First-Order Topology Optimization via Inexact Finite Element Analysis

Zherong Pan[a], Xifeng Gao[a], and Kui Wu[a]

[a]*Lightspeed Studios, Tencent-America, 10900 Northeast 8th Street, Bellevue, 98004, WA, USA*

Abstract

Topology Optimization (TO) is an essential tool for optimizing the structural robustness of load-bearing mechanical parts. An ideal TO solver should be computationally efficient for designers to preview the results, while ultimately converge to locally optimal designs. However, existing TO solvers either incur a high iterative cost or fail to provide the convergence guarantee. Borrowing ideas from recent advances in first-order bilevel optimization, we propose a new TO solver combining the Projected Gradient Descent (PGD) algorithm and inexact Finite Element Analysis (FEA). We further show that our method is convergent to a first-order critical point. Our proposed First-Order Bilevel Topology Optimization (FBTO) can solve several, important problems in the robot design paradigm, including TO under self-weight and multiple external loads. Finally, we evaluate and compare FBTO with prior TO solvers on a row of 2D and 3D problems.

Keywords: Topology Optimization, Bilevel Optimization, Low-Cost Robotics, Mechanisms and Design

1. Introduction

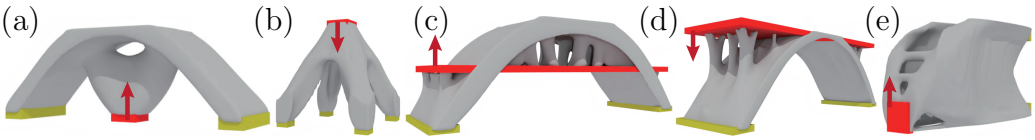

Figure 1: 3D mechanical parts computed via FBTO Algorithm 3. The red and yellow blocks are external loads (arrows) and fixed regions.

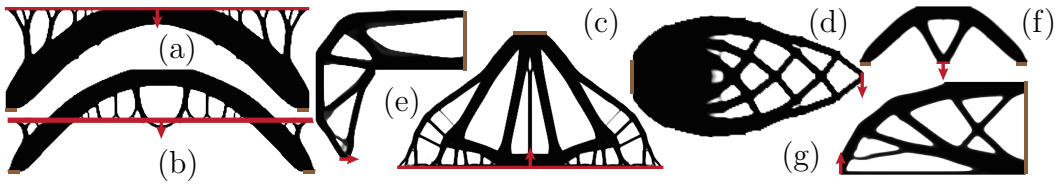

Figure 2: 2D mechanical parts computed via FBTO Algorithm 3. These problems are selected from [1, 2] and 2D version of [3].

To determine the shape of mechanical parts, designers need to compromise between several conflicting factors, such as total weight, manufacturing cost, structural strength, and mobility. These factors are strongly correlated in an obscure manner, and even experienced designers cannot figure out the optimal design without massive trial and error. Topology Optimization (TO) [4] can largely alleviate designers' burden. TO algorithms fine tune each mechanical part's center-of-mass [5], inertia [6], infill pattern [7], or material distribution [8] under hard constraints such as total weight and material cost. In this paper, we focus on a subclass of TO problems assuming stiff materials undergoing infinitesimal deformation due to external loads, which is the most widely adopted setting in the mechanical design paradigm.

Prior works on automatic mechancial design rely on Finite Element Analysis (FEA) to predict the relationship between the extrinsic (boundary loads) and intrinsic status (stress, strain, and energy densities) of a mechanical part, from which a TO solver iteratively updates the design. However, we argue that these off-the-shelf algorithms are less suitable in the mechanical design

paradigm for two reasons. First, FEM predictions are made by solving gigantic linear systems, which incur a high iterative cost. For example, an 64^3 design space illustrated in Figure 1 incurs 10^5 decision variables. As a result, a designer needs to wait for hours [9, 10, 11] before a design is optimized, significantly slowing down the overall design refinement loop. Second, some fast TO solvers [12, 2, 13] rely on heuristic step size choices, which cannot guarantee convergence to a locally optimize design.

To resolve the two above-mentioned issues, we propose a new method for solving the subclass of TO problems known as Solid Isotropic Material with Penalization (SIMP) [12] that searches for fine-grained mechanical part design by optimizing the infill levels. We borrow ideas from recent advances in first-order bilevel optimization [14] and revisit SIMP as a bilevel optimization with a strictly convex, quadratic low-level objective function (Section 3). For these bilevel problems, we propose a new class of solvers, namely First-Order Bilevel Topology Optimization (FBTO). FBTO differs from all prior methods by only requiring inexact FEA during each iteration. Through theoretical analysis and a set of 2D and 3D computational benchmarks in Figure 1,2, we show that FBTO exhibits the following desirable properties, making it a stellar fit for mechanical design problems:

- By using inexact FEA, FBTO incurs a much lower iterative cost of $\mathcal{O}(E \log E)$ on CPU and $\mathcal{O}(\log E)$ on GPU (Section 4.3, E is the number of decision variables), as compared with superlinear iterative cost of conventional TO solvers. This feature allows designers to quickly preview their designs and perform design refinements when necessary.

- Our theoretical analysis shows that FBTO is guaranteed to converge to

the first-order critical point of SIMP problems. Our result also explains the convergence behavior of off-the-shelf TO solvers [7, 13] that use iterative algorithms to approximately perform FEA.

- Besides standard problem settings, FBTO can solve importance variants of SIMP problems, including TO under self-weights and multiple external loads. These features are essential for modeling mechanical parts built with heavy compound metals or working under various conditions.

2. Related Work

We review representative TO formulations in various applications, numerical solvers for TO problems, and bilevel optimizations from which our method is derived.

TO Formulations & Applications: We deal with the SIMP model [12] where the geometric shape is discretized using FEA and decision variables are infill levels. The same formulation has been used in [15] to design the five-bar mechanism and in [16] to design humanoid arms. In [16], the humanoid arm is considered under multiple load conditions. Jain and Saxena [17] considered TO with self-weight. Our new method can solve TO in all the above-mentioned cases. On the other hand, SIMP model is based on the linear elasticity theory that is accurate only under small deformations. To model large deformations, we need to use nonlinear elasticity theory [18, 19]. In addition, standard SIMP model does not have hard constraints to limit deformations, while alternative formulations include strain/stress constraints [20], e.g., Sha et al. [21] used stress-constrained formulation to design lightweight

robots. Unfortunately, our method does not apply to these extensions.

Numerical TO Solvers: Existing mechanical design tools rely on two kinds of numerical approaches: truncated gradient method and general-purpose method. The reference implementation [12] of the SIMP solver uses the heuristic, truncated gradient method [22]. Amir et al., Amir [23, 24] proposed to combine the SIMP solver with inexact FEA analysis, which shares the main idea with our approach. These methods have good empirical performance but do not have theoretical convergence guarantee. Recently, efficient, inexact FEA system solvers such as the multigrid method [2, 25] have been used with the truncated gradient method to achieve the best performance so far, which further complicates the convergence analysis. General-purpose methods are off-the-shelf optimization algorithms, e.g., sequential quadratic programming (SQP)/interior point method [26], augmented Lagrangian method (ALM) [27], and method of moving asymptotes [28]). These methods have second order convergence guarantee but incur a superlinear iterative cost by solving large FEA systems. In particular, Choi et al. [29] proposed to use reduced-order modeling to acceleration the solution of linear systems within each iteration of the interior point method, but their overall iterative cost and convergence properties are not theoretically investigated.

Bilevel Optimization [30]: When a constraint of an optimization problem (high-level problem) involves another optimization (low-level problem), the optimization is considered bilevel. In our formulation, the high-level problem determines the material infill levels, while the low-level problem performs the FEA analysis. Bilevel formulation has been used in TO community for years, which is also known as Nested Analysis and Design (NAND) [31]. How-

ever, a NAND solver requires finding the (near) exact solution of low-level problem, leading to a several performance penalty. More prominent methods rely on Simultaneous Analysis and Design (SAND), which use general-purpose algorithms (see e.g. [32, 33]) to satisfy the first-order optimality conditions of the underlying optimization problem, thus allowing inexact solutions of the FEM system. However, these SAND solvers involve a Newton's step during each iteration, which incurs a high iterative solve. Empirical acceleration techniques have been proposed [23, 24, 29] to use inexact Newton's step, but their overall iterative cost and convergence properties are not theoretically investigated. Recently, it has been shown that first-order bilevel optimization is convergent [14, 34, 35]. We show that, when applied to TO problems, first-order solvers are convergent and have a much lower iterative cost of $\mathcal{O}(E \log E)$ on CPU and $\mathcal{O}(\log E)$ on GPU as compared with prior TO solvers. In summary, our approach is built off of the NAND formulation, while pertains the computational advantage of SAND, i.e., allowing inexact solvers for the low-level PDE.

3. Problem Formulation

In this section we introduce the SIMP model and its extensions to self-weight and multiple load conditions, using notations in [2]. (see Table 3 in our appendix for a list of symbols)

3.1. The SIMP Model

SIMP uses FEA to discretize a mechanical part governed by the linear elastic constitutive law. If the material is undergoing an internal displacement $u(x) : \Omega \to \mathbb{R}^2$ where Ω is the material domain, then the internal

potential energy is accumulated according to the constitutive law as follows:

$$P[u(\bullet), v(\bullet)] = \int_\Omega \frac{1}{2} u(x)^T k_e(v(x), x) u(x) dx,$$

where $P[\bullet, \bullet]$ is the potential energy functional, k_e is the spatially varying stiffness tensor parameterized by infinite-dimensional decision variable $v(x) : \Omega \to \mathbb{R}$ that determines the infill levels. FEA discretizes the material domain Ω into a set of E elements each spanning the sub-domain Ω_e. The elements are connected by a set of N nodes. By restricting $P[\bullet]$ to a certain finite-dimensional functional space (the shape space), the potential energy can be written as the following sum over elements:

$$\begin{aligned} P(u, v) =& \frac{1}{2} u^T \sum_{e=1}^{K} K_e(v) u \triangleq \frac{1}{2} u^T K(v) u \\ K_e(v) \triangleq& \int_{\Omega_e} k_e(v, x) dx. \end{aligned}$$

Here K_e is the stiffness matrix of the eth element, K is the stiffness matrix assembled from K_e. With a slight abuse of notation, we denote $P(\bullet, \bullet)$ as a finite-dimensional potential energy function, $u \in \mathbb{R}^{2N}$ without bracket as a vector of 2D nodal displacements ($u \in \mathbb{R}^{3N}$ in 3D), and $v \in \mathbb{R}^E$ without bracket as a vector of elementwise infill levels. The SIMP model is compatible with all kinds of FEA discretization. The case with regular grid is illustrated in Figure 3.

Suppose an external force f is exerted on the mechanical part, then the total (internal+external) potential energy is:

$$P_f(u, v) = \frac{1}{2} u^T K(v) u - f^T u.$$

SIMP model works with arbitrary force setup, but external forces are only applied to the boundary in practice. For example, if only the ith boundary

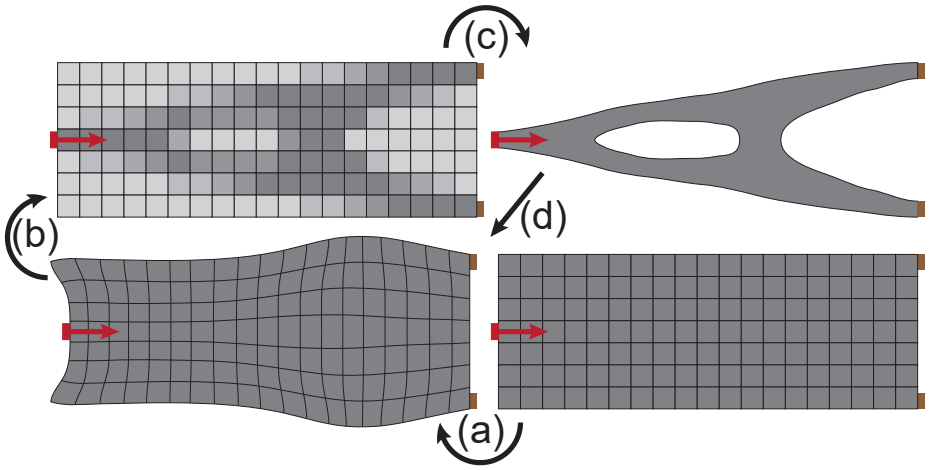

Figure 3: Key steps of the SIMP model and the design loop: define the SIMP problem: external load (red) and fixture (brown); compute displacement u using FEA (a); perform sensitivity analysis to compute ∇l (b); update infill levels (c); repeat (b,c,d) until convergence.

node is under a force f_i, then we have $f = e_i f_i$ with e_i being the unit vector. The equilibrium state is derived by minimizing P_f with respect to u, giving $u_f = K^{-1}(v)f$. As the name suggests, the two variables u_f, f are linearly related under the linear elastic constitutive law. The goal of TO is to minimize the induced internal potential energy due to f:

$$\underset{v\in X}{\operatorname{argmin}}\, l(v) \triangleq \frac{1}{2} u_f^T(v) K(v) u_f(v) = \frac{1}{2} f^T K^{-1}(v) f, \tag{1}$$

where X is a compact, convex set bounding v away from potentially singular configurations. Several widely used stiffness matrix parameterization of $K(v)$ includes the linear law $K(v) = \sum_{e=1}^{E} v_e K_e$ or the power law $K(v) = \sum_{e=1}^{E} v_e^\eta K_e$ where $\eta > 0$ is some constant. In both cases, we need to choose X such that the spectrum of K is bounded from both sides:

$$\mathbb{0} < \underline{\rho}\mathbb{1} \le \rho(K(v)) \le \bar{\rho}\mathbb{1}, \tag{2}$$

for any $v \in X$, where $\rho(\bullet)$ is the eigenvalues of a matrix. A typical choice of X to this end is:

$$X = \{v | \mathbb{1} \geq v \geq \underline{v}\mathbb{1} > \mathbb{0} \wedge \mathbb{1}^T v \leq \bar{v}\},$$

where v_e is the infill level of the eth element, $\underline{v}$ is the minimal elementwise infill level, $\bar{v}$ is the total infill level, and $\mathbb{1}, \mathbb{0}$ are the all-one and all-zero vectors, respectively. In this work, we use the following more general assumption on the stiffness matrix parameterization and the shape of X:

Assumption 3.1. *X is a compact polytopic subset of $\mathbb{R}^E$, on which Equation 2 holds and $K(v)$ is smooth.*

Remark 3.2. *The compactness of X is obvious. X is also polytopic because we only have linear constraints, which is essential when we measure optimality using relative projection error (see Section 4 for details). Assumption 3.1 is more general than the standard SIMP model. Indeed, $K(v)$ can be parameterized using any smooth activation function $\mathcal{A}_e(v)$ for the eth element as $K(v) = \sum_{e=1}^{E} \mathcal{A}_e(v) K_e$, which obviously satisfies Assumption 3.1 as long as $\mathcal{A}_e(v)$ is bounded from both above and below on X, away from zero. Our assumption also allows a variety of topology modifications and constraints, some of which are illustrated here.* ***Symmetry-Constraint:*** *We can plug in a left-right symmetric mapping matrix $\mathbb{S}$ and define $K(v) = K(\mathbb{S}v_s)$ where v_s is the infill levels for the left-half of the material block.* ***Component-Cost-Constraint:*** *A mechanical part can be divided into different sub-components and the amount of material can be assigned for each component as follows:*

$$X \triangleq \{v | v = \left(v^{1^T}, v^{2^T}, \ldots^T, v^{\#^T}\right)^T \wedge \mathbb{1}^T v^i \leq \bar{v}^i\},$$

where $\#$ is the number of sub-components, v^i is the sub-vector of v consisting of decision variables for the ith component, and $\bar{v}^i$ is the total allowed amount of material for that sub-component. ***Material-Filtering:*** *Guest et al. [36] proposed to control the thickness of materials by filtering the infill levels using some convolutional kernel denoted as $\mathbb{C}$. As long as the convolution operator $\mathbb{C}(\bullet)$ is smooth, Assumption 3.1 holds by plugging in $K(v) = \sum_{e=1}^{E} \mathcal{A}_e(\mathbb{C}(v)) K_e$.*

3.2. Self-Weight and Multiple Load Conditions

The standard SIMP model can be used to find mechanical designs under a single external load f without any internal load. In practice, however, a mechanical part must work under various load conditions, which depend on its kinematic configuration. Furthermore, mechanical parts are typically built from compound metals with non-trivial self-weight. We can extend the SIMP model to consider these factors. To consider self-weight, we assume that f depends on the infill levels v, denoted as a function $f(v)$. A typical choice of $f(v)$ is as follows:

$$f(v) = \int_{\Omega} v(x)\rho g^T u(x) dx, \tag{3}$$

where ρ is the material density and g is the gravitational coefficient. The function $f(v)$ can be discretized using the same FEA scheme as that for the internal energy. We further consider a row of Q external load conditions but minimizing the sum of potential energy over each case. We use subscript to denote the external load case. Putting things together, we have the following extended objective function:

$$l(v) \triangleq \frac{1}{2} \sum_{q=1}^{Q} f^q(v) K^{-1}(v) f^q(v).$$

Accordingly, we define u_f^q as the displacement corresponding to the qth external load.

3.3. TO Solvers for the SIMP Model

Existing methods such as the (GC)MMA [37, 28], SQP [33, 26], and [2] for solving Equation 1 involve computing the exact gradient via sensitivity analysis:

$$\nabla l = \frac{1}{2} \sum_{q=1}^{Q} (u_f^q)^T \left[\frac{\partial f^q}{\partial v} - \frac{\partial K}{\partial v} u_f^q \right],$$

where we have used the derivatives of the matrix inverse. The major bottleneck lies in the computation of u_f^q that involves exact matrix inversion. Here we assume that $\frac{\partial^2 K}{\partial v^2}$ is a third-order tensor of size $|u| \times |u| \times |v|$ and its left- or right-multiplication means contraction along the first and second dimension. Note that the convergence of GCMMA and SQP solvers rely on the line-search scheme that involves the computation of objective function values, which in turn requires matrix inversion. Practical methods [2] and [12] avoid the line-search schemes using heuristic step sizes. Although working well in practice, the theoretical convergence of these heuristic rules is difficult to establish.

4. Bilevel Topology Optimization

In this section, we first propose our core formulation, then discuss extensions to accelerate convergence via preconditioning (Section 4.1) and fast projection operators (Section 4.2), and finally present GPU parallel implementations (Section 4.3). Inspired by the recent advent of first-order bilevel optimization algorithms [14, 34, 35], we investigate the reformulation of Equation 1 as the following bilevel optimization:

$$\begin{aligned} \underset{v \in X}{\operatorname{argmin}}\ f(u_f^q, v) &\triangleq \frac{1}{2} \sum_{q=1}^{Q} (u_f^q)^T K(v) u_f^q \\ \text{s.t.} \quad u_f^q &\in \underset{u}{\operatorname{argmin}}\ \frac{1}{2} u^T K(v) u - f^q(v)^T u, \end{aligned} \tag{4}$$

which is an well-known problem of NAND [31]. The low-level part of Equation 4 is a least-square problem in u and, since $K(v)$ is always positive definite when $v \in X$, the low-level solution is unique. Plugging the low-level solution into the high-level objective function and Equation 1 is recovered. The first-order bilevel optimization solves Equation 4 by time-integrating the discrete dynamics system as described in Algorithm 1 and we denote this algorithm as First-Order Bilevel Topology Optimization (FBTO). Line 3 of Algorithm 1 is a single damped Jacobi iteration for refining the low-level solution using an adaptive step size of β_k. But instead of performing multiple Jacobi iterations until convergence, we go ahead to use the inexact result after one iteration for sensitivity analysis. Finally, we use a projected gradient descend to update the infill levels with an adaptive step size of α_k. Here $\text{Proj}_X(\bullet)$ is the projection onto the convex set X under Euclidean distance.

Algorithm 1 FBTO

1: **for** k$\leftarrow 1, 2, \cdots$ **do**
2: **for** q$\leftarrow 1, 2, \cdots, Q$ **do**
3: $u_{k+1}^q \leftarrow u_k^q - \beta_k(K(v_k)u_k^q - f^q(v_k))$
4: $v_{k+1} \leftarrow \mathrm{Proj}_X\left(v_k + \frac{1}{2}\sum_{q=1}^Q (u_k^q)^T\left[\frac{\partial K}{\partial v_k}u_k^q - \frac{\partial f^q}{\partial v_k}\right]\right)$

The approximate low-level solve has a linear iterative cost of $\mathcal{O}(E)$ as compared with conventional methods that have a superlinear iterative cost due to sparse matrix inversions.

Remark 4.1. *The succinct form of our approximate gradient ($\sum_{q=1}^Q (u_k^q)^T(\frac{\partial f^q}{\partial v_k} - \frac{\partial K}{\partial v_k}u_k^q)/2$ in Line 4) makes use of the special structure of the SIMP model, which is not possible for more general bilevel problems. If a general-purpose first-order bilevel solver is used, e.g. [14] and follow-up works, one would first compute/store $\nabla_{uv}P_f$ and then approximate $\nabla_{uu}P_f^{-1}\nabla_{uv}P_f$ via sampling. Although the stochastic approximation scheme does not require exact matrix inversion, they induce an increasing number of samples with a larger number of iterations. Instead, we make use of the cancellation between $K(v)$ in the high-level objective and $K^{-1}(v)$ in the low-level objective to avoid matrix inversion or its approximations, which allows our algorithm to scale to high dimensions without increasing sample complexity.*

As our main result, we show that Algorithm 1 is convergent under certain choices of parameters in Section 7 where we further show that the high-level optimality error scales as $\mathcal{O}(k^{m-1})$, so that m should be as small as possible for the best convergence rate. Our analysis requires $m > 3/4$, so the convergence rate can be arbitrarily close to $\mathcal{O}(k^{-1/4})$ as measured by the following relative projection error [38]:

$$\Delta_k^v \triangleq \frac{1}{\alpha_k^2}\|\mathrm{Proj}_X\left(v_k - \alpha_k\nabla l(v_k)\right) - v_k\|^2.$$

Although our analysis uses a similar technique as that for the two-timescale method [34], our result is only single-timescale. Indeed, the low-level step size β_k can be constant and users only need to tune a decaying step size for α_k. We will further show that certain versions of our algorithm allow a large choice of $\beta_k = 1$ (see Section 8.5). In addition, our formula for choosing α_k does not rely on the maximal number of iterations of the algorithm.

Algorithm 2 PFBTO

for k$\leftarrow 1, 2, \cdots$ **do**
  for q$\leftarrow 1, 2, \cdots, Q$ **do**
    $\delta_k^q \leftarrow M^{-1}(v_k)(K(v_k)u_k^q - f^q(v_k))$
    $u_{k+1}^q \leftarrow u_k^q - \beta_k K(v_k)M^{-1}(v_k)\delta_k^q$
  $v_{k+1} \leftarrow \text{Proj}_X\left(v_k + \frac{1}{2}\sum_{q=1}^{Q}(u_k^q)^T\left[\frac{\partial K}{\partial v_k}u_k^q - \frac{\partial f^q}{\partial v_k}\right]\right)$

Algorithm 3 CPFBTO

for k$\leftarrow 1, 2, \cdots$ **do**
  for q$\leftarrow 1, 2, \cdots, Q$ **do**
    $u_{k+1}^q \leftarrow u_k^q - \beta_k M^{-1}(v_k)(K(v_k)u_k^q - f^q(v_k))$
  $v_{k+1} \leftarrow \text{Proj}_X\left(v_k + \frac{1}{2}\sum_{q=1}^{Q}(u_k^q)^T\left[\frac{\partial K}{\partial v_k}u_k^q - \frac{\partial f^q}{\partial v_k}\right]\right)$

4.1. Preconditioning

Our Algorithm 1 uses steepest gradient descend to solve the linear system in the low-level problem, which is known to have a slow convergence rate of $(1 - 1/\kappa)^{2k}/(1 + 1/\kappa)^{2k}$ [39] with κ being the condition number of the linear system matrix. A well-developed technique to boost the convergence rate is preconditioning [40], i.e., pre-multiplying a symmetric positive-definite matrix $M(v)$ that approximates $K(v)$ whose inverse can be computed at a low-cost. Preconditioning is a widely used technique in the TO community [2, 13] to accelerate the convergence of iterative linear solvers in solving $u_f^q(v) = K(v)^{-1}f^q(v)$. In this section, we show that FBTO can be extended to this setting by pre-multiplying $M(v)$ in the low-level problem and we name Algorithm 2 as Pre-conditioned FBTO or PFBTO. Algorithm 2 is solving the following different bilevel program from Equation 4:

$$\begin{aligned} \underset{v\in X}{\text{argmin}}\ f(u_f^q, v) &\triangleq \frac{1}{2}\sum_{q=1}^{Q}(u_f^q)^T K(v)u_f^q \\ \text{s.t.}\quad u_f^q &\in \underset{u}{\text{argmin}}\ \|K(v)u - f^q(v)\|^2, \end{aligned} \tag{5}$$

where the only difference is the low-level system matrix being squared to $K^2(v)$. Since $K(v)$ is non-singular, the two problems have the same solution set. Equation 5 allows us to multiply $M^{-1}(v)$ twice in Algorithm 2 to get the symmetric form of $K(v)M^{-2}(v)K(v)$. To show the convergence of Algo-

rithm 2, we only need the additional assumption on the uniform boundedness of the spectrum of $M(v)$:

Assumption 4.2. *For any $v \in X$, we have:*

$$\mathbb{0} < \underline{\rho}_M \mathbb{1} \leq \rho(M(v)) \leq \bar{\rho}_M \mathbb{1}.$$

Assumption 4.2 holds for all the symmetric-definite preconditioners. The convergence can then be proved following the same reasoning as that of Algorithm 1 with a minor modification summarized in Section 7.5. Our analysis sheds light on the convergent behavior of prior works using inexact FEA system solvers such as geometric multigrid [2, 13] and conjugate gradient method [41]. We further extend these prior works by enabling convergent algorithms using a single iteration of a lightweight preconditioners, including Jacobi/Gauss-Seidel iterations and approximate inverse schemes, to name just a few. The practical performance of Algorithm 2 would highly depend on the design and implementation of specific preconditioners.

On the down side of Algorithm 2, the system matrix in the low-level problem of Equation 5 is squared and so is the condition number. Since the convergence speed of low-level problem is $(1-1/\kappa)^{2k}/(1+1/\kappa)^{2k}$, squaring the system matrix can significantly slow down the convergence, counteracting the acceleration brought by preconditioning. This is because we need to derive a symmetric operator of form $K(v)M^{-2}(v)K(v)$. As an important special case, however, only a single application of $M^{-1}(v)$ suffice if $M(v)$ commutes with $K(v)$, leading to the Commutable PFBTO or CPFBTO Algorithm 3. A useful commuting preconditioner is the Arnoldi process used by the GMRES solver [42], which is defined as:

$$\begin{aligned} M^{-1}(v)b &\triangleq \sum_{i=0}^{D} c_i^* K^i(v)b \\ c_i^* &\triangleq \underset{c_i}{\operatorname{argmin}} \|b - \sum_{i=1}^{D+1} c_{i-1} K^i(v)b\|^2, \end{aligned} \tag{6}$$

where D is the size of Krylov subspace. By sharing the same eigenvectors, $M(v)$ is clearly commuting with $K(v)$. This is a standard technique used as the inner loop of the GMRES solver, where the least square problem Equation 6 is solved via the Arnoldi iteration. The Arnoldi process can be efficiently updated through iterations if $K(v)$ is fixed, which is not the case with our problem. Therefore, we choose the more numerically stable

Householder QR factorization to solve the least square problem, and we name Equation 6 as Krylov-preconditioner. To show the convergence of Algorithm 3, we use a slightly different analysis in Section 7.5, which allows the choice of a large, constant step size $\beta_k = 1$ when the Krylov-preconditioner is used.

4.2. Efficient Implementation

We present some practical strategies to further accelerate the computational efficacy of all three Algorithm 1,2,3 that are compatible with our theoretical analysis. First, the total volume constraint is oftentimes active as noted in [37]. Therefore, it is useful to maintain the total volume constraint when computing the approximate gradient. If we define the approximate gradient as $\tilde{\nabla} l \triangleq \sum_{q=1}^{Q} (u_k^q)^T (\frac{\partial f^q}{\partial v_k} - \frac{\partial K}{\partial v_k} u_k^q)/2$ and consider the total volume constraint $\mathbb{1}^T v_k = \bar{v}$, then a projected gradient can be computed by solving:

$$\tilde{\nabla}_P l \triangleq \underset{\mathbb{1}^T \tilde{\nabla}_P l = 0}{\operatorname{argmin}} \|\tilde{\nabla}_P l - \tilde{\nabla} l\|^2,$$

with the analytic solution being $\tilde{\nabla}_P l = \tilde{\nabla} l - \mathbb{1}\mathbb{1}^T \tilde{\nabla} l / E$. In our experiments, using $\tilde{\nabla}_P l$ in place of $\tilde{\nabla} l$ in the last line of FBTO algorithms can boost the convergence rate at an early stage of optimization. On the other hand, if the volume constraint is detected to be inactive, then we could use the original gradient without hindering the convergence of the overall algorithm.

Second, the implementation of the projection operator $\text{Proj}_X(\bullet)$ can be costly in high-dimensional cases. However, our convex set X typically takes a special form that consists of mostly bound constraints $\mathbb{1} \geq v \geq \underline{v}\mathbb{1}$ with only one summation constraint $\mathbb{1}^T v \leq \bar{v}$. Such a special convex set is known as a simplex and a special $\mathcal{O}(E \log(E))$ algorithm exists for implementing the $\text{Proj}_X(\bullet)$ as proposed in [43]. Although the original algorithm [43] only considers the equality constraint $\mathbb{1}^T v = 1$ and single-sided bounds $v \geq 0$, a similar technique can be adopted to handle our two-sided bounds and inequality summation constraint and we provide its derivation for completeness. We begin by checking whether the equality constraint is active. We can immediately return if $\mathbb{1}^T \text{Proj}_{\{\mathbb{1} \geq v \geq \underline{v}\}}(v_k) \leq \bar{v}$. Otherwise, the projection operator amounts to solving the following quadratic program:

$$\underset{\mathbb{1} \geq v \geq \underline{v}\mathbb{1}}{\operatorname{argmin}} \frac{1}{2}\|v - v_k\|^2 \quad \text{s.t. } \mathbb{1}^T v = \bar{v},$$

whose Lagrangian $\mathcal{L}$ and first-order optimality conditions are:

$$\begin{aligned}\mathcal{L} =& \frac{1}{2}\|v - v_k\|^2 + \lambda_{\mathbb{1}}^T(\mathbb{1} - v) + \lambda_{\underline{v}}^T(v - \underline{v}\mathbb{1}) + \lambda(\mathbb{1}^T v - \bar{v}) \\ v =& v_k + \lambda_{\mathbb{1}} - \lambda_{\underline{v}} - \lambda\mathbb{1} \wedge \mathbb{0} \le \lambda_{\mathbb{1}} \perp \mathbb{1} - v \ge \mathbb{0} \wedge \mathbb{0} \le \lambda_{\underline{v}} \perp v - \underline{v}\mathbb{1} \ge \mathbb{0},\end{aligned}$$

where $\lambda_{\mathbb{1}}, \lambda_{\underline{v}}, \lambda$ are Lagrange multipliers. We can conclude that $v = \text{Clamp}(v_k - \lambda\mathbb{1}, \underline{v}\mathbb{1}, \mathbb{1})$ and the following equality holds due to the constraint $\mathbb{1}^T v = \bar{v}$ being active:

$$\bar{v} = \mathbb{1}^T \text{Clamp}(v_k - \lambda\mathbb{1}, \underline{v}\mathbb{1}, \mathbb{1}), \tag{7}$$

which is a piecewise linear equation having at most $2E$ pieces. There are E left-nodes in the form of $v_k - \underline{v}\mathbb{1}$ and E right-nodes in the form of $v_k - \mathbb{1}$, separating different linear pieces. The piecewise linear equation can be solved for λ and thus v by first sorting the $2E$ nodes at the cost of $\mathcal{O}(E \log E)$ and then looking at each piece for the solution. We summarize this process in Algorithm 4 where we maintain the sorted end points of the line segments via running sums.

4.3. Fine-Grained Parallelism

Prior work [2] proposes to accelerate TO solver on GPU, but they require a complex GPU multgrid implementation which is also costly to compute per iteration. In comparison, our Algorithm 1,2,3 can make full use of many-core hardwares with slight modifications. In this section, we discuss necessary modifications for a GPU implementation assuming the availability of basic parallel scan, reduction, inner-product, and radix sort operations [44]. A bottleneck in Algorithm 1,2,3 is matrix-vector production $K(v)u$, of which the implementation depends on the type of FEA discretization. If the discretization uses a regular pattern, then the element-to-node mapping is known and can be hard coded, so the computation for each node is independent and costs $\mathcal{O}(1)$ if E thread blocks are available. For irregular discretizations, we have to store the sparse matrix explicitly and suggest using a parallel sparse matrix-vector product routine [45]. To implement the Krylov-preconditioner, we perform in-place QR factorization to solve for c_i^*. The in-place QR factorization involves computing the inner-product for D^2 times and then solving the upper-triangular system. Altogether, the least square solve in Equation 6 costs $\mathcal{O}(D^2 \log E + D^2)$ when E threads are available. Here we use a serial implementation of the upper-triangular solve, which is not a performance

Algorithm 4 Project v_k onto X (SLOPE is the slope of each line segment, RHS is the righthand side of Equation 7)

if $\mathbb{1}^T \text{Proj}_{\{\mathbb{1}\geq v\geq \underline{v}\}}(v_k) \leq \bar{v}$ **then**
  Return $\text{Proj}_{\{\mathbb{1}\geq v\geq \underline{v}\}}(v_k)$
ARRAY $\leftarrow \varnothing$
for $v_k^e \in v_k$ **do**
  ARRAY $\leftarrow$ ARRAY $\bigcup \{v_k^e - \underline{v}, v_k^e - 1\}$
Sort ARRAY from low to high $\triangleright\ \mathcal{O}(E \log E)$
SLOPE $\leftarrow 0$, RHS $\leftarrow E$, NODES $\leftarrow \varnothing$, $\lambda^*_{\text{last}} \leftarrow 0$
for $\lambda^* \in$ ARRAY **do** $\triangleright\ \mathcal{O}(E)$
  RHS $\leftarrow$ RHS + SLOPE$(\lambda^* - \lambda^*_{\text{last}})$
  if λ^* of form $v_k^e - 1$ **then**
    SLOPE $\leftarrow$ SLOPE $- 1$
  else SLOPE $\leftarrow$ SLOPE $+ 1$ $\triangleright\ \lambda^*$ of form $v_k^e - \underline{v}$
  NODES $\leftarrow$ NODES $\bigcup \{< \lambda^*, \text{RHS} >\}$, $\lambda^*_{\text{last}} \leftarrow \lambda^*$
for Consecutive NODE, NODE$' \in$ NODES **do** $\triangleright\ \mathcal{O}(E)$
  if Segment $<$ NODE, NODE$' >$ passes through $\bar{v}$ **then**
    Solve for λ and return Clamp$(v_k - \lambda\mathbb{1}, \underline{v}\mathbb{1}, \mathbb{1})$

penalty when $D \ll E$. Finally, Algorithm 4 involves one radix sort that costs $\mathcal{O}(\log E)$ and three for loops, the first and last for loops do not have data dependency and cost $\mathcal{O}(1)$. The second for loop accumulates two variables (Slope, Total) that can be accomplished by a parallel scan taking $\mathcal{O}(\log E)$. In summary, the cost of each iteration of Algorithm 1,2,3 can be reduced to $\mathcal{O}(D^2 \log E)$ when $\mathcal{O}(E)$ threads are available. We profile the parallel acceleration rate in Figure 4.

5. Evaluation

We implemented FBTO using native C++ on a laptop machine with a 2.5G 6-core Intel i7 CPU and a 1.48G Nvidia GTX 1060 mobile GPU having 1280 cores. Our implementation makes full use of the cores on CPU via OpenMP and GPU via Cuda and we implement the regular grid FEA discretization. In all the examples, we apply a $3^2 - 7^2$ separable Gaussian kernel $\mathbb{C}$ as our material filter (see Remark 3.2), which can be implemented efficiently on GPU as a product of multiple linear kernels. The exact kernel

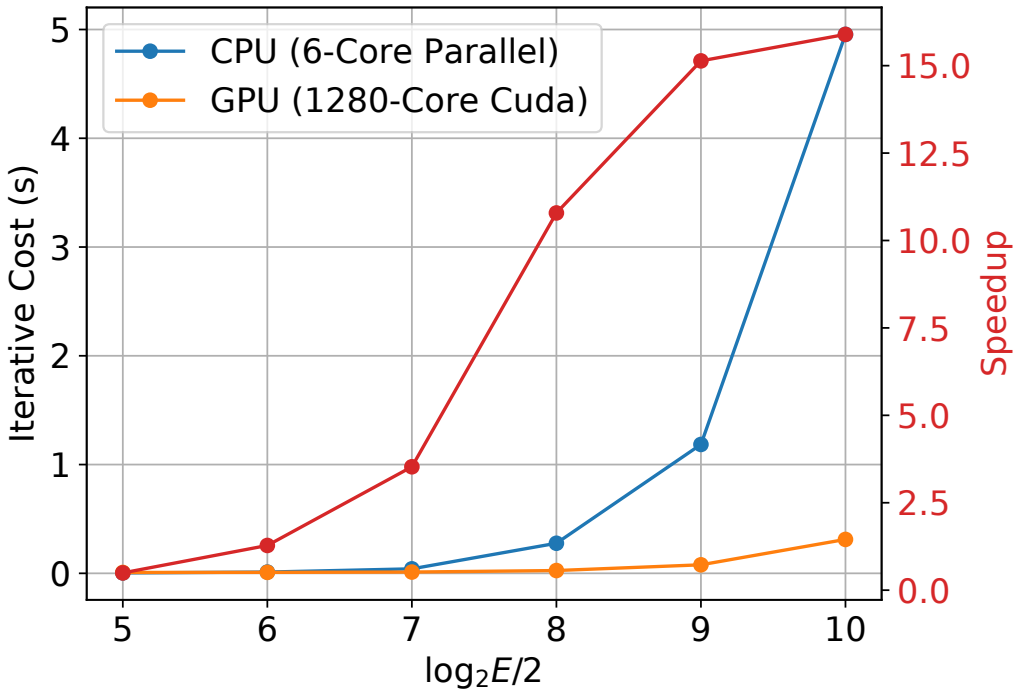

Figure 4: The acceleration rate of low-end mobile GPU (1280 cores) versus CPU (6 cores) implementation under different resolutions of discretization. We achieve a maximal speedup of 16 times.

size varies across examples. Unless otherwise stated, we choose Algorithm 3 with $\underline{v} = 0.1, \eta = 3, \alpha_k = \alpha_0 k^{-3/4}, \beta_k = 1, D = 20$ for our Krylov-preconditioner in all the experiments where α_0 varies across different experiments. We justify these parameter choices in Section 8.5. For the three algorithms in Section 4, we terminate when the relative change to infill levels over two iterations ($\|v_{k+1} - v_k\|_\infty$) is smaller than 10^{-4} and the absolute error of the linear system solve ($\max_{q=1,\cdots,Q} \|K(v_k)u_k^q - f^q(v_k)\|_\infty$) is smaller than 10^{-2}, which can be efficiently checked on GPU by using a reduction operator that costs $\mathcal{O}(\log E)$. The two conditions are measuring the optimality of the high- and low-level problems, respectively.

Low-Iterative Cost. We run FBTO on a row of standard 2D and 3D computational benchmarks described in [1, 2, 3]. These benchmarks only consider a single load condition without self-weight. We summarize the iterative cost and total computational cost in Table 1 and Table 2. The iterative cost

Benchmark	Res.	Frac.	#Iter.	Time (Total)	Time (Iter.)	Mem.
Figure 2a	192×576	0.3	14600	584	0.040	24
Figure 2b	192×576	0.3	8900	356	0.040	24
Figure 2c	192×384	0.3	37400	1159	0.031	18
Figure 2d	160×240	0.4	20400	346	0.017	8
Figure 2e	160×160	0.5	13300	146	0.011	4
Figure 2f	128×384	0.5	9300	195	0.021	10
Figure 2g	128×256	0.3	19400	252	0.013	8

Table 1: Statistics of 2D benchmark problems in Figure 2: benchmark index, grid resolution, volume fraction ($\bar{v}$), iterations until convergence, total computational time(s), iterative cost(s), and GPU memory cost (mb).

ranges from 4 – 24ms in 2D and 40 – 120ms in 3D. We further compare our method with Projected Gradient Descent (PGD). Using a single-level formulation, PGD differs from Algorithm 3 by using exact matrix inversions, i.e., replacing Line 3 with $u^q_{k+1} \leftarrow K^{-1}(v_k) f^q(v_k)$. Since exact, sparse matrix factorization is a serial algorithm, we implement other steps of PGD on GPU while revert to CPU for the factorization. According to Table 1, our method incurs an order of magnitude lower iterative cost. We further plot the convergence history of both methods for the benchmark in Figure 1a, where our method converges after 576s on this example. In Figure 5, we plot the design evolution against iteration numbers for the benchmark in Figure 1g. Even at an early stage of optimization, the design is already very similar to the converged solution. As a result, our low iterative cost provides the unique opportunity for designers to quickly preview the optimized design.

Convergent Behavior. Our algorithm relies on two critical parameters: initial high-level step size α_0 and Krylov subspace size D for preconditioning. We profile the sensitivity with respect to α_0 in Figure 6 for benchmark problem in Figure 2e. We find our method convergent over a wide range of α_0, although an overly large α_0 could lead to excessive initial fluctuation. In practice, matrix-vector product is the costliest step and takes up 96% of the computational time, so the iterative cost is almost linear in D. Algorithm 1 would require an extremely small α_0 and more iterations to converge, while

Benchmark	Res.	Frac.	#Iter.	Time (Total)	Time (Iter.)	Mem.
Figure 1a	$32 \times 96 \times 32$	0.3	37700	4524	0.120	10
Figure 1b	$32 \times 32 \times 32$	0.3	17400	696	0.040	3
Figure 1c	$32 \times 96 \times 32$	0.3	49000	5880	0.120	10
Figure 1d	$32 \times 96 \times 32$	0.3	42600	5112	0.120	10
Figure 1e	$32 \times 64 \times 32$	0.3	29100	2328	0.080	7

Table 2: Statistics of 3D benchmark problems in Figure 1: benchmark index, grid resolution, volume fraction ($\bar{v}$), iterations until convergence, total computational time(s), iteration cost(s), and GPU memory cost (mb).

Algorithm 3 even with a small D can effectively reduce the low-level error and allow a wide range of choices for α_0, leading to a faster overall convergence speed. We profiled the sensitivity to subspace size D in Figure 7, and we observe convergent behaviors for all α_0 when $D \geq 15$, so we choose $D = 20$ in all examples and only leave α_0 to be tuned by users.

Comparison with Optimality Criteria (OC). We compare our method with the heuristic solver OC [12]. For fairness, we implement a GPU-based OC in our framework. During each iteration, we use the standard conjugate gradient algorithm to solve the FEA system to sufficiently high-precision on GPU. We use all the parameters suggested in [12]. Note that it is inherently difficult to find a single performance indicator for both solvers. We find a reasonable indicator to be the relative projection error $\sqrt{\Delta_k^v}$. However, even computing $\sqrt{\Delta_k^v}$ requires the exact gradient, which is intractable for our method, so we use our inexact gradient as a replacement of $\nabla l(v_k)$ in $\sqrt{\Delta_k^v}$. For OC, a trust-region-like move limit is adopted, which is set to 0.2, so we set all $\alpha_k = 0.2$ in $\sqrt{\Delta_k^v}$. As illustrated in Figure 8, OC visually converges within 100 outer iterations, while numerically the projection error cannot be reduced to zero, i.e., OC does not converge to locally optimal solution. Instead, our projection error is continually reducing to zero. On the other hand, the visual quality of both methods are comparable as illustrated in Figure 9. After 100s of computation, both methods can find the coarse structures, while the detailed structures are optimized after 200s of computation.

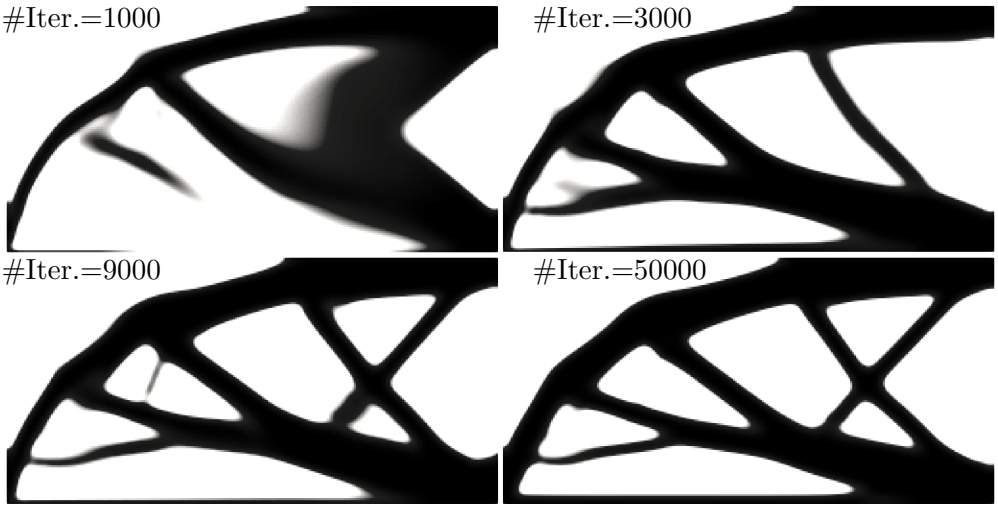

Figure 5: We plot the intermediary designs at different iteration numbers for the benchmark in Figure 1g. The result after 3000 iterations is already quite similar to the ultimate solution.

Self-Weight and Multiple Load Conditions. In Figure 10, we show 3D mechanical designs optimized under different conditions. We use the same power law to discretize Equation 3 with $\eta = 3$. Drastically different designs are found under self-weight or multiple load conditions. The additional cost of considering self-weight is marginal, while considering multiple load conditions lead to a major computational overhead. Since low-level solve is the most costly step of our method, considering C external loads can lead to an approximately C times higher iterative cost. For the example in Figure 10, we use a resolution of $32 \times 64 \times 32$ and the iterative cost with a single, and four loads are 80ms and 350ms, respectively.

6. Conclusion & Limitation

We present a new solver for a subclass of TO problems: the SIMP model. Our method revisits the TO formulation as a bilevel program and solve the optimization via first-order algorithms with inexact FEA system solvers. Both theoretical analysis and computational benchmarks show that our method has a much lower iterative cost of $\mathcal{O}(E \log E)$ on CPU and $\mathcal{O}(\log E)$ on GPU, and our method is guaranteed to converge. In practice, the iterative cost is less than 1 second, allowing designers to interactively

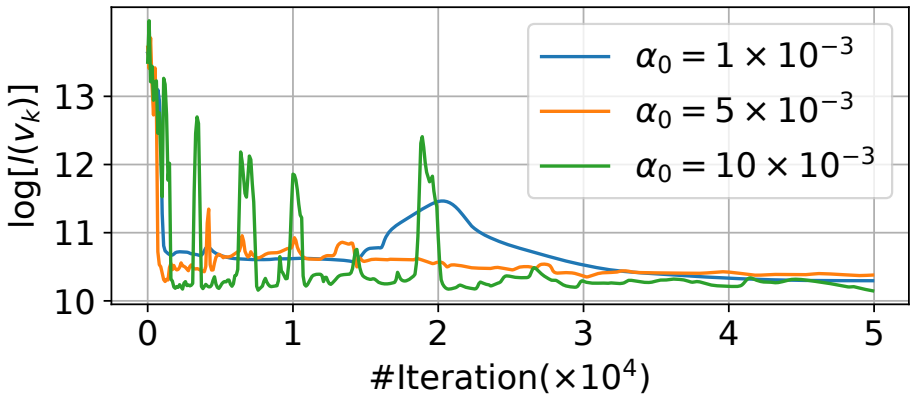

Figure 6: For the benchmark problem in Figure 2e, we compared the convergence history of three choices of α_0. A larger α_0 would lead to more fluctuations at an early stage, but all three optimizations would ultimately converge. When $\alpha_0 > 10^{-2}$, we observe divergence.

preview the optimized design and accelerate the design refinement loop. We further show that our method can solve TO under self-weight and multiple external load conditions, which is essential for designing mechanical parts built from heavy materials and work under a variety of conditions.

Our theoretical framework has several limitations that open doors to future research. First, our method falls into the category of first-order method that can quickly converge to a solution of medium accuracy. For highly accurate solutions, our method can take a large number of iterations making the performance inferior than convention method [12] using exact FEA solvers. Second, our theoretical convergence speed is not optimal and techniques such as momentum acceleration [46] can be adopted for an improvement, although it introduces additional parameters to be tuned. In addition, our Krylov preconditioner cannot scale to high-resolution FEA meshes because its convergence speed is not resolution-independent. A widely used method for resolution-independent performance is multigrid [25], but the convergence properties of our method under such preconditioner require further investigation. In particular, it is unknown whether we can still use $\beta_k = 1$ with a multigrid preconditioner. Finally, there are many advanced TO formulations using additional constraints for stress [20] and minimum length-scale controls [47, 48]. It is unclear whether inexact FEA solvers can be utilized in these formulations.

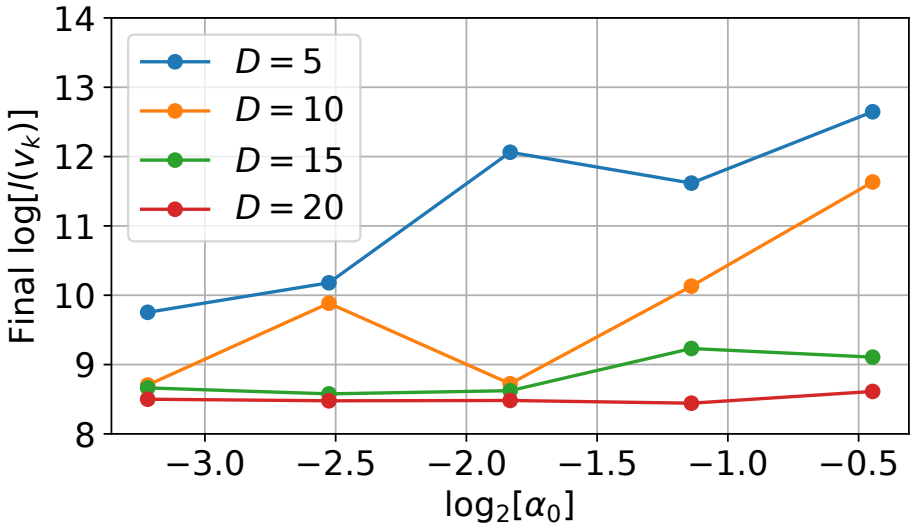

Figure 7: We run our method for 5000 iterations and plot the final exact energy value against α_0 where $\log(l(v_k)) > 9$ indicates divergence. When $D \geq 15$, our method is convergent under all the step sizes. Therefore, we choose $D = 20$ in all examples and only leave α_0 to be tuned by users.

7. Convergence Analysis of Algorithm 1

In this section, we introduce a set of assumptions on the choices of parameters, under which we prove the convergence of our method. For simplicity of presentation, we always assume a single load condition $C = 1$ and no self-weight. The analysis with multiple load conditions or self-weight follows exactly the same reasoning, with only minor notational changes.

Assumption 7.1. *We choose constant β_k and some constant $p > 0$ satisfying the following condition:*

$$0 < \beta_k < \frac{2\underline{\rho}}{\bar{\rho}^2} \wedge p > \frac{1 - 2\beta_k\underline{\rho} + \bar{\beta}^2\beta_k^2}{2\beta_k\underline{\rho} - \bar{\beta}^2\beta_k^2}. \tag{8}$$

Assumption 7.2. *Define $\Delta_k \triangleq \|u_k - u_f(v_k)\|^2$ and suppose $\Delta_1 \leq U^2$, we choose the following uniformly bounded sequence $\{\Gamma_k\}$ with constant $\bar{\Gamma}, \bar{\Theta}$:*

$$\bar{\Theta} \triangleq \frac{p+1}{p}(1 - 2\beta_k\underline{\rho} + \beta_k^2\bar{\rho}^2) \wedge \Gamma_k \leq \bar{\Gamma} \leq \frac{U^2(1-\bar{\Theta})}{(U+U_f)^4},$$

where U_f is the finite upper bound of $u_f(v)$ on X.

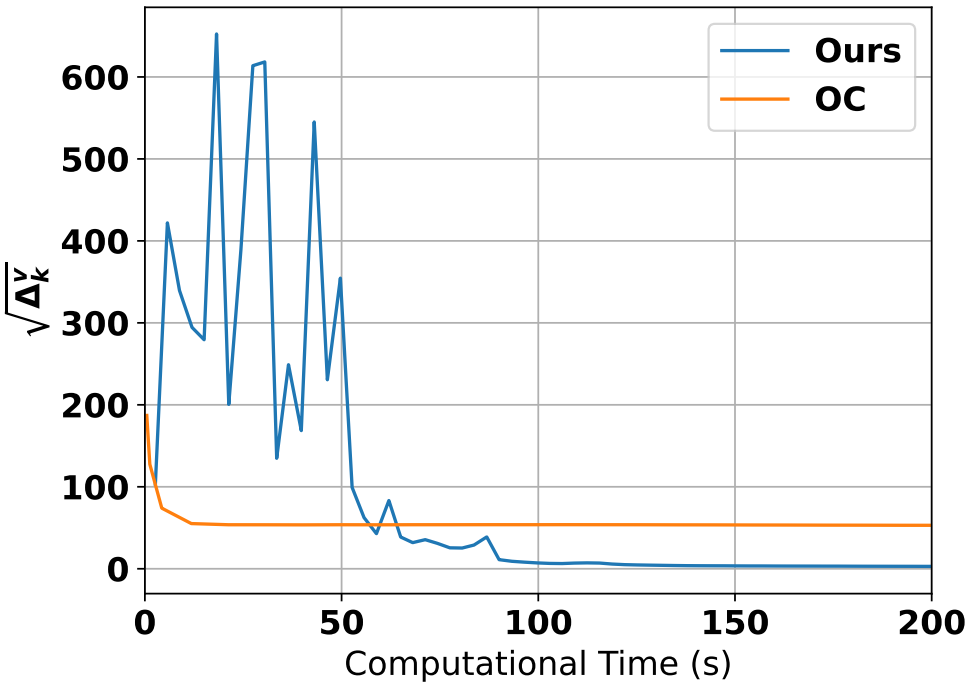

Figure 8: We plot the relative projection error $\sqrt{\Delta_k^v}$ against computational time for both our method and OC [12] on benchmark of Figure 2a. OC visually converges to a non-optimal solution within 100 outer iterations, while our method takes more iterations to converge to a locally optimal solution.

Assumption 7.3. *We choose α_k as follows:*

$$\alpha_k = \frac{1}{k^m}\sqrt{\frac{\bar{\Gamma}}{L_u^2 L_{\nabla K}^2}\frac{4}{p+1}},$$

for positive constants m, p, where L_u is the L-constant of u_f and $L_{\nabla K}$ is the upper bound of the Frobenius norm of $\partial K(v)/\partial v$ when $v \in X$.

Assumption 7.4. *We choose positive constants m, n satisfying the following condition:*

$$\max(1-2m+mn, 2-2m-mn, 2-3m) < 0 \wedge m < 1.$$

Theorem 7.5. *Suppose $\{v_k\}$ is the sequence generated by Algorithm 1 under Assumption 3.1, 7.1, 7.2, 7.3, 7.4, with a single load condition and no*

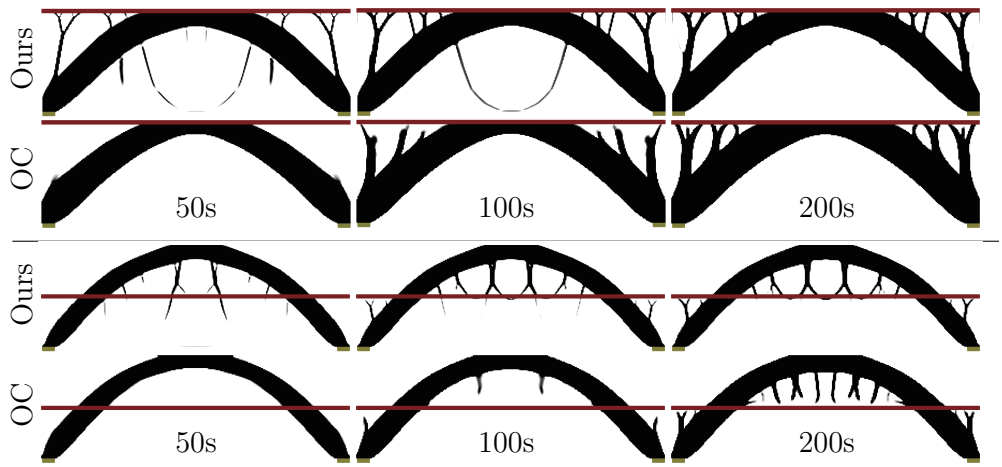

Figure 9: We show the intermediary result of our method and OC after 50s, 100s, and 200s of computation.

self-weight. For any $\epsilon > 0$*, there exists an iteration number* k *such that* $\|\nabla_X l(v_k)\| \leq \epsilon$*, where* $\nabla_X l(v_k)$ *is the projected negative gradient into the tangent cone of* X*,* $\mathcal{T}_X$*, defined as:*

$$\nabla_X l(v_k) = \underset{d \epsilon \mathcal{T}_X}{argmin} \|d + \nabla l(v_k)\|.$$

To prove Theorem 7.5, we start from some immediate observations on the low- and high-level objective functions. Then, we prove the rule of error propagation for the low-level optimality, i.e., the low-level different between u_k and its optimal solution $u_f(v_k)$. Next, we choose parameters to let the difference diminish as the number of iterations increase. Finally, we focus on the high-level objective function and show that the difference between v_k and a local optima of $l(v)$ would also diminish. Our analysis resembles the recent analysis on two-timescale bilevel optimization, but we use problem-specific treatment to handle our novel gradient estimation for the high-level problem without matrix inversion.

7.1. Low-Level Error Propagation

If Assumption 3.1 holds, then the low-level objective function is $\underline{\rho}$-strongly convex and Lipschitz-continuous in u for any v with $\bar{\rho}$ being the L-constant,

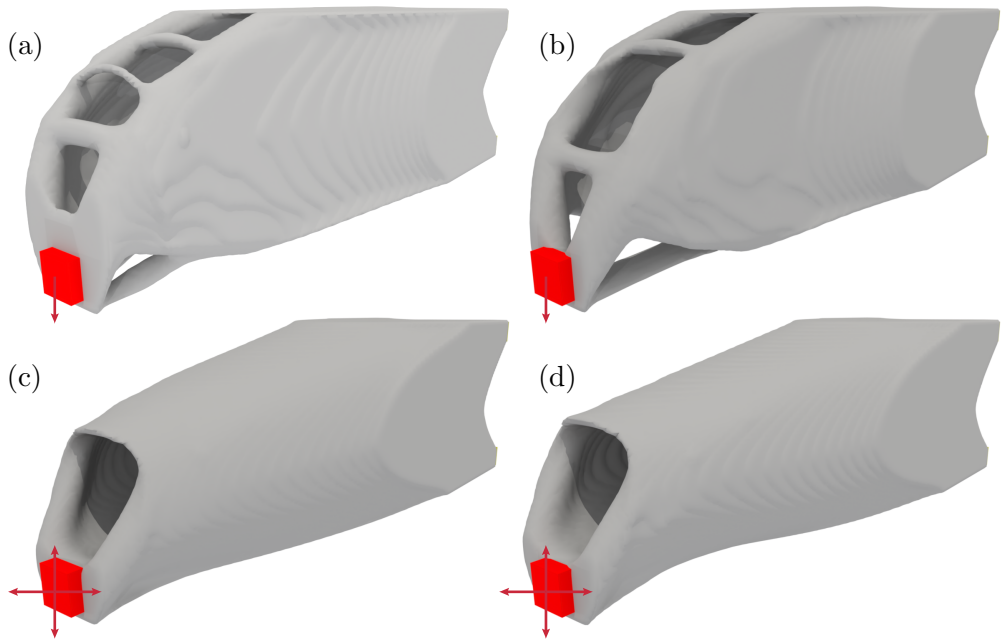

Figure 10: We show 3D mechanical parts optimized under different conditions: (a) single load without self-weight; (b) single load with self-weight; (c) 4 loads without self-weight; (d) 4 loads with self-weight. For this example, we set the volume fraction to be 0.3. The existence of self-weight and multiple load conditions can drastically change the design.

so that:

$$P_f(u_{k+1}, v_k) \leq P_f(u_k, v_k) + \left\langle u_{k+1} - u_k, \nabla_u P_f(u_k, v_k) \right\rangle + \frac{\bar{\rho}}{2}\|u_{k+1} - u_k\|^2 \leq P_f(u_k, v_k) + \left[\frac{\bar{\rho}}{2} - \frac{1}{\beta_k}\right]\|u_{k+1} - u_k\|^2. \tag{9}$$

Similarly, we can immediately estimate the approximation error of the low-level problem due to an update on u:

Lemma 7.6. *Under Assumption 3.1, the following relationship holds for all $k \geq 1$:*

$$\Delta_{k+1} \leq \frac{p+1}{p}(1 - 2\beta_k \underline{\rho} + \beta_k^2 \bar{\rho}^2)\Delta_k + L_u^2 L_{\nabla K}^2 \alpha_k^2 \frac{p+1}{4}\|u_k\|^4.$$

Proof. The following result holds by the triangle inequality and the bounded

spectrum of $K(v)$ (Equation 2):

$$\begin{aligned}
&\|u_{k+1} - u_f(v_k)\|^2 \\
=&\|u_{k+1} - u_k\|^2 + \|u_k - u_f(v_k)\|^2 + \\
&2\langle u_{k+1} - u_k, u_k - u_f(v_k)\rangle \\
=&\|u_{k+1} - u_k\|^2 + \|u_k - u_f(v_k)\|^2 - \\
&2\beta_k \langle K(v_k)u_k - f, u_k - u_f(v_k)\rangle \\
\leq&\|u_{k+1} - u_k\|^2 + (1 - 2\beta_k\underline{\rho})\|u_k - u_f(v_k)\|^2 \\
\leq&\beta_k^2\bar{\rho}^2\|u_k - u_f(v_k)\|^2 + (1 - 2\beta_k\underline{\rho})\|u_k - u_f(v_k)\|^2. \qquad (10)
\end{aligned}$$

The optimal solution to the low-level problem is $u_f(v)$, which is a smooth function defined on a compact domain, so any derivatives of $u_f(v)$ is bounded L-continuous and we have the following estimate of the change to u_f due to an update on v:

$$\begin{aligned}
&\|u_{k+1} - u_f(v_k)\|\|u_f(v_k) - u_f(v_{k+1})\| \\
\leq&\frac{1}{2p}\|u_{k+1} - u_f(v_k)\|^2 + \frac{p}{2}\|u_f(v_k) - u_f(v_{k+1})\|^2 \\
\leq&\frac{1}{2p}\|u_{k+1} - u_f(v_k)\|^2 + \frac{pL_u^2}{2}\|v_k - v_{k+1}\|^2 \\
\leq&\frac{1}{2p}\|u_{k+1} - u_f(v_k)\|^2 + \frac{pL_u^2\alpha_k^2}{8}\|u_k^T\frac{\partial K}{\partial v_k}u_k\|^2, \qquad (11)
\end{aligned}$$

where we have used Young's inequality and the contractive property of the $\mathrm{Proj}_X(\bullet)$ operator. Here we denote L_u as the L-constant of u_f. By triangular

inequality, we further have:

$$
\begin{aligned}
&\|u_{k+1} - u_f(v_{k+1})\|^2 \\
\leq &\|u_{k+1} - u_f(v_k)\|^2 + \|u_f(v_k) - u_f(v_{k+1})\|^2 + \\
&2\|u_{k+1} - u_f(v_k)\|\|u_f(v_k) - u_f(v_{k+1})\| \\
\leq &\frac{p+1}{p}\|u_{k+1} - u_f(v_k)\|^2 + \|u_f(v_k) - u_f(v_{k+1})\|^2 + \\
&\frac{pL_u^2\alpha_k^2}{4}\|u_k^T\frac{\partial K}{\partial v_k}u_k\|^2 \\
\leq &\frac{p+1}{p}\|u_{k+1} - u_f(v_k)\|^2 + L_u^2\|v_k - v_{k+1}\|^2 + \\
&\frac{pL_u^2\alpha_k^2}{4}\|u_k^T\frac{\partial K}{\partial v_k}u_k\|^2 \\
\leq &\frac{p+1}{p}\|u_{k+1} - u_f(v_k)\|^2 + L_u^2\alpha_k^2\frac{p+1}{4}\|u_k^T\frac{\partial K}{\partial v_k}u_k\|^2 \\
\leq &\frac{p+1}{p}(1 - 2\beta_k\underline{\rho} + \beta_k^2\bar{\rho}^2)\|u_k - u_f(v_k)\|^2 + \\
&L_u^2\alpha_k^2\frac{p+1}{4}\|u_k^T\frac{\partial K}{\partial v_k}u_k\|^2 \\
\leq &\frac{p+1}{p}(1 - 2\beta_k\underline{\rho} + \beta_k^2\bar{\rho}^2)\|u_k - u_f(v_k)\|^2 + \\
&L_u^2L_{\nabla K}^2\alpha_k^2\frac{p+1}{4}\|u_k\|^4, \qquad (12)
\end{aligned}
$$

where we have used Equation 10 and Equation 11. □

The result of Lemma 7.6 is a recurrent relationship on the low-level optimality error Δ_k, which will be used to prove low-level convergence via recursive expansion.

7.2. Low-Level Convergence

We use the following shorthand notation for the result in Lemma 7.6:

$$
\begin{aligned}
&\Delta_{k+1} \leq \Theta_k\Delta_k + \Gamma_k\|u_k\|^4 \\
&\Theta_k \triangleq \frac{p+1}{p}(1 - 2\beta_k\underline{\rho} + \beta_k^2\bar{\rho}^2) \quad \Gamma_k \triangleq L_u^2L_{\nabla K}^2\alpha_k^2\frac{p+1}{4}. \qquad (13)
\end{aligned}
$$

By taking Assumption 7.1 and direct calculation, we can ensure that $\Theta_k \leq \bar{\Theta} < 1$ (i.e., the first term is contractive). To bound the growth of the second term above, we show by induction that both Δ_k and u_k can be uniformly bounded for all k via a sufficiently small, constant $\Gamma_k \leq \bar{\Gamma}$.

Lemma 7.7. *Taking Assumption 3.1, 7.1, 7.2, we have* $\Delta_k \le U^2, \|u_k\| \le U_f + U$ *for all* $k \ge 1$*, where* U_f *is the uniform upper bound for* $\|u_f(v)\|$*.*

Proof. First, since $v \in X$ and X is compact, we have $\|u_f(v)\| \le U_f < \infty$. Next, we prove $\Delta_k \le U^2$ by induction. We already have $\Delta_1 \le U^2$. Now suppose $\Delta_k \le U^2$, then Equation 13 and our assumption on $\bar{\Gamma}$ immediately leads to:

$$\begin{aligned}\Delta_{k+1} \le& \bar{\Theta}U^2 + \bar{\Gamma}(\|u_k - u_f(v_k)\| + \|u_f(v_k)\|)^4 \\ \le& \bar{\Theta}U^2 + \bar{\Gamma}(U + U_f)^4 \le U^2.\end{aligned}$$

Finally, we have: $\|u_k\| \le \|u_k - u_f(v_k)\| + \|u_f(v_k)\| \le U + U_f$ and our lemma follows. □

The shrinking coefficient $\bar{\Theta}$ and the uniform boundedness of Δ_k, u_k allows us to establish low-level convergence with the appropriate choice of $\alpha_k = \mathcal{O}(k^{-m})$ with $m \ge 1$.

Theorem 7.8. *Taking Assumption 3.1, 7.1, 7.2, 7.3, we can upper bound* Δ_{k+1} *as:*

$$\Delta_{k+1} \le \bar{\Theta}^{k-1}\Delta_1 + (U + U_f)^4\Gamma_1 \sum_{i=0}^{k-1} \frac{\bar{\Theta}^i}{(k-i)^{2m}} = \mathcal{O}(k^{1-2m}).$$

Proof. Recursively expand on Equation 13 and we have:

$$\begin{aligned}&\Delta_{k+1} \le \bar{\Theta}\Delta_k + \Gamma_k\|u_k\|^4 \\ \le& \bar{\Theta}(\bar{\Theta}\Delta_{k-1} + \Gamma_{k-1}\|u_{k-1}\|^4) + \Gamma_k\|u_k\|^4 \\ =& \bar{\Theta}^2\Delta_{k-1} + \bar{\Theta}\Gamma_{k-1}\|u_{k-1}\|^4 + \Gamma_k\|u_k\|^4 \\ \le& \bar{\Theta}^2(\bar{\Theta}\Delta_{k-2} + \Gamma_{k-2}\|u_{k-2}\|^4) + \bar{\Theta}\Gamma_{k-1}\|u_{k-1}\|^4 + \Gamma_k\|u_k\|^4 \\ &\dots \\ \le& \bar{\Theta}^k\Delta_1 + (U + U_f)^4 \sum_{i=0}^{k-1} \bar{\Theta}^i\Gamma_{k-i} \\ \le& \bar{\Theta}^k\Delta_1 + (U + U_f)^4\Gamma_1 \sum_{i=0}^{k-1} \frac{\bar{\Theta}^i}{(k-i)^{2m}} \\ \le& \bar{\Theta}^k\Delta_1 + (U + U_f)^4\Gamma_1 \left[\sum_{i=0}^{\lceil \frac{k-1}{2} \rceil} \frac{\bar{\Theta}^i}{(k-i)^{2m}} + \sum_{i=\lceil \frac{k+1}{2} \rceil}^{k-1} \frac{\bar{\Theta}^i}{(k-i)^{2m}}\right] \\ \le& \bar{\Theta}^k\Delta_1 + (U + U_f)^4\Gamma_1 \left[\frac{\lceil \frac{k+1}{2} \rceil}{(k - \lceil \frac{k-1}{2} \rceil)^{2m}} + \frac{\bar{\Theta}^{\lceil \frac{k}{2} \rceil}}{1 - \bar{\Theta}}\right] = \mathcal{O}(k^{1-2m}),\end{aligned}$$

where we have used our choice of α_k and Lemma 7.7. □

Theorem 7.8 is pivotal by allowing us to choose m and tune the convergence speed of the low-level problem, which is used to establish the convergence of high-level problem.

7.3. High-Level Error Propagation

The high-level error propagation is similar to the low-level analysis, which is due to the L-continuity of any derivatives of $l(v)$ in the compact domain X. The following result reveals the rule of error propagation over a single iteration:

Lemma 7.9. *Under Assumption 3.1, 7.1, 7.2, 7.3, the following high-level error propagation rule holds for all $k \geq 1, q > 0$:*

$$l(v_{k+1}) - l(v_k) \leq -\frac{1}{\alpha_k}\|v_{k+1} - v_k\|^2 + \frac{L_{\nabla K}^2 \alpha_k^2 (L_{\nabla l} q + 1)}{8q}(U + U_f)^4 + \frac{L_{\nabla K}^2 q}{8}\Delta_k (U + 2U_f)^2.$$

Proof. By the smoothness of $l(v)$, the compactness of X, and the obtuse angle criterion, we have:

$$\begin{aligned}
&l(v_{k+1}) - l(v_k) \leq \langle v_{k+1} - v_k, \nabla l(v_k)\rangle + \frac{L_{\nabla l}}{2}\|v_{k+1} - v_k\|^2 \\
\leq &\left\langle v_{k+1} - v_k, -\frac{1}{2}u_f^T(v_k)\frac{\partial K}{\partial v_k}u_f(v_k)\right\rangle + \frac{L_{\nabla l} L_{\nabla K}^2 \alpha_k^2}{8}\|u_k\|^4 \\
= &\left\langle v_{k+1} - v_k, -\frac{1}{2}u_k^T\frac{\partial K}{\partial v_k}u_k\right\rangle + \frac{L_{\nabla l} L_{\nabla K}^2 \alpha_k^2}{8}\|u_k\|^4 + \\
&\left\langle v_{k+1} - v_k, \frac{1}{2}u_k^T\frac{\partial K}{\partial v_k}u_k - \frac{1}{2}u_f^T(v_k)\frac{\partial K}{\partial v_k}u_f(v_k)\right\rangle \\
\leq &-\frac{1}{\alpha_k}\|v_{k+1} - v_k\|^2 + \frac{L_{\nabla l} L_{\nabla K}^2 \alpha_k^2}{8}\|u_k\|^4 + \\
&\left\langle v_{k+1} - v_k, \frac{1}{2}u_k^T\frac{\partial K}{\partial v_k}u_k - \frac{1}{2}u_f^T(v_k)\frac{\partial K}{\partial v_k}u_f(v_k)\right\rangle, \qquad (14)
\end{aligned}$$

where $L_{\nabla l}$ is the L-constant of ∇l on X. For the last term above, we can

bound it by applying the Young's inequality:

$$\begin{aligned}
&\left\langle v_{k+1}-v_k, \frac{1}{2}u_k^T\frac{\partial K}{\partial v_k}u_k - \frac{1}{2}u_f^T(v_k)\frac{\partial K}{\partial v_k}u_f(v_k)\right\rangle \\
\leq &\frac{1}{2q}\|v_{k+1}-v_k\|^2 + \frac{q}{2}\|\frac{1}{2}u_k^T\frac{\partial K}{\partial v_k}u_k - \frac{1}{2}u_f^T(v_k)\frac{\partial K}{\partial v_k}u_f(v_k)\|^2 \\
\leq &\frac{L_{\nabla K}^2\alpha_k^2}{8q}\|u_k\|^4 + \frac{q}{2}\|\frac{1}{2}u_k^T\frac{\partial K}{\partial v_k}u_k - \frac{1}{2}u_f^T(v_k)\frac{\partial K}{\partial v_k}u_f(v_k)\|^2 \\
= &\frac{L_{\nabla K}^2\alpha_k^2}{8q}\|u_k\|^4 + \frac{q}{2}\|\frac{1}{2}(u_k-u_f(v_k))^T\frac{\partial K}{\partial v_k}(u_k+u_f(v_k))\|^2 \\
\leq &\frac{L_{\nabla K}^2\alpha_k^2}{8q}\|u_k\|^4 + \frac{L_{\nabla K}^2 q}{8}\Delta_k\|u_k+u_f(v_k)\|^2. \qquad (15)
\end{aligned}$$

The lemma follows by combining Equation 14, Equation 15, and Lemma 7.7. □

7.4. High-Level Convergence

We first show that Δ_k^v is diminishing via the follow lemma:

Lemma 7.10. *Under Assumption 3.1, 7.1, 7.2, 7.3, we have the following bound on the accumulated high-level error* Δ_k^v*:*

$$\begin{aligned}
&\frac{1}{2}\sum_{i=1}^k \alpha_i\Delta_i^v \leq l(v_1) - \underline{l} + \\
&\frac{L_{\nabla K}^2}{8}(U+U_f)^4\sum_{i=1}^k \alpha_i^{2-n}(L_{\nabla l}\alpha_i^n + 1) + \\
&\frac{L_{\nabla K}^2}{8}(U+2U_f)^2\sum_{i=1}^k \alpha_i^n\Delta_i + \frac{L_{\nabla K}^2}{4}(U+2U_f)^2\sum_{i=1}^k \alpha_i\Delta_i,
\end{aligned}$$

where $\underline{l}$ *is the lower bound of* l *on* X*.*

Proof. By choosing $q = \alpha_k^n$ for some constant n and summing up the recursive rule of Lemma 7.9, we have the following result:

$$\begin{aligned}
&l(v_{k+1}) - l(v_1) \leq -\sum_{i=1}^k \frac{1}{\alpha_i}\|v_{i+1}-v_i\|^2 + \\
&\frac{L_{\nabla K}^2}{8}(U+U_f)^4\sum_{i=1}^k \alpha_i^{2-n}(L_{\nabla l}\alpha_i^n + 1) + \\
&\frac{L_{\nabla K}^2}{8}(U+2U_f)^2\sum_{i=1}^k \alpha_i^n\Delta_i. \qquad (16)
\end{aligned}$$

Equation 16 can immediately lead to the $\mathcal{O}(k^{-1})$ convergence of $\|v_{i+1}-v_i\|^2/\alpha_i$. However, the update from v_i to v_{i+1} is using the approximate gradient. To show the convergence of $l(v)$, we need to consider an update using the exact gradient. This can be achieved by combining Equation 16 and the gradient error estimation in Equation 15:

$$\begin{aligned}
&\Delta_k^v \leq \frac{1}{\alpha_k^2}\|v_{k+1}-v_k\|^2+\\
&\frac{1}{\alpha_k^2}\|\frac{\alpha_k}{2}u_f(v_k)^T\frac{\partial K}{\partial v_k}u_f(v_k)-\frac{\alpha_k}{2}u_k^T\frac{\partial K}{\partial v_k}u_k\|^2+\\
&\frac{2}{\alpha_k^2}\|v_{k+1}-v_k\|\|\frac{\alpha_k}{2}u_f(v_k)^T\frac{\partial K}{\partial v_k}u_f(v_k)-\frac{\alpha_k}{2}u_k^T\frac{\partial K}{\partial v_k}u_k\|\\
\leq&\frac{2}{\alpha_k^2}\|v_{k+1}-v_k\|^2+\frac{2}{\alpha_k^2}\|\frac{\alpha_k}{2}u_f(v_k)^T\frac{\partial K}{\partial v_k}u_f(v_k)-\frac{\alpha_k}{2}u_k^T\frac{\partial K}{\partial v_k}u_k\|^2\\
\leq&\frac{2}{\alpha_k^2}\|v_{k+1}-v_k\|^2+\frac{L_{\nabla K}^2}{2}\Delta_k(U+2U_f)^2. \qquad (17)
\end{aligned}$$

We can prove our lemma by combining Lemma 7.9, Equation 16, Equation 17. □

The last three terms on the righthand side of Lemma 7.10 are power series, the summand of which scales at the speed of $\mathcal{O}(k^{-2m+mn}), \mathcal{O}(k^{1-2m-mn}), \mathcal{O}(k^{1-3m})$, respectively. Therefore, for the three summation to be upper bounded for arbitrary k, we need the first condition in Assumption 7.4. The following corollary is immediate:

Corollary 7.11. *Under Assumption 3.1, 7.1, 7.2, 7.3, 7.4, we have* $\min_{i=1,\cdots,k}\alpha_i\Delta_i^v \leq Ck^{-1}$ *and there are infinitely many* k *such that* $\Delta_k^v \leq C_v k^{m-1}$ *for some constants* C, C_v *independent of* k.

Proof. By Lemma 7.10 and Assumption 7.4, we have $\sum_{i=1}^k \alpha_i\Delta_i^v \leq C$ for some constant C. Now suppose only finitely many k satisfies $\Delta_k^v \leq C_v k^{m-1}$, then after sufficiently large $k \geq K_0$, we have:

$$\sum_{i=K_0}^{k}\alpha_i\Delta_i^v \geq \sqrt{\frac{\bar{\Gamma}}{L_u^2L_{\nabla K}^2}\frac{4}{p+1}}C_v\sum_{i=K_0}^{k}\frac{1}{i},$$

which is divergent, leading to a contradiction. □

Our remaining argument is similar to the standard convergence proof of PGD [38], with minor modification to account for our approximate gradient:

Proof of Theorem 7.5. We denote by $\tilde{v}_{k+1}$:

$$\tilde{v}_{k+1} \triangleq \mathrm{Proj}_X\left(v_k - \alpha_k \nabla l(v_k)\right).$$

Note that $\tilde{v}_{k+1}$ is derived from v_k using the true gradient, while v_{k+1} is derived using our approximate gradient. Due to the polytopic shape of X in Assumption 3.1, for $v_k \in X$, we can always choose a feasible direction d_k from $\mathcal{T}_X(v_k)$ with $\|d_k\| \leq 1$ such that:

$$\|\nabla_X l(v_k)\| \leq -\langle \nabla l(v_k), d_k \rangle + \frac{\epsilon}{4}.$$

We further have the follow inequality holds for any $z \in X$ due to the obtuse angle criterion and the convexity of X:

$$\begin{aligned}
&\langle \alpha_k \nabla l(v_k), \tilde{v}_{k+1} - z \rangle \\
\leq &\langle \alpha_k \nabla l(v_k), \tilde{v}_{k+1} - z \rangle + \langle v_k - \alpha_k \nabla l(v_k) - \tilde{v}_{k+1}, \tilde{v}_{k+1} - z \rangle \\
= &\langle v_k - \tilde{v}_{k+1}, \tilde{v}_{k+1} - z \rangle \leq \|v_k - \tilde{v}_{k+1}\| \|\tilde{v}_{k+1} - z\|.
\end{aligned}$$

Applying Corollary 7.11 and we can choose sufficiently large k such that:

$$\begin{aligned}
&-\left\langle \nabla l(\tilde{v}_{k+1}), \frac{\tau_k d_k}{\tau_k \|d_k\|} \right\rangle \leq \frac{1}{\alpha_k} \|v_k - \tilde{v}_{k+1}\| \\
\leq &\frac{1}{\alpha_k} \|v_k - v_{k+1}\| + \frac{1}{\alpha_k} \|v_{k+1} - \tilde{v}_{k+1}\| \\
= &\sqrt{\Delta_k^v} + \frac{1}{\alpha_k} \|v_{k+1} - \tilde{v}_{k+1}\| \\
\leq &\frac{\epsilon}{4} + \sqrt{\frac{L_{\nabla K}^2}{4} \Delta_k \|u_k + u_f(v_k)\|^2} = \frac{\epsilon}{4} + \mathcal{O}(k^{(1-2m)/2}).
\end{aligned}$$

Here the first inequality holds by choosing sufficiently small k-dependent τ_k such that $z = \tilde{v}_{k+1} + \tau_k d_k \in X$ and using the fact that $\|d_k\| \leq 1$. The third inequality holds by choosing sufficently large k and Corollary 7.11. The last equality holds by the contractive property of projection operator, Theorem 7.8, and Lemma 7.9. □

7.5. Convergence Analysis of Algorithm 2

Informally, we establish convergence for the preconditioned Algorithm 2 by modifying Equation 10 as follows:

$$\begin{aligned}
&\|u_{k+1} - u_f(v_k)\|^2 \\
=&\|u_{k+1} - u_k\|^2 + \|u_k - u_f(v_k)\|^2 + \\
&2\langle u_{k+1} - u_k, u_k - u_f(v_k)\rangle \\
=&\|u_{k+1} - u_k\|^2 + \|u_k - u_f(v_k)\|^2 - \\
&2\beta_k \left\langle K(v_k)M^{-2}(v_k)(K(v_k)u_k - f), u_k - u_f(v_k)\right\rangle \\
\leq&\|u_{k+1} - u_k\|^2 + (1 - 2\beta_k \frac{\underline{\rho}^2}{\bar{\rho}_M^2})\|u_k - u_f(v_k)\|^2 \\
\leq&\beta_k^2 \frac{\bar{\rho}^4}{\underline{\rho}_M^4}\|u_k - u_f(v_k)\|^2 + (1 - 2\beta_k \frac{\underline{\rho}^2}{\bar{\rho}_M^2})\|u_k - u_f(v_k)\|^2.
\end{aligned}$$

Assumption 7.1 must also be modified to account for the spectrum $M(v)$. All the other steps are identical to those of Theorem 7.5.

8. Convergence Analysis of Algorithm 3

The main difference in the analysis of Algorithm 3 lies in the use of a different low-level error metric defined as $\Xi_k \triangleq \|K(v_k)u_k - f\|^2$. Unlike Δ_k which requires exact matrix inversion, Ξ_k can be computed at a rather low cost. We will further show that using Ξ_k for analysis would lead to a much larger, constant choice of low-level step size $\beta_k = 1$. Our result is formalized below:

Assumption 8.1. *We choose constant β_k and some constant $p > 0$ satisfying the following condition:*

$$0 < \beta_k < \frac{2\underline{\rho}\underline{\rho}_M^2}{\bar{\rho}^2\bar{\rho}_M} \wedge p > \frac{1 - 2\beta_k \frac{\underline{\rho}}{\bar{\rho}_M} + \beta_k^2 \frac{\bar{\rho}^2}{\underline{\rho}_M^2}}{2\beta_k \frac{\underline{\rho}}{\bar{\rho}_M} - \beta_k^2 \frac{\bar{\rho}^2}{\underline{\rho}_M^2}}. \tag{18}$$

Assumption 8.2. *Define $\Xi_k \triangleq \|K(v_k)u_k - u_f\|^2$ and suppose $\Xi_1 \leq U^2$, we choose the following uniformly bounded sequence $\{\Gamma_k\}$ with constant $\bar{\Gamma}, \bar{\Theta}$:*

$$\Theta_k \triangleq \frac{p+1}{p}(1 - 2\beta_k \frac{\underline{\rho}}{\bar{\rho}_M} + \beta_k^2 \frac{\bar{\rho}^2}{\underline{\rho}_M}) \wedge \Gamma_k \leq \bar{\Gamma} \leq \frac{U^2(1 - \bar{\Theta})}{(U/\underline{\rho} + U_f)^6},$$

where U_f is the finite upper bound of $u_f(v)$ on X.

Assumption 8.3. *We choose α_k as follows:*

$$\alpha_k = \frac{1}{k^m}\sqrt{\frac{\bar{\Gamma}}{L_K^2 L_{\nabla K}^2}\frac{4}{p+1}},$$

for some positive constants m, p, where L_K is the L-coefficient of $K(v)$'s Frobenius norm on X.

Theorem 8.4. *Suppose $M(v)$ commutes with $K(v)$, $\{v_k\}$ is the sequence generated by Algorithm 3 under Assumption 3.1, 8.1, 8.2, 8.3, 7.4, 4.2, with a single load condition and no self-weight. For any $\epsilon > 0$, there exists an iteration number k such that $\|\nabla_X l(v_k)\| \le \epsilon$, where $\nabla_X l(v_k)$ is the projected negative gradient into the tangent cone of X.*

To prove Theorem 8.4, we follow the similar steps as Theorem 7.5 and only list necessary changes in this section.

8.1. Low-Level Error Propagation

Lemma 8.5. *Assuming $M(v)$ commutes with $K(v)$ and 3.1, the following relationship holds for all $k \ge 1$:*

$$\Xi_{k+1} \le \frac{p+1}{p}(1 - 2\beta_k\frac{\underline{\rho}}{\bar{\rho}_M} + \beta_k^2\frac{\bar{\rho}^2}{\underline{\rho}_M^2})\Xi_k + L_K^2 L_{\nabla K}^2 \alpha_k^2 \frac{p+1}{4}\|u_k\|^6,$$

where we define: $\|K(v_{k+1}) - K(v_k)\|_F^2 \le L_K\|v_{k+1} - v_k\|^2$ for any $v_k, v_{k+1} \in X$.

Proof. The following result holds by the triangle inequality and the bounded spectrum of $K(v)$ (Equation 2):

$$\begin{aligned}
&\|K(v_k)u_{k+1} - f\|^2 \\
=&\|K(v_k)(u_{k+1} - u_k)\|^2 + \|K(v_k)u_k - f\|^2 + \\
&2\langle K(v_k)(u_{k+1} - u_k), K(v_k)u_k - f\rangle \\
=&\|K(v_k)(u_{k+1} - u_k)\|^2 + \|K(v_k)u_k - f\|^2 - \\
&2\beta_k\left\langle K(v_k)M^{-1}(v_k)(K(v_k)u_k - f), K(v_k)u_k - f\right\rangle \\
\le&\|K(v_k)(u_{k+1} - u_k)\|^2 + (1 - 2\beta_k\frac{\underline{\rho}}{\bar{\rho}_M})\|K(v_k)u_k - f\|^2 \\
\le&(1 - 2\beta_k\frac{\underline{\rho}}{\bar{\rho}_M} + \beta_k^2\frac{\bar{\rho}^2}{\underline{\rho}_M^2})\|K(v_k)u_k - f\|^2. \qquad (19)
\end{aligned}$$

Next, we estimate an update due to v_k via the Young's inequality:

$$\begin{aligned}
&\|(K(v_{k+1}) - K(v_k))u_{k+1}\|\|K(v_k)u_{k+1} - f\| \\
\leq &\frac{1}{2p}\|K(v_k)u_{k+1} - f\|^2 + \frac{p}{2}\|(K(v_{k+1}) - K(v_k))u_{k+1}\|^2 \\
\leq &\frac{1}{2p}\|K(v_k)u_{k+1} - f\|^2 + \frac{pL_K^2\alpha_k^2}{8}\|u_k^T\frac{\partial K}{\partial v_k}u_k\|^2\|u_{k+1}\|^2.
\end{aligned}$$

Putting the above two equations together, we have:

$$\begin{aligned}
&\|K(v_{k+1})u_{k+1} - f\|^2 \\
\leq &\|(K(v_{k+1}) - K(v_k))u_{k+1}\|^2 + \|K(v_k)u_{k+1} - f\|^2 + \\
&2\|(K(v_{k+1}) - K(v_k))u_{k+1}\|\|K(v_k)u_{k+1} - f\| \\
\leq &\frac{p+1}{p}(1 - 2\beta_k\frac{\underline{\rho}}{\bar{\rho}_M} + \beta_k^2\frac{\bar{\rho}^2}{\underline{\rho}_M^2})\|K(v_k)u_k - f\|^2 + \\
&L_K^2\alpha_k^2\frac{p+1}{4}\|u_k^T\frac{\partial K}{\partial v_k}u_k\|^2\|u_{k+1}\|^2,
\end{aligned} \tag{20}$$

which is a recursive relationship to be used for proving the low-level convergence. □

8.2. Low-Level Convergence

We use the following shorthand notation for the result in Lemma 8.5:

$$\begin{aligned}
&\Xi_{k+1} \leq \Theta_k\Xi_k + \Gamma_k\|u_k\|^6 \\
&\Theta_k \triangleq \frac{p+1}{p}(1 - 2\beta_k\frac{\underline{\rho}}{\bar{\rho}_M} + \beta_k^2\frac{\bar{\rho}^2}{\underline{\rho}_M}) \quad \Gamma_k \triangleq L_K^2L_{\nabla K}^2\alpha_k^2\frac{p+1}{4}.
\end{aligned} \tag{21}$$

By taking Assumption 7.1 and direct calculation, we can ensure that $\Theta_k \leq \bar{\Theta} < 1$ (i.e., the first term is contractive). To bound the growth of the second term above, we show by induction that both Ξ_k and u_k can be uniformly bounded for all k via a sufficiently small, constant $\Gamma_k \leq \bar{\Gamma}$.

Lemma 8.6. *Assuming $M(v)$ commutes with $K(v)$, 3.1, 8.1, 8.2, we have $\Xi_k \leq U^2, \|u_k\| \leq U_f + U$ for all $k \geq 1$, where U_f is the uniform upper bound for $\|u_f(v)\|$.*

Proof. First, since $v \in X$ and X is compact, we have $\|u_f(v)\| \leq U_f < \infty$. Next, we prove $\Xi_k \leq U^2$ by induction. We already have $\Xi_1 \leq U^2$. Now

suppose $\Xi_k \le U^2$, then Equation 21 and our assumption on $\bar{\Gamma}$ immediately leads to:

$$\begin{aligned}\Xi_{k+1} \le & \bar{\Theta}U^2 + \bar{\Gamma}(\|K^{-1}(v_k)(K(v_k)u_k - f)\| + \|u_f(v_k)\|)^6 \\ \le & \bar{\Theta}U^2 + \bar{\Gamma}(U/\underline{\rho} + U_f)^6 \le U^2.\end{aligned}$$

Finally, we have: $\|u_k\| \le \|u_k - u_f(v_k)\| + \|u_f(v_k)\| \le U + U_f$ and our lemma follows. □

The shrinking coefficient $\bar{\Theta}$ and the uniform boundedness of Ξ_k, u_k allows us to establish low-level convergence.

Theorem 8.7. *Taking Assumption 3.1, 8.1, 8.2, 8.3, we can upper bound Ξ_{k+1} as:*

$$\Xi_{k+1} \le \bar{\Theta}^{k-1}\Xi_1 + (U + U_f)^6\Gamma_1 \sum_{i=0}^{k-1} \frac{\bar{\Theta}^i}{(k-i)^{2m}} = \mathcal{O}(k^{1-2m}).$$

Proof. Recursively expand on Equation 13 and we have:

$$\begin{aligned}\Xi_{k+1} \le & \bar{\Theta}^k\Xi_1 + (U + U_f)^6 \sum_{i=0}^{k-1} \bar{\Theta}^i\Gamma_{k-i} \\ \le & \bar{\Theta}^k\Xi_1 + (U + U_f)^6\Gamma_1 \left[\frac{\lceil \frac{k+1}{2} \rceil}{(k - \lceil \frac{k-1}{2} \rceil)^{2m}} + \frac{\bar{\Theta}^{\lceil \frac{k}{2} \rceil}}{1 - \bar{\Theta}} \right] = \mathcal{O}(k^{1-2m}),\end{aligned}$$

where we have used our choice of α_k, Lemma 8.6, and a similar argument as in Theorem 7.8. □

Theorem 8.7 allows us to choose m and tune the convergence speed of the low-level problem, which is used to establish the convergence of high-level problem.

8.3. High-Level Error Propagation

We first establish the high-level rule of error propagation over a single iteration:

Lemma 8.8. *Assuming $M(v)$ commutes with $K(v)$, 3.1, 7.1, 7.2, 7.3, the following high-level error propagation rule holds for all $k \ge 1, q > 0$:*

$$\begin{aligned}& l(v_{k+1}) - l(v_k) \le -\frac{1}{\alpha_k}\|v_{k+1} - v_k\|^2 + \\ & \frac{L_{\nabla K}^2\alpha_k^2(L_{\nabla l}q + 1)}{8q}(U + U_f)^4 + \frac{L_{\nabla K}^2 q}{8\underline{\rho}}\Xi_k(U + 2U_f)^2.\end{aligned}$$

Proof. Equation 14 holds by the same argument as in Lemma 7.9. For the last term in Equation 14, we have:

$$\begin{aligned}
&\left\langle v_{k+1} - v_k, \frac{1}{2} u_k^T \frac{\partial K}{\partial v_k} u_k - \frac{1}{2} u_f^T(v_k) \frac{\partial K}{\partial v_k} u_f(v_k) \right\rangle \\
\leq &\frac{L_{\nabla K}^2 \alpha_k^2}{8q} \|u_k\|^4 + \frac{q}{2} \left\| \frac{1}{2} (u_k - u_f(v_k))^T \frac{\partial K}{\partial v_k} (u_k + u_f(v_k)) \right\|^2 \\
\leq &\frac{L_{\nabla K}^2 \alpha_k^2}{8q} \|u_k\|^4 + \frac{L_{\nabla K}^2 q}{8\underline{\rho}} \Xi_k \|u_k + u_f(v_k)\|^2. \qquad (22)
\end{aligned}$$

The lemma follows by combining Equation 14, Equation 22, and Lemma 8.6. □

8.4. High-Level Convergence

We first show that Ξ_k^v is diminishing via the follow lemma:

Lemma 8.9. *Assuming $M(v)$ commutes with $K(v)$, 3.1, 8.1, 8.2, 8.3, we have the following bound on the accumulated high-level error Δ_k^v:*

$$\begin{aligned}
&\frac{1}{2} \sum_{i=1}^{k} \alpha_i \Delta_i^v \leq l(v_1) - \underline{l} + \\
&\frac{L_{\nabla K}^2}{8} (U + U_f)^4 \sum_{i=1}^{k} \alpha_i^{2-n} (L_{\nabla l} \alpha_i^n + 1) + \\
&\frac{L_{\nabla K}^2}{8\underline{\rho}} (U + 2U_f)^2 \sum_{i=1}^{k} \alpha_i^n \Xi_i + \frac{L_{\nabla K}^2}{4\underline{\rho}} (U + 2U_f)^2 \sum_{i=1}^{k} \alpha_i \Xi_i,
\end{aligned}$$

where $\underline{l}$ is the lower bound of l on X.

Proof. By choosing $q = \alpha_k^n$ for some constant n and summing up the recursive rule of Lemma 8.8, we have the following result:

$$\begin{aligned}
&l(v_{k+1}) - l(v_1) \leq -\sum_{i=1}^{k} \frac{1}{\alpha_i} \|v_{i+1} - v_i\|^2 + \\
&\frac{L_{\nabla K}^2}{8} (U + U_f)^4 \sum_{i=1}^{k} \alpha_i^{2-n} (L_{\nabla l} \alpha_i^n + 1) + \\
&\frac{L_{\nabla K}^2}{8\underline{\rho}} (U + 2U_f)^2 \sum_{i=1}^{k} \alpha_i^n \Xi_i. \qquad (23)
\end{aligned}$$

We further estimate the update using true gradient:

$$\Delta_k^v \le \frac{2}{\alpha_k^2}\|v_{k+1}-v_k\|^2 + \frac{L_{\nabla K}^2}{2\underline{\rho}}\Xi_k(U+2U_f)^2. \tag{24}$$

We can prove our lemma by combining Lemma 8.8, Equation 23, and Equation 24. □

The following corollary can be proved by the same argument as Corollary 7.11:

Corollary 8.10. *Assuming $M(v)$ commutes with $K(v)$, 3.1, 8.1, 8.2, 8.3, 7.4, we have $\min_{i=1,\cdots,k}\alpha_i\Xi_i^v \le Ck^{-1}$ and there are infinitely many k such that $\Xi_k^v \le C_v k^{m-1}$ for some constants C, C_v independent of k.*

Finally, we list necessary changes to prove Theorem 8.4:

Proof of Theorem 8.4. We choose $\tilde{v}_{k+1}, d_k, \tau_k$ as in proof of Theorem 7.5. Applying Corollary 8.10 and we can choose sufficiently large k such that:

$$\begin{aligned}
&-\left\langle \nabla l(\tilde{v}_{k+1}), \frac{\tau_k d_k}{\tau_k\|d_k\|}\right\rangle \le \frac{1}{\alpha_k}\|v_k-\tilde{v}_{k+1}\| \\
\le&\frac{1}{\alpha_k}\|v_k-v_{k+1}\| + \frac{1}{\alpha_k}\|v_{k+1}-\tilde{v}_{k+1}\| \\
=&\sqrt{\Delta_k^v} + \frac{1}{\alpha_k}\|v_{k+1}-\tilde{v}_{k+1}\| \\
\le&\frac{\epsilon}{4} + \sqrt{\frac{L_{\nabla K}^2}{4\underline{\rho}}\Xi_k\|u_k+u_f(v_k)\|^2} = \frac{\epsilon}{4} + \mathcal{O}(k^{(1-2m)/2}).
\end{aligned}$$

The last inequality holds by the contractive property of projection operator, Theorem 8.7, and Lemma 8.8. □

8.5. Parameter Choices

We argue that $\beta_k = 1$ is a valid choice in Algorithm 3 if a Krylov-preconditioner is used. Note that choosing $\beta_k = 1$ is generally incompatible with Assumption 8.1, which is because we make minimal assumption on the preconditioner $M(v)$, only requiring it to be commuting with $K(v)$ and positive definite with bounded spectrum. However, many practical preconditioners can provide much stronger guarantee leading to $\beta_k = 1$ being a

valid choice. To see this, we observe that the purpose of choosing β_k according to Assumption 8.1 is to establish the following contractive property in Equation 19:

$$\|K(v_k)u_{k+1} - f\|^2 \leq \bar{\Theta}\|K(v_k)u_k - f\|^2. \tag{25}$$

Any preconditioner can be used if they pertain such property with $\beta_k = 1$. The following result shows that Krylov-preconditioner pertains the property:

Lemma 8.11. *Suppose $M(v)$ is the Krylov-preconditioner with positive D and $\beta_k = 1$, then there exists some constant $\bar{\Theta} < 1$ where Equation 25 holds in Algorithm 3 for any k.*

Proof. We define $b \triangleq K(v_k)u_k - f$ and, by the definition of $M(v)$, we have:

$$\begin{aligned}
&\|K(v_k)u_{k+1} - f\|^2 = \|K(v_k)(u_k - M^{-1}(v_k)b) - f\|^2 \\
=&\|b - K(v_k)M^{-1}(v_k)b\|^2 = \|b - \sum_{i=1}^{D+1} c_i^* K^i(v_k)b\|^2 \\
\leq&\|(I - \frac{1}{\bar{\rho}}K(v_k))b\|^2 \leq (1 - \frac{\rho}{\bar{\rho}})\|b\|^2 \triangleq \bar{\Theta}\|K(v_k)u_k - f\|^2,
\end{aligned}$$

where the first inequality holds by the definition of c_i^*. □

Unfortunately, it is very difficult to theoretically establish this property for other practical preconditioners, such as incomplete LU and multigrid, although it is almost always observed in practice.

References

[1] S Ivvan Valdez, Salvador Botello, Miguel A Ochoa, José L Marroquín, and Victor Cardoso. Topology optimization benchmarks in 2d: Results for minimum compliance and minimum volume in planar stress problems. *Archives of Computational Methods in Engineering*, 24(4): 803–839, 2017.

[2] Jun Wu, Christian Dick, and Rüdiger Westermann. A system for high-resolution topology optimization. *IEEE Transactions on Visualization and Computer Graphics*, 22(3):1195–1208, 2016. doi: 10.1109/TVCG.2015.2502588.

[3] Sandilya Kambampati, Carolina Jauregui, Ken Museth, and H Alicia Kim. Large-scale level set topology optimization for elasticity and heat conduction. *Structural and Multidisciplinary Optimization*, 61(1):19–38, 2020.

[4] Ole Sigmund and Kurt Maute. Topology optimization approaches. *Structural and Multidisciplinary Optimization*, 48(6):1031–1055, 2013.

[5] Jun Wu, Lou Kramer, and Rüdiger Westermann. Shape interior modeling and mass property optimization using ray-reps. *Computers & Graphics*, 58:66–72, 2016.

[6] Moritz Bächer, Emily Whiting, Bernd Bickel, and Olga Sorkine-Hornung. Spin-it: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph.*, 33(4), 07 2014. ISSN 0730-0301. doi: 10.1145/2601097.2601157. URL https://doi.org/10.1145/2601097.2601157.

[7] Jun Wu, Anders Clausen, and Ole Sigmund. Minimum compliance topology optimization of shell–infill composites for additive manufacturing. *Computer Methods in Applied Mechanics and Engineering*, 326:358–375, 2017. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2017.08.018. URL https://www.sciencedirect.com/science/article/pii/S0045782517305984.

[8] Thanh T. Banh and Dongkyu Lee. Topology optimization of multi-directional variable thickness thin plate with multiple materials. *Struct. Multidiscip. Optim.*, 59(5):1503–1520, 05 2019. ISSN 1615-147X. doi: 10.1007/s00158-018-2143-8. URL https://doi.org/10.1007/s00158-018-2143-8.

[9] Jesús Martínez-Frutos, Pedro J. Martínez-Castejón, and David Herrero-Pérez. Efficient topology optimization using gpu computing with multi-level granularity. *Advances in Engineering Software*, 106:47–62, 2017. ISSN 0965-9978. doi: https://doi.org/10.1016/j.advengsoft.2017.01.009. URL https://www.sciencedirect.com/science/article/pii/S0965997816302332.

[10] A Mahdavi, R Balaji, M Frecker, and Eric M Mockensturm. Topology optimization of 2d continua for minimum compliance using parallel

computing. *Structural and Multidisciplinary Optimization*, 32(2):121–132, 2006.

[11] Niels Aage, Erik Andreassen, and Boyan Stefanov Lazarov. Topology optimization using petsc: An easy-to-use, fully parallel, open source topology optimization framework. *Structural and Multidisciplinary Optimization*, 51(3):565–572, 2015.

[12] Ole Sigmund. A 99 line topology optimization code written in matlab. *Structural and multidisciplinary optimization*, 21(2):120–127, 2001.

[13] Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. Narrow-band topology optimization on a sparsely populated grid. *ACM Trans. Graph.*, 37(6), 12 2018. ISSN 0730-0301. doi: 10.1145/3272127.3275012. URL https://doi.org/10.1145/3272127.3275012.

[14] Saeed Ghadimi and Mengdi Wang. Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*, 2018.

[15] Sébastien Briot and Alexandre Goldsztejn. Topology optimization of industrial robots: Application to a five-bar mechanism. *Mechanism and Machine Theory*, 120:30–56, 2018.

[16] Lukas Krischer, Anand Vazhapilli Sureshbabu, and Markus Zimmermann. Modular topology optimization of a humanoid arm. In *2020 3rd International Conference on Control and Robots (ICCR)*, pages 65–72, 2020. doi: 10.1109/ICCR51572.2020.9344316.

[17] Naman Jain and Rakesh Saxena. Effect of self-weight on topological optimization of static loading structures. *Alexandria Engineering Journal*, 57(2):527–535, 2018. ISSN 1110-0168. doi: https://doi.org/10.1016/j.aej.2017.01.006. URL https://www.sciencedirect.com/science/article/pii/S111001681730008X.

[18] Thomas Buhl, Claus BW Pedersen, and Ole Sigmund. Stiffness design of geometrically nonlinear structures using topology optimization. *Structural and Multidisciplinary Optimization*, 19(2):93–104, 2000.

[19] Tyler E Bruns and Daniel A Tortorelli. Topology optimization of nonlinear elastic structures and compliant mechanisms. *Computer methods in applied mechanics and engineering*, 190(26-27):3443–3459, 2001.

[20] Erik Holmberg, Bo Torstenfelt, and Anders Klarbring. Stress constrained topology optimization. *Structural and Multidisciplinary Optimization*, 48(1):33–47, 2013.

[21] Liansen Sha, Andi Lin, Xinqiao Zhao, and Shaolong Kuang. A topology optimization method of robot lightweight design based on the finite element model of assembly and its applications. *Science Progress*, 103(3):0036850420936482, 2020. doi: 10.1177/0036850420936482. URL https://doi.org/10.1177/0036850420936482. PMID: 32609583.

[22] Martin P Bendsøe and Ole Sigmund. *Optimization of structural topology, shape, and material*, volume 414. Springer, 1995.

[23] Oded Amir, Mathias Stolpe, and Ole Sigmund. Efficient use of iterative solvers in nested topology optimization. *Structural and Multidisciplinary Optimization*, 42(1):55–72, 2010.

[24] Oded Amir. Efficient stress-constrained topology optimization using inexact design sensitivities. *International Journal for Numerical Methods in Engineering*, 122(13):3241–3272, 2021.

[25] Oded Amir, Niels Aage, and Boyan S Lazarov. On multigrid-cg for efficient topology optimization. *Structural and Multidisciplinary Optimization*, 49(5):815–829, 2014.

[26] Susana Rojas-Labanda and Mathias Stolpe. Benchmarking optimization solvers for structural topology optimization. *Structural and Multidisciplinary Optimization*, 52(3):527–547, 2015.

[27] Gustavo Assis da Silva and André Teófilo Beck. Reliability-based topology optimization of continuum structures subject to local stress constraints. *Structural and Multidisciplinary Optimization*, 57(6):2339–2355, 2018.

[28] Krister Svanberg. The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987. doi: https://doi.org/10.1002/nme.1620240207. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.1620240207.

[29] Youngsoo Choi, Geoffrey Oxberry, Daniel White, and Trenton Kirchdoerfer. Accelerating design optimization using reduced order models. *arXiv preprint arXiv:1909.11320*, 2019.

[30] Benoît Colson, Patrice Marcotte, and Gilles Savard. Bilevel programming: A survey. *4or*, 3(2):87–107, 2005.

[31] Claude Fleury. Mathematical programming methods for constrained optimization: dual methods. *Progress in astronautics and aeronautics*, 150:123–123, 1993.

[32] Michal Kočvara. Topology optimization with displacement constraints: a bilevel programming approach. *Structural optimization*, 14(4):256–263, 1997.

[33] Susana Rojas-Labanda and Mathias Stolpe. An efficient second-order sqp method for structural topology optimization. *Structural and Multidisciplinary Optimization*, 53(6):1315–1333, 2016.

[34] Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

[35] Prashant Khanduri, Siliang Zeng, Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. A near-optimal algorithm for stochastic bilevel optimization via double-momentum. *Advances in neural information processing systems*, 34, 2021.

[36] James K Guest, Jean H Prévost, and Ted Belytschko. Achieving minimum length scale in topology optimization using nodal design variables and projection functions. *International journal for numerical methods in engineering*, 61(2):238–254, 2004.

[37] Krister Svanberg. Mma and gcmma, versions september 2007. *Optimization and Systems Theory*, 104, 2007.

[38] Paul H Calamai and Jorge J Moré. Projected gradient methods for linearly constrained problems. *Mathematical programming*, 39(1):93–116, 1987.

[39] David G Luenberger, Yinyu Ye, et al. *Linear and nonlinear programming*, volume 2. Springer, 1984.

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

[41] Thomas Borrvall and Joakim Petersson. Large-scale topology optimization in 3d using parallel computing. *Computer methods in applied mechanics and engineering*, 190(46-47):6201–6229, 2001.

[42] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[43] Laurent Condat. Fast projection onto the simplex and the l1-ball. *Mathematical Programming*, 158(1):575–585, 2016.

[44] Jason Sanders and Edward Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[45] Nathan Bell and Michael Garland. Efficient sparse matrix-vector multiplication on cuda. Technical report, Citeseer, 2008.

[46] Junjie Yang, Kaiyi Ji, and Yingbin Liang. Provably faster algorithms for bilevel optimization. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 13670–13682. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/file/71cc107d2e0408e60a3d3c44f47507bd-Paper.pdf.

[47] Mingdong Zhou, Boyan S Lazarov, Fengwen Wang, and Ole Sigmund. Minimum length scale in topology optimization by geometric constraints. *Computer Methods in Applied Mechanics and Engineering*, 293: 266–282, 2015.

[48] Linus Hägg and Eddie Wadbro. On minimum length scale control in density based topology optimization. *Structural and Multidisciplinary Optimization*, 58(3):1015–1032, 2018.

Table 3: Table of Symbols

Variable	Definition
$x \in \Omega$	a point in material domain
u	displacement field
P	internal potential energy
k_e	stiffness tensor field
v, v_e	infill level, infill level of eth element
u_k, v_k, δ_k	solutions at kth iteration
$\underline{v}, \bar{v}$	infill level bounds
Ω_e	sub-domain of eth element
E, N	number of elements, nodes
K, K_e	stiffness matrix, stiffness of eth element
f	external force field
P_f	total energy
e_i	unit vector at ith element
u_f	displacement caused by force f
l	induced internal potential energy
η	power law coefficient
$\rho(\bullet)$	eigenvalue function
$\underline{\rho}, \bar{\rho}$	eigenvalue bounds
$\mathbb{1}, \mathbb{0}$	all-one, all-zero vectors
$\mathcal{A}_e$	activation function of eth element
$\mathbb{S}$	symmetry mapping matrix
v_s	infill levels of the left-half material block
$\#$	number of sub-components
$\bar{v}^i$	total infill level of ith component
$\mathbb{C}$	material filter operator
α_k, β_k	high-level, low-level step size
Δ_k, Δ_k^v	low-level, high-level error metrics
Ξ_k	low-level error metric used to analyze Algorithm 3
U	upper bound of $\sqrt{\Delta_1}$
Γ_k, Θ_k	coefficients of low-level error
$\bar{\Gamma}, \bar{\Theta}$	upper bounds of Γ_k, Θ_k
p, m, n	algorithmic constants
$L_u, L_K L_{\Delta K}, L_{\nabla l}$	L-constants of $u_f, K, \Delta K, \nabla l$
C, C_v	coefficients of reduction rate of Δ_k^v
$\mathcal{T}_X$	tangent cone of X
d_k, τ_k	feasible direction in $\mathcal{T}_X$ and step size
$\underline{l}$	lower bound of l
U_f	uniform upper bound of $\|u_f\|$
∇_X	projected gradient into normal cone
ϵ	error tolerance of gradient norm
$\mathrm{Proj}_X(\bullet)$	projection operator onto X
κ	condition number
M	preconditioner
$\underline{\rho}_M, \bar{\rho}_M$	eigenvalue bounds of preconditioner
c_i	coefficients of Krylov vectors
D	dimension of Krylov subspace
$\tilde{\nabla} l$	approximate gradient
$\tilde{\nabla}_P l$	mean-projected approximate gradient
$\lambda_{\mathbb{1}}, \lambda_{\underline{v}}$	Lagrangian multiplier for projection problem