm of the error vector $\mathbf{I} - \mathbf{AM}$ by solving a constrained minimisation problem given the sparse pattern of $\mathbf{M}$. The challenge of this method is in choosing a good sparse pattern for $\mathbf{M}$. Algorithms of finding such patterns are reviewed in Benzi's papers [4] and [6]. Another type of approach is called factored approximate inverse. This approach is similar to the incomplete factorisation, but instead of factorise $\mathbf{A}$, the inverse of $\mathbf{A}$ is factorised. Using the generalised Gram-Schmidt process, called bi-conjugation, a triangular factorisation of $\mathbf{A^{-1}}$ is calculated. This is referred as the AINV algorithm (see [7] [8]).

Multi-Grid/Multi-Level This method is very well explained in Brandt's paper [20] in 1977. He pointed out the drawback of the traditional numerical process of solving partial differential equations. Due to the lack of interaction between discretisation and solution, the solution is not properly resolved and a huge amount of computational resource is wasted while obtaining a solution of low accuracy. Also, traditional processes failed to take the advantage of proximity of different systems given by different meshes. Given those drawbacks, Multi-Level method proposed an intermix of discretisation and solution processes, such that the drawbacks are overcome.

This technique has two main concepts: multi-grid and adaptive discretisation. In multigrid method, for any given mesh, the solution process will constantly interact with a hierarchy of successively coarser grids. This process has the advantage of the consistency property of discretisation: different discretised system are approximating solution of the same continuous problem. Multigrid concepts let coarse and fine mesh solutions correct each other, accelerates the convergence of fine mesh calculation, while increase the accuracy of approximations on the coarse mesh. This point of view gives the idea of constructing preconditioners using multigrid technique. An easily solvable system resulting from a coarse grid could be a good preconditioner for a complicated system resulted from a very fine grid, and a low order method may provide the preconditioner for the high order approximation. The adaptive discretisation concept is very powerful in determining the proper local order of approximation, and give an optimal allocation of computational resources to the global calculation domain. This concept gives indication on how local refinement could be done, thus making multilevel methods closely related and usually combined with domain decomposition methods.

Domain Decomposition Domain decomposition (DD), also known as "substructuring", "Schwartz method", was originally preposed to solve PDEs by decomposing the calculation domain into smaller subdomains, then different treatments could be employed separately, reducing the complexity of the overall problem. But as a preconditioning technique, it helps reformulating the problem thus the discrete system is well conditioned and suitable to be solved in parallel. Its mathematical foundation was established by constructing a transmission condition at the selected interface within the global calculation domain. This transmission condition got further extended to Steklov-Poincaré interface equation, from which Steklov-Poincaré operator is abstracted. And the discretised counterpart of Steklov-Poincaré operator is exactly the Schur complement of the $(1,1)$ block (refer to the block $\mathbf{M}$ in $\mathbf{A}$ from (2.5) in the final saddle point system. It is useful to look into domain decomposition technique while constructing preconditioners for the Schur complement block.

Domain decomposition and its variation is such a huge topic that cannot be covered in depth here. However, a few applications in solving saddle point problems should be mentioned. For example, the mixed formulation of finite element approximation using a Lagrange multiplier is an important variant of DD. In 1990s, Bramble, Pasciak and Schatz published a series of four papers (see [13] [15] [17] [18] [12] [14]), demonstrated a preconditioning method using domain decomposition technique. The first two papers gave two approaches in constructing a preconditioner for the $(1,1)$ block of the saddle point system along with an analytic prediction of the condition number after preconditioning. The later two papers discussed the iteration method to cope with the preconditioners as well as the parallel implementation of the proposed technique. They also proposed a local refinement scheme based on substructuring during that period (see [16]).

2.3. Precondition FEM-related Problems

In this section, we will review some important preconditioners for FEM related problems. First of all, these problems leads to a linear system of equations with the following block structure in equation (2.5):

$$\mathbf{A}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix}\mathbf{M} & \mathbf{B^T}\\ \mathbf{B} & -\mathbf{C}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{y}\end{bmatrix} = \begin{bmatrix}\mathbf{f}\\ \mathbf{g}\end{bmatrix}, \tag{2.5}$$

where $\mathbf{M}$ is a symmetric real square matrix in $\mathbf{R^{n\times n}}$, while $\mathbf{B}$ is an $m \times n$ matrix. If the $-\mathbf{C}$ block is all zero and the global system is symmetric, the problem is call classical saddle point problem. Otherwise,

it is a generalised/regularised saddle point problem. A naturally stable FEM method only leads to or could be transformed to a classical saddle point problem, which is the one that is involved in most of our discussion.

2.3.1. Preconditioner Utilising Block Structure

The block structure in (2.5) appears a lot in finite element discretisation, thus this structure has been widely discussed before. In fact, preconditioners thrived on this block structure, which is a good demonstration of the "knowing the source" type of philosophy.

In (2.5), matrix $\mathbf{M}$ is usually symmetric positive definite (SPD), and we can block diagonalise the large system as:

$$\begin{bmatrix} \mathbf{M} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & -\mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{B}\mathbf{M}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{O} \\ \mathbf{O} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{M}^{-1}\mathbf{B}^{\mathbf{T}} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}, \tag{2.6}$$

where, $\mathbf{S}$ is the Schur complement of $\mathbf{M}$:

$$\mathbf{S} = \mathbf{C} + \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathbf{T}}. \tag{2.7}$$

Block diagonalisation turns the coupled system into a two decoupled separated system in (2.8):

$$\mathbf{M}\mathbf{x} = \mathbf{f} - \mathbf{B}^{\mathbf{T}}\mathbf{y}, \tag{2.8a}$$

$$\mathbf{S}\mathbf{y} = -\mathbf{g} - \mathbf{B}\mathbf{M}^{-1}\mathbf{f}. \tag{2.8b}$$

If matrix $\mathbf{M}$ is SPD, then its Schur complement is also SPD. We can see from above that it's crucial to find a good preconditioner for the mass matrix. The precondition has to support fast inversion, and better be sparse. Using the information above, we introduce the following preconditioners and techniques appeared in history.

2.3.2. Uzawa's Algorithm

Uzawa's algorithm was originally developed from problems in nonlinear programming (see [19]). Inspired by the decoupled system in (2.8), it solves (2.8b) at the first stage, then solves (2.8a) at the second stage. This method is very robust and accurate with preconditioned iterative solvers and its convergence rate is independent of the number of unknowns (see [11]). This algorithm is very attractive when the memory budget is limited in computers. The drawbacks of Uzawa's algorithm are mainly in two aspects. Firstly, the inverse of the mass matrix $\mathbf{M}$ has to be found out before the first stage calculation can start, which is usually expensive. Secondly, in the two stage iteration, in order to make sure the second stage iteration converge, the first stage iteration needs to converge to machine round off, which makes the whole process very costly.

2.3.3. Bramble-Pasciak CG Reformulation

After Uzawa's algorithm, Bramble and Pasciak proposed a positive definite reformulation of the saddle point problem similar to the process given above, and developed a single-level iteration approach for the solution of (2.5) in paper [11]. In this paper, they made use of the knowledge of preconditioners designed for second-order elliptic equations, reformulated a positive definite system looks like:

$$\mathbf{A}\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_\mathbf{0}^{-1}\mathbf{M} & \mathbf{M}_\mathbf{0}^{-1}\mathbf{B}^{\mathbf{T}} \\ \mathbf{B}\mathbf{M}_\mathbf{0}^{-1}(\mathbf{M} - \mathbf{M}_\mathbf{0}) & \mathbf{C} + \mathbf{B}\mathbf{M}_\mathbf{0}^{-1}\mathbf{B}^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_\mathbf{0}^{-1}\mathbf{f} \\ \mathbf{B}\mathbf{M}_\mathbf{0}^{-1}\mathbf{f} - \mathbf{g} \end{bmatrix}, \tag{2.9}$$

and the coefficient matrix $\mathbf{A}$ could spectrally be estimated by matrix $\tilde{\mathbf{A}}$

$$\tilde{\mathbf{A}} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{O} & \mathbf{C} + \mathbf{B}\mathbf{M}^{-1}\mathbf{B}^{\mathbf{T}} \end{bmatrix}. \tag{2.10}$$

The reformulated system is uniformly well conditioned in Stokes u-p system if the mass matrix preconditioner $\mathbf{M}_\mathbf{0}$ is good enough and the $inf - sup$ condition is globally met. This method is advantageous due to the rich availability of preconditioners designed for CG acceleration (see [28]), but this method damaged the symmetry of the original system and required proper scaling for general application. For mixed formulation, the condition of the final system is dependent on mesh size.

2.3.4. Block Preconditioner of Split Type

To overcome the asymmetry of the system in (2.9), Rusten and Winther proposed a block diagonal preconditioner of the split type shown in (2.11) for classic saddle point problem.

$$\mathbf{P} = \begin{bmatrix} \mathbf{L} & \mathbf{O} \\ \mathbf{O} & \mathbf{U} \end{bmatrix} \tag{2.11}$$

In the block diagonal matrix $\mathbf{P}$, the $(1,1)$ block $\mathbf{L}$ is derived from Cholesky factorisation of mass matrix's approximation, i.e.,

$$\mathbf{M} \approx \mathbf{L}\mathbf{L}^{\mathbf{T}}. \tag{2.12}$$

Then the $(2,2)$ block $\mathbf{U}$ is naturally determined by

$$\mathbf{S} = \mathbf{B}\mathbf{M}^{-\mathbf{1}}\mathbf{B}^{\mathbf{T}} \approx \mathbf{B}(\mathbf{L}\mathbf{L}^{\mathbf{T}})^{-\mathbf{1}}\mathbf{B}^{\mathbf{T}} = (\mathbf{B}\mathbf{L}^{-\mathbf{T}})(\mathbf{B}\mathbf{L}^{-\mathbf{T}})^{\mathbf{T}} = \mathbf{U}\mathbf{U}^{\mathbf{T}}. \tag{2.13}$$

Winther's paper also provided algorithm to obtain cheap sparse factor matrix $\mathbf{U}$. In their work, they pointed out that the convergence rate of MINRES solver is bounded by three important parameters: the condition number of the mass matrix $\kappa(\mathbf{M})$, the condition number of the self-adjoint operator $\kappa(\mathbf{B})$ and the scaling of those two matrices $\rho(\mathbf{M},\ \mathbf{B})$. This leads to the point that as long as the three parameters are bounded by the matrices' own property independent from the mesh size, the convergence rate will be bounded independent of the mesh size as well. However, most of the discretisation techniques results in a system such that "at least one of the condition number will increase when the discretisation is refined". Thus a good preconditioner may be dependent on the mesh size and the way mesh is refined. One further remark Winther mentioned is that MINRES converges independently from the sparsity structure of $\mathbf{M}$ and $\mathbf{B}$.

2.3.5. Constraint Preconditioner

The previous work discussed all combined Schur decomposition with preconditioner of $(1,1)$ block $\mathbf{M}$. In 2000, Keller, Gould and Wathen proposed a new type of preconditioner for saddle point problems (see [37]), named constraint preconditioner, inspired by Lukšan and Vlček (see [40]) in the context of constrained non-linear programming. Constraint preconditioner still use approximation (or approximate inverse) of $\mathbf{M}$ on its $(1,1)$ block, then include the exact representation of the self-adjoint operator blocks in the preconditioner. The preconditioner take the form

$$\mathcal{A}_{\mathbf{0}} = \begin{bmatrix} \mathbf{G} & \mathbf{B}^{\mathbf{T}} \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \tag{2.14}$$

This is an indefinite preconditioner which made the application of CG and GMRES on preconditioned system possible again. They have shown that the inclusion of $\mathbf{B}$ in the preconditioner is able to cluster at least $2m$ eigenvalues to 1. This method is easier to implement compare to the work discussed before, and has lower computational complexity and iteration count. However, this method has a pitfall-kind drawback: the preconditioned system has a deficient eigenvector space and the standard Krylov subspace convergence theory is not applicable. They proposed two solvers to incorporate with the deficiency, one is Reduced Conjugate Gradient (RCG) and the other is Preconditioned GMRES (PGMRES). Later in 2003, Axelsson and Neytcheva (see [3]) proposed a way to avoid such deficient eigenvector space for constraint preconditioners. Further efforts on constraint preconditioner are investigated in finding better $\mathbf{G}$ using factorisation (see [25]) and application on regularised (augmented) saddle point problems (see [26] [27]).

3
Poisson Problem

This chapter will discretise an elliptic equation (Poisson problem) using MSEM.

Consider a two dimensional bounded domain $\boldsymbol{\Omega} \in \mathbf{R}^2$, with boundary $\partial\boldsymbol{\Omega}$. We solve a the Poisson problem on this domain with either Dirichlet or Neumann boundary conditions (BC). The Poisson problem with Dirichlet BC is given in (3.1):

$$-\Delta\phi(x,y) = f, \qquad \text{in } \boldsymbol{\Omega}, \tag{3.1a}$$
$$\phi(x,y) = \phi_{tr}, \qquad \text{in } \boldsymbol{\Gamma_D}. \tag{3.1b}$$

We rewrite this equation into a mixed formulation with subequations (3.2a) and (3.2b):

$$\mathbf{q} - \text{grad}(\phi) = \mathbf{0}, \qquad in\ \boldsymbol{\Omega}, \tag{3.2a}$$
$$-\text{div}(\mathbf{q}) = f, \qquad \text{in } \boldsymbol{\Omega}, \tag{3.2b}$$
$$\phi = \phi_{tr}, \qquad \text{on } \boldsymbol{\partial\Omega}. \tag{3.2c}$$

Let $\mathcal{L}^2(\boldsymbol{\Omega})$ denote the function space for square-integrable scalar functions, $\mathcal{H}^1(\boldsymbol{\Omega})$ denote the function space for $\mathcal{L}^2$ functions, whose partial derivatives also lay in $\mathcal{L}^2(\boldsymbol{\Omega})$. The restriction of functions in $\mathcal{H}^1(\boldsymbol{\Omega})$ to the boundary $\partial\boldsymbol{\Omega}$ constitutes the functions space $\mathcal{H}^{1/2}(\partial\boldsymbol{\Omega})$.

3.1. Weak Formulation

We derive a variational formulation of the second order using integration by parts:

$$(\mathbf{q},\ \hat{\mathbf{q}})_{\mathbf{L}^2,\Omega} = (\text{grad}(\phi), \hat{\mathbf{q}})_{\mathcal{L}^2,\Omega} \tag{3.3a}$$
$$= -(\phi, \text{div}(\hat{\mathbf{q}})) + (\phi, \hat{\mathbf{q}})_{tr}, \tag{3.3b}$$

where $\hat{\mathbf{q}}$ is the chosen test function. We use linear combination of basis functions as a representation of the continuous function restricted to finite dimensional function space, therefore the inner product $(\mathbf{q},\ \hat{\mathbf{q}})$ could be written by a product of mass matrix and the expansion coefficient vector of $\mathbf{q}$:

$$(\mathbf{q},\ \hat{\mathbf{q}})_{\mathcal{L}^2,\Omega} = \mathbf{M} \cdot \mathbf{q}, \tag{3.4}$$

and similarly, we have for the right-hand-side of (3.3):

$$\int_\Omega d\,(\phi \cdot \hat{\mathbf{q}}) - \int_\Omega \phi d\hat{\mathbf{q}} = \mathbf{B_w} \cdot \phi_{tr} - \mathbf{E^T W} \cdot \phi. \tag{3.5}$$

The discretised equation gives:

$$\mathbf{M} \cdot \mathbf{q} + \mathbf{E^T W} \cdot \phi = \mathbf{B_w} \cdot \phi_{tr}. \tag{3.6}$$

In addition, we have constraint in (3.2b) expressed by algebraic equation:

$$\mathbf{E} \cdot \mathbf{q} = f. \tag{3.7}$$

Then, we obtain a linear system of equations:

$$\begin{bmatrix} \mathbf{M} & \mathbf{E}^{\mathbf{T}}\mathbf{W} \\ \mathbf{W}^{\mathbf{T}}\mathbf{E} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \phi \end{bmatrix} = \begin{bmatrix} \mathbf{B}_{\mathbf{w}}\phi_{tr} \\ \mathbf{W}^{\mathbf{T}}f \end{bmatrix}. \tag{3.8}$$

One can also see a full derivation in [43], §6.2.

3.2. Spectral Elements on a Mapped Domain

Consider the domain $\boldsymbol{\Omega} \ni (x, y)$ mentioned in the previous section, we call it the reference domain. Now, we consider a physical domain denoted by $\tilde{\boldsymbol{\Omega}} \ni (\xi, \eta)$, and there exists a mapping (difeomorphism) Φ that maps the DoF from the reference domain to the physical domain. The $\boldsymbol{\mathcal{L}}^2$ inner product of functions on the physical domain is dependent on the metrics of the coordinate system.

We let $\hat{\mathbf{u}}^{(\mathbf{n-1})}_{(\boldsymbol{\xi},\boldsymbol{\eta})} := \Phi(\hat{\mathbf{q}}^{(\mathbf{n-1})}_{(\mathbf{x},\mathbf{y})})$ denotes the basis functions $\hat{\mathbf{q}}^{(\mathbf{n-1})}$ expressed on the mapped domain coordinates. And we use a tilde notation $\tilde{\circ}$ to denote the DoF on the mapped physical domain. Then (3.3) becomes:

$$(\tilde{\mathbf{q}},\ \hat{\mathbf{u}})_{\boldsymbol{\mathcal{L}}^2,\tilde{\hat{\Omega}}} = \left(\boldsymbol{\mathcal{J}}^{-1}\left(\frac{x,y}{\xi,\eta}\right)\mathrm{grad}\left(\tilde{\phi}\right), \boldsymbol{\mathcal{J}}^{-1}\left(\frac{x,y}{\xi,\eta}\right)\hat{\mathbf{q}}\right)_{\boldsymbol{\mathcal{L}}^2,\boldsymbol{\Omega}}, \tag{3.9a}$$

where $\boldsymbol{\mathcal{J}}$ is the Jacobian transformation matrix, also referred in differential geometry as the pullback of Φ.

3.3. p-version MSEM

Let $\mathbf{I} = [-1, 1] \in \mathbf{R}$, and a reference element on a square domain $\boldsymbol{\Omega} = \mathbf{I} \times \mathbf{I}$. The interior nodes are located on Legendre Gauss-Lobatto points $\{x_i | i = 0, \ldots, p\} \in \mathbf{I}$ where the degrees of freedom will be associated. To be more clear, Legendre Gauss-Lobatto points can be expressed in form:

$$x_i = cos(\varphi_i),$$

where x_i are the roots of Legendre polynomials of degree $p - 1$ and the end points of the interval $\mathbf{I}$. φ_i could be bounded by the interval (see [45]):

$$\frac{i}{p + 1/2}\pi \leq \varphi_i \leq \frac{i + 1/2}{p + 1/2}\pi, \qquad i = 0, 1, \ldots, p.$$

3.3.1. Lagrange Basis Function

1-D Lagrange polynomial basis function $l_n(x)$ is defined by:

$$l_n(x) = \prod_{0 \leq m \leq p}^{m \neq n} \frac{x - x_m}{x_n - x_m}, \qquad m, n = 0, 1, \ldots, p. \tag{3.10}$$

Using this basis function, we can construct basis functions for ϕ, thus the discrete representation of ϕ is given by:

$$\phi(x, y) \rightarrow \sum_{i,j} \phi_{ij} l_i(x) l_j(y). \tag{3.11}$$

3.3.2. Edge Basis Function

1-D edge basis function is derived from Lagrange basis function. The key of this basis function is to ensure that the discretised system mimics the same topological structure as the differential operators. The edge basis function is defined by:

Definition 3.3.1.

$$e_i(x) = \sum_{j=i}^{p} l'_j(x), \qquad i = 1, 2, ..., p. \tag{3.12}$$

We can use the combination of edge functions with Lagrange basis function to represent $\mathbf{q}$:

$$\mathbf{q} = \frac{\partial \phi}{\partial x} dx + \frac{\partial \phi}{\partial y} dy \rightarrow \sum_{i,j} q^x_{ij} e_i(x) l_j(y) dx + \sum_{i,j} q^y_{ij} e_i(y) l_j(x) dy. \tag{3.13}$$

It could be shown that the expansion factor $q^x_{i,j} = \phi_{i+1,j} - \phi_{i,j}$, and $q^y_{i,j} = \phi_{i,j+1} - \phi_{i,j}$. These are purely topological relations.

The two basis functions introduced above have been thoroughly discussed in Kreeft's PhD thesis [38], and an interpolation error estimate of thesis basis functions is also given in his work.

3.4. h-version MSEM with Domain Decomposition

Consider the same Poisson problem with Dirichlet boundary conditions, we partition the domain $\boldsymbol{\Omega}$ into subdomains $\boldsymbol{\Omega_i}$ that are not overlapping. For example, we can substructure a square domain into several subdomains as shown in Fig.3.1. We let $\boldsymbol{\Gamma_D}$ denote the real boundary of the global domain.

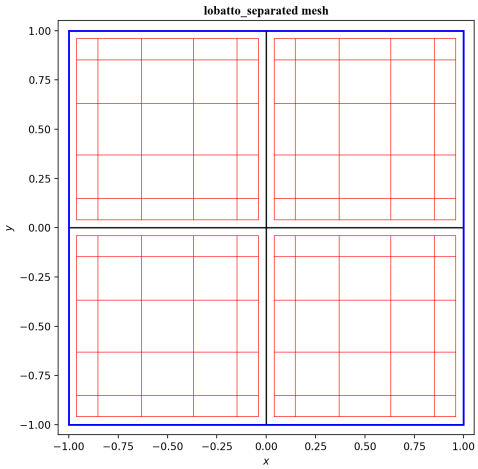

Figure 3.1: h-version MSEM Mesh with Substructures

We define the subproblems in each subdomain:

$$\mathbf{q}_i - \text{grad}(\phi_i) = \mathbf{0}, \qquad in\ \boldsymbol{\Omega}_i, \tag{3.14a}$$

$$\text{div}(\mathbf{q}_i) = f, \qquad in\ \boldsymbol{\Omega}_i, \tag{3.14b}$$

$$\phi_i = \phi_{tr}, \qquad on\ \partial\boldsymbol{\Omega}_i \cap \boldsymbol{\Gamma_D}, \tag{3.14c}$$

$$\phi_i = \phi_{in}, \qquad on\ \partial\boldsymbol{\Omega}_i \backslash \boldsymbol{\Gamma_D}. \tag{3.14d}$$

Each subproblems could be treated as independent p-version MSEM problems, e.g., in Fig.3.1, the four subproblems on the subdomains (coloured in red) are four independent p-version MSEM problem with polynomial order $p = 5$. The variables ϕ_{in} (stored on black edges) function as Lagrange multipliers, forcing the DoF on each side to be identical. The variables ϕ_{tr} (stored in blue edges) are determined by the given Dirichlet boundary conditions.

Using integration by part, we can write the sub-equation (3.14a) in the weak form:

$$(\mathbf{q}_i,\ \hat{\mathbf{q}})_{\mathcal{L}^2,\boldsymbol{\Omega}} + (\phi_i,\ \mathrm{div}(\hat{\mathbf{q}}))_{\mathcal{L}^2,\boldsymbol{\Omega}} - (\hat{\mathbf{q}}_i,\ \phi_{in})_{\mathcal{L}^2,\partial\boldsymbol{\Omega}\setminus\boldsymbol{\Gamma}_D} = (\hat{\mathbf{q}}_i,\ \phi_{tr})_{\mathcal{L}^2,\partial\boldsymbol{\Omega}_i\cap\boldsymbol{\Gamma}_D}\,, \tag{3.15}$$

where ϕ_{in} are the DoF that need to be found in between the elements. And to closure the system, we need Lagrange multipliers to force the equality of neighbouring $\mathbf{q}_i$ on the element boundaries, and we also need the divergence constraint in 3.14b. In this case, we choose to impose this equality in the trace form function space $\mathcal{H}^{1/2}$:

$$\left(\mathbf{q}_i,\ \hat{\phi}\right)_{\mathcal{L}^2,\ \partial\boldsymbol{\Omega}_i\setminus\boldsymbol{\Gamma}_D} = 0. \tag{3.16}$$

Thus the final discrete system will be of form:

$$\begin{bmatrix} \mathbf{M} & \mathbf{E^T W} & \mathbf{Wb} \\ \mathbf{W^T E} & \mathbf{0} & \mathbf{0} \\ \mathbf{Wb^T} & \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ \phi \\ \phi_{in} \end{bmatrix} = \begin{bmatrix} \mathbf{B_w}\phi_{tr} \\ \mathbf{W^T} f \\ \mathbf{0} \end{bmatrix}. \tag{3.17}$$

An extensive discussion on the block structure will be given in the coming Chapter 6.

3.5. Linear Equation System Solver

For a problem in physics, no matter at which level it is represented, the soul structure of the source problem will always be present. A Poisson problem in essence, is to find a steady state of a system where its potential energy is minimised. On the continuous level, the minimisation problem is to find function φ that minimise the energy integral:

$$\mathcal{F}(\varphi) \quad = \int \left\{-\tfrac{1}{2}\varphi\Delta\varphi\right\} d\Omega. \tag{3.18}$$

And for a mixed formulation, the minimisation problem degenerates to a constraint minimisation problem, that is to find a vector field $\mathbf{v}$ that minimise the integral:

$$\mathcal{J}(\mathbf{v}) \quad = \int \tfrac{1}{2} \mid \mathbf{v} \mid^2 d\Omega, \tag{3.19a}$$

$$\textit{subject to:} \qquad div(\mathbf{v}) = f. \tag{3.19b}$$

3.5.1. Uzawa Two-stage Solver

After discretisation, we obtained a linear system of equations in (3.8). To solve this system, we can use the Uzawa's algorithm as in (2.8). The first stage is to solve:

$$\mathbf{S}\phi = \mathbf{W^T E M^{-1} B_w}\phi_{tr} - \mathbf{W^T} f, \tag{3.20}$$

where, $\mathbf{S} = \mathbf{W^T E M^{-1} E^T W}$. $\mathbf{S}$ is SPD, thus a CG iterative method could be employed. In this case, we are minimising the norm:

$$\frac{1}{2}\phi^T \mathbf{S}\phi - \phi^T \mathbf{W^T}(\mathbf{E M^{-1} B_w}\phi_{tr} - f). \tag{3.21}$$

This minimisation problem is exactly mimicking the problem in (3.18).

The second stage could be solved by direct multiplication of $\mathbf{M^{-1}}$, since the inverse of the mass matrix is needed in the first stage.

4 Condition Analysis

From the last chapter, we see a hierarchy of problems with increasing complexity from p-MSEM to hp-MSEM, and also from orthogonal domain to arbitrarily mapped domain. Thus this chapter will make a thorough analysis of the linear equation system in (3.8), raised from p-version MSEM on an orthogonal domain.

4.1. Condition Analysis by Blocks

The LHS matrix in (2.5) is usually treated by block diagonalisation, thus the condition number of the blocks are inspected first. For rectangular matrix $\mathbf{E}$, we will inspect the condition number of the singular value spectrum.

First, the condition number growth against increasing polynomial orders is shown in Table 4.1. In this table, we evaluate the matrices with quadrature order equal to $2p$-1. Although in practical cases, the quadrature order is often chosen to be the same as discretisation order p. The reason is explained in §4.2.3.

Table 4.1: Condition Number of Matrices

p-order	5	9	13	25
$\mathbf{M}$	33.35	88.39	170.53	578.33
$\mathbf{S}$	29.90	106.39	248.40	1301.06
LHS	22.60	24.20	48.82	251.93
$\mathbf{E}\mathbf{E}^T$	13.93	39.86	78.77	273.31
$\mathbf{W}$	1.82	2.06	2.17	2.30

The condition number of wedge matrix $\mathbf{W}$ barely change with the p-order, because the wedge matrix is formed by the duality pairing of 2D nodal basis forms and 2D volume basis forms, which does not rely on the metrics. This duality pairing should return an approximate identity on an orthogonal domain. The condition number of the mass matrix is growing fast, thus preconditioners made for the mass matrix is widely discussed in literatures. The Schur complement matrix $\mathbf{S} = \mathbf{W}^T\mathbf{E}\mathbf{M}^{-1}\mathbf{E}^T\mathbf{W}$, has the fastest worsening condition. After inspection on the conditioning of the large matrix on the left, the coming sections will discuss them separately starting with the mass matrix.

4.2. Mass Matrix Condition Analysis

In this section, we will go through an approach to analyse the eigenvalue bounds and the condition of the mass matrix that we might encounter in MSEM.

4.2.1. Condition Number of $\mathbf{M}_l$ with 1D Nodal Basis Functions

A very important paper [42] given by Melenk showed the eigenvalue bounds of Spectral Elements with Gauss-Lobatto based basis functions. The intermediate steps before he arrive at his results are highly

relevant to our research. Thus some useful theorems and lemmas in his paper are taken out from this paper and its reference sources.

Proposition 4.2.0.1. *There are constants $c_1,\ c_2 > 0$ independent of p that help bound the Legendre polynomial:*

$$\frac{c_1}{1+(p+1/2)sin\varphi_i} \le L_p^2(x_i) \le \frac{c_2}{1+(p+1/2)sin\varphi_i},\ i \in \{0,1,...,p\}.$$

Proof. Since the Gauss-Lobatto points are symmetric, that is:

$$x_i + x_{p-i} = 0,\ i = 0,1,...,p$$

and $L_p^2(x)$ is an even function. Thus

$$L_p^2(x_i) = L_p^2(x_{p-i}),\ i = 0,1,...,p.$$

Then it is sufficient to investigate the circumstances for $i = 0,1,...,[p/2]+1$.

The Legendre polynomial could be written as the asymptotic expansion of uniformly convergent series of normalised Bessel functions (see [34] or [45] Theorem 8.21.6):

$$L_p(cos\varphi_i) = \sqrt{\frac{\varphi_i}{sin\varphi_i}} \cdot J_0\{(p+1/2)\varphi_i\} + O(p^{-\frac{3}{2}}), \qquad p \to \infty,$$

where J_0 is the Bessel function of the first kind with $m = 0$,

$$J_m(x) = \frac{(x/2)^m}{\sqrt{\pi}} \frac{\int_0^\pi [cos(xcos\theta)](sin\theta)^{2m} d\theta}{\Gamma(m+\frac{1}{2})}, \qquad m > -\frac{1}{2},$$

(integral representation by Poison see [2], Section 6.3). Thus

$$J_0(x) = \frac{2}{\pi}\int_0^\pi cos(xcos\theta)d\theta.$$

When x is taking a large positive value, we can further expand the first Bessel function in integral form (see [22], x1.4; or see [47],):

$$J_0(x) = \sqrt{\frac{2}{\pi x}} \sum_{k=0}^{\infty} \left\{ cos[(x-\frac{1}{4}\pi) - \frac{k\pi}{2}] \right\} \cdot \frac{\prod_{n=0}^{k}\{(2n+1)^2\}}{k!\cdot(8x)^k}.$$

As $x \to \infty$,

$$J_0(x) - \sqrt{\frac{2}{\pi x}}\left[cos\left(x-\frac{1}{4}\pi\right) \to O(x^{-\frac{3}{2}}\right].$$

Thus we have:

$$J_0^2(x) = \frac{1+sin(2x)}{\pi x} + O(x^{-2}), \qquad x \to \infty.$$

Recall that:

$$i\pi \le (p+1/2)\varphi_i \le (i+1/2)\pi, \qquad i = 0,1,...,[p/2]+1, \qquad p \to \infty.$$

Now, we first consider one limiting case where $i \to [p/2]+1$ while $p \to \infty$, that is: $\forall \varepsilon > 0$, there $\exists I_0 \in \{0,1,...,[p_0/2]+1\}$ and a sufficiently large $p_0 \in \mathbf{N}$, such that $\forall i > I_0$ and $p > p_0$, the following inequality holds:

$$\begin{aligned}
&\left| L_p^2(x_i) - \frac{1+sin(2(p+\frac{1}{2})\varphi_i)}{\pi(p+\frac{1}{2})sin\varphi_i} \right| \\
&= \left| L_p^2(x_i) - \frac{\varphi_i}{sin\varphi_i} J_0^2((p+\frac{1}{2})\varphi_i) + \frac{\varphi_i}{sin\varphi_i} J_0^2((p+\frac{1}{2})\varphi_i) - \frac{1+sin(2(p+\frac{1}{2})\varphi_i)}{\pi(p+\frac{1}{2})sin\varphi_i} \right| \\
&\le \left| L_p^2(x_i) - \frac{\varphi_i}{sin\varphi_i} J_0^2((p+\frac{1}{2})\varphi_i) \right| + \left| \frac{\varphi_i}{sin\varphi_i} J_0^2((p+\frac{1}{2})\varphi_i) - \frac{1+sin(2(p+\frac{1}{2})\varphi_i)}{\pi(p+\frac{1}{2})sin\varphi_i} \right| \\
&\le \lambda_1 p_0^{-\frac{2}{3}} + \lambda_2 I_0^{-2} = \varepsilon,
\end{aligned}$$

where $\lambda_1,\ \lambda_2$ are bounded coefficients from asymptotic expansions.

Since $1 + sin(2(p + \frac{1}{2})\varphi_i) \in [1, 2]$, it is easy to find positive constants $c_1,\ c_2$ independent from p such that:

$$\frac{c_1}{1 + (p + \frac{1}{2})sin\varphi_i} \leq L_p^2(x_i) \leq \frac{c_2}{1 + (p + \frac{1}{2})sin\varphi_i}.$$

It remains to see that for cases when $0 \leq i \leq I_0$, while $p \to \infty$. We have:

$$\frac{\varphi_i}{sin\varphi_i} J_0^2((p + \frac{1}{2})\varphi_i) - \lambda_1 p_0^{-\frac{2}{3}} \leq L_p^2(x_i) \leq \frac{\varphi_i}{sin\varphi_i} J_0^2((p + \frac{1}{2})\varphi_i) + \lambda_1 p_0^{-\frac{2}{3}}.$$

According to Schafheitlin's investigation for the zeros of $J_o(x)$ (see [47], 15.32), zeros of $J_0(x)$ are only lying in the intervals $[i\pi + \frac{3}{4}\pi, i\pi + \frac{7}{8}\pi],\ (i = 0, 1, \ldots)$. Thus $J_0^2((p + \frac{1}{2})\varphi_i)$ is always positive and finite for all $i \in (0, I_0]$. Thus again, we can find proper positive constant $c_1,\ c_2$ such that:

$$\frac{c_1}{1 + (p + \frac{1}{2})sin\varphi_i} \leq L_p^2(x_i) \leq \frac{c_2}{1 + (p + \frac{1}{2})sin\varphi_i}.$$

For a special limit case when $i = 0$, $sin(\varphi_i) = 0$ and $L_p(x_i) = 1$. This case fits in the proposition. □

The quadrature weights with LGL points is given by:

$$\rho_i = \frac{2}{p(p + 1)L_p^2(x_i)}. \tag{4.1}$$

Then we are able to compute $\mathcal{L}^2$ norm with discrete nodal values of function:

$$\| \phi(x) \|_{\mathcal{L}^2(\mathbf{I})}^2 = \sum_{i=0}^{p} (\rho_i \phi^2(x_i)). \tag{4.2}$$

Based on this quadrature rule, we have the following theorem.

Theorem 4.2.1. *Let* $\mathbf{I} = (-1, 1)$*, and* $(x_i)_{i=0}^p \in \mathbf{I}$ *are the LGL points,* $p \in \{1, \ldots\}$*. We have a function* $\phi(x) \in \mathcal{H}^1(\mathbf{I})$ *that could be expanded by the nodal basis functions in (3.10):* $\phi(x) = \sum_{i=0}^{p} \phi(x_i) l_i(x)$*. We express expansion coefficients* $\phi(x_i)$ *into vector* $\boldsymbol{\phi} \in \mathbf{R^{p+1}}$*. Then there exists two positive constant* C_1, C_2 *independent of p such that:*

$$C_1 p^{-2} \leq \frac{\| \phi(x) \|_{\mathcal{L}^2(\mathbf{I})}^2}{\| \boldsymbol{\phi} \|_2^2} \leq C_2 p^{-1}.$$

Proof. From the LGL quadrature rule in 4.2, we have:

$$\min\{\rho_i\} \sum_{i=0}^{p} \phi^2(x_i) \leq \| \phi(x) \|_{\mathcal{L}^2(\mathbf{I})}^2 \leq \max\{\rho_i\} \sum_{i=0}^{p} \phi^2(x_i).$$

According to the quadrature rule,

$$\min\{\rho_i\} = \frac{2}{p(p + 1)} \frac{1}{\max\{L_p^2(x_i)\}} \geq \frac{2}{c_2 p(p + 1)} \sim o(p^{-2}), \qquad p \to \infty,$$

and

$$\max\{\rho_i\} = \frac{2}{p(p + 1)} \frac{1}{min\{L_p^2(x_i)\}} \leq \frac{2(1 + (p + \frac{1}{2}))}{c_1 p(p + 1)} \sim o(p^{-1}), \qquad p \to \infty.$$

Since c_1, c_2 are independent of p, the theorem holds for properly chosen constants C_1, C_2. □

Theorem 4.2.1 indicates that the condition number of the 1D problem mass matrix using $\mathbf{M_l}$ scales with polynomial order p. To verify this is not only correct but also sharp, we set up a 1D mass matrix using polynomial orders varying from 5 to 81, and then examine the variation of eigenvalues and condition number. The results are given in the logarithmic plot in Fig. 4.1.

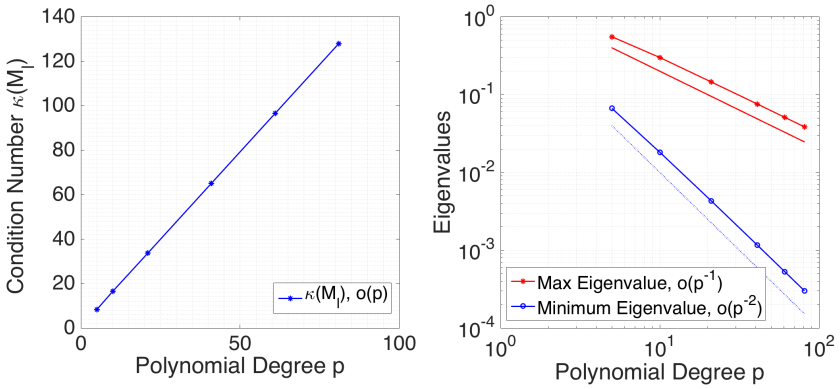

Figure 4.1: Conditioning of $\mathbf{M_l}$

4.2.2. Condition Number of $\mathbf{M_e}$ with 1D Edge Basis Functions

We can express $u(x)$, the derivative of function $\phi(x)$ as in Theorem 4.2.1, with edge basis functions:

$$u(x) = \frac{d\phi}{dx} = \sum_{i=1}^{p} u(x_i) e_i(x), \tag{4.3}$$

and we have an important property of the expansion coefficients:

$$u(x_i) = \phi(x_i) - \phi(x_{i-1}),\ i = 1, 2, ..., p, \tag{4.4}$$

which is a purely topological relation, only determined by the topology of the mesh, independent from mapping.

Since edge function is already introduced, we have two ways to express the derivative of an $\mathcal{H}^1$ function. One way is to use edge basis function, and another way is to use the traditional way:

$$\frac{d\phi}{dx} = \frac{d}{dx}\left(\sum_{i=0}^{p} \phi_i l_i(x)\right) = \sum_{i=0}^{p} \phi_i l_i'(x). \tag{4.5}$$

Now we inspect the inequality between $\mathcal{L}^2$ norm of $\frac{d\phi}{dx}$ and discrete vector 2-norm of $\boldsymbol{\phi_i}$ in the following content.

First, we define a matrix $H \in \mathbf{R^{(p+1)\times(p+1)}}$ given by:

$$H_{ij} = \rho_i l_j'(x_i) = \frac{2}{p(p+1)L_p^2(x_i)} l_j'(x_i),\ i, j = 0, 1, ..., p. \tag{4.6}$$

Then, we immediately have 4.7 according to the identities of Jacobi Polynomials (see [21]):

$$H_{ij} = \begin{cases} \frac{1}{4} & \text{if } i = j = 0, \\ -\frac{1}{4} & \text{if } i = j = p, \\ 0 & \text{if } i = j \text{ and } i \notin \{0, p\}, \\ \frac{2}{p(p+1)L_p(x_i)L_p(x_j)(x_i - x_j)} & \text{else.} \end{cases} \tag{4.7}$$

Followed by this definition, we have the coming proposition.

Proposition 4.2.1.1. *$\forall p \in \mathbf{N}$, $\exists C > 0$, such that the induced 2-norm $\|H\boldsymbol{\phi}\|_2 \leq C \|\boldsymbol{\phi}\|_2$, for all $\phi(x)$ that is not constant.*

The proof of the right inequality is given in [42] (proposition 4.9 and corollary 4.10). What is worth to point out is that the matrix H has a row sum equal to zero, which is because that there are $(p+1)$ $l_j'(x)$ functions with polynomial degree $p-1$, which are not linearly independent from each other. This is indicating that H is not only singular, but also have at least 2 zero eigenvalues. We get the same conclusion if we inspect the centrosymmetry of the H matrix with respect to the point x_m, $m = \left[\frac{p+1}{2}\right]$.

Theorem 4.2.2. *There exists a constant C independent of the polynomial order p such that:*

$$\|u\|^2_{\mathcal{L}^2(\mathbf{I})} \leq Cp^2 \|\boldsymbol{\phi}\|_2^2 .$$

Proof.

$$\begin{aligned}
\|u\|^2_{\mathcal{L}^2(\mathbf{I})} &= \sum_{i=0}^{p} \rho_i u^2(x_i) \\
&\leq \sum_{i=0}^{p} \frac{\rho_i}{\min\{\rho_i\}} \rho_i u^2(x_i) \\
&\underset{Theo.4.2.1}{\leq} Cp(p+1) \sum_{i=0}^{p} (\rho_i u(x_i))^2 \\
&= Cp(p+1) \sum_{i=0}^{p} \Big(\sum_{j=0}^{p} \rho_i l_j'(x_i)\phi_j\Big)^2 \\
&= Cp(p+1) \|H\boldsymbol{\phi}\|_2^2 \\
&\underset{Prop.4.2.1.1}{\leq} Cp^2 \|\boldsymbol{\phi}\|_2^2 .
\end{aligned}$$

Notice, in each step, the constant C are updated, but does not affect the fact that it is independent from the choice of p. □

In the theorem above, since the maximum of ρ_i could be achieved by a properly chosen $u(x)$, e.g., $u(x) = l_0'(x)$. In this case, the maximum 2-norm of the row vector space of H, i.e., $\max_i\{\sum_j H_{ij}^2\}$, is sharply bounded by a constant independent of p.

The sharpness of the above theorem ensured that the following techniques will work well. We are going to make use of the mimetic property of the method to further link $\|\boldsymbol{\phi}\|_2$ with $\|\mathbf{u}\|_2$. To compute $\mathbf{u}$, we rewrite (4.4) as $\mathbf{u} = \mathbf{E_l}\boldsymbol{\phi}$, with the incidence matrix $\mathbf{E_l}$:

$$\mathbf{E_l} = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}_{p \times p+1} . \tag{4.8}$$

Thus $\|\mathbf{u}\|_2^2 = \mathbf{u}\mathbf{u}^T = \boldsymbol{\phi}^T \mathbf{E_l}^T \mathbf{E_l}\boldsymbol{\phi}$. We let $\boldsymbol{T}_1 = \mathbf{E_l}^T \mathbf{E_l}$, and $\boldsymbol{T}_1$ is a positive semidefinite square matrix with p positive eigenvalues $\lambda(\boldsymbol{T}_1) = \sigma^2(\mathbf{E_l})$, where σ are the singular values of the incidence matrix $\mathbf{E_l}$. Thus to find $\lambda(\boldsymbol{T}_1)$ is to find eigenvalues of the Toeplitz matrix $\boldsymbol{T}_2 = \mathbf{E_l}\mathbf{E_l}^T = blokdiag(-1, 2, -1)$.

Proposition 4.2.2.1. *The eigenvalues of the Toeplitz matrix $\boldsymbol{T}_p(a,b,c) = blockdiag(a,b,c)$ are: $a - 2\sqrt{bc} \cdot cos((i\pi)/(p+1))$, $i = 1, 2, ..., p$.*

Proof. This proposition has been proven in many textbooks about Toeplitz matrix and circulant matrices. See [39]. □

The proposition above indicates that $\lambda_i(\boldsymbol{T}_1) = 2[1-cos(i\pi/(p+1))]$, $i = 0, 1, 2, ..., p$. Notice that we absorbed the only zero eigenvalue of $\boldsymbol{T}_1$ in the case of $i = 0$. We have the following theorem as a result of the singular value decomposition of $\mathbf{E_l}$.

Theorem 4.2.3. *The matrix* $\mathbf{E_l}$ *admits to the singular value decomposition:* $\mathbf{E_l} = \mathbf{P\Sigma Q}$*, where* $\mathbf{P}_{p\times p}$ *and* $\mathbf{Q}_{(p+1)\times(p+1)}$ *are orthogonal matrices, and* $\mathbf{\Sigma}$ *is a square diagonal matrix with singular values in an increasing order:* $\sigma_i = \sqrt{2-2cos(i\pi/(p+1))}$, $(i = 1, ..., p)$*. Further more, there exists a constant* C *independent of* p *such that* $\sigma_i \in [Cp^{-1}, 2]$*. As* $p \to \infty$, $\sigma_1 \sim Cp^{-1}$*, and* $\sigma_p \to 2$.

Proof. From proposition 4.2.2.1, it can be inferred that $\lambda_i(\boldsymbol{T}_1) \in [0, 4]$. When $p \to \infty$, the following Taylor expansion around point $x \to \frac{\pi}{p+1}$ hold:

$$\begin{aligned}\lambda_1 &= 2[1-cos\,(\pi/(p+1))] \\ &= 2[1-\{cos\left(\frac{\pi}{p+1}\right)-\frac{\pi}{p+1}sin\left(\frac{\pi}{p+1}\right)-\frac{1}{2}\left(\frac{\pi}{p+1}\right)^2 cos\left(\frac{\pi}{p+1}\right)+o(p^{-3})\}] \\ &= \left(\frac{\pi}{p+1}\right)^2+o(p^{-3}) \\ &\sim p^{-2},\end{aligned}$$

and $\lambda_p \to 4$. Then the theorem is obvious. □

Theorem 4.2.4. *There exists a constant* C *independent of the polynomial order* p *such that:*

$$\|u\|^2_{\mathcal{L}^2(\mathbf{I})} = \mathbf{u}^T\mathbf{M_e}\mathbf{u} \le Cp^2\,\|\mathbf{u}\|_2^2.$$

Proof. The $\mathcal{L}^2$ norm of function u could be computed by quadrature:

$$\|u\|^2_{\mathcal{L}^2(\mathbf{I})} = \mathbf{u}^T\mathbf{M_e}\mathbf{u} = \boldsymbol{\phi}^T[\mathbf{Q}^T\mathbf{\Sigma}^T\mathbf{P}^T]\mathbf{M_e}[\mathbf{P\Sigma Q}]\boldsymbol{\phi} \le Cp^2\,\|\boldsymbol{\phi}\|_2^2.$$

The last inequality is taken from theorem 4.2.2. Thus it can be inferred that the upper bound of the eigenvalue of matrix: $[\mathbf{Q}^T\mathbf{\Sigma}^T\mathbf{P}^T]\mathbf{M_e}[\mathbf{P\Sigma Q}]$ is Cp^2. Since the orthogonal matrices $\mathbf{P}$ and $\mathbf{Q}$ have vector induced 2-norm equal to 1, and the maximum singular value of $\mathbf{\Sigma}$ is always bounded by 2, we assert that the maximum eigenvalue of the mass matrix $\mathbf{M_e}$ is bounded by Cp^2, with chosen C independent of p.

□

Further, the null space of $\mathbf{E_l}$ is a set of vectors with uniform values of $p+1$ entries, i.e., when $\phi(x)$ is constant. Thus for the case where $\boldsymbol{\phi}$ is not in the null space of $\mathbf{E_l}$, the theorem above is always sharp.

The theorems above could be seen as an extension of Melenk's results. But due to the lack of knowledge about the lower eigenvalue bound of H_{ij}, the lower eigenvalue of $\mathbf{M_e}$ will be hard to find out. However, we can make use of the properties of the edge function itself to make a rough estimation of the spectrum limits of $\mathbf{M_e}$.

Proposition 4.2.4.1. *The edge basis function satisfy:*

$$\int_{x_{j-1}}^{x_j} e_i(x)dx = \delta_{ij},$$

for $i, j = 1, 2, ..., p$.

Proof. If we work out the integral by definition in 3.3.1, we have:

$$\int_{x_{j-1}}^{x_j} e_i(x)dx = \int_{x_{j-1}}^{x_j}\sum_{k=i}^{p} l_k'(x)dx \tag{4.9a}$$

$$= \sum_{k=i}^{p} l_k(x)\Big|_{x_{j-1}}^{x_j} \tag{4.9b}$$

$$= \delta_{ij} \tag{4.9c}$$

□

Cauchy mean value theorem states the simple fact that there exists a point $\xi_j \in [x_{j-1}, x_j]$, such that $e_i(\xi_j) \cdot (x_j - x_{j-1}) = \delta_{ij}$. Thus we have:

$$e_i(\xi_j) = \frac{\delta_{ij}}{x_j - x_{j-1}}, \tag{4.10}$$

where $i, j = 1, 2, ..., p$. Notice that the quadrature points and weights used with edge functions are Gauss Legendre points $\eta_j \in [x_{j-1}, x_j]$, which is inserted between the GLL points. Since the $(x_j - x_{j-1}) \to 0$ as $p \to \infty$, $\xi_j \to \eta_j$ is expected. Thus at η, the quadrature points of edge functions, we have:

$$e_i(\eta_j) \to \frac{\delta_{ij}}{x_j - x_{j-1}}, \tag{4.11}$$

with $i, j = 1, 2, ..., p$. Thus we expect that matrix $e_i(\eta_j)$ is diagonally dominant. Also, for the diagonal entries, we have:

$$Cp\delta_{ij} \leq \frac{\delta_{ij}}{x_j - x_{j-1}} \leq Cp^2\delta_{ij}. \tag{4.12}$$

Now, we give an estimation by numerical experiment instead of a pure theoretical proof. The following Fig. 4.2 shows the conditioning of the mass matrix with edge basis functions:

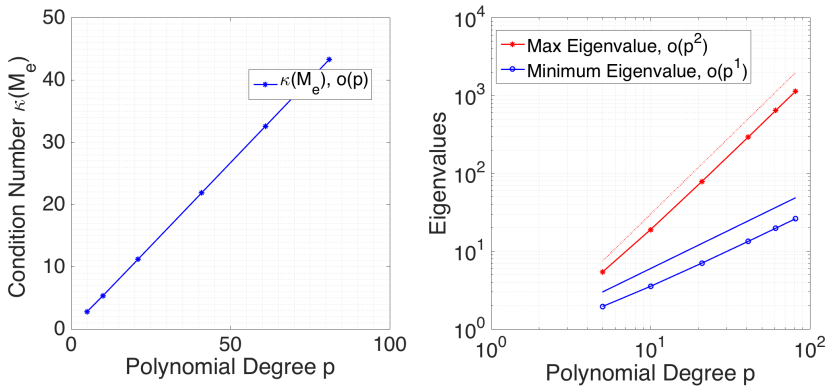

Figure 4.2: Conditioning of $\mathbf{M_e}$

4.2.3. Condition Number of Mass Matrix M with 2D Basis Functions

The condition number of the mass matrix in problem (3.8) could now be estimated. In fact, our ambition could be a bit larger than that. The coming analysis will help us analyse the condition of any mass matrix on a reference element (orthogonal domain).

First, we take the mass matrix $\mathbf{M}$ as an example. This mass matrix is structured in 2×2 blocks as in the following equation.

$$\mathbf{M} = \begin{bmatrix} (\hat{q}_x, \hat{q}_x)_{\mathcal{L}^2, \hat{\Omega}} & (\hat{q}_x, \hat{q}_y)_{\mathcal{L}^2, \hat{\Omega}} \\ (\hat{q}_y, \hat{q}_x)\,\mathcal{L}^2, \hat{\Omega} & (\hat{q}_y, \hat{q}_y)\,\mathcal{L}^2, \hat{\Omega} \end{bmatrix} \tag{4.13}$$

On the orthogonal domain $\hat{\Omega}$, the off-diagonal block of $\mathbf{M}$ are zero. Each element in the $(1, 1)$ or $(2, 2)$ block of $\mathbf{M}$ is calculated by the $\mathcal{L}^2$ inner product on $\hat{\mathbf{\Omega}}$ as in (4.14a). If we take the $(2, 2)$ block as an example, we get the rest integrals.

$$m_{(K_1,K_2)} = \left(\hat{\mathbf{q}}^{\mathbf{n-1}}_{K_1(r,q)}\,,\ \hat{\mathbf{q}}^{\mathbf{n-1}}_{K_2(m,n)}\right)_{\mathcal{L}^2,\hat{\mathbf{\Omega}}} \tag{4.14a}$$

$$= \iint_{\Omega} l_p(x)e_q(y) \cdot l_m(x)e_n(y)dxdy \tag{4.14b}$$

$$= \int_{x_0}^{x_p} l_r(x)l_m(x)dx \cdot \int_{y_0}^{y_p} e_q(y)e_n(y)dy \tag{4.14c}$$

$$= \lambda_{(r,m)} \cdot \mu_{(q,n)} \tag{4.14d}$$

In this equation, (r, q) and (m, n) are numbering pairs for the 2D $\hat{\mathbf{q}}$ basis functions. However, while building the mass matrix, $\hat{\mathbf{q}}$ are stored in a 1D sequence. K_1 and K_2 are functions that arranges the counting sequence while putting the 2D structured basis functions into a 1D sequence. The equation above also showed that the $(2, 2)$ block of the mass matrix could be factored by a Kronecker product of two 1D mass matrices that we made analysis before. λ and μ are elements from mass matrix $\mathbf{M_l}$ and $\mathbf{M_e}$.

Before we proceed to the condition analysis of $\mathbf{M}$, we give a brief introduction of the definition and properties of matrix Kronecker product to support our analysis.

Definition 4.2.1. *Kronecker Product: Let* $\mathbf{A} \in \mathbf{R^{m\times n}}$, $\mathbf{B} \in \mathbf{R^{i\times j}}$ *be two random matrices, there elements are denoted by* $A_{(m,n)}$ *and* $B_{(i,j)}$ *with explicite rows and columns. The Kronecker product of* $\mathbf{A}$ *and* $\mathbf{B}$ *is denoted by* $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$*, such that* $\mathbf{C}$ *follows the block structure:*

$$\mathbf{C} = \begin{bmatrix} A_{(1,1)}\mathbf{B} & A_{(1,2)}\mathbf{B} & \dots & A_{(1,n)}\mathbf{B} \\ A_{(2,1)}\mathbf{B} & A_{(2,2)}\mathbf{B} & \dots & A_{(2,n)}\mathbf{B} \\ \dots & \dots & \dots & \dots \\ A_{(m,1)}\mathbf{B} & A_{(m,2)}\mathbf{B} & \dots & A_{(m,n)}\mathbf{B} \end{bmatrix}.$$

Now we are able to write the 2D mass matrix into the following form:

$$\mathbf{M} = \begin{bmatrix} \mathbf{M_e} \otimes \mathbf{M_l} & \mathbf{0} \\ \mathbf{0} & \mathbf{M_l} \otimes \mathbf{M_e} \end{bmatrix} \tag{4.15}$$

Followed by the definition 4.2.1, we can give the mixed-product property.

Theorem 4.2.5. *If we have matrices* $\mathbf{A}, \mathbf{B}, \mathbf{C}$*, and* $\mathbf{D}$*,* $\mathbf{A}$ *and* $\mathbf{C}$ *are multiplicable,* $\mathbf{B}$ *and* $\mathbf{D}$ *are multiplicable. Then,*

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = (\mathbf{AC}) \otimes (\mathbf{BD}).$$

This theorem is obviously resulted from the definition, and is very important since we will have the following corollary on the spectral decomposition property by Kronecker product.

Corollary 4.2.5.1. *If a real matrix* $\mathbf{M}$ *is formed by Kronecker product of two matrices* $\mathbf{A}$ *and* $\mathbf{B}$*. There exists two unique unitary matrices* $\mathbf{P}$ *and* $\mathbf{Q}$ *that does the diagonalisation:* $\mathbf{A} = \mathbf{P\Lambda_a P^T}$*, and* $\mathbf{B} = \mathbf{Q\Lambda_b Q^T}$*. Thus, we have:*

$$\mathbf{M} = \mathbf{A} \otimes \mathbf{B} = ((\mathbf{P\Lambda_a}) \otimes (\mathbf{Q\Lambda_b}))(\mathbf{P^T} \otimes \mathbf{Q^T}) = (\mathbf{P} \otimes \mathbf{Q})(\mathbf{\Lambda_a} \otimes \mathbf{\Lambda_b})(\mathbf{P^T} \otimes \mathbf{Q^T}).$$

Combining the information covered above, we can give the most important Theorem 4.2.6.

Theorem 4.2.6. *The minimum (maximum) eigenvalue of mass matrix* $\mathbf{M}$ *is the product of the minimum (maximum) eigenvalue of* $\mathbf{M_l}$ *and* $\mathbf{M_e}$*. And naturally,* $\kappa(\mathbf{M}) = \kappa(\mathbf{M_e}) \cdot \kappa(\mathbf{M_l})$.

The above theorem states that the minimum eigenvalue of $\mathbf{M}$ scales with p^{-1} if the experimental result of the minimum eigenvalue of $\mathbf{M_e}$ is correct and sharp. And it can be asserted that the maximum eigenvalue of $\mathbf{M}$ scales with p, proven by theorem. And lastly, $\kappa(\mathbf{M}) \sim p^2$. These results are verified by numerical tests and are correct so far. The only imperfection is that the $\min\{eig(\mathbf{M_e})\}$ is only given by experiment instead of a proven theorem.

In the end, we show the shape and sparsity pattern of the mass matrix $\mathbf{M}$ for $p = 5$. In Fig. 4.3, the $(1, 1)$ block of the mass matrix is permuted to be of the form $\mathbf{M_l} \otimes \mathbf{M_e}$, thus is the same as the $(2, 2)$ block.

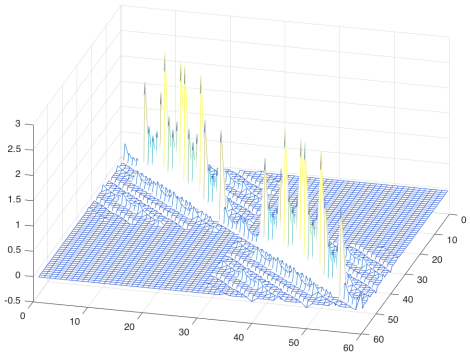

Figure 4.3: $\mathbf{M}$ with Quadrature Order $2p-1$

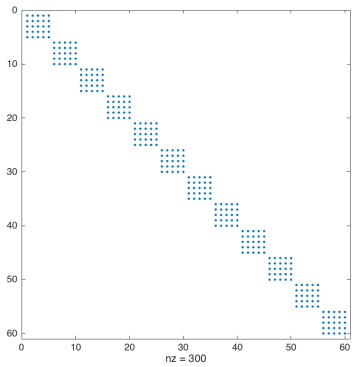

Figure 4.4: Sparsity Pattern of $\mathbf{M}$ with Quadrature Order $p=5$

The quadrature order is chosen to be $2p-1$ in Fig.4.3, thus the integral is exact. In applications, it is usually safe to set the quadrature order to be the same as p, since the discretisation error with respect to p is usually larger than the integration error with respect to p. If the quadrature order is the same as the discretisation order, the 1D mass matrix $\mathbf{M_l}$ is a purely diagonal matrix, and the mass matrix $\mathbf{M}$ will be block diagonal, as shown in Fig.4.4. However, if the discretisation order is high enough such that interpolation error is dominant. We give a surface plot below to demonstrate Kronecker product in a graphical sense. On the left is the shape of a block of 2D mass matrix $\mathbf{M}$. In the middle is the shape of the 1D mass matrix $\mathbf{M_l}$ and the on the right is the 1D mass matrix $\mathbf{M_e}$.

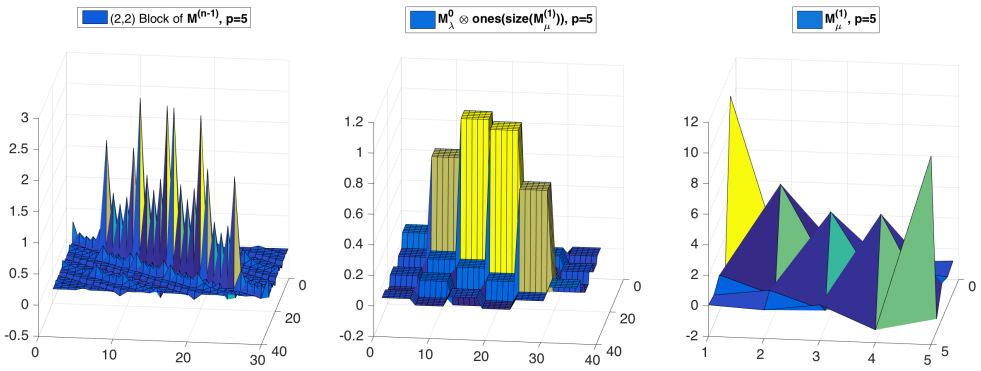

Figure 4.5: Kronecker Product Decomposition of Mass Matrix

4.3. Condition of the 2D Incidence Matrix

The 1D incidence matrix is already introduced in §4.2.2. Using this 1D incidence matrix $\mathbf{E_l}$, we can construct the 2D incidence matrix $\mathbf{E}$ in problem (3.8). One of the biggest difference between MSEM and hp-FEM is the application of the incidence matrices instead of the differentiation $\mathcal{D}$ matrix with $l_i'(x_j)$, which is purely topological. The 2D incidence matrix could be written in 2×1 blocks which is corresponding to the 2×2 block of the mass matrix:

$$\mathbf{E} = \begin{bmatrix}\mathbf{E_x} & \mathbf{E_y}\end{bmatrix}. \tag{4.16}$$

And the blocks are given by the Kronecker product of 1D incidence matrix $\mathbf{E}_{\mathbf{l},(p+1)\times p}$ with an identity matrix $\boldsymbol{I}_{p\times p}$:

$$\mathbf{E_x} = \mathbf{E_l} \otimes \boldsymbol{I} \tag{4.17a}$$

$$\mathbf{E_y} = \boldsymbol{I} \otimes \mathbf{E_l} \tag{4.17b}$$

. In order to show the Kronecker product structure of $\mathbf{E}$, we give a sparsity pattern plot in Fig. 4.6 with $p = 5$.

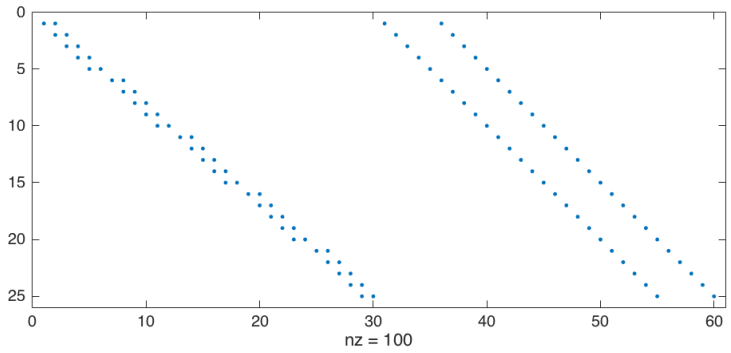

Figure 4.6: Sparsity Pattern of $\mathbf{E}$ with $p = 5$

Now, in order to analyse the conditioning of the incidence matrix, we perform a singular value decomposition admitting the block structure. We have the following equations.

$$\mathbf{E}\mathbf{E}^{\mathbf{T}} = \mathbf{E_x}\mathbf{E_x}^T + \mathbf{E_y}\mathbf{E_y}^T \tag{4.18a}$$
$$= (\mathbf{E_l} \otimes \boldsymbol{I})(\mathbf{E_l}^T \otimes \boldsymbol{I}) + (\boldsymbol{I} \otimes \mathbf{E_l})(\boldsymbol{I} \otimes \mathbf{E_l}^T) \tag{4.18b}$$
$$= (\mathbf{E_l}\mathbf{E_l}^T \otimes \boldsymbol{I}) + (\boldsymbol{I} \otimes \mathbf{E_l}\mathbf{E_l}^T) \tag{4.18c}$$

Until now, we have two points to make. One is that the two matrices $(\mathbf{E_l}\mathbf{E_l}^T \otimes \boldsymbol{I})$ and $(\boldsymbol{I}) \otimes (\mathbf{E_l}\mathbf{E_l}^T)$ are spectrally equivalent, which is a result of theorem 4.2.5.1. Another less obvious point is that these two matrices could be diagonalised by the same orthogonal transformation. To see this, one can check that:

$$(\mathbf{E_l}\mathbf{E_l}^T \otimes \boldsymbol{I})(\boldsymbol{I} \otimes \mathbf{E_l}\mathbf{E_l}^T) = (\boldsymbol{I} \otimes \mathbf{E_l}\mathbf{E_l}^T)(\mathbf{E_l}\mathbf{E_l}^T \otimes \boldsymbol{I}) = \mathbf{E_l}\mathbf{E_l}^T \otimes \mathbf{E_l}\mathbf{E_l}^T, \tag{4.19}$$

according to the mixed product property.

Given the two points above, the eigenvalues of $\mathbf{E}\mathbf{E}^{\mathbf{T}}$, if denoted by $\lambda_{i,j}(\mathbf{E}\mathbf{E}^T)$, could be given by:

$$\lambda_{i,j}(\mathbf{E}\mathbf{E}^T) = \lambda_i(\mathbf{E_l}\mathbf{E_l}^T) + \lambda_j(\mathbf{E_l}\mathbf{E_l}^T) \tag{4.20a}$$
$$= 4 - 2cos(i\pi/(p+1)) - 2cos(j\pi/(p+1)), \tag{4.20b}$$

where $i, j = 1, 2, ..., p$. Thus we conclude that 2D incidence matrix $\mathbf{E}$ has p^2 singular values:

$$\sigma_{i,j}(\mathbf{E}) = \sqrt{4 - 2cos(i\pi/(p+1)) - 2cos(j\pi/(p+1))}, \tag{4.21}$$

where $i, j = 1, 2, ..., p$. Moreover, there exists a $C > 0$ independent from p, such that $\sigma_{min} > Cp^{-1}$ and $\sigma_{max} \to 2\sqrt{2}$.

4.4. Condition of the Wedge Matrix W

The wedge matrix could be analysed just as the mass matrix. Entries in **W** are calculated by:

$$W_{(K_1,K_2)} = \left\langle \hat{\boldsymbol{\phi}}^{(\mathbf{0})}_{K_1(r,q)} , \hat{\mathbf{f}}^{(\mathbf{2})}_{K_2(m,n)} \right\rangle_{\mathcal{L}^2,\hat{\Omega}} \tag{4.22a}$$

$$= \iint_{\Omega} e_r(x)e_q(y) \cdot l_m(x)l_n(y)dxdy \tag{4.22b}$$

$$= \int_{x_0}^{x_p} e_r(x)l_m(x)dx \cdot \int_{y_0}^{y_p} e_q(y)l_n(y)dy \tag{4.22c}$$

$$= w_{(r,m)} \cdot w_{(q,n)}. \tag{4.22d}$$

The wedge matrix could be written as the Kronecker product of 1D wedge matrix $\mathbf{W_l}$:

$$\mathbf{W} = \mathbf{W_l} \otimes \mathbf{W_l}. \tag{4.23}$$

The elements in 1D wedge matrix could be calculated by quadrature in order of p:

$$w_{(r,m)} = \int_{x_0}^{x_p} e_r(x)l_m(x)dx \tag{4.24a}$$

$$= \sum_{k=0}^{p} \rho_k e_r(x_k)l_m(x_k) \tag{4.24b}$$

$$= \sum_{k=0}^{p} \rho_k e_r(x_k)\delta_{mk} \tag{4.24c}$$

$$= \rho_m e_r(x_m). \tag{4.24d}$$

From the definition of the edge basis function in 3.3.1, we know that the row sum of $\mathbf{W_l}$ is 1. Moreover, from §4.2.2, we immediately see that $\mathbf{W_l}$ should be approximately an identity matrix. According to the property of Kronecker product, we draw a conclusion that **W** is also approximately an identity matrix, which is also confirmed by the numerical test in Table 4.1.

4.5. Condition of the Schur Complement of Mass Matrix

Now we can write out the full system of equations of the saddle point problem in the following pattern:

$$\begin{bmatrix} \mathbf{M_e} \otimes \mathbf{M_l} & \mathbf{0} & \mathbf{E_x}^T\mathbf{W} \\ \mathbf{0} & \mathbf{M_l} \otimes \mathbf{M_e} & \mathbf{E_y}^T\mathbf{W} \\ \mathbf{W}^T\mathbf{E_x} & \mathbf{W}^T\mathbf{E_y} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{q_x} \\ \mathbf{q_y} \\ \boldsymbol{\phi} \end{pmatrix} = \begin{pmatrix} \mathbf{B_w}\phi_{x\_tr} \\ \mathbf{B_w}\phi_{y\_tr} \\ \mathbf{W^T}f \end{pmatrix}. \tag{4.25}$$

Thus the Schur complement of the mass matrix could be written as follows.

$$\mathbf{S} = \mathbf{W}^T\mathbf{E}\mathbf{M}^{-1}\mathbf{E}^T\mathbf{W} \tag{4.26a}$$

$$= \mathbf{W}^T[(\mathbf{E_l}\mathbf{M_l}^{-1}\mathbf{E_l}^T) \otimes (\mathbf{M_e}^{-1}) + (\mathbf{M_l}^{-1}) \otimes (\mathbf{E_l}\mathbf{M_e}^{-1}\mathbf{E_l}^T)]\mathbf{W}. \tag{4.26b}$$

Unfortunately, the two Kronecker product matrix in (4.26b) cannot be diagonalised simultaneously. However, we have sufficient information to estimate its spectrum. In previous numerical tests, we know that the condition number of **W** barely change with polynomial degree p and **W** is expected to be close to identity. Thus we give the upper bound of $\kappa(\mathbf{S})$:

$$\kappa(\mathbf{S}) = \|\mathbf{S}\|_2 \left\|\mathbf{S}^{-1}\right\|_2 \tag{4.27a}$$

$$\leq \|\mathbf{W}\|_2 \left\|\mathbf{E}\mathbf{M}^{-1}\mathbf{E}^{\mathbf{T}}\right\|_2 \left\|\mathbf{W}^T\right\|_2 \left\|\mathbf{W}^{-1}\right\|_2 \left\|\left[\mathbf{E}\mathbf{M}^{-1}\mathbf{E}^T\right]^{-1}\right\|_2 \left\|\mathbf{W}^{\mathbf{T}^{-1}}\right\|_2 \tag{4.27b}$$

$$\leq \kappa^2(\mathbf{W}) \cdot \kappa(\mathbf{E}\mathbf{M}^{-1}\mathbf{E}^T) \tag{4.27c}$$

$$\leq \kappa^2(\mathbf{W}) \cdot \kappa^2(\mathbf{E}) \cdot \kappa(\mathbf{M}) \tag{4.27d}$$

$$\sim Cp^4. \tag{4.27e}$$

The inequalities takes the equal sign only if all of the maximum singular values multiplied with each other, and all of the minimum singular values multiplied together. Thus it remains to see if the estimation above for $\kappa(\mathbf{S})$ is sharp. Again, we make a numerical examination and give the coming figure 4.7.

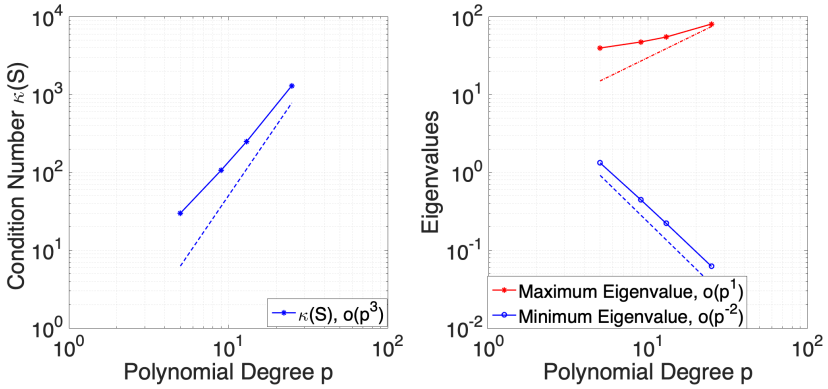

Figure 4.7: Conditioning of $\mathbf{M}_\mathbf{I}$

As we can see that the largest eigenvalue of the Schur complement matrix is tending toward the order of p, which is in agreement with the theoretical analysis before. The lower bound of the eigenvalue decrease with order p^{-2}. However, by theory, it is estimated to have order larger than p^{-3}, which is not sharp. Thus the real condition number of $\mathbf{S}$ scales with Cp^3.

5

Preconditioners for p-version MSEM

In this chapter, we will build preconditioners for a poisson problem on the orthogonal domain $\mathbf{\Omega}$, with only p-refinement. And then, we will test the preconditioners on both orthogonal domain and domains with increasingly distorted mesh.

5.1. Mass Matrix Preconditioner

Consider the mass matrix $\mathbf{M}$ in the form in (4.15), we see a clear diagonal dominance in the matrix. Consider the mass matrix $\mathbf{M_l}$ evaluated with a quadrature order of p, we have matrix $\mathbf{M_l}$ completely diagonal. And $\mathbf{M_e}$ is diagonal dominant according to the previous analysis in §4.2.2. Thus a Jacobi preconditioner is expected to work great.

5.1.1. Jacobi Preconditioner

The Jacobi preconditioner is also known as the diagonal scaling technique. We let the Jacobi preconditioner to be denoted by $\mathbf{M_0}$. We have:

$$\mathbf{M_0} = diag(\mathbf{M}) = \begin{bmatrix} diag(\mathbf{M_e}) \otimes diag(\mathbf{M_l}) & \mathbf{0} \\ \mathbf{0} & diag(\mathbf{M_l}) \otimes diag(\mathbf{M_e}) \end{bmatrix}. \tag{5.1}$$

Here, the $diag(\circ)$ means replacing the matrix $(\circ)$ with only the diagonal part of itself. To illustrate how well the diagonal scaling can tune the spectrum of the original matrix we give the following estimations.

For the mass matrix $\mathbf{M_l}$, since the matrix itself is diagonal while evaluated with quadrature order equal to p at the same quadrature nodes, the diagonal scaling should give an approximate identity matrix in the end, independent from the choice of quadrature order larger then p. The coming theorem 5.1.1 will show that the diagonal part of $\mathbf{M_l}$ has the same condition as the full $\mathbf{M_l}$ matrix.

Theorem 5.1.1. *There exists a constant $C > 0$, such that all of the entries in $diag(\mathbf{M_l})$ fall in the interval $[Cp^{-2}, Cp^{-1}]$.*

Proof. Take the diagonal entries in $\mathbf{M_l}$ out, we have:

$$diag(\mathbf{M_l}) = \int_{x_0}^{x_p} l_i(x) l_j(x) \delta_{ij} dx = \sum_{k=0}^{p} \rho_k l_i^2(x_k). \tag{5.2}$$

Since ρ_k and $l_i^2(x_k)$ are both non-negative, we have:

$$\min\{\rho_k\} \sum_{k=0}^{p} l_i^2(x_k) \le \sum_{k=0}^{p} \rho_k l_i^2(x_k) \le \max\{\rho_k\} \sum_{k=0}^{p} l_i^2(x_k) \tag{5.3a}$$

$$\min\{\rho_k\} \sum_{k=0}^{p} \delta_{ik}^2 \le \sum_{k=0}^{p} \rho_k l_i^2(x_k) \le \max\{\rho_k\} \sum_{k=0}^{p} \delta_{ik}^2 \tag{5.3b}$$

$$Cp^{-2} \le \int_{x_0}^{x_p} l_i(x) l_j(x) \delta_{ij} dx \le Cp^{-1} \tag{5.3c}$$

. □

The theorem above also indicates the sharpness of theorem 4.2.1.

For the mass matrix $\mathbf{M_e}$, we write out the diagonal entries in $\mathbf{M_e}$ using GLL quadrature nodes and weights:

$$\int_{x_0}^{x_p} e_i(x)e_j(x)\delta_{ij}dx = \sum_{k=0}^{p} \rho_k^2 e_i^2(x_k) \cdot \frac{1}{\rho_k}. \tag{5.4}$$

In §4.4, we found out that the matrix made by $\rho_i e_j(x_i)$ is an approximate identity with row sum equal to 1. It is reasonable to expect that the matrix made by $\rho_i^2 e_j^2(x_i)$ is also an approximate identity with row sum of order C independent of the choice of p. Thus we expect the integral $\int_{x_0}^{x_p} e_i(x)e_j(x)\delta_{ij}dx$ to be bounded by the limits of $1/\rho_k$. Thus we expect the diagonal entries of $\mathbf{M_e}$ to be bounded by $[Cp^{-2}, Cp^{-1}]$. To verify if the expectation is true, we extract the diagonal of $\mathbf{M_e}$ and record the maximum and minimum on the diagonal as p raises, we get exactly the same plot as in Fig.4.2.

Since the mass matrix is symmetric, any preconditioner applied to them should be in the split form. Thus we can precondition a $\mathbf{M}$ mass matrix following the given steps:

- Calculate the square root of the diagonal entries of $\mathbf{M}$, then store the results in a diagonal matrix $\mathbf{L_0}$.
- Calculate the preconditioned matrix $\mathbf{L_0 M L_0^T}$.

and we give condition number before and after preconditioning in the table 5.1 below.

Table 5.1: Condition Number of Preconditioned Mass Matrices

p-order	5	9	13	25	51
$\kappa(\mathbf{M})$	33.35	88.39	170.53	578.33	2292.49
$\kappa(\mathbf{L_0 M L_0^T})$	1.79	2.03	2.15	2.29	2.31

Concerning the sparsity pattern, it doesn't change after diagonal scaling. Diagonal scaling can bring down the condition number uniformly to the order of 1.

5.1.2. Pre-inversion of Mass Matrix

In (4.15), a Kronecker structure of mass matrix $\mathbf{M}$ is given. We can make use of this structure and quickly pre-calculate the inversion of the mass matrix. This is highly meaningful for improving the 2-stage algorithm's efficiency. Followed from corollary 4.2.5.1, we can further give another obvious corollary:

Corollary 5.1.1.1. *If a real non-singular square matrix* $\mathbf{M}$ *is formed by Kronecker product of two square matrices* $\mathbf{A}$ *and* $\mathbf{B}$*, then the inverse of* $\mathbf{M}$ *could be calculated by:*

$$\mathbf{M}^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}.$$

Thus in order to compute the inverse of $\mathbf{M}$, we only need to compute $\mathbf{M_e}^{-1}$ and $\mathbf{M_l}^{-1}$. If we inspect the size of the three matrices, the 1D problem mass matrices' size scale with p, but the 2D mass matrix's size scales with p^2. Thus it's expected that as p goes high, the pre-inversion method is very advantageous.

5.1.3. Numerical Experiment

In order to make a comparison for the preconditioning methods discussed above, we set up a linear algebra problem:

$$\mathbf{Mx} = \mathbf{b}, \tag{5.5}$$

where **b** is set to be a vector with all 1 entries. Three methods are used to solve this problem. For the first method, we use a CG solver from LAPACK routine combining the Jacobi preconditioner, and we call this group PCG. For the second method, we use a sparse direct linear equation system solver form LAPACK as well. The third method is pre-inversion as we proposed in §5.1.2, which is directly calculate the inverse of **M** and then multiply the inverse on the equation system. The time and steps (if applicable) used by these three methods are recorded. In addition, an extra reference group is set to solve the problem with CG solver without preconditioning. The iteration stop tolerance is set to be at $1e$-17, which is also verified with the results from direct solver or preinversion. We set the polynomial order to vary from 5 to 91. The following table shows the results that is averaged from 3 times' repetition:

Table 5.2: Solution Efficiency Evaluation

p-order	5	9	13	25	51	81	91
M size	60^2	180^2	364^2	1300^2	5304^2	13284^2	16744^2
Jacobi PCG	0.000419 s 4 steps	0.000588 s 5 steps	0.00196 s 7 steps	0.03058 s 13 steps	1.0806 s 20 steps	6.60 s 22 steps	10.4 s 22 steps
CG	0.000760 s 15 steps	0.00246 s 33 steps	0.0124 s 71 steps	0.472 s 214 steps	29.0 s 731 steps	/ /	/ /
Direct Solver	0.000408 s	0.00163 s	0.0101 s	0.296 s	18.0 s	/	/
Pre-inversion	0.0219 s	0.0497 s	0.0768 s	0.219 s	0.991 s	4.34 s	6.20 s

In the experiment, we use a machine with 16GB RAM, and we eliminate all other programs' occupation on the machine CPU. After experiment, several observations are made:

- The direct sparse linear system solver is powerful and fast for small problems (In our case, "small" means a system with unknowns no more than order of 10^4). But its resource requirements grows fast as problem size increase. Direct solver breaks down before the smallest eigenvalue **M** reaches the smallest machine limit, thus preconditioning for direct solvers are not necessary in most of the cases.

- The problem size of the pre-inversion method scales with p instead of p^2, thus at high polynomial order, this method is becoming more and more advantageous. This is clearly shown by a semi-logarithm plot in Fig.5.1. One can see that for a sparse diagonal dominant system, direct solver is less preferable than a preconditioned iterative solver as the problem size grows.

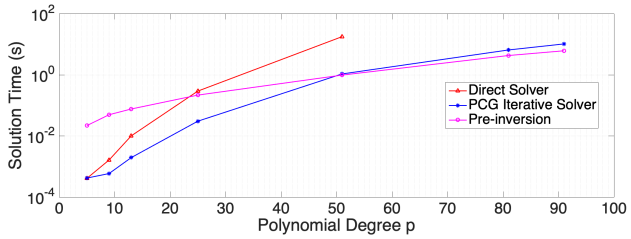

Figure 5.1: Solution Time Compare

- The efficiency of conjugate gradient solver is highly dependent on the conditioning. However, when an effective preconditioner is applied, the resulted PCG solver is stable and efficient. This is clear in Fig.5.2.

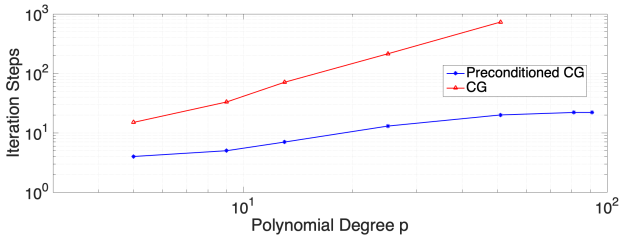

Figure 5.2: Iteration Step Compare With and Without Preconditioning

5.2. Schur Complement Matrix Preconditioner

From the literature review, it is commonly used in 2-stage solvers to compute preconditioners for Schur complement from the Cholesky factorisation of mass matrices. In previous 2-stage algorithms, there is an inverse of mass matrix in the Schur complement, which is unknown thus is usually replaced by an approximate inverse of mass matrix. However, in our case, the inversion of mass matrix is pre-calculated with predictable error and time.

With Cholesky factorisation, we can write the inversion of 1D problem mass matrices in the following form:

$$\mathbf{M_l}^{-1} = \mathbf{L_l L_l^T}, \tag{5.6a}$$

$$\mathbf{M_e}^{-1} = \mathbf{L_e L_e^T}, \tag{5.6b}$$

where $\mathbf{L}$ matrices are left lower triangular matrix. The 2D problem mass matrix can be given by:

$$\mathbf{M}^{-1} = \begin{bmatrix} (\mathbf{L_l} \otimes \mathbf{L_e})(\mathbf{L_l^T} \otimes \mathbf{L_e^T}) & \mathbf{0} \\ \mathbf{0} & (\mathbf{L_l} \otimes \mathbf{L_e})(\mathbf{L_l^T} \otimes \mathbf{L_e^T}) \end{bmatrix} = \mathbf{L L^T}. \tag{5.7}$$

Thus the Schur complement matrix could be written by:

$$\mathbf{S} = \mathbf{W}^T \mathbf{E L L^T E}^T \mathbf{W}. \tag{5.8}$$

To solve system:

$$\mathbf{S}\boldsymbol{\phi} = \mathbf{W^T E M^{-1} B_w} \boldsymbol{\phi}_{\mathbf{tr}} - \mathbf{W}^T f, \tag{5.9}$$

we set up an experiment as in §5.1.3. But this time, we use different techniques to precondition the Schur complement.

First of all, we see $\mathbf{W}$ as an approximate identity, thus we only need to find an approximate inverse of $\mathbf{E L L^T E}^T$. Secondly, we know that the matrix $\mathbf{E E}^T$ is a sparse matrix, and the Cholesky factorisation of this matrix is a lower triangular matrix with diagonal offsets clustered near the main diagonal. To show this is true, we give the sparsity pattern plot of $\mathbf{E M_0^{-1} E}^T$ and its Cholesky factor matrix $Chol(\mathbf{E M_0^{-1} E}^T)$ in Fig. 5.3, for $p = 5$. $\mathbf{M_0}$ is the diagonal part of $\mathbf{M}$.

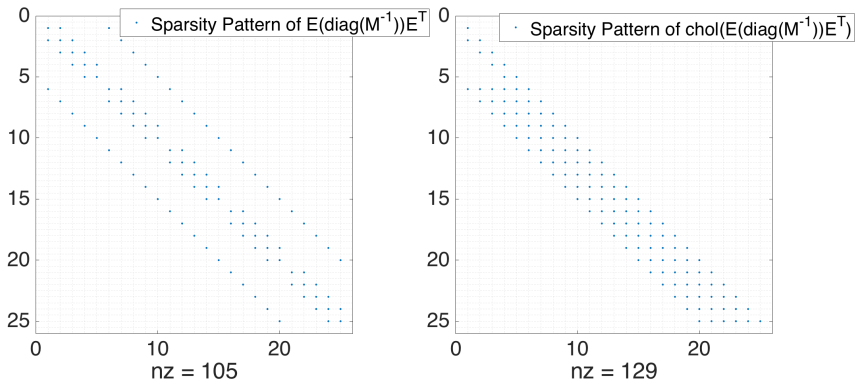

Figure 5.3: Sparsity Pattern of $\mathbf{E L_0 L_0^T E^T}$ and its Cholesky Factor

Moreover, since the Jacobi preconditioner of $\mathbf{M}^{(-1)}$ worked well in the previous section, we use the Jacobi preconditioner $\mathbf{L_0 L_0^T}$ to substitute $\mathbf{L L^T}$. Notice that $\mathbf{L_0 L_0^T}$ is diagonal thus the preconditioner $\mathbf{E L_0 L_0^T E^T}$ will have the same sparsity pattern as $\mathbf{E E^T}$, and the Cholesky factorisation of $\mathbf{E L_0 L_0^T E^T}$ will be of the same pattern as $chol(\mathbf{E E^T})$. This indicates that taking the inverse of $chol(\mathbf{E L_0 L_0^T E^T})$ will be fast under sparse inverse algorithms.

To construct the preconditioner of $\mathbf{S}$, we follow the steps below:

- Compute the inversion of the mass matrix use the method proposed in §5.1.2. Find out $\mathbf{L_0}$.
- Compute the inverse of Cholesky factor of $\mathbf{E L_0 L_0^T E^T}$, denoted by $\mathbf{U_0}$, i.e.,

$$\mathbf{U_0 E L_0 L_0^T E^T U_0}^T = Id. \tag{5.10}$$

- Transfer the original equation system in (5.9) about $\boldsymbol{\phi}$ into the following one:

$$\left[\mathbf{U_0 S U_0^T}\right] \mathbf{y} = \mathbf{U_0}(\mathbf{W^T E M^{-1} B_w} \boldsymbol{\phi}_{\mathbf{tr}} - \mathbf{W^T} f), \tag{5.11a}$$

$$\boldsymbol{\phi} = \mathbf{U_0^T y}. \tag{5.11b}$$

5.2.1. Numerical Experiment

In this numerical test, we solve the Poisson problem in (3.1) with:

$$f(x,y) = -9\pi^2(x^2+y^2)cos(3\pi xy), \qquad (x,y) \in \boldsymbol{\Omega}.$$

The analytic solution of this problem is known as:

$$\phi(x,y) = cos(3\pi xy), \qquad (x,y) \in \boldsymbol{\Omega}.$$

The analytic solution provides a reference for error evaluation on the $\mathcal{H}^1(\boldsymbol{\Omega})$ functional space.

We first compute the Schur complement matrix $\mathbf{S}$, then compute the preconditioner $\mathbf{U_0}$ for $\mathbf{S}$. We solve the problem with polynomial degree p vary from 5 to 61. To show how well preconditioner U_0 could perform, the following Table 5.3 compares the condition number of $\mathbf{S}$ and $\mathbf{U_0 S U_0^T}$. Notice that the exact value of $\kappa(\mathbf{S})$ differed from what is in Table 4.1, which is due to a different quadrature order choice. In this case, the quadrature order is chosen to be p. It should also be pointed out that although the exact value of condition number changes under different quadrature order, the order of condition number growth is fixed and bounded by the eigenvalue estimations in §4.5.

Table 5.3: Condition Number of Preconditioned Schur Complement Matrix

p-order	5	9	13	25	51	61
$\kappa(\mathbf{S})$	13.96	57.98	159.23	1046.78	8563.75	14574
$\kappa(\mathbf{U_0 S U_0^T})$	2.19	2.45	2.57	2.70	2.78	2.79

Now, we set up 2 groups of tests using different methods to solve the equation system for $\boldsymbol{\phi}$. The first group, we use a direct solver to solve system $\mathbf{S}\boldsymbol{\phi} = \mathbf{W^T E M^{-1} B_w}\boldsymbol{\phi}_{\mathbf{tr}} - \mathbf{W^T} f$, without any extra operation. The second group, we first compute preconditioner $\mathbf{U_0}$, then apply a CG solver to solve equation system (3.8) under convergence residual $1e$-17, and in the end compute $\boldsymbol{\phi}$ use (5.11b). In the test, we record the time consumption, $\mathcal{L}^2$ error, and convergence steps if available. Notice that the time consumption for group is recorded by two time intervals, the first interval measures time consumed to construct the preconditioner, and the second interval measures the time used for the CG solver to solve and post process the final result of $\boldsymbol{\phi}$. The results are given in Table 5.4.

Table 5.4: Solution Methods Comparison for Schur Complement System

p-order	5	9	13	25	51	61
S size	25^2	81^2	169^2	625^2	2601^2	3721^2
Direct Solver Time	0.000235 s	0.000395 s	0.00163 s	0.00427 s	2.217 s	6.450 s
$\mathbf{U_0}$ Build Time	0.00126 s	0.00294 s	0.00606 s	0.0895 s	1.918 s	4.37s
CG Solver Time	0.000687 s	0.00147 s	0.00567 s	0.0968 s	4.121 s	11.088 s
Step	6 steps	15 steps	23 steps	27 steps	28 steps	28 steps
$\mathcal{L}^2$ Error	1.777	0.261	0.007280	$2.517e-10$	$1.972e-13$	$4.000e-13$

To discuss the results, we refer to the figures given below. In Fig. 5.4, we compare the solution time consumed by direct and iterative CG solver. It's clear that both direct solver and iterative solver's time consumption scale with problem size at the same rate. This is due to the small size of the Schur complement system, which is highly reduced from the original mixed **LHS** system. In this experiment, the **S** matrix column (or row) size did not exceed 10^5 with no nonzero entries, thus in this case, the direct solver is more competitive. But as the dimension of the matrix grow to the scale no less than 10^6, we expect the preconditioned CG iterative methods to perform well, as long as the preconditioned system has a uniformly bounded condition number, just as shown by Table 5.3.

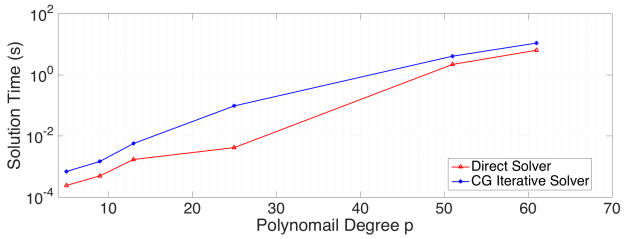

Figure 5.4: Solver Time Comparison

We had a stable preconditioner for the Schur complement matrix, and we have to consider the price of obtaining such preconditioners. Fig.5.5 shows that the time consumed to obtain preconditioner $\mathbf{U_0}$ is growing slower than the iterative solver as p increase. Recall from Fig.5.3, the number of nonzero entries in $\mathbf{EL_0L_0^TE^T}$ is predictable, thus the cost of FLOPs required for the preconditioner is also known. Notice that in order to compute a Schur complement matrix of size $p^2 \times p^2$, the full left-hand-side matrix is of size $(3p^2 + 2p) \times (3p^2 + 2p)$. Due to the limit of machine memory, here we can only predict that the preconditioned system will perform well when problem size grow large enough.

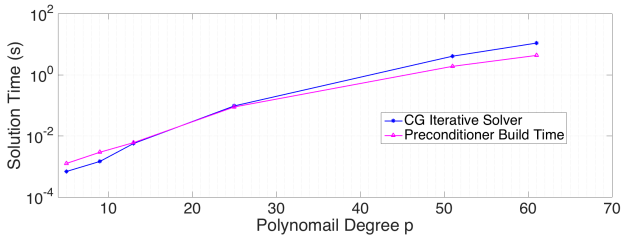

Figure 5.5: Preconditioner Construction Time

Finally, we give an observation on the error analysis. The $\mathcal{L}^2$ error is computed by the vector norm $\|\phi(x_i, y_j) - \phi_{i,j}\|_2 = \sqrt{\sum_{i,j} \{\phi(x_i, y_j) - \phi_{i,j}\}^2}$, with (x_i, y_j) be the Legendre-Gauss-Lobatto points. The results give the same error by the two methods, and is shown in Fig.5.6.

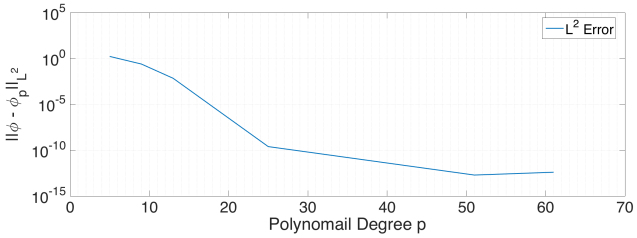

Figure 5.6: $\mathcal{L}^2$ Error Quantification

Notice that the error goes down to $1e{-}13$ and stopped decreasing, although the quadrature order chosen is sufficiently high. This is due to the machine roundoff error in each sampling point summed up.

5.3. Two-stage Solver

In this section, we will combine the preconditioners discussed before to build a 2-stage solver similar to the Uzawa's algorithm. First, we give a flow chart of the program in Fig.5.7.

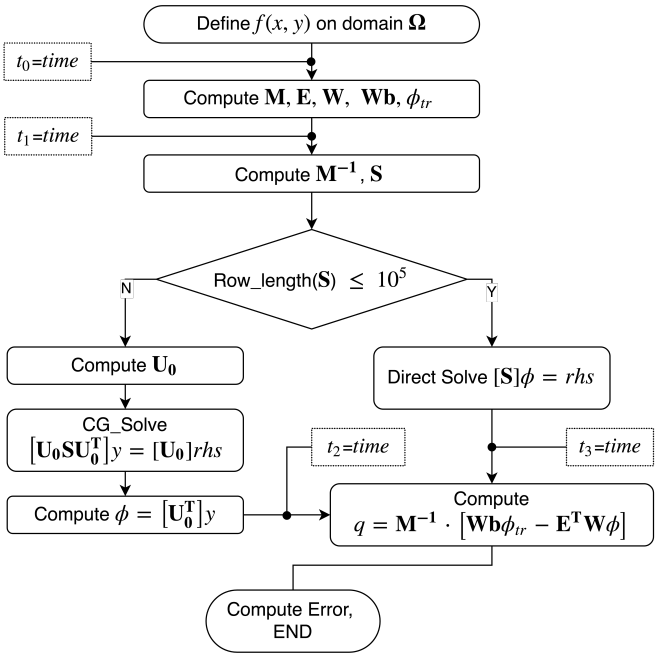

Figure 5.7: Two-stage Algorithm Flowchart

A few comments need to be made to the flow diagram above. First, the timing point t_1 is just before the mass matrix inversion, because the mass matrix inversion is crucial and considerably costly in the two-stage algorithm. Secondly, the criterion for choosing iterative or direct solver is given by the number of unknowns in the system. The number 10^5 is referred to the condition of the machine. In our case, one can refer to §5.1.3 and see that 10^5 is the point where iterative solver and direct solver have the same efficiency level. Lastly, in this experiment, we can only afford a system with row length of $\mathbf{S}$ smaller than 10^5, thus the time consumption is counted by $t_3 - t_1$. We compare this method with a direct solver solving system in (3.8). The results given in Table 5.5 below are averaged from 3 times' experiment repetition. We also record the time cost to make the mesh and assemble the matrices on the LHS.

Table 5.5: Solution Efficiency Comparison

p-order	5	9	13	25	51	61
Preprocess ($t_1 - t_0$)	0.572 s	1.107 s	1.817 s	5.078 s	22.879 s	37.971 s
2-Stage Solver ($t_3 - t_1$)	0.0224 s	0.0502 s	0.0786 s	0.229 s	3.254 s	8.115 s
Direct Solve **LHS**	0.00136 s	0.00544 s	0.0328 s	1.0221 s	65.180 s	/

As we can see that the 2-stage algorithm significantly increased the efficiency of solving the Poisson problem. But the weak spot of this method is that it relies on the efficiency of inverting the mass matrix. However, the mass matrix inversion is not always cheap and fast, because mass matrix is metrics dependent. Also, when system size grow larger and larger, the matrix multiplications will take a growing portion of time compared to the time spent on the whole solution process.

5.4. Constraint Preconditioner

Inspired by Wathen's idea on constraint preconditioner introduced in Chapter 2, we seek the possibility of solving the large **LHS** system directly with a preconditioned iterative solver. In this section, we will try to find such a preconditioner for the **LHS** matrix **A**.

In order to find an approximate **LHS**, denoted by $\mathbf{A_0}$, we first use block diagonalisation process discussed in (2.6) for the saddle point problem in (3.8), we obtain:

$$\mathbf{A} = \begin{bmatrix} \mathbf{M} & \mathbf{E^T W} \\ \mathbf{W^T E} & \mathbf{O} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{W^T E M^{-1}} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{O} \\ \mathbf{O} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{M^{-1} E^T W} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}. \tag{5.12}$$

In §4.4, we pointed out that **W** is an approximate identity but full and clumsy. Thus **W** in **A** is discarded. We write out the new $\mathbf{A_0}$:

$$\mathbf{A_0} = \begin{bmatrix} \mathbf{I} & \mathbf{O} \\ \mathbf{E M^{-1}} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{M} & \mathbf{O} \\ \mathbf{O} & -\mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{M^{-1} E^T} \\ \mathbf{O} & \mathbf{I} \end{bmatrix}. \tag{5.13}$$

Now, the matrices **M** and **S** are exactly the two matrices solved in the 2-stage algorithm. The preconditioners are available for them both, which means that we can implicitly absorb the preconditioners of the two small blocks into the big system! Thus we replace **M** with its cholesky factorisation $\mathbf{P_0 P_0^T}$, and replace **S** with cholesky factorisation $\mathbf{Q_0 Q_0^T}$, then we've found out a triangular indefinite factorisation of $\mathbf{A_0}$:

$$\mathbf{A_0} = \begin{bmatrix} \mathbf{P_0} & \mathbf{O} \\ \mathbf{E M^{-1} P_0} & -\mathbf{Q_0} \end{bmatrix} \begin{bmatrix} \mathbf{P_0^T} & \mathbf{P_0^T M^{-1} E^T} \\ \mathbf{O} & \mathbf{Q_0^T} \end{bmatrix} = \mathbf{A_L A_R}. \tag{5.14}$$

Notice that $\mathbf{A_L}$ and $\mathbf{A_R}$ are not each other's transpose. Also, we need to point out that $\mathbf{A_0}$ is an indefinite matrix. After block triangularisation, negative eigenvalues are concentrated by $-\mathbf{S}$. To form a preconditioner for the **LHS** matrix, we can either reformulate the problem into a positive definite one by applying asymmetric split type of preconditioners (just like $\mathbf{A_L}$ and $\mathbf{A_R}$) or use an SPD preconditioner. In this case, we chose to use a splitted SPD preconditioner to cope with the MINRES solver. Thus the positive factor $\mathbf{A_R}$ is used.

We denote the constraint preconditioner with $\mathbf{B_0}$, and the preconditioned system takes the form of $\mathbf{B_0 A B_0^T}$, where

$$\mathbf{B_0} = \begin{bmatrix} \mathbf{P_0^{-1}} & \mathbf{O} \\ -\mathbf{Q_0^{-1} E M^{-1}} & \mathbf{Q_0^{-1}} \end{bmatrix}. \tag{5.15}$$

Considering the price of obtaining $\mathbf{B_0}$ as a preconditioner, we give a flow diagram in 5.8 as a demonstration.

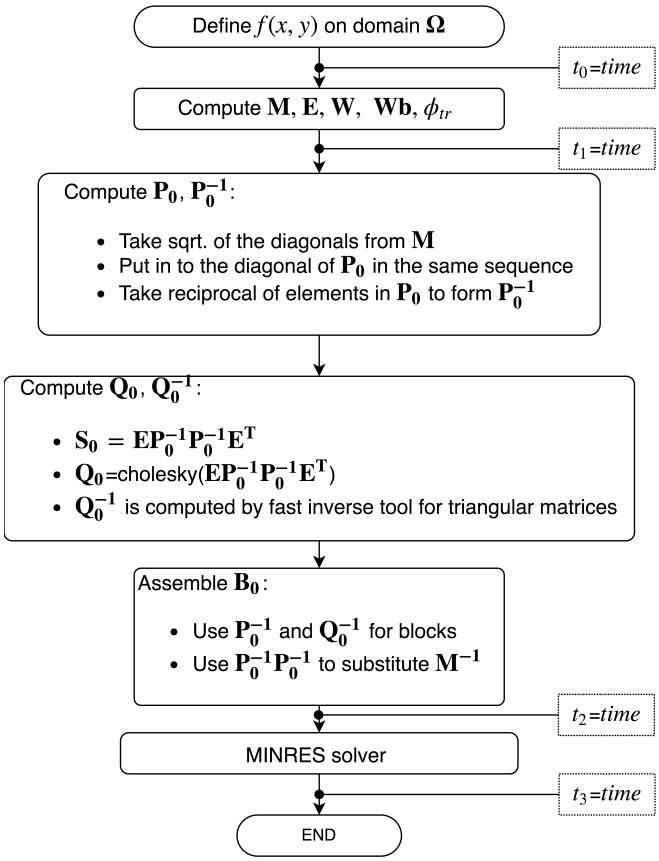

Figure 5.8: Flow Diagram for Constraint Preconditioner

In this method, the exact inverse of the mass matrix and the exact **S** are not calculated. However, the information of the spectrum are carried by the preconditioners.

5.4.1. Results for Problems on Orthogonal Domain

To see how this method performs, we follow the flow diagram in Fig.5.8 and repeat the same numerical test as in §5.3. The following table 5.6 will show the condition number change before and after preconditioning, iteration steps and time spent until convergence, and the $\mathcal{L}^2$ error.

Table 5.6: Solution Results with Constraint Preconditioner on Orthogonal Domain

p-order	5	9	13	25	51	61
LHS size	85^2	261^2	533^2	1925^2	7905^2	11285^2
κ(**LHS**)	18.30	26.53	52.73	258.48	1889.10	3176.47
$\kappa(\mathbf{B_0AB_0^T})$	2.32	2.32	2.75	2.90	2.98	2.99
Preprocess $(t_1 - t_0)$	0.561 s	1.110 s	1.805 s	5.019 s	22.801 s	37.127 s
Precondition $(t_2 - t_1)$	0.00273 s	0.00531 s	0.0194 s	0.680 s	40.218 s	129.779 s
MINRES Solver Time	0.00121 s	0.00329 s	0.00973 s	0.109 s	2.041 s	4.494s
Step	14 steps	28 steps	39 steps	46 steps	48 steps	48 steps
$\mathcal{L}^2$ Error	1.777	0.261	0.007280	$2.517e{-10}$	$1.488e{-13}$	$1.900e{-13}$

The constraint preconditioner $\mathbf{B_0}$ will fill the $(2, 2)$ block of the original system and increase the number of nonzero entries, but $\mathbf{B_0}$ itself is quite sparse. In Fig.5.9, we show the constraint preconditioner's sparsity pattern and its block structure same as in (5.15), taken at $p = 5$. In Fig.5.10, we show the sparsity pattern of the system after preconditioning.

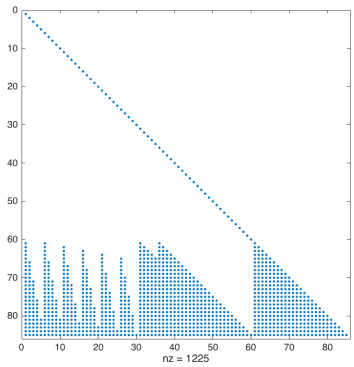

Figure 5.9: Sparsity Pattern of Constraint Preconditioner

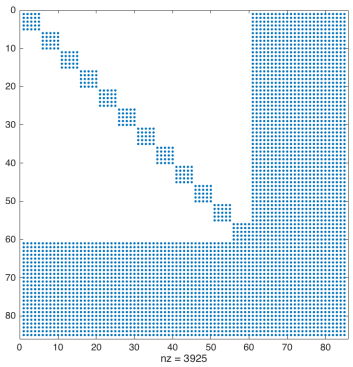

Figure 5.10: Sparsity Pattern after Preconditioning

The preconditioned system is well-conditioned. Besides the condition number in the chart, we plot the spectrum of the **LHS** system before and after preconditioning in Fig.5.11, taken at $p = 25$.

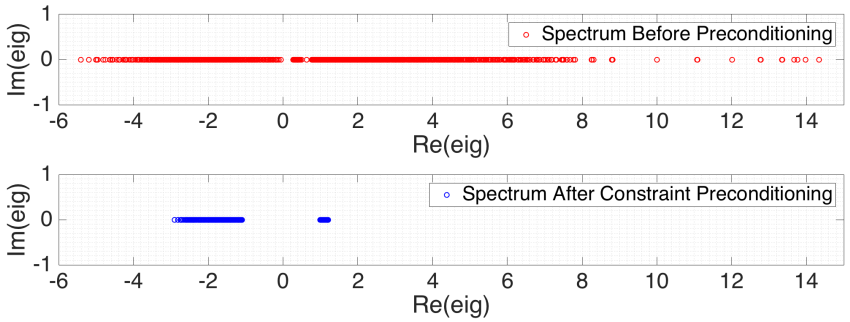

Figure 5.11: Spectrum Compare before and after Preconditioning

5.4.2. Results for Problems on Mapped Domain

Now, we use the same strategy and the same preconditioner on a mapped domain $\tilde{\mathbf{\Omega}} \ni (\xi, \eta)$ with increasingly distorted mesh controlled by the distortion factor c. The mapping is defined as:

$$\eta(x, y) = x + c \cdot sin(\pi x) sin(\pi y),$$
$$\xi(x, y) = y + c \cdot sin(\pi x) sin(\pi y),$$

where $\xi, \eta \in [-1, 1]$. We will take this test for $c = 0.05,\ 0.1,\ 0.15,\ 0.20$, and we show the distorted domain at $c = 0.05,\ 0.20$ in the following Fig.5.12 and Fig.5.13.

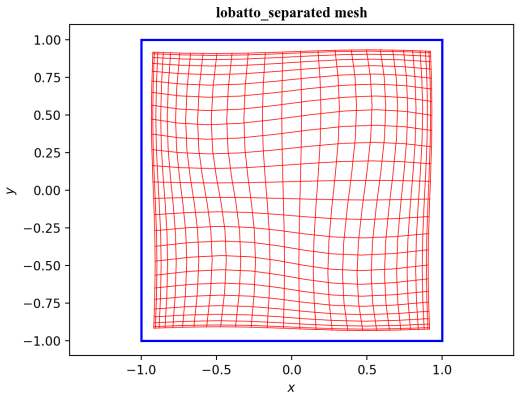

Figure 5.12: Mapped Domain with $p = 25$, $c = 0.05$

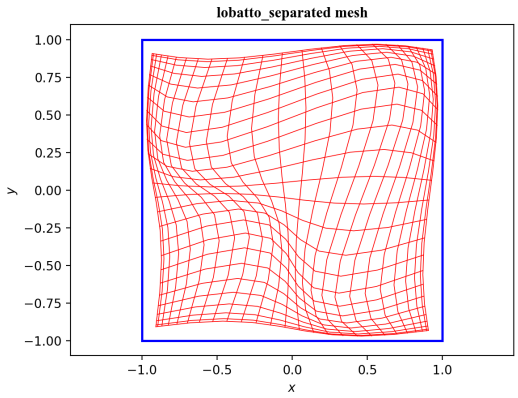

Figure 5.13: Mapped Domain with $p = 25$, $c = 0.20$

The p-order numbers we choose to test are 5, 9, 13 and 25. The following tables will present the results.

Table 5.7: Constraint Preconditioner on Non-orthogonal Domain, $p = 5$

c	0.05	0.10	0.15	0.20
LHS size	85^2	85^2	85^2	85^2
$\kappa(\mathbf{LHS})$	18.87	19.82	21.38	24.20
$\kappa(\mathbf{B_0 A B_0^T})$	2.72	3.23	3.90	5.25
Preprocess $(t_1 - t_0)$	0.561 s	0.561 s	0.561 s	0.561 s
MINRES Solver Time Step	0.00260 s 33 steps	0.00320 s 41 steps	0.00372 s 48 steps	0.00489 s 57 steps
Direct Solver Time	0.000540 s	0.000394 s	0.000464 s	0.000401 s
$\mathcal{L}^2$ Error	1.798	1.848	1.895	1.909

Table 5.8: Constraint Preconditioner on Non-orthogonal Domain, $p = 9$

c	0.05	0.10	0.15	0.20
LHS size	261^2	261^2	261^2	261^2
$\kappa(\mathbf{LHS})$	27.08	28.99	34.64	47.84
$\kappa(\mathbf{B_0 \tilde{A} B_0^T})$	3.21	4.03	5.18	9.91
Preprocess $(t_1 - t_0)$	1.110 s	1.110 s	1.110 s	1.110 s
MINRES Solver Time Step	0.00743 s 54 steps	0.0106 s 71 steps	0.0129 s 95 steps	0.0175 s 132 steps
Direct Solver Time	0.00466 s	0.00357 s	0.00574 s	0.00404 s
$\mathcal{L}^2$ Error	0.280	0.332	0.395	0.444

Table 5.9: Constraint Preconditioner on Non-orthogonal Domain, $p = 13$

c	0.05	0.10	0.15	0.20
LHS size	533^2	533^2	533^2	533^2
$\kappa(\mathbf{LHS})$	55.54	63.40	77.60	105.77
$\kappa(\mathbf{B_0 \tilde{A} B_0^T})$	3.40	4.36	5.77	10.43
Preprocess $(t_1 - t_0)$	1.805 s	1.805 s	1.805 s	1.805 s
MINRES Solver Time Step	0.0189 s 59 steps	0.0263 s 83 steps	0.0395 s 118 steps	0.0542 s 177 steps
Direct Solver Time	0.0248 s	0.0248 s	0.0242 s	0.0248 s
$\mathcal{L}^2$ Error	0.0174	0.0457	0.0957	0.167

Table 5.10: Constraint Preconditioner on Non-orthogonal Domain, $p = 25$

c	0.05	0.10	0.15	0.20
LHS size	1925^2	1925^2	1925^2	1925^2
κ(**LHS**)	280.13	334.21	431.79	611.57
$\kappa(\mathbf{B_0}\tilde{\mathbf{A}}\mathbf{B_0^T})$	3.69	4.99	7.01	12.83
Preprocess $(t_1 - t_0)$	5.019 s	5.019 s	5.019 s	5.019 s
MINRES Solver Time	0.197 s	0.320 s	0.479 s	0.983 s
Step	65 steps	96 steps	147 steps	253 steps
Direct Solver Time	1.0285 s	1.036 s	0.906 s	1.030 s
$\mathcal{L}^2$ Error	$1.95e{-6}$	$3.47e{-5}$	0.000229	0.000894

In the previous results, we make the following observations:

- At any fixed distortion parameter c, the preconditioner bound the condition number at a fixed level, independent from p. To make this point, we take the worst case when $c = 0.20$, and we plot the condition number change before and after preconditioning with growing polynomial order in Fig. 5.14 on the left.
- At a fixed p-order, the more distorted the mesh is, the higher the condition number grows, as well as the solution steps. What we also observe is that the preconditioner built according to an orthogonal mesh reduces the condition number due to p-refinements while preserving the portion of contribution of mesh distortion on the condition number. To make this point, see Fig.5.14 on the right, the two curves raise with increasing c almost parallel.
- The error grows fast as c increase.

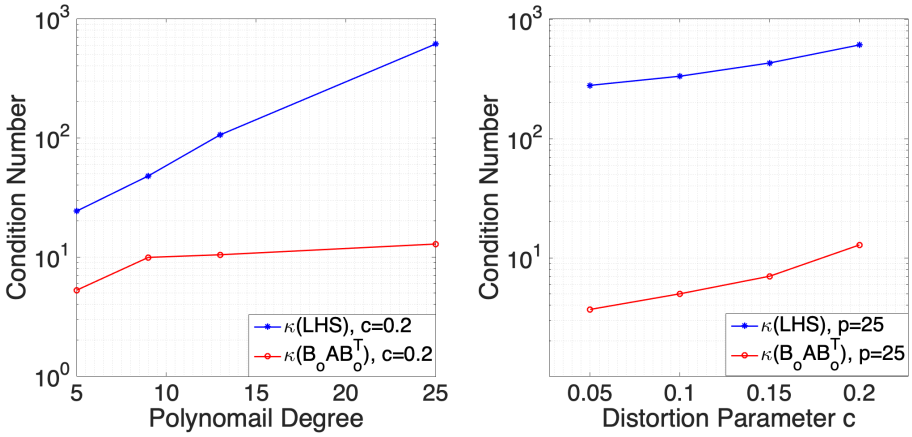

Figure 5.14: Condition Number Change before and after Preconditioning

5.5. Discussion

In this section, we make a rough comparison on a bigger picture to expose the pros and cons of the two preconditioning methods used above. The discussion will focus on system size, arithmetic complexity and solution time/steps.

By referring to the LAPACK working notes [1], the Floating Point Operation Counts (FLOPs) are known for critical steps encountered while finding the solution. For a square matrix of row(column) size n, Cholesky factorisation costs about $O(\frac{1}{3}n^3)$ FLOPs, inverting a full matrix cost about $O(n^3)$ FLOPs, and inverting a triangular matrix costs $O(n^2)$ FLOPs. It takes $O(n)$ FLOPs in each iteration step for both CG and MINRES solvers. Knowing these information, we can give a brief estimation of how the FLOPs grow with the method order p, as in Table 5.11.

In Table 5.11, we can see that as problem size grow, preconditioned iterative solver will become the only choice. The 2-stage solver is advantageous when p order is not high enough, and the machine memory is on a tight budget. The large system is dissected into two smaller systems thus give results fast when p is not large enough. However, this method is highly dependent on the availability of the inverse of mass matrix. If the mass matrix inverse is not cheap to obtain, for instance, when the mass matrix is evaluated on an increasingly distorted mesh, the advantages may vanish quickly. Constraint preconditioners works on the large system, which is not sparse, thus very demanding on the machine memory. However, building a constraint preconditioner does not require the exact inverse of the mass matrix, or the Schur complement, thus gives this method more freedom on the problem type. Also, as problem size grow, a powerful computer is expected, constraint preconditioning technique will be much more advantageous in solving large problems.

Table 5.11: Arithmetic Complexity Analysis

Method	2-stage System				Constraint Preconditioner	
	1st stage		2nd stage			
Solver Type	Inversion	Pre-inversion	Iterative	Direct Solve	Iterative	Direct
System Size	$(2p^2+2p) \times (2p^2+2p)$	$p \times p$	$p^2 \times p^2$		$(3p^2+2p) \times (3p^2+2p)$	
FLOPs Order	$O(p^6)$	$O(p^3)$	$O(p^2)$	$O(p^6)$	$O(p^2)$	$O(p^6)$
Portion of Nonzero Entries	$\rightarrow O(p^{-1})$	0	0		$\rightarrow O(1)$	
Iteration Steps	None	None	Independent from p	None	Independent from p	None
Preconditioner Construction	None		$O(p^4)$		$O(p^4)$	

6

Preconditioned System with Hybrid *hp*-refinements on Partitioned Domain

The previous chapter introduced the preconditioning techniques for p-version MSEM. In this chapter, we will seek the ways to precondition hp-refined MSEM on a partitioned orthogonal domain using Lagrange multipliers. This part of the work is supported by the derivation in §3.4.

6.1. Block Structure

First of all, we look into the block structure of the big square matrix in (3.17):

$$\mathbf{K} = \left[\begin{array}{ccc|c} \mathbf{M} & \mathbf{E}^{\mathbf{T}}\mathbf{W} & \mathbf{Wb} & \mathbf{G} & \mathbf{L}^{\mathbf{T}} \\ \mathbf{W}^{\mathbf{T}}\mathbf{E} & \mathbf{O} & \mathbf{O} & & \\ \hline \mathbf{Wb}^{\mathbf{T}} & \mathbf{O} & \mathbf{O} & \mathbf{L} & \mathbf{O} \end{array}\right],$$

where $\mathbf{L} = \begin{bmatrix}\mathbf{Wb}^{\mathbf{T}} & \mathbf{O}\end{bmatrix}$, and $\mathbf{G} = \begin{bmatrix}\mathbf{M} & \mathbf{E}^{\mathbf{T}}\mathbf{W} \\ \mathbf{W}^{\mathbf{T}}\mathbf{E} & \mathbf{O}\end{bmatrix}$. We can show the sparsity pattern of the $\mathbf{K}$ matrix in Fig.6.1 with respect to the mesh plotted in Fig.3.1 to further reveal the blocking of $\mathbf{K}$.

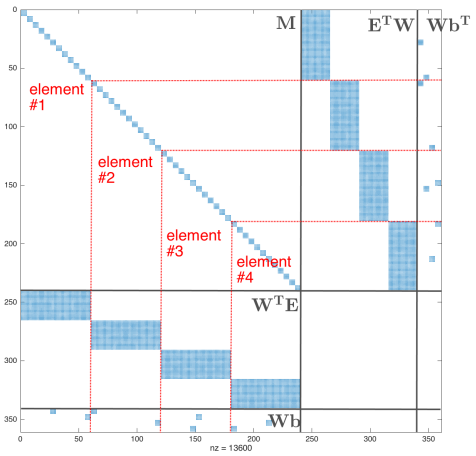

Figure 6.1: Blocking of $\mathbf{K}$

In Fig.6.1, the grey lines indicate the 3×3 block structure as in (3.17), and the red lines indicate that $\mathbf{G}$ is exactly assembled independently from algebraic equations of four subproblems's, just as $\mathbf{A}$ in (3.8).

6.2. Steklov-Poincaré Operator

One of the highlights of domain decomposition techniques is to reduce a global problem to several independent subproblems, thus significantly reduce the problem size and as well provide more freedom

in choice of governing equation, discretisation scheme, resolution order, etc.. In our case, this is done by constructing a Steklov-Poincaré operator, which is the Schur complement of $\mathbf{G}$ in the large system $\mathbf{K}$. In our case, the Steklov-Poincaré operator maps the unknown boundary values ϕ_{in} on $\boldsymbol{\Omega \backslash \Gamma_D}$ to the known boundary values ϕ_{tr} on $\boldsymbol{\Omega \cap \Gamma_D}$. If we denote the Steklov-poincaré operator by $-\boldsymbol{\mathcal{S}}$, then we have:

$$\boldsymbol{\mathcal{S}} = \mathbf{L}\mathbf{G}^{-1}\mathbf{L}^{\mathbf{T}}, \tag{6.1}$$

and the reduced system is:

$$\boldsymbol{\mathcal{S}}\boldsymbol{\phi}_{\mathbf{in}} = \mathbf{L}\mathbf{G}^{-1}\begin{bmatrix}\mathbf{W}\mathbf{b}\boldsymbol{\phi}_{\mathbf{tr}} \\ \mathbf{W}^{\mathbf{T}}\mathbf{f}\end{bmatrix}. \tag{6.2}$$

To see how much domain decomposition can reduced the system, we take the same square mesh topology as in Fig.3.1, denote the number of elements per row by n, and denote the polynomial order in each element by p. The original system has DoF amounts to $n^2(3p^2+4p)-2np$, and the reduced system will only have $2p(n^2-n)$ DoF. By looking at the ratio of the DoF numbers:

$$\frac{2p(n^2-n)}{n^2(3p^2+4p)-2np} = \frac{2n-2}{n(3p+4)-2}, \qquad (n>1, p\geq 1),$$

we can see that the higher the p-order is chosen, the more advantageous domain decomposition will become. And for an increasingly h-refined problem, the system reduction ratio will tend to a stable number determined by the polynomial order p.

Knowing the amount of improvement of this technique, now we consider the price of calculating $\boldsymbol{\mathcal{S}}$, or, the price of computing $\mathbf{G}^{-1}$.

6.2.1. Computing $\mathbf{G}^{-1}$

The block structure of $\mathbf{G}$ is shown in Fig.6.1. We see that $\mathbf{G}$ has a clear pattern:

$$\mathbf{G} = \begin{bmatrix}\mathbf{I_4}\otimes\mathbf{M}_{local} & \mathbf{I_4}\otimes\mathbf{E}^{\mathbf{T}}{}_{local}\mathbf{W}_{local} \\ \mathbf{I_4}\otimes\mathbf{W}^{\mathbf{T}}{}_{local}\mathbf{E}_{local} & \mathbf{I_4}\otimes\mathbf{O}\end{bmatrix}, \tag{6.3}$$

where $\mathbf{I_4}$ is a 4×4 identity matrix, and matrices $\mathbf{M}_{local}$, $\mathbf{E}_{local}$ and $\mathbf{W}_{local}$ are local matrices from p-version MSEM, just as matrices in (3.8). There exists a permutation matrix denoted by $\boldsymbol{\mathcal{P}}$ such that:

$$\boldsymbol{\mathcal{P}}\mathbf{G}\boldsymbol{\mathcal{P}}^{T} = \mathbf{I_4}\otimes\begin{bmatrix}\mathbf{M}_{local} & \mathbf{E}^{\mathbf{T}}{}_{local}\mathbf{W}_{local} \\ \mathbf{W}^{\mathbf{T}}{}_{local}\mathbf{E}_{local} & \mathbf{O}\end{bmatrix} = \mathbf{I_4}\otimes\mathbf{A}. \tag{6.4}$$

Matrix $\mathbf{A}$ is exactly the same as in §5.4. Thus in order to compute $\mathbf{G}^{-1}$, we only need to compute $\mathbf{A}^{-1}$.

Again, recall (5.12), we can derive an indefinite triangular factorisation of $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix}\mathbf{I} & \mathbf{O} \\ \mathbf{W}^{\mathbf{T}}\mathbf{E}\mathbf{M}^{-1} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{M} & \mathbf{O} \\ \mathbf{O} & -\mathbf{S}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{M}^{-1}\mathbf{E}^{\mathbf{T}}\mathbf{W} \\ \mathbf{O} & \mathbf{I}\end{bmatrix} \tag{6.5a}$$

$$= \begin{bmatrix}\mathbf{P} & \mathbf{O} \\ \mathbf{W}^{\mathbf{T}}\mathbf{E}\mathbf{M}^{-1}\mathbf{P} & -\mathbf{Q}\end{bmatrix}\begin{bmatrix}\mathbf{P}^{\mathbf{T}} & \mathbf{P}^{\mathbf{T}}\mathbf{M}^{-1}\mathbf{E}^{\mathbf{T}}\mathbf{W} \\ \mathbf{O} & \mathbf{Q}^{\mathbf{T}}\end{bmatrix}, \tag{6.5b}$$

where $\mathbf{P}\mathbf{P}^{\mathbf{T}} = \mathbf{M}_{local}$, and $\mathbf{Q}\mathbf{Q}^{\mathbf{T}} = \mathbf{S}$. Both $\mathbf{P}$ and $\mathbf{Q}$ are lower triangular matrix. We use the following notations for the two large triangular matrices:

$$\mathbf{A}_{\boldsymbol{\alpha}} = \begin{bmatrix}\mathbf{P} & \mathbf{O} \\ \mathbf{W}^{\mathbf{T}}\mathbf{E}\mathbf{M}^{-1}\mathbf{P} & -\mathbf{Q}\end{bmatrix}, \qquad \mathbf{A}_{\boldsymbol{\beta}} = \begin{bmatrix}\mathbf{P}^{\mathbf{T}} & \mathbf{P}^{\mathbf{T}}\mathbf{M}^{-1}\mathbf{E}^{\mathbf{T}}\mathbf{W} \\ \mathbf{O} & \mathbf{Q}^{\mathbf{T}}\end{bmatrix}.$$

Then we can compute ${\mathbf{A}_{\boldsymbol{\alpha}}}^{-1}$ and ${\mathbf{A}_{\boldsymbol{\beta}}}^{-1}$ using $\mathbf{P}^{-1}$ and $\mathbf{Q}^{-1}$:

$$\mathbf{A}_{\boldsymbol{\alpha}}^{-1} = \begin{bmatrix}\mathbf{P}^{-1} & \mathbf{O} \\ \mathbf{Q}^{-1}\mathbf{W}^{\mathbf{T}}\mathbf{E}\mathbf{M}^{-1}\mathbf{P}^{-1} & -\mathbf{Q}^{-1}\end{bmatrix}, \qquad \mathbf{A}_{\boldsymbol{\beta}}^{-1} = \begin{bmatrix}{\mathbf{P}^{-1}}^{T} & -{\mathbf{P}^{-1}}^{T}\mathbf{M}^{-1}\mathbf{E}^{\mathbf{T}}\mathbf{W}{\mathbf{Q}^{-1}}^{T} \\ \mathbf{O} & {\mathbf{Q}^{-1}}^{T}\end{bmatrix}.$$

To find $\mathbf{P^{-1}}$, we refer to (5.7). Finding $\mathbf{Q^{-1}}$ is much trickier. It requires Cholesky factorisation of matrix $\mathbf{S}$, then taking the inverse. This will be the most expensive part in finding the inverse of $\mathbf{G^{-1}}$.

Now, we are ready to compute $\mathbf{A^{-1}}$.

$$\mathbf{A^{-1}} = \mathbf{A_\beta^{-1}A_\alpha^{-1}} \tag{6.6a}$$

$$= \begin{bmatrix} \mathbf{M^{-1} - M^{-1}E^TWS^{-1}W^TEM^{-1}} & \mathbf{M^{-1}E^TWS^{-1}} \\ \mathbf{S^{-1}W^TEM^{-1}} & \mathbf{-S^{-1}} \end{bmatrix}. \tag{6.6b}$$

To assemble the $\mathbf{G^{-1}}$ from a local matrix $\mathbf{A^{-1}}$, we use (6.4):

$$\mathbf{G^{-1}} = \boldsymbol{\mathcal{P}}^T \left(\mathbf{I_4} \otimes \mathbf{A^{-1}}\right) \boldsymbol{\mathcal{P}} \tag{6.7a}$$

$$= \boldsymbol{\mathcal{P}}^T \left(\mathbf{I_4} \otimes \begin{bmatrix} \mathbf{M^{-1} - M^{-1}E^TWS^{-1}W^TEM^{-1}} & \mathbf{M^{-1}E^TWS^{-1}} \\ \mathbf{S^{-1}W^TEM^{-1}} & \mathbf{-S^{-1}} \end{bmatrix}\right) \boldsymbol{\mathcal{P}} \tag{6.7b}$$

$$= \begin{bmatrix} \mathbf{I_4} \otimes (\mathbf{M^{-1} - M^{-1}E^TWS^{-1}W^TEM^{-1}}) & \mathbf{I_4} \otimes (\mathbf{M^{-1}E^TWS^{-1}}) \\ \mathbf{I_4} \otimes (\mathbf{S^{-1}W^TEM^{-1}}) & \mathbf{I_4} \otimes (\mathbf{-S^{-1}}) \end{bmatrix}. \tag{6.7c}$$

Thus it follows naturally to compute Steklov-Poincaré operator $\boldsymbol{\mathcal{S}}$:

$$\boldsymbol{\mathcal{S}} = \mathbf{LG^{-1}L^T} \tag{6.8a}$$

$$= \begin{bmatrix} \mathbf{Wb^T} & \mathbf{O} \end{bmatrix} \begin{bmatrix} \mathbf{I_4} \otimes (\mathbf{M^{-1} - M^{-1}E^TWS^{-1}W^TEM^{-1}}) & \mathbf{I_4} \otimes (\mathbf{M^{-1}E^TWS^{-1}}) \\ \mathbf{I_4} \otimes (\mathbf{S^{-1}W^TEM^{-1}}) & \mathbf{I_4} \otimes (\mathbf{-S^{-1}}) \end{bmatrix} \begin{bmatrix} \mathbf{Wb} \\ \mathbf{O} \end{bmatrix} \tag{6.8b}$$

$$= \mathbf{Wb^T} \left\{\mathbf{I_4} \otimes (\mathbf{M^{-1} - M^{-1}E^TWS^{-1}W^TEM^{-1}})\right\} \mathbf{Wb} \tag{6.8c}$$

As we can see, for a domain decomposed by $n \times n$ subdomains and then discretised uniformly in order p, the major computational cost of constructing a Steklov-Poincaré operator is the time cost to compute the inverse of $\mathbf{A}$, which is independent from n. Although the memory demand is different depending on how fine h-refinement is done.

6.2.2. Condition Number of Steklov-Poincaré Operator

In this subsection, we look into the condition number change of $\boldsymbol{\mathcal{S}}$ with respect to h-refinement. We set up a test case where n increases from 2 to 6, and let p order be 5 uniformly in each element. We record the size and condition number of the big matrix $\mathbf{K}$ and $\boldsymbol{\mathcal{S}}$.

Table 6.1: Condition Number of $\mathbf{K}$ and $\boldsymbol{\mathcal{S}}$

n	2	3	4	5	6
$\kappa(\mathbf{K})$	54.06	108.70	184.98	282.89	402.48
size($\mathbf{K}$)	360^2	825^2	1480^2	2325^2	3360^2
$\kappa(\boldsymbol{\mathcal{S}})$	20.27	37.98	63.46	96.39	136.71
size($\boldsymbol{\mathcal{S}}$)	20^2	60^2	120^2	200^2	300^2

As we can see that the major challenge is no longer solving the Steklov-Poincaré operator related system, the system is small enough to be solved by a direct solver. However, we expect most of the work to be done is computing the solution of the sub-structured problems.

6.3. Test Case

In this test case, we will show the characteristics of domain decomposition methods. We let $f(x, y)$ be defined on $\boldsymbol{\Omega} = (-1, 1) \times (-1, 1)$ by:

$$f(x, y) = -900\pi^2(x^2y^4 + x^4y^2)cos(15\pi x^2y^2) - 30\pi(x^2 + y^2)sin(15\pi x^2y^2).$$

And we expect the solution to be:

$$\phi(x, y) = cos(15\pi x^2y^2).$$

Choosing this function ϕ is to mimic multiple scales with different wavenumber in one field thus increase the difficulty of convergence.

6.3.1. Flow Diagram

In this section, two flow diagrams are presented in Fig.6.2 for two solution schemes presented. One is the normal solution method and the other is a two-stage solution scheme.

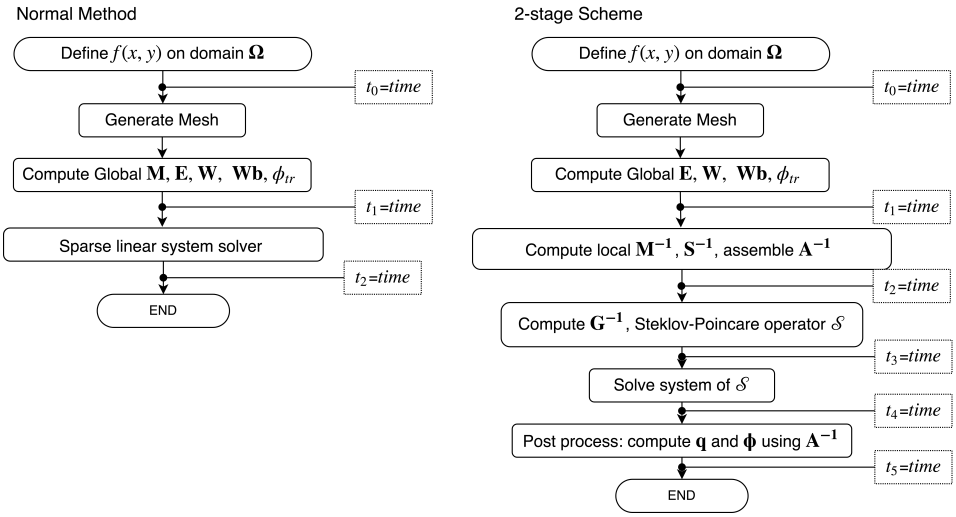

Figure 6.2: Flow Diagram for Test Programs

On the left, we use a normal method with two timed intervals. The first timed interval times the meshing and assembling time cost, and the second times the time used to solve the large linear equation system. On the right, we have five timed intervals. The first interval does not compute the global $\mathbf{M^{-1}}$ since it's no longer needed in the 2-stage algorithm. The second interval records the time cost to compute a local mass matrix inverse, and also the $\mathbf{S^{-1}}$ matrix in a single element. Both $\mathbf{M^{-1}}$ and $\mathbf{S^{-1}}$ are computed according to §5.2. The third interval records the time cost of assembling $\mathbf{G^{-1}}$ and $\boldsymbol{\mathcal{S}}$, which is following the derivation of §6.2.1. The fourth interval solves the $\boldsymbol{\mathcal{S}}$ operator. And the fifth interval computes the post-processing time cost.

6.3.2. Experiment Setup

This test case includes both h- and p- refinements. In the beginning of §6.2, we see that as p goes higher, the more system size reduction we could expect. Thus we let n increase by 1 at a time, from 2 to 7, which is called one campaign. For both methods, we repeat the campaigns under three p orders: $p = 5, 13, 25$.

6.3.3. Normal Solution Method Results

We first present the results in the following three tables, then we will discuss the results.

Table 6.2: h-refined Normal Method with $p = 5$

n	2	3	4	5	6	7
size($\mathbf{K}$)	360^2	825^2	1480^2	2325^2	3360^2	4585^2
$t_1 - t_0$	0.575 s	0.681 s	0.847 s	1.253 s	1.850 s	3.127 s
$t_2 - t_1$	0.00138 s	0.00376 s	0.00720 s	0.0152 s	0.0204 s	0.0308 s
$\mathcal{L}^2$ Error	1.455	1.058	1.059	0.879	0.786	0.779

Table 6.3: h-refined Normal Method with $p = 13$

n	2	3	4	5	6	7
size(**K**)	2184^2	4953^2	8840^2	13856^2	19968^2	27209^2
$t_1 - t_0$	2.512 s	5.767 s	13.131 s	30.814 s	59.994 s	132.649 s
$t_2 - t_1$	0.0705 s	0.193 s	0.674 s	1.175 s	1.654 s	2.613 s
$\mathcal{L}^2$ Error	1.341	0.715	0.498	0.257	0.0976	0.0335

Table 6.4: h-refined Normal Method with $p = 25$

n	2	3	4	5	6	7
size(**K**)	7800^2	17625^2	31400^2	49125^2	70800^2	96425^2
$t_1 - t_0$	17.434 s	62.556 s	241.824 s	632.099 s	/	/
$t_2 - t_1$	2.343 s	6.345 s	17.768 s	50.056 s	/	/
$\mathcal{L}^2$ Error	0.398	0.0331	0.00122	$4.118e{-5}$	/	/

The first thing we need to discuss is the system scaling. Since the normal method solves DoF at once, the system row size is $n^2(3p^2 + 4p) - 2np$, with $n^2p^2(p + 1) + 2p^3(p + 1)$ non-zero elements. If we check the ratio of non-zero element in **K** with respect to the total number of elements:

$$\frac{n^2p^2(p+1) + 2p^3(p+1)}{(n^2(3p^2+4p) - 2np)^2} = 2\left(\frac{p+1}{3np + 4n - 2}\right)^2,$$

we see that h-refinement is more favourable in terms of sparsity pattern, thus easy to solve. However, from the error analysis perspective, p-refinement makes convergence much more efficient. Tests on this method is only used as a reference, since a system formed by domain decomposition is not meant to be solve directly anyhow.

6.3.4. 2-stage Scheme Results

The results are presented with notions according to Fig.6.2.

Table 6.5: h-refined 2-stage Scheme with $p = 5$

n	2	3	4	5	6	7
size($\boldsymbol{S}$)	20^2	60^2	120^2	200^2	300^2	420^2
$t_1 - t_0$	0.493 s	0.542 s	0.676 s	0.8837 s	1.177 s	1.713 s
$t_2 - t_1$	0.324 s	0.315 s	0.368 s	0.346 s	0.340 s	0.358 s
$t_3 - t_2$	0.00110 s	0.00164 s	0.00235 s	0.00346 s	0.00464 s	0.00586 s
$t_4 - t_3$	0.000250 s	0.000292 s	0.000544 s	0.000806 s	0.00128 s	0.00174 s
$t_5 - t_4$	0.00182 s	0.00312 s	0.00621 s	0.00804 s	0.0106 s	0.0146 s
$\mathcal{L}^2$ Error	1.455	1.058	1.059	0.879	0.786	0.779

Table 6.6: h-refined 2-stage Scheme with $p = 13$

n	2	3	4	5	6	7
size($\boldsymbol{S}$)	52^2	156^2	312^2	520^2	780^2	1092^2
$t_1 - t_0$	2.014 s	3.686 s	7.534 s	15.573 s	28.623 s	53.151 s
$t_2 - t_1$	1.277 s	1.207 s	1.241 s	1.206 s	1.234 s	1.227 s
$t_3 - t_2$	0.00793 s	0.0136 s	0.0223 s	0.0352 s	0.0508 s	0.0670 s
$t_4 - t_3$	0.000437 s	0.000913 s	0.00229 s	0.00738 s	0.0111 s	0.0171 s
$t_5 - t_4$	0.0120 s	0.0240 s	0.0413 s	0.0662 s	0.103 s	0.132 s
$\mathcal{L}^2$ Error	1.341	0.715	0.498	0.257	0.0976	0.0335

Table 6.7: h-refined 2-stage Scheme with $p = 25$

n	2	3	4	5	6	7
size($\boldsymbol{\mathcal{S}}$)	100^2	300^2	600^2	1000^2	1500^2	2100^2
$t_1 - t_0$	10.830 s	33.871 s	93.888 s	209.720 s	436.211 s	771.850 s
$t_2 - t_1$	6.098 s	5.969 s	6.219 s	5.948 s	5.854 s	6.225 s
$t_3 - t_2$	0.0540 s	0.116 s	0.202 s	0.323 s	0.465 s	0.614 s
$t_4 - t_3$	0.000941 s	0.00499 s	0.00796 s	0.0166 s	0.0546 s	0.0841 s
$t_5 - t_4$	0.0903 s	0.181 s	0.269 s	0.497 s	0.712 s	0.957 s
$\mathcal{L}^2$ Error	0.398	0.0331	0.00122	$4.118e{-5}$	$1.710e{-6}$	$6.581e{-8}$

Shown the results, a few remarks have to be made. We see that the major cost of time is spent on the computation of $\mathbf{A^{-1}}$, however, the cost is only related to the p-order of a single local element. Thus this part of the cost is invariant with h-refinement, as shown in Fig.6.3. Computing $\mathbf{G^{-1}}$ from $\mathbf{A^{-1}}$ is no more than assembling by block, thus the cost creeps slowly with n.

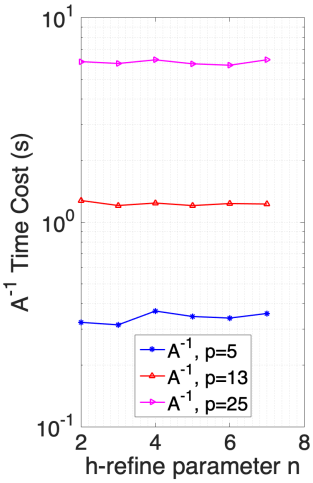

Figure 6.3: Time Cost to Find $\mathbf{A^{-1}}$

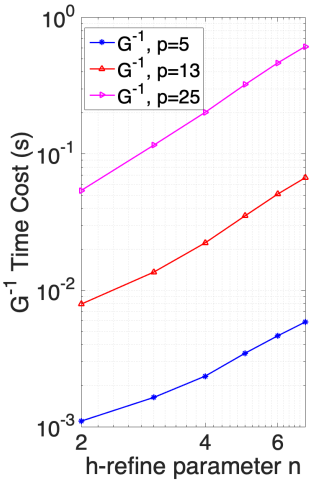

Figure 6.4: Time Cost to Find $\mathbf{G^{-1}}$

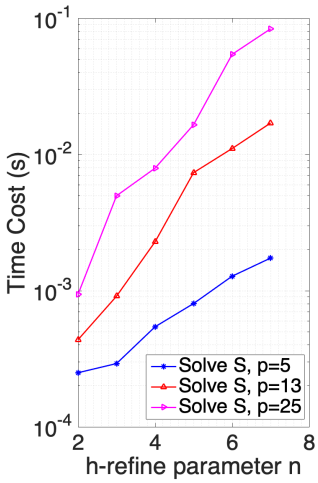

Figure 6.5: Time Cost to Solve $\boldsymbol{\mathcal{S}}$

Also, we are interested in the cost of solving $\boldsymbol{\mathcal{S}}$, thus we plot the time cost in Fig.6.5. The zigzag curve in Fig.6.5 indicates that three times' repetition is not enough to approach to a stable averaged out come, and it also indicates that solving $\boldsymbol{\mathcal{S}}$ is cheap, thus the time cost variance is very large.

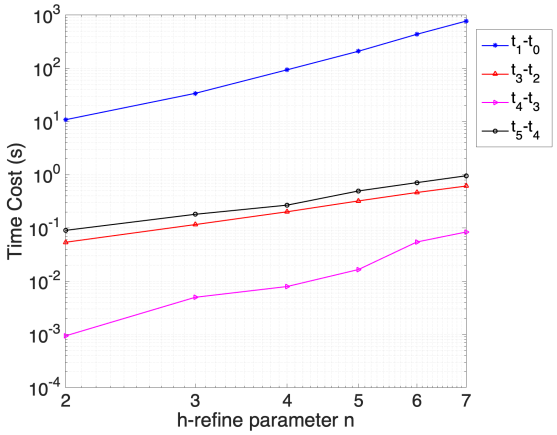

Figure 6.6: Time Cost in 2-stage Solver, $p = 25$

At last, we plot the time cost of each step in Table 6.7 into the figure below. We see that making mesh and assembling matrices are the most costly process, and this part of the cost increase with n at the fastest rate.

6.3.5. Result Comparison

First, we compare the solution time of the two methods. For the 2-stage method, we count $t_5 - t_1$ as the time used to solve the system. And $t_1 - t_0$ is the time used for generating mesh and assembling linear equation systems. Take $p = 13$, compare results in Table 6.3 and 6.6, we plot the following graph 6.7.

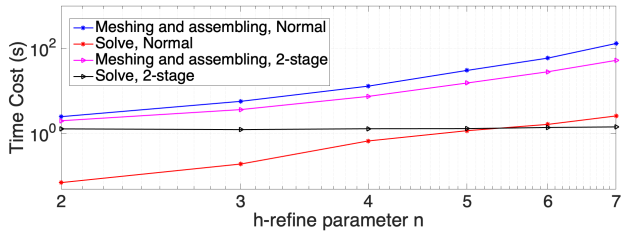

Figure 6.7: Time Cost Compare, $p = 13$

We can see that the 2-stage method uses much less time to generate mesh and assemble equation system. Also, to solve the linear equation system, the 2-stage solver's solution time is also very stable as n increases, and the normal direct solver's solution time scales up fast as n increases. When $p = 25$, the direct solver barely has any advantage in solving such a large system.

6.3.6. Error Analysis and Comments

We plot the error against h-refinement in Fig.6.8. h-refinement is known to have algebraic convergence behaviour, which means that the error decrease algebraically on the semi-logarithm plot as n increase. The slop is determined by the method order p within each element. Thus the higher the polynomial degree is, the more efficient the h-refinement will be. Compare to p-refinement's exponential convergence, algebraic convergence may not be as attractive. However, recall from the previous results, e.g., in Fig.6.3 and Fig.6.7, the time cost to solve an increasingly h-refined system does not differ much (as long as p is fixed), but solving an increasingly p-refined system will cause the solution time grow significantly. We should also remember that $\mathbf{A}^{-1}$ is reusable if further h-refinement is done!

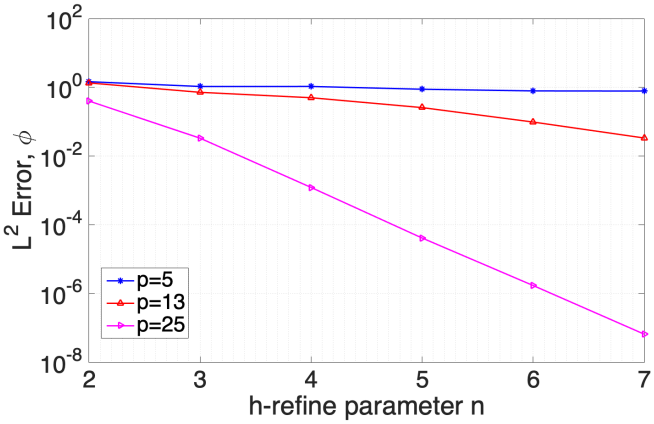

Figure 6.8: Convergence with h-refinement

In real simulations, the level of refinement is determined by balancing the factors such as resolution requirements, complexity of geometry and computation time available. Thus for a simulation problem, we can use p-refinement at the beginning until the solution cost become unacceptable before

the expected accuracy is reached. Then we can use the highest p-order ever reached and then go for h-refinements. The principal is the let h-refinement "ride" on the highest p-order thus produce the most convergence efficiency.

We present some simulation results in Appendix A. The exact solution is plotted here:

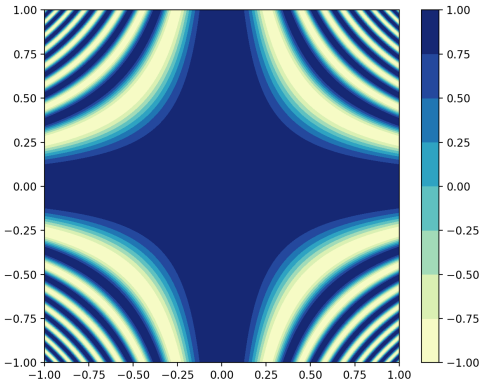

Figure 6.9: Exact Solution $\phi(x, y)$

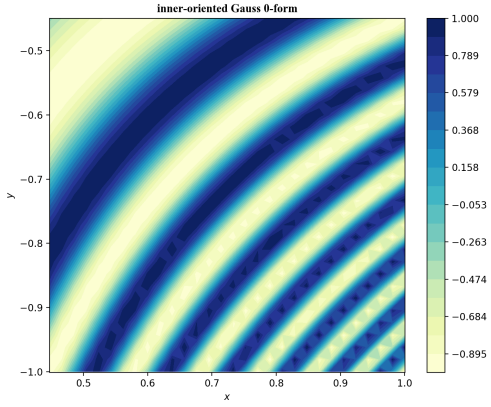

Figure 6.10: Result, Right Corner, $p = 25$, $n = 7$

Compare the simulation results, the cross-like part in the middle represents a macro-structure in the solution. These large structures with small wavenumber are well captured and represented by the basis functions. At the four corners, the exact solution has a large wavenumber. In order to resolve the high frequency components in the solution, the mesh has to be fully refined to the same geometric scale. When $p = 25$ and $n = 7$, we get a result close to the exact solution, however, if we zoom in to the corner as in Fig.6.10, we can see that high frequency part needs to be resolved completely by further refinements.

7
Conclusions and Recommendations

In this chapter, we will look back on what we have achieved, then give conclusions and recommendations as the ending of this report.

7.1. Conclusions

In this thesis, a systematic research was made on preconditioning techniques suitable for MSEM implementations. In Chapter 4, we made analysis on the limits of spectrum of discrete operators that will appear in MSEM discretisation. We analytically saw me demonstrated how the maximum and minimum of singular values (or eigenvalues) scale with p-refinement on an orthogonal mesh. Using the knowledge about the spectrum, in Chapter 5 and Chapter 6, we developed a series of preconditioning techniques and then showed the significant decrease in solution time after these preconditioners are applied. Now, we will focus on the research questions and make the following conclusions and discussions.

7.1.1. Cause of Ill-conditioned Linear Systems in MSEM

In MSEM, the discretisation is "mimetic", in the sense that it differentiates topological and metrical relations, thus the discrete operators mimics the PDE operators and preserves the structure from continuous level to the discrete level. Thus the discrete operators may be metric dependent and metric independent. In MSEM, the mass matrix $\mathbf{M}$ represents the inner product operation in PDE. This operation conducts a measurement, thus is metric dependent. The incidence matrix $\mathbf{E}$ is a purely topological differential operator with no metrics involved. The wedge matrix $\mathbf{W}$ is a discrete representation of basis duality pairing, which will be metric independent as well.

From §4.2, we see how the mass matrix condition number grows with p-refinements. Recall that the maximum and minimum of the eigenvalues in $\mathbf{M_l}$ and $\mathbf{M_e}$ corresponds to the DoF that occupies the smallest and the largest cell, see Theorem 4.2.1 for instance. This means that the squeezed cells makes the metrics unfavourable since it makes the spectrum scatter. The major task of preconditioning is to find the distorted metrics and then use these information to construct a preconditioner. To give an example, we see that the diagonal scaling of mass matrix is actually correcting the squeezing on the mesh (this is explained more in-depth in §7.1.2). Operators that are only topological, such as $\mathbf{E}$ and $\mathbf{W}$, do not change in different coordinate systems. Although an increasingly complicated topological structure may induce a growing condition number such as matrix $\mathbf{E}$, the pattern of these matrices are fixed. Thus finding preconditioners for these types of operators could be a once-and-for-all kind of effort.

Thus we conclude that the ill-condition of the linear system in MSEM is primarily caused by metric dependent factors such as mesh deformation, constitutive parameter change, which deserves a lot of attention. And we point out that ill-conditioning is also produced by an increasingly complex topological structure, e.g., the singular values of $\mathbf{E}$ scatter as the mesh size go up.

After this thesis project, we can analytically estimate the condition number change under mesh compression. And we have found simple preconditioners for the mass matrix on an orthogonal domain, which performed well on orthogonal domain and was proved to be very effective on highly distorted meshes as well. From numerical experiments in §5.4.2, we see that mesh compression affects the condition number much more than mesh distortion.

Until now, the first three research questions are answered, and the last research question will be given in the coming sections.

7.1.2. Think in Terms of Consistency

Consistent discretisation schemes ensure that the solution pointwisely converges to the exact solution of the continuous PDE. This means that the linear system of equations formed by different numerical schemes on the same mesh should be at least similar in some way.

Remember from Theorem 4.2.1, the eigenvalues of 1D mass matrix $\mathbf{M_l}$ lies in the interval $[Cp^{-2}, Cp^{-1}]$. If we think in terms of consistency, we consider a finite volume scheme, then there is an equivalent mass matrix $\mathbf{M_{FV}}$ which is diagonal, and the diagonals will be a measurement of the "volume" (length in 1D) of each cell. According to the definition of Legendre-Gauss-Lobatto points in §3.3, we see exactly that the minimum cell volume scales with p^{-2} and the maximum cell volume scales with p^{-1}.

This perspective allows us to see more possibilities in constructing preconditioners. As a matter of fact, many preconditioning techniques are developed using the consistency of discretisation techniques such as multigrid techniques.

7.1.3. Framework of Studying Preconditioning Techniques

There is a hidden story line in this thesis which is not explicitly discussed. But in order to show that the value of this preliminary research is far more than successfully preconditioned a Poisson problem, we give a common framework for studies on preconditioning.

In this thesis, we followed a 3-layer hierarchy to approach the problem. The first layer is the fundamental knowledge about the discrete operators. On this fundamental level of research, we are interested in the eigenvalue distribution of these operators. And by inspecting the limits of the eigenvalues, the cause of the scatted spectrum may naturally be exposed, just like what we found in Chapter 4. On top of the first layer, we approach to the block structure of the system given by the PDE. This level of research finds a suitable way to factorise the matrix and implement the preconditioners for each operator. The third layer of research goes to a detailed research on how to properly choose the preconditioners for operators thus the preconditioned system will be compatible with the linear system solver and gain the best efficiency.

Table 7.1: Operator Spectrum Studied at the Current Stage

		p-version		Hybrid hp-version		Continuous hp-version
		Orthogonal	Distorted	Orthogonal	Distorted	
Spectrum Study of Discrete Operators	**M**	✓		✓		
	W	✓	✓	✓	✓	✓
	E	✓	✓	✓	✓	✓

In Table 7.1, we show the operators that have been studied and this gives motivations for the coming up study proposed in the recommendation chapter. The block structure we studied until now is always a saddle point problem. This type of structure will be typical for Poisson problems, Stokes Problem.... For this type of structure, previous studies give mature solutions that performed extremely well in our experiments. For the future studies, the problem type may be different, however, we expect the study to follow the same framework, from the fundamentals of the operators to the global linear system.

7.2. Recommendation

From the conclusion section, we make the following recommendations.

7.2.1. About Follow-up Research

From this thesis, we have seen the eigenvalue bounds of the mass matrices (on orthogonal domain) and incidence matrices through theorems. However, the lower bound of the eigenvalue of matrix $\mathbf{M_e}$ is not proven by a solid theorem, as well as the wedge matrices $\mathbf{W}$. However, the essence of these missing works is focused on one inequality about edge functions. Recent research on dual polynomials (see [35]) indicates that $\mathbf{W}$ may no longer be necessary to appear in the system, however, a follow up on $\mathbf{W}$ spectrum will be good. Also, for a distorted mesh, we could also give a theoretical study on the influence of mappings over the spectrum of mass matrices.

As the test case problems become more and more complicated, for instance, convection problems, convection diffusion problems..., we expect the following research to cover even more operators other than only mass matrix, incidence matrix and wedge matrix. And studies on the corresponding block structures will need to be studied.

7.2.2. About Hybrid and Continuous *hp*-refinement

In hybrid *hp*-MSEM, we decoupled the influence of subdomains by using discontinuous DoF on the boundary between neighbouring elements, and then use a Lagrange multiplier to enforce the equality of these boundary DoF. But in the test case that we did, we can accomplish the same kind of domain decomposition by using a continuous *hp*-MSEM. With a continuous *hp*-MSEM, we expect a small intersection of local mass matrices since the DoF on the boundary of neighbouring cells are correlated to elements on both sides. However, we can use a proper ordering (or a permutation matrix) to achieve a block diagonal structure of the mass matrix and then partition the computation task.

7.2.3. About Mesh Deformation

In one of my experiments with orthogonal meshes, I tried to precondition a mass matrix $\mathbf{M}_p$ with p-refinement with another mass matrix $\mathbf{M}_h$ obtained from a continuous h-refined MSEM. If we look at the shape of $\mathbf{M}_p$ and the shape of $\mathbf{M}_h$, other than the scale and detail, they does look similar, which could also be explained by the consistency principle discussed before. However the $\mathbf{M}_h$ matrix is not a good preconditioner for $\mathbf{M}_p$.

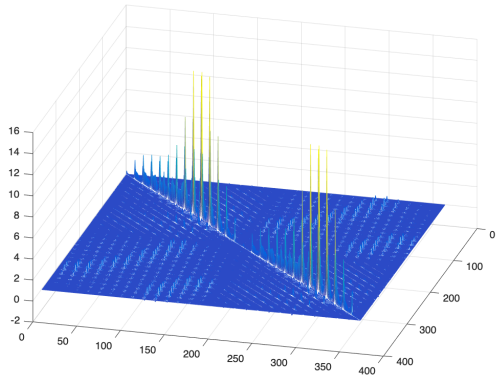

Figure 7.1: Shape of $\mathbf{M}_p$, $p = 13$, $n = 1$, $c = 0.2$

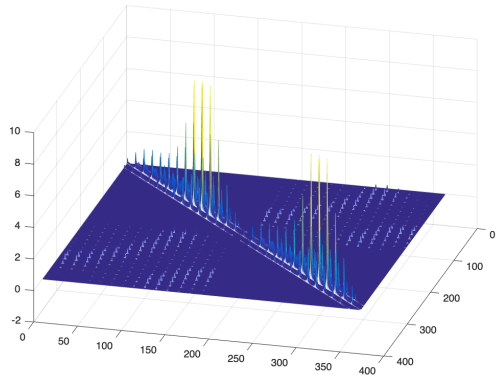

Figure 7.2: Shape of $\mathbf{M}_h$, $p = 1$, $n = 13$, $c = 0.2$

If we see the sparsity of these two matrices, we see a good sparsity pattern of $\mathbf{M}_h$. This sparsity pattern reveals the "windows" in the matrix that is affected by mesh distortion and compression. What we can do later on is to figure out a way to combine the preconditioner we had from this research with the $\mathbf{M}_h$ such that the ill-condition caused by mesh distortion could be eliminated.

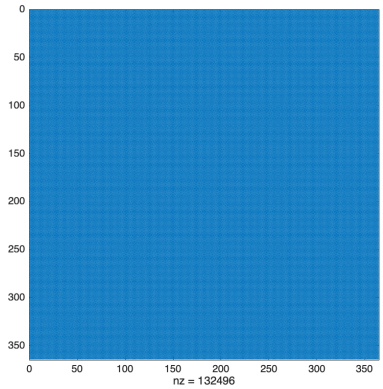

Figure 7.3: Sparsity of $\mathbf{M}_p$, $p = 13$, $n = 1$, $c = 0.2$

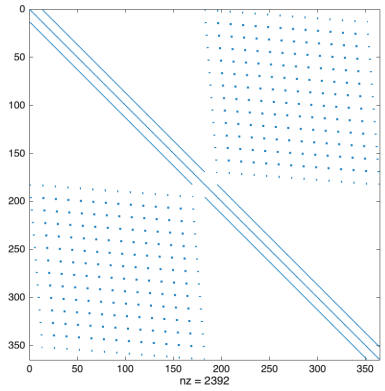

Figure 7.4: Sparsity of $\mathbf{M}_h$, $p = 1$, $n = 13$, $c = 0.2$

Acknowledgements

I am always wondering: where does my soul and spirit come from and where will my spirit and memories be? Is physics and biology really capable of explaining such an abstract question in the way I wanted? Everytime I think about this question, it makes me feel a desperate lonesome, since as an independent soul, I see a finite world line for my life, my memories, my love and my thoughts. However, this is a question with no answer. The best I can do for now is to cherish everything I had and appreciate my script of life, which is maybe partially determined by myself.

Thus in the end of the thesis report, I would like to give my sincere gratitude and acknowledgement to my supervisor, Marc Gerritsma. Marc is very inspiring and passionate about his work. During my thesis, I received a lot of his help and I will always appreciate. The program used in this thesis was based on Zhang Yi's previous work. Without his explanation and contribution to the MSEM library, there won't be this thesis. Thus I would like to acknowledge Yi, Varun, and the people that helped me with my thesis project.

This thesis is the termination of my student life in Delft, thus hereby, I want to dedicate my gratitude again to my dearest parents, who loved me unconditionally and permanently wherever we will be. I am very luck to be their child and will always love them. In Delft, my friends are my family. I would like to give my thanks to Shengtai, Qian, Shuaidong, David, Boyao, Linxi, Rob, Albertine, and all the friends that supported me so much.

A

Simulation Plots

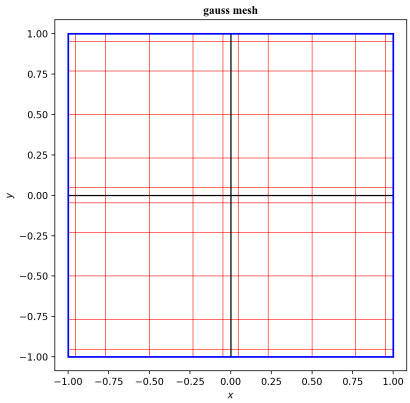

Figure A.1: Mesh, $p = 5$, $n = 2$

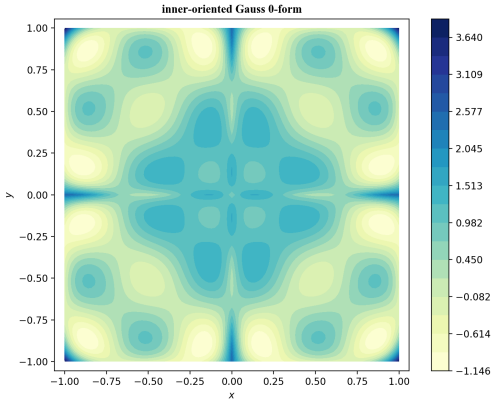

Figure A.2: Mesh, $p = 5$, $n = 2$

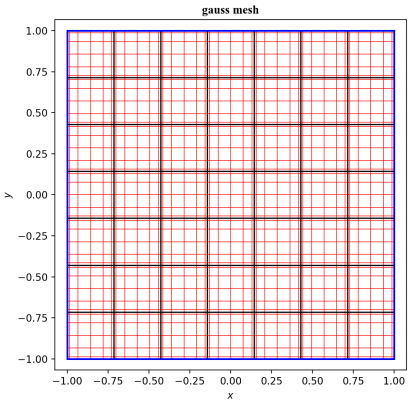

Figure A.3: Mesh, $p = 5$, $n = 7$

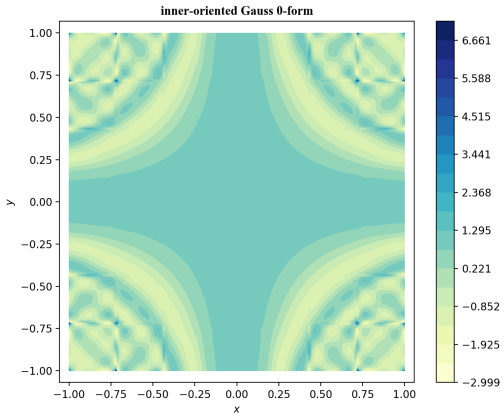

Figure A.4: Result, $p = 5$, $n = 7$

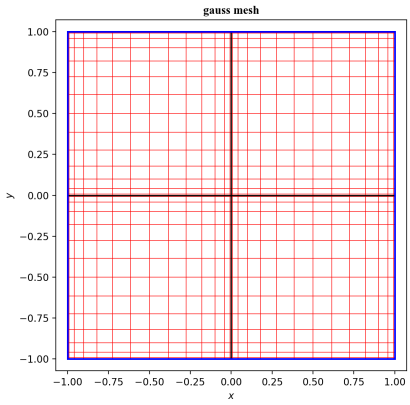

Figure A.5: Mesh, $p = 13$, $n = 2$

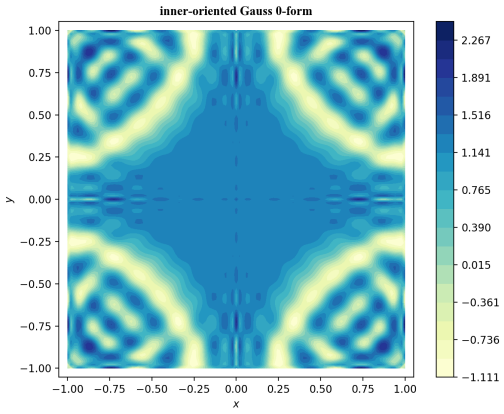

Figure A.6: Result, $p = 13$, $n = 2$

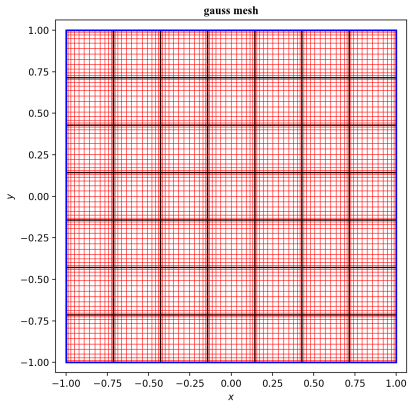

Figure A.7: Mesh, $p = 13$, $n = 7$

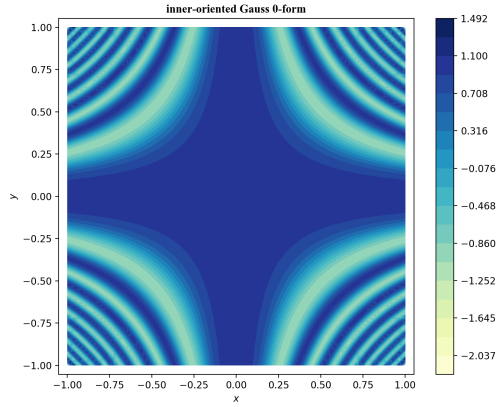

Figure A.8: Result, $p = 13$, $n = 7$

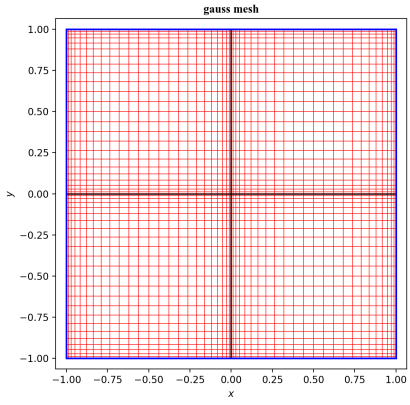

Figure A.9: Mesh, $p = 25$, $n = 2$

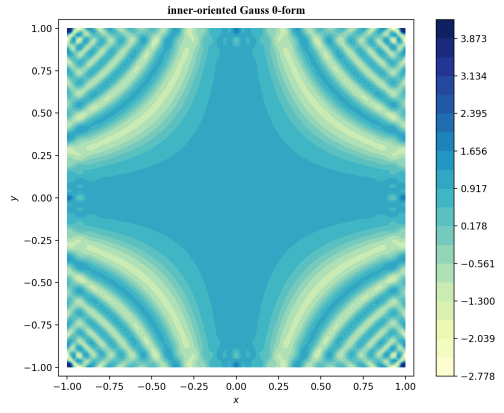

Figure A.10: Result, $p = 25$, $n = 2$

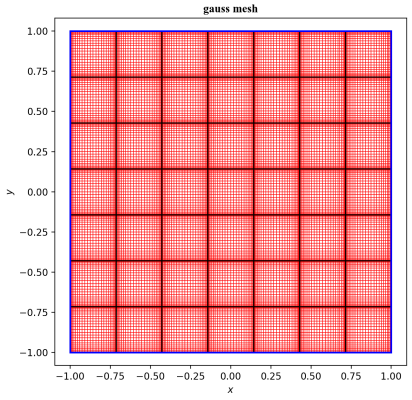

Figure A.11: Mesh, $p = 25$, $n = 7$

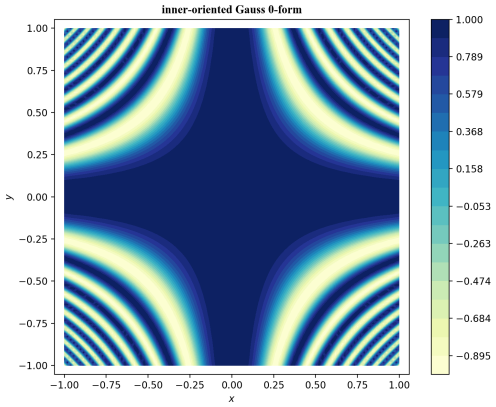

Figure A.12: Result, $p = 25$, $n = 7$

Bibliography

[1] E. Anderson, Z. Bai, C. Bischof, L. S. Blackford, J. Demmel, J. Dongarra, J. du Croz, A. Greenbaum, S. Hammarling, A. McKenney, et al. *LAPACK Users' guide*. SIAM, 1999.

[2] L. Andrews. *Special functions of mathematics for engineers*. McGraw-Hill New York, 1992.

[3] O. Axelsson and M. Neytcheva. Preconditioning methods for linear systems arising in constrained optimization problems. *Numerical linear algebra with applications*, 10(1-2):3–31, 2003.

[4] M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of computational Physics*, 182(2):418–477, 2002.

[5] M. Benzi and C. Meyer. A direct projection method for sparse linear systems. *SIAM Journal on Scientific Computing*, 16(5):1159–1176, 1995.

[6] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30(2-3):305–340, 1999.

[7] M. Benzi, C. D. Meyer, and M. Tůma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 17(5):1135–1149, 1996.

[8] M. Benzi, J. K. Cullum, and M. Tůma. Robust approximate inverse preconditioning for the conjugate gradient method. *SIAM Journal on Scientific Computing*, 22(4):1318–1332, 2000. doi: 10.1137/S1064827599356900.

[9] P. Bochev. A discourse on variational and geometric aspects of stability of discretizations. *33rd Computational Fluid Dynamics Lecture Series, VKI LS*, 5, 2003.

[10] P. B. Bochev and J. M. Hyman. Principles of mimetic discretizations of differential operators. In *Compatible spatial discretizations*, pages 89–119. Springer, 2006.

[11] J. H. Bramble and J. E. Pasciak. A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems. *Mathematics of Computation*, 50(181):1–17, 1988.

[12] J. H. Bramble and J. E. Pasciak. A domain decomposition technique for stokes problems. *Applied Numerical Mathematics*, 6(4):251–261, 1990.

[13] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring. i. *Mathematics of Computation*, 47(175):103–134, 1986.

[14] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. An iterative method for elliptic problems on regions partitioned into substructures. *Mathematics of Computation*, 46(174):361–369, 1986.

[15] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring. ii. *Mathematics of Computation*, 49(179):1–16, 1987.

[16] J. H. Bramble, R. E. Ewing, J. E. Pasciak, and A. H. Schatz. A preconditioning technique for the efficient solution of problems with local grid refinement. *Computer Methods in Applied Mechanics and Engineering*, 67(2):149–159, 1988.

[17] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring. iii. *Mathematics of Computation*, 51(184):415–430, 1988.

[18] J. H. Bramble, J. E. Pasciak, and A. H. Schatz. The construction of preconditioners for elliptic problems by substructuring. iv. *Mathematics of Computation*, 53(187):1–24, 1989.

[19] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev. Analysis of the inexact uzawa algorithm for saddle point problems. *SIAM Journal on Numerical Analysis*, 34(3):1072–1092, 1997.

[20] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

[21] C. Canuto, Y. M. Hussaini, A. Quarteroni, A. Thomas Jr, and Others. *Spectral methods in fluid dynamics*. Springer Science & Business Media, 2012.

[22] F. Carlini. *Ricerche sulla convergenza della serie che serve alla soluzione del problema di Keplero*.

[23] E. Chow and Y. Saad. Approximate inverse preconditioners for general sparse matrices. *Res. Rep. UMSI*, 94(1.01), 1994.

[24] T. A. Davis. *Direct methods for sparse linear systems*, volume 2. Siam, 2006.

[25] H. S. Dollar and A. J. Wathen. Approximate factorization constraint preconditioners for saddle-point matrices. *SIAM Journal on Scientific Computing*, 27(5):1555–1572, 2006.

[26] H. S. Dollar, N. I. M. Gould, W. H. A. Schilders, and A. J. Wathen. Using constraint preconditioners with regularized saddle-point problems. *Computational Optimization and Applications*, 36 (2-3):249–270, 2007.

[27] H. S. Dollar, N. I. M. Gould, M. Stoll, and A. J. Wathen. Preconditioning saddle-point systems with applications in optimization. *SIAM Journal on Scientific Computing*, 32(1):249–270, 2010.

[28] R. E. Ewing, R. D. Lazarov, P. Lu, and P. S. Vassilevski. Preconditioning indefinite systems arising from mixed finite element discretization of second-order elliptic problems. In *Preconditioned conjugate gradient methods*, pages 28–43. Springer, 1990.

[29] M. Gerritsma. Edge functions for spectral element methods. In *Spectral and High Order Methods for Partial Differential Equations*, volume 76, pages 199–207, 10 2010.

[30] M. Gerritsma. An introduction to a compatible spectral discretization method. *Mechanics of Advanced Materials and Structures*, 19(1-3):48–67, 2012.

[31] A. Günnel, R. Herzog, and E. Sachs. A note on preconditioners and scalar products in krylov subspace methods for self-adjoint problems in hilbert space. *Electronic Transactions on Numerical Analysis*, 41:13–20, 2014.

[32] I. Gustafsson. A class of first order factorization methods. *BIT Numerical Mathematics*, 18(2): 142–156, 1978.

[33] N. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, second edition, 2002.

[34] E. Hilb. Über die laplacesche reihe. *Mathematische Zeitschrift*, 5:17–25, 1919.

[35] V. Jain, Y. Zhang, A. Palha, and M. Gerritsma. Construction and application of algebraic dual polynomial representations for finite element methods. *ArXiv e-prints*, dec 2017.

[36] W. Kahan. Ieee standard 754 for binary floating-point arithmetic. *Lecture Notes on the Status of IEEE*, 754(94720-1776):11, 1996.

[37] C. Keller, N. I. M. Gould, and A. J. Wathen. Constraint preconditioning for indefinite linear systems. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1300–1317, 2000.

[38] J. J. Kreeft. Mimetic spectral element method, a discretization of geometry and physics, 2013.

[39] D. Kulkarni, D. Schmidt, and S. Tsui. Eigenvalues of tridiagonal pseudo-toeplitz matrices. *Linear Algebra and its Applications*, 297:63–80, 1999.

[40] L. Luksan and J. Vlcek. Indefinitely preconditioned inexact newton method for large sparse equality constrained nonlinear programming problems. *Numerical linear algebra with applications*, 5(3):219–247, 1998.

[41] K-A. Mardal and R. Winther. Preconditioning discretizations of systems of partial differential equations. *Numerical Linear Algebra with Applications*, 18(1):1–40, 2011.

[42] J. M. Melenk. On condition numbers in hp-fem with gauss-lobatto-based shape functions. *Journal of Computational and Applied Mathematics*, 139(1):21–48, 2002.

[43] A. Palha, P. P. Rebelo, R. Hiemstra, J. Kreeft, and M. Gerritsma. Physics-compatible discretization techniques on single and dual grids, with application to the poisson equation of volume forms. *Journal of Computational Physics*, 257:1394 – 1422, 2014.

[44] Y. Saad. Ilut: A dual threshold incomplete lu factorization. *Numerical linear algebra with applications*, 1(4):387–402, 1994.

[45] G. Szeg. *Orthogonal polynomials*, volume 23. American Mathematical Soc., 1939.

[46] A. J. Wathen. Preconditioning. *Acta Numerica*, 24:329–376, 2015.

[47] G. N. Watson. *A treatise on the theory of Bessel functions*. Cambridge university press, 1995.

[48] J. W. Watts III et al. A conjugate gradient-truncated direct method for the iterative solution of the reservoir simulation pressure equation. *Society of Petroleum Engineers Journal*, 21(03): 345–353, 1981.