Towards Scalable Topology Optimization, Classical or Quantum?

by

Zisheng Ye

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2025

Date of final oral examination: 05/05/2025

The dissertation is approved by the following members of the Final Oral Committee:

Wenxiao Pan, Associate Professor, Mechanical Engineering
Xiaoping Qian, Professor, Mechanical Engineering
Dan Negrut, Professor, Mechanical Engineering
Jinlong Wu, Assistant Professor, Mechanical Engineering
Bu Wang, Associate Professor, Civil and Environmental Engineering
Xiaozhe Hu, Associate Professor, Mathematics, Tufts University

ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere gratitude to those who have contributed to the completion of my doctoral thesis.

First and foremost, I am deeply thankful to my advisor, Dr. Wenxiao Pan, for her invaluable guidance, unwavering support, and insightful advice throughout my PhD journey. Her systematic training and mentorship have played a pivotal role in shaping me into a more rigorous and logical researcher.

I am also grateful to the members of my dissertation committee, Dr. Xiaoping Qian, Dr. Dan Negrut, Dr. Jinlong Wu, Dr. Bu Wang, Dr. Xiaozhe Hu and Dr. Wenxiao Pan, for their time, patience, and valuable feedback on this work. Their expertise and constructive criticism have significantly enriched the quality of this thesis.

Next, I want to express my thanks to my lab-mates and collaborators, especially Zhan Ma, for their exceptional collaboration and support. I extend my appreciation to all friends Chuanqi Chen, Chao Hu, Zhiwei Tu and my cousin Ziqi Gu for their assistance throughout my PhD career.

Furthermore, I would like to extend my heartfelt gratitude to Ziying (Sarah) Han. Her unwavering belief in me, even in moments of self-doubt, has propelled me forward. Her love, companionship and support have illuminated my path and enriched every aspect of my life.

Finally, I extend my thanks to my parents, Xin Ye and Hanbing Gu, my grandparents Huankui Ye, Judi Wang, Tianhu Gu and Liudi Wu for their encouragement and sacrifices. Their boundless support has been the cornerstone of my academic journey, and I am profoundly grateful for having them in my life.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Continuum topology optimization (TO), originated from structural mechanics, aims to find optimal distributions of materials to improve the performance of designs under governing physical equations described with partial differential equations (PDEs). Discrete variable topology optimization (DVTO) employs binary design variables to represent optimal topologies with sharp and clear boundaries, eliminating the need for post-processing. However, achieving high-fidelity designs requires fine discretization, leading to large-scale mixed-integer nonlinear programming (MINLP) problems. This thesis proposes a new scalable framework for solving the large-scale TO problems, with implementations for both classical and quantum computing. It discusses the proposed framework in the integration of classical and quantum computing for solving the large-scale TO problems.

The proposed framework in this thesis can tremendously reduce the number of iteration steps required to achieve optimality, compared to the conventional continuous relaxation based methods like the solid isotropic material with penalization (SIMP) method. A series of mixed integer linear programming (MILP) problems are constructed to the MINLP formulation under the proposed framework. A new optimizer based on Dantzig-Wolfe (DW) decomposition is proposed to solve the MINLP formulation, which leverages the block-angular structure of TO problems. The proposed optimizer enables a parallel implementation of the optimization process, which can take advantage of the computational resources available in the scalable computation environment used for solving the large-scale PDEs. The new proposed formulation based on DW decomposition also enables a simple implementation of a quadratic unconstrained binary optimization (QUBO) problem, which can be embeded on near-term quantum computers for further acceleration of the optimization process. The pro-

posed framework is validated through a series of numerical experiments, ranging from the single-material minimum compliance problem to the multi-material compliant mechanism design problem. The results demonstrates the effectiveness of the proposed framework in solving large-scale TO problems, including the design of complex structures with multiple candidate materials.

A geometric multi-grid (GMG) preconditioner, as a classically scalable approach, based on the generalized moving least square (GMLS) method is presented to implement a scalable PDE solver with moving boundaries in fluid-solid interaction problems. Due to the lack of large enough mature quantum computers and the ill-conditioning of the linear systems arising from the discretization of PDEs, this thesis only investigates and discusses the commonly used quantum computing algorithms for solving the linear systems and the potential approach to develop the quantum algorithms for solving the linear systems arising from TO problems.

1 INTRODUCTION

> In terms of size [of transistors] you can see that we're approaching the size of atoms which is a fundamental barrier, but it'll be two or three generations before we get that far—but that's as far out as we've ever been able to see. We have another 10 to 20 years before we reach a fundamental limit. By then they'll be able to make bigger chips and have transistor budgets in the billions.
>
> — Gordon Moore, 2006

1.1 Background

Continuum topology optimization (TO) has emerged as a design methodology, enabling the identification of optimal shapes and material distributions to optimize specified objectives and satisfy constraints across variables fields of physics and engineering. These applications span multiscale mechanics [82], fluid mechanics [14, 26, 42, 52], electromagnetics [39, 55], photonics [51, 20], and coupled multi-physics problems [22, 41, 68, 44]; as well as automotive [19] and aerospace industries [87]. Since its introduction [11], density based methods, including Solid Isotropic Material with Penalty (SIMP) method [10], Evolutionary Structure Optimization (ESO) method [34], or topological representative methods like Level Set method [77] and Phase Field method [71] have proven their versatility in addressing the diverse physical phenomena and manufacturing constraints encountered.

To expand the design space and enhance the precision of the resulting designs, the adoption of large-scale simulations has become essential. Among the available techniques, the density based method is predominantly favored [1, 48] for its simplicity in avoiding additional partial

differential equations (PDEs) computation, inherent in methods like Level Set (LS), and for its straightforward implementation conducive to scalability across different computational architectures [48, 74].

1.2 Acceleration via Classical Computing

According to the Moore's law [62], the capacity of the chip can be doubled every 18 months. However, the increase in the number of transistors on a chip does not necessarily translate to a proportional increase in the performance of the chip. The performance of the chip is also limited by the power consumption and the heat generated by the chip. The increase of performance with respect to a single chip has been slowing down in recent years. To overcome this limitation, the parallel computing [38] has been proposed to sustain the scalability in scientific computing. In recent years, as the quick advance of machine learning, general-purpose computing on graphics processing units (GPUs) [24] has becoming a popular option for the acceleration of classical computing as well [70, 46].

Many efforts have been paid for the acceleration of continuum TO problem via classical computing. Both the parallel computing based on CPU clusters [2, 48] or based on GPUs [74] has been studied in recent years. These efforts mainly focuses on applying the SIMP method onto modern computing clusters, both including the implementation to reduce the cost of solving the governing equations via parallel computing and the adoption of the classical optimizer onto modern devices [3].

1.3 Acceleration via Quantum Computing

Rather than a law of physics, Moore's law is only an empirical observation and the advancement of the semiconductor industry has already slowed down since around 2010 [50]. The quantum computing [66] has been

proposed as a potential alternative to overcome the limitation of the classical computing. The quantum computing is based on the principles of quantum mechanics, which allows the quantum computer to perform calculations that are impossible for classical computers. The quantum computer can perform calculations in parallel and can solve certain problems much faster than classical computers. The quantum computer has the potential to achieve significant speedup on many classical computational tasks [30, 27] or to solve problems that are currently intractable for classical computers [29].

Only very recently, people have paid attention on utilizing the quantum computing for solving TO problems [79]. Most of these efforts [78, 61, 81] has been paid onto the truss systems. Instead, this thesis focuses on solving large-scale continuum TO problems discretized by a mesh.

1.4 Outline

This thesis proposes a new multi-cuts framework for solving the general continuum TO problem based on the mixed-integer nonlinear programming (MINLP). It can both harness the current classical computers and near-term quantum devices for the acceleration of finding the optimal layout for TO design tasks. The remainder of the thesis is organized as follows. Chapter 2 introduces the Discrete Variable Topology Optimization (DVTO) and the proposed multi-cuts formulation based framework for solving general linear elasticity problems. It uses 2D examples to demonstrate the capability of the proposed framework in reducing the number of evaluations of objective functions and quality of resulted optimal designs. Chapter 3 discusses the a meshless discretization framework for solving complex PDE constrained problems and the Geometric Multigrid Method (GMG) for accelerating the evaluation of complex governing equations. Chapter 4 proposes an accelerated optimizer for solving large-scale TO

problems. Chapter 5 discusses the quantum based approaches for accelerating the solving of TO problems.

2 DISCRETE VARIABLE TOPOLOGY OPTIMIZATION

The conventional density based method, e.g. the Solid Isotropic Material with Penalization (SIMP) method, conceptualizes the continuum TO problem as a nonlinear integer optimization challenge, employing interpolation schemes to soften the binary design variables into a continuum. Consequently, the issue transforms into a solvable nonlinear optimization problem, typically navigated via sophisticated nonlinear optimizers such as the Method of Moving Asymptotes (MMA) [69]. Additionally, the Heaviside projection technique is instrumental in nudging intermediate outcomes towards binary solutions, ensuring a controlled transition between solid and void states and mitigating optimization oscillations inherent to the problem's discontinuous nature. Nevertheless, the choice of an interpolation scheme is critical and context-specific, often influencing the optimization's efficacy and accuracy. A notable illustration [88, 49] is the application of the SIMP method in scenarios involving multiple materials under mass constraints, where the choice of interpolation can inadvertently introduce new local minima, potentially diverting the optimization from identifying the true coexistence conditions of different materials.

Other than using the continuous relaxation to formulate a well-posed TO problem, the continuum TO problem can also be formulated as a discrete optimization problem. Directly dealing with the discrete nature of the TO problem also has a long history. Early attempts like [58, 9] tried to use sequential integer programming to find the optimal material layouts. Methods like Bidirectional Evolutionary Structure Optimization (BESO) method [35] and Topology Optimization of Binary Structures (TOBS) [57], have already demonstrated their success in finding optimal material configuration as heuristic approaches. Recently, Liang et al. [47] tried to use canonical relaxation to bypass the linear binary programming

problem while using the sequential integer programming to preserve the discrete feature of the original TO problem.

In this thesis, a novel framework on top of discrete programming is established in this chapter. Based on the results on numerical experiments, it is expected to deliver a better performance compared to the conventional density based method and many other methods based on discrete variables formulations. By taking advantage of the proposed framework, the number of iterations required to reach the optimality condition can be greatly reduced and the design of problem-specific interpolation schemes, e.g. the ordered SIMP interpolation [88], can be avoided. Based on the theoretical analysis, the introduction of the adaptive trust region method has demonstrated the superior performance for ensuring the desired fewer number of iterations accompanying with the parameter relaxation scheme for controlling the reduction of parameters in the design challenge.

The chapter is organized as follows: the formal statement of the continuum TO problem with respect to the discrete variables is formulated in Section 2.1; all of the design objectives discussed in this thesis is introduced in Section 2.2 in Section 2.3, the multi-cuts formulation used for solving the defined continuum TO problem is established; a parameter relaxation scheme which can mitigate the ill-conditioning of the TO problem is discussed in Section 2.4; finally, the numerical examples in 2D design space are presented in Section 2.5 to demonstrate the effectiveness of the proposed framework.

2.1 Problem Statement

Continuum TO optimizes the material layout $\rho(\mathbf{x})$ within a continuum bounded design domain $\Omega \in \mathbb{R}^d$ (where $d = 2$ or 3). The continuum TO problem is constrained by a set of partial differential equations (PDEs) with respect to a set of field variables $\mathbf{u}(\mathbf{x})$ subject to prescribed boundary

conditions (BCs) which together describe the physical laws governing the underlying problem, as well as the target material usage limit and local constraints on the field variables or design variables, etc. In a formal way, the problem can be stated as:

$$\begin{aligned}
\min_{e} \quad & \int_{\Omega} F[\mathbf{u}(\mathbf{x}, \rho(\mathbf{x})), \rho(\mathbf{x})] d\Omega \\
\text{s.t.} \quad & \mathcal{L}(\mathbf{u}(\rho(\mathbf{x}))) + \mathbf{b} = \mathbf{0} && \forall \mathbf{x} \in \Omega \\
& \mathbf{u}(\mathbf{x}, \rho(\mathbf{x})) = \mathbf{u}_{\Gamma}(\mathbf{x}) && \forall \mathbf{x} \in \Gamma_D \\
& \mathbf{n} \cdot \nabla \mathbf{u}(\mathbf{x}, \rho(\mathbf{x})) = \mathbf{h}_{\Gamma}(\mathbf{x}) && \forall \mathbf{x} \in \Gamma_N \\
& G_{i_G}(\mathbf{u}(\mathbf{x}, \rho(\mathbf{x}))) \leqslant 0, && i_G = 1, \dots n_G \\
& H_{i_H}(\rho(\mathbf{x})) \leqslant 0, && i_H = 1, \dots, n_H \, ,
\end{aligned} \tag{2.1}$$

where $\mathbf{u}$ denotes the field variable defined in the design domain Ω; ρ is the design variable, denoting the material usage; $\mathcal{L}$ denotes some linear differential operator; $\mathbf{b}$ is the source term; and $\Gamma = \partial\Omega$ represents the boundary of the design domain with Γ_D denoting the boundary where Dirichlet BC is imposed and Γ_N the boundary where Neumann BC is enforced. In addition to governing equations and the corresponding BCs, it considers other two types of inequality constraints: G_{i_G} denotes the constraints applied on the field variable $\mathbf{u}$ or the derived variables from $\mathbf{u}$; H_{i_H} denotes the linear constraints only exerted on design variables $\boldsymbol{\rho}$. A typical example of G_{i_G} would be the local stress constraint which controls the maximum value of local stresses; one typical example of H_{i_H} would be the volume constraints which limits the total volume of material can be utilized within the design domain Ω.

There are two general approaches can be applied to solve the continuum TO problem as in Equation (2.1): the first approach optimizes the problem and follows with discretization of the continuum problem; the second approach discretizes the design domain and then optimizes the

discretized problem. The first approach generally utilizes the level set method to implicitly represent the material layout and utilizes the topological derivatives to move forward in the optimization iterations [77]. The second approach, which is generally denoted as density method [6], utilizes the sensitivity analysis to find routes for the optimal solution in iterations. The second approach is followed in this report and the density method is examined in the discrete programming setup to solve a series of topology optimization problems under the physics governed by linear elasticity and exerted with different kinds of constraints. The design domain Ω is first discretized into rectangular elements in 2D or hexahedron elements in 3D. The finite element method (FEM) is commonly utilized to discretize the governing equations in (2.1). The resulting discretized nonlinear optimization problem can be generally written as follows:

$$\begin{aligned} \min_{\mathbf{u},\boldsymbol{\rho}} \quad & f(\mathbf{u},\boldsymbol{\rho}) \\ \text{s.t.} \quad & \mathbf{K}(\boldsymbol{\rho})\mathbf{u} = \mathbf{f} \\ & G_{i_G}(\mathbf{u}(\mathbf{x},\rho(\mathbf{x}))) \leqslant 0, \qquad i_G = 1,\dots n_G \\ & H_{i_H}(\boldsymbol{\rho}) \leqslant 0, \qquad i_H = 1,\dots,n_H \\ & \mathbf{u} \in \mathbb{R}^{n_u}, \boldsymbol{\rho} \in \{0,1\}^{n_e}, \end{aligned} \tag{2.2}$$

where $\mathbf{f}$ is a known discretized BCs and source terms; $\mathbf{u}$ is the discretized field variable defined on the vertices of the mesh; and ρ_e is the discretized design variable defined at the centroid of each mesh element e; $\mathbf{K}(\boldsymbol{\rho})$ is the discretized linear operator. In this thesis, it only considers the constraints $H_{i_H}(\boldsymbol{\rho})$ that directly applied on the design variables and it does not consider the constraints $G_{i_G}(\mathbf{u}(\mathbf{x},\rho(\mathbf{x})))$ that applied on the field variable $\mathbf{u}$.

2.1.1 Linear Elasticity Problem

In this thesis, it is focused on the linear elasticity problems as the underlying physics for TO problems. The governing equation is written as:

$$\begin{aligned} \nabla \cdot \boldsymbol{\sigma}(\mathbf{u}) + \mathbf{b} &= \mathbf{0} \quad \text{in} \quad \Omega \,, \\ \boldsymbol{\sigma}(\mathbf{u}) \cdot \mathbf{n} &= \mathbf{h}_\Gamma \quad \text{on} \quad \Gamma_N \,, \\ \mathbf{u} &= \mathbf{u}_\Gamma \quad \text{on} \quad \Gamma_D \,, \end{aligned} \tag{2.3}$$

where $\boldsymbol{\sigma}(\mathbf{u}) = \mathbb{C} : \boldsymbol{\varepsilon}(\mathbf{u})$ is the stress tensor; $\mathbf{b}$ is the body force; $\mathbb{C}$ is the fourth-order elasticity tensor; and $\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^\intercal)$ is the strain tensor. The elasticity tensor of plane stress in 2D is written as:

$$\mathbb{C} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} , \tag{2.4}$$

and in 3D is written as:

$$\mathbb{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} , \tag{2.5}$$

where E is the Young's modulus and ν is the Poisson's ratio. n_M candidate materials can be utilized in the design domain Ω and the material properties are different for different materials, with prescribed Young's modulus E_i and Poisson's ratio ν_i for all n_M candidate materials. The discrete material optimization scheme is used to evaluate the elasticity

tensor on each element e based on the design variable $\rho_{e,m}$ as:

$$\mathbb{C}_e = \sum_{m=1}^{n_M} \rho_{e,m} \mathbb{C}_m \, , \tag{2.6}$$

where $\mathbb{C}_m$ is the elasticity tensor for the m-th material defined as:

$$\mathbb{C}_m = \frac{E_m}{1-\nu_m^2} \begin{bmatrix} 1 & \nu_m & 0 \\ \nu_m & 1 & 0 \\ 0 & 0 & \frac{1-\nu_m}{2} \end{bmatrix} \, , \tag{2.7}$$

in 2D or

$$\mathbb{C}_m = \frac{E_m}{(1+\nu_m)(1-2\nu_m)} \begin{bmatrix} \mathbf{C}_m & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \frac{1-2\nu_m}{2}\mathbf{I}_{3\times3} \end{bmatrix} \, , \tag{2.8}$$

in 3D with

$$\mathbf{C}_m = \begin{bmatrix} 1-\nu_m & \nu_m & \nu_m \\ \nu_m & 1-\nu_m & \nu_m \\ \nu_m & \nu_m & 1-\nu_m \end{bmatrix} \, .$$

When using the finite element method to discretize the governing equation in (2.3), the nodal displacement $\mathbf{u}$ is considered as the field variable in the optimization problem. $\mathbf{u}_e$ denotes the nodal displacement on the element e and the nodal strain $\boldsymbol{\varepsilon}_e$ is calculated based on the nodal displacement as:

$$\boldsymbol{\varepsilon}_e = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \end{bmatrix} = \mathbf{B}\mathbf{u}_e \, , \tag{2.9}$$

in 2D or

$$\boldsymbol{\varepsilon}_e = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial y} & \frac{\partial w}{\partial z} & \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} & \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} & \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \end{bmatrix} = \mathbf{B}\mathbf{u}_e \, , \tag{2.10}$$

in 3D, where $\mathbf{B}$ is the strain-displacement matrix. The elemental stiffness

matrix $\mathbf{K}_e$ is calculated based on the elemental elasticity tensor $\mathbb{C}_e$ as:

$$\mathbf{K}_e = \int_{\Omega_e} \mathbf{B}^\intercal \mathbb{C}_e \mathbf{B} d\Omega \ , \tag{2.11}$$

where

$$\mathbb{C}_e = \sum_{m=1}^{n_M} \rho_{e,m} \mathbb{C}_m \ . \tag{2.12}$$

The global stiffness matrix $\mathbf{K}$ is assembled based on the elemental stiffness matrix $\mathbf{K}_e$ as:

$$\mathbf{K} = \mathbf{K}_0 + \sum_{e=1}^{n_e} \mathbf{K}_e \ , \tag{2.13}$$

where $\mathbf{K}_0$ is the stiffness matrix assembled from void elements infilling the entire design domain with the elasticity tensor $\mathbb{C}_0$ formed from $E_0 = \varepsilon \max_m(E_m)$ and $\nu_0 = 0.3$. With the inclusion of $\mathbf{K}_0$, it is guaranteed that the global stiffness matrix $\mathbf{K}$ is symmetric positive definite. To minimize the influence from the $\mathbf{K}_0$, ε is selected as a small constant value, e.g. 10^{-4}. The nodal displacement $\mathbf{u}$ is calculated by solving the linear system $\mathbf{Ku} = \mathbf{f}$.

2.1.2 Volume Constraint

One of the most commonly used constraints with respect to the design variables $\boldsymbol{\rho}$ is the volume constraint:

$$H_V(\boldsymbol{\rho}) = \sum_{i=1}^{n_e} \frac{1}{n_e} \rho_i \leqslant V_T \ , \tag{2.14}$$

where it assumes the underlying discretization of the computational domain Ω with a uniform square/cubic mesh and V_T denotes the target volume fraction in the resulting material configuration. If the computational domain Ω is discretized with a non-uniform mesh, the volume

constraint can be generalized as:

$$H_V(\boldsymbol{\rho}) = \sum_{i=1}^{n_e} V_i \rho_i \leqslant V_T \ , \tag{2.15}$$

2.1.3 Mass Constraint

By generalizing the volume constraint in (2.14), the total mass constraint can be utilized when multiple candidate materials are included in the design problem and ready for selection. The total mass constraint is defined as:

$$\begin{aligned} H_{M_0}(\boldsymbol{\rho}) &= \sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \frac{\bar{M}_m}{n_e} \rho_{e,m} \leqslant \bar{M}_{max} \\ H_{M_e}(\boldsymbol{\rho}) &= \sum_{m=1}^{n_M} \rho_{e,m} \leqslant 1, \quad e = 1, 2, \ldots, n_e \ , \end{aligned} \tag{2.16}$$

where $\bar{M}_{max}$ is the maximum allowed mass of the structure. This constraint contains two components: the first line denotes that the total mass of the resulting structure is no greater than the allowed maximum mass; the second line denotes that each element can only be a void element or take at most a single material from the candidates. Different from many other methods in literature [33, 49], there is no prior assumption that the candidate materials have to been sorted in any specific order.

2.1.4 Sensitivity Analysis and Filtering

In this thesis, it is targeting to propose a new gradient based optimization framework. Before getting into the detailed derivation of the proposed framework, several useful gradients or sensitivities are calculated in advance. First, the gradient of the elemental stiffness matrix $\mathbf{K}_e$ with respect

to the design variable $\rho_{e,m}$ is calculated as:

$$\frac{\partial \mathbf{K}_e}{\partial \rho_{e,m}} = \int_{\Omega_e} \mathbf{B}^\intercal \frac{\partial \mathbb{C}_e}{\partial \rho_{e,m}} \mathbf{B} d\Omega = \int_{\Omega_e} \mathbf{B}^\intercal \mathbb{C}_m \mathbf{B} d\Omega = \mathbf{K}_{e,m} \ . \tag{2.17}$$

To evaluate the sensitivity of the objective function with respect to the design variable $\rho_{e,m}$, the Lagrangian of the optimization problem is first defined as:

$$\mathcal{L}(\mathbf{u}, \boldsymbol{\rho}, \boldsymbol{\mu}) = f(\mathbf{u}, \boldsymbol{\rho}) + \boldsymbol{\mu}^\intercal (\mathbf{K}(\boldsymbol{\rho})\mathbf{u} - \mathbf{f}) \ , \tag{2.18}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{n_u}$ is the adjoint variable. According to the Karush–Kuhn—Tucker (KKT) conditions [54], the adjoint variable $\boldsymbol{\mu}$ takes the form:

$$\boldsymbol{\mu} = -\mathbf{K}^{-1} \frac{\partial f}{\partial \mathbf{u}} \ , \tag{2.19}$$

With the adjoint variable $\boldsymbol{\mu}$, the gradient of the objective function with respect to the design variable $\rho_{e,m}$ is calculated as:

$$\begin{aligned} \frac{\partial f}{\partial \rho_{e,m}} =& \left(\frac{\partial f}{\partial \mathbf{u}} \right)^\intercal \cdot \frac{\partial \mathbf{u}}{\partial \rho_{e,m}} \\ =& - \left(\mathbf{K}^{-1} \frac{\partial f}{\partial \mathbf{u}} \right)^\intercal \cdot \frac{\partial \mathbf{K}}{\partial \rho_{e,m}} \mathbf{K}^{-1} \mathbf{f} \\ =& \boldsymbol{\mu}^\intercal \frac{\partial \mathbf{K}}{\partial \rho_{e,m}} \mathbf{u} \\ =& \boldsymbol{\mu}_e^\intercal \frac{\partial \mathbf{K}_e}{\partial \rho_{e,m}} \mathbf{u}_e \\ =& \boldsymbol{\mu}_e^\intercal \left(\int_{\Omega_e} \mathbf{B}^\intercal \mathbb{C}_m \mathbf{B} d\Omega \right) \mathbf{u}_e \end{aligned} \ , \tag{2.20}$$

where $\mathbf{u}_e$ and $\boldsymbol{\mu}_e$ are the field and the adjoint variable on the element e, respectively. This relation is valid for both of the convex and non-convex objective functions discussed above. Finally, the sensitivity of the objective

function with respect to the design variable $\rho_{e,m}$ can be calculated as:

$$w_{e,m} = \begin{cases} \dfrac{\partial f}{\partial \rho_{e,m}} & \text{if } \rho_{e,m} = 1 \\ \dfrac{2\dfrac{\partial f}{\partial \rho_{e,m}} \cdot \dfrac{\partial f}{\partial \rho_{e,m'}}}{\dfrac{\partial f}{\partial \rho_{e,m}} + \dfrac{\partial f}{\partial \rho_{e,m'}}} & \text{if } \rho_{e,m} = 0, \rho_{e,m'} = 1, m \neq m' \\ \boldsymbol{\mu}^{\intercal} \left(\displaystyle\int_{\Omega_e} \mathbf{B}^{\intercal} \mathbb{C}_0 \mathbf{B} d\Omega \right) \mathbf{u} & \text{if } \displaystyle\sum_{m=1}^{n_M} \rho_{e,m} = 0 \end{cases} . \tag{2.21}$$

It is known that differentiation of the objective function with respect to the binary design variable $\rho_{e,m}$ is not well-defined [67], especially for the case when $\rho_{e,m} = 0$. Therefore, in Equation (2.21), the sensitivity takes the gradient of the objective function value when $\rho_{e,m} = 1$ and it is regulated when $\rho_{e,m} = 0$. Two scenarios are considered for regulation: the first scenario is when $\rho_{e,m} = 0$ and $\rho_{e,m'} = 1$ for $m \neq m'$, when the element e is occupied by the material m', the sensitivity is calculated as the harmonic mean of the gradients of the two candidate materials; the second scenario is when $\sum_{m=1}^{n_M} \rho_{e,m} = 0$, indicating the case that the element e is assigned as a void element, the sensitivity is calculated as the sensitivity of the void material. The use of harmonic mean over the arithmetic mean is inspired from the Reuss model [60] in the composite material design. And the numerical experiments indicate that the harmonic mean is superior when selecting the optimal material in the design problem.

Similar to many other methods in literature [45, 47], no matter continuous or binary design variables, the sensitivity of the objective function with respect to the design variable $\rho_{e,m}$ is filtered to avoid the checkerboard pattern. Due to the sharp transition between different candidate materials and void elements, the Helmholtz filter, which expects a smooth transition, is not suitable for the binary design variables. In this thesis, the radius

filter with a prescribed size of r is utilized to smooth the sensitivity of the objective function with respect to the design variable $\rho_{e,m}$ as:

$$\hat{w}_{e,m} = \frac{\sum_{e' \in \mathcal{N}_e^r} h_{e,e'}(r) w_{e,m}}{\sum_{e' \in \mathcal{N}_e^r} h_{e,e'}(r)} , \tag{2.22}$$

with $h_{e,e'}(r) = \max(0, r - \|\mathbf{x}_e - \mathbf{x}_{e'}\|_2)$, following [57, 67].

2.2 Design Objectives

2.2.1 Minimum Compliance Problem

One of the most commonly used design objectives is the minimum compliance problem. The minimum compliance problem targets to minimize the compliance of the structure under the given loading conditions. The objective function of the the minimum compliance problem is defined as:

$$f_{MC}(\mathbf{u}, \boldsymbol{\rho}) = \mathbf{u}^T \mathbf{K} \mathbf{u} . \tag{2.23}$$

As $\mathbf{K}$ is symmetric positive definite, it is not difficult to find that

$$\begin{aligned} 0 =& \frac{\partial(\mathbf{K}\mathbf{K}^{-1})}{\partial \boldsymbol{\rho}} \\ =& \frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}} \mathbf{K}^{-1} + \mathbf{K} \frac{\partial \mathbf{K}^{-1}}{\partial \boldsymbol{\rho}} \end{aligned} . \tag{2.24}$$

It gives the gradient of the stiffness matrix $\mathbf{K}$ with respect to the design variable $\boldsymbol{\rho}$ as:

$$\frac{\partial \mathbf{K}^{-1}}{\partial \boldsymbol{\rho}} = -\mathbf{K}^{-1} \frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}} \mathbf{K}^{-1} . \tag{2.25}$$

With the gradient of the stiffness matrix $\mathbf{K}$, the gradient of the compliance function $f(\mathbf{u}, \boldsymbol{\rho})$ can be calculated as:

$$\begin{aligned} \frac{\partial f_{MC}}{\partial \boldsymbol{\rho}} =& \frac{\partial(\mathbf{f}^\intercal \mathbf{K}^{-1}\mathbf{f})}{\partial \boldsymbol{\rho}} \\ =& \mathbf{f}^\intercal \frac{\partial \mathbf{K}^{-1}}{\partial \boldsymbol{\rho}}\mathbf{f} \\ =& -\mathbf{u}^\intercal \frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}}\mathbf{u} \, . \end{aligned} \tag{2.26}$$

With the gradient, one can calculate the Hessian of the compliance function $f(\mathbf{u}, \boldsymbol{\rho})$ as:

$$\begin{aligned} \frac{\partial^2 f_{MC}}{\partial^2 \boldsymbol{\rho}} =& -\frac{\partial\left(\mathbf{u}^\intercal \frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}}\mathbf{u}\right)}{\partial \boldsymbol{\rho}} \\ =& -\frac{\partial\left(\mathbf{f}^\intercal \mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}}\mathbf{K}^{-1}\mathbf{f}\right)}{\partial \boldsymbol{\rho}} \\ =& -2\mathbf{f}^\intercal \frac{\partial \mathbf{K}^{-1}}{\partial \boldsymbol{\rho}} \cdot \frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}}\mathbf{K}^{-1}\mathbf{f} - \mathbf{f}^\intercal \mathbf{K}^{-1}\frac{\partial^2 \mathbf{K}}{\partial^2 \boldsymbol{\rho}}\mathbf{K}^{-1}\mathbf{f} \\ =& 2\mathbf{u}^\intercal \frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}}\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}}\mathbf{u} \, , \end{aligned} \tag{2.27}$$

where $\frac{\partial^2 \mathbf{K}}{\partial^2 \boldsymbol{\rho}}$ is eliminated as $\mathbf{K}$ is only a linear function of $\boldsymbol{\rho}$. Let $\mathbf{y} = \frac{\partial \mathbf{K}}{\partial \boldsymbol{\rho}}\mathbf{u}$, the Hessian of the compliance function $f(\mathbf{u}, \boldsymbol{\rho})$ can be further simplified as:

$$\frac{\partial^2 f_{MC}}{\partial^2 \boldsymbol{\rho}} = 2\mathbf{y}^\intercal \mathbf{K}^{-1}\mathbf{y} \, . \tag{2.28}$$

Since $\mathbf{K}$ is symmetric positive definite, $\mathbf{y}^\intercal \mathbf{K}^{-1}\mathbf{y}$ is always positive when $\mathbf{y}$ is not zero. It indicates that the compliance function $f(\mathbf{u}, \boldsymbol{\rho})$ is convex with respect to the design variable $\boldsymbol{\rho}$.

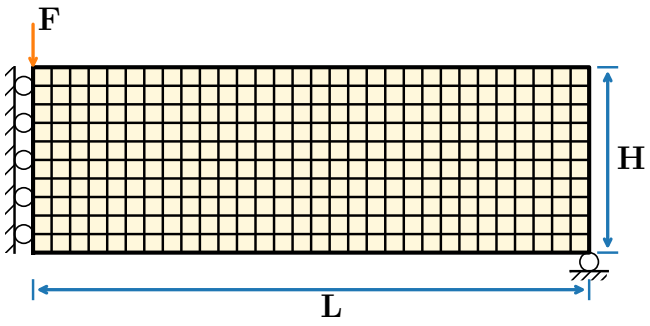
2.2.1.1 2D design space

In this thesis, two different settings in a rectangular design space is considered, as depicted in Figure 2.1. The first setting corresponds to the Messerschmitt-Bölkow-Blohm (MBB) beam problem, as illustrated in Figure 2.1a. In this thesis, it is designated in a rectangular domain with a length-height ratio of $L : H = 3 : 1$ if without any further notification. The left-side edge of the rectangular domain is constrained in x (horizontal) direction but can freely move along y (vertical) direction. The movement of the bottom-right corner is restricted solely along x-axis. An external force $\mathbf{F}_y = -1$ is exerted at the top-left corner of the design domain. The second setting represents the design of a cantilever, defined in a rectangular domain with a length-height ratio of $L : H = 2 : 1$, as shown in Figure 2.1b. The left side of the rectangular domain is constrained in both x and y axes.

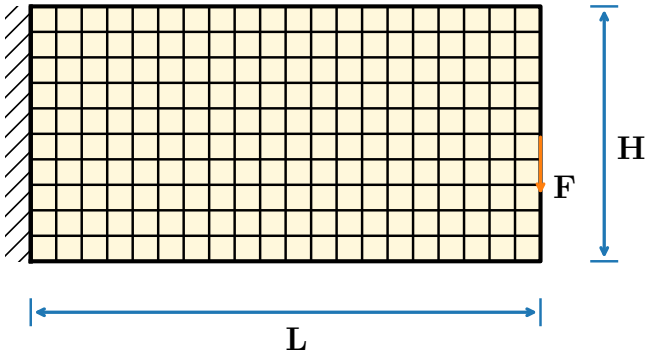
Up to five candidate materials can be considered in the 2D design space, as listed in Table 2.1. All of the materials are using the normalized Young's modulus or density. The first material set contains five materials with Young's modulus E_m ranging from 0.4 to 1.0. The second material set contains five materials with Young's modulus E ranging from 0.43 to 1.0. The normalized densities $\bar{M}_m$ are ranging from 0.3 to 1.0. The main difference the two material sets is the ratio between the Young's modulus and the density. This ratio basically determines the selection of the optimal material in the design problem, especially for the minimum compliance problem as discussed in [33, 49]. The two material sets are designed to demonstrate the capability of the proposed method in selecting the optimal combination of materials from the candidate set in the design problem.

2.2.1.2 3D design space

In this thesis, the design space in 3D is a box with the length of $L = 40$ cm, the width of $W = 10$ cm, and the height of $H = 10$ cm, as depicted



(a) MBB



(b) Cantilever

Figure 2.1: Minimum compliance: design domain and boundary conditions for two different problems.

Table 2.1: 2D design space: Properties of the five candidate materials.

	MAT	1	2	3	4	5
Material Set 1	E_m	0.4	0.7	0.85	0.9	1.0
	$\bar{M}_m$	0.3	0.5	0.65	0.8	1.0
	$E_m/\bar{M}_m$	1.33	1.4	1.31	1.13	1.0
Material Set 2	E	0.43	0.7	0.85	0.94	1.0
	$\bar{M}_m$	0.3	0.5	0.65	0.8	1.0
	$E_m/\bar{M}_m$	1.43	1.4	1.31	1.175	1.0

in Figure 2.2. On top of the box, the design space is assigned to plate with a height of 0.4, always assigned as solid elements with the strongest candidate material, as indicated by the red region in the illustration. At the bottom surface, the design space is fixed in the region where y is greater than 38 or less than 2, as illustrated as the orange regions in Figure 2.2. All the other region rendered in gray is the free design space that can be filled any candidate materials. A uniform distributed force of -0.1 is applied on the top surface of the design space as rendered in blue arrows in Figure 2.2.

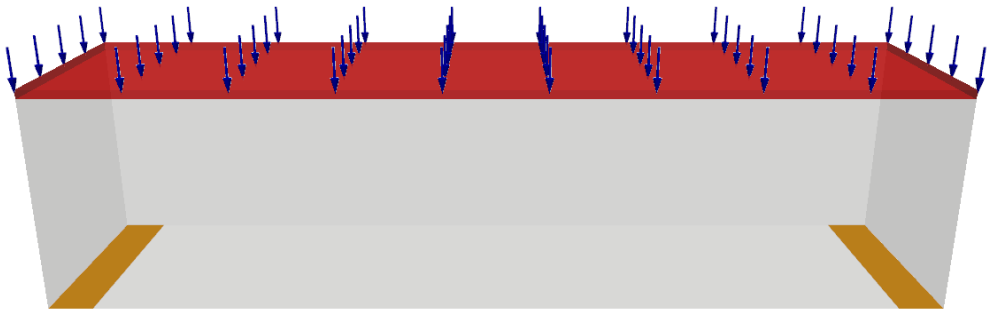


Figure 2.2: Minimum compliance: design domain and boundary conditions for the bridge design problem in 3D.

Up to four candidate metallic materials can be considered in the 3D design space, as listed in Table 2.2. The four materials are selected from

the most commonly used metallic materials in the engineering field. The first material is magnesium, which is the lightest metallic material with a relatively low Young's modulus. The second material is aluminum, which is the most commonly used metallic material in the engineering field. The third material is titanium, which is a high-strength metallic material with a high Young's modulus. The fourth material is stainless steel, which is a high-strength metallic material with a high Young's modulus and a high density. The four materials are selected to demonstrate the capability of the proposed method in selecting the optimal material in the design problem.

Table 2.2: 3D design space: Properties of the four candidate materials.

Materials	**Magnesium**	**Aluminum**	**Titanium**	**Stainless steel**
Young's modulus E_m (GPa)	44	73	100	210
Poisson's ratio ν_m	0.28	0.33	0.36	0.29
Mass density $\bar{M}_m(10^3 kg/m^3)$	1.74	2.70	4.50	7.80

2.2.2 Compliant Mechanism Problem

The compliant mechanism problem targets to maximize the displacement at specific locations with respect to prescribed directions under the given loading conditions. The objective function of the compliant mechanism problem is defined as:

$$f_{CM}(\mathbf{u}, \boldsymbol{\rho}) = \sum_{i=1}^{n} \mathbf{n}_i^T \mathbf{u}_i \ . \tag{2.29}$$

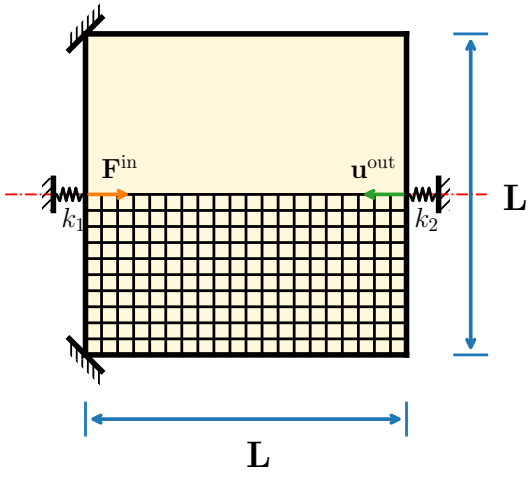
Different from the compliance function, the objective function of the compliant mechanism problem is a non-convex problem with respect to the design variable $\boldsymbol{\rho}$. By solving the compliant mechanism problem, it is expected to demonstrate that the proposed method can be applied to generalized TO problems.

The design domain and the associated BCs are depicted in Figure 2.3, where the domain is a square with each side measuring L, and the top-left and bottom-left corners are fixed in place. The domain is also connected to two springs each at the middle of the left or right side, respectively. The stiffness coefficient of both springs is $k_1 = k_2 = 0.1$. In addition, an input force $\mathbf{F}_x^{in} = 1$ is applied at the midpoint of the left side. The TO design herein aims to maximize the displacement along x-direction at the midpoint of the right side, i.e., minimizing $\mathbf{u}_x^{out}$ as denoted in Figure 2.3. Due to the problem's axial symmetry around the center line (as highlighted in red dotted-dash in Figure 2.3), only half of the design domain needs to be considered in the actual solution procedure and hence was discretized with a uniform mesh. The target volume fraction is $V_T = 0.3$. The material parameters are specified as: Young's modulus $E_1 = 1.0$; Poisson's ratio $\nu = 0.3$. The convergence tolerance required in Algorithm 1 was set as $\varepsilon = 5 \times 10^{-3}$.

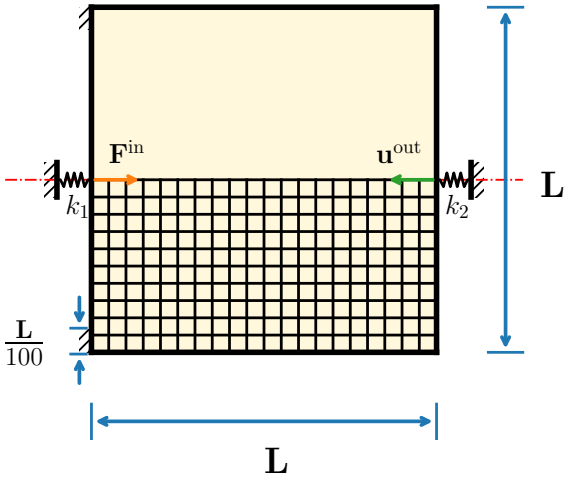
2.3 Multi-cuts Formulation

As the gradient of a general objective function is non-zero as shown in Equation (2.26), the TO problem is commonly seen as a nonlinear optimization problem with respect to the binary design variables $\boldsymbol{\rho}$. Directly solving a nonlinear binary optimization problem is intractable due to the combinatorial nature of the problem. In this section, a multi-cuts formulation based framework to solve the TO problem with a general linear elasticity governing equation is proposed. The proposed framework is inspired from the Generalized Benders' Decomposition (GBD) method. By incorporating with the trust region method, the proposed framework can solve the TO problem with a general linear elasticity design problem efficiently and robustly.

The framework based on the GBD method to solve the continuum



(a) Scenario 1



(b) Scenario 2

Figure 2.3: Compliant mechanism: Design domain and boundary conditions.

TO problems is first discussed in Section 2.3.1. Next, the trust region constraints are introduced to the framework for the extension to the non-convex problems, e.g. compliant mechanism design problems in Sec-

tion 2.3.2. An adaptive trust region radius adjustment scheme is presented in Section 2.3.3 to improve the robustness of the optimization procedure. Finally, the details of the algorithm implementation are summarized in Section 2.3.4.

2.3.1 Generalized Benders' Decomposition

Inspired by the early attempt of the Generalized Benders' Decomposition method on the TO problem with a truss system [53], the continuum TO problem can be addressed by the GBD method through forming the following primal problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & f_{MC}(\mathbf{u}, \boldsymbol{\rho}^k) \\ \text{s.t.} \quad & \mathbf{K}(\boldsymbol{\rho}^k)\mathbf{u} = \mathbf{f} \\ & \mathbf{u} \in \mathbb{R}^{n_u} , \end{aligned} \tag{2.30}$$

and master problem:

$$\begin{aligned} \min_{\boldsymbol{\rho},\eta} \quad & \eta \\ \text{s.t.} \quad & f_{MC}(\mathbf{u}^j, \boldsymbol{\rho}^j) + \sum_{e=1}^{n_e}\sum_{m=1}^{n_M} \hat{w}^j_{e,m}(\rho_{e,m} - \rho^j_{e,m}) \leqslant \eta, \quad j = 1, \dots, k \\ & H_{i_H}(\boldsymbol{\rho}) \leqslant 0, \quad i_H = 1, \dots, n_H \\ & \boldsymbol{\rho} \in \{0,1\}^{n_e \times n_M} , \end{aligned} \tag{2.31}$$

as well when the design challenge is the minimization of the compliance of the structure and is subject to the volume constraint or total mass constraint. In the master problem (2.31), $\widetilde{f}^j_{MC}(\boldsymbol{\rho}) = f_{MC}(\mathbf{u}^j, \boldsymbol{\rho}^j) + \sum_{e=1}^{n_e}\sum_{m=1}^{n_M} \hat{w}^j_{e,m}(\rho_{e,m} - \rho^j_{e,m})$ is used to form a lower estimator to the objective function f_{MC} while satisfying the underlying governing equation $\mathbf{K}(\boldsymbol{\rho})\mathbf{u} = \mathbf{f}$ at the point $(\mathbf{u}^j, \boldsymbol{\rho}^j)$. Therefore, $\widetilde{f}^j_{MC}(\boldsymbol{\rho})$ is considered as a cut to the original optimization problem (2.1).

2.3.2 Trust Region Method

However, the GBD framework can only works for convex problems, e.g. minimum compliance problem with the objective function f_{MC}, which greatly limits the application of the proposed method. One of the common way to extend the gradient-based methods to solving a general non-convex problem is utilizing the trust region method. Additionally, due to the underlying governing PDEs in (2.1) is generally an ill-conditioned problem, the introduction of trust region method can also improve the robustness of the optimization procedure in TO problems. As the trust region is only required within the master problem and only required to be applied on the binary design variables, the trust region constraint can be constructed about any fixed point $\boldsymbol{\rho}^j$ with a size d^j in a L_2 norm as:

$$t^j(\boldsymbol{\rho}) = \frac{1}{n_e}\sum_{e=1}^{n_e}\left|\sum_{m=1}^{n_M}\rho_{e,m} - \sum_{m=1}^{n_M}\rho_{e,m}^j\right|^2 \leqslant d^j\ . \tag{2.32}$$

Since any given element can take at most one solid material (as stated by the total mass constraint H_{M_e} in (2.16)), the bilinear term $\rho_{e,m}\rho_{e,m'}$ with $m \neq m'$ derived from $(\sum_{m=1}^{n_M}\rho_{e,m})^2$ is always zero. As a result, the trust region extended to multi-material scenarios remains a linear constraint with respect to the design variable $\rho_{e,m}$, as can be seen from:

$$\begin{aligned}&\sum_{e=1}^{n_e}\left|\sum_{m=1}^{n_M}\rho_{e,m} - \sum_{m=1}^{n_M}\rho_{e,m}^j\right|^2\\ =&\sum_{e=1}^{n_e}\left(1-2\sum_{m=1}^{n_M}\rho_{e,m}^j\right)\left(\sum_{m=1}^{n_M}\rho_{e,m}\right) + \sum_{e=1}^{n_e}\left(\sum_{m=1}^{n_M}\rho_{e,m}^j\right)^2\ .\end{aligned} \tag{2.33}$$

If there are multiple linear approximations (cuts) to the general objec-

tive function f:

$$\widetilde{f}^{j}(\boldsymbol{\rho}) = f(\mathbf{u}^{j}, \boldsymbol{\rho}^{j}) + \sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \hat{w}_{e,m}^{j}(\rho_{e,m} - \rho_{e,m}^{j}), \quad j = 1, \ldots, k\,, \tag{2.34}$$

each within the neighborhood of $\boldsymbol{\rho}^{j}$, are found, a piece-wise linear approximation to the objective function $f(\mathbf{u}, \boldsymbol{\rho})$ is obtained and can be represented by introducing a new set of binary variables $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_k\}$ as:

$$\begin{aligned} &\widetilde{f}^{j}(\boldsymbol{\rho})\alpha_j - \Pi(1 - \alpha_i) \leqslant \eta, \quad j = 1, \ldots, k \\ &t^{j}(\boldsymbol{\rho})\alpha_j \leqslant d, \quad j = 1, \ldots, k \\ &\sum_{j=1}^{k} \alpha_j \geqslant 1, \quad \boldsymbol{\alpha} \in \{0, 1\}^{k}\,, \end{aligned} \tag{2.35}$$

where Π is a large positive penalty real number. α_j is introduced to control the validity of the linear approximation $\widetilde{f}^{j}(\boldsymbol{\rho})$: when $\alpha_i = 1$ (referred to as an active cut), it indicates that the linear approximation $\widetilde{f}^{j}(\boldsymbol{\rho})$ holds and the solution locates within the trust region $t^{j}(\boldsymbol{\rho})$; when $\alpha_j = 0$ (referred to as an inactive cut), it implies that the linear approximation is invalid, because the solution can be too far away from $\boldsymbol{\rho}^{j}$, and the linear approximation is not accurate enough.

The master problem can then be rewritten as

$$\begin{aligned}
\min_{\boldsymbol{\rho},\boldsymbol{\alpha},\eta} \quad & \eta \\
\text{s.t.} \quad & \widetilde{f}^{j}(\boldsymbol{\rho})\alpha_j - \Pi(1-\alpha_j) \leqslant \eta, \quad j = 1,\dots,k \\
& t^{j}(\boldsymbol{\rho})\alpha_j \leqslant d^{j}, \quad j = 1,\dots,k \\
& H_{i_H}(\boldsymbol{\rho}) \leqslant 0, \quad i_H = 1,\dots,n_H \\
& \sum_{j=1}^{k} \alpha_j \geqslant 1 \\
& \boldsymbol{\alpha} \neq \boldsymbol{\alpha}^{j}, \quad j = 1,\dots,k-1 \\
& \boldsymbol{\rho} \in \{0,1\}^{n_e \times n_M}, \boldsymbol{\alpha} \in \{0,1\}^{k}\,.
\end{aligned} \tag{2.36}$$

All optimal solutions are rejected that have already appeared in the iteration steps $j = 1,\dots,k-1$ by considering the constraint $\boldsymbol{\alpha} \neq \boldsymbol{\alpha}^{j}$, as repeated results won't help finding the convergent optimal solutions.

The master problem (2.36) is a bi-linear integer programming problem, which requires branch-and-bound methods to solve. In §4.1, a new algorithm is proposed to solve the master problem (2.36) efficiently.

2.3.3 Adaptive Trust Region Radius Adjustment

Employing appropriate trust-region radii in Equation (2.32) is crucial for ensuring both solution accuracy and efficiency. Thus, it can be advantageous to vary it adaptively during the solution process when solving the master problem (2.36). By leveraging a merit function as given by [86]:

$$\omega^{k} = \min_{j \in \mathcal{P}_{s^*}(k)} \left(\frac{f(\mathbf{u}^{j}, \boldsymbol{\rho}^{j}) - f(\mathbf{u}^{k}, \boldsymbol{\rho}^{k})}{f(\mathbf{u}^{j}, \boldsymbol{\rho}^{j}) - \eta^{k}} \right), \tag{2.37}$$

the trust-region radius d^k can be updated at each iteration step k, following the rule:

$$d^k = \begin{cases} \max(\theta_1 d^*, d_{\min}), & 0 \leqslant \omega^k < 1 \\ \min(\theta_2 d^*, d_{\max}), & \omega^k \geqslant 1 \\ \max(0.5 d^*, d_{\min}), & \omega^k < 0 \end{cases} \tag{2.38}$$

with $d^* = \min_{j \in \mathcal{P}_{s^*(k)}}(d^j)$. Here, θ_1 and θ_2 are the adjustment (shrinking/enlarging) factors for the trust-region radius and are assigned as 0.7 and 1.5, respectively, in all numerical tests. And it adopts $d_{\min} = 10^{-3}$ (sufficiently small) and $d_{\max} = 0.6$ (sufficiently large) in this paper.

The merit function defined in Equation (2.37) evaluates the ratio between the exact reduction in the objective function and the reduction estimated from problem (2.36). The desired trust-region radius d^k renders the merit function satisfying $\omega^k \geqslant 1$. Thus, if $\omega^k \geqslant 1$, d^k may be enlarged, for which a magnification factor 1.5 is employed to enlarge the trust-region radius for the newly added k-th cut for the next iteration step. Conversely, if $0 \leqslant \omega^k < 1$, the exact reduction in the objective function is smaller than the one estimated from problem (2.36), and hence d^k needs to be decreased. Here, a shrinking factor 0.7 is employed to reduce the trust-region radius for the k-th cut. However, if $\omega^k < 0$, it implies that the radius is too large that the searching goes to the wrong direction. Hence, a more aggressive shrinkage for the trust-region radius is demanded, and here it empirically chooses a factor of 0.5.

2.3.4 Algorithm Implementation

The iterations of solving the primal problem (2.30) and the master problem (2.36) yield a series of upper bounds and lower bounds, respectively, for the objective function in the original problem (2.2). The gap between the lowest upper bound $U = \min_j(f(\mathbf{u}^j, \boldsymbol{\rho}^j))$ with $j \leqslant k$ and the lower bound η^k should be minimized until reaching $\frac{|\eta^k - U|}{|U|} < \varepsilon$ (with ε the

preset tolerance) to terminate the optimization iterations. Additionally, the criterion $\eta^k > U$ is enforced, meaning the iterations should stop once the lower bound exceeds the upper bound, i.e., when the master problem (2.36) finds a solution greater than U. The entire framework proposed for solving the TO problem in Eq. (2.2) is summarized in Algorithm 1.

Algorithm 1 TOSolver($\boldsymbol{\rho}^0$)

Input: Initial material configuration $\boldsymbol{\rho}^0$
Output: Optimal material layout $\boldsymbol{\rho}^*$

Employ a linear system solver to obtain $\mathbf{u}^0 = \mathbf{K}^{-1}(\boldsymbol{\rho}^0)\mathbf{f}$
Evaluate the objective function value $f(\mathbf{u}^0, \boldsymbol{\rho}^0)$
$\boldsymbol{\rho}^* = \boldsymbol{\rho}^0$, $U = \infty$
for $k = 1, \cdots$ **do**
    $\boldsymbol{\alpha}^k, \boldsymbol{\rho}^k, \eta^k \leftarrow$ MasterProbSolver($\boldsymbol{\alpha}^i$)
    Employ a linear system solver to obtain $\mathbf{u}^k = \mathbf{K}^{-1}(\boldsymbol{\rho}^k)\mathbf{f}$
    Evaluate the objective function value $f(\mathbf{u}^k, \boldsymbol{\rho}^k)$
    if $f(\mathbf{u}^k, \boldsymbol{\rho}^k) < U$ **then**
        $U = f(\mathbf{u}^k, \boldsymbol{\rho}^k)$, $\boldsymbol{\rho}^* = \boldsymbol{\rho}^k$
    end if
    if $\frac{|\eta^k - U|}{|U|} < \varepsilon$ or $\eta^k > U$ **then**
        break
    end if
    Adjust the trust region radius according to Equation (2.38)
end for
Return: $\boldsymbol{\rho}^*$

It is worth noting that the stopping criterion used here differs from those in other methods. In SIMP [6], the optimization iterations are terminated when the changes in the design variables become small over multiple consecutive iterations. In the methods that directly handle binary design variables and draw on integer optimization [35, 57, 47], including the framework proposed in this work, the stopping criterion is typically based on the objective function values. However, BESO [35], TOBS [57], and SAIP [47] terminate the optimization iterations when the changes

in the objective function values become small over multiple consecutive iterations, usually requiring at least 10 consecutive iterations. In contrast, the proposed framework allows for the estimation of the upper and lower bounds of the original objective function by decomposing the problem into primal and master sub-problems, which are expected to converge as the solution nears the true value. Thus, the stopping criterion is established by comparing the difference between these upper and lower bounds, which is both theoretically sounder and practically more efficient.

2.4 Parameter Relaxation

Due to the small value of the minimum Young's modulus assigned to void elements (i.e., $E_0 = 10^{-4}$), and the small target volume fraction (e.g., $V_T = 0.3$) or small maximum permissible total mass fraction (e.g., $\bar{M}_{max} = 0.3$), the optimization problem can be ill-conditioned, as evidenced in Table. 2.3. Thus, different TO methods usually relax some parameter's values during the solution process, i.e., starting with a relatively larger value and later reducing the parameter's value to the desired one, in order to ease the ill-conditioning issue. Typically, for single-material problems, the target volume fraction is chosen to relax, e.g., in [84, 57, 47, 34]; for multi-material problems, the target maximum permissible mass fraction is typically relaxed, e.g., in [35, 49].

In the present work, it further explores relaxing the minimum Young's modulus E_0, alongside the target volume fraction V_T or the target maximum permissible mass fraction $\bar{M}_{max}$. From numerical experiments, it is found that the minimum Young's modulus contributes the most to the conditioning of the optimization problem. Thus, it considers the minimum Young's modulus as the primary parameter for relaxation and the target volume fraction or the maximum permissible total mass fraction as the secondary parameter for relaxation. Accordingly, the proposed parameter

relaxation scheme consists of two parts. In the first part, the value of the target volume fraction or the maximum permissible total mass fraction is changed, while the minimum Young's modulus is fixed at a larger value 10^{-2}. This part can consist of N_P stages. In each stage, the parameter's value is gradually changed from an initial larger value towards the desired one, following:

$$P_l = P_0 e^{-\frac{l-1}{N_P-1}\log\frac{P_0}{P_d}}, \quad l = 1, \dots N_P \,, \tag{2.39}$$

where P_0 denotes the initial value for the parameter; P_d denotes the desired value of the parameter; and P can be either the target volume fraction V_T or the maximum permissible total mass fraction $\bar{M}_{max}$. Here, an exponential function is employed to achieve a substantial initial reduction in the parameter values, followed by a gradual and smoother decrease, ultimately reaching the desired value. The second part consists of only one stage, where E_0 is varied from 10^{-2} to 10^{-9}, while the value of the target volume fraction or the maximum permissible total mass fraction is fixed at its desired value. As a result, the entire parameter relaxation scheme involves $N_P + 1$ stages in total. Such proposed parameter relaxation scheme is summarized in Algorithm 2. In the present work, for the very first stage, the initial material configuration $\boldsymbol{\rho}^0$ is set to a gray material layout, same as that in SIMP. For subsequent stages, the initial material configuration $\boldsymbol{\rho}^0$ in Algorithm 2 is the resulting binary (black/white) material layout yield at the end of the previous stage.

In this section, a thorough analysis for parameter relaxation is presented, exploring its effectiveness in improving both solution efficiency and quality. It discusses the rationale behind its efficacy and the selection of suitable parameters for relaxation.

During the solution process, especially in the initial stages, the optimization problem encountered may suffer from ill-conditioning. This implies that slight variations on the design variable (i.e., a few elements

Algorithm 2 ParameterRelaxation($\mathbf{X}$)

Input: $\mathbf{X}$, which are the minimum Young's modulus $\mathbf{E} = \{E_0^1, \cdots, E_0^{N_P+1}\}$ and the target volume fraction $\mathbf{V} = \{V_1, \cdots, V_{N_P+1}\}$ or the maximum permissible total mass fraction $\bar{\mathbf{M}} = \{\bar{M}_1, \cdots, \bar{M}_{N_P+1}\}$
Output: Optimal material layout $\boldsymbol{\rho}^*$

Initialize the design variables as $\boldsymbol{\rho}^* = V_T\mathbf{1}$ or $\boldsymbol{\rho}^* = \bar{M}_{max}\mathbf{1}$, where $\mathbf{1}$ is a unit vector
Initialize the trust-region radius d^0
for $l = 1, \cdots N_P + 1$ **do**
    $\boldsymbol{\rho}^0 \leftarrow \boldsymbol{\rho}^*$
    if $l \leqslant N_P$ **then**
        $E_0 \leftarrow E_0^l = 10^{-2}$
        Evaluate V_l or $\bar{M}_l$ from Eq. (2.39)
        $V_T \leftarrow V_l$ or $\bar{M}_{max} \leftarrow \bar{M}_l$
    end if
    if $l = N_P + 1$ **then**
        $E_0 \leftarrow E_0^l = 10^{-9}$
        $V_T \leftarrow V_l = V_d$ or $\bar{M}_{max} \leftarrow \bar{M}_l = \bar{M}_d$
    end if
    $\boldsymbol{\rho}^* \leftarrow$ TOSolver($\boldsymbol{\rho}^0$)
end for
Return: $\boldsymbol{\rho}^*$

transitioning from solid to void or vice versa), could lead to significant fluctuations in the objective function value. To demonstrate this, it first compares the objective function values corresponding to different topological configurations obtained with the minimum Young's modulus $E_0 = 10^{-9}$ vs. $E_0 = 10^{-2}$, as denoted as f_1 and f_2, respectively, in Table. 2.3.

Table 2.3(a) and 2.3(b) are what it was obtained from the first two consecutive iteration steps, where the target volume fraction was set as $V_T = 0.3$. In these two steps, the difference in the design variable, measured in 2-norm, is $\frac{\|\boldsymbol{\rho}^1 - \boldsymbol{\rho}^2\|_2}{n_e} = 0.3$, which is considered a minor change. However, the substantial disparity in objective function values between these two steps, with $E_0 = 10^{-9}$, indicates the presence of ill-conditioning

Table 2.3: Single-material minimum compliance for the MBB design: The roles of the minimum Young's modulus and the target volume fraction in the optimization problem's conditioning.

Configuration	V_T	f_1 ($E_0 = 10^{-9}$)	f_2 ($E_0 = 10^{-2}$)
a 	0.3	10186564604.77	1473.14
b 	0.3	109405.30	764.62
c 	0.6	83637287.04	521.03
d 	0.6	2902.42	398.42

in the optimization problem at this stage. Solving this ill-conditioned problem could lead to an unstable or non-converging solution. In contrast, by relaxing the minimum Young's modulus to $E_0 = 10^{-2}$, the ill-conditioning is mitigated. This elucidates why relaxing the value of the minimum Young's modulus during the optimization process can effectively enhance both solution efficiency and quality. It is noted that in literature, other TO methods that rely on discrete programming chose to relax the target volume fraction during the optimization process, e.g., as reported in [47, 57, 84]. Thus, it repeated the same numerical experiment but with a larger target volume fraction, i.e., $V_T = 0.6$, as outlined in Table 2.3(c) and 2.3(d). The difference in the design variable $\boldsymbol{\rho}$, measured in 2-norm, is less than 0.3 between the two configurations in Table 2.3(c) and 2.3(d). With $E_0 = 10^{-9}$, by setting a larger target volume fraction from 0.3 to 0.6, the optimization problem is still ill-conditioning. It is evident that enlarging the target volume fraction has a limited effect on improving the problem's

conditioning, when compared with elevating the minimum Young's modulus. Thus, in the prior sections, it chose to focus on the minimum Young's modulus E_0 in parameter relaxation.

In practice, relaxing more than one parameters can further improve the solution efficiency and/or solution quality, depending on the conditioning of the problem. Henceforth, it further explored adjusting both the minimum Young's modulus E_0 and the target volume fraction V_T. One exemplary implementation is given as **Scheme 2** in Table 2.7, where the parameter relaxation process described in §2.5.1.5 was partitioned into 9 stages with $N_P = 8$ and the initial value for the volume fraction V_1 set as 0.6 in Eq. (2.39). The results, as summarized in Table 2.8, are compared with those obtained by relaxing only E_0 in two stages as employed in the prior sections, denoted as **Scheme 1** in Table 2.7.

2.5 Numerical Results

For the validation of the proposed multi-cuts framework, a series of benchmark problems in 2D are considered. It includes the minimum compliance problem and compliant mechanism design. The numerical experiments are conducted on a workstation with two Intel Xeon E5-2690 v4 CPUs and 512 GB of RAM. MATLAB R2023a is used as the programming environment. The Gurobi optimizer v11.0 [28] is utilized for solving the mixed integer programming problems. The proposed method is compared with the SIMP method with the Heaviside projection [6]. It is also compared with other discrete variable TO methods, e.g. the FP method [32, 33] and the SAIP method [47, 49]. To eliminate the influence of the programming environment, the number of FEM evaluations and the objective function value are used as the primary metrics for comparison.

2.5.1 Minimum Compliance

2.5.1.1 Problem steup

The minimum compliance design is first considered, but with two different settings, as depicted in Figure 2.1. The first setting corresponds to the Messerschmitt-Bölkow-Blohm (MBB) beam problem, as illustrated in Figure 2.1a. It is designated in a rectangular domain with a length-height ratio of $L : H = 3 : 1$. The left-side edge of the rectangular domain is constrained in x (horizontal) direction but can freely move along y (vertical) direction. The movement of the bottom-right corner is restricted solely along x-axis. An external force $\mathbf{F}_y = -1$ is exerted at the top-left corner of the design domain. The second setting represents the design of a cantilever, defined in a rectangular domain with a length-height ratio of $L : H = 2 : 1$, as shown in Figure 2.1b. The left side of the rectangular domain is constrained in both x and y axes. In both problems, the goal of optimization is to minimize the compliance of the entire structure within the corresponding design domain. The single material's Young's modulus is specified as $E_1 = 1.0$ and Poisson ratio as $\nu = 0.3$. The design domain is always discretized into quadrilateral elements, but with different resolutions. In Eq. (2.2), $f(\mathbf{u}, \boldsymbol{\rho}) = \mathbf{f}^\intercal \mathbf{u}$, where $\mathbf{f}$ is the discretized external force vector; and there is only one constraint, i.e., the volume constraint that constrains the volume fraction of solid material toward the target value V_T. In each problem, it considers three different target volume fractions, i.e., $V_T = 0.3$, 0.4, or 0.5, for a systematic analysis and validation. Particular emphasis is placed on the scenario with the highest discretization resolution and lowest target volume fraction, because of the high fidelity of designs provided by high discretization resolutions and the preference of low volume fractions of solid material in practical designs [1]. The solution process consistently started with an initial gray configuration satisfying the desired target volume fraction, same as that in SIMP [6]. The convergence tolerance (as

delineated in Algorithm 1) was set as $\varepsilon = 5 \times 10^{-3}$.

2.5.1.2 Fixed trust-region radius

A fixed trust-region radius was firstly employed without dynamically varying it in the solution procedure. Thus, the trust-region radius was always equal to its initial value d^0, and three different fixed radii were examined, each with $d^0 = 0.05, 0.1$ or 0.2, respectively. In the parameter relaxation scheme proposed in Section 2.5.1.5, only the minimum Young's modulus E_0 was relaxed. Thereby, the optimization procedure was split into two stages: in the first stage, it was set as $E_0 = 10^{-2}$; in the second stage, it was set with the required value in the original problem as $E_0 = 10^{-9}$. The numerical results are summarized in Table 2.4, where N_{FEM} denotes the number of FEM analyses performed, or equivalently the number of iterations required throughout the optimization procedure; f denotes the resultant value of the objective function. Here, the maximum allowed number of iterations was set as 100.

From the results, it is obvious that using a fixed trust-region radius may not ensure reaching a converged solution within the maximum allowed number of iterations, especially when the target volume fraction is small (e.g. $V_T = 0.3$) or the trust-region radius is set too large (e.g. with $d^0 = 0.2$). This implies that if a fixed trust-region radius is employed, how to select its proper value should be problem dependent, and finding a consistent value applicable across different problem setups, each characterized by different BCs, discretization resolutions, and/or volume constraints, can be inherently challenging. Generally speaking, the optimization process with either too small or too large trust-region radii could require more iteration steps to reach convergence [54]. More sophisticate discussions are provided below.

As indicated in sensitivity analysis about the discrete TO problems [67], the accuracy of the sensitivities estimated for void elements ($\rho_i =$

Table 2.4: Single-material minimum compliance: Results with fixed trust-region radii.

r	MBB					Cantilever				
	Resolution	V_T	d^0	N_{FEM}	f	Resolution	V_T	d^0	N_{FEM}	f
4	240×80	0.3	0.05	100	446.36	240×120	0.3	0.05	24	103.14
			0.1	100	368.66			0.1	31	135.55
			0.2	100	446.36			0.2	100	349.87
		0.4	0.05	34	241.60		0.4	0.05	23	82.86
			0.1	37	232.20			0.1	28	86.81
			0.2	66	275.53			0.2	29	75.51
		0.5	0.05	30	210.79		0.5	0.05	26	62.92
			0.1	32	196.96			0.1	24	63.27
			0.2	100	265.65			0.2	25	64.35
6	360×120	0.3	0.05	65	380.72	360×180	0.3	0.05	100	115.12
			0.1	100	442.64			0.1	33	121.38
			0.2	100	742.72			0.2	100	381.12
		0.4	0.05	34	240.08		0.4	0.05	24	82.93
			0.1	100	251.10			0.1	27	77.85
			0.2	100	355.44			0.2	35	79.22
		0.5	0.05	31	227.37		0.5	0.05	26	62.92
			0.1	30	196.03			0.1	24	63.59
			0.2	29	192.64			0.2	25	64.97

0) is usually lower than that for solid elements ($\rho_i = 1$). Therefore, if the target volume fraction is large (e.g. $V_T = 0.5$), the fidelity of the sensitivity analysis can be maintained, and the proposed framework can easily converge within a reasonable number of iterations (around 30) with the optimal values of the objective function comparable across various choices of d^0. On the contrary, if the target volume fraction is small (e.g. $V_T = 0.3$), a smaller trust-region radius (e.g., with $d^0 = 0.05$) is preferred, in order to enhance the sensitivity analysis' accuracy and ensure a fast converged solution. A representative instance matching this discussion is the MBB problem solved at the resolution of 360×120 for $V_T = 0.3$ or 0.4. The convergence of solution is achieved solely when d^0 is set to

0.05, resulting in 65 or 34 iteration counts, whereas setting d^0 to 0.1 or 0.2 both fails to yield convergence within the maximum allowed number of iterations. This suggests that if a fixed trust-region radius is employed, its value should be set sufficiently small to ensure convergence. However, if its value is set too small, the optimization process may be terminated early by the stopping criterion $\frac{|\eta^k - U|}{|U|} < \varepsilon$ before the local minimum of the objective function is actually found. For instance, in the MBB problem at the resolution of 360×120 and with the target volume fraction of $V_T = 0.5$, employing a small trust-region radius ($d^0 = 0.05$) results in a higher objective function value compared to using $d^0 = 0.2$, with the difference between the resultant objective function values as much as 18.0%.

The findings of this study indicate that trust-region radii need to be determined by the problem and dynamically adjusted during the optimization process to ensure fast convergence of the solution.

2.5.1.3 Adaptive trust-region radius

For the tests in this section, instead of maintaining a fixed value, the trust-region radius was dynamically varied following the rule described in Eq. (2.38). Figure 2.4 depicts the evolution of the trust-region radius and the objective function value during the optimization process, along with the resulting material configuration at different iteration steps for the MBB design. Same as in the prior section, only the minimum Young's modulus E_0 was relaxed, and the optimization process was split into two stages with $E_0 = 10^{-2}$ in the first stage and $E_0 = 10^{-9}$ in the second stage, as indicated in Figure 2.4.

The results for all the cases considered are compared to the best results achieved using fixed trust-region radii, as presented in Table 2.5. The column of **Fixed Radius** denotes the best performed cases with respect to the resultant objective function value seen in Table 2.4. For the results using adaptive radii, as listed in the column of **Adaptive Radius**, d^0 denotes

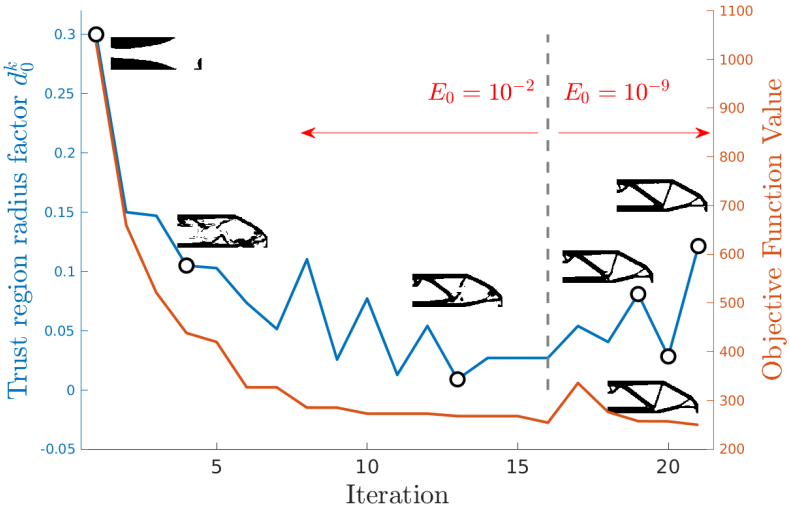


Figure 2.4: Single-material minimum compliance for the MBB design: Evolution of the adaptive trust-region radius and the objective function value during the optimization process, along with the resulting material configuration at different iteration steps. Here, the target volume fraction is $V_T = 0.4$; the discretization resolution is 240×80; and the initial trust-region radius is set as $d^0 = 0.3$. The minimum Young's moduli are $E_0 = 10^{-2}$ and $E_0 = 10^{-9}$, respectively, for the iteration steps before and after the gray dashed line.

the *initial* trust-region radius, whose value is then varied according to Eq. (2.38) in subsequent iterations throughout the solution procedure. For both MBB and cantilever problems and across various discretization resolutions and target volume fractions, employing adaptive trust-region radii consistently ensured convergence to the optimal solution within the maximum allowed number of iterations. Next, the number of optimization iterations (or N_{FEM}) required for convergence remains notably small and relatively consistent across different scenarios. When compared to the outcomes obtained using fixed trust-region radii, N_{FEM} is consistently smaller in each scenario considered. Finally, with respect to the resultant objective function value, while employing fixed radii may occasionally lead

Table 2.5: Single-material minimum compliance: Comparison for employing adaptive vs. fixed trust-region radii.

Problem	**Resolution**	r	V_T	**Fixed Radius**			**Adaptive Radius**		
				d^0	N_{FEM}	f	d^0	N_{FEM}	f
MBB	240×80	4	0.3	0.1	100	368.66	0.3	36	294.43
			0.4	0.1	37	232.20	0.4	19	233.80
			0.5	0.1	32	196.96	0.4	20	193.45
	360×120	6	0.3	0.05	65	380.72	0.4	28	315.02
			0.4	0.05	34	240.08	0.4	22	236.72
			0.5	0.2	29	192.64	0.4	26	214.60
Cantilever	240×120	4	0.3	0.05	24	103.14	0.5	19	99.62
			0.4	0.2	29	75.51	0.5	15	77.79
			0.5	0.05	26	62.92	0.5	17	63.62
	360×180	6	0.3	0.1	100	115.12	0.5	19	104.71
			0.4	0.1	27	77.85	0.5	20	76.54
			0.5	0.05	26	62.92	0.5	16	64.46

to slightly lower objective function values, particularly for higher target volume fractions (e.g., $V_T = 0.5$), the superiority of adaptive radii becomes apparent for lower target volume fractions (especially, $V_T = 0.3$), which are more relevant in practical designs. This superiority is characterized by not only achieving lower objective function values but also requiring fewer iteration counts, thus highlighting the effectiveness of adaptive radii over fixed radii when dealing with designs for lower target volume fractions.

Regarding the initial value of the trust-region radius d^0, it tested it ranging from 0.05 to 0.5. The best results with respect to N_{FEM} are included in Table 2.5. It is worth noting that by adaptively adjusting the trust-region radius instead of using a fixed value, it could begin with a larger d^0 while achieving convergence and comparable objective function values with fewer iterations. And this becomes particularly evident when the target volume fraction is small (e.g. $V_T = 0.3$). In such cases, employing adaptive trust-region radii consistently yield lower objective function values and

required far fewer iteration steps compared to employing fixed trust-region radii.

2.5.1.4 Comparison with other TO methods

It is then turned to the comparison with other TO methods, including SIMP with Heaviside projection [65] and GBD [53, 84], for which the MBB problem was considered. The comparison results are summarized in Table 2.6. Here, the column **Proposed Method** contains the results pre-

Table 2.6: Single-material minimum compliance: Comparison with other TO methods for solving the MBB problem.

Resolution	r	V_T	**Porposed Method**		**SIMP-Heaviside**		**GBD-E_0**		**GBD-V_T**			
			N_{FEM}	f	N_{FEM}	f	N_{FEM}	f	$\Delta V = \frac{1}{24}$		$\Delta V = \frac{1}{12}$	
									N_{FEM}	f	N_{FEM}	f
240×80	4	0.3	36	294.43	300	310.24	300	8.19e+10	119	286.95	104	414.51
		0.4	19	233.80	300	239.53	300	5.80e+10	107	223.50	99	266.23
		0.5	20	193.45	300	193.10	300	2.05e+10	82	193.18	76	190.07
360×120	6	0.3	28	350.48	300	312.31	300	7.38e+10	104	454.84	72	496.75
		0.4	22	236.72	300	238.30	300	6.18e+10	92	233.25	70	232.64
		0.5	26	214.60	300	194.81	300	2.02e+10	76	190.92	54	194.25

sented in the rightmost column of Table 2.5, i.e., with adaptive trust-region radii, for the MBB problem. For GBD, it examined two ways of parameter relaxation: one for the minimum Young's modulus and the other for the target volume fraction. In the former, the optimization procedure was split into two stages: $E_0 = 10^{-2}$ was employed in the first stage, and in the second stage, it was set with the required value in the original problem as $E_0 = 10^{-9}$, same as that employed in the prior two sections. The corresponding results are presented under the column **GBD-**E_0 in Table 2.6. In the latter, the target volume fraction in the volume constraint is decreased incrementally per stage until reaching the desired value in the original problem, with the initial value set to 1.0 and the change increment fixed at $\Delta V = \frac{1}{24}$ or $\Delta V = \frac{1}{12}$. The corresponding results are presented under

the column **GBD-**V_T in Table 2.6. The maximum number of iterations was capped at 300 for all the methods compared herein.

Across different discretization resolutions and target volume fractions, the proposed method consistently converges faster with significantly fewer iteration steps—about one order of magnitude fewer compared to the iteration counts needed by SIMP, as indicated by N_{FEM}, the number of FEM analyses called during the solution process. The resultant values of the objective function are comparable to, and sometimes even lower than, those obtained by the SIMP method, particularly for cases with lower target volume fractions. The proposed method also outperforms the GBD method in both the iteration counts and solution quality. When the minimum Young's modulus was chosen as the parameter to relax, GBD struggles to identify the optimal solutions, particularly for the cases with lower target volume fractions. Considering that the minimum compliance problem is a convex problem, solving it using the GBD method is equivalent to using the proposed framework with a fixed trust-region radius of $d^k \equiv 1$. As shown in Table 2.4, a too large fixed trust-region radius cannot ensure a stable and converged solution. When the target volume fraction (in the volume constraint) was chosen as the parameter to relax, GBD performed better. This is because by gradually changing the volume fraction, a varying trust region is implicitly enforced. The change in the design variable $\boldsymbol{\rho}$ in each iteration is constrained by the volume fraction increment ΔV, which limits the allowable total change in $\boldsymbol{\rho}$. To ensure convergence, a smaller ΔV is preferred, but it could result in more iteration steps, which can be found when comparing the results for $\Delta V = \frac{1}{24}$ vs. $\Delta V = \frac{1}{12}$.

The resulting optimal topology is illustrated in Figure 2.5. The proposed method achieved a clear-cut 0/1 material layout, in contrast to that produced by the SIMP method, where part of the topology is still in gray scale, requiring more optimization iterations or post-processing steps to reach a clear-cut 0/1 configuration. For **GBD-**V_T with $\Delta V = \frac{1}{24}$, al-

though the optimization converged, the resulting objective function value remained high, higher than those obtained by the proposed method and SIMP by more than 30%. This discrepancy leads to a notably distinct final topology generated by the GBD method, as depicted in Figure 2.5.

(a) Proposed method (b) SIMP-Heaviside (c) GBD-V_T

Figure 2.5: Single-material minimum compliance: The optimal topology obtained for the MBB design from different methods, with the discretization resolution of 360×120 and the target volume fraction of $V_T = 0.3$.

However, the resulting optimal topology in Figure 2.5a with the deflected top-left and bottom-right parts, may be still undesirable. To further improve the topology, the parameter relaxation scheme (proposed in Algorithm 2) was employed, as discussed in the next subsection. The improved topology is then shown in Figure 2.6a.

2.5.1.5 Parameter relaxation

In this section, a thorough analysis for parameter relaxation is presented, exploring its effectiveness in improving both solution efficiency and quality. It discusses the rationale behind its efficacy and the selection of suitable parameters for relaxation.

In practice, relaxing more than one parameters can further improve the solution efficiency and/or solution quality, depending on the conditioning of the problem. Henceforth, it further explored adjusting both the minimum Young's modulus E_0 and the target volume fraction V_T. One exemplary implementation is given as **Scheme 2** in Table 2.7, where the parameter relaxation process described in §2.5.1.5 was partitioned into 9 stages with $N_P = 8$ and the initial value for the volume fraction V_1 set as 0.6 in Eq. (2.39). The results, as summarized in Table 2.8, are compared

with those obtained by relaxing only E_0 in two stages as employed in the prior sections, denoted as **Scheme 1** in Table 2.7. For a more systematic

Table 2.7: Single-material minimum compliance for the MBB design: Two parameter relaxation schemes.

	Stage	1	2	3	4	5	6	7	8	9
Scheme 1	V_T	0.3	0.3							
	E_0	10^{-2}	10^{-9}							
Scheme 2	V_T	0.600	0.543	0.492	0.446	0.404	0.366	0.331	0.300	0.300
	E_0	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-9}

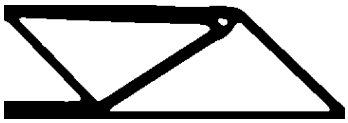
analysis, it also extended the design domain to a larger length-height ratio, $L : H = 4 : 1$. For the domain of each ratio, two different discretization resolutions were examined. The results are summarized in Table 2.8. The

Table 2.8: Single-material minimum compliance for the MBB design: Comparison between the two parameter relaxation schemes, using the results from the SIMP method as the reference.

Resolution	**Scheme 1**		**Scheme 2**		**SIMP-Heaviside**	
	f	N_{FEM}	f	N_{FEM}	f	N_{FEM}
240×80	294.43	36	294.07	52	310.24	300
320×80	662.61	41	613.37	58	635.74	300
360×120	350.48	28	292.88	51	312.31	300
480×120	808.75	84	605.63	51	639.60	300

final material configuration for each domain, obtained with the finest discretization resolution, are exhibited in Figure 2.6. Through comparison, it is found that the resultant objective function values from **Scheme 2** are consistently lower. More pronounced improvements are observed at higher discretization resolutions and greater aspect ratios of the design domain, e.g., in the case of 480×120, with the improvement in the objective function value reaching as high as 26.4% and the iteration counts reduced by about 40%. These findings suggest that relaxing the minimum

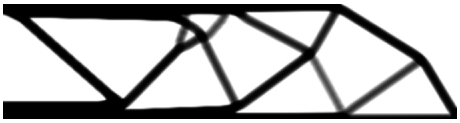
Young's modulus along with another parameter in multiple stages could potentially offer greater advantages in achieving lower objective function values and possibly reducing iteration counts as well. The extent of these enhancements can vary depending on the specific problem, particularly its conditioning.



(a) Length-height ratio: 3 : 1; discretization resolution: 360 × 120, obtained by employing **Scheme 2** in Table 2.7 for parameter relaxation.

(b) Length-height ratio: 3 : 1; discretization resolution: 360 × 120, obtained by SIMP.

(c) Length-height ratio: 4 : 1; discretization resolution: 480 × 120, obtained by employing **Scheme 2** in Table 2.7 for parameter relaxation.



(d) Length-height ratio: 4 : 1; discretization resolution: 480 × 120, obtained by SIMP.

Figure 2.6: Single-material minimum compliance for the MBB design: The final topological configuration with the target volume fraction of $V_T = 0.3$.

2.5.2 Compliant Mechanism

The next benchmark is a non-convex TO problem, the compliant mechanism design, where the displacement objective exhibits non-convexity with respect to the design variables $\boldsymbol{\rho}$. It represents a broader class of TO challenges than the minimum compliance design addressed earlier, which is regarded as a convex TO problem.

2.5.2.1 Problem setup

It addresses the topology optimization of a typical compliant mechanism. The design domain and boundary conditions for the problem are illustrated in Fig. 2.3. The design domain is square-shaped with a side length L. Both the top-left and bottom-left corners are fixed. Additionally, the domain is connected to two springs at the center, as depicted in the figure. Both springs have a stiffness coefficient of $k_1 = k_2 = 0.1$. The input force applied to the center of the left edge is $\mathbf{F}_x^{in} = 1$. The objective of this design challenge is to maximize the displacement magnitude of the central point on the right edge along the x-axis which is equivalent to minimize $\mathbf{u}_x^{out}$. Given the problem's axial symmetry about the red dotted-dash line in the center, it is only necessary to discretize one half of the domain. The discretized domain, represented by a uniform mesh, can also be viewed in Fig. 2.3. The Young's modulus is $E = 1.0$ and the Poisson ratio is $\nu = 0.3$. The target volume fraction is $V_T = 0.3$. The convergence tolerance is set as $\varepsilon = 5 \times 10^{-3}$. The domain is discretized by quadrilateral elements and the governing equation is discretized by the finite element method (FEM). It is a non-convex optimization problem, as the displacement objective is non-convex with respect to the design variables $\boldsymbol{\rho}$. It can be seen as an example of more general TO challenges compared to the minimum compliance problem in the previous section. The effectiveness of the proposed method shown in this numerical experiment suggests that the multi-cuts framework can be a promising candidate for solving many other more complex TO problems.

2.5.2.2 Running results and comparison to the SIMP method

Similar to the minimum compliance problem, the design of compliant mechanism requires the parameter relaxation scheme as well. In this numerical experiment, two stages of the parameter relaxation is applied. For all two stages, the target volume fraction is stuck the final value as

$V_T = 0.3$. The value of the minimum Young's modulus is different in the two stages, $E_0 = 10^{-2}$ in the first stage and $E_0 = 10^{-9}$ in the second stage. Different initial trust region radius d_0 is tested with different discretization resolutions for finding the optimal material configuration to the compliant mechanism problem. The results are collected in Table 2.9 under the column **Adaptive Multi-cuts**. Based on the running results, the resulting objective function value f can be very close to each other with different initial trust region radius d_0. The relative difference between the maximum and minimum f is less than 3.9%, which happens at the discretization resolution of 200×100. Besides, the number of FEM analysis N can be close to each in most cases. More importantly, the number is no greater than 40 which is much smaller than the number of iterations required by other methods. This implies that the proposed method is not quite sensitive to the setup of parameters used in the optimization and it is robust enough for accepting a wide range of parameters after the introduction of the adaptive update scheme on the trust region radius.

The SIMP method with the heaviside projection [6] has been seen as one of the standard methods to solve the TO problems. Therefore, it is taken as the baseline in this subsection to evaluate the performance of the proposed method. The running results with different discretization resolutions has been organized in Table 2.9 under the column **SIMP-Heaviside**. The maximum number of iteration for the SIMP method is set as 300. In terms of the resulting objective function value f, only three setups in the adaptive multi-cuts framework give a larger results, and the relative differences in these three cases are no larger than 1.5% when compared to the value f obtained by the SIMP method. It suggests that the proposed method can effectively find good material configurations with better objective function values and the inclusion of trust region can guide the optimizer to find the optimal solution even for a non-convex TO problem. Furthermore, the number of FEM analysis required to find the

Table 2.9: Single-material compliant mechanism: Results and comparison with the SIMP method.

Resolution	r	**The Proposed Method**			**SIMP-Heaviside**	
		d^0	N_{FEM}	f	N_{FEM}	f
200×100	2	0.05	35	-0.9271	300	-0.8960
		0.1	24	-0.9139		
		0.2	26	-0.8919		
		0.3	24	-0.9226		
		0.4	24	-0.9098		
		0.5	29	-0.9282		
400×200	4	0.05	29	-0.8661	300	-0.8511
		0.1	25	-0.8652		
		0.2	36	-0.8677		
		0.3	31	-0.8602		
		0.4	38	-0.8728		
		0.5	33	-0.8763		
800×400	8	0.05	36	-0.8313	300	-0.8132
		0.1	28	-0.8238		
		0.2	25	-0.8056		
		0.3	28	-0.8167		
		0.4	27	-0.8085		
		0.5	37	-0.8270		

optimal solution is quite different between these two methods. While the SIMP method requires a fixed number of iterations as 300, the multi-cuts framework can find the optimal solution with an one-order magnitude smaller in N. A great amount of time can be saved for performing the time consuming FEM analysis, especially in large-scale problems.

2.5.3 Multi-Material Cantilever

Following the completion of benchmark single-material TO problems, including both convex and non-convex cases, it was proceeded to address

more challenging TO problems involving multiple materials. When addressing the inclusion of multiple candidate materials in an TO framework, a vital consideration is whether the proposed method can accurately discern the coexistence of all or some of the candidate materials in the derived material configuration. It first tackled a problem involving two materials, from which it demonstrates the effectiveness of the proposed parameter relaxation scheme for further enhancing solution quality (in terms of the objective function value). Next, it considered a design scenario incorporating five different candidate materials, by which it aimed to evaluate the proposed framework's ability to identify and select the optimal combination of candidate materials in the resulting material layout. Two recent studies [49, 33] were referred to validate the results and assess the performance of the proposed framework.

It is tackled a cantilever design problem involving five materials to further assess the applicability and effectiveness of the proposed method for general multi-material design challenges. The problem setup is the same as depicted in Figure 2.1b, with the design domain's length-height ratio as $L : H = 2 : 1$. For a more systematic assessment, it examined two distinct sets of five candidate materials, with the respective Young's moduli E and normalized densities for the five candidate materials detailed in Table 2.1.

The parameter relaxation scheme was implemented with 8 distinct stages, as delineated in Table 2.10, where the maximum permissible total mass fraction $\bar{M}_{max}$ is decreased progressively from 0.5 to the designated target of 0.3 within the first $N_P = 7$ stages, following Eq. (2.39). In line with the previous tests, it maintained the same convergence tolerance of $\varepsilon = 5 \times 10^{-3}$.

The results reported in the literature [49, 33] are used to validate and compare with the results, as presented in Table 2.11. While the referenced papers [49, 33] reported outcomes solely at a discretization resolution of

Table 2.10: Five-material cantilever: Parameter relaxation scheme employed.

Stage	1	2	3	4	5	6	7	8
$\bar{M}_{max}$	0.500	0.459	0.422	0.387	0.356	0.327	0.300	0.300
E_0	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-2}	10^{-9}

120×80, the numerical experiments extended to consider two higher resolutions as well. The optimal material configurations obtained at various

Table 2.11: Five-material cantilever: Main results, for both material sets considered and compared with the reference results reported in literature.

	Resolution	r	**The Proposed Method**		**Reference** [33]		**Reference** [49]	
			N_{FEM}	f	N_{FEM}	f	N_{FEM}	f
Material Set 1	120×80	3	34	37.500	250	37.881	> 200	36.568
	240×160	6	36	37.724	-	-	-	-
	480×320	12	36	38.458	-	-	-	-
Material Set 2	120×80	3	33	36.631	250	36.722	> 200	35.755
	240×160	6	35	36.614	-	-	-	-
	480×320	12	34	37.325	-	-	-	-

resolutions for two distinct material sets are illustrated in Figures. 2.7 and 2.8.

(a) Discretization resolution 120×80.

(b) Discretization resolution 240×160.

(c) Discretization resolution 480×320.

Figure 2.7: Five-material cantilever: The resultant material configurations at various resolutions for Material Set 1. The color code for denoting different candidate materials is specified as: denotes MAT 1; denotes MAT 2; denotes MAT 3; denotes MAT 4; and denotes MAT 5.

(a) Discretization resolution 120×80.

(b) Discretization resolution 240×16.

(c) Discretization resolution 480×320.

Figure 2.8: Five-material cantilever: The resultant material configurations at various resolutions for Material Set 2. The color code for denoting different candidate materials is specified as: denotes MAT 1; denotes MAT 2; denotes MAT 3; denotes MAT 4; and denotes MAT 5.

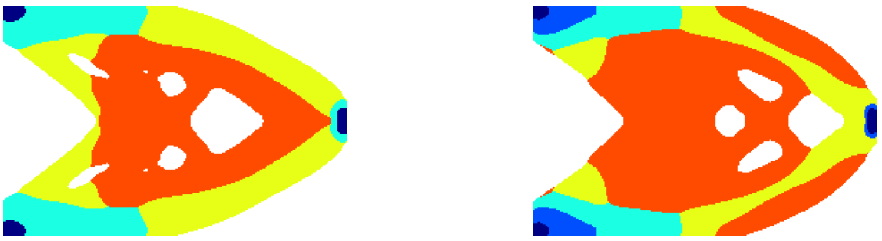
As seen in Table 2.11, for both material sets the proposed method yield comparable resultant objective function values with those reported in the state-of-the-art literature [49, 33]. For Material Set 1, the obtained optimal material layout does not select all candidate materials but instead selects four out of five materials (i.e., MAT 1, 2, 3, and 5), as shown in Figure 2.7. This is consistent with the finding reported in [49], as the material preferences, implicitly embedded in the sensitivity specified in Eq. (2.21), align with those implied in [49]. For Material Set 2, since MAT 4 exhibits an increased Young's modulus (E_4) along with enhanced specific stiffness ($E_4/\bar{M}_4$), all five candidate materials are retained in the final material layout, as depicted in Figure 2.8, consistent with the findings reported in [49, 33].

By comparing the number of FEM analyses N_{FEM}, the proposed method demonstrates its superior performance, achieving a reduction of approximately one order of magnitude in N_{FEM}. The iteration counts or N_{FEM} were not provided in reference [49]. However, based on their reported number for the two-material case and the fact that the same methodology was employed for both the two-material and five-material cases, it is estimated to be more than 200. Furthermore, for both sets of candidate materials,

the variance in N_{FEM} across different discretization resolutions remains marginal, with a discrepancy of fewer than 3 iterations. Significantly, increasing the discretization resolution does not correlate with an increased iteration count, suggesting that the proposed framework maintains efficiency and scalability across various resolutions, demonstrating promise for tackling large-scale design problems. As the filter radius scales in accordance with the discretization resolution, the consistency of the resultant material layouts can be maintained by the proposed framework, as shown in Figures. 2.7 and 2.8. That is, the primary structural features and the distribution of different candidate materials remain consistent. Slight variations are mostly confined to the interfaces between distinct materials and some fine details, such as the configuration of minor structural elements like holes. This consistency across varying discretization resolutions, evidenced by both material sets, indicates the robustness of the proposed TO framework. Such robustness along with its computational efficiency and scalability affirm the superior capacity of the proposed TO framework to address multi-material design challenges.

Finally, it further validates that the proposed methodology does not rely on any specific encoding of the candidate materials, i.e., it has no preference on the specific order of candidate materials, as the selection is solely based on the calculated sensitivity. To this end, the design variables $\boldsymbol{\rho}_1$-$\boldsymbol{\rho}_5$ are randomly ordered for representing MAT 1, MAT 3, MAT 5, MAT 4, MAT 2, respectively. With this new assignment of design variables, it re-solved the problem at the discretization resolution of 240×160. The results and optimal topologies are summarized in Figure 2.9, where the materials are rendered with the original monolithic increasing order for ease of comparison. As compared with Figure 2.7(b) and Figure 2.8(b), the topologies are almost identical; the iteration counts and resultant objective function values are also similar to those reported in Table 2.11. These findings confirm that the proposed method does not need any specific

sorting of candidate materials for multi-material TO designs.



(a) Material Set 1 (Disordered). $N_{FEM} = 34$, $f = 37.610$.

(b) Material Set 2 (Disordered). $N_{FEM} = 35$, $f = 36.563$.

Figure 2.9: Five-material cantilever: The resultant material configurations by solving the problems with the candidate materials disordered, at the discretization resolution of 240×160. The color code is the same as in Figure 2.7 and Figure 2.8.

2.5.4 Multi-Material Compliant Mechanism

Finally, it investigated a compliant mechanism design incorporating five candidate materials, so as to assess the performance of the proposed framework for solving a non-convex TO problem with the complexity of involving multiple materials and mass constraints. Same as in the prior section, two distinct material sets, each with five candidate materials, were examined, with the respective Young's moduli E and normalized densities for all candidate materials in each set outlined in Table 2.1. The problem setup is the same as depicted in Figure 2.3. In the solution process, the parameter relaxation followed a two-stage relaxation with $E_0 = 10^{-2}$ in the first stage and $E_0 = 10^{-9}$ in the second stage while keeping $\bar{M}_{max} = 0.3$ in both stages. Being consistent with the other sections, the convergence tolerance was first set as $\varepsilon = 5 \times 10^{-3}$.

The results are summarized in Table 2.9 and compared with those of the single-material case. The numbers of FEM analyses (N_{FEM}) required to reach convergence are consistently low (typically in twenties) across

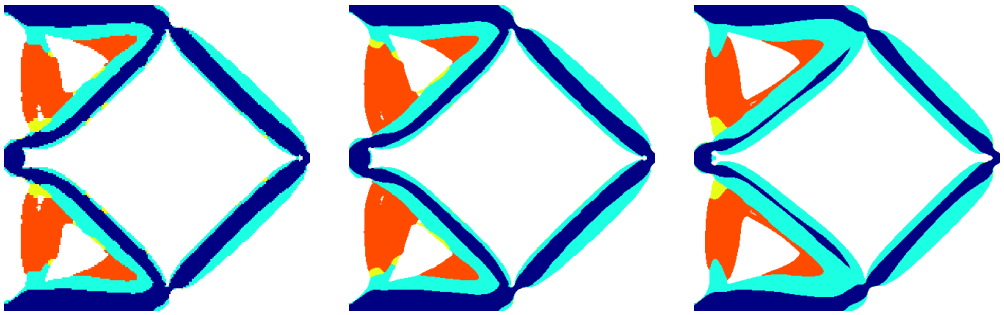
different discretization resolutions, same as the single-material compliant mechanism case and the five-material minimum compliance case, indicating the proposed framework's efficiency and scalability for dealing with multi-material non-convex designs. Due to the lack of literature data available for this test case, the results are not compared with any references. It is noted that the resultant displacement objective in this compliant mechanism design is improved by about 7% across various discretization resolutions, by involving multiple candidate materials. While this difference may not be notably significant for the two material sets considered, the benefits of expanding material diversity in TO design can be magnified when considering factors such as cost, mass, and other material properties.

Table 2.12: Five-material compliant mechanism: Main results and comparison with the single-material case.

	k_1	k_2	**Resolution**	r	**Five-material**		**Single-material**	
					N_{FEM}	f	N_{FEM}	f
Material Set 1	0.1	0.1	200×100	2	22	-0.9318	24	-0.9226
			400×200	4	22	-0.8895	31	-0.8602
			600×300	6	23	-0.8659	32	-0.8413
			800×400	8	31	-0.8478	28	-0.8167
Material Set 2	0.1	0.1	200×100	2	22	-0.9421	24	-0.9226
			400×200	4	25	-0.8951	31	-0.8602
			600×300	6	24	-0.8718	32	-0.8413
			800×400	8	29	-0.8644	28	-0.8167
Material Set 2	1.0	0.001	200×100	2	73	-2.0733	68	-2.0157
			400×200	4	80	-1.9919	71	-1.9926
			800×400	8	64	-1.8677	67	-2.0159

The final material layouts are depicted in Figures 2.10 and 2.11. Similarly to the five-material minimum compliance design, for Material Set 1, the design objective chose MAT 1, 2, 3 and 5 in the optimal layout; for Material Set 2, all five candidate materials were retained in the resultant material layout. For each material set, while lower resolutions

yield rougher interfaces between different materials, the core structural attributes are maintained across various discretization resolutions.



(a) Discretization resolution 200 × 100.

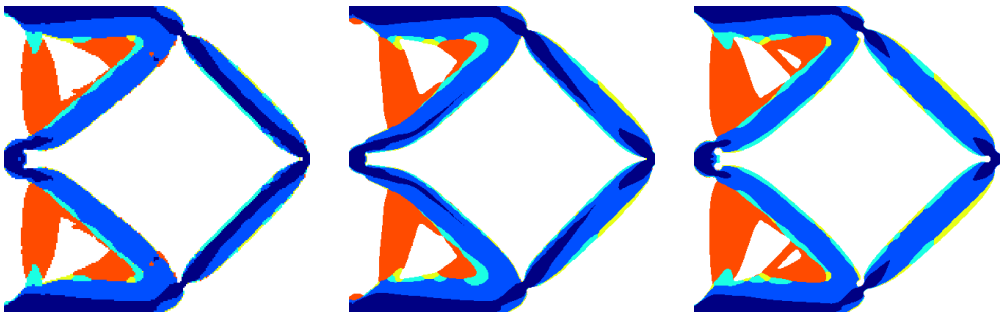
(b) Discretization resolution 400 × 200.

(c) Discretization resolution 800 × 400.

Figure 2.10: Five-material compliant mechanism: The resulting material configurations at various resolutions for Material Set 1. The color code for denoting different candidate materials is specified as: denotes MAT 1; denotes MAT 2; denotes MAT 3; denotes MAT 4; and denotes MAT 5.

In the next test, it studied the effect of the two adjustment factors (θ_1 and θ_2) in Eq. (2.38) for the trust-region radius. The values of θ_1 and θ_2 are varied with different combinations. As indicated in Table 2.13, the optimization solutions show little dependency on these two factors: N_{FEM} varies within ± 3; the optimal objective function value f varies within 6%. As for the resulting topology, the two factors' effects are different, as depicted in Figure 2.12. θ_2, which controls the degree of enlarging the trust-region radius, shows minimum influence. However, θ_1, the shrinking factor, has a more pronounced effect on the resultant topology. This is because shrinking the trust-region radius tends to increase the importance of the corresponding trust-region constraint in the multi-cut formulation Equation (4.2), and hence notably perturbs the resultant topology.

In the last test, it systematically studied the impact of the convergence tolerance ε set in Algorithm 1 on the final solution. Specifically, it examined



(a) Discretization resolution 200×100.

(b) Discretization resolution 400×200.

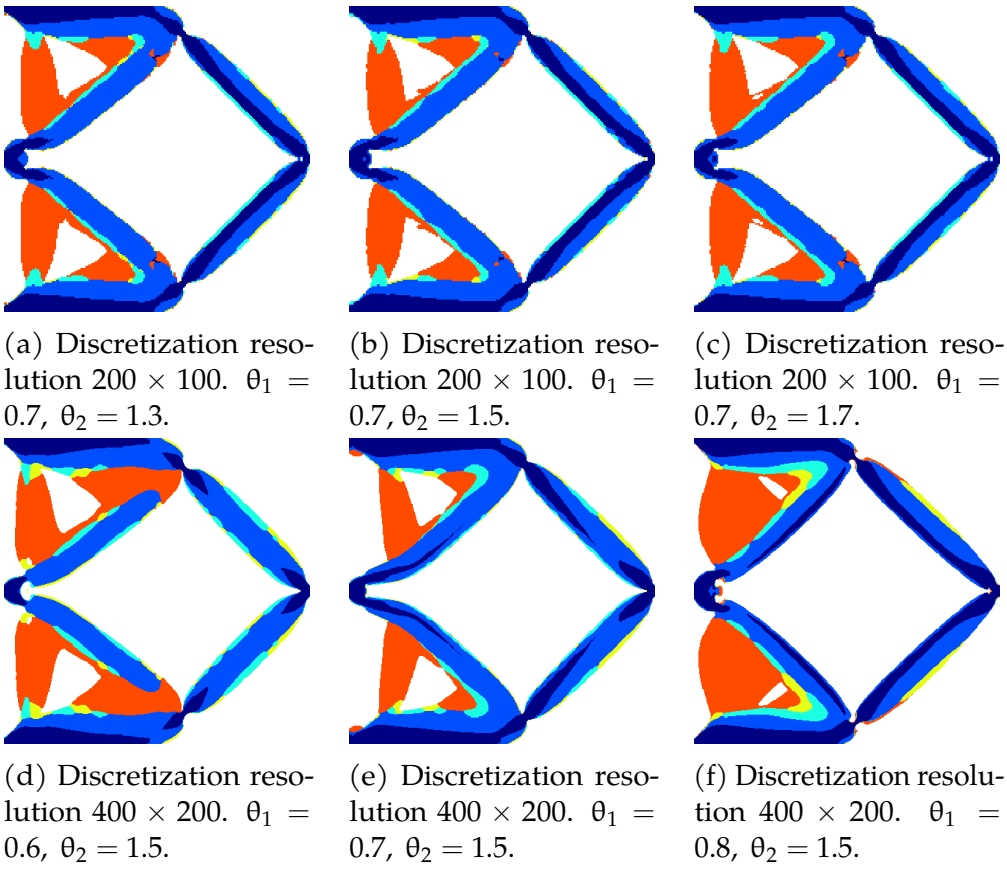
(c) Discretization resolution 800×400.

Figure 2.11: Five-material compliant mechanism: The resulting material configurations at various resolutions for Material Set 2. The color code for denoting different candidate materials is specified as: ■ denotes MAT 1; ■ denotes MAT 2; ■ denotes MAT 3; ■ denotes MAT 4; and ■ denotes MAT 5.

Table 2.13: Five-material compliant mechanism: Results with different adjustment factors for the trust-region radius (i.e., θ_1 and θ_2 in Eq. (2.38)).

Factor	200×100		400×200	
	N_{FEM}	f	N_{FEM}	f
$\theta_1 = 0.7, \theta_2 = 1.5$	22	-0.9421	25	-0.8951
$\theta_1 = 0.7, \theta_2 = 1.7$	22	-0.9333	27	-0.9238
$\theta_1 = 0.7, \theta_2 = 1.3$	22	-0.9351	25	-0.9032
$\theta_1 = 0.6, \theta_2 = 1.5$	25	-0.9608	25	-0.9107
$\theta_1 = 0.8, \theta_2 = 1.5$	24	-0.9466	28	-0.8729

three different tolerances, as outlined in Table 2.12. By comparing N_{FEM} and the optimal objective function value f, it is found that decreasing the tolerance leads to more optimization iterations to reach a better solution (or a lower objective function value). However, once the best solution has been found, further decreasing the tolerance does not change the solution, as indicted in both Table 2.14 and Figure 2.13. And it is worth noting that the proposed method did not result in redundant iterations or fluctuating



(a) Discretization resolution 200×100. $\theta_1 = 0.7$, $\theta_2 = 1.3$.

(b) Discretization resolution 200×100. $\theta_1 = 0.7$, $\theta_2 = 1.5$.

(c) Discretization resolution 200×100. $\theta_1 = 0.7$, $\theta_2 = 1.7$.

(d) Discretization resolution 400×200. $\theta_1 = 0.6$, $\theta_2 = 1.5$.

(e) Discretization resolution 400×200. $\theta_1 = 0.7$, $\theta_2 = 1.5$.

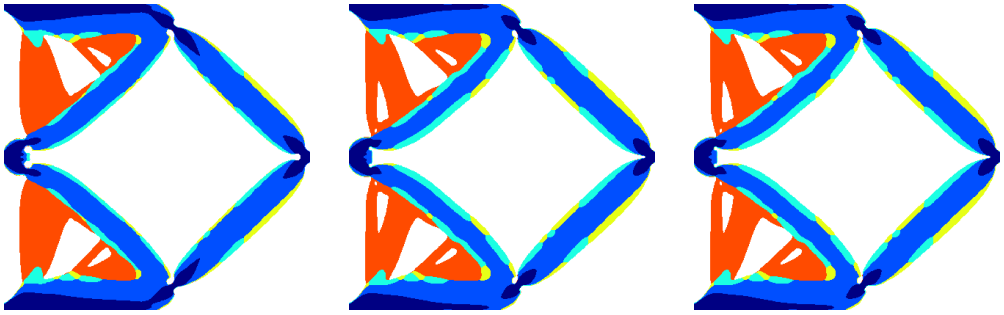
(f) Discretization resolution 400×200. $\theta_1 = 0.8$, $\theta_2 = 1.5$.

Figure 2.12: Five-material compliant mechanism: The resulting material configurations for Material Set 2 with different adjustment factors (θ_1 and θ_2) for the trust-region radius. The color code for denoting different candidate materials is specified as: denotes MAT 1; denotes MAT 2; denotes MAT 3; denotes MAT 4; and denotes MAT 5.

objective function values when further decreasing the tolerance, which contrasts with the behavior observed in other methods, e.g., [34, 47].

Table 2.14: Five-material compliant mechanism: Results with different convergence tolerance ε.

Resolution	r	$\varepsilon = 5 \times 10^{-3}$		$\varepsilon = 10^{-3}$		$\varepsilon = 5 \times 10^{-4}$	
		N_{FEM}	f	N_{FEM}	f	N_{FEM}	f
200×100	2	22	-0.9421	36	-0.9738	32	-0.9738
400×200	4	25	-0.8951	39	-0.9348	39	-0.9348
600×300	6	24	-0.8718	24	-0.8718	24	-0.8718
800×400	8	29	-0.8644	35	-0.8770	35	-0.8770



(a) Convergence tolerance 5×10^{-3}.

(b) Convergence tolerance 10^{-3}.

(c) Convergence tolerance 5×10^{-4}.

Figure 2.13: Five-material compliant mechanism: The resulting material configurations obtained with different convergence tolerance ε, all at the discretization resolution of 800×400 and for Material Set 2. The color code for denoting different candidate materials is specified as: denotes MAT 1; denotes MAT 2; denotes MAT 3; denotes MAT 4; and denotes MAT 5.

2.6 Summary and Conclusions

A new TO framework has been presented that can efficiently solve both convex and non-convex TO problems involving single or multiple materials. It directly handles binary design variables, eliminating the need for additional efforts to convert binary variables into continuous variables and vice versa (e.g., through interpolation and projection as in SIMP). When dealing with multi-material TO designs, it does not need to enumerate

all possible combinations of candidate materials and solve multiple independent TO problems like SIMP [88], nor does it need to sort candidate materials in a specific order and interpolate design variables between different materials like other methods [88, 33, 49]. Instead, it only needs to expand the dimension of design variables and incorporate additional inequality constraints arising from the mass constraint.

Through numerical tests and comparisons with other methods, it has been demonstrated that the proposed framework can significantly reduce the number of optimization iterations (and thereby the number of FEM analyses required) by about one order of magnitude, while maintaining comparable solution quality in terms of the optimal value of the objective function and the resulting material layout. Despite increasing discretization resolutions and the inclusion of multiple materials—which significantly increase the number of design variables and introduce more inequality constraints—we observe consistency in solution quality and the number of optimization iterations required to reach the optimal solution. It has also been found that the minimum Young's modulus has the greatest impact on the conditioning of the optimization problem and is therefore considered as the primary parameter for relaxation. Relaxing it, along with a secondary parameter in multiple stages, could potentially offer greater advantages in achieving lower objective function values. The extent of improvement can vary depending on the specific problem, particularly its conditioning.

3 CLASSICAL PDE SOLVER FOR COMPLEX GEOMETRY

The linear systems of equations generated from the generalized moving least square (GMLS) discretization need to be solved properly to achieve scalability and efficiency. Krylov methods are usually used for solving large-scale linear systems of equations, for which robust preconditioners must be developed. Due to its meshless nature, there are no hierarchical geometric *meshes* in the GMLS discretization, making the development of a monolithic GMG method challenging. Thus, in previous works [75, 31], the algebraic multigrid (AMG) method was used in conjunction with the block-factorization-based preconditioners. In particular, since AMG methods are developed mainly for scalar Poisson-like problems, they are used to invert the diagonal blocks that appear in the block preconditioners. However, due to the complexity of the fluid-solid interaction problems and the limitation of block-factorization-based preconditioners, such AMG-based preconditioning strategies cannot ensure convergence and achieve scalability. Therefore, in this work, it proposes to use the geometric information of the adaptively refined GMLS nodes to build the hierarchical structure and develop a monolithic GMG-based preconditioner for solving the linear systems arising from the GMLS discretization.

This thesis considers the coupled dynamics of steady Stokes flow of incompressible fluid and freely moving solid bodies suspended in the fluid. Hence, the computational domain contains the fluid and N_s solid bodies. Denote $\Omega_f \subset \mathbb{R}^d$ as the sub-domain occupied by the fluid, Γ_0 as the outer wall boundary of the whole domain, and $\Gamma_n, n = 1, 2, \dots N_s$ as the boundaries of N_s freely moving solids in d-dimensional physical space. Assume $\Gamma_n \cap \Gamma_m = \emptyset, m, n = 0, 1, 2, \dots N_s$ always hold during the whole dynamics process, i.e., there is finite separation between any two boundaries. Each solid body is assumed to follow rigid-body dynamics, and its motion is tracked by its center of mass (COM) position $\mathbf{X}_n$ and

orientation $\boldsymbol{\Theta}_n$. Its translational and angular velocities are accordingly donated as $\dot{\mathbf{X}}_n$ and $\dot{\boldsymbol{\Theta}}_n$, respectively. The fluid flow is coupled to the motions of all solid bodies through the following steady Stokes problem:

$$\begin{cases} \dfrac{\nabla p}{\rho} - \nu \nabla^2 \mathbf{u} = \mathbf{f} & \forall \mathbf{x} \in \Omega_f \\ \nabla \cdot \mathbf{u} = 0 & \forall \mathbf{x} \in \Omega_f \\ \mathbf{u} = \mathbf{w} & \forall \mathbf{x} \in \Gamma_0 \\ \mathbf{u} = \dot{\mathbf{X}}_n + \dot{\boldsymbol{\Theta}}_n \times (\mathbf{x} - \mathbf{X}_n) & \forall \mathbf{x} \in \Gamma_n, n = 1, \dots, N_s \\ \mathbf{n} \cdot \dfrac{\nabla p}{\rho} - \nu \mathbf{n} \cdot \nabla^2 \mathbf{u} = \mathbf{n} \cdot \mathbf{f} & \forall \mathbf{x} \in \Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \cdots \cup \Gamma_{N_s}, \end{cases} \tag{3.1}$$

where ν is the kinematic viscosity of fluid; ρ is the density of the fluid; $\mathbf{f}$ denotes the body force exerted on the fluid; $\mathbf{w}$ is the velocity of the wall boundary; and $\mathbf{n}$ is the unit normal vector outward facing at boundary Γ.

For a divergence-free velocity field ($\nabla \cdot \mathbf{u} = 0$), Eq. (3.1) is equivalent to:

$$\begin{cases} \dfrac{\nabla p}{\rho} + \nu \nabla \times \nabla \times \mathbf{u} = \mathbf{f} & \forall \mathbf{x} \in \Omega_f \\ -\dfrac{\nabla^2 p}{\rho} = -\nabla \cdot \mathbf{f} & \forall \mathbf{x} \in \Omega_f \\ \mathbf{u} = \mathbf{w} & \forall \mathbf{x} \in \Gamma_0 \\ \mathbf{u} = \dot{\mathbf{X}}_n + \dot{\boldsymbol{\Theta}}_n \times (\mathbf{x} - \mathbf{X}_n) & \forall \mathbf{x} \in \Gamma_n, n = 1, \dots, N_s \\ \mathbf{n} \cdot \dfrac{\nabla p}{\rho} + \nu \mathbf{n} \cdot \nabla \times \nabla \times \mathbf{u} = \mathbf{n} \cdot \mathbf{f} & \forall \mathbf{x} \in \Gamma = \Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \cdots \cup \Gamma_{N_s}, \end{cases} \tag{3.2}$$

where the velocity identity $\nabla^2 \mathbf{u} = -\nabla \times \nabla \times \mathbf{u}$ is utilized for $\nabla \cdot \mathbf{u} = 0$. Solving Eq. (3.2) instead of Eq. (3.1) can avoid the saddle-point structure of the Stokes operator. However, it necessitates a numerical reconstruction of velocity faithful to the divergence-free constraint.

In Stokes limit, the inertia effect of solids can be neglected. Hence, each

solid body's translational and angular motions are subject to the force-free and torque-free constraints as:

$$\begin{cases} \int_{\Gamma_n} \boldsymbol{\sigma} \cdot \mathrm{d}\boldsymbol{\mathcal{A}} + \mathbf{f}_{e,n} = \mathbf{0} \\ \int_{\Gamma_n} (\mathbf{x} - \mathbf{X}_n) \times (\boldsymbol{\sigma} \cdot \mathrm{d}\boldsymbol{\mathcal{A}}) + \boldsymbol{\tau}_{e,n} = \mathbf{0}\,, \end{cases} \tag{3.3}$$

where $n = 1, \ldots, N_s$; $\boldsymbol{\sigma} = -p\mathbf{I} + \mu[\nabla\mathbf{u} + (\nabla\mathbf{u})^T]$ is the stress exerted by the fluid on the boundary of each solid body; $\mathbf{f}_{e,n}$ and $\boldsymbol{\tau}_{e,n}$ represent the external force and torque applied on each solid body, respectively.

Therefore, by solving Eqs. (3.2) and (3.3) concurrently as a monolithic system, each solid's translational and angular velocities $\{\dot{\mathbf{X}}_n, \dot{\boldsymbol{\Theta}}_n\}_{n=1,2,\ldots,N_s}$ as well as the fluid's velocity $(\mathbf{u})$ and pressure (p) fields are obtained at each instant time. Given $\{\dot{\mathbf{X}}_n, \dot{\boldsymbol{\Theta}}_n\}_{n=1,2,\ldots,N_s}$ and the initial value of $\{\mathbf{X}_n, \boldsymbol{\Theta}_n\}_{n=1,2,\ldots,N_s}$, the solid bodies' new positions and orientations are updated by invoking a temporal integrator. In this work, it adopts the 5^{th} order Runge-Kutta integrator with adaptive time stepping [23], also known as ODE45 [64], and implement it in the solver. To achieve adaptive time stepping, this integrator needs to specify an initial time step and relative error tolerance, which are set as $\Delta t_0 = 0.2s$ and 10^{-5}, respectively, in this work.

3.1 Generalized Moving Least Square Method

3.1.1 Basic Approximation

The fluid domain Ω_f is discretized by a set of collocation points (referred to as interior GMLS nodes). All the boundaries $\Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \cdots \cup \Gamma_{N_s}$ are discretized by another set of discrete points (referred to as boundary GMLS nodes), where the boundary conditions (BCs) are imposed. For a GMLS node at $\mathbf{x}_i$ and a given scalar function ψ evaluated at its neighbor domain:

$\psi_j = \psi(\mathbf{x}_j)$, a polynomial $\psi_h(\mathbf{x})$ of order m is sought to approximate ψ and its derivatives ($D^\alpha\psi$) at $\mathbf{x}_i$. That is, $\psi_h(\mathbf{x}) = \mathbf{P}^\intercal(\mathbf{x})\mathbf{c}^*$ with a polynomial basis $\mathbf{P}(\mathbf{x})$ and coefficient vector $\mathbf{c}^*$ such that the following weighted residual functional is minimized:

$$J(\mathbf{x}_i) = \sum_{j\in\mathcal{N}_{\epsilon_i}} [\psi_j - \mathbf{P}_i^\intercal(\mathbf{x}_j)\mathbf{c}_i]^2 W_{ij} \ . \tag{3.4}$$

For this quadratic programming optimization problem, its solution can be easily given as:

$$\mathbf{c}_i^* = \left(\sum_{k\in\mathcal{N}_{\epsilon_i}} \mathbf{P}_i(\mathbf{x}_k) W_{ik} \mathbf{P}_i^\intercal(\mathbf{x}_k)\right)^{-1} \left(\sum_{j\in\mathcal{N}_{\epsilon_i}} \mathbf{P}_i^\intercal(\mathbf{x}_j) W_{ij}\psi_j\right) \ . \tag{3.5}$$

Note that for $\psi \in \text{span}(\mathbf{P}(\mathbf{x}))$, this approximation ensures ψ to be exactly reconstructed. This property of polynomial reconstruction grants high-order accuracy achievable for the GMLS approximation by taking large m, e.g. $m = 4, 6$ [80]. An arbitrary α^{th} order derivative of the function can then be approximated by:

$$D^\alpha\psi(\mathbf{x}) \approx D_h^\alpha\psi_h(\mathbf{x}) \coloneqq (D^\alpha\mathbf{P}(\mathbf{X}))^\intercal\mathbf{c}^*. \tag{3.6}$$

The weight function involved in Eqs. (3.4)-(3.5) is defined as $W_{ij} = W(r_{ij})$ with $r_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|$ and $W(r) = 1 - \left(\frac{r}{\epsilon}\right)^4$ for $r < \epsilon$, or otherwise, $W(r) = 0$. Here, ϵ is the compact support, and hence, it is only necessary to include GMLS nodes within an ϵ-neighborhood of the ith GMLS node, i.e., $j \in \mathcal{N}_{\epsilon_i} = \{\mathbf{x}_j \text{ s.t. } \|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon_i\}$. Therefore, the approximation errors in both the interpolant and derivatives of ψ depend on the value of ϵ, or on the normalized support size $\epsilon/\Delta x$ if the characteristic nodal distance Δx within the support is fixed. $\epsilon/\Delta x$ must be large enough to ensure unisolvency over the reconstruction space and thereby a well-posed

solution to Eq. (3.4). Larger $\epsilon/\Delta x$ also leads to a more accurate least-square approximation. However, too large $\epsilon/\Delta x$ can yield very dense linear operators and ill-conditioned linear systems. Denote $N_{m,p}$ the minimal number of neighbors required to solve Eq. (3.4) for the polynomial basis $\mathbf{P}(\mathbf{x})$ of order m. In practice, to ensure a well-posed and accurate solution of Eq. (3.4), it determines the size of compact support ϵ_i for each node by taking the minimum value of $\epsilon_i/\Delta x_i$ such that $|\mathcal{N}_{\epsilon_i}| \geqslant 2^{d/2} N_{m,p}$, where $|\mathcal{N}_{\epsilon_i}|$ denotes the number of nodes in the set $\mathcal{N}_{\epsilon_i}$; d represents the physical dimension.

3.1.2 Divergence-free GMLS reconstruction for the velocity field

The GMLS approximation discussed above is applied to both the velocity and pressure fields. However, the polynomial basis used to approximate the velocity field $\mathbf{u}$ is chosen from the space of m^{th} order divergence-free vector polynomials $\mathbf{P}^{\text{div}}(\mathbf{x})$ to enforce compatibility with the divergence-free constraint on the velocity. Thus, the following polynomial reconstruction is built for $\mathbf{u}$ and for discretizing its gradient and curl-curl operators:

$$\begin{aligned} \mathbf{u}_h(\mathbf{x}_i) &= (\mathbf{P}_i^{\text{div}})^T(\mathbf{x}_i)\mathbf{c}_i^{\text{div}*} \\ \nabla_h \mathbf{u}_h(\mathbf{x}_i) &= (\nabla \mathbf{P}_i^{\text{div}})^T(\mathbf{x}_i)\mathbf{c}_i^{\text{div}*} \\ (\nabla \times \nabla \times)_h \mathbf{u}_h(\mathbf{x}_i) &= (\nabla \times \nabla \times \mathbf{P}_i^{\text{div}})^T(\mathbf{x}_i)\mathbf{c}_i^{\text{div}*} , \end{aligned} \tag{3.7}$$

where

$$\mathbf{c}_i^{\text{div}*} = \mathbf{M}_i^{\text{div}^{-1}} \left(\sum_{j \in \mathcal{N}_{\epsilon_i}} \mathbf{P}_i^{\text{div}}(\mathbf{x}_j)\mathbf{u}(\mathbf{x}_j) W_{ij} \right) , \tag{3.8}$$

with

$$\mathbf{M}_i^{\text{div}} = \sum_{j \in \mathcal{N}_{\epsilon_i}} \mathbf{P}_i^{\text{div}}(\mathbf{x}_j) W_{ij} (\mathbf{P}_i^{\text{div}})^T(\mathbf{x}_j) .$$

Let $x_\epsilon = \frac{x-x_i}{\epsilon_i}$, $y_\epsilon = \frac{y-y_i}{\epsilon_i}$ and $z_\epsilon = \frac{z-z_i}{\epsilon_i}$. The complete divergence-free polynomial basis $\mathbf{P}_i^{\text{div}}(\mathbf{x})$ of different orders m in both 2D ($d = 2$) and 3D ($d = 3$) are provided as follows. The basis When $d = 2$ and $m = 2$, the divergence-free vector polynomial basis $\mathbf{P}^{\text{div}}(\mathbf{x})$ at particle i is given by:

$$\mathbf{P}_i^{\text{div}}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & y_\epsilon & x_\epsilon & 0 & y_\epsilon^2 & x_\epsilon^2 & -2x_\epsilon y_\epsilon \\ 0 & 1 & x_\epsilon & 0 & -y_\epsilon & x_\epsilon^2 & 0 & -2x_\epsilon y_\epsilon & y_\epsilon^2 \end{bmatrix}^T ; \quad (3.9)$$

when $d = 2$ and $m = 4$, it is given by:

$$\mathbf{P}_i^{\text{div}}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & y_\epsilon & x_\epsilon & 0 & y_\epsilon^2 & x_\epsilon^2 & -2x_\epsilon y_\epsilon \\ 0 & 1 & x_\epsilon & 0 & -y_\epsilon & x_\epsilon^2 & 0 & -2x_\epsilon y_\epsilon & y_\epsilon^2 \end{bmatrix}$$

$$\begin{matrix} 0 & y_\epsilon^3 & x_\epsilon^3 & -3x_\epsilon y_\epsilon^2 & x_\epsilon^2 \frac{y-y_i}{\epsilon_i} \\ x_\epsilon^3 & 0 & -3x_\epsilon^2 \frac{y-y_i}{\epsilon_i} & y_\epsilon^3 & -x_\epsilon y_\epsilon^2 \end{matrix} \quad ; \quad (3.10)$$

$$\begin{matrix} 0 & y_\epsilon^4 & x_\epsilon^4 & -4x_\epsilon y_\epsilon^3 & x_\epsilon^3 \frac{y-y_i}{\epsilon_i} & x_\epsilon^2 y_\epsilon^2 \\ x_\epsilon^4 & 0 & -4x_\epsilon^3 \frac{y-y_i}{\epsilon_i} & y_\epsilon^4 & -x_\epsilon y_\epsilon^3 & -x_\epsilon^2 y_\epsilon^2 \end{matrix}\Bigg]^T$$

and when $d = 3$ and $m = 2$, it is given by:

$$\mathbf{P}_i^{\text{div}}(\mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & y_\epsilon & 0 & x_\epsilon & 0 & x_\epsilon \\ 0 & 1 & 0 & x_\epsilon & 0 & 0 & 0 & 0 & z_\epsilon & -y_\epsilon \\ 0 & 0 & 1 & 0 & x_\epsilon & 0 & y_\epsilon & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{matrix} x_\epsilon & 0 & 0 & y_\epsilon^2 & 0 & z_\epsilon^2 & 0 & x_\epsilon^2 & x_\epsilon^2 \\ 0 & x_\epsilon^2 & 0 & 0 & 0 & 0 & z_\epsilon^2 & -2x_\epsilon y_\epsilon & 0 \\ -z_\epsilon & 0 & x_\epsilon^2 & 0 & y_\epsilon^2 & 0 & 0 & 0 & -2x_\epsilon z_\epsilon \end{matrix} .$$

$$\begin{matrix} -2x_\epsilon y_\epsilon & 0 & -2x_\epsilon z_\epsilon & 0 & y_\epsilon z_\epsilon & 0 & 0 \\ y_\epsilon^2 & y_\epsilon^2 & 0 & -2y_\epsilon z_\epsilon & 0 & x_\epsilon z_\epsilon & 0 \\ 0 & -2y_\epsilon z_\epsilon & z_\epsilon^2 & z_\epsilon^2 & 0 & 0 & x_\epsilon y_\epsilon \end{matrix}^T \tag{3.11}$$

To impose the Dirichlet (no-slip) BCs in Eq. (3.2) for $\mathbf{u}$, the boundary GMLS nodes take the velocity of the corresponding boundary Γ_n that they belong to.

3.1.3 Staggered discretization of div-grad operator

To ensure compatibility and thereby numerical stability, the staggered GMLS discretization [76, 31] is employed for the div-grad operator of pressure p in Eq. (3.2). It builds upon each ϵ-neighborhood graph that plays as a local primal-dual complex for each GMLS node. It is referred as virtual cell, where a set of primal edges are constructed as: $\mathbf{E}_i = \{\mathbf{x}_i - \mathbf{x}_j | \mathbf{x}_j \in \mathcal{N}_{\epsilon_i}\}$, each associated with a midpoint $\mathbf{x}_{ij} = \frac{\mathbf{x}_i + \mathbf{x}_j}{2}$ and a virtual dual face at the midpoint and normal to the edge. The staggered GMLS discretization of the div-grad operator is then constructed from a topological gradient over primal edges and a GMLS divergence recovered from local virtual

dual faces. Thus, it has:

$$\begin{aligned} p_h(\mathbf{x}_i) =& \mathbf{P}_i^\intercal(\mathbf{x}_i)\mathbf{c}_i^* \\ \nabla_h p_h(\mathbf{x}_i) =& \frac{1}{2}(\nabla \mathbf{P}_i)^\intercal(\mathbf{x}_i)\mathbf{c}_i^* \\ \nabla_h^2 p_h(\mathbf{x}_i) =& \frac{1}{4}(\nabla^2 \mathbf{P}_i)^\intercal(\mathbf{x}_i)\mathbf{c}_i^* \ , \end{aligned} \tag{3.12}$$

where

$$\mathbf{c}_i^* = \mathbf{M}^{-1}\left(\sum_{j\in\mathcal{N}_{\epsilon_i}} \mathbf{P}_i(\mathbf{x}_{ij})W_{ij}(p_i - p_j)\right) \ , \tag{3.13}$$

with

$$\mathbf{M} = \left(\sum_{j\in\mathcal{N}_{\epsilon_i}} \mathbf{P}_i(\mathbf{x}_{ij})W_{ij}\mathbf{P}_i^\intercal(\mathbf{x}_{ij})\right) \ , \tag{3.14}$$

and the polynomial basis $\mathbf{P}$ can be the ϵ_i-scaled Taylor m^{th} order monomials; and the coefficient $\mathbf{c}_i^*$ is the solution of the quadratic program:

$$\mathbf{c}_i(\mathbf{x}) = \arg\min_{\mathbf{c}_i}\left\{\sum_{j\in\mathcal{N}_{\epsilon_i}} [(p_i - p_j) - \mathbf{P}_i^\intercal(\mathbf{x}_{ij})\mathbf{c}_i]^2 W_{ij}\right\} . \tag{3.15}$$

Note that pressure p is subject to an inhomogeneous Neumann BC in Eq. (3.2), written in the form of $\partial_{\mathbf{n}}|_\Gamma = g$. To impose this BC, it can be added to the quadratic program in Eq. (3.15) for the boundary GMLS nodes the following equality constraint:

$$\mathbf{n}_i \cdot \left[\frac{1}{2}(\nabla \mathbf{P}_i)^\intercal(\mathbf{x}_i)\mathbf{c}_i^*\right] = g(\mathbf{x}_i) \ , \qquad \mathbf{x}_i \in \Gamma \ .$$

In addition, a zero-mean constraint is imposed for p to ensure the uniqueness and physical consistency of the solution. Different from the previous work [31], it does not employ a Lagrangian multiplier for enforcing this zero-mean constraint in order to achieve scalability. Details are explained

in §3.3.4.

3.1.4 Adaptive refinement

In order to reduce computational cost and to recover optimal convergence in the presence of singularities governing lubrication effects, the computational domain and boundaries are discretized with a hierarchy of different resolutions based on an adaptive refinement algorithm [31].

At each time step, it starts from a uniform, coarse initial discretization resolution $\Delta x_i = \Delta x^0$, resulting in a set of GMLS nodes, Ω^0. Then, more GMLS nodes are added adaptively with refined Δx_i, leading to new sets Ω^I of GMLS nodes. In each refinement iteration, a four-stage procedure is followed:

$$\textbf{SOLVE} \rightarrow \textbf{ESTIMATE} \rightarrow \textbf{MARK} \rightarrow \textbf{REFINE} \,. \tag{3.16}$$

In **SOLVE** stage, Eq. (3.2) is numerically solved using the GMLS discretization discussed in §3.1 and the linear solver introduced in §3.2; both the velocity and pressure fields are updated. In **ESTIMATE** stage, the recovery-based *a posteriori* error estimator η_i^r is evaluated from Eq. (3.18) for all GMLS nodes. According to the value of the error estimator on each node, all nodes are sorted in descending order into a new sequence $\mathbf{x}_{i'}$. In **MARK** stage, a fraction of GMLS nodes with largest errors are selected and marked for refining. This fraction of GMLS nodes contribute to α percentage of the total recovered error, where α is a preset parameter and $\alpha = 0.8$ in the present work. In **REFINE** stage, the marked GMLS nodes are refined. For solving 2D problems, generally any marked interior GMLS node $\mathbf{x}_i \in \Omega_f$ is split into four nodes with smaller spacing, and a marked boundary node $\mathbf{x}_i \in \Gamma$ is split into two nodes. For solving 3D problems, any marked interior GMLS node $\mathbf{x}_i \in \Omega_f$ is split into eight nodes with smaller spacing; a marked boundary node on the wall boundary $\mathbf{x}_i \in \Gamma_0$

is split into four nodes; and the details about how to refine any marked boundary node on the surface of a moving solid $\mathbf{x}_i \in \Gamma_n$ $(n = 1, 2, \ldots, N_s)$ are provided in 3.1.6. Denote Δx_i as the original spacing (or resolution) of the marked nodes, the nodes newly generated by splitting have the spacing $\Delta x_{i_{new}} = \frac{1}{2}\Delta x_i$. The updated set of GMLS nodes at the end of **REFINE** stage is denoted as Ω^{I+1}. These four stages (3.16) are repeated iteratively at each time step until the total recovered error (Eq. (3.19)) is less than a preset tolerance, i.e,

$$\eta^r \leqslant \varepsilon_{tol} \,. \tag{3.17}$$

The recovery-based *a posteriori* error estimator is defined based on velocity gradient. Due to the lack of exact solutions in practical applications, true errors cannot be evaluated. Thus, the *recovered* error is used to estimate the true error, which measures the difference between the direct and recovered velocity gradients and can be practically determined in any application problem. More specifically, it is defined as:

$$\eta_i^r = \frac{\sum_{j\in\mathcal{N}_{\epsilon_i}} \|\mathbf{R}[\nabla\mathbf{u}]_j - \nabla\mathbf{u}_{h,i\to j}\|^2 V_j}{\sum_{j\in\mathcal{N}_{\epsilon_i}} V_j} \,. \tag{3.18}$$

Here, the recovered velocity gradient can be evaluated locally on each GMLS node as:

$$\mathbf{R}[\nabla\mathbf{u}]_i = \frac{1}{N_i} \sum_{j\in\mathcal{N}_{\epsilon_i}} \nabla_h \mathbf{u}_{h,j\to i} \,,$$

where $\nabla_h \mathbf{u}_{h,j\to i} = (\nabla \mathbf{P}_j^{\text{div}})^T(\mathbf{x}_i)\mathbf{c}_j^*$ is the velocity gradient reconstructed at $\mathbf{x}_j$ but evaluated at $\mathbf{x}_i$. In order to properly weigh the contributions from nodes with different discretization resolutions $\Delta\mathbf{x}_i$, a volumetric weight V_i is assigned to each node at $\mathbf{x}_i$ and defined as $V_i = \Delta x_i^d$, where d is the dimension of the problem's physical space. Then, the total recovered error

over all GMLS nodes is:

$$\eta^r = \frac{\sum_{\mathbf{x}_i \in \Omega} \eta_i^r V_i}{\sum_{\mathbf{x}_i \in \Omega} \|\nabla \mathbf{u}_{h,i}\|^2 V_i} \,. \tag{3.19}$$

For applications of dilute suspensions of solids, the above adaptive refinement procedure can start from an uniform, coarse initial discretization. However, for concentrate suspensions of solids, or when gaps between some solid boundaries are rather narrow, even smaller than Δx^0, the initial coarse discretization can result in none GMLS nodes allocated within the gaps. In that case, adaptive refinement would not take place within those gaps. Thus, a preprocessing step is evoked to guarantee GMLS nodes allocated everywhere throughout the entire computational domain including narrow gaps, before the adaptive refinement is conducted following the algorithm discussed above.

In this preprocessing step, it examines the ϵ-neighborhood of each boundary GMLS node and check if the following requirement is satisfied:

$$\text{For } \mathbf{x}_i \in \Gamma_n \text{ and } \forall j \in \mathcal{N}_{\epsilon_i},\ \ \mathbf{x}_j \notin \Gamma_m \ \text{ with } \ m \neq n \text{ and } m, n = 0, 1, \ldots, N_s \,. \tag{3.20}$$

This requirement ensures that each boundary GMLS node has enough interior GMLS nodes in its ϵ-neighborhood. If it is not satisfied, the corresponding boundary node and all its neighbor nodes are refined. Note that in this preprocessing step, the neighbor nodes of a boundary node can be within solid bodies.

After preprocessing and any iteration of refinement, a post-processing step is performed to enforce quasi-uniform discretization within any ϵ-neighborhood $\mathcal{N}_{\epsilon_i}$, because large difference in the discretization resolution within an ϵ-neighborhood would result in ill-conditioned GMLS approxi-

mation. Thus, for any ϵ-neighborhood that does not satisfy:

$$\frac{\max(\Delta x_j)}{\min(\Delta x_k)} \leqslant 2\,, \quad \text{for } \forall \mathbf{x}_i \in \Omega_f \cup \Gamma \text{ and } j,k \in \mathcal{N}_{\epsilon_i}\,, \tag{3.21}$$

it will mark and refine the nodes with the coarsest resolution in that ϵ-neighborhood, until the requirement in Eq. (3.21) is satisfied.

3.1.5 Numerical quadrature

With the GMLS discretization, a composite quadrature rule is employed for approximating the integrals in Eq. (3.3) as:

$$\begin{aligned}
&\int_{\Gamma_n} \boldsymbol{\sigma} \cdot \mathrm{d}\boldsymbol{\mathcal{A}} \\
\approx &\sum_{\mathbf{x}_i \in \Gamma_n} (-p_i \mathbf{I} + \nu[\nabla_h \mathbf{u}_{h,i} + (\nabla_h \mathbf{u}_{h,i})^\intercal]) \cdot (\Delta\mathcal{A}_i \mathbf{n}_i) \\
&\int_{\Gamma_n} (\mathbf{x} - \mathbf{X}_n) \times (\boldsymbol{\sigma} \cdot \mathrm{d}\boldsymbol{\mathcal{A}}) \\
\approx &\sum_{\mathbf{x}_i \in \Gamma_n} ((\mathbf{x}_i - \mathbf{X}_n) \times (-p_i \mathbf{I} + \nu[\nabla_h \mathbf{u}_{h,i} + (\nabla_h \mathbf{u}_{h,i})^\intercal])) \cdot (\Delta\mathcal{A}_i \mathbf{n}_i)\,,
\end{aligned} \tag{3.22}$$

where $\nabla_h \mathbf{u}_{h,i} = (\nabla \mathbf{P}_i^{\mathrm{div}})^\intercal(\mathbf{x}_i)\mathbf{c}_i^*$; $\Delta\mathcal{A}_i = \Delta x_i$ for 2D problems, and how to determine $\Delta\mathcal{A}_i$ in 3D is provided in §3.1.6. As shown in [31], this quadrature rule is sufficient for the entire numerical method to achieve high-order convergence.

3.1.6 Discretization and Adaptive Refinement in 3D for the Surfaces of Moving Solids

The boundaries of solid bodies in fluid-solid interactions can generally be curved surfaces or manifolds, so generating and refining point clouds on them need additional care. First of all, a tool to generate initial, uniform

point clouds in the curved surfaces (e.g., spherical surfaces) needs to be invoked. In this work, *DistMesh* [56] is adopted, which can generate a quasi-uniform point cloud with a given discretization resolution via triangulation. The point cloud generated by *DistMesh* with the initial resolution Δx^0 on the surface of each solid, Γ_n ($n = 1, 2, \ldots, N_s$), is denoted as $\mathcal{P}_n^0 = \{\mathcal{V}_n^0, \mathcal{E}_n^0, \mathcal{T}_n^0\}$, where $\mathcal{V}_n^0 = \{\mathbf{x}_i^0 \in \Gamma_n\}$ denotes the set of boundary GMLS nodes on Γ_n; $\mathcal{E}_n^0 = \{e_{ij}^0 = (\mathbf{x}_i^0, \mathbf{x}_j^0) \mid \mathbf{x}_i^0, \mathbf{x}_j^0 \in \mathcal{V}_n^0\}$ denotes the set of edges determined by *DistMesh* connecting two adjacent nodes; and $\mathcal{T}_n^0 = \{t_{ijk}^0 = (\mathbf{x}_i^0, \mathbf{x}_j^0, \mathbf{x}_k^0) \mid \mathbf{x}_i^0, \mathbf{x}_j^0, \mathbf{x}_k^0 \in \mathcal{V}_n^0; e_{ij}^0, e_{ik}^0, e_{jk}^0 \in \mathcal{E}_n^0\}$ represents the set of triangles formed from three adjacent nodes by *DistMesh*. Starting from $\mathcal{P}_n^0$, the nodes marked for refinement are refined. $\mathcal{P}_n^I$ denotes the point cloud on Γ_n resulted from the I-th iteration of adaptive refinement on $\mathcal{P}_n^{I-1}$.

Specifically, for a node $\mathbf{x}_i^{I-1} \in \mathcal{V}_n^{I-1}$ marked for refinement, a new resolution $\Delta x_{i_{new}} = \frac{1}{2}\Delta x_i$ is assigned to this node, and the midpoints on all edges connected with $\mathbf{x}_i^{I-1}$ are newly generated nodes to be added into $\mathcal{V}_n^I$. For an edge $e_{ij}^{I-1} \in \mathcal{E}_n^{I-1}$, its midpoint $\mathbf{x}_{ij}^{I-1}$ is determined as the intersection between Γ_n and the line connecting $\frac{\mathbf{x}_i^{I-1}+\mathbf{x}_j^{I-1}}{2}$ and the origin of the coordinate system defining the surface manifold of the solid body. In the end of refinement, such determined midpoints along with $\mathcal{V}_n^{I-1}$ make up $\mathcal{V}_n^I$. The edge between the midpoint $\mathbf{x}_{ij}^{I-1}$ and the marked node $\mathbf{x}_i^{I-1}$ is regarded as a new edge to be added into $\mathcal{E}_n^I$. If the node $\mathbf{x}_j^{I-1}$ is also marked for refinement, the edge between the midpoint $\mathbf{x}_{ij}^{I-1}$ and $\mathbf{x}_j^{I-1}$ is also added into $\mathcal{E}_n^I$, but the original edge e_{ij}^{I-1} is removed from $\mathcal{E}_n^I$. As for the triangles, when all the three nodes in t_{ijk}^{I-1} are marked for refinement, they along with the midpoints on the three edges form four new triangles to be added into $\mathcal{T}_n^I$, but t_{ijk}^{I-1} is removed from $\mathcal{T}_n^I$. As such, a clear derivation relation between $\mathcal{P}_n^I$ and $\mathcal{P}_n^{I-1}$ is still retained and can be utilized in the interpolation and restriction operations.

For each node $\mathbf{x}_i^I \in \mathcal{V}_n^I$, $\Delta\mathcal{A}_i$ in Eq. (3.22) is evaluated from the areas

of the triangles $t^I_{ijk} \in \mathcal{T}^I_n$ connected to the node. The area for each such triangle is denoted as $\mathcal{A}^I_{ijk}$. $\Delta\mathcal{A}_i$ is then calculated as:

$$\Delta\mathcal{A}_i = \frac{1}{3} \sum_{\{j,k|t^I_{ijk}\}} \mathcal{A}^I_{ijk} \tag{3.23}$$

3.2 Geometric Multigrid Preconditioner

Solving the linear systems resulting from discretization dominates the entire computational cost in simulations. Therefore, the main focus of this work is to design a scalable preconditioner for the Krylov method to solve the linear systems generated in the **SOLVE** stage. A simple preconditioner, such as the Gauss-Seidel preconditioner, cannot ensure convergence and is not effective in practice. In the previous work [75, 31], a block-factorization-based preconditioner based on decoupling of the velocity and pressure fields was employed, and AMG methods were applied for inverting each block. For solving benchmark smaller-scale fluid-solid interaction problems [75, 31], such a linear solver can work reasonably well. However, as the number of solid bodies increases, e.g., in the examples considered in the present work, the performance of such a block preconditioner deteriorates. One major reason is that the velocity block used in the block-factorization-based preconditioner consists of unknowns related to both the fluid and solid bodies. In particular, it combines the discretized curl-curl operator in Eq. (3.2) for the fluid with the discretized integrals in Eq. (3.22) for the solid bodies. In addition, the inclusion of many solid bodies results in a very irregular shape for the overall computational domain. All of these make it difficult for an AMG method to find a proper coarsening. Consequently, it fails to converge for the velocity block and strongly affects the overall performance of the block-factorization-based preconditioner.

This section introduces a monolithic GMG method to build a preconditioner for the Krylov linear solver such as GMRES to solve the linear systems resulted from GMLS discretization. The new multigrid method utilizes the hierarchical sets of GMLS nodes generated during the adaptive refinement. Therefore, no coarsening is needed, avoiding the major difficulty that the AMG method previously encountered. Using the geometric information of these hierarchical sets of the GMLS nodes, it can construct interpolation operators based on the GMLS approximation and restriction operators based on averaging. Furthermore, a two-stage smoother is developed based on the splitting between the fluid domain and the solid bodies. Finally, the interpolation/restriction operators and the smoother are used in a V-cycle fashion to define the overall monolithic GMG preconditioner.

It first discusses about the block structure of the resultant linear system in §3.2.1. Then, it briefly reviews the required operations and entire process of multigrid preconditioning in §3.2.2. In §3.2.3, it explaines how to construct the interpolation and restriction operators. In §3.2.4, the smoothers designed based on physics splitting is presented. The entire monolithic GMG method introduced in this work is finally summarized in §3.2.5.

3.2.1 Block structure of the linear system

After the governing equations (3.2)-(3.3) are discretized by the GMLS discretization, the resulting linear system has the following block structure:

$$\mathbf{A}\chi = \mathbf{y} \quad \text{with} \quad \mathbf{A} = \begin{bmatrix} \mathbf{K} & \mathbf{G} & \mathbf{C} \\ \mathbf{B} & \mathbf{L} & \\ \mathbf{D} & \mathbf{T} & \end{bmatrix}, \quad \chi = \begin{bmatrix} \mathbf{u} \\ p \\ \dot{\mathbf{X}} \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} \mathbf{f}_{tot} \\ g \\ \mathbf{f}_s \end{bmatrix}. \tag{3.24}$$

Here, $\dot{\mathbf{X}}$ includes all solids' both translational and rotational velocities, i.e., $\dot{\mathbf{X}} = \{\dot{\mathbf{X}}_n, \dot{\boldsymbol{\Theta}}_n\}_{n=1,2,\dots,N_s}$. $\mathbf{K}$ corresponds to the discretized curl-curl operator

$\nabla \times \nabla \times$ of velocity in Eq. (3.2). It is noted that the curl-curl operator for a divergence-free polynomial basis is equivalent to Laplacian. $\mathbf{L}$ corresponds to the discretized Laplacian operator ∇^2 of pressure in Eq. (3.2), which is obtained from the staggered discretization of div-grad operator. As such, the nonzero diagonal blocks in $\mathbf{A}$, i.e., $\mathbf{K}$ and $\mathbf{L}$, are all discretized Laplacian operators. While $\mathbf{B}$ denotes the contribution from the $\nu\nabla \times \nabla \times$ operator in the inhomogeneous Neumann BC in Eq. (3.2), $\mathbf{G}$ represents the discretized $\frac{1}{\rho}\nabla$ operator. For the discretized force-free and torque-free constraints of Eq. (3.3), $\mathbf{D}$ denotes the contribution from viscous stress with velocity gradient; $\mathbf{T}$ denotes the contribution from pressure. $\mathbf{C}$ contains the velocity constraints (no-slip BCs) from each solid's kinematics on their boundaries in Eq. (3.2). $\mathbf{f}_{tot}$ combines the body force $\mathbf{f}$ exerted in fluid and the velocity $\mathbf{w}$ on the wall boundary. g contains $\nabla \cdot \mathbf{f}$ and $\mathbf{n} \cdot \mathbf{f}$ in Eq. (3.2). Finally, $\mathbf{f}_s$ represents the external force and torque applied on the solid bodies, such that $\mathbf{f}_s = [-\mathbf{f}_{e,n} \ -\boldsymbol{\tau}_{e,n}]^T_{n=1,2,\ldots,N_s}$.

From its block structure, it is seen that $\mathbf{A}$ is neither symmetric nor positive definite. The strong coupling between the fluid and solid DOFs deteriorates the conditioning of $\mathbf{A}$. Thus, the linear system in Eq. (3.24) is challenging to solve in practice. Furthermore, since enough neighbor nodes are needed in GMLS discretization, the matrix $\mathbf{A}$ is dense. Therefore, designing a robust and efficient linear solver is crucial for achieving scalability in practical simulations.

3.2.2 Multigrid methods

Most relaxation-type iterative solvers, like Gauss-Seidel (GS), converge slowly for linear systems with fine resolution discretized from partial differential equations, e.g., (3.2). The main reason is that the convergence rates of different error components can vary significantly, if it decomposes the error using the eigenvectors of the linear system, e.g., $\mathbf{A}$ in Eq. (3.24). The components with the eigenvectors corresponding to large eigenvalues

(in terms of magnitude) are usually referred to as high-frequency components of the error; the components with the eigenvectors corresponding to small eigenvalues are called low-frequency components of the error. In general, for relaxation-type iterative solvers, the high-frequency components converge to zero quickly independent of the discretization resolution. However, the low-frequency components converge slowly, getting worse when the resolutions are refined and hence slowing down the overall convergence. On the other hand, low-frequency components of the error on fine resolutions can be well approximated on coarse discretization and hence become high-frequency error components on coarse resolutions [15]. This fact motivates the idea of moving to a coarser resolution to eliminate the low-frequency error components and leads to a multi-grid approach known as the multigrid method [15, 16].

Multigrid methods exploit a discretization with different resolutions to obtain optimal convergence rate and hence are naturally compatible with adaptive h-refinement. The operations of going back and forth between coarse and fine resolutions are called the restriction and interpolation, respectively. To minimize the approximation errors across different resolution levels, a smoothing procedure is usually executed before the restriction operation, and another smoothing step is applied after the interpolation operation. This report follows the V-cycle for building the multigrid preconditioner.

In the monolithic setting, the interpolation operators $\mathfrak{I}^I$, the restriction operators $\mathcal{R}^I$, and the smoothers (i.e., subroutine Smooth$(\mathbf{A}, \mathbf{y})$) need to be carefully designed to properly handle the coupling of fluid and solid-body DOFs. The adaptive refinement yields a hierarchical series of node sets Ω^I, $I = 0, 1, 2, \ldots$. It can utilize the hierarchical structures between node sets to construct the corresponding interpolation $\mathfrak{I}^I$ and restriction $\mathcal{R}^I$ operators for each node set Ω^I.

Algorithm 3 V-cycle process: V-Cycle($\mathbf{A}^I, \mathbf{y}^I$)

Input: Coefficient matrix $\mathbf{A}^I$ and right-hand side (RHS) vector $\mathbf{y}^I$
Output: Approximate solution χ^I

if $I == 0$ **then**
  Solve: $\mathbf{A}^I\chi^I = \mathbf{y}^I$
else
  Pre-smooth: $\chi^I = \text{Smooth}(\mathbf{A}^I, \mathbf{y}^I)$
  Compute residual: $\mathbf{r}^I = \mathbf{y}^I - \mathbf{A}^I\chi^I$
  Restrict: $\mathbf{y}^{I-1} = \mathcal{R}^I\mathbf{r}^I$
  Apply V-cycle recursively: $\mathbf{z}^{I-1} = \text{V-Cycle}(\mathbf{A}^{I-1}, \mathbf{y}^{I-1})$
  Interpolate: $\chi^I = \chi^I + \mathcal{I}^I\mathbf{z}^{I-1}$
  Post-smooth: $\chi^I = \chi^I + \text{Smooth}(\mathbf{A}^I, \mathbf{y}^I - \mathbf{A}^I\chi^I)$
end if

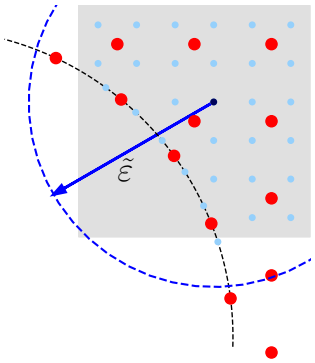
Return: χ^I

3.2.3 Interpolation and restriction operators

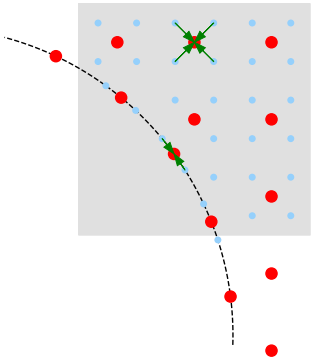
In mesh-based discretization methods, e.g. finite element method, the interpolation/restriction operators can be constructed from the nested function spaces across different levels of meshes. In this work, instead of using function spaces, it builds the interpolation and restriction operators through local approximations, i.e., approximating any scalar or vector field locally from a set of neighbor nodes. More specifically, if the fine-level nodes Ω^F as the "target" nodes and the coarse-level nodes Ω^C are taken as the "source", by constructing GMLS approximation (§**??**) on the target from the source, it defines the interpolation operator. If the target are coarse-level nodes, and the source are fine-level nodes, then a local averaging approximation on the target from the source builds the restriction operator. An illustrative description about the interpolation and restriction is provided in Fig. 3.1.

Thus, the interpolation operator for the pressure field is constructed as:

$$p(\mathbf{x}_i^F) = \mathbf{P}_i^T(\mathbf{x}_i^F)\mathbf{c}_i^* \,, \tag{3.25}$$



(a) The interpolation operator at a fine-level node $\mathbf{x}_i^F$ (highlighted in dark blue) is constructed as the GMLS approximation from the coarse-level nodes (red) within the fine-level node's $\tilde{\epsilon}$-neighborhood.



(b) The restriction operator at a coarse-level node $\mathbf{x}_i^C$ (red) is constructed as averaging over its own child nodes (light blue with green arrows) generated in **REFINE** stage during an iteration of adaptive refinement.

Figure 3.1: Schematic of the interpolation and restriction. Coarse-level nodes are displayed in red and fine-level nodes are in blue. The black dashed lines indicate part of a solid body's boundary.

with

$$\mathbf{c}_i^* = \mathbf{M}_i^{-1} \left(\sum_{j \in \mathcal{N}_{\tilde{\epsilon}_i}} \mathbf{P}_i(\mathbf{x}_j^C) p(\mathbf{x}_j^C) W_{ij} \right) , \tag{3.26}$$

and

$$\mathbf{M}_i = \sum_{j \in \mathcal{N}_{\tilde{\epsilon}_i}} \mathbf{P}_i(\mathbf{x}_j^C) W_{ij} \mathbf{P}_i^\intercal(\mathbf{x}_j^C), \;\; \mathbf{x}_i^F \in \Omega^F, \;\; \mathbf{x}_j^C \in \Omega^C , \tag{3.27}$$

where the polynomial basis $\mathbf{P}$ is the $\tilde{\epsilon}$-scaled Taylor monomials. For velocity, the interpolation operator is built by directly using the divergence-free reconstruction space as follows:

$$\mathbf{u}(\mathbf{x}_i^F) = (\mathbf{P}_i^{div})^\intercal(\mathbf{x}_i^F) \mathbf{c}_i^{div*} , \tag{3.28}$$

with

$$\mathbf{c}_i^{div*} = \mathbf{M}_i^{div^{-1}} \left(\sum_{j \in \mathcal{N}_{\tilde{\epsilon}_i}} \mathbf{P}_i^{div}(\mathbf{x}_j^C) \mathbf{u}(\mathbf{x}_j^C) W_{ij} \right) , \tag{3.29}$$

and

$$\mathbf{M}_i^{div} = \sum_{j \in \mathcal{N}_{\tilde{\epsilon}_i}} \mathbf{P}_i^{div}(\mathbf{x}_j^C) W_{ij} (\mathbf{P}_i^{div})^\intercal(\mathbf{x}_j^C), \;\; \mathbf{x}_i^F \in \Omega^F, \;\; \mathbf{x}_j^C \in \Omega^C . \tag{3.30}$$

As such, it ensures the divergence-free constraint for velocity is consistently preserved during interpolation across different fine/coarse levels of nodes. As is well-known [63], for solving the incompressible Stokes equations, interpolations that maintain the divergence-free property are crucial for developing an efficient GMG preconditioner. In the above GMLS approximations for constructing the interpolation operator, $\tilde{\epsilon}_i$ for each fine-level node needs to be large enough such that sufficient coarse-level nodes are included in each $\tilde{\epsilon}$-neighborhood to ensure unisolvency over the reconstruction space and a well-posed solution to the weighted least square

optimization.

For the restriction operator, since the matrix $\mathbf{A}$ in the linear system (3.24) is non-symmetric, it is unnecessary to require the restriction operator to be the transpose of the interpolation operator. Hence, it constructs the restriction operation based on h-refinement. A coarse-level node $\mathbf{x}_i^C \in \Omega^C$ marked in the **MARK** stage of the adaptive refinement algorithm is called a "parent" node. The newly generated nodes (within the fluid domain or on solid boundaries) in the **REFINE** stage are called the "child nodes" corresponding to their parent node $\mathbf{x}_i^C$. As a result, an interior parent node generally has four child nodes, and a boundary parent node has two child nodes in 2D, or eight child nodes for an interior parent node and four child nodes for a boundary parent node in 3D. The approximation at a parent node is given by averaging the field values from its child nodes, which provides the restriction operator. By such, the matrix corresponding to the restriction operator is much sparser than that of the interpolation operator, leading to cheaper cost for matrix multiplication during the restriction operations.

For the blocks in $\mathbf{A}$ related to solid DOFs, the interpolation and restriction operators are simply identity matrices. Assembled together, the interpolation and restriction operators can be summarized as:

$$\begin{aligned}
\begin{bmatrix} \mathbf{u}^{I+1} \\ p^{I+1} \\ \dot{\mathbf{X}} \end{bmatrix} &= \mathcal{I}^I \begin{bmatrix} \mathbf{u}^I \\ p^I \\ \dot{\mathbf{X}} \end{bmatrix} = \begin{bmatrix} \mathcal{I}_{\mathbf{u}} & & \\ & \mathcal{I}_p & \\ & & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}^I \\ p^I \\ \dot{\mathbf{X}} \end{bmatrix}, \\
\begin{bmatrix} \mathbf{u}^I \\ p^I \\ \dot{\mathbf{X}} \end{bmatrix} &= \mathcal{R}^I \begin{bmatrix} \mathbf{u}^{I+1} \\ p^{I+1} \\ \dot{\mathbf{X}} \end{bmatrix} = \begin{bmatrix} \mathcal{R}_{\mathbf{u}} & & \\ & \mathcal{R}_p & \\ & & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}^{I+1} \\ p^{I+1} \\ \dot{\mathbf{X}} \end{bmatrix},
\end{aligned} \tag{3.31}$$

where the super index I corresponds to the node set Ω^I resulted from the I-th iteration of adaptive refinement; while $\mathcal{I}_{\mathbf{u}}$ denotes the interpolation for velocity, $\mathcal{I}_p$ represents the interpolation for the pressure field; $\mathbf{I}$ is an

identity matrix; $\mathcal{R}_p$ is the restriction operator for a scalar valued function (e.g., pressure); $\mathcal{R}_{\mathbf{u}} = \text{diag}(\mathcal{R}_p, \cdots, \mathcal{R}_p)$ contains d copies of $\mathcal{R}_p$ on the diagonal and is the restriction operator for velocity (a d-dimensional vector field).

3.2.4 Smoother based on physics splitting

The design of the smoother is introduced in this subsection, i.e., the subroutine Smooth$(\mathbf{A}, \mathbf{y})$ used in Algorithm 3. As discussed, it is a relaxation-type iterative method that can efficiently smooth the high-frequency components of the error. For solving a fluid-solid interaction problem, it needs the smoother to handle the high-frequency components of the errors for the fluid part and solid part, respectively, as well as the strong coupling between them.

Eqs. (3.2)-(3.3) inherently state two types of physics: One is Stokesian flow in a confined space, and the other describes the dynamics of several rigid solid bodies undergoing external loads as well as drags exerted by surrounding Stokesian flow. The coupling of different types of physics inspires the design of the physics-based smoother, which contains two stages. The first stage takes care of the fluid DOFs, and the second handles the solid bodies as well as their neighboring fluid nodes. The details are presented as follows.

3.2.4.1 Smoother for the fluid DOFs

The submatrix

$$\mathbf{F} = \mathcal{B}_F \mathbf{A} \mathcal{B}_F^T = \begin{bmatrix} \mathbf{K} & \mathbf{G} \\ \mathbf{B} & \mathbf{L} \end{bmatrix} \tag{3.32}$$

is considered as the fluid part of the system. Here, $\mathcal{B}_F$ is a Boolean matrix, which picks the fluid field variables, $\mathbf{u}$ and p, out of the whole unknown vector. $\mathbf{F}$ contains all the field variables (velocity and pressure) directly

related to the interior GMLS nodes. The diagonal part $\mathbf{K}$ and $\mathbf{L}$ denotes the curl-curl operator onto the velocity field and the Laplacian operator onto the pressure field, respectively. As mentioned in §3.2.1, the curl-curl operator $\mathbf{K}$ is equivalent to the Laplacian operator on the divergence-free polynomial basis. Therefore, the diagonal blocks of the fluid part matrix $\mathbf{F}$ are all Laplacian-type operators. Further, as seen in the governing equation (3.2), there exists a strong coupling between the velocity and pressure fields. It is suggested using a node-wise Gauss-Seidel (GS) smoother. In 2D, all field values are organized into a 3×3 sub-matrix for each GMLS node. In 3D, all field values are organized into a 4×4 sub-matrix for each GMLS node. Therefore, it would include its diagonal entities related to the two Laplacian operators and the discretized coupling terms at the node in the smoother. From the numerical experiments, the node-wise smoother outperforms the normal entry-wise one, and inverting small dense matrices does not affect the scalability of the smoother. This node-wise GS smoother is denoted as $\mathbf{S}_{\mathbf{F}}$ in Algorithm 4.

3.2.4.2 Smoother for the solid bodies

Now consider the sub-matrix directly related to each solid body and its neighbor fluid nodes. The submatrix

$$\mathbf{N}_n = \mathcal{B}_{N_n} \mathbf{A} \mathcal{B}_{N_n}^T = \begin{bmatrix} \mathbf{K}_n & \mathbf{G}_n & \mathbf{C}_n \\ \mathbf{B}_n & \mathbf{L}_n & \\ \mathbf{D}_n & \mathbf{T}_n & \end{bmatrix} \tag{3.33}$$

corresponds to the n-th solid body and its neighbor fluid nodes. It is a square matrix and contains discretized drag force and torque exerted by the fluid. Each $\mathbf{N}_n$ is constructed by the following procedure. First, for each solid body, it is constructed as an index set Q_n such that

$$Q_n = \{i, j \mid \mathbf{x}_i \in \Gamma_n,\ j \in \mathcal{N}_{\epsilon_i}\}, \quad n = 1, 2, \ldots, N_s\ . \tag{3.34}$$

Figure 3.2 illustrates the construction of Q_n, where the green and red nodes are the interior GMLS (fluid) nodes; the blue nodes are the GMLS nodes on Γ_n (the boundary of the n-th solid body). Since the red nodes

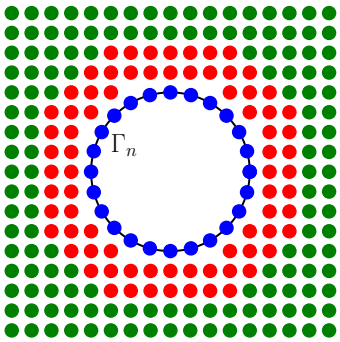


Figure 3.2: Illustration of construction of Q_n. Blue nodes denote the boundary nodes on Γ_n. Red nodes represent the nodes near the boundary Γ_n and contribute to the force and torque terms to the n-th solid. The green nodes denote the normal interior GMLS nodes.

.

are within the ϵ-neighborhood of the blue nodes, Q_n includes the indices of the nodes rendered in blue and red in Figure 3.2 but excludes the nodes rendered in green. Second, the Boolean matrix $\mathcal{B}_{N_n}$ is formed from Q_n, which picks the variables related to the n-th solid body and its neighbor fluid nodes out of the whole unknown vector. The submatrix $\mathbf{N}_n$ is then built according to Eq. (3.33). Note that there is no intersection between most Q_n for $n = 1, 2, \ldots, N_s$, especially when after several iterations of adaptive refinement, there are plenty of neighbor fluid nodes between any two closely contacting solid bodies. Thus, the parallel scalability can still be ensured.

Once all sub-matrices $\mathbf{N}_n$ are constructed, a Schwarz-type relaxation method is used to build the smoother. The resulting additive Schwarz-type

smoother can be defined as $\mathbf{S}_{\mathbf{N}}^{\text{exact}} := \sum_{n=1}^{N_s} \mathcal{B}_{N_n}^{\intercal} \mathbf{N}_n^{-1} \mathcal{B}_{N_n}$. Exactly inverting $\mathbf{N}_n$ could be expensive. Therefore, a Schur complement approach is used to invert it approximately, i.e.,

$$\widetilde{\mathbf{N}}_n^{-1} = \begin{bmatrix} \mathbf{I} & -\widetilde{\mathbf{F}}_n^{C^{-1}} \begin{bmatrix} \mathbf{C}_n \\ \mathbf{0} \end{bmatrix} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{F}}_n^{C^{-1}} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{\Psi}}_n^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\begin{bmatrix} \mathbf{D}_n & \mathbf{T}_n \end{bmatrix} \widetilde{\mathbf{F}}_n^{C^{-1}} & \mathbf{I} \end{bmatrix}, \quad (3.35)$$

where $\widetilde{\mathbf{F}}_n^{C^{-1}}$ approximates the inverse of the submatrix

$$\mathbf{F}_n^C = \begin{bmatrix} \mathbf{K}_n & \mathbf{G}_n \\ \mathbf{B}_n & \mathbf{L}_n \end{bmatrix}.$$

In the numerical experiments, one iteration of the node-wise GS smoother is used to define $\widetilde{\mathbf{F}}_n^{C^{-1}}$. In addition, the approximated Schur complement $\widetilde{\mathbf{\Psi}}_n$ is given as:

$$\widetilde{\mathbf{\Psi}}_n = \begin{bmatrix} \mathbf{D}_n & \mathbf{T}_n \end{bmatrix} \text{block-diag}^{-1}(\mathbf{F}_n^C) \begin{bmatrix} \mathbf{C}_n \\ \mathbf{0} \end{bmatrix}.$$

Here, block-diag$^{-1}(\cdot)$ denotes the diagonal block inversion of a matrix, which takes a 3×3 or 4×4 sub-matrix for each GMLS node in $\mathbf{F}_n^C$ and inverts those sub-matrices. Since the size of matrix $\widetilde{\mathbf{\Psi}}_n$ is quite small, only 3×3 in 2D and 6×6 in 3D, a direct solver is applied whenever the inversion of $\widetilde{\mathbf{\Psi}}_n$ is needed. Finally, for the solid bodies, the overall additive Schwarz-type smoother using the approximate Schur complement approach is given by:

$$\mathbf{S}_{\mathbf{N}} := \sum_{n=1}^{N_s} \mathcal{B}_{N_n}^{\intercal} \widetilde{\mathbf{N}}_n^{-1} \mathcal{B}_{N_n}. \quad (3.36)$$

Given the two smoothers $\mathbf{S}_{\mathbf{F}}$ and $\mathbf{S}_{\mathbf{N}}$, it connects them through an overlapping multiplicative Schwarz approach and hence establish the

proposed two-stage smoothing scheme. Algorithm 4 summarizes the established overall smoother.

Algorithm 4 Smoother: Smooth$(\mathbf{A}, \mathbf{y})$

Input: Coefficient matrix $\mathbf{A}$ and RHS vector $\mathbf{y}$
Output: Relaxed solution χ

1: Initialize $\chi = \mathbf{0}$
2: Perform k steps of node-wise GS smoother on the fluid DOFs:
3: **for** $i = 1, \cdots, k$ **do**
4: $\quad \chi \leftarrow \chi + \mathcal{B}_F^T \mathbf{S_F} \mathcal{B}_F (\mathbf{y} - \mathbf{A}\chi)$
5: **end for**
6: Apply the additive Schwarz-type smoother as in Eq. (3.36) for the solid bodies:

$$\chi \leftarrow \chi + \mathbf{S_N}(\mathbf{y} - \mathbf{A}\chi), \quad \text{where } \mathbf{S_N} = \sum_{n=1}^{N_s} \mathcal{B}_{N_n}^T \widetilde{\mathbf{N}}_n^{-1} \mathcal{B}_{N_n}$$

7: **Return**: χ

3.2.5 Linear solver summary

Given the constructed interpolation/restriction operators (Eq. (3.31)) and smoothers (Algorithm 4), it follows the V-cycle process in Algorithm 3 to build the entire monolithic GMG preconditioner. Since the coefficient matrix $\mathbf{A}$ is non-symmetric, the GMRES method is employed as the Krylov iterative solver for solving Eq. (3.24). In each GMRES iteration, it calls once the multigrid preconditioning, i.e., Algorithm 3. With the proposed monolithic GMG preconditioner, it aims to ensure the convergence of the linear solver and to optimize the scaling of the number of GMRES iterations required with respect to the numbers of solid bodies and total DOFs.

3.3 Parallel Implementation

To solve large-scale fluid-solid interaction problems, the parallel implementation of the proposed monolithic GMG preconditioner is developed. It aims to achieve the parallel scalability of the numerical solver.

3.3.1 Domain decomposition and neighbor search

All the GMLS nodes, $\forall \mathbf{x}_i \in \Omega^I$, yield after each iteration I of adaptive refinement, are evenly distributed into N_c sets, where N_c is the number of CPU cores invoked in the simulation. Each set of nodes, denoted as Ω^I_k ($k = 1, 2, \ldots, N_c$), is then allocated into one core, following the Recursive Coordinate Bisection (RCB) method [72]. In each core, `Compadre` [43] is used to generate the coefficients resulted from the GMLS discretization for the node-set Ω^I_k. This way of domain decomposition guarantees that the nodes in the same set Ω^I are evenly distributed among CPU cores and spatially clustered on each core, and hence balances the workload and minimizes the communication required between cores for solving the linear system in Eq. (3.24).

After domain decomposition, a ghost node set $\mathcal{G}^I_{k_\xi}$ is determined according to the order (m) of GMLS discretization and the spatial locations of the GMLS nodes in the node set Ω^I_k, as:

$$\mathcal{G}^I_{k_\xi} = \{\mathbf{x}_j \,|\, \|\mathbf{x}_j - \mathbf{x}_i\| < \xi \Delta x_i,\ \forall\, \mathbf{x}_i \in \Omega^I_k,\ \mathbf{x}_j \in \Omega^I\}\,,\ k = 1, 2, \ldots, N_c\,,\ \xi = \xi(m)\,,$$

where ξ is a function of m and $\xi = 4m$ is used in the numerical tests; Δx_i denotes the discretization resolution of node $\mathbf{x}_i$. Collecting this ghost node set only calls one communication between all cores. Once the collection is done, neighbor search for each node in Ω^I_k is performed by calling `nanoflann` [13], which is a function building KD-trees [12] and finds all nodes in the ϵ-neighborhood of any node $\mathbf{x}_i \in \Omega^I_k$ from the ghost node

set $\mathcal{G}^{I}_{k_\xi}$, i.e.,

$$\mathcal{N}_{\epsilon_i} = \{j \mid \|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon_i,\ \forall\, \mathbf{x}_i \in \Omega^{I}_{k},\ \mathbf{x}_j \in \mathcal{G}^{I}_{k_\xi}\}\,.$$

$\mathcal{N}_{\epsilon_i}$ hence provides the neighbor list needed in the GMLS discretization that leads to the coefficient matrix $\mathbf{A}$ of the linear system in Eq. (3.24).

Note that when building the interpolation operator in §3.2.3, GMLS approximation is needed, for which it also needs to build the neighbor list. Different from the above, the neighbor nodes of $\mathbf{x}_i$ in this GMLS approximation are not in the same node set as $\mathbf{x}_i$. Thus, for that purpose the ghost node set is determined as:

$$\widetilde{\mathcal{G}}^{I}_{k_\xi} = \{\mathbf{x}_j \mid \|\mathbf{x}_j - \mathbf{x}_i\| < \xi\Delta x_i,\ \forall \mathbf{x}_i \in \Omega^{I+1}_{k},\ \mathbf{x}_j \in \Omega^{I}\}\,,\ \ k = 1, 2, \ldots, N_c\,,$$

where $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the node sets yield from different iterations of adaptive refinement, respectively. And then the neighbor list is built. This time, `nanoflann` finds all nodes in the ϵ-neighborhood of any node $\mathbf{x}_i \in \Omega^{I+1}_{k}$ in the ghost node set $\widetilde{\mathcal{G}}^{I}_{k_\xi}$. Thus, it has:

$$\widetilde{\mathcal{N}}_{\epsilon_i} = \{j \mid \|\mathbf{x}_i - \mathbf{x}_j\| < \epsilon_i,\ \forall \mathbf{x}_i \in \Omega^{I+1}_{k},\ \mathbf{x}_j \in \widetilde{\mathcal{G}}^{I}_{k_\xi}\}\,,$$

which provides the neighbor list for the GMLS approximation needed to construct the interpolation operator.

3.3.2 Data storage

To reduce the data storage, all DOFs related to the solid bodies can be stored in a single CPU core (e.g., the last core). Thus, following the convention of PETSc, the sub-matrices $\mathbf{D}$ and $\mathbf{T}$ reside in the last rows of the matrix $\mathbf{A}$. However, the number of nonzero entities in $\mathbf{D}$ and $\mathbf{T}$ would increase with the inclusion of more solid bodies. As a result, the parallel scalability deteriorates as the number of solid bodies increases. Therefore, it spreads

the whole storage of data and workload related to $\mathbf{D}$ and $\mathbf{T}$ among all cores as follows. When assembling $\mathbf{D}$ and $\mathbf{T}$, each core would go through the GMLS nodes stored in it to generate its local $\mathbf{D}_i$ and $\mathbf{T}_i$. Accordingly, any matrix-vector multiplication operation for $\mathbf{D}$ and $\mathbf{T}$ would be split among all cores. That is, the matrix-vector multiplication is done locally within each core, followed by a parallel summation over all cores. For example, the matrix-vector multiplication for $\mathbf{D}$ is given by:

$$\mathbf{D}\mathbf{u} = \sum_{i=0}^{N_c-1} \mathbf{D}_i\mathbf{u} \, , \tag{3.37}$$

where N_c is the total number of CPU cores invoked in a simulation; $\mathbf{D}_i\mathbf{u}$ is computed in each core in parallel and then summed up over all cores. By such, the parallel scalability is recovered as the number of solid bodies increases.

3.3.3 Linear algebra operations

The parallel implementation of the linear solver is achieved by interfacing with PETSc package [7]. Once all coefficients resulting from the GMLS discretization are generated from each core, the entire coefficient matrix $\mathbf{A}$ is assembled in parallel through PETSc. And all linear algebra operations associated with the proposed monolithic GMG preconditioner, as well as the Krylov iterations, are performed by calling the corresponding PETSc functions, including the matrix-vector multiplication, matrix-matrix multiplication, and matrix inversion by a direct solver, as well as the node-wise GS iteration and the GMRES iterative solver.

3.3.4 Neumann BC for the pressure

As stated in Eq. (3.2), an inhomogeneous Neumann BC is imposed for pressure. To ensure the uniqueness and physical consistency of the solu-

tion, it additionally enforces a zero-mean constraint for the pressure field, i.e., requiring $\boldsymbol{\Xi}^{\intercal}\mathbf{p} = 0$, where $\boldsymbol{\Xi}$ is a constant vector (e.g. $\boldsymbol{\Xi} = \mathbf{1}$); $\mathbf{p}$ denotes the vector of discretized pressure with entities $p(\mathbf{x}_i)$, $\mathbf{x}_i \in \Omega$. If it uses a Lagrange multiplier to impose this zero-mean constraint, it would break the block structure of $\mathbf{F}$ in Eq. (3.32), and in the meanwhile introduce a dense row and column, i.e., $\boldsymbol{\Xi}^{\intercal}$ and $\boldsymbol{\Xi}$, respectively, into $\mathbf{F}$, which in turn would deteriorate the parallel scalability of the proposed preconditioner. Thus, it instead solves the following problem:

$$\left(\mathbf{I} - \frac{\boldsymbol{\Xi}\boldsymbol{\Xi}^{\intercal}}{\boldsymbol{\Xi}^{\intercal}\boldsymbol{\Xi}}\right)\mathbf{L}\mathbf{p} = \left(\mathbf{I} - \frac{\boldsymbol{\Xi}\boldsymbol{\Xi}^{\intercal}}{\boldsymbol{\Xi}^{\intercal}\boldsymbol{\Xi}}\right)\mathbf{g}\,, \tag{3.38}$$

which is equivalent to using the Lagrangian multiplier. Here, $\mathbf{g}$ denotes the vector with entities $g(\mathbf{x}_i)$, $\mathbf{x}_i \in \Gamma$. In practice, the matrix $\left(\mathbf{I} - \frac{\boldsymbol{\Xi}\boldsymbol{\Xi}^{\intercal}}{\boldsymbol{\Xi}^{\intercal}\boldsymbol{\Xi}}\right)$ does not need to be explicitly assembled. Noting that $\frac{\boldsymbol{\Xi}^{\intercal}\mathbf{p}}{\boldsymbol{\Xi}^{\intercal}\boldsymbol{\Xi}}$ actually calculates the mean of the entities in $\mathbf{p}$, the application of $\left(\mathbf{I} - \frac{\boldsymbol{\Xi}\boldsymbol{\Xi}^{\intercal}}{\boldsymbol{\Xi}^{\intercal}\boldsymbol{\Xi}}\right)$ can be implemented as: first calculating the mean of all entities of the vector $\mathbf{p}$ in parallel and then subtracting the mean from each entity of $\mathbf{p}$. By such, the block structure of $\mathbf{F}$ can be preserved, and the parallel scalability is unaffected.

3.4 Numerical Results

In this section, the effectiveness and scalability of the proposed monolithic GMG preconditioner is assessed through several numerical examples, including pure fluid flows and fluid-solid interaction problems.

3.4.1 Pure fluid flows

It first verifies the GMLS discretization, GMG preconditioner, and parallel implementation by solving problems of pure fluid flows. It starts with a

simple Taylor–Green vortex flow followed by a more complicated Stokes flow in an artificial vascular network.

3.4.1.1 Taylor–Green vortex

The fluid domain is set as $\Omega_f = [-1,1] \times [-1,1]$. Given the source term in Equation (3.2) as:

$$\mathbf{f} = \begin{bmatrix} 2\pi^2 \cos(\pi x)\sin(\pi y) + 2\pi \sin(2\pi x) \\ -2\pi^2 \sin(\pi x)\cos(\pi y) + 2\pi \sin(2\pi y) \end{bmatrix}, \quad \forall\, \mathbf{x} = (x,y) \in \Omega_f\,, \tag{3.39}$$

and the no-slip BC for velocity as

$$\begin{cases} u = \cos(\pi x)\sin(\pi y) \\ v = -\sin(\pi x)\cos(\pi y) \end{cases}, \quad \forall\, \mathbf{x} = (x,y) \in \Gamma_0\,, \tag{3.40}$$

where $\Gamma_0 = \partial\Omega_f$ denotes the outer boundary of the fluid domain, the analytical solution of Eq. (3.2) is the following:

$$\begin{cases} u = \cos(\pi x)\sin(\pi y) \\ v = -\sin(\pi x)\cos(\pi y) \\ p = -\cos(2\pi x) - \cos(2\pi y) \end{cases}, \quad \forall\, \mathbf{x} = (x,y) \in \Omega_f. \tag{3.41}$$

The numerical solutions of this problem obtained with the 2nd-order or 4th-order GMLS discretization, respectively, are compared with the analytical solution. The root mean square (RMS) errors are computed for both velocity and pressure. As shown in Figure 3.3, the numerical results exhibit the consistent 2nd-order or 4th-order convergence, as theoretically expected; the velocity and pressure fields achieve equal-order optimal convergence.

After verifying the accuracy and convergence of the numerical solutions, it next examines the scalability of the proposed GMG preconditioner,

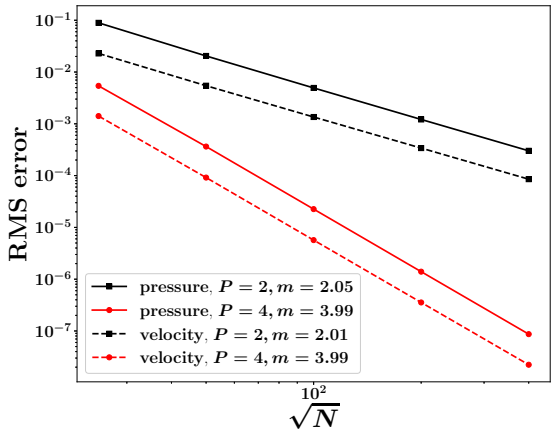
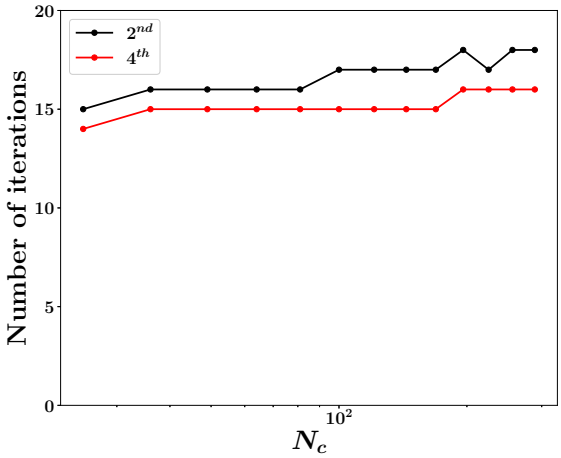


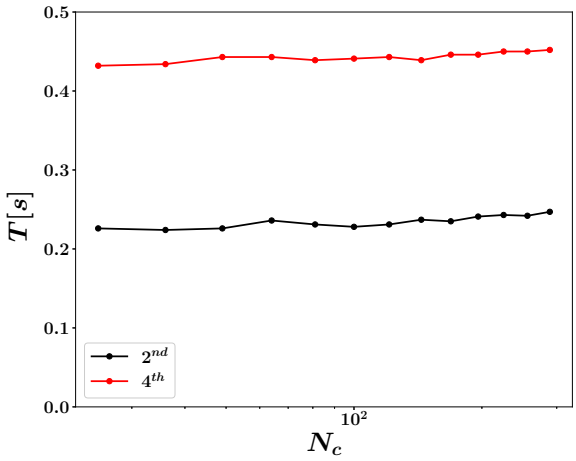
Figure 3.3: Pure fluid flow—Taylor–Green vortex: RMS errors and convergence for the numerical solutions of the velocity (dashed line) and pressure (solid line). Here, N denotes the total number of GMLS nodes; P denotes the order of polynomial basis used in the GMLS discretization; m is the slope of each line.

.

as well as the parallel scalability of the implementation. To this end, the numbers of CPU cores N_c are varied from 25 to 289 in the tests. For such a pure fluid flow problem without singularities, adaptive refinement is not needed, and hence only uniform refinement is conducted. To maintain the same number of GMLS nodes distributed in each CPU core, it starts from 1×1 GMLS nodes in each core, and then execute the same uniform refinement for the GMLS nodes within each core. After seven iterations of uniform refinement, there are 128×128 GMLS nodes in each CPU core. To verify the scalability of the proposed GMG preconditioning method, the number of iterations required for the linear solver to converge is examined. To evaluate the parallel scalability of the implementation, it records the computer time spent for a single step of preconditioning. The tests are particularly for the linear system generated from the last iteration $(I = 7)$ of (uniform) refinement, i.e., with the most DOFs. In Figure 3.4a, it shows



(a) Number of iterations required for the linear solver to converge.



(b) Computer time (in seconds) spent for a single step of preconditioning.

Figure 3.4: Pure fluid flow—Taylor–Green vortex: Scalability of the proposed GMG preconditioner and the weak scalability of the parallel implementation of the preconditioner, tested for different order of GMLS discretization (black for the 2nd order and red for the 4th order). Here, N_c denotes the number of CPU cores used in each test.

the variation of the number of iterations with respect to different numbers of CPU cores used and find that it only varies slightly from 15 to 18 for the 2-nd order GMLS discretization and 14 to 16 for the 4-th order GMLS as the number of CPU cores increases from 25 to 289. The computer time spent on a single step of preconditioning stays almost constant with increasing CPU cores, independent of the order of GMLS discretization, as depicted in Figure 3.4b. By these results, the scalability of the proposed GMG preconditioner is demonstrated and the weak scalability of the parallel implementation of the preconditioner.

3.4.1.2 Artificial vascular network

A more complicated pure fluid flow problem is solved next, which is in an artificial vascular network, mimicking the structure of a zebrafish's eye [5]. The artificial vascular network is built as in Figure 3.5. A constant inflow flux drives the flow in this network at the inner circular boundary in the center, and a constant outflow flux is imposed at the outermost circular boundary. The overall volume of the fluid in the network is conserved. All the other boundaries in the network are imposed no-slip BCs for the velocity. Due to the irregular computational domain with corner singularities, adaptive refinement is required such that the numerical solution can achieve optimal convergence. Starting from a coarse resolution at $\Delta x^0 = 0.02$ and setting $\alpha = 0.8$ (marking percentage) in **MARK** stage, it performs 8 iterations of adaptive refinements. The velocity field computed after the 8th iteration of adaptive refinement is shown in Figure 3.5, where there are in total 930,240 GMLS nodes, and the 2nd-order GMLS discretization is employed.

Due to a lack of the true solution, the total recovered errors (Eq. (3.19)) is calculated, instead of true errors, during iterations of adaptive refinement to evaluate the accuracy and convergence of the numerical solutions, as shown in Figure 3.6. It is noted that the convergence rate does not

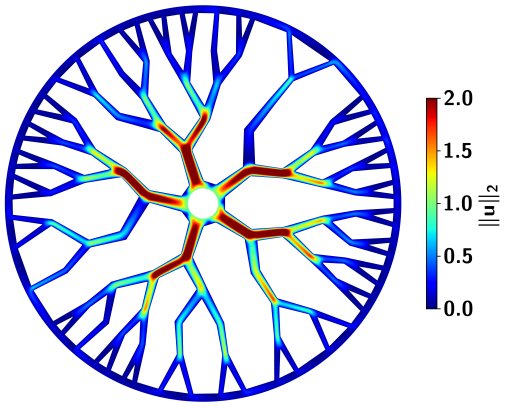


Figure 3.5: Pure fluid flow—Artificial vascular network: Computed velocity field, where the color bar indicates the magnitude of velocity.

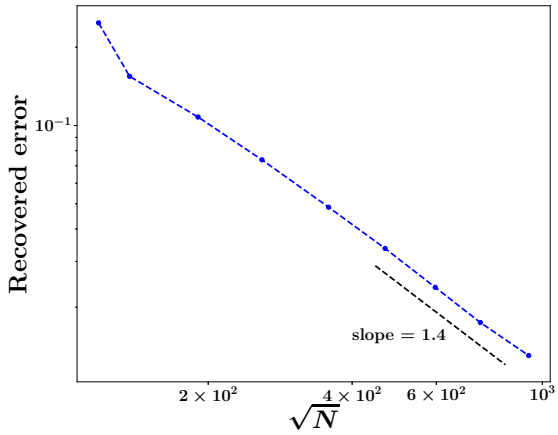
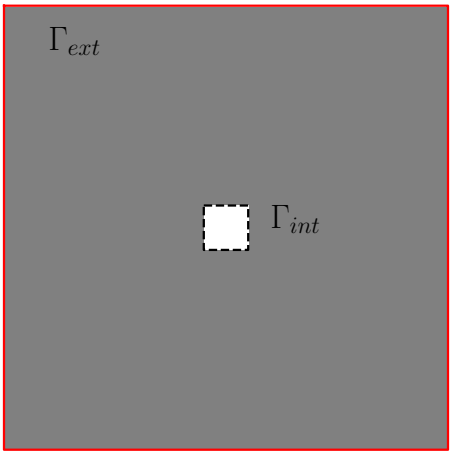


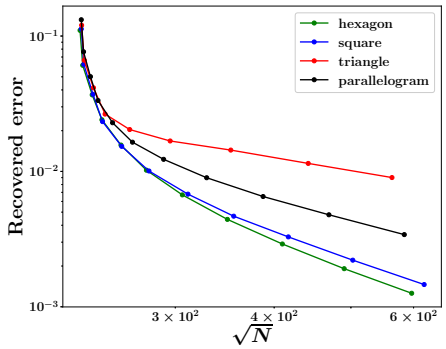
Figure 3.6: Pure fluid flow—Artificial vascular network: Convergence of the total recovered error. Here, the slope is regressed from the last 4 points; N denotes the total number of GMLS nodes.

reach the theoretical 2nd order. This arises from the presence of significant portions of non-convex boundaries in the computational domain. It is well-known that non-convexity can lead to low regularity of the solutions to elliptic PDEs, which in turn affects the performance and liability of recovery-based *a posteriori* error estimator, resulting in deteriorated convergence in adaptive refinement [83, 17]. Therefore, a numerical test is designed to elucidate this issue. In the test, it lets the computational domain be constrained by two boundaries, an exterior boundary Γ_{ext} and an interior boundary Γ_{int}. Γ_{ext} is set the same as that in the Taylor-Green vortex case with the same BCs (Eq. (3.40)). Γ_{int} is stationary but with different shapes, including square, hexagon, triangle, and parallelogram. Taking Γ_{int} of a square as an example, the computational domain Ω_f is depicted in Figure 3.7a. For each shape of Γ_{int}, the problem is solved with 10 iterations of adaptive refinement, and the total recovered errors calculated during adaptive refinement is summarized in Figure 3.7b. As expected, the convergence rate decreases from 2 to 1 when Ω_{int} changes from hexagon to triangle, with increasing non-convexity. This confirms the sub-optimal convergence results observed in Figure 3.6.

After examining the convergence of the numerical solutions, it further assesses the scalability of the proposed GMG preconditioner and parallel implementation. Due to the complexity of the computational domain, it cannot guarantee that the total number of GMLS nodes resulted in each adaptive refinement iteration increases proportionally, thus, the weak scaling test is not appropriate for this problem. Instead, a strong scaling test is performed to demonstrate the parallel scalability of the proposed GMG preconditioning method. In the strong scaling test, the number of cores invoked changes from 40 to 240 cores in the simulation. The statistics of the test is collected in Table 3.1. The average DOFs per core after the 8-th adaptive refinement iteration is listed in the column *DOFs/Core*, which shows the scale of the workload to this problem. As expected, the workload



(a) Illustration of the computational domain: Fluid (gray) is confined in Ω_f by Γ_{ext} and Γ_{int}. Γ_{ext} (red solid line) is a square with side length of 1. Γ_{int} (black dashed line) is at the center of domain and varied from a square to a hexagon, triangle, or parallelogram, all with an equal side length of 0.2.



(b) Convergence of the total recovered error. N denotes the total number of GMLS nodes. The convergence rate is estimated as the slope regressed from the last 4 data points: slop≈2.0 for hexagon and square; slop≈1.6 for parallelogram; slop≈1.0 for triangle.

Figure 3.7: Convergence test with respect to the non-convexity of computational domain.

decreases when the number of cores increases, since a strong scalability test is performed and the total DOFs are fixed. The average number of GMRES iterations required in each adaptive refinement iteration step is listed in the column *Average Iterations*. It stays around 66, independent of different numbers of cores invoked, which indicates that the parallel implementation does not affect the convergence of the proposed GMG preconditioning method. To evaluate the parallel scalability of the implementation, it records the CPU wall time spent for solving the linear systems in each adaptive refinement iteration and sum them up, which is denoted as T_s in Table 3.1; it also tracks the total CPU wall time spent for the entire simulation, denoted as T. By comparing T_s and T, it is noted

Table 3.1: Pure fluid flow—Artificial vascular network: Strong scaling test.

N_c	DOFs/Core	Avg Iter	Time for (3.24) T_s [s]	Overall T [s]	Speedup S
40	69,768	65.5	90.52	128.47	1.00
80	34,884	67.1	52.43	76.50	1.68
160	17,442	66.5	33.71	46.73	2.75
240	11,628	66.3	28.29	36.68	3.50

that the time spent for solving the linear systems indeed dominates the overall computing time, by more than 70%, regardless of how many cores used. The GMLS discretization can be trivially parallelized, and hence the overall parallel scalability is dictated by the proposed preconditioning method. By invoking different numbers of cores, it can compare the overall execution time and thereby evaluate the speed up factor (denoted as S) using the least number of cores (i.e., 40 cores) as the base. Hence, the Amdahl's law [18] is used to assess the performance of the parallel implementation, which is given by:

$$S = \frac{1}{(1-w) + \frac{w}{N_c}},$$

where w denotes the parallel portion of a numerical solver. To estimate the value of w for the solver, it takes the reciprocal of Amdahl's law as:

$$\frac{1}{S} = (1 - w) + \frac{w}{N_c} . \tag{3.42}$$

Using the data of S in Table 3.1, w can be determined by linear regression. In particular, only the data of $N_c = 40$ and $N_c = 80$ are employed to determine w, and then use the data of $N_c = 160$ and $N_c = 240$ for testing. As depicted in Figure 3.8, the last two data points fall very close to the line fitted from the first two data points, and the estimated parallel portion $w = 85.5\%$. It hence demonstrates the consistency and scalability of the parallel implementation of the proposed monolithic GMG preconditioner.

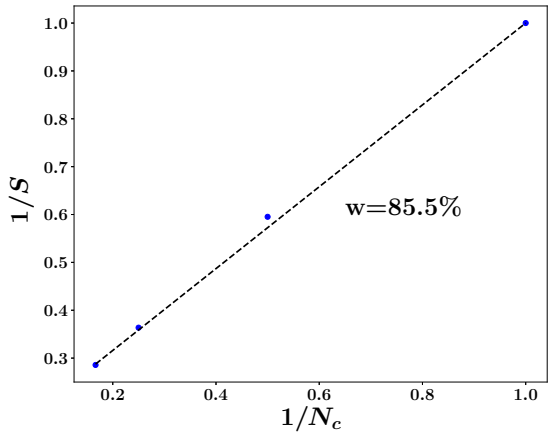


Figure 3.8: Pure fluid flow—Artificial vascular network: Parallel portion w determined from Amdahl's law in Eq. (3.42).

3.4.2 Fluid-solid interactions

After pure fluid flows, it next addresses fluid-solid interaction problems with the inclusion of multiple solid bodies of different shapes.

3.4.2.1 Duplicate cells with cylinders

First, for an accurate assessment of the scalability of the proposed monolithic GMG preconditioner, a numerical example is designed that allows both the solid bodies and the GMLS nodes to be evenly distributed among computer cores. To this end, the entire computational domain is partitioned into N_c square cells, each of which includes four cylinders, as shown in Figure 3.9. All DOFs associated in each square cell are allocated

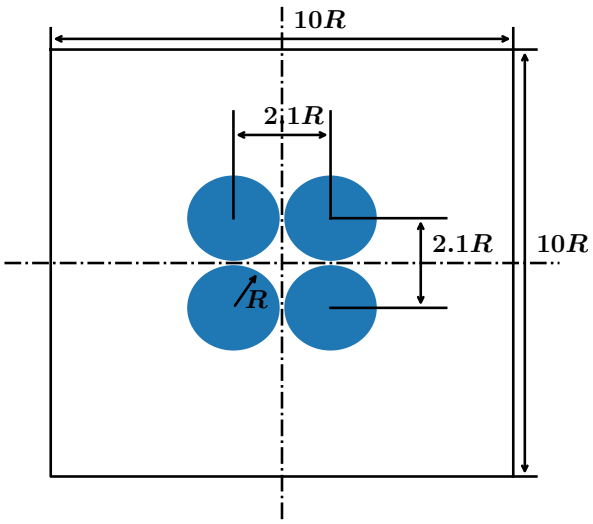


Figure 3.9: Fluid-solid interactions—Duplicate cells with cylinders: The schematic of a square cell with four cylinders.

to a single CPU core. Thus, there are in total $N_s = 4N_c$ solid bodies, where N_c is the number of CPU cores for each test. The test intentionally places the cylinders in close contacts to demonstrate that the spatially adaptive

GMLS method can resolve the singularities governing the lubrication effects. To maintain the same total DOFs after adaptive refinement in each cell, the four cylinders are placed near the center of each cell such that the distances between the solid bodies in different cells are much larger than the distances between the solid bodies within one cell.

The source term in Eq. (3.2) and the BC at the outer boundary of the entire computational domain are set the same as those in the Taylor-Green vortex case, i.e., Eqs. (3.39)- (3.40). The 2nd-order GMLS discretization is employed for solving this problem. Seven iterations of adaptive refinement with $\alpha = 0.8$ are conducted until the total recovered error reaches the preset error tolerance $\varepsilon_{tol} = 10^{-3}$ in Eq. (3.17). The stopping criterion for the GMRES iteration is set as 10^{-6}. The resultant pressure field in a single cell is shown in Figure 3.10. It can be seen that all the singular pressures within the narrow gaps between cylinders are correctly captured.

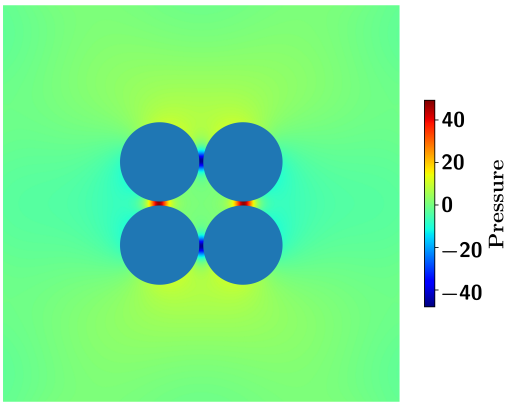
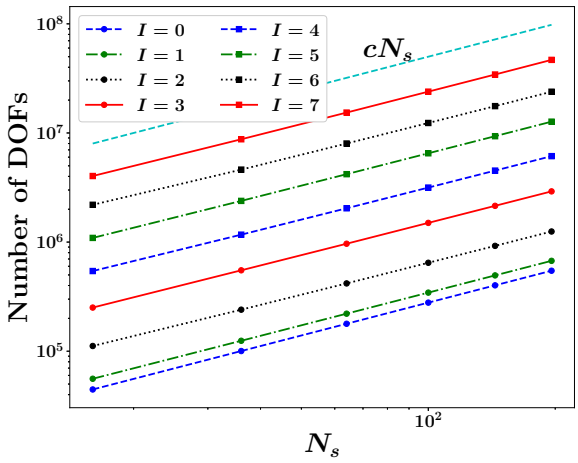


Figure 3.10: Fluid-solid interactions—Duplicate cells with cylinders: The pressure field computed in each cell. The color is correlated to the magnitude of pressure.

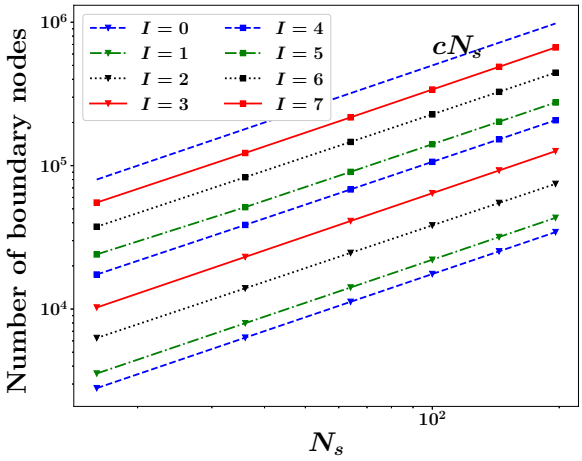
Before assessing the parallel scalability of the preconditioner, it examines how the total DOFs consisting of fluid (GMLS) nodes and boundary (GMLS) nodes grow with adaptive refinement and the inclusion of more

solid bodies. In Figure 3.11a, it can be seen that the total DOFs increases linearly with respect to the number of solids included in the domain, regardless of after any iteration of adaptive refinement; in Figure 3.11b, it is found that the total boundary nodes also increases linearly with respect to the number of solids, after each iteration of adaptive refinement. From these examinations, they confirm that the workload related to the node-wise GS smoother $\mathbf{S_F}$, the additive Schwarz-type smoother $\mathbf{S_N}$, and the MG preconditioner can be evenly distributed among all cores. Only with that ensured, an accurate assessment of scalability can be made.

First, the number of iterations required for the GMRES iterative solver to converge is checked, i.e., to reach the stopping criterion. In Figure 3.12, it shows the number of GMRES iterations required in the **SOLVE** stage of each iteration of adaptive refinement. As can be seen, for a fixed number of solids, the number of GMRES iterations required generally stays constant or decreases slightly. Noting that the total DOFs continuously increase during iterations of adaptive refinement, it hence demonstrates the weak scalability of the preconditioner with respect to the number of DOFs for fixed number of solids. With inclusion of more solids, it is expected that the number of GMRES iterations required would increase. However, the scaling of this increase is critical for the sake of scalability. Figure 3.13a shows that the number of GMRES iterations scales with $O(\sqrt{N_s})$, and the scaling is generally consistent for different iteration step of adaptive refinement. This thesis next checks the computer time spent finishing all seven adaptive refinement iterations. Although it includes the time spent on the GMLS discretization and the time for solving the linear system and executing other stages of adaptive refinement, solving the linear system dominates the computer time. Figure 3.13b depicts how the computer time varies with an increasing number of solids. Overall, it exhibits a scaling of $\mathcal{O}(\sqrt{N_s}\log N_s)$, for which the factor $\sqrt{N_s}$ arises from the scaling of the number of GMRES iterations and the factor $\log N_s$ is mainly contributed by



(a) Number of total DOFs.



(b) Number of the boundary GMLS nodes.

Figure 3.11: Fluid-solid interactions—Duplicate cells with cylinders: The growths of total DOFs and the boundary (GMLS) nodes with respect to the number of solids included in the domain and different iterations of adaptive refinement.

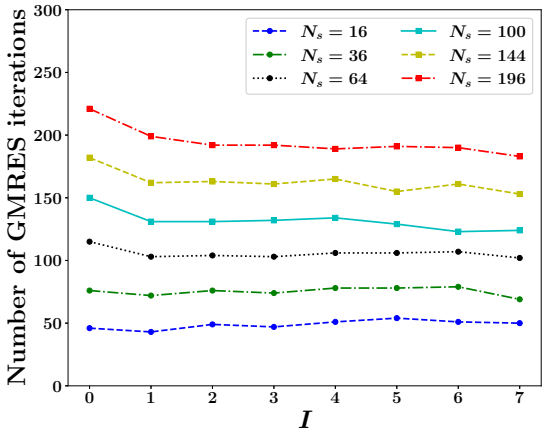
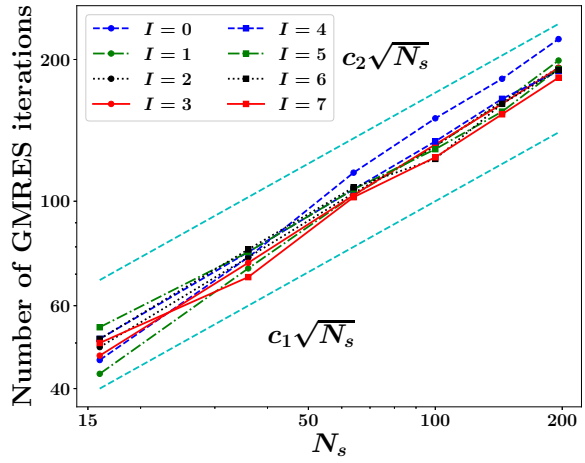


Figure 3.12: Fluid-solid interactions—Duplicate cells with cylinders: The number of GMRES iterations required in the **SOLVE** stage of each adaptive refinement iteration, for fixed numbers of solids.

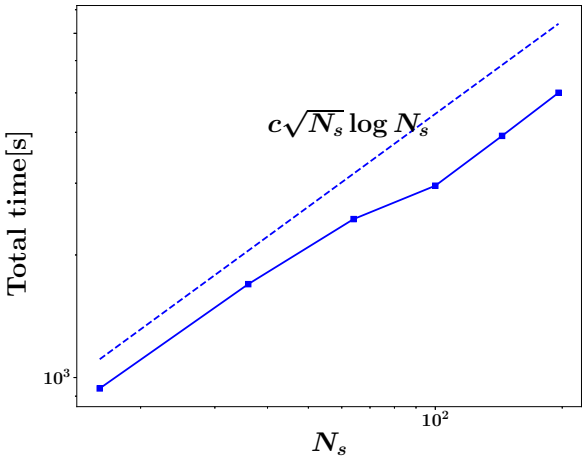
the additional operation introduced in §3.3.2 such as Eq. (3.37). Note that in this numerical example, the number of CPU cores N_c is proportional to the number of solids N_s, in fact $N_s = 4N_c$. Thus, the above scaling behaviors are shown in Figure 3.13 also hold with respect to the number of cores N_c.

3.4.2.2 Particulate suspensions in 2D

A long-time simulation is performed to examine the performance of the proposed monolithic GMG preconditioning method. The test simulates 2D suspension flows of freely moving particles of different shapes. The flow is driven by the source term in Eq. (3.2) and the BC at the outer boundary of fluid domain ($[-1,1] \times [-1,1]$). One hundred solid particles are suspended in the flow, subject to bidirectional hydrodynamic couplings. Initially, all particles are evenly distributed throughout the domain. Due to hydrodynamic couplings, particles are freely moving with the flow. The physical time of the entire simulation is $T = 5$. The 2nd-order GMLS is



(a) Scaling of the number of GMRES iterations required at different iteration step of adaptive refinement.



(b) Scaling of the total computer time spent for all seven adaptive refinement iterations.

Figure 3.13: Fluid-solid interactions—Duplicate cells with cylinders: Scalability results. Here, N_s denotes the number of solids, and $N_s = 4N_c$ with N_c the number of CPU cores.

employed for spatial discretization. The 5th-order Runge-Kutta integrator with adaptive time stepping is used for temporal integration to update the particles' translational and angular positions. An initial time step $\Delta t = 0.2$ is applied. In each time step, adaptive h-refinement with $\alpha = 0.8$ are conducted until the total recovered error reaches the preset error tolerance $\varepsilon_{tol} = 10^{-3}$ in Eq. (3.17). For the linear solver, the stopping criterion for the GMRES iteration is set as 10^{-6}.

100 similar particles In the first simulation, the 100 particles are all circular with the radius $R = 0.04$. The snapshot of the particles' configuration at the terminal time $T = 5$ is shown in Figure 3.14. The zoom-in pressure distributions at several locations are also shown in Figure 3.14, for which is intentionally chosen to show where the particles are in close contact either with each other or with the outer wall. It can seen that even though

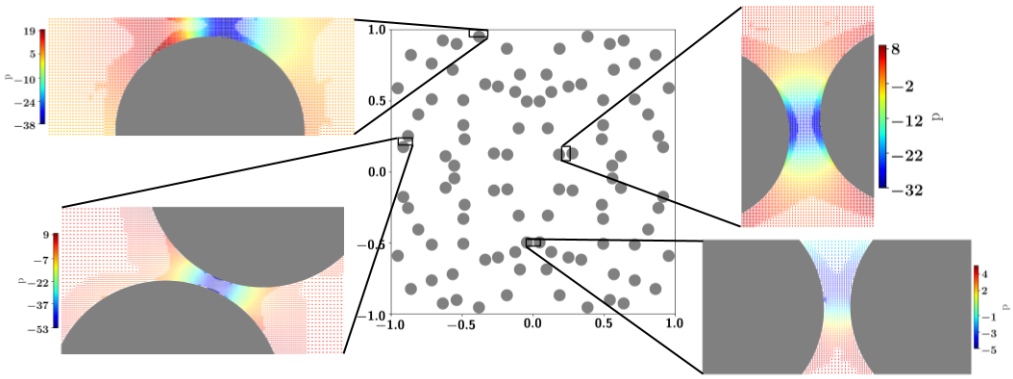
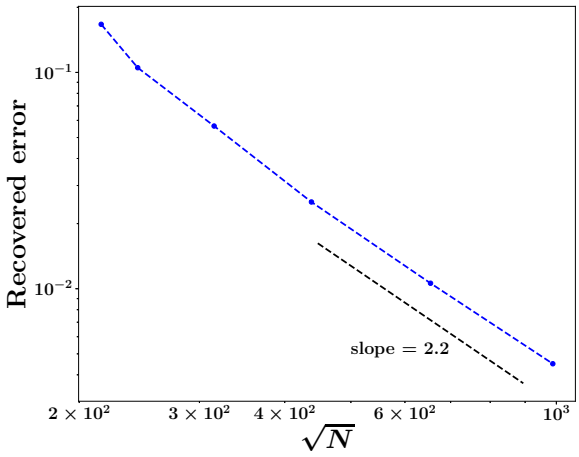


Figure 3.14: Particulate suspensions in 2D: Configuration of 100 freely moving circular particles in a Taylor–Green vortex flow at the terminal time. The zoom-in images are the computed pressure distributions at selected locations where the particles are either in close contact with each other or with the outer wall. Here, the color is correlated to the magnitude of pressure, and the point clouds are the GMLS nodes with adaptive refinement.

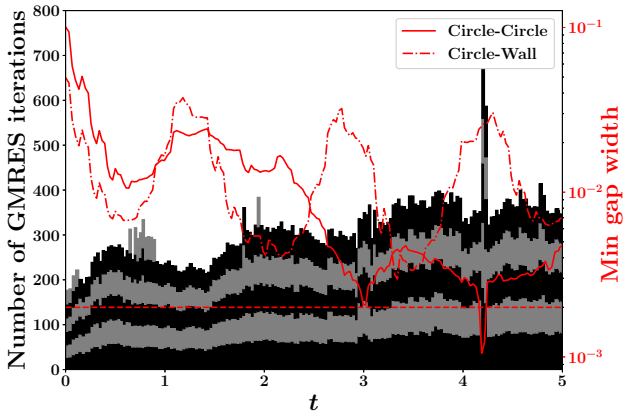
it is challenging in a dynamic simulation to resolve all point singularities

governing lubrication effects, the numerical solver can stably predict the pressure fields without invoking any artificial subgrid-scale lubrication models. The convergence of the total recovered error as defined in Eq. (3.19) for the last time step is shown in Figure 3.15a. It is found that the convergence rate reaches the theoretically expected 2nd order.

To thoroughly examine the performance of the proposed preconditioner, it tracks the number of GMRES iterations required in each adaptive h-refinement iteration and at each time step, as depicted in Figure 3.15b. Several iterations of adaptive h-refinement are needed at each step to reach the preset error tolerance. Hence, the number of GMRES iterations required in each iteration of adaptive h-refinement at different time steps is rendered as the height of the bar with different colors and stacked together for various h-refinement iterations. For example, the lowest black bar represents the number of GMRES iterations required in the first iteration ($I = 1$) of adaptive h-refinement at different time steps; the upper gray bar depicts the number of GMRES iterations needed for the second iteration ($I = 2$) of adaptive h-refinement at different time steps; and so forth. By comparing the height of each bar at a fixed time step, it can be seen that the number of GMRES iterations generally stays constant across different iterations of adaptive h-refinement, indicating the scalability of the proposed monolithic GMG preconditioner in terms of increasing total DOFs. By comparing the heights of a bar at different time steps, it is noted that the number of GMRES iterations is highly correlated with the minimum gap width between solid boundaries (particle-particle or particle-wall). To elaborate on that, two lines (red) are added in Figure 3.15b showing the minimum gap widths between solid boundaries at different time instances. Note that the minimum gap width can be as small as $0.03R$ with $R = 0.04$ the particles' radius. Generally, when the minimum gap widths between solid boundaries are small, more GMRES iterations are required for the linear solver to converge. That is because finer GMLS nodes are needed



(a) Convergence of the total recovered error (Eq. (3.19)) at the last time step. The slope measured by the last four data points is 2.2. N denotes the total number of GMLS nodes.



(b) Number of GMRES iterations required in each adaptive h-refinement iteration and at each time step during the entire simulation.

Figure 3.15: Particulate suspensions in 2D—100 similar particles: Convergence of the recovered error and the required number of GMRES iterations.

to resolve narrower gaps between solid boundaries, resulting in more ill-conditioned linear systems to solve. The numbers of required GMRES iterations are comparable when the minimum gap widths are larger than 0.002 (indicated by the horizontal red dash line in Figure 3.15b), which is equivalent to 0.05R.

100 dissimilar particles In the second simulation, the suspended particles are a mixture of eighty circles and twenty squares. The radius of circular particles is still $R = 0.04$; the side length of square particles is $L = 2R = 0.08$. To mimic the particles in real applications of particulate suspensions, the squares are rounded at the corners with a rounding radius $R' = 0.1L = 0.008$. The snapshot of the particles' configuration at the terminal time $T = 5$ is shown in Figure 3.16, where the zoom-in pressure distributions, particularly around square particles, are also shown at selected locations. Regardless of particle shapes, the numerical solver can stably solve the problem even when the particles are in close contact with each other or with the outer wall boundary. No any artificial subgrid-scale lubrication model is employed during the entire simulation. The convergence of the total recovered error as defined in Equation (3.19) for the last time step is shown in Figure 3.17a. With the inclusion of different shapes of solids, it still sees the convergence rate reaching the theoretically expected 2nd order.

By tracking the number of GMRES iterations required in each adaptive h-refinement iteration and at each time step, the performance of the proposed preconditioning method is assessed. Similarly, it compares the number of GMRES iterations, as depicted in Figure 3.17b. For this case with mixed shapes of solids, more iterations of adaptive h-refinement are needed to reach the preset error tolerance at each time step. By comparing the height of each bar at a fixed time step, it is also seen that the number of GMRES iterations generally stays constant across different iterations of

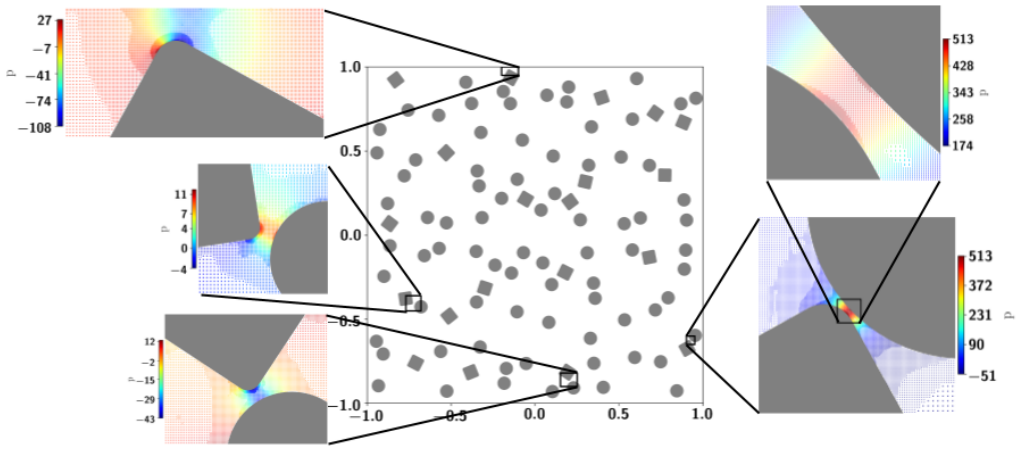
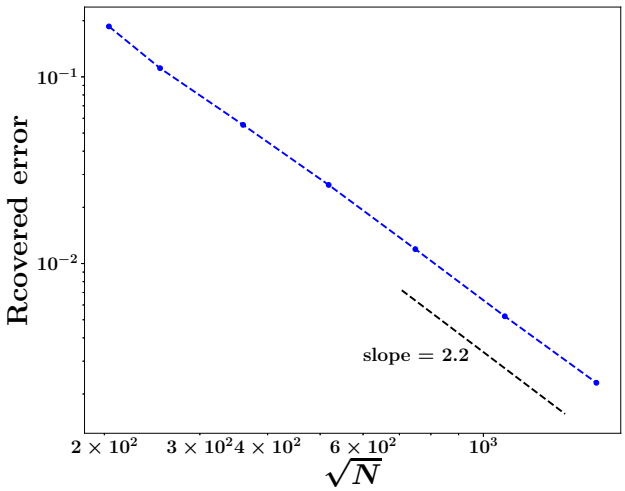
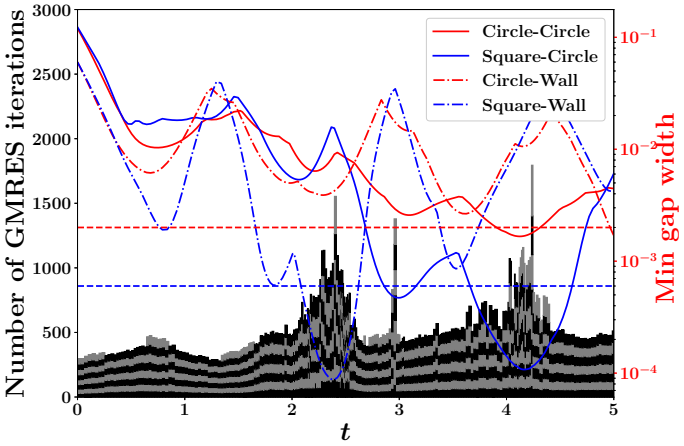


Figure 3.16: Fluid-solid interactions—Particulate suspensions in 2D: Configuration of 100 freely moving dissimilar particles in a Taylor–Green vortex flow at the terminal time. The zoom-in images are the computed pressure distributions at selected locations where the particles are either in close contact with each other or with the outer wall. Here, the color is correlated to the magnitude of pressure, and the point clouds are the GMLS nodes with adaptive refinement.

adaptive h-refinement, implying the scalability of the proposed monolithic GMG preconditioner in terms of increasing total DOFs. As in the case of 100 similar particles, the number of GMRES iterations is correlated with the minimum gap width between solid boundaries (particle-particle or particle-wall). In addition, the inclusion of square-shaped particles can worsen the problem's conditioning in the continuous limit. To reflect the combined effects of the minimum gap width between solid boundaries and the shape of particles, in Figure 3.17b lines are added to track the minimum gap widths associated with particles of different shapes (circular or square). It is found that the square particles make the dominant contributions to the required GMRES iterations. When the minimum gap widths associated with square particles are larger than 0.0006 (indicated by the horizontal blue dash line in Figure 3.17b), which is equivalent to $0.075R'$ with $R' = 0.008$ the square particles' corner rounding radius, the



(a) Convergence of the total recovered error (Equation (3.19)) at the last time step. The slope measured by the last four data points is 2.2. N denotes the total number of GMLS nodes.



(b) Number of GMRES iterations required in each adaptive h-refinement iteration and at each time step during the entire simulation.

Figure 3.17: Particulate suspensions in 2D—100 dissimilar particles: Convergence of the recovered error and the required number of GMRES iterations.

number of GMRES iterations required at a given h-refinement iteration is generally comparable across different time instances; when they are smaller than 0.0006, significantly more GMRES iterations are required for the linear solver to converge, because the corresponding linear system can become severely ill-conditioned.

3.4.2.3 Particulate suspensions in 3D

A long-time simulation is performed next to demonstrate that the proposed monolithic GMG preconditioning method can be extended to 3D, with the scalability and robustness maintained. In particular, a 3D suspension flow of 27 freely moving spherical particles is considered. The source term that drives a Taylor-Green vortex flow in 3D is given by:

$$\mathbf{f} = \begin{bmatrix} 3\pi^2 \cos(\pi x)\sin(\pi y)\sin(\pi z) + 2\pi \sin(2\pi x) \\ -6\pi^2 \sin(\pi x)\cos(\pi y)\sin(\pi z) + 2\pi \sin(2\pi y) \\ 3\pi^2 \sin(\pi x)\sin(\pi y)\cos(\pi z) + 2\pi \sin(2\pi z) \end{bmatrix}, \quad \forall (x, y, z) \in \Omega_f \,, \tag{3.43}$$

and the no-slip BC for the velocity imposed at the outer boundary of the 3D fluid domain ($[-1, 1] \times [-1, 1] \times [-1, 1]$) is specified as:

$$\begin{cases} u = \cos(\pi x)\sin(\pi y)\sin(\pi z) \\ v = -2\sin(\pi x)\cos(\pi y)\sin(\pi z) \,, \quad \forall (x, y, z) \in \Gamma_0 \,. \\ w = \sin(\pi x)\sin(\pi y)\cos(\pi y) \end{cases} \tag{3.44}$$

The suspended 27 spherical particles with the radius $R = 0.1$ are subject to bidirectional hydrodynamic couplings and freely moving in the flow. Initially, all particles are evenly distributed throughout the domain. The 2nd-order GMLS is employed for spatial discretization with initial discretization resolution at $\Delta x^0 = 0.04$. The 5th-order Runge-Kutta integrator with adaptive time stepping is used for temporal integration to update the particles' positions and orientations, with the initial time step set as

$\Delta t = 0.1$. The physical time of the entire simulation is $T = 1$, and the configuration of particles at the last time step is shown in Figure 3.18. At each time step, the adaptive h-refinement is conducted according to the recovered error estimator (Eq. (3.18)), with $\alpha = 0.9$, and for three times until the total recovered error close to the preset tolerance 10^{-2}. The convergence of the total recovered error as defined in Eq. (3.19) for the last time step is presented in Figure 3.19a. It is found that the convergence rate reaches the theoretically expected 2nd order.

For the linear solver, the stopping criterion for the GMRES iteration is set as 10^{-6}. The performance of the proposed preconditioner is again assessed by tracking the number of GMRES iterations required in each adaptive h-refinement iteration and at each time step during the entire simulation. Without any modification on the preconditioner, the proposed preconditioning method preserves the attributes seen in the 2D counterparts. As shown in Figure 3.19b, by inspecting the heights of the bars with gray/black colors at each time step, it is seen that the number of GMRES iterations almost stay constant across different iterations of adaptive h-refinement. This indicates that, with the increasing total DOFs, the scalability attribute of the proposed monolithic GMG preconditioner is preserved when applying it to solving a 3D problem. By examining the variation of the height of each bar with respect to time, it is found that the numbers of GMRES iterations are generally comparable across different time steps. The slight increase can be correlated to the decreased minimum gap width between solid boundaries (sphere-sphere or sphere-wall). The decreased gaps between solid boundaries require finer GMLS nodes to resolve and hence lead to more ill-conditioned linear system to solve, which in turn calls for more GMRES iterations to reach convergence.

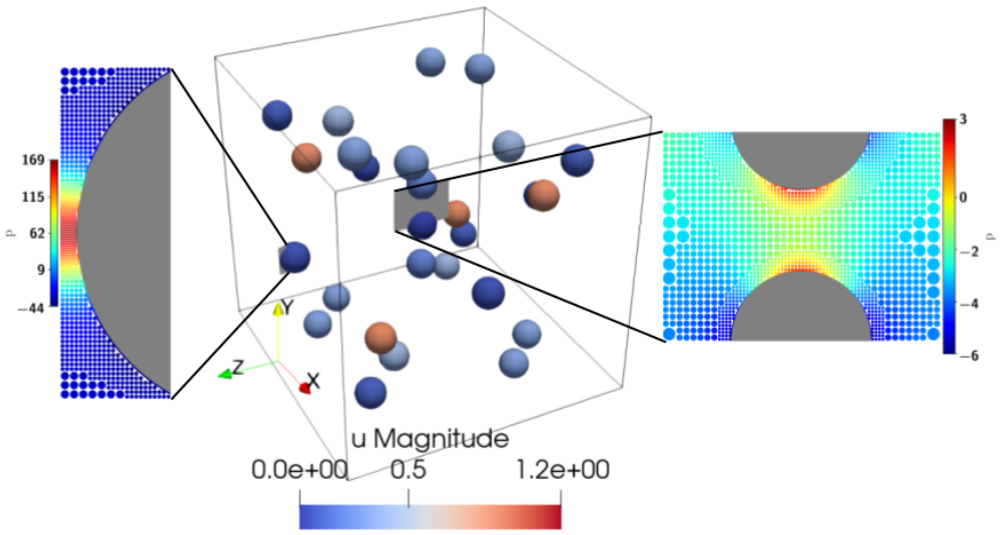
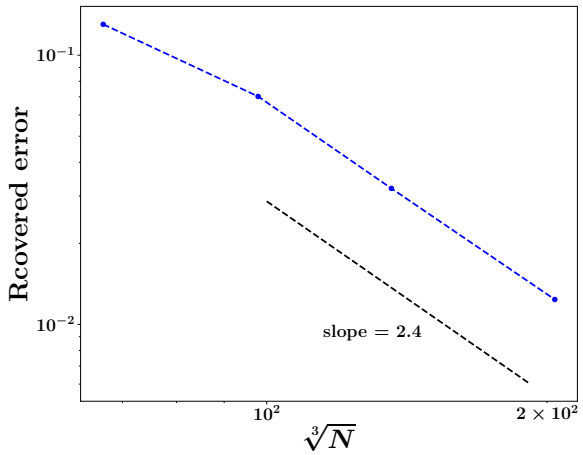


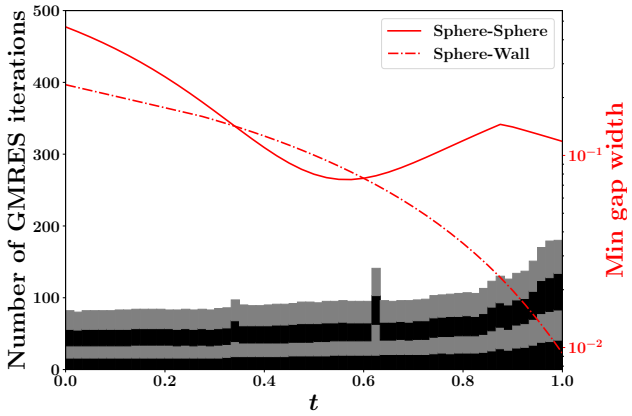
Figure 3.18: Particulate suspensions in 3D: Configuration of 27 freely moving spherical particles in a Taylor–Green vortex flow at the terminal time. The color on each particle is correlated to the magnitude of its velocity. The zoom-in images show the flow-field pressure distributions on selected planes where the particles are either in close contact with each other or with the outer wall. Here, the color is correlated to the magnitude of pressure, and the point clouds are the GMLS nodes with adaptive refinement.

3.5 Conclusions

A monolithic GMG preconditioner for solving fluid-solid interaction problems in Stokes limit is presented in this chapter. The linear systems of equations are generated from the spatially adaptive GMLS discretization, which was developed in the previous work [31]. The GMLS discretization is meshless and can handle large displacements and rotations of solid bodies without the expensive cost of generating and managing meshes. It guarantees the same order of accuracy for both the velocity and pressure fields, and high-order accuracy is achievable by its polynomial reconstruction property and choosing appropriate polynomial bases. A staggered



(a) Convergence of the total recovered error (Equation (3.19)) at the last time step. The slope measured by the last three data points is 2.4. N denotes the total number of GMLS nodes.



(b) Number of GMRES iterations required in each adaptive h-refinement iteration and at each time step during the entire simulation.

Figure 3.19: Particulate suspensions in 3D: Convergence of the recovered error and the required number of GMRES iterations.

discretization approximates the div-grad operator to ensure stable solutions of Stokes equations. With adaptive h-refinement directed by an *a posteriori* recovery-based error estimator, it can resolve the singularities governing the lubrication effects between solid bodies without invoking any artificial subgrid-scale lubrication models. With the proposed monolithic GMG preconditioner in this work, it is also able to scale up the spatially adaptive GMLS discretization for solving larger-scale fluid-solid interaction problems and achieve scalability with increasing numbers of solid bodies and total DOFs, while preserving accuracy and stability.

The preconditioner is composed of two main ingredients: the interpolation/restriction operators and the smoothers. While the interpolation/restriction operators transfer the velocity and pressure values between different resolution levels, the smoothers damp the high-frequency error components on each resolution level. These two ingredients and their interplay determine the performance and scalability of the preconditioner and, thereby, the linear solver. The hierarchical structure of the adaptively refined GMLS nodes has provided us with an appropriate geometric setting for constructing the interpolation/restriction operators. In particular, to construct the interpolation operator, local GMLS approximations are employed from the coarse-level nodes to the fine-level nodes. To build the restriction operator, it lets the value of a variable on a parent GMLS node equal the average value of its child nodes generated in a new iteration of adaptive refinement. During the interpolation and restriction processes, the divergence-free property of velocity is preserved, which plays an essential role in designing efficient MG preconditioning methods for solving Stokes equations. For the smoothers, a smoother is built for the fluid domain and a smoother for the solid bodies and then integrate them following a multiplicative overlapping Schwarz method. It uses a node-wise block Gauss-Seidel smoother for the fluid domain to address the coupling between the two field variables: velocity and pressure. The

smoother for the solid bodies handles each solid body separately: for each solid body, a submatrix is first assembled corresponding to the solid body and its neighboring interior GMLS nodes; a Schur complement approach is then employed to approximately invert the submatrix based on the block splitting between the fluid and solid DOFs.

Such a developed preconditioner is integrated with the Krylov iterative solver, GMRES, for solving the linear systems of equations generated from the GMLS discretization. It has leveraged PETSc [7] for the parallel implementation of the proposed monolithic GMG preconditioner. In addition to all associated linear algebra operations handled by PETSc, this work has carefully taken care of domain decomposition, neighbor search, data storage, and imposing inhomogeneous Neumann BC for pressure in parallel implementation in order to warrant the parallel scalability of the numerical solver. Through a series of numerical tests, including simulating pure fluid flows and fluid-solid interactions with the inclusion of different numbers and shapes of solid bodies, it has demonstrated the performance and parallel scalability of the proposed preconditioner. Specifically, as the number of solid bodies and total DOFs increases, the convergence of the Krylov iterative solver can be ensured. For a fixed number of solid bodies, the preconditioner scales nearly linearly with respect to the total DOFs. When the number of solid bodies increases, the preconditioner exhibits sublinear optimality with respect to the number of solid bodies. In addition, the parallel implementation has achieved weak scalability for both pure fluid flows and fluid-solid interactions. For the flow in an artificial vascular network, the numerical result shows consistency with the prediction of Amdahl's law, indicating strong scalability and efficiency of the parallel implementation.

4 CLASSICAL OPTIMIZER FOR DISCRETE VARIABLE TOPOLOGY OPTIMIZATION

4.1 Algorithm for Solving the Bilinear Programming (2.36)

The master problem as described in Eq. (2.36) contains the nontrivial bilinear terms formed by the binary variables $\boldsymbol{\rho}$ and $\boldsymbol{\alpha}$. As a result, it cannot be directly solved by employing an off-the-shelf integer programming solver. Thus, it herein proposes a feasible and efficient way to solve it. By noting that the size of $\boldsymbol{\alpha}$ is much smaller than that of $\boldsymbol{\rho}$, branching on $\boldsymbol{\alpha}$ is not difficult. The branching essentially selects one or multiple indices j, $j \in \{0, 1, \cdots, k-1\}$, such that $\alpha_j = 1$ but the rest elements in $\boldsymbol{\alpha}$ are zero. For example, 1 and 2 are selected, leading to $\boldsymbol{\alpha} = (0, 1, 1, 0, \dots)$. By doing so, the constraint $\sum_{j=0}^{k-1} \alpha_j \geqslant 1$ is satisfied. Further, the selection must also satisfy the constraint: $\boldsymbol{\alpha} \neq \boldsymbol{\alpha}^i$, $i = 1, 2, \cdots, k-1$. Let $\mathcal{P}_s \subseteq \{0, 1, \cdots, k-1\}$ denote one selection for the index j that satisfy both constraints, and all such selections are collected into the set $\mathcal{C}(k)$.

Among all selections in $\mathcal{C}(k)$, there is only one selection (denoted as $\mathcal{P}_1$), for which only one element of $\boldsymbol{\alpha}$ is nonzero, and this only nonzero element is α_{k-1}, i.e., $\mathcal{P}_1 = \{k-1\}$ and $\boldsymbol{\alpha} = (0, 0, \dots, 1)$. (Note that $\boldsymbol{\alpha}$ with a single nonzero element α_j for $\forall j < k-1$ has been considered in previous iterations and hence would not be considered again in the current k-th iteration step.) For this particular selection, the master problem in Eq.

(2.36) is reduced to a single-cut problem and can be simplified as:

$$\begin{aligned} \min_{\boldsymbol{\rho},\eta} \quad & \eta \\ \text{s.t.} \quad & \widetilde{f}^{k-1}(\boldsymbol{\rho}) \leqslant \eta \\ & t^{k-1}(\boldsymbol{\rho}) \leqslant d^{k-1} \\ & H_{i_H}(\boldsymbol{\rho}) \leqslant 0, \quad i_H = 1, \ldots, n_H \\ & \boldsymbol{\rho} \in \{0,1\}^{n_e} . \end{aligned} \tag{4.1}$$

Its solution is denoted as $\eta_{k,1}$. (The solution for each of the other single-cut problems that have been solved in the previous iteration steps is denoted as $\eta_{l,1}$ with $\forall l < k$.)

For every other selection in $\mathcal{C}(k)$, $|\mathcal{P}_s| \geqslant 2$, i.e., $\boldsymbol{\alpha}$ has at least two nonzero elements. For that, the master problem in Eq. (2.36) must involve multiple cuts (at least two) and can be rewritten as:

$$\begin{aligned} \min_{\boldsymbol{\rho},\eta} \quad & \eta \\ \text{s.t.} \quad & \widetilde{f}^{j}(\boldsymbol{\rho}) \leqslant \eta, \quad \forall j \in \mathcal{P}_s \\ & t^{j}(\boldsymbol{\rho}) \leqslant d^{j}, \quad \forall j \in \mathcal{P}_s \\ & H_{i_H}(\boldsymbol{\rho}) \leqslant 0, \quad i_H = 1, \ldots, n_H \\ & \boldsymbol{\rho} \in \{0,1\}^{n_e} . \end{aligned} \tag{4.2}$$

Its solution is denoted as $\eta_{k,s}$. As the multi-cut problem includes more than one cuts (or constraints), its solution $\eta_{k,s}$ cannot be lower than the solution of any single-cut problem containing just one of those cuts, i.e., $\eta_{k,s} \geqslant \max_{l \in \mathcal{P}_s}(\eta_{l,1})$. This feature can be utilized for early stopping the branching on $\boldsymbol{\alpha}$, thereby enhancing computational efficiency.

To proceed, it is denoted with $\bar{\eta}_{k,s} = \max_{l \in \mathcal{P}_s}(\eta_{l,1})$. All selections represented by $\mathcal{P}_s$ with $|\mathcal{P}_s| \geqslant 2$ in $\mathcal{C}(k)$ are ranked in non-descending order based on the value of $\bar{\eta}_{k,s}$. Starting from the first in the rank list, the corresponding multi-cut problem is solved, as formulated in Eq. (4.2),

until it is found that the minimum of the solutions of all solved multi-cut problems is smaller than $\bar{\eta}_{k,s}$ of the next multi-cut problem in the rank list, i.e., $\min_s(\eta_{k,s}) < \bar{\eta}_{k,s+1}$. Once this is achieved, branching on $\boldsymbol{\alpha}$ can be terminated, because the remaining selections in the list would not yield better solutions. Thus, the final solution of η for the master problem (2.36) is given by:

$$\eta^k = \min \left[\min_s(\eta_{k,s}),\ \eta_{k,1} \right] , \tag{4.3}$$

i.e., the minimum among the solutions of all solved multi-cut problems (as in Eq. (4.2)) and the single-cut problem (as in Eq. (4.1)). The final solution of $\boldsymbol{\alpha}$ and $\boldsymbol{\rho}$ will then be derived from the multi-cut or single-cut problem that yields the minimum η.

If the final solution for the master problem (2.36) is determined as the solution of a multi-cut problem, i.e., $\eta^k = \min_s(\eta_{k,s})$, it indicates that using a single cut solely based on the solution of the last iteration step, as done by the methods like SIMP [6] and TOBS [57], does not always yield the best solution; but including multiple cuts based on the solutions from at least two previous iteration steps can improve the solution. Therefore, allowing for adaptively including multiple cuts would enable us to find the best solution at each iteration step, thereby accelerating convergence and improving solution efficiency. This highlights the distinction of the framework proposed in this paper from other methods, including SIMP [6], TOBS [57], FP [32, 33], and SAIP [47, 49].

In summary, solving the master problem as formulated in Eq. (2.36) involves solving two types of linear integer programming problems, i.e., the single-cut problem in Eq. (4.1) and the multi-cut problems in Eq. (4.2). Each type of problem can be directly solved using a linear integer programming solver; in this work, it uses the Gurobi optimizer [28]. The algorithm proposed for solving the master problem (2.36) is summarized in Algorithm 5.

Algorithm 5 MasterProbSolver($\boldsymbol{\alpha}^i$)

Input: $\boldsymbol{\alpha}^i$ from previous iteration steps, $i = 1, \ldots, k-1$.
Output: The optimal solution $\boldsymbol{\alpha}^k$, the minimizer $\boldsymbol{\rho}^k$ and the minimum value η^k
Utility: Solve the master problem (2.36)

Form the set $\mathcal{C}(k)$ for branching on $\boldsymbol{\alpha}$
Solve the single-cut problem in Eq. (4.1) and yield the solution $\boldsymbol{\rho}_{k,1}$ and $\eta_{k,1}$
Sort $\mathcal{P}_s \in \mathcal{C}(k)$ (with $|\mathcal{P}_s| \geqslant 2$) in non-descending order based on the value of $\bar{\eta}_{k,s}$
for $s = 2, \ldots, |\mathcal{C}(k)|$ **do**
    Solve the multi-cut problem in Eq. (4.2) and yield the solution $\boldsymbol{\rho}_{k,s}$ and $\eta_{k,s}$
    if $\min_s(\eta_{k,s}) < \bar{\eta}_{k,s+1}$ **then**
        break
    end if
end for
$s^* = \arg\min_s(\eta_{k,s})$
$\boldsymbol{\rho}^k = \boldsymbol{\rho}_{k,s^*}$, $\eta^k = \eta_{k,s^*}$
Let $\alpha_l^k = 1,\ \forall l \in \mathcal{P}_{s^*}$; $\alpha_l^k = 0,\ \forall l \notin \mathcal{P}_{s^*}$
Return: $\boldsymbol{\alpha}^k$, $\boldsymbol{\rho}^k$, η^k

4.2 Modified Dantzig-Wolfe (DW) Decomposition

The total mass constraint $H_{\mathcal{M}_0}$ in (2.16), the sensitivity cuts $\widetilde{f}^j(\boldsymbol{\rho})$ and the trust regions $t^j(\boldsymbol{\rho})$ are categorized as global constraints as they evolve all design variables; the material usage constraint $H_{\mathcal{M}_e}$ in (2.16) is considered as local constraint since only $n_{\mathcal{M}}$ variables are restricted by it. Thus, for the multi-cuts formulation (4.2), the programming shows the block-angular structure and can be applied with the Dantzig-Wolfe (DW) decomposition [21] to formulate a series reduced size sub-problems. The total mass constraints defined in (2.16) are considered as the general case for the

following derivations and the volume constraints H_V defined in (2.14) can be treated in the same way.

To apply the DW decomposition, the entire design variables have to be divided into $n_{\mathcal{D}}$ blocks. So, all of the design variables are indexed according to the following value:

$$\widetilde{\rho}_e = \sum_{m=1}^{n_M} m\rho_{e,m} \ , \tag{4.4}$$

which gives a new value for each design variable $\rho_{e,m}$ defined on the element e. $\widetilde{\rho}_e$ can then be sorted in a non-descending order and it can gives each element e a new index e'. As $\rho_{e,m}$ is binary, the possible values for $\widetilde{\rho}_e$ is finite. Therefore, the sorting can be performed in parallel and it only needs to communicate the possible values for $\widetilde{\rho}_e$ between different processes. The cost of communication can be greatly saved compared to the general sorting algorithms which requires communicating sub-vectors of $\boldsymbol{\rho}$. Next, the entire design variables are evenly divided into $n_{\mathcal{D}}$ blocks according to this new index e' and each block $\mathcal{D}_i$ is defined as:

$$\mathcal{D}_i = \left\{ e \left| \left\lfloor i \cdot \frac{n_e}{n_{\mathcal{D}}} \right\rfloor \leqslant e' \leqslant \left\lfloor (i+1) \cdot \frac{n_e}{n_{\mathcal{D}}} \right\rfloor \right. \right\} \ , \tag{4.5}$$

which collects the indices of the elements e with the same or close value of $\widetilde{\rho}_e$ in the i-th block. Each block $\mathcal{D}_i$ then roughly manages $\frac{n_e n_M}{n_{\mathcal{D}}}$ design variables. The design variables then form a vector $\boldsymbol{\rho}_i$ such that $\boldsymbol{\rho}_i = \{\rho_{e,m} | \forall e \in \mathcal{D}_i, m = 1, \dots n_M\}$. Finally, two new problems can be generated from DW decomposition: a sub-problem concerning the local constraints and a sub-problem concerning the global constraints, by decomposing the entire design variables into multiple blocks according to the index set $\mathcal{D}_i$. The sub-problem concerning the local constraints can be denoted as the *local sub-problem* and the sub-problem concerning the global constraints can be denoted as the *global sub-problem*. The size of the local sub-problem is

only related to the size of each block $\mathcal{D}_i$. The size of the global sub-problem is determined by the number of the blocks $n_{\mathcal{D}}$ and the number of iterations L. As the number of iterations, shown in the numerical experiments 4.3a, is irrelevant to the size of the design variables, the size of the global sub-problem can be tractable for the large-scale TO problems.

Different from the standard DW decomposition that utilizing all of the extreme points in the local sub-problem to formulate the global sub-problem, the proposed method modifies the standard DW decomposition that it generates candidate solutions from the local sub-problem and iteratively enlarges the size of the global sub-problem until convergence. In each iteration, the sub-problem would generate a candidate solution to the original problem. This guess at iteration step l of the proposed method is denoted as $\hat{\rho}^l_{e,m}$, which uses a third subscript to denote the iteration step. Due to the slight difference in the formulation, the single cut formulation (4.1) can be considered as a special case when $|\mathcal{P}_s| = 1$ in the multi-cuts formulation (4.2). Therefore, only the derivation for the multi-cuts formulation (4.2) is considered in this section. The initialization and iteration of the proposed method is discussed at the end of this section.

4.2.1 Multi-cuts formulation

The multi-cuts problem (4.2) takes a single continuous variable η that can not be decomposed into multiple blocks. Therefore, a series variables $\eta_i, i = 1, \ldots, n_{\mathcal{D}}$ are introduced to replace η and the multi-cuts formulation

(4.2) can be rewritten as:

$$\begin{aligned}
\min_{\boldsymbol{\rho},\eta} \quad & \sum_{i=1}^{n_{\mathcal{D}}} \eta_i \\
\text{s.t.} \quad & \widetilde{f}^j(\boldsymbol{\rho}) \leqslant \sum_{i=1}^{n_{\mathcal{D}}} \eta_i, \qquad \forall j \in \mathcal{P}_s \\
& t^j(\boldsymbol{\rho}) \leqslant d^j, \qquad \forall j \in \mathcal{P}_s \\
& H_{M_0}(\boldsymbol{\rho}) \leqslant \bar{M}_{max} \\
& \sum_{m=1}^{n_M} \rho_{e,m} \leqslant 1, \qquad e = 1, 2, \ldots, n_e \\
& \boldsymbol{\rho} \in \{0,1\}^{n_e} .
\end{aligned} \tag{4.6}$$

With the new multi-cuts formulation (4.6), the i-th local sub-problem by applying the proposed modified DW decomposition can be written as:

$$\begin{aligned}
\min_{\boldsymbol{\rho}_i,\eta_i} \quad & \eta_i + \sum_{j\in\mathcal{P}_s} \pi_j^f(\widetilde{f}^j(\boldsymbol{\rho}_i) - \eta_i) + \sum_{j\in\mathcal{P}_s} \pi_j^t t^j(\boldsymbol{\rho}_i) + \pi^M H_{M_0}(\boldsymbol{\rho}_i) \\
\text{s.t.} \quad & \sum_{m=1}^{n_M} \rho_{e,m} \leqslant 1, \quad e \in \mathcal{D}_i \\
& \boldsymbol{\rho}_i \in \{0,1\}^{|\mathcal{D}_i|\times n_M} ,
\end{aligned} \qquad , \tag{4.7}$$

In the local sub-problem (4.7), as η_i is unbounded, $\sum_{j\in\mathcal{P}_s} \pi_j^f$ can then only be 1 to make the problem bounded. η_i, as a redundant variable, can be omitted from the local sub-problem (4.7) and it can be rewritten as:

$$\begin{aligned}
\min_{\boldsymbol{\rho}_i} \quad & \sum_{j\in\mathcal{P}_s} \pi_j^f \widetilde{f}^j(\boldsymbol{\rho}_i) + \sum_{j\in\mathcal{P}_s} \pi_j^t t^j(\boldsymbol{\rho}_i) + \pi^M H_{M_0}(\boldsymbol{\rho}_i) \\
\text{s.t.} \quad & \sum_{m=1}^{n_M} \rho_{e,m} \leqslant 1, \quad e \in \mathcal{D}_i \\
& \boldsymbol{\rho}_i \in \{0,1\}^{|\mathcal{D}_i|\times n_M} .
\end{aligned} \tag{4.8}$$

The local sub-problem (4.8) is a binary optimization problem and can be solved by any standard MILP solver with the B&B method in parallel.

The global sub-problem for the multi-cuts formulation (4.6) can be written as:

$$\begin{aligned} \min_{\lambda,\eta} \quad & \eta \\ \text{s.t.} \quad & \widetilde{f}^j(\bar{\rho}) \leqslant \eta, \qquad \forall j \in \mathcal{P}_s \\ & t^j(\bar{\rho}) \leqslant d^j, \qquad \forall j \in \mathcal{P}_s \\ & H_{\mathcal{M}_0}(\bar{\rho}) \leqslant \bar{M} \\ & \sum_{l=1}^{L} \lambda_i^l = 1 \\ & \boldsymbol{\lambda} \in [0,1]^{n_{\mathcal{D}} \times L}, \end{aligned} \tag{4.9}$$

where $\bar{\rho}_{e,m} = \sum_{l=1}^{L} \hat{\rho}_{e,m}^l \lambda_i^l$, $\forall e \in \mathcal{D}_i, m = 1, \dots n_{\mathcal{M}}$. It is a linear programming problem and can be solved by any standard linear programming solver which can calculate the Lagrange multipliers to the constraints in the programming problem (e.g. Gurobi [28]). When solving this problem, one does not need to get the solution of $\boldsymbol{\lambda}$ but only needs the value of Lagrange multipliers π_j^f, π_j^t and $\pi^{\mathcal{M}}$ to the cuts, the trust region and the total mass constraint. If the linear programming solver does not have the feature to calculate the Lagrange multipliers when solving the programming (4.9), the dual problem to (4.9) can be solved instead to get the Lagrange multipliers. The Lagrange multipliers π_j^f, π_j^t and $\pi^{\mathcal{M}}$ to the first three lines in the constraints can be calculated and collected into the vector $\boldsymbol{\pi}$.

4.2.2 Initialization

To initialize the DW decomposition requires a initial guess to the programming (4.1) or (4.2), a reduced size problem with a clustered design variable $\hat{\boldsymbol{\rho}} \in \{0,1\}^{n_{\mathcal{D}} \times n_{\mathcal{M}}}$ is considered. It is expected to use a single vari-

able $\hat{\rho}_{i,m}$ to represent the design variables $\rho_{e,m}$ in the i-th block:

$$\rho_{e,m} = \hat{\rho}_{i,m}, \quad \forall e \in \mathcal{D}_i \, . \tag{4.10}$$

As it aggregates the design variables $\rho_{e,m}$ in the i-th block, the coefficients with respect to the design variables in the original programming (4.2) are also aggregated in the reduced size problem. The reduced size multi-cuts problem can be written as:

$$\begin{aligned}
\min_{\hat{\boldsymbol{\rho}},\eta} \quad & \eta \\
\text{s.t.} \quad & f(\boldsymbol{\rho}^j, \mathbf{u}^j) + \sum_{m=1}^{n_M} \sum_{i=1}^{n_{\mathcal{D}}} \left(\sum_{e \in \mathcal{D}_i} \hat{w}^j_{e,m} \right) \hat{\rho}_{i,m} \\
& \quad - \sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \hat{w}^j_{e,m} \rho^j_{e,m} \leqslant \eta, \quad \forall j \in \mathcal{P}_s \\
& \frac{1}{n_e} \sum_{m=1}^{n_M} \sum_{i=1}^{n_{\mathcal{D}}} \left[\sum_{e \in \mathcal{D}_i} \left(1 - 2 \sum_{m=1}^{n_M} \rho^j_{e,m} \right) \right] \hat{\rho}_{i,m} \\
& \quad + \sum_{e=1}^{n_e} \left(\rho^j_{e,m} \right)^2 \leqslant d^j, \quad \forall j \in \mathcal{P}_s \\
& \sum_{m=1}^{n_M} \sum_{i=1}^{n_{\mathcal{D}}} \frac{\bar{M}_m \cdot |\mathcal{D}_i|}{n_e} \hat{\rho}_{i,m} \leqslant \bar{M}_{max} \\
& \sum_{m=1}^{n_M} \hat{\rho}_{i,m} \leqslant 1, \quad i = 1, 2, \ldots, n_{\mathcal{D}} \\
& \hat{\boldsymbol{\rho}} \in \{0,1\}^{n_{\mathcal{D}} \times n_M} \, .
\end{aligned} \tag{4.11}$$

In case of the reduced size problem (4.11) is infeasible, the initialization procedure should be restarted with a doubled size of $n_{\mathcal{D}}$. This process is repeated until a feasible solution to is found. Based on the numerical experiment, it is rare to encounter the case of infeasible solution during initialization stage. When the solution is feasible, the Lagrange multipliers

to the sensitivity cuts π_j^f, the trust regions π_j^t and the total mass constraint π^M needs to be calculated by the linear programming solver or obtained from the dual problem of (4.9). All of the Lagrange multipliers are collected to a vector and is denoted as $\boldsymbol{\pi}$. The initial value for $\boldsymbol{\rho}$ is then update with (4.10) and is denoted as $\hat{\rho}^1$.

As the size of the clustering design variable $\hat{\boldsymbol{\rho}}$ is only related to the number of blocks $n_{\mathcal{D}}$ and the number of materials n_M, the cost of the initialization is expected to be low for the large-scale TO problems.

4.2.3 Iteration of Modified DW decomposition

The iteration of the modified DW decomposition is similar to the general optimization algorithm. The iteration starts with the initialization of the reduced size problem (4.11). The local sub-problem (4.7) is solved by a standard MILP solver with B&B method. The global sub-problem (4.9) is solved by a linear programming solver. The Lagrange multipliers π_j^f (when necessary), π_j^t and π^M are calculated by the linear programming solver. The design variables $\rho_{e,m,k}$ are updated by the Lagrange multipliers and the design variables $\rho_{e,m}$ in the original programming (4.1) or (4.2) are updated by the design variables $\rho_{e,m,k}$. The iteration stops when the convergence criterion is satisfied. By checking the Lagrange multipliers π_j^f, π_j^t and π^M in the global sub-problem, the convergence criterion can be defined as:

$$\frac{\left\|\boldsymbol{\pi}^l - \boldsymbol{\pi}^{l-1}\right\|_2}{\left\|\boldsymbol{\pi}^{l-1}\right\|_2} \leqslant 10^{-6} \,. \tag{4.12}$$

The detail implementation of the proposed modified DW decomposition is summarized in Algorithm 6.

Algorithm 6 ModifiedDantzigWolfeDecomposition

Input: The single cut formulation (4.1) or the multi-cuts formulation (4.2), and the initial value of $n_{\mathcal{D}}$

Output: The optimal solution $\boldsymbol{\rho}$ and the objective function value η to the programming (4.1) or (4.2)

Utility: Use the modified DW decomposition to solve the programming (4.1) or (4.2)

Calculate $\widetilde{\boldsymbol{\rho}}$ according to (4.4) and sort $\widetilde{\boldsymbol{\rho}}$ in non-descending order
Divide the design variables into $n_{\mathcal{D}}$ blocks according to (4.5)
Initialize the reduced size problem (4.11)
if The initialization is infeasible **then**
    $n_{\mathcal{D}} = 2 \cdot n_{\mathcal{D}}$
    Go to Step 2
else
    Update the design variables $\rho^0_{e,m}$ with the solution of the reduced size problem and update the corresponding Lagrange multiplier $\boldsymbol{\pi}^0$
end if
for $l = 1, 2, \ldots$ **do**
    Solve the local sub-problem (4.8) by a standard MILP solver
    Update the design variables $\rho^l_{e,m}$
    Solve the global sub-problem (4.9) by a linear programming solver and update the corresponding Lagrange multipliers $\boldsymbol{\pi}^l$
    if $\frac{\left|\boldsymbol{\pi}^l - \boldsymbol{\pi}^{l-1}\right|_2}{|\boldsymbol{\pi}^{l-1}|_2} \leqslant 10^{-6}$ **then**
        Break
    end if
end for
Set $\boldsymbol{\rho}^* = \boldsymbol{\rho}^l$

Return: $\boldsymbol{\rho}, \eta$

4.3 Numerical Results

4.3.1 Benchmark Results

A benchmark problem, the bridge design problem with a minimum compliance objective as discuss in §2.2.1, is tested in this section. The resulting objective function value f, the number of FEM evaluations N_{FEM} and the running time is reported in Table 4.1 with two scenarios. Considering the candidate materials listed in Table 2.2, the first scenario only takes steel as the candidate material; the second scenario employs all four materials as candidates. The maximum total mass is set as 3.744 kg, equivalent to 12% volume fraction when only using steel. As the top plane is fixed to using steel in this problem, the free mass can be utilized for optimization is 2.496 kg, equivalent to 8% volume fraction when only using steel. The The reported running time includes the time spent by the FEM evaluations T_{FEM}, the time spent by the optimizer Algorithm 1 solving the MILP problem (4.1) or (4.2) via the the modified DW decomposition T_{OPT}. The benchmark problem is performed on a computation node with two AMD EPYC 7763 64-Core Processors and 512 GB memory. Each test is launched by the Slurm system [85] on a single node using 128 physical cores. The benchmark problem is tested with different discretization resolutions, including $50 \times 200 \times 50$, $100 \times 400 \times 100$, and $150 \times 600 \times 150$. It sets $n_D = 100$ as the initial value when applying the modified DW decomposition and no enlargement of n_D was observed during the initialization stage of the modified DW decomposition in Algorithm 6. For the convenience of implementation, each core only solves the local sub-problem with respect to the local elements to avoid the communication between cores. Therefore, n_D local sub-problems are not assembled in the real implementation but only solved in an alternative way with the mathematical equivalence.

The evaluation of governing equations is performed by *FEniCSx* [8], which based on a recent scalable implementation of *FEniTop* [36] for solv-

Table 4.1: Single-material minimum compliance: Results with different discretization resolutions using the modified DW decomposition.

Resolution	f	N_{FEM}	T_{FEM} (s)	T_{FEM} (%)	T_{OPT} (s)	T_{OPT} (%)
$50 \times 200 \times 50$	15.229	77	303.325	81	67.62	18
$100 \times 400 \times 100$	14.115	67	2282.22	93	163.5	6
$150 \times 600 \times 150$	13.205	47	6233.96	96	200.2	3

Table 4.2: Multi-material minimum compliance: Results with different discretization resolutions using the modified DW decomposition.

Resolution	f	N_{FEM}	T_{FEM} (s)	T_{FEM} (%)	T_{OPT} (s)	T_{OPT} (%)
$50 \times 200 \times 50$	12.957	39	123.601	66	61.05	33
$100 \times 400 \times 100$	13.052	44	1746.55	85	285.2	14
$150 \times 600 \times 150$	12.983	51	9667.09	82	2089	17

ing TO problems. As shown in Table 4.1 and Table 4.2, the evaluation of FEM dominates the total running, especially for the large-scale problems. Compared to directly solving the problem (4.1) or (4.2) with a standard MILP solver as discussed in §4.3.4, the utilization of the the modified DW decomposition greatly reduced the optimization time and made it reasonable optimizer for large-scale TO problems.

The resulting configuration of the optimal topology when only considering a stainless steel as the candidate material is shown in Figure 4.1. As the increase of the discretization resolution, more branches under the top plane have been added into the optimal topology as the details added by the high discretization resolutions.

In the multi-material scenario, though considered four candidate materials, only three of them appear in the optimal topology. This is because titanium is much less efficient material under the setup of minimum compliance and linear elasticity assumption as the specific strength $E_m/\bar{M}_m$ is much lower than other candidate materials. As shown in Figure 4.2, the stainless steel (rendered in red) as the strongest candidate material appears at the center of the structure, especially shown at the corner of

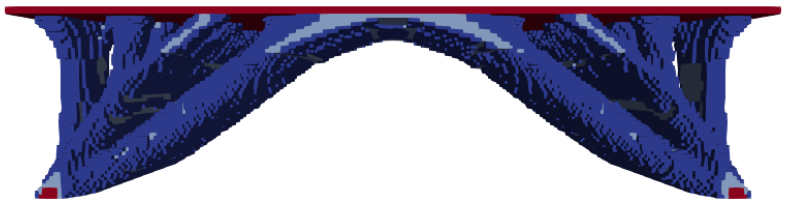
(a) Discretization resolution of $50 \times 200 \times 50$.

(b) Discretization resolution of $100 \times 400 \times 100$.

(c) Discretization resolution of $150 \times 600 \times 150$.

Figure 4.1: The optimal topology of the bridge design problem using single candidate material and different discretization resolutions.

the bridge. The softest candidate material magnesium (rendered in dark blue) has been observed as the out coating of the structure. The second candidate material aluminum then appears as the intermediate material between magnesium and stainless steel working as transition material. The layout of the materials in the optimal topology fits the intrinsic way when manually assigning materials of the structure.



(a) Discretization resolution of $50 \times 200 \times 50$.

(b) Discretization resolution of $100 \times 400 \times 100$.

(c) Discretization resolution of $150 \times 600 \times 150$.

Figure 4.2: The optimal topology of the bridge design problem using four candidate materials and different discretization resolutions. Magnesium is rendered in dark blue; aluminum is rendered in light blue; titanium is rendered in orange; stainless steel is rendered in red.

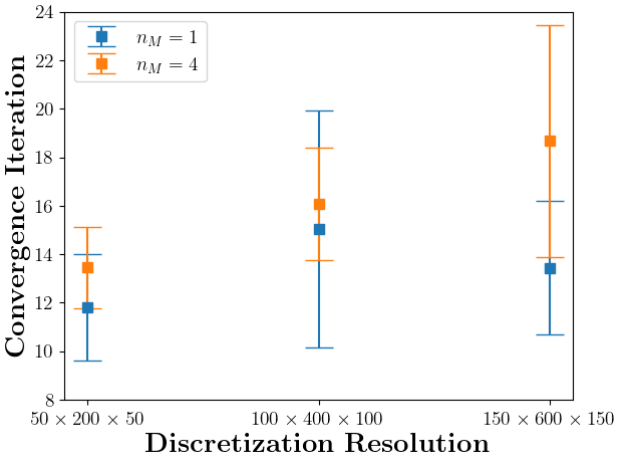
4.3.2 Time Analysis for Solving the Global Sub-problem of the the modified DW decomposition

As the size of the global sub-problem generated from the the modified DW decomposition is determined by the number of blocks $n_{\mathcal{D}}$ and the number of iterations, the relationship between the number of iterations and the size of the design variables is first investigated. It considers the 3D minimum compliance problem with different discretization resolutions and different number of materials up to $n_M = 4$ introduced in Table 2.2.

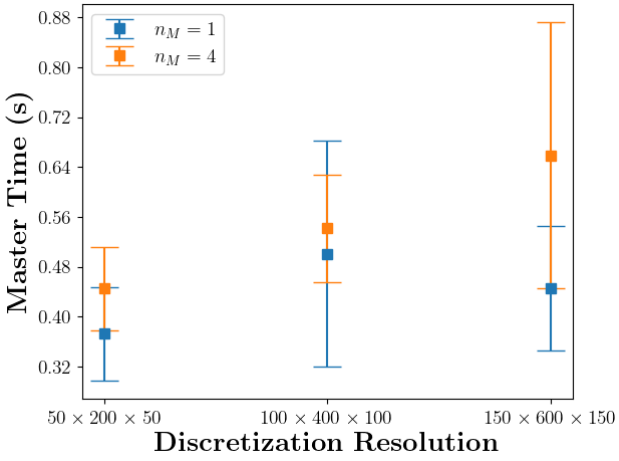
As shown in Figure 4.3a, the average number and its standard deviation of iterations required for convergence does not increase significantly with the discretization resolution for the single material ($n_M = 1$) case. Thus, the size of the global sub-problem generated from the the modified DW decomposition is expected to be the same across different discretization resolutions. As shown in Figure 4.3b, the average time of solving the global sub-problem does not increase significantly with the discretization resolutions. However, when the number of materials increases (e.g. $n_M = 4$), the average number of iterations would increase slightly with respect to the increase of the discretization resolutions. As the global sub-problem (4.9) has to be solved multiple times in the iteration process, the solving time would scale quadratically with respect to the number of iterations. This is because the number of constraints linearly scales with the number of iterations and the complexity of linear programming scales linearly with respect to the product of the number of constraints and the number of variables [59]. As the increase of the average iteration is very slight, the solving time of the global sub-problem does not increase significantly with the discretization resolutions. This indicates that the the modified DW decomposition is still applicable to the problems with complex local constraints (e.g. multiple candidate materials) and different discretization resolutions. In summary, the the modified DW decomposition is expected to be efficient for solving large-scale TO problems.

4.3.3 Comparison to the SIMP Method

The benchmark problem is tested with the standard SIMP method based on a recent scalable implementation of *FEniTop* [36]. In this thesis, it is set to use the optimality criterion (OC) for its optimization performance. The maximum number of iteration is set as 400 and the radius of the Helmholtz filter is set as 0.6. As the SIMP method is not able to select the optimal combination of multiple candidate materials, it only considers



(a) The average number of iteration of the the modified DW decomposition with respect to different discretization resolutions.



(b) The average time of solving the global sub-problem (4.9) with respect to the different discretization resolutions.

Figure 4.3: The scaling analysis of the the modified DW decomposition method on TO problems.

the single material case. The running results are collected in Table 4.3, 4.4 and. In Table 4.3, it compares the resulting objective function value f, the difference of the objective function value between the SIMP method and the the modified DW decomposition $\Delta f = \frac{f_{SIMP} - f_{DW}}{f_{DW}} \times 100\%$. In Table 4.4, it compares the total running spent by the FEM analysis T_F, the average time of FEM analysis $\bar{T}_F$ for each iteration. It also compares the difference of the total and average time of FEM analysis between the SIMP method and the the modified DW decomposition $\Delta T_F = \frac{T_{F,\,SIMP} - T_{F,\,MDW}}{T_{F,\,MDW}} \times 100\%$ and $\Delta \bar{T}_F = \frac{\bar{T}_{F,\,SIMP} - \bar{T}_{F,\,MDW}}{\bar{T}_{F,\,MDW}} \times 100\%$, respectively.

Table 4.3: Single-material minimum compliance: Comparison between the modified DW decomposition and the SIMP method in terms of the objective function.

Resolution	**Modified DW Decomposition**		**SIMP** ($N_{FEM} = 200$)		**SIMP** ($N_{FEM} = 400$)	
	f	N_{FEM}	f	Δf (%)	f	Δf (%)
$50 \times 200 \times 50$	15.229	77	19.172	25.89	16.148	6.035
$100 \times 400 \times 100$	14.115	67	18.418	30.49	14.499	2.721
$150 \times 600 \times 150$	13.205	47	18.231	38.06	14.048	6.384

Table 4.4: Single-material minimum compliance: Comparison between the modified DW decomposition and the SIMP method in terms of the time of FEM analysis.

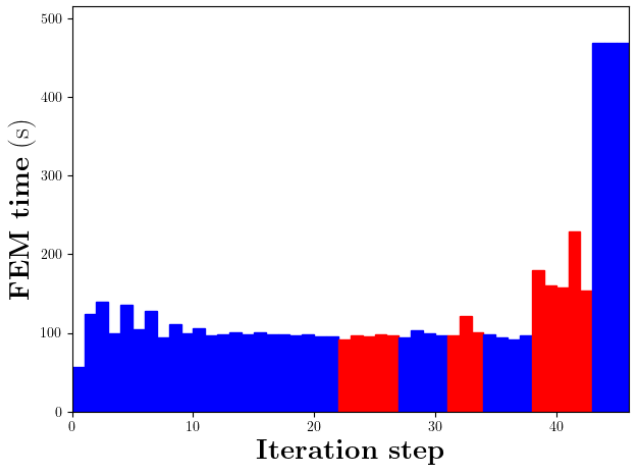
Resolution	**Modified DW Decomposition**			**SIMP** ($N_{FEM} = 200$)				**SIMP** ($N_{FEM} = 400$)			
	N_{FEM}	T_F (s)	$\bar{T}_F$ (s)	T_F (s)	ΔT_F (%)	$\bar{T}_F$ (s)	$\Delta \bar{T}_F$ (%)	T_F (s)	ΔT_F (%)	$\bar{T}_F$ (s)	$\Delta \bar{T}_F$ (%)
$50 \times 200 \times 50$	77	303.3	3.939	1009.6	232.9	5.048	28.15	2810.1	826.4	7.025	78.34
$100 \times 400 \times 100$	67	2282	34.06	9105.5	299.0	45.53	33.66	20007	776.7	50.02	46.84
$150 \times 600 \times 150$	47	6234	132.6	36354	483.2	181.8	37.04	82865	1229	207.2	56.19

As shown in Table 4.4, the proposed the modified DW decomposition method outperforms the SIMP method in terms of the objective function value and the average time of FEM analysis. When using a small number of iterations when using the SIMP method (e.g. $N_{FEM} = 200$), the the modified DW decomposition method achieves a better objective function value, with a difference larger than 20%. When the number of iterations

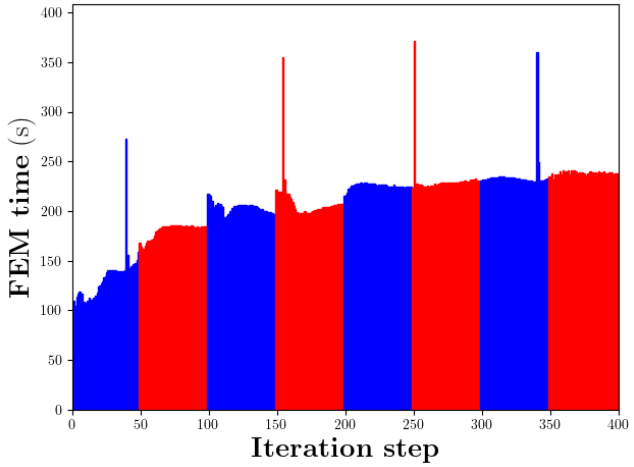
increases (e.g. $N_{FEM} = 400$), the difference of the objective function value between the SIMP method and the the modified DW decomposition can get reduced to less than 5%.

However, increasing the number of iterations is very expensive as the time of FEM analysis can increase very significantly. The average time of FEM analysis scales with respect to the number of iterations as shown in Table 4.4. As the evolution of the optimization process, the continuous design variable gets close to 0 or 1. This worses the conditioning of the stiffness matrix and elongates the time of FEM analysis as shown in Section 4.3.3. In contrast, the proposed framework does not to sustain such issue since the parameter relaxation discussed in §2.4 improves the conditioning of the problem at the first stages of the optimization process when using a single minimum Young's modulus factor (e.g. $E_{min} = 10^{-2}$). Other than the last two stages as shown in Section 4.3.3, the time of FEM analysis is very stable and relatively small when using the proposed framework. In the last two stages, as the total number of iterations is very limited (no more than 10 in total), this does not significantly increase the total time of FEM analysis. The parameter relaxation scheme is then very effective for reducing the average running time of FEM analysis while the SIMP method can hardly utilize the scheme in practice. This in turn makes the proposed framework more efficient than the SIMP method in terms of the total time of FEM analysis. For example, when using the discretization resolution of $150 \times 600 \times 150$, the total time of FEM analysis of the proposed framework is one order of magnitude smaller than that of the SIMP method.

The SIMP method presented in this thesis uses an optimizer based on the optimality criteria (OC) to solve the optimization problem. The OC method is very efficient as the average optimization time in each iteration $\bar{T}_O$ is much shorter than the average optimization of the the modified DW decomposition as shown in Table 4.5. It is only the reason that the total



(a) the modified DW decomposition



(b) SIMP method

Figure 4.4: Single-material minimum compliance: Time of FEM analysis for each iteration step with a discretization resolution of $150 \times 600 \times 150$. Different colors indicate the different stages of the optimization process.

number of iterations is very large that the total optimization time T_O of the SIMP method is larger than that of the proposed method.

The optimal topologies obtained by the SIMP method are reported in Figure 4.5. The visualized results are filtered with a threshold of 0.5 for

showing the configuration of the structure. Compared to the topology generated by the the modified DW decomposition as shown in Figure 4.1, the topology shows a better consistency between different discretization resolutions in terms of the number, the orientation and the size of the branches shown in the structure. By comparing the left and right half of the topology, the most apparent difference happens at the change of the orientation of the branch near the center of the bridge. Instead, the orientation and the size of the branches at the rear ends of the bridge does not significantly change. As the objective function value f has been greatly reduced as shown in Table 4.4, this implies that the additional iterations, though only modify part of the topology of the structure, can still lead to significant performance improvements of the design and necessitate the advantage of a sharp 0-1 configuration provided by the additional iterations.

4.3.4 Comparison to the general MILP solver

In order to consistently compare the performance of the the modified DW decomposition with the general MILP solver, the optimization solutions used in the optimization process of TO benchmark problems are obtained from the the modified DW decomposition. Each MILP problems (4.2) and (4.1)is solved both by the the modified DW decomposition and the

Table 4.5: Single-material minimum compliance: Comparison between the modified DW decomposition and the SIMP method in terms of the time of optimization.

Resolution	**Proposed Method**		**SIMP**			
	T_O (s)	$\bar{T}_O$ (s)	T_O (s)	$\bar{T}_O$ (s)	ΔT_O (%)	$\Delta \bar{T}_O$ (%)
$50 \times 200 \times 50$	67.76	0.880	11.74	0.029	−82.7	−96.7
$100 \times 400 \times 100$	163.8	2.445	254.4	0.636	55.33	−74.0
$150 \times 600 \times 150$	201.2	4.280	1298	3.244	545.0	−24.2

(a) Discretization resolution of $50 \times 200 \times 50$.

(b) Discretization resolution of $100 \times 400 \times 100$.

(c) Discretization resolution of $150 \times 600 \times 150$.

Figure 4.5: The optimal topology of the bridge design problem using a single candidate material and different discretization resolutions. The left half shows the topology obtained at $N_{FEM} = 200$; the right half shows the topology obtained at $N_{FEM} = 400$. The visualized results are filtered with a threshold of 0.5, colored with the design variable ρ.

standard MILP solver Gurobi [28]. The tolerance of the MILP solver is set as 10^{-9}.

4.3.4.1 Single material case

The total running time that directly using the standard MILP solver is reported in Table 4.6, with a reference of the total FEM analysis time T_F. As shown in the table, the optimization time of the MILP solver $T_{O,\,MILP}$ is much longer than that of the the modified DW decomposition $T_{O,\,MDW}$.

The difference can be larger than one magnitude of order as indicated by the relative time difference $\Delta T_O = \frac{T_{O,\,MILP} - T_{O,\,MDW}}{T_{O,\,MDW}} \times 100\%$. Taking the time spent by the FEM analysis as a reference, the optimization time of the MILP solver is comparable to the time of FEM analysis. This indicates that the optimization time of the MILP solver is not acceptable for large-scale TO problems. In contrast, the optimization time of the the modified DW decomposition is much smaller than the time of FEM analysis. This indicates that the the modified DW decomposition is a reasonable optimizer for large-scale TO problems.

Table 4.6: Single-material minimum compliance: Comparison between the modified DW decomposition and the standard MILP solver in terms of the time of optimization.

Resolution	T_F (s)	$T_{O,\,MDW}$ (s)	$T_{O,\,MILP}$ (s)	ΔT_O (%)
$50 \times 200 \times 50$	303.325	67.616	268.11	296.5
$100 \times 400 \times 100$	2282.22	163.45	2824.8	1628
$150 \times 600 \times 150$	6233.96	200.24	5254.6	2524

4.3.4.2 Multi-material case

Compared to the case of single material, the optimization time of the MILP solver is much longer as shown in Table 4.7. In case of the discretization resolution of $150 \times 600 \times 150$, the optimization time of the MILP solver is too long that it is not reported in the table. The optimization time of the MILP solver $T_{O,\,MILP}$ is much larger than the time of FEM analysis T_F. It indicates that the optimization time of the MILP solver is not scalable in the case of multi-material TO problems and is not acceptable for large-scale TO problems. In contrast, the the modified DW decomposition is still a reasonable optimizer as the optimization time $T_{O,\,MDW}$ is much smaller than the time of FEM analysis T_F.

Table 4.7: Multi-material minimum compliance: Comparison between the modified DW decomposition and the standard MILP solver in terms of the time of optimization.

Resolution	T_F (s)	$T_{O,\,MDW}$ (s)	$T_{O,\,MILP}$ (s)	ΔT_O (%)
$50 \times 200 \times 50$	123.601	61.054	6118.6	9922
$100 \times 400 \times 100$	1746.55	285.21	349929	122590
$150 \times 600 \times 150$	9667.09	2089.3	—	—

5 QUANTUM OPTIMIZER FOR DISCRETE VARIABLE TOPOLOGY OPTIMIZATION

5.1 Quantum Computing for Optimization

Quantum computing (QC) is emerging as a new computing paradigm that could be superior to classical computing for a variety of problems, especially when performing binary programming problems. The Quantum Annealing (QA) [4] and the Quantum Approximate Optimization Algorithm (QAOA) [25] method is the two most protege algorithm that can handle binary programming problems and is promised to out-perform the best classical optimizers on these problems. As the QAOA method requires the quantum circuit which is still limited in the size of available qubits at present, the QA method is expected to be suitable for testing the proposed algorithms on the real devices. In this chapter, all proposed algorithms and all running results are performed and collected from the real QA devices.

5.1.1 Quantum Annealing

The quantum annealing accepts the quadratic unconstrained binary optimization (QUBO) problem

$$J = \mathbf{x}^T \mathbf{Q} \mathbf{x} = \sum_{i=1}^{n} Q_{ii} x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} Q_{ij} x_i x_j, \quad \mathbf{x} \in \{0,1\}^n \tag{5.1}$$

as an input to find the solution $\mathbf{x}^*$ that minimizes J. To find the solution, the quantum annealer, the physical device to perform the quantum annealing,

takes the Hamiltonian

$$\begin{aligned} H(s) = &-A(s)\left(\sum_{i=1}^{n}\sigma_i^{(x)}\right) \\ &+B(s)\left(\sum_{i=1}^{n}Q_{ii}\sigma_j^{(z)}+\sum_{i=1}^{n}\sum_{j=i+1}^{n}Q_{ij}\sigma_i^{(z)}\sigma_j^{(z)}\right), \quad s\in[0,1] \end{aligned} \tag{5.2}$$

where $A(s)$ and $B(s)$ are functions of time, $s = t/t_f$ is the normalized annealing time and t_f is the physical annealing time that can be set in front of the execution of the Hamiltonian on the physical device.

5.2 Preliminary of QUBO

As quantum optimization, e,g, QA and QAOA, only accepts QUBO problems as inputs, it is required to introduce the commonly used approaches for the conversion from general optimization problems into a QUBO problem as preliminary for the following discussion on the proposed algorithms for acceleration of classical implementation proposed in the previous chapters.

The discussion starts from the representation of general continuous or integer variables in the binary space. Then it turns to the treatment of forming QUBO problems for equality and inequality constraints. Finally, it discusses the scaling of the QUBO problems and how to manage the form of generated QUBO problems on quantum devices.

5.2.1 Basis Encoding

When performing (mixed) binary optimization on quantum devices, it considers the basis encoding of different variables, including both the continuous and integer variables. The continuous and integer variables are encoded into binary variables through the following expansion functions.

For a continuous variable χ in the range $[a, b]$, it can use n binary variables x_i to any value within in the range $[a, b]$ as

$$\widetilde{\chi}(\mathbf{x}, a, b) = a + \frac{b-a}{2^{n_\chi}} x_0 + \sum_{i=1}^{n_\chi - 1} \frac{b-a}{2^i} x_i, \quad \mathbf{x} \in \mathbb{B}^n \tag{5.3}$$

By increasing n_χ, it can improve the approximation accuracy to the original continuous variable χ. The number of binary variables n required for the expansion of the continuous variable χ with the accuracy ε, such that $|\widetilde{\chi} - \chi| < \varepsilon$, can be estimated as $n = \left\lceil \log_2 \left(\frac{b-a}{\epsilon} \right) \right\rceil$. This kind of digits is also known as fixed-point representation.

For an arbitrary integer variable ψ in the range $[c, d], c, d \in \mathbb{Z}$, it can use $n = \lceil \log_2(d - c + 1) \rceil$ binary variables y_i to exactly represent each integer within the range $[c, d]$ as

$$\psi(\mathbf{y}, c, d) = c + \sum_{i=0}^{n-2} 2^i y_i + C_0 y_{n-1}, \quad \mathbf{y} \in \mathbb{B}^n \ , \tag{5.4}$$

where $C_0 = (d - c + 1 - 2^{\lceil \log_2(d-c+1) \rceil - 1})$.

5.2.2 From Constraints to QUBO

The equality constraints can be integrated into a unconstrained through the penalty method. For example, the following linear constrained minimization problem

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & \mathbf{G}\mathbf{x} = 0 \\ & \mathbf{x} \in \mathbb{B}^n \end{aligned} \tag{5.5}$$

where $f(\mathbf{x})$ is the objective function $\mathbf{x}$. It can be converted into a unconstrained problem as $\min f(\mathbf{x}) + P g^2(\mathbf{x})$, where P is a large penalty factor constant. Due to the limited accuracy of currently available device, the

penalty factor can not be an arbitrary large value. it is preferred to select a large value with respect to the maximum possible value of the objective function $f(\mathbf{x})$. As $\mathbf{x}$ is a binary vector, an estimated upper bound of $f(\mathbf{x})$ can be $\sum_{i=1}^{n} |f_i|$. Then P can be taken as $P = c \sum_{i}^{n} |f_i|$, where c is constant in the range of $[0, 10]$.

In terms of inequality constraints

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h(\mathbf{x}) \leqslant 0 \\ & \mathbf{x} \in \mathbb{B}^n \end{aligned} \tag{5.6}$$

where $h(\mathbf{x})$ is a linear function with respect to the binary variable $\mathbf{x}$. It first requires the introduction a slackness variable to convert the inequality constraint $h(\mathbf{x}) \leqslant 0$ into an equality constraint. As $\mathbf{x}$ is bounded, $h(\mathbf{x})$ can be bounded in the range $[a, b]$. One possible selection of a and b can be

$$a = \sum_{i}^{n} -|h_i|, \quad b = \sum_{i}^{n} |h_i| \tag{5.7}$$

One can also find other tighter bounds to replace the current selection of a and b. Once a and b is selected, the introduced slackness variable can then be represented with a series of binary variables $\mathbf{z}$ within the range $[a, b]$. If the coefficients of $h(\mathbf{x})$ are all integers, then $\chi(\mathbf{z}, a, b)$ can be selected as the expansion function. If the coefficients of $h(\mathbf{x})$ contain fractional values, $\psi(\mathbf{z}, a, b)$ can then be selected as the expansion function to the slackness variable. For both cases, one can get the following problem with equality constraint only as:

$$\begin{aligned} \min \quad & f(\mathbf{x}) \\ \text{s.t.} \quad & h(\mathbf{x}) + \phi(\mathbf{z}, a, b) = 0 \\ & \mathbf{x} \in \mathbb{B}^n, \quad \mathbf{z} \in \mathbb{B}^{n_z} \end{aligned} \tag{5.8}$$

where $\phi(\mathbf{z}, a, b)$ is the selected expansion function according to the prop-

erty of the coefficients in $h(\mathbf{x})$ and n_z is the used number of binary variables used for the expansion of the slackness variable. Then the problem (5.8) is an equality constrained linear binary programming with respect to the binary variables $\mathbf{x}$ and $\mathbf{z}$. The unconstrained problem then can be formed according to the procedure introduced in the first part in this section.

5.3 Quantum Implementation of the Discrete Topology Optimization Problem

5.3.1 Direct Conversion of the Discrete Topology Optimization Problem

To solve the continuum TO problem with quantum annealing, it can be achieved by directly converting the bilinear problem (2.36) into a QUBO problem. By using the penalty method, the inequality constraints in the problem (2.36) can be converted into the penalty terms in the objective function. There exist two bilinear inequality constraints $\widetilde{f}^j(\boldsymbol{\rho})\alpha_j - \Pi(1-\alpha_j) \leqslant \eta$ and $t^j(\boldsymbol{\rho})\alpha_j \leqslant d^j$ in the problem (2.36). The penalty terms would contain cubic term that is not suitable for near-term quantum devices which can only handle quadratic terms. Thus, it requires the introduction of additional binary variables to convert the cubic terms into quadratic ones [37]. However, to solve the problem (2.36) through this approach requires a lot of logical qubits (three times of the design variables) to embed the problem onto a real equipment. The required qubits scale with respect to the number of design variables, which has greatly exceeded the capability of the near-term available devices.

5.3.2 Heuristic Approach to Solve the Continuum Topology Optimization Problem

The second approach has been proposed in the previous paper [84], the problem (4.2) can be reorganized into smaller sub-problems according to the solution to each cut i in $\mathcal{P}_s$. As shown in the paper [84], this approach can greatly reduce the size of problem needed to be embedded onto the quantum devices, but it requires that the problem (4.1) can be efficiently solved. This requirement cannot be satisfied with the introduction of the complex constraints on the design variables, e.g. the multi-material constraints. Furthermore, the number of logic qubits required to embed the problem scales with respect to the increase of the discretization resolution as shown in the previous paper [84]. It makes the approach hardly competitive in a large-scale TO problem.

In this thesis, a new heuristic approach is considered to find the solution to the problem (4.1) and (4.2) by utilizing linear programming and the quantum annealing. The fractional coefficients in the total mass constraint makes the solution obtained from the linear programming containing fractional values by a chance. It is no more avoidable to utilize a binary programming solver and ensure a binary solution. In order to reduce the cost of solving the binary constrained problem (4.1) and (4.2), a two-step heuristic approach is deployed. In the first step, a relaxed problem of (4.1) and (4.2) is solved, by removing the integrity constraint on $\boldsymbol{\rho}$ with the solution $\boldsymbol{\rho}^*$, through an off-the-shelf linear programming solver. In this step, most of the design variables is expected to be binary in the solution. Only partial of the fractional solution needs to be fixed to binary values. Then the variables taking binary values can then be picked out to form a new reduced-size binary problem to find the proper values. In the second step, once the solution $\boldsymbol{\rho}^*$ taking any non-binary values, a reduced binary programming problem can then be formed according the non-binary variables in $\boldsymbol{\rho}^*$. Such non-binary variables can form a

subset $\tilde{\boldsymbol{\rho}} \subset \boldsymbol{\rho}^*$, where $\tilde{\boldsymbol{\rho}} = \left\{\rho_{e,m} | 0 < \rho^*_{e,m} < 1, \rho^*_{e,m} \in \boldsymbol{\rho}^*\right\}$. Therefore, the reduced binary programming problem to the single cut problem (4.1) and multi-cut problem (4.2) can be written as:

$$\begin{aligned}
\min_{\boldsymbol{\rho}} \quad & \widetilde{f}^j(\boldsymbol{\rho}) \\
\text{s.t.} \quad & t^j(\boldsymbol{\rho}) \leqslant d \\
& \sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \frac{\bar{M}_m}{n_e} \rho_{e,m} \leqslant \bar{M} \\
& \sum_{m=1}^{n_M} \rho_{e,m} \leqslant 1, \quad e = 1, 2, \ldots, n_e \\
& \boldsymbol{\rho} \in \{0,1\}^{n_e \times n_m} \\
& \rho_{e,m} = \rho^*_{e,m}, \quad \rho_{e,m} \notin \tilde{\boldsymbol{\rho}}\,,
\end{aligned} \tag{5.9}$$

and

$$\begin{aligned}
\min_{\boldsymbol{\rho},\eta} \quad & \eta \\
\text{s.t.} \quad & \widetilde{f}^j(\boldsymbol{\rho}) \leqslant \eta, \quad \forall j \in \mathcal{P}_t(k) \\
& t^j(\boldsymbol{\rho}) \leqslant d, \quad \forall j \in \mathcal{P}_t(k) \\
& \sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \frac{\bar{M}_m}{n_e} \rho_{e,m} \leqslant \bar{M} \\
& \sum_{m=1}^{n_M} \rho_{e,m} \leqslant 1, \quad e = 1, 2, \ldots, n_e \\
& \boldsymbol{\rho} \in \{0,1\}^{n_e \times n_m} \\
& \rho_{e,m} = \rho^*_{e,m}, \quad \rho_{e,m} \notin \tilde{\boldsymbol{\rho}}\,.
\end{aligned} \tag{5.10}$$

5.3.3 Forming QUBO Problem

As the problem (5.9) only contains binary variables in the problem, the conversion of the problem from the constrained binary programming to QUBO only needs the introduction of slackness variables and the penalty

method to form the QUBO. By following the expansion of integer variables in 5.2, the resulting constrained problem can then be written as:

$$
\begin{aligned}
\min_{\boldsymbol{\rho}} \quad & \widetilde{f}^{j}(\boldsymbol{\rho}) \\
\text{s.t.} \quad & t^{j}(\boldsymbol{\rho}) + \chi_t(\mathbf{z}_t, t_{min}, t_{max}) = d \\
& \sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \frac{\bar{M}_m}{n_e} \rho_{e,m} + \chi_M(\mathbf{z}_M, \bar{M}_{min}, \bar{M}_{max}) = \bar{M} \\
& \sum_{m=1}^{n_M} \rho_{e,m} + z_e = 1, \quad e = 1, 2, \ldots, n_e \\
& \boldsymbol{\rho} \in \{0,1\}^{n_e \times n_m} \\
& \rho_{e,m} = \rho^{*}_{e,m}, \quad \rho_{e,m} \notin \tilde{\boldsymbol{\rho}} \\
& \mathbf{z}_t, \mathbf{z}_M \in \mathbb{B}, z_e \in \{0,1\}\,.
\end{aligned}
\tag{5.11}
$$

Therefore, the resulting QUBO reads as:

$$
\begin{aligned}
& \tilde{f}^{j}(\boldsymbol{\rho}) + P_t \left[t^{j}(\boldsymbol{\rho}) + \chi_t(\mathbf{z}_t, t_{min}, t_{max}) - d \right]^2 \\
& + P_M \left[\sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \frac{\bar{M}_m}{n_e} \rho_{e,m} + \psi_M(\mathbf{z}_M, \bar{M}_{min}, \bar{M}_{max}) - \bar{M} \right]^2 \\
& + \sum_{e \in \tilde{\boldsymbol{\rho}}} P_e \left[\sum_{m=1}^{n_M} \rho_{e,m} + z_e - 1 \right]^2 ,
\end{aligned}
\tag{5.12}
$$

where P_t, P_M and P_e are penalty factors and is selected as the upper bound value of $\widetilde{f}^{j}(\boldsymbol{\rho})$.

As η is a continuous variable and the cut $\widetilde{f}^{j}(\boldsymbol{\rho})$ takes non-identical fractional coefficients, the expansion of continuous variables in 5.2 is required for the conversion of the problem (5.10) for forming a QUBO. After the

introduction of slackness variables, the QUBO reads as:

$$
\begin{aligned}
&\psi_\eta(\mathbf{z}_\eta, \eta_{\min}, \eta_{\max}) \\
&+ \sum_{j \in \mathcal{P}_t(k)} P^j \left[\widetilde{f^j}(\boldsymbol{\rho}) + \psi_{f^j}(\mathbf{z}_{f^j}, f^j_{\min}, f^j_{\max}) - \psi_\eta(\mathbf{z}_\eta, \eta_{\min}, \eta_{\max})\right]^2 \\
&+ \sum_{j \in \mathcal{P}_t(k)} P_{t^j} \left[t^j(\boldsymbol{\rho}) + \chi_t(\mathbf{z}_{t^j}, t^j_{\min}, t^j_{\max}) - d\right]^2 \\
&+ P_M \left[\sum_{e=1}^{n_e} \sum_{m=1}^{n_M} \frac{\bar{M}_m}{n_e} \rho_{e,m} + \psi_M(\mathbf{z}_M, \bar{M}_{\min}, \bar{M}_{\max}) - \bar{M}\right]^2 \\
&+ \sum_{e \in \tilde{\boldsymbol{\rho}}} P_e \left[\sum_{m=1}^{n_M} \rho_{e,m} + z_e - 1\right]^2,
\end{aligned}
\tag{5.13}
$$

5.3.4 Quantum Implementation of Modified DW Decomposition

The modified DW decomposition discussed in §4.2 proposes a systematic way to decompose the mixed integer programming problem like (4.2) into a series of two types of sub-problems: binary optimization problems (4.8); and linear programming problems (4.9). The linear programming problems can be solved by off-the-shelf linear programming solvers, while the binary optimization problems can be solved by quantum annealers. In this section, it is focused on the quantum implementation of the binary optimization problems for problems (4.8).

For the problem (4.8), the binary constraints are the same and can be converted to the QUBO form as:

$$
\sum_{e \in \mathcal{D}_i} \left[\sum_{m=1}^{n_M} \rho_{e,m} + \phi_e - 1\right]^2, \tag{5.14}
$$

where ϕ_e is the introduced slackness variable for the binary constraint

restricting the material usage on element e. As objective function is linear for both problems, forming the QUBO for each problem (4.8) requires only $(n_M+1)\times|\mathcal{D}_i|$ physical qubits in sequence or $(n_M+1)\times n_e$ physical qubits in parallel, respectively.

The QUBO for the multi-cuts problem (4.8) is given by:

$$\begin{aligned}&\sum_{e\in\mathcal{D}_i}\sum_{j\in\mathcal{P}_s}\pi_j^f\widetilde{f}^j(\boldsymbol{\rho}_i)+\sum_{j\in\mathcal{P}_s}\pi_j^t t^j(\boldsymbol{\rho}_i)+\pi^M H_{M_0}(\boldsymbol{\rho}_i)\\&+\Pi\sum_{e\in\mathcal{D}_i}\left[\sum_{m=1}^{n_M}\rho_{e,m}+\phi_e-1\right]^2.\end{aligned}\tag{5.15}$$

Π in the above equations is a large positive constant to ensure the binary constraints are satisfied. As the sensitivity cut $\widetilde{f}^j$, the trust region constraint t^j and the total mass constraint H_{M_0} are bounded with respect to the design variables $\boldsymbol{\rho}$, Π can then be selected with respect to the upper bounds of these constraints as:

$$\Pi=\max\left(\pi_j^f\sum_{e\in\mathcal{D}_i}\sum_{m=1}^{n_M}|\hat{w}_{e,m}^j|,\ \pi_j^t d^j,\ \pi^M\bar{M}_{max}\right),\tag{5.16}$$

for the multi-cuts problem (4.8).

5.4 Numerical Results

5.4.1 Acceleration via Heuristic Approach

The minimum compliance, as formulated in §2.2.1, is concerned in this subsection. To solve this TO problem, it followed the proposed methodology, consisting of the GBD, conversion of the MILP into QUBO, and the splitting approach. More specifically, GBD mentioned in §2.3.1 was executed, but with the heuristic approach proposed in §5.3.2. The binary programming problem (5.10) was solved on the quantum annealer

provided by D-Wave. Considering the accuracy of the current quantum devices and the required number of qubits, the parameters for the binary approximations were set as: $n_\eta = n_{\alpha^j} = 10$ in (5.3). Once the solution of the design variable $\boldsymbol{\rho}^{k+1}$ is determined, the problem (5.10) can be treated as a continuous optimization problem, from which the continuous variable η^k can be solved for its slightly better accuracy.

To implement the problem (5.10) on the quantum annealer provided by D-Wave, two different ways were examined. In the first way, it directly embedded the QUBO problem (5.13) in the QPU, which is named as "*Heuristic-Direct*". In the second way, the problem (5.10) were solved by taking advantage of the hybrid solver provided by D-Wave, particularly the constrained quadratic model (CQM), which is hence referred to as "*Heuristic-CQM*". The results are compared in terms of solution quality and wall time, as summarized in Table 5.1, where the discretization resolution is chosen as 60×20; the parameter for filtering, as described in Eq. (2.22), is set as $r = 2$; T denotes the total wall time spent for finding the optimal topology layout; f denotes the objective function's value obtained corresponding to the optimal material layout; and N_B denotes the total number of binary programming. Details about each implementation and the main findings are provided as below.

Table 5.1: Comparison between different implementations for the reduced binary global sub-problem (5.10) on the quantum annealer provided by D-Wave, where T denotes the total wall time spent for finding the optimal topology.

Resolution	n_ρ	r	**Heuristic-Direct**			**Heuristic-CQM**		
			T (s)	f	N_B	T (s)	f	N_B
60×20	1200	2	234.12	186.3727	80	111.97	178.9243	74

In *Heuristic-Direct*, each sampling is set with 20μs annealing time, and the sampling is repeated for 1000 times. The final states reached at the end of annealing were recorded, and the state with the lowest energy among

1000 samplings was regarded as the ground state of the Hamiltonian defined in (5.13) and hence the solution to the binary programming problem (5.10). From the statistics, totally 80 binary programming problems were invoked, 11 among which are the problem (5.13) and were solved by the quantum annealer. For each, it is carefully examined with the number of the resultant reduced variables ($|\mathcal{J}^C|$), the number of physical qubits required for embedding the problem (5.13), and the solution time spent on the QPU, as summarized in Table 5.2, where the 11 problems are organized according to their sequence of being solved. With the discretization

Table 5.2: Statistics about the 11 QUBO problems (5.13) solved in the implementation of *Heuristic-Direct*, where T_{QUBO} denotes the time spent for embedding and annealing on the QPU.

$\lvert\mathcal{J}^C\rvert$	n_q^r	n_q^e	T_{QUBO} (s)
16	49	198	6.20
20	53	234	7.11
34	67	512	16.63
22	55	260	8.57
23	56	559	29.05
27	60	498	21.01
23	46	432	32.71
14	47	172	4.20
32	65	423	13.23
18	51	212	10.52
33	66	616	33.99

resolution of 60×20, the number of design variables is $n_\rho = 1200$. From the value of $|\mathcal{J}^C|$ reported in Table 5.2, it is seen that the splitting approach proposed in §5.3.2 has greatly reduced the size of the problem to be solved on a quantum computer. Note that the total *logical* qubits needed to represent the problem (5.13) should also include those required to represent the binary approximations of η and α^j. Thus, the number of logical qubits required in total is: $n_q^r = |\mathcal{J}^C| + n_\eta + 1 + |\mathcal{P}(k)| + \sum_{j\in\mathcal{P}(k)} n_{\alpha^j}$. By reducing

$|\mathcal{J}^C|$, n_q^r can be reduced accordingly. If the *physical* qubits have all-to-all connections, the number of physical qubits required would be consistent with n_q^r. However, due to the sparse connectivity provided by the current quantum annealing machines, the number of physical qubits required for embedding the problem, denoted as n_q^e is in fact much larger than n_q^r [40], as reported in Table 5.2. Also, with the increase of n_q^r, n_q^e can grow very fast [40], although fluctuating due to inhomogeneous connections between qubits. As a result, most of the time spent on the QPU is dominated by the embedding overhead, as indicated by the values of T_{QUBO} in Table 5.2, noting that the total annealing time for 1000 repetitions of sampling is only 20 ms. The current D-Wave Advantage system permits access to no more than 5000 qubits. To constrain n_q^e not beyond 5000, $|\mathcal{J}^C|$ has to be no more than a few hundreds. Thus, even with the splitting approach, the number of design variables in the original problem must be limited to moderate, which makes solving the minimum compliance problem with finer discretization resolutions challenging.

To tackle this challenge, the second way of implementation was pursued, i.e., *Heuristic-CQM*, where the problem (5.10) was submitted, and the CQM, provided by D-Wave, was called for embedding and solving the problem. All default setups were used when employing the CQM hybrid solver in all the numerical tests. As indicated by the wall time T in Table 5.1, the implementation through CQM greatly saved the entire solving time. The reason for that is the CQM can further reduce the size of the problem embedded on the QPU and in turn save the time spent for embedding, knowing the fact that embedding dominates the time consumed on the quantum devices. In addition, the implementation through CQM results in fewer binary programming problems invoked and a lower value for the objective function. Recalling the discussion about the inexact solution in §5.3.2 that any inexact solution to the problem (4.2) can potentially increase the number of GBD iterations, fewer binary programming

problem invoked suggest that the solution found by the CQM is closer to the exact solution. Possible reasons for that include: the CQM further reduces the problem embedded on the QPU and makes the resultant optimization problem easier to be accurately solved; in *Heuristic-Direct*, the values chosen for the penalty factors may not be optimal, and the probability of recovering the global optimal solution is highly dependent on the embedding and annealing schedule.

Based on the above comparison and findings, when it was scaled up the minimum compliance problem with increasing numbers of design variables (with finer discretization resolutions), the implementation of *Heuristic-CQM* was employed, owing to its better performance in terms of the solution quality, wall time, and the capacity to embed larger problems. In particular, two different shapes of material domains were considered: $L : H = 3 : 1$ and $L : H = 2 : 1$, and the discretization resolution varies from 120×40 to 480×240, leading to the number of design variables (n_ρ) varying from 4800 to 115200. The results about the obtained minimum values of the objective function, the associated computing time, and the corresponding optimal material layouts are presented in Table 5.3 and Figure 5.1.

Table 5.3: Comparison between different methods for solving the minimum compliance problem.

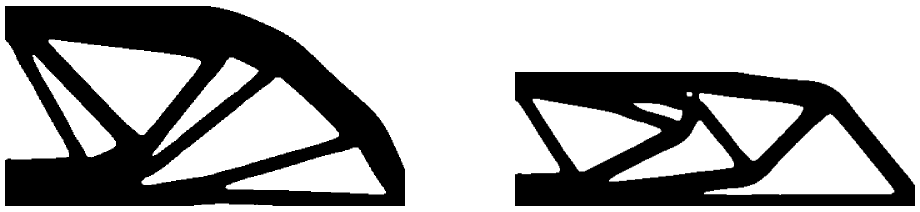
Resolution	n_ρ	r	**Heuristic-CQM**			**SIMP**		
			T (s)	f	N_{FEM}	T (s)	f	N_{FEM}
120×40	4800	2	134.18	183.6182	74	19.84	188.176	433
120×60	7200	2	61.51	75.1399	77	37.58	77.3887	450
240×80	19200	4	123.34	183.4272	73	132.44	188.887	530
240×120	28800	4	88.21	78.2722	89	255.84	78.8675	665
480×160	76800	8	193.90	185.1250	89	1861.68	190.777	1000
480×240	115200	8	261.34	79.3158	90	2929.59	81.1982	1000

Taking the discretization resolution of 480×160 as an example, the run time of the D-Wave's hybrid CQM solver for solving problem (4.11)

was examined. Out of the total 89 iterations required for solving the TO problem (2.31), 6 iterations involve solving (4.11), and their statistics of run time are summarized in Table 5.4. Here, the size of problem (4.11) reduced by the proposed splitting approach is reported as $|\mathcal{J}^C|$, which is significantly smaller than the original number of variables, $n_\rho = 76800$. The total time spent by the CQM is denoted as T_{CQM}. T_{QPU} reports the time spent for annealing in QPUs, which is expected to be short.

Table 5.4: Run time of the D-Wave's hybrid CQM solver for solving problem (4.11), where the discretization resolution is 480×160, T_{QPU} denotes the annealing time spent on QPUs, and T_{CQM} is the total wall time cost by the CQM.

T_{QPU}	T_{CQM} (s)	$\lvert\mathcal{J}^C\rvert$
0.016	4.311	1994
0.032	5.123	2708
0.032	5.064	1526
0.016	4.582	998
0.032	5.129	1308
0.032	5.039	1694



(a) Discretization resolution of 480×240.

(b) Discretization resolution of 480×160.

Figure 5.1: The resultant optimal material layouts with different discretization resolutions by using the Heuristic-CQM method.

5.4.2 Acceleration via the Modified DW Decomposition

It is expected that solving the local sub-problem (4.7) can be embedded onto a quantum computer following the QUBO (5.15). The classical computer only needs to solve the LP problem (4.9) to obtain the optimal solution. As indicated in Table 5.5 and table 5.6, the major part of the computational time is used for solving the sub-problems $T_{O,S}$ when compared to the time for solving the global sub-problems $T_{O,M}$, especially in the case of large-scale multi-material problems. The main overhead other than solving the global sub-problems is the time spent on forming the problem itself T_F. This mainly includes the time of evaluating the coefficients for λ based on $\rho^l_{e,m}$ in (4.9) through the iterations of the the modified DW decomposition. It requires the aggregation on the local process and the communication between different processes with respect to different λ^l_i. This portion of time can be reduced once the Python code is optimized by using the just-in-time mechanism, an optimized vectorization code or compiling the code to C++. Therefore, this section focuses on the quantum acceleration on solving the local sub-problem (4.7).

Table 5.5: Single-material minimum compliance: Time of main components of the the modified DW decomposition optimizer.

Resolution	T_F (s)	$T_{O,S}$ (s)	$T_{O,M}$ (s)	$T_{O,F}$ (s)	$T_{O,S}$ (%)	$T_{O,M}$ (%)	$T_{O,F}$ (%)
$50 \times 200 \times 50$	67.6161	29.02	32.77	2.067	42	48	3
$100 \times 400 \times 100$	163.453	90.34	48.08	19.69	55	29	12
$150 \times 600 \times 150$	200.237	137.8	23.15	34.97	68	11	17

Table 5.6: Multi-material minimum compliance: Time of main components of the the modified DW decomposition optimizer.

Resolution	T_F (s)	$T_{O,S}$ (s)	$T_{O,M}$ (s)	$T_{O,F}$ (s)	$T_{O,S}$ (%)	$T_{O,M}$ (%)	$T_{O,F}$ (%)
$50 \times 200 \times 50$	61.0540	30.28	21.81	4.483	49	35	7
$100 \times 400 \times 100$	285.215	224.7	25.46	30.53	78	8	10
$150 \times 600 \times 150$	2089.30	1626	49.40	385.6	77	2	18

First, the proposed QUBO (5.15) is going to be verified. It considers utilizing QAOA for finding the ground states based on the simulator provided by *cuda-quantum* [73]. Due to the limited number of qubits can be simulated on a classical computer, the size of $|\mathcal{D}_i|$ is set up to 8 when doing the verification. The single material case is utilized as the benchmark problem. The number of layers in the ansatz is set to be $L = 4$ or $L = 5$. Up to 30 sub-problems are randomly selected from the case of discretization resolution of $50 \times 200 \times 50$. The solutions obtained from QAOA are compared to the ones obtained from the classical MILP solver. The accuracy of the solutions are reported in Table 5.7. As shown in the table, the proposed QUBO can guide to find the desired solution when working enough number of layers in the ansatz. This implies that the proposed QUBO is valid for updating the design variables in the TO process.

Table 5.7: Solving accuracy of the randomly selected sub-problems ($n = 30$) by using different number of layers L of the ansatz and different sizes of the sub-problems $|\mathcal{D}_i|$.

$L = 4$						
$\lvert\mathcal{D}_i\rvert$	3	4	5	6	7	8
Accuracy	100%	100%	100%	98.3%	94.3%	64.2%
$L = 5$						
$\lvert\mathcal{D}_i\rvert$	3	4	5	6	7	8
Accuracy	100%	100%	100%	100%	100%	95.4%

As QAOA still requires a classical optimizer, it is not an ideal optimizer when large-scale quantum devices are available. Then, it considers the quantum annealers which can solve the optimization problem without evolvement of a classical computer during optimization.

In order to leverage the quantum annealers, the classical computer needs to form the QUBO problem (5.15) before the optimization is initi-

ated. The expected running time together with the cost spent on quantum computer is reported in Table 5.8 and 5.9, for both the single and multi-material scenario, respectively. $N_{O,S}$ denotes total number of local sub-problems involved in each TO problem; $T_{O,C}$ denotes the total time spent on forming the QUBO (5.15) accumulated from every local sub-problem evolved in the TO problem; and $T_{O,Q}$ denotes the expected time spent on the quantum annealer to find out the solution to all of the local sub-problems evolved in the TO problem with 1000 repetitions and each using 20 μs for annealing calculated as

$$T_{O,Q} = N_{O,S} \times 1000 \times 20\mu s \ .$$

For comparison, the time spent on the classical computer for solving the local sub-problem $T_{O,S}$ is also included in the tables. The expected speedup is calculated as

$$\text{Speedup} = \frac{T_{O,S}}{T_{O,C} + T_{O,Q}} \ ,$$

in which calculates the ratio between the time spent for solving the local sub-problems on a classical computer $T_{O,S}$ and the combination of $T_{O,C}$ and $T_{O,Q}$. For both scenarios, it expects a speedup in solving the local sub-problems, especially for the case with a high discretization resolution and multiple candidate materials. This implies that the quantum annealer can serve as a potential replacement to the classical computers when large enough quantum computer is accessible in the future.

Table 5.8: Single-material minimum compliance: Time of the expected quantum accelerated implementation of the the modified DW decomposition optimizer.

Resolution	$T_{O,S}$ (s)	$N_{O,S}$	$T_{O,C}$ (s)	$T_{O,Q}$ (s)	Speedup
$50 \times 200 \times 50$	29.024	1039	0.0763	20.78	1.39
$100 \times 400 \times 100$	90.338	1444	1.4938	28.88	2.97
$150 \times 600 \times 150$	137.83	699	1.7456	13.98	8.76

Table 5.9: Multi-material minimum compliance: Time of the expected quantum accelerated implementation of the the modified DW decomposition optimizer.

Resolution	$T_{O,S}$ (s)	$N_{O,S}$	$T_{O,C}$ (s)	$T_{O,Q}$ (s)	Speedup
$50 \times 200 \times 50$	30.276	659	0.2859	13.18	2.25
$100 \times 400 \times 100$	224.66	756	2.2416	15.12	12.9
$150 \times 600 \times 150$	1626.1	1400	29.703	28.00	28.2

6 SUMMARY

This thesis aims to propose a new efficient continuum TO framework to accelerate the process to find the optimal material layouts for a wide range of large-scale TO design tasks.

The main contribution of this thesis can be summarized as follows: (1) a discrete variable TO framework based on mixed-integer nonlinear programming (MINLP) is proposed, incorporating a trust-region constraint and a parameter relaxation scheme to accelerate convergence in solving a wide range of TO design problems; (2) both classical and hybrid algorithms based on DW decomposition are developed to exploit the advantages of classical parallel computing and quantum computation; (3) a geometric multi-grid (GMG) preconditioner is introduced for the meshless method based on the generalized moving least square method with moving boundaries in fluid-solid interaction problems.

Extensive numerical examples presented in the thesis demonstrate that the proposed framework can substantially reduce the computational cost associated with evaluating the governing equations that constrain TO design problems. By leveraging the proposed optimizer based on the DW decomposition and utilizing hybrid computing platforms that combine classical and quantum resources, the framework significantly accelerates the update of design variables compared to standard MILP solvers. Together with the proposed GMG preconditioner, the proposed approach shows a strong potential for accelerating the search for optimal topology in large-scale TO problems and is a promising candidate to replace conventional methods across a wide range of applications.

REFERENCES

[1] Aage, Niels, Erik Andreassen, Boyan S Lazarov, and Ole Sigmund. 2017. Giga-voxel computational morphogenesis for structural design. *Nature* 550(7674):84–86.

[2] Aage, Niels, Erik Andreassen, and Boyan Stefanov Lazarov. 2015. Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization* 51:565–572.

[3] Aage, Niels, and Boyan S Lazarov. 2013. Parallel framework for topology optimization using the method of moving asymptotes. *Structural and Multidisciplinary Optimization* 47(4):493–505.

[4] Albash, Tameem, and Daniel A Lidar. 2018. Adiabatic quantum computation. *Reviews of Modern Physics* 90(1):015002.

[5] Alvarez, Yolanda, Maria L Cederlund, David C Cottell, Brent R Bill, Stephen C Ekker, Jesus Torres-Vazquez, Brant M Weinstein, David R Hyde, Thomas S Vihtelic, and Breandan N Kennedy. 2007. Genetic determinants of hyaloid and retinal vasculature in zebrafish. *BMC Developmental Biology* 7(1):1–17.

[6] Andreassen, Erik, Anders Clausen, Mattias Schevenels, Boyan S Lazarov, and Ole Sigmund. 2011. Efficient topology optimization in MATLAB using 88 lines of code. *Structural and Multidisciplinary Optimization* 43(1):1–16.

[7] Balay, Satish, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Dmitry Karpeyev, Dinesh Kaushik, Matthew G. Knepley, Dave A. May, Lois Curfman McInnes, Richard Tran Mills,

Todd Munson, Karl Rupp, Patrick Sanan, Barry F. Smith, Stefano Zampini, Hong Zhang, and Hong Zhang. 2020. PETSc users manual. Tech. Rep. ANL-95/11 - Revision 3.13, Argonne National Laboratory.

[8] Baratta, Igor A., Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. 2023. DOLFINx: the next generation FEniCS problem solving environment. preprint.

[9] Beckers, Muriel. 1999. Topology optimization using a dual method with discrete variables. *Structural Optimization* 17:14–24.

[10] Bendsøe, Martin P, and Ole Sigmund. 1999. Material interpolation schemes in topology optimization. *Archive of Applied Mechanics* 69: 635–654.

[11] Bendsøe, Martin Philip, and Noboru Kikuchi. 1988. Generating optimal topologies in structural design using a homogenization method. *Computer Methods in Applied Mechanics and Engineering* 71(2):197–224.

[12] Bentley, Jon Louis. 1975. Multidimensional binary search trees used for associative searching. *Communications of the ACM* 18(9):509–517.

[13] Blanco, Jose Luis, and Pranjal Kumar Rai. 2014. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees.

[14] Borrvall, Thomas, and Joakim Petersson. 2003. Topology optimization of fluids in Stokes flow. *International Journal for Numerical Methods in Fluids* 41(1):77–107.

[15] Brandt, Achi. 1980. Multilevel adaptive computations in fluid dynamics. *AIAA Journal* 18(10):1165–1172.

[16] Brandt, Achi, and Oren E Livne. 2011. *Multigrid techniques: 1984 guide with applications to fluid dynamics, revised edition*. SIAM.

[17] Brenner, Susanne C.., and L Ridgway Scott. 2008. *The mathematical theory of finite element methods*, vol. 3. Springer.

[18] Bryant, Randal E, O'Hallaron David Richard, and O'Hallaron David Richard. 2003. *Computer systems: a programmer's perspective*, vol. 2. Prentice Hall Upper Saddle River.

[19] Cavazzuti, Marco, Andrea Baldini, Enrico Bertocchi, Dario Costi, Enrico Torricelli, and Patrizio Moruzzi. 2011. High performance automotive chassis design: a topology optimization based approach. *Structural and Multidisciplinary Optimization* 44:45–56.

[20] Christiansen, Rasmus E., and Ole Sigmund. 2021. Inverse design in photonics by topology optimization: tutorial. *Journal of the Optical Society of America B* 38(2):496–509.

[21] Dantzig, George B, and Philip Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* 8(1):101–111.

[22] Dede, Ercan M. 2009. Multiphysics topology optimization of heat transfer and fluid flow systems. In *proceedings of the comsol users conference*, vol. 715.

[23] Dormand, John R, and Peter J Prince. 1980. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics* 6(1):19–26.

[24] Fan, Zhe, Feng Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. 2004. Gpu cluster for high performance computing. In *Sc'04: Proceedings of the 2004 acm/ieee conference on supercomputing*, 47–47. IEEE.

[25] Farhi, Edward, Jeffrey Goldstone, and Sam Gutmann. 2014. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*.

[26] Gersborg-Hansen, Allan, Ole Sigmund, and Robert B Haber. 2005. Topology optimization of channel flow problems. *Structural and Multidisciplinary Optimization* 30(3):181–192.

[27] Grover, Lov K. 1996. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual acm symposium on theory of computing*, 212–219.

[28] Gurobi Optimization, LLC. 2022. Gurobi optimizer reference manual.

[29] Han, Kuk-Hyun, and Jong-Hwan Kim. 2000. Genetic quantum algorithm and its application to combinatorial optimization problem. In *Proceedings of the 2000 congress on evolutionary computation. cec00*, vol. 2, 1354–1360. IEEE.

[30] Harrow, Aram W, Avinatan Hassidim, and Seth Lloyd. 2009. Quantum algorithm for linear systems of equations. *Physical Review Letters* 103(15):150502.

[31] Hu, Wei, Nathaniel Trask, Xiaozhe Hu, and Wenxiao Pan. 2019. A spatially adaptive high-order meshless method for fluid–structure interactions. *Computer Methods in Applied Mechanics and Engineering* 355:67–93.

[32] Huang, Xiaodong. 2020. Smooth topological design of structures using the floating projection. *Engineering Structures* 208:110330.

[33] Huang, Xiaodong, and Weibai Li. 2021. A new multi-material topology optimization algorithm and selection of candidate materials. *Computer Methods in Applied Mechanics and Engineering* 386:114114.

[34] Huang, Xiaodong, and Mike Xie. 2010. *Evolutionary topology optimization of continuum structures: methods and applications*. John Wiley & Sons.

[35] Huang, Xiaodong, and Yi Min Xie. 2009. Bi-directional evolutionary topology optimization of continuum structures with one or multiple materials. *Computational Mechanics* 43:393–401.

[36] Jia, Yingqi, Chao Wang, and Xiaojia Shelly Zhang. 2024. FEniTop: a simple FEniCSx implementation for 2d and 3d topology optimization supporting parallel computing. *Structural and Multidisciplinary Optimization* 67(8):140.

[37] Jiang, Shuxian, Keith A Britt, Alexander J McCaskey, Travis S Humble, and Sabre Kais. 2018. Quantum annealing for prime factorization. *Scientific Reports* 8(1):17667.

[38] Karniadakis, George, and Robert M Kirby. 2003. *Parallel scientific computing in c++ and mpi: a seamless approach to parallel algorithms and their implementation*, vol. 2. Cambridge University Press.

[39] Kiziltas, Gullu, Dimitris Psychoudakis, John L Volakis, and Noboru Kikuchi. 2003. Topology design optimization of dielectric substrates for bandwidth improvement of a patch antenna. *IEEE Transactions on Antennas and Propagation* 51(10):2732–2743.

[40] Könz, Mario S., Wolfgang Lechner, Helmut G. Katzgraber, and Matthias Troyer. 2021. Embedding overhead scaling of optimization problems in quantum annealing. *PRX Quantum* 2:040322.

[41] Kreissl, Sebastian, Georg Pingen, Anton Evgrafov, and Kurt Maute. 2010. Topology optimization of flexible micro-fluidic devices. *Structural and Multidisciplinary Optimization* 42(4):495–516.

[42] Kreissl, Sebastian, Georg Pingen, and Kurt Maute. 2011. Topology optimization for unsteady flow. *International Journal for Numerical Methods in Engineering* 87(13):1229–1253.

[43] Kuberry, Paul, Peter Bosler, and Nathaniel Trask. 2019. Compadre toolkit.

[44] Kumar, Prabhat, Jan S Frouws, and Matthijs Langelaar. 2020. Topology optimization of fluidic pressure-loaded structures and compliant mechanisms using the Darcy method. *Structural and Multidisciplinary Optimization* 61(4):1637–1655.

[45] Lazarov, Boyan Stefanov, and Ole Sigmund. 2011. Filters in topology optimization based on helmholtz-type differential equations. *International Journal for Numerical Methods in Engineering* 86(6):765–781.

[46] Li, Ruipeng, and Yousef Saad. 2013. Gpu-accelerated preconditioned iterative linear solvers. *The Journal of Supercomputing* 63:443–466.

[47] Liang, Yuan, and Gengdong Cheng. 2019. Topology optimization via sequential integer programming and canonical relaxation algorithm. *Computer Methods in Applied Mechanics and Engineering* 348:64–96.

[48] Liu, Haixiang, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-band topology optimization on a sparsely populated grid. *ACM Transactions on Graphics (TOG)* 37(6):1–14.

[49] Liu, Hongliang, Cheng Wang, Yewei Zhang, and Yuan Liang. 2024. Multi-material structural discrete variable topology optimization with minimum length scale control under mass constraint. *Computer Methods in Applied Mechanics and Engineering* 420:116701.

[50] Lundstrom, Mark. 2003. Moore's law forever? *Science* 299(5604): 210–211.

[51] Molesky, Sean, Zin Lin, Alexander Y. Piggott, Weiliang Jin, Jelena Vucković, and Alejandro W. Rodriguez. 2018. Inverse design in nanophotonics. *Nature Photonics* 12(11):659–670.

[52] Moscatelli, Eduardo, LuÃs Fernando Nogueira de Sá, Shahin Ranjbarzadeh, Renato Picelli, Rafael dos Santos Gioria, and EmÃlio Carlos Nelli Silva. 2022. Hybrid geometry trimming algorithm based on integer linear programming for fluid flow topology optimization. *Computers & Fluids* 244:105561.

[53] Muñoz, Eduardo, and Mathias Stolpe. 2011. Generalized Benders' decomposition for topology optimization problems. *Journal of Global Optimization* 51(1):149–183.

[54] Nocedal, Jorge, and Stephen J Wright. 1999. *Numerical optimization*. Springer.

[55] Nomura, Tsuyoshi, Kazuo Sato, Kenji Taguchi, Tatsuya Kashiwa, and Shinji Nishiwaki. 2007. Structural topology optimization for the design of broadband dielectric resonator antennas using the finite difference time domain technique. *International Journal for Numerical Methods in Engineering* 71(11):1261–1296.

[56] Persson, Per-Olof, and Gilbert Strang. 2004. A simple mesh generator in matlab. *SIAM Review* 46(2):329–345.

[57] Picelli, Renato, Raghavendra Sivapuram, and Yi Min Xie. 2021. A 101-line MATLAB code for topology optimization using binary variables and integer programming. *Structural and Multidisciplinary Optimization* 63(2):935–954.

[58] Querin, Osvaldo M, Grant P Steven, and Yi Min Xie. 1998. Evolutionary structural optimisation (ESO) using a bidirectional algorithm. *Engineering Computations* 15(8):1031–1048.

[59] Renegar, James. 1995. Linear programming, complexity theory and elementary functional analysis. *Mathematical Programming* 70(1): 279–351.

[60] Reuss, A. 1929. Berechnung der fließgrenze von mischkristallen auf grund der plastizitätsbedingung für einkristalle . *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik* 9(1):49–58.

[61] Sato, Yuki, Ruho Kondo, Satoshi Koide, and Seiji Kajita. 2023. Quantum topology optimization of ground structures for near-term devices. In *2023 ieee international conference on quantum computing and engineering (qce)*, vol. 1, 168–176. IEEE.

[62] Schaller, Robert R. 1997. Moore's law: past, present and future. *IEEE Spectrum* 34(6):52–59.

[63] Schöberl, Joachim. 1998. Robust multigrid preconditioning for parameter-dependent problems I: The Stokes-type case. In *Multigrid methods v*, 260–275. Springer.

[64] Shampine, Lawrence F, and Mark W Reichelt. 1997. The MATLAB ODE suite. *SIAM Journal on Scientific Computing* 18(1):1–22.

[65] Sigmund, Ole. 2007. Morphology-based black and white filters for topology optimization. *Structural and Multidisciplinary Optimization* 33(4):401–424.

[66] Steane, Andrew. 1998. Quantum computing. *Reports on Progress in Physics* 61(2):117.

[67] Sun, Kai, Yuan Liang, and Gengdong Cheng. 2022. Sensitivity analysis of discrete variable topology optimization. *Structural and Multidisciplinary Optimization* 65(8):1–18.

[68] Sun, Sicheng, Piotr Liebersbach, and Xiaoping Qian. 2020. 3D topology optimization of heat sinks for liquid cooling. *Applied Thermal Engineering* 178:115540.

[69] Svanberg, Krister. 1987. The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering* 24(2):359–373.

[70] Świrydowicz, Kasia, Eric Darve, Wesley Jones, Jonathan Maack, Shaked Regev, Michael A Saunders, Stephen J Thomas, and Slaven Peleš. 2022. Linear solvers for power grid optimization problems: a review of gpu-accelerated linear solvers. *Parallel Computing* 111: 102870.

[71] Takezawa, Akihiro, Shinji Nishiwaki, and Mitsuru Kitamura. 2010. Shape and topology optimization based on the phase field method and sensitivity analysis. *Journal of Computational Physics* 229(7):2697–2718.

[72] Zoltan2 Project Team. 2020. *The Zoltan2 Project Website*.

[73] development team, The CUDA-Q. 2025. Cuda-q.

[74] Träff, Erik A, Anton Rydahl, Sven Karlsson, Ole Sigmund, and Niels Aage. 2023. Simple and efficient gpu accelerated topology optimisation: Codes and applications. *Computer Methods in Applied Mechanics and Engineering* 410:116043.

[75] Trask, Nathaniel, Martin Maxey, and Xiaozhe Hu. 2018. A compatible high-order meshless method for the Stokes equations with applications to suspension flows. *Journal of Computational Physics* 355: 310–326.

[76] Trask, Nathaniel, Mauro Perego, and Pavel Bochev. 2017. A high-order staggered meshless method for elliptic problems. *SIAM Journal on Scientific Computing* 39(2):A479–A502.

[77] Wang, Michael Yu, Xiaoming Wang, and Dongming Guo. 2003. A level set method for structural topology optimization. *Computer Methods in Applied Mechanics and Engineering* 192(1-2):227–246.

[78] Wang, Xiaojun, Zhenghuan Wang, and Bowen Ni. 2024. Mapping structural topology optimization problems to quantum annealing. *Structural and Multidisciplinary Optimization* 67(5):74.

[79] Wang, Yan, Jungin E Kim, and Krishnan Suresh. 2023. Opportunities and challenges of quantum computing for engineering optimization. *Journal of Computing and Information Science in Engineering* 23(6): 060817.

[80] Wendland, Holger. 2004. *Scattered data approximation*, vol. 17. Cambridge University Press.

[81] Wils, Kevin, and Boyang Chen. 2023. A symbolic approach to discrete structural optimization using quantum annealing. *Mathematics* 11(16):3451.

[82] Wu, Jun, Ole Sigmund, and Jeroen P Groen. 2021. Topology optimization of multi-scale structures: a review. *Structural and Multidisciplinary Optimization* 63:1455–1480.

[83] Xu, Jinchao, and Zhimin Zhang. 2004. Analysis of recovery type a posteriori error estimators for mildly structured grids. *Mathematics of Computation* 73(247):1139–1152.

[84] Ye, Zisheng, Xiaoping Qian, and Wenxiao Pan. 2023. Quantum topology optimization via quantum annealing. *IEEE Transactions on Quantum Engineering* 4.

[85] Yoo, Andy B, Morris A Jette, and Mark Grondona. 2003. Slurm: Simple linux utility for resource management. In *Workshop on job scheduling strategies for parallel processing*, 44–60. Springer.

[86] Yuan, Yaxiang. 2015. Recent advances in trust region algorithms. *Mathematical Programming* 151:249–281.

[87] Zhu, Ji-Hong, Wei-Hong Zhang, and Liang Xia. 2016. Topology optimization in aircraft and aerospace structures design. *Archives of Computational Methods in Engineering* 23:595–622.

[88] Zuo, Wenjie, and Kazuhiro Saitou. 2017. Multi-material topology optimization using ordered simp interpolation. *Structural and Multidisciplinary Optimization* 55:477–491.