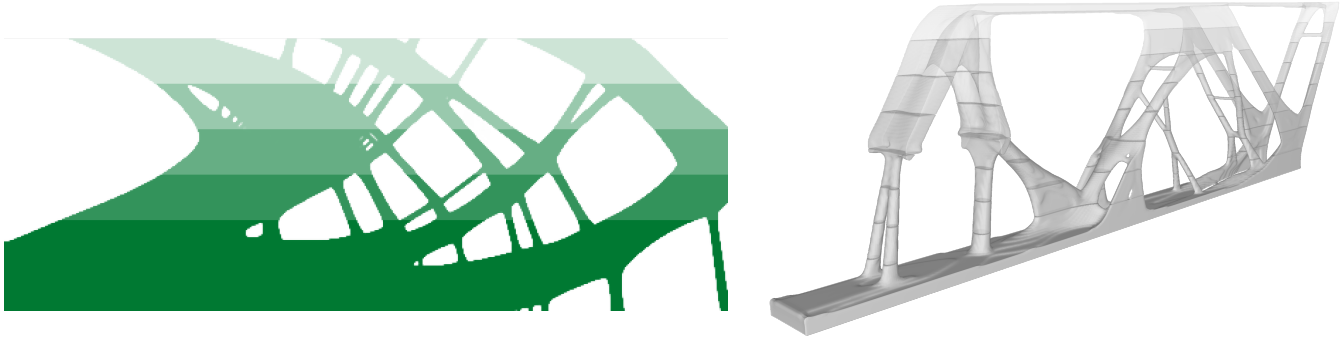


# Efficient Layer-by-Layer Simulation for Topology Optimization

Weixian Lan  
wlan@ucdavis.edu  
University of California, Davis  
Davis, CA, United States

Julian Panetta  
jpanetta@ucdavis.edu  
University of California, Davis  
Davis, CA, United States



**Figure 1: Our custom-tuned high-performance cut-cell Multigrid Preconditioned Conjugate Gradient (MGPCG) solver combined with a fast subspace initialization enables a tractable voxel-level layer-by-layer physics-based objective term for promoting robustly manufacturable designs at high resolution.**

## ABSTRACT

Topology optimization and additive manufacturing together enable the optimal design and direct fabrication of complex geometric parts with groundbreaking performance for diverse applications. However constraining the optimization to ensure that the generated object can be reliably manufactured via layer-by-layer 3D printing processes is challenging. The typical solution is to enforce design rules based only on geometric heuristics like overhang angles, minimum wall widths, and maximum bridge spans. Recent work has proposed instead to simulate the robustness of each partial object generated from bottom-to-top during the fabrication process as a more accurate, physics-aware printability assessment. However, this approach comes at the cost of an vast increase in the number of simulations run per design iteration, making existing implementations intractable at high resolution. We demonstrate that by developing a custom solver leveraging the close relationships between these many simulations, even voxel-level layer-by-layer simulations are feasible to incorporate into high-resolution 2D and 3D topology optimization problems on a single workstation.

## KEYWORDS

Digital Fabrication, Topology Optimization, Multigrid Preconditioners, Model Reduction

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

SCF '22, October 26–28, 2022, Seattle, WA, USA

© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-9872-5/22/10...\$15.00

<https://doi.org/10.1145/3559400.3562000>

## ACM Reference Format:

Weixian Lan and Julian Panetta. 2022. Efficient Layer-by-Layer Simulation for Topology Optimization. In *Symposium on Computational Fabrication (SCF '22)*, October 26–28, 2022, Seattle, WA, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3559400.3562000>

## 1 INTRODUCTION

Digital fabrication and additive manufacturing in particular promise to revolutionize designs across a broad range of application domains. The primary limiting factor is the tools we have at our disposal for discovering and optimizing new classes of designs. Arguably the most powerful and general of such tools is *topology optimization* which gives an optimization algorithm free reign to construct novel high-performance designs from scratch. The downside is that, without carefully constraining its exploration, topology optimization is prone to generate designs that are not manufacturable by our fabrication technologies—at least not without a post-processing stage in which an expert engineer manually reinterprets the design, potentially sacrificing some of its optimized performance.

Numerous efforts have been made over the years to develop constraints that ensure objects can be sent directly to the fabrication machine without heavy post-processing. These efforts typically center around enforcing geometric proxies for printability like minimum feature thicknesses and overhang angles. However, what ultimately matters is whether the part can survive the printing process and emerge as a faithful manifestation of the generated design. We believe that ideal solution is to directly analyze each intermediate stage of the fabrication process via simulation, and in this paper we consider the layer-by-layer fabrication of additive manufacturing. Past work [Allaire et al. 2017; Haverth et al. 2022] has proposed to model this process by running a large number of simulations of partially printed parts, and in our paper, we show that

with several new insights and algorithms, this is actually feasible to do at realistic resolutions.

Specifically, our work makes the following contributions:

- An efficient cut-cell multigrid preconditioned conjugate gradient (MGPCG) solver to support the continually changing size of the simulation grid with neither the overhead of rebuilding the MG hierarchy nor the implementation complexity of supporting non-power-of-two grid coarsening.
- An inexpensive strategy for generating high-quality initializations for the MGPCG solve.
- An investigation of how cost and effectiveness of the layer-by-layer objective scale with solver parameters like force residual tolerance and downsampling levels, showing the feasibility of layer-by-layer simulation for realistic-scale design problems in terms of both time and space complexity.

We emphasize that although our validation and analyses in Section 5 are performed by incorporating our term in a traditional compliance minimization design problem, the term can be included to enforce robust printability in *any* topology optimization design problem (e.g., topology optimization of metamaterials [Panetta et al. 2015; Schumacher et al. 2015], compliant mechanisms [Nishiwaki et al. 1998], heat sinks [Yan et al. 2019] and fluidic devices [Du et al. 2020]). We are releasing our high-performance implementation as open source, and we hope that it will be a valuable asset for fabrication-aware multidisciplinary topology optimization.

## 2 RELATED WORK

*Density and level set-based topology optimization.* Topology optimization for performance metrics governed by elasticity and other physics is a broad area encompassing a vast body of related work. Most competitive approaches to topology optimization can be broadly categorized into two general frameworks: density-based [Bendsoe and Sigmund 2003] and level-set [Van Dijk et al. 2013]. Density-based methods divide the design domain into a regular grid of voxels and introduce density variables controlling which material (if any) fills each voxel. Level-set topology optimization instead optimizes for one or more level-set functions defined over the domain whose zero contours represent the boundary of the solid regions and the interfaces between materials. For density-based methods, it is most natural and common to solve the state equations (i.e., the physics PDE constraints) over the same regular grid mesh introduced to define the density field. Level-set methods can solve the state equations either by rasterizing the sublevel sets to a regular grid mesh (the “ersatz material approach” [Allaire et al. 2004]) or by extracting a conforming unstructured finite element mesh of them [Dapogny et al. 2014]. Our solver can be used for either framework, provided a regular grid mesh is used for the solve, though our validations are performed in the density-based framework.

*Multigrid Solvers in Topology Optimization.* The extreme resolutions and regular structure of the rectangular grid meshes employed in density-based topology optimization make geometric multigrid solvers in general, and the multigrid preconditioned conjugate gradient method in particular, an excellent choice. Recent instances of this line of work in the computer graphics and computational fabrication are a GPU-accelerated solver developed by Wu et al.

[2016] and a narrow-band solver employing a sparse grid data structure developed by Liu et al. [2018]. We choose to implement our high-performance solver on the CPU rather than the GPU for improved memory capacity and implementation flexibility. Likewise, we choose to work with a simple regular grid data structure to simplify our implementations. We note that the narrow-band approach cannot provide gains at the initial stages of topology optimization from a completely agnostic uniform-density initialization (the peak memory bottleneck). Amir et al. [2014] provides an insightful discussion of the influence of contrast ratio on the convergence of MGPGC solvers, an issue we have noticed in our experiments, especially with the large number of void voxels that appear when masking out yet-to-be-printed layers, which we address in Section 4.1.1. We comment more on this issue in Section 6.

*AM Constraints in Topology Optimization.* For lack of a computationally tractable simulation-based approach to promoting manufacturability, many heuristic geometric-based approaches have been developed, aiming at enforcing certain design rules like minimum thickness constraints and overhang angles. In density-based methods, minimum thickness constraints are typically handled using a filtering approach [Wang et al. 2011], and several overhang constraint formulations have been developed [Gaynor and Guest 2016; Langelaar 2017; Qian 2017], though these are approximate, require parameter tuning, and can exhibit poor convergence. In the level-set framework, Allaire et al. [2016] propose and analyze several formulations for minimum thickness constraints with various trade-offs and implementation challenges; none offers a perfect solution to this fundamentally difficult problem of enforcing such a non-local constraint. These constraint formulations also do not admit a topological derivative (needed to nucleate new holes).

*Layer-By-Layer Objectives.* Our work was inspired by Allaire et al. [2017], who introduced the idea of layer-by-layer simulation in the level-set topology optimization framework. Their work reported *hundreds* of hours spent optimizing a  $300 \times 100$  2D example, though explicitly mentions no efforts were made to accelerate the implementation. The strategy they propose to reduce the computation time is to evaluate not only the partial object compliance but also *its derivative with respect to printing height* of a smaller number of intermediate shapes so that a first-order approximation can be constructed. This scheme is still slow without the sorts of optimizations we contribute (around 80 hours for the same 2D example) and risks poor approximation: we note that compliances of unprintable designs in particular often change discontinuously as a function of layer height, an effect that cannot be captured by even a higher order Taylor expansion-based strategy. Concurrently to our work, Haveroth et al. [2022] have brought the layer-by-layer self-weight compliance idea to the density-based topology optimization framework. They again do not pursue the sorts of accelerations we contribute (using a direct solver in 2D and an off-the-shelf PCG solver in 3D) and instead focus on the layer skipping strategy without a first-order approximation.

*Staged Fabrication Objectives.* The layer-by-layer fabrication process of 3D printing can be generalized to any multi-stage process that fabricates the object by attaching one (potentially curved) piece at a time. Amir and Mass [2018] perform topology optimization in

a design domain that has been manually partitioned into a number of ordered stages. Wang et al. [2020] extend this idea by simultaneously optimizing the segmentation and fabrication sequence of the design along with its shape and topology. Neither of these works attempts to accelerate the per-stage simulation problems, and the expense of the solves restricts them to consider a limited number of discrete stages that can easily fail to resolve problematic sub-stage geometry. Extending our cut-cell solver and initialization strategies to address the curved segmentations employed in these formulations is an interesting direction for future work.

*Worst-Case Design and Stochastic Analysis.* One effective strategy for designing objects to be robust against uncertain loads (for instance, if loads the part will be subjected to during fabrication and shipment are difficult to predict) is to analyze and optimize them under worst-case load. The worst-case stress analysis problem was addressed in [Zhou et al. 2013] for general parts under hand manipulation, and the problem of optimal design under worst-case loads has been addressed for compliance by [Cherkaev and Cherkaev 2008], for stresses in periodic microstructures in [Panetta et al. 2017], and for the asymmetric stress-based yield criteria of concrete-type materials in [Schumacher et al. 2018]. When the worst-case scenario is unrealistically pessimistic, a *stochastic* approach like [Langlois et al. 2016] can be preferable that determines the likelihood of failure by running a large number of simulations with random initial conditions. However, all of these works consider the robustness of the fully formed part and cannot guarantee robustness of intermediate structures during fabrication. Considering how much more expensive these worst-case and stochastic formulations tend to be than simulation under known loads, developing a tractable layer-by-layer worst-case optimization would be an interesting but challenging future research direction.

*Simulating the Printing Process.* The layer-by-layer self-weight simulations we perform are intended as a rough physical model of the printing process. However, they are far from a precise, detailed simulation of the physics at play, which limits the predictive power of our robustness measure. More precise simulation necessarily must be heavily tailored to the specific targeted additive fabrication technology; for example, simulations of varying complexity have been developed for metal (L-PBF and DED) [Bayat et al. 2021], MultiJet [Kim et al. 2016], and extrusion [Faria et al. 2020; Oyinloye and Yoon 2021] printers. These more sophisticated and expensive simulations have not yet been incorporated in the inner loop of a topology optimization.

*Model Reduction.* Our initialization strategy can be viewed as employing model reduction [Sifakis and Barbic 2012] to approximately solve the simulation problem. This approach is widely popular in graphics [Barbič and James 2005; Treuille et al. 2006; Xu et al. 2015], particularly for interactive applications where speed matters more than accuracy. They have, however, found use in less obvious way for stress analysis in fabrication applications [Chen et al. 2016]. We employ model reduction as an *initialization* for a solver with directly tunable accuracy that is independent of the reduced basis. Traditional model reduction typically requires an expensive offline basis construction that is incompatible with our use case of accelerating solves on a constantly-changing design. We instead use a

natural, high-quality low-dimensional basis that comes essentially for free and supports an efficient construction of the reduced model equations via a recursive update rule.

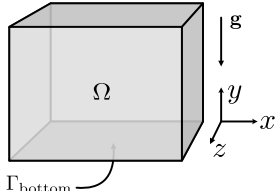
### 3 BACKGROUND

The printability regularization term that we seek to accelerate simulates deformation under self-weight of an object attached to the build platform during intermediate stages of the 3D printing process (when only part of the structure has been fabricated). We first review the standard topology optimization formulation for stiffening linearly elastic objects against deformations under self-weight and then introduce the layer-by-layer objective term.

#### 3.1 Self-Weight Compliance Minimization

Single-material density-based topology optimization represents the design in the form of a density field  $\rho : \Omega \rightarrow [0, 1]$  expressing the material occupancy at each point within the *design domain*  $\Omega$ . Solid regions are represented by  $\rho = 1$  and void by  $\rho = 0$ . The volume of material used is therefore  $V(\rho) := \int_{\Omega} \rho \, dx$ , and the self-weight force density is given by the vector field  $\rho \mathbf{g}$  where  $\mathbf{g}$  is the gravitational acceleration vector multiplied by the physical mass density of the fabrication material.

To evaluate stiffness of a candidate design, first the following linear elasticity PDE in  $d$ -dimensions is solved for the displacement field  $\mathbf{u} : \Omega \rightarrow \mathbb{R}^d$ :

$$\begin{aligned} -\nabla \cdot \boldsymbol{\sigma} &= \rho \mathbf{g} \text{ in } \Omega \\ \mathbf{u} &= 0 \text{ on } \Gamma_{\text{bottom}} \\ \boldsymbol{\sigma} \mathbf{n} &= 0 \text{ on } \partial\Omega \setminus \Gamma_{\text{bottom}} \\ \boldsymbol{\sigma} &:= C(\rho) : \underbrace{\frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^{\top})}_{\boldsymbol{\varepsilon}(\mathbf{u})}, \end{aligned} \quad (1)$$


and then the compliance  $\frac{1}{2} \int_{\Omega} \mathbf{u} \cdot \rho \mathbf{g} \, dx$  is evaluated. Notice that a Dirichlet constraint is used to glue the bottom of the design domain  $\Gamma_{\text{bottom}}$  to the build platform, while the rest of the faces are left traction free ( $\boldsymbol{\sigma} \mathbf{n} = 0$ ). In this equation,  $C(\rho)$  denotes the elasticity tensor of the material mixture associated with infill density  $\rho$ . Whereas density values  $\rho = 1$  and  $\rho = 0$  have obvious physical meaning as fully solid and void, the properties of intermediate density levels are typically defined using a heuristic *interpolation law*. We employ the RAMP interpolation law [Stolpe and Svanberg 2001] recommended in the literature for optimization under self-weight loads since the common SIMP interpolation law has been shown to be problematic in this setting [Bruyneel and Duysinx 2005]:

$$C(\rho) := s(\rho)C^{\text{base}}, \quad s(\rho) := s_{\min} + \frac{\rho(s_{\max} - s_{\min})}{1 + q(1 - \rho)}, \quad (2)$$

where  $C^{\text{base}} = C(1.0)$  is the elasticity tensor of the fully solid fabrication material that the scaling function  $s(\rho)$  attenuates according to the infill density. Notice that this interpolation law interpolates down to  $s(0) = s_{\min} > 0$  in the void regions; a small stiffness is retained so that the simulation remains well-posed (and a singular matrix is avoided in the discretized linear system). This contrast ratio parameter  $s_{\min}$  strongly influences the conditioning of the linear system as discussed in Section 4.1.1 and Section 6; we select  $s_{\min} = 10^{-5}$  for all of our benchmarking experiments.

We employ a standard finite element discretization on a regular grid of bilinear rectangle elements ( $d = 2$ ) or trilinear voxel elements ( $d = 3$ ). Denoting the element resolution along each dimension by  $N_x, N_y$ , and  $N_z$ , there are  $n_n = (N_x + 1)(N_y + 1)(N_z + 1)$  nodes and  $n_e = N_x N_y N_z$  elements in the 3D case. We obtain the discrete counterpart to (1),

$$\begin{aligned} K(\rho_{\text{fem}}) \mathbf{u}_{\text{fem}}(\rho_{\text{fem}}) &= \mathbf{f}_{\text{fem}}(\rho_{\text{fem}}), \\ K(\rho_{\text{fem}}) &:= \sum_e s(\rho_{\text{fem}}[e]) S_e^\top K_0 S_e, \\ \mathbf{f}_{\text{fem}}(\rho_{\text{fem}}) &:= \sum_e (\rho_{\text{fem}}[e]) S_e^\top \mathbf{f}_0. \end{aligned} \quad (3)$$

with discrete nodal displacement and load fields  $\mathbf{u}_{\text{fem}}, \mathbf{f}_{\text{fem}} \in \mathbb{R}^{dn_n}$  and element densities  $\rho_{\text{fem}} \in [0, 1]^{n_e}$ . Here  $S_e$  is a *selection* matrix that extracts the  $d \cdot 2^d$  displacement variables associated with the corners of element  $e$  from a nodal field vector; it is used here to assemble the per-element stiffness matrix and load contributions for each element  $e$  into the global sparse matrix  $K$  and load vector  $\mathbf{f}_{\text{fem}}$ . The precomputed quantities  $K_0 \in \mathbb{R}^{d2^d \times d2^d}$  and  $\mathbf{f}_0 \in \mathbb{R}^{d2^d}$  are the stiffness matrix and gravity load vector produced by a single full density *reference element*  $\mathcal{R}$  with vector-valued finite element basis functions  $\vec{\phi}_i$ :

$$\begin{aligned} [K_0]_{ij} &:= \int_{\mathcal{R}} \varepsilon(\vec{\phi}_i) : C^{\text{base}} : \varepsilon(\vec{\phi}_j) \, dx, \\ [\mathbf{f}_0]_i &:= \int_{\mathcal{R}} (\rho_{\text{fem}}[e]) \vec{\phi}_i \cdot \mathbf{g} \, dx. \end{aligned}$$

To enforce the  $\mathbf{u} = 0$  Dirichlet conditions, the associated rows and columns of  $K$  are replaced with the corresponding rows and columns of the  $dn_n \times dn_n$  identity matrix, and the associated entries of  $\mathbf{f}_{\text{fem}}$  are set to zero. After solving (3), we can evaluate the *self-weight compliance*:

$$J(\rho_{\text{fem}}) := \frac{1}{2} \mathbf{u}_{\text{fem}}(\rho_{\text{fem}}) \cdot \mathbf{f}_{\text{fem}}(\rho_{\text{fem}}).$$

Key to the efficiency of our method, the gradient of self-weight compliance  $\frac{\partial J}{\partial \rho_{\text{fem}}}$  can be evaluated *without solving an adjoint equation* due to the self-adjoint nature of the compliance objective:

$$\begin{aligned} \frac{\partial J}{\partial \rho_{\text{fem}}} &= \frac{1}{2} \frac{\partial \mathbf{f}}{\partial \rho_{\text{fem}}} \cdot \mathbf{u}_{\text{fem}} + \frac{1}{2} \underbrace{\mathbf{f}_{\text{fem}} \cdot K^{-1}}_{\mathbf{u}_{\text{fem}}} \left( \frac{\partial \mathbf{f}_{\text{fem}}}{\partial \rho_{\text{fem}}} - \frac{\partial K}{\partial \rho_{\text{fem}}} \mathbf{u}_{\text{fem}} \right) \\ &= \frac{\partial \mathbf{f}}{\partial \rho_{\text{fem}}} \cdot \mathbf{u}_{\text{fem}} - \frac{1}{2} \mathbf{u}_{\text{fem}} \cdot \frac{\partial K}{\partial \rho_{\text{fem}}} \mathbf{u}_{\text{fem}}. \end{aligned}$$

The required derivatives  $\frac{\partial \mathbf{f}}{\partial \rho_{\text{fem}}}$  and  $\frac{\partial K}{\partial \rho_{\text{fem}}}$  are straightforward to compute from (3) and yield the following efficient, explicit formula for the derivative with respect to the density of element  $e$  in terms of the element corner displacements  $\mathbf{u}_e = S_e \mathbf{u}_{\text{fem}}$ :

$$\left[ \frac{\partial J}{\partial \rho_{\text{fem}}} \right]_e = \mathbf{u}_e \cdot \mathbf{f}_0 - \frac{1}{2} s'(\rho_{\text{fem}}[e]) \mathbf{u}_e \cdot K_0 \mathbf{u}_e.$$

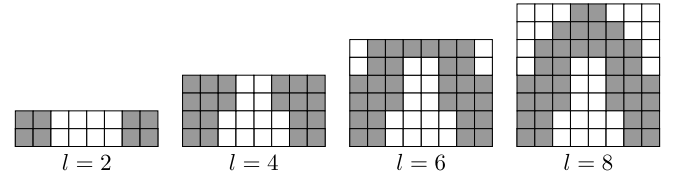
Directly minimizing compliance with respect to each element's density variable is notorious for producing nonphysical, mesh-dependent artifacts like artificially stiff checkerboard patterns [Sigmund and Petersson 1998]. We avoid this using the standard filtering approach: we express the simulation densities  $\rho_{\text{fem}}$  in terms of *design parameters*  $\mathbf{p}$  via a chain of filters:  $\rho_{\text{fem}} = \mathcal{F}(\mathbf{p})$ . Specifically,

we use a radial smoothing filter with linearly decaying coefficients [Bruns and Tortorelli 2001] to blur the design followed by a Heaviside projection filter to sharpen it. Derivatives with respect to  $\rho_{\text{fem}}$  are backpropagated through  $\mathcal{F}$  to obtain the optimization gradient  $\frac{\partial J(\mathcal{F}(\mathbf{p}))}{\partial \mathbf{p}}$ . We note that our method is fully compatible with alternative strategies for ensuring mesh-independence, such as using a neural design representation with direct control over the spatial frequency of the design [Doosti et al. 2021].

Putting everything together, the standard discrete formulation of compliance minimization seeks the stiffest structure under a given volume budget  $V_{\text{max}}$ :

$$\min_{\mathbf{p}} J(\mathcal{F}(\mathbf{p})) \quad \text{s.t.} \quad \sum_e \rho_{\text{fem}}[e] \text{Vol}(e) \leq V_{\text{max}}. \quad (4)$$

### 3.2 Layer-By-Layer Simulation



**Figure 2: Partial structures within the partial design domains  $\Omega^{(l)}$  fabricated at intermediate stages of the printing process ( $l$  completed layers).**

The layer-by-layer simulation objective term that our work accelerates builds on the discrete self-weight compliance  $J(\rho_{\text{fem}})$ . The underlying idea is that if any partial structure generated during the fabrication process exhibits weak or unprintable features (in the extreme, a detached solid region due to downward hanging geometry), this can be detected as significant drooping under self-weight that induces high compliance. Denoting by  $\Omega^{(l)}$  the partial design domain including only the bottommost  $l$  “layers,” we can restrict the design’s density field  $\rho_{\text{fem}}$  to  $\Omega^{(l)}$  and follow the previous section to compute its FEM load vector  $\mathbf{f}^{(l)}$ , nodal displacements  $\mathbf{u}^{(l)}$ , and finally compliance  $J^{(l)} := \frac{\mathbf{u}^{(l)} \cdot \mathbf{f}^{(l)}}{2}$ . Averaging the compliance over all  $L$  layer simulations obtains a global physics-based **Layer-By-Layer** measure,  $J_{\text{LBL}}$ , of the object’s robustness during fabrication:

$$J_{\text{LBL}}(\rho_{\text{fem}}) := \frac{1}{L} \sum_{l=1}^L J^{(l)}(\rho_{\text{fem}}).$$

Evaluating this objective term requires solving  $L$  elasticity simulations. We aim to evaluate this objective *at the voxel level* ( $L = N_y$ ), or as close to it as possible, so that no hanging features are missed by skipping layers. The vast number of simulations this entails necessitates a highly efficient solver and new algorithms that we develop in the next section.

## 4 METHOD

Our main insight is that the close relationships between the  $L$  simulation problems solved for each fabrication stage can be exploited to dramatically accelerate the solver. We first develop a highly efficient multigrid-preconditioned conjugate gradient (MGPCG) solver

tailored specifically to the layer-by-layer simulation setup (which necessarily involves operating on grids with non-power-of-two sizes along  $N_y$ ), described in Section 4.1. As an iterative method, our solver can benefit from good initial guesses, and our second contribution is an efficient method for constructing high-quality initializations by solving a reduced elasticity problem over the subspace spanned by a small number of previous layer simulations (Sections 4.2-4.3). Finally, we show how the gradient  $\frac{\partial J_{\text{BL}}}{\partial \rho_{\text{fem}}}$  can be evaluated quickly and with space complexity independent of  $L$  (Section 4.4).

Our approach is most effective when the simulation problems are solved *top-to-bottom*, in order of decreasing  $l$ : this way, updates to the linear systems are sparse, and information from the previous  $N$  solves on domains  $\Omega^{(l+N)} \supset \dots \supset \Omega^{(l+1)}$  is available on the *entirety* of domain  $\Omega^{(l)}$ .

#### 4.1 Multigrid Preconditioner

Our starting point is a geometric multigrid preconditioner similar to those developed in recent high-resolution topology optimization efforts [Liu et al. 2018; Wu et al. 2016]. Like Liu et al. [2018], we employ  $V$ -cycles with a  $d \times d$  block Gauss-Seidel smoother and carefully vectorize and parallelize all operations. We differ from both past works in using matrix-free matvecs and smoothers for not only the finest grid, but also the *second*-finest (performing the first level of coarsening on the fly); explicitly constructing the first coarsened sparse block stiffness matrix incurs a multifold increase in the solver’s memory requirements and degrades performance due the high memory bandwidth requirements in our experiments. We also implement *full multigrid* [Briggs et al. 2000], which we find accelerates the solves of the full design simulations, but disable for the subsequent partial object simulations. For partial simulations, we find the cheaper—though less effective—pure  $V$ -cycle preconditioner to strike a better trade-off. This is because, thanks to our high-quality initializations, our solver typically requires *fewer than one* PCG iteration per simulation on average to reduce the force residual below an acceptable tolerance, and this iteration count cannot be reduced much by employing a more effective preconditioner in any one solve.

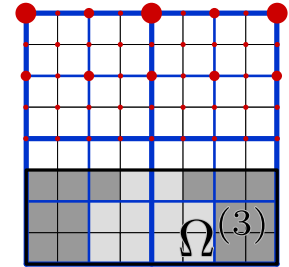
**4.1.1 Cut-Cell Preconditioner.** Our most significant deviation from past MGPCG solvers regards our handling of the constantly-changing non-power-of-two grid sizes along the build direction. The easiest approach would be simply to set to zero the entries of  $\rho_{\text{fem}}$  for elements above the  $l$  retained layers and solve over the full grid. Of course this involves wasted computation within each PCG iteration solving for the displacements of superfluous void nodes. But the problem actually is worse: the nearly-undefined displacements of nodes in the ultra-low stiffness void regions are difficult to predict by our initialization strategy, requiring more PCG iterations Section 5.1. Consequently, it is beneficial to restrict our solver to only the  $l$  layers of the grid that are actively simulated, which means solving on a grid whose vertical element resolution is arbitrary. The most principled approach would be support a mix of 2-to-1, 3-to-1, and possibly more coarsening rules, but this involves significant implementation complexity, especially if one wishes to avoid a full reconstruction of the multigrid hierarchy on every grid change (Section 4.1.2).

We instead propose a simple and efficient *cut-cell* approach that retains the excellent convergence rate of standard MGPCG. Conceptually, we set both  $\rho$  and the *elasticity tensor attenuation factors* ( $s(\rho)$  in (2)) to precisely zero for all elements above layer  $l$ , which we emphasize is different from setting entries of  $\rho_{\text{fem}}$  alone to zero.

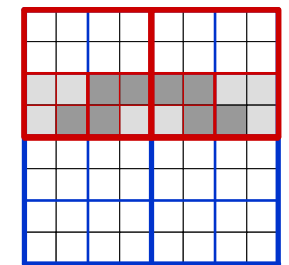
This introduces a high dimensional nullspace in  $K$  that is harmless for the CG algorithm itself: CG essentially ignores the nullspace, taking no steps within it and accumulating no residual from it. However, it breaks the Gauss-Seidel smoother for certain *detached* nodes in the multigrid hierarchy and leads to a singular system at the coarsest level that causes the direct solve (Cholesky factorization) employed there to fail. We correct this by identifying these detached nodes—nodes at any level within the MG hierarchy whose coarsened shape functions’ supports overlap only the zero-stiffness elements above layer  $l$ —and omitting them from smoothing and pinning their displacements to zero for the coarse Cholesky solve. Detached nodes at a given coarse grid in the hierarchy can be identified in constant time by checking whether the neighboring coarse node below is directly at or above the top border of the partial design domain  $\Omega^{(l)}$  (Figure 3).

For a  $\sim 2\times$  speedup, we also skip all other operations involving these detached nodes in the MGPCG algorithm (all matvecs, interpolation, restriction, dot products, etc.), effectively eliminating them from the system so that only the partial grid is solved. Up to a variable rescaling, our approach can be interpreted as the implementation of two different vertical coarsening schemes: the standard 2-to-1 scheme away from the layer and no-op 1-to-1 scheme in the vicinity of the boundary.

**4.1.2 Fast Coarsening Updates.** Re-computing the coarsened stiffness matrices is a bottleneck if done from scratch upon each layer removal. A moderate speedup can be gained for especially loose residual tolerances by updating lazily, only when the simulation needs at least one PCG iteration. However, full brute-force updates of the coarsening hierarchy are still costly with this strategy. Our implementation accelerates these MG hierarchy updates by leveraging the fact that actually only a small fraction of the coarsened stiffness matrices must change (those for elements overlapping the layer(s) that were removed since the last update). We maintain a record of the current layer height for



**Figure 3: Partially occupied coarsened cells in our multigrid hierarchy (blue). Detached nodes at the coarsest, second-coarsest, and fine level are visualized by overlapping, progressively smaller red circles.**



**Figure 4: We update stiffnesses matrices only for coarsened elements overlapping the modified layer(s), visualized in red for update interval [4, 6].**

which the coarsened stiffness matrices have been computed,  $l^{\text{current}}$ . The first time a PCG iteration is needed for the simulation at layer height  $l$ , we check if  $l < l^{\text{current}}$ , and if so perform a sparse update for layer interval  $[l, l^{\text{current}})$ , clearing the stiffnesses of the fine elements whose *zero-based* vertical index falls in that layer interval as visualized in Figure 4 and described in the next paragraph. If  $l = l^{\text{current}}$  we do nothing, and if  $l > l^{\text{current}}$  we do a full recomputation of all coarsened matrices (in our framework, this happens only when advancing to simulate a new design candidate, when  $l$  is reset to  $L$ ). Finally, we update  $l^{\text{current}} = l$ .

Our stiffness matrix coarsening is implemented recursively, which makes it especially easy to surgically update only the portions of the coarsened sparse block stiffness matrices cached at each level in the MG hierarchy that have been invalidated by removing stiffness from fine voxel layers  $[l, l^{\text{current}})$ . We iterate over the elements at the coarsest level that overlap the updated fine layers and call our sparse update routine on each. The sparse update routine is responsible for (a) computing the coarsened per-element stiffness matrix  $K_{\text{update}}$  due exclusively to the densities in layer update interval  $[l, l^{\text{current}})$  (i.e., pretending all other finest-level Young's moduli are identically zero); (b) *subtracting*  $K_{\text{update}}$  from the values stored in the hierarchy (accomplishing the required update); and (c) returning  $K_{\text{update}}$  to the next-coarser element that called it. The update routine achieves (a) for coarse element  $e_c$  by recursively calling itself for each finer element  $e_f$  nested within  $e_c$  that overlaps  $[l, l^{\text{current}})$ , coarsening the resulting stiffness matrices, and summing them together.

## 4.2 Subspace Initialization

As an iterative method, our MGPCG solver can benefit greatly from a good initial guess. One of our key insights is that, due to the small change in geometry made when advancing from  $\Omega^{(l+1)}$  to  $\Omega^{(l)}$ , we expect the nodal displacements under self-weight of the partial structure in  $\Omega^{(l)}$  to be close to those of the previously simulated structures in domains  $\Omega^{(l+1)}$ ,  $\Omega^{(l+2)}$ , and so on. Furthermore, we expect this similarity to strengthen as the grid is refined to higher resolutions.

Already simply taking the previous simulation result as the initial guess ( $\mathbf{u}_{\text{init}}^{(l)} = \mathbf{u}^{(l+1)}$ ) achieves a good speedup over the trivial initialization  $\mathbf{u}_{\text{init}}^{(l)} = \mathbf{0}$ ; we refer to these two initializations as “constant” and “zero” in Section 5.3. A more powerful way to leverage the valuable information obtained by the simulation of the previous partial structures ( $l+1, \dots, l+N$ ) is to construct the initial guess by solving the self-weight simulation problem in the subspace spanned by those previous  $N$  solutions. In other words, we seek to minimize the total potential energy of the simulation (the exact same energy that the subsequent CG iterations effectively minimize) over the span of those previous displacement fields, which we can collect into the columns of a matrix  $U^{(l)} := [\mathbf{u}^{(l+1)} \ \dots \ \mathbf{u}^{(l+N)}]$ . This amounts to solving the dense system:

$$\left[ U^{(l)} \right]^T K^{(l)} U^{(l)} \mathbf{c} = \left[ U^{(l)} \right]^T \mathbf{f}^{(l)}, \quad (5)$$

for coefficients  $\mathbf{c}$  and constructing the initial guess  $\mathbf{u}_{\text{init}}^{(l)} = U^{(l)} \mathbf{c}$ . In our experiments, only the most recent  $N \approx 4$  simulations provide useful information, so only a small upper-left block of this system

will be constructed and solved. However, to avoid the clutter of block indexing notation in Section 4.3, we derive formulas for the full system.

It turns out that for a uniform-density design and an isotropic material  $C^{\text{base}}$  with Poisson's ratio  $\nu = 0$ , our initial guess is actually *perfect* for  $N \geq 2$ . This follows from the fact that the analytical solution to (1) that can be obtained in this case—in both continuous and discrete settings—is a linear function of structure layer height  $l$ . Specifically, displacements parallel to the build platform are zero, and the displacements along the gravity direction in the continuous setting are given by  $u_y(x, y, z) = \frac{\|g\|}{Y} \left( \frac{1}{2}y^2 - ly \right)$ , where  $Y$  is the Young's modulus attenuated according to the density level. For nonzero Poisson's ratios and nonuniform densities, the guess is imperfect but still gains an appreciable speedup.

To initialize the self-weight simulation of the full design candidate, we use the full-structure displacement field  $\mathbf{u}^{(L)}$  computed for the previous design candidate (under the previous  $\rho_{\text{fem}}$ ).

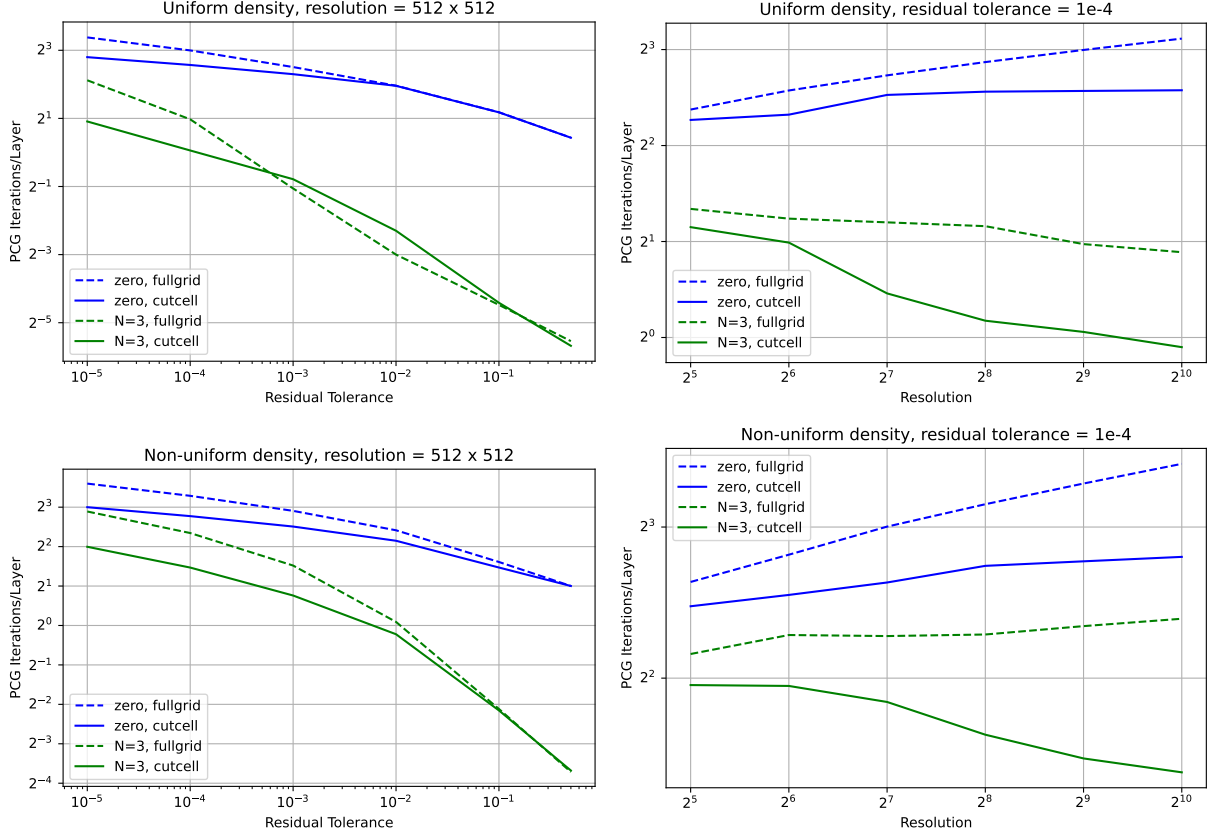
## 4.3 Fast Reduced System Construction

An initialization strategy is worthwhile only if it costs less than the PCG iterations it displaces. While it is straightforward to construct the reduced system matrix  $A^{(l)} := \left[ U^{(l)} \right]^T K^{(l)} U^{(l)}$  and right-hand-side vector  $\mathbf{b}^{(l)} := \left[ U^{(l)} \right]^T \mathbf{f}^{(l)}$  with applications of sparse matrix  $K^{(l)}$  and several dense linear algebra operations, for large  $N$ , these operations can exceed the cost of the PCG iterations saved. We however observe that the close relationship between the quantities of different partial structures can once again be exploited to efficiently build  $A^{(l)}$  and  $\mathbf{b}^{(l)}$ . In particular, the differences  $\tilde{K}^{(l)} := K^{(l)} - K^{(l+1)}$  and  $\tilde{\mathbf{f}}^{(l)} := \mathbf{f}^{(l)} - \mathbf{f}^{(l+1)}$  are both highly sparse, containing nonzeros only for entries corresponding to nodes of the voxels whose densities were voided by the removal of layer  $(l+1)$ . This enables an efficient recursive construction at significantly lower cost than one single PCG iteration:

$$\begin{aligned} \mathbf{b}^{(l)} &= \left[ U^{(l)} \right]^T \mathbf{f}^{(l+1)} + \left[ U^{(l)} \right]^T \tilde{\mathbf{f}}^{(l)} = \begin{bmatrix} \mathbf{u}^{(l+1)} \cdot \mathbf{f}^{(l+1)} \\ \mathbf{b}^{(l+1)} \end{bmatrix} + \left[ U^{(l)} \right]^T \tilde{\mathbf{f}}^{(l)}, \\ A^{(l)} &= \left[ U^{(l)} \right]^T K^{(l+1)} U^{(l)} + \left[ U^{(l)} \right]^T \tilde{K}^{(l)} U^{(l)} \\ &= \begin{bmatrix} a_{00}^{(l)} & \left[ \mathbf{a}^{(l)} \right]^T \\ \mathbf{a}^{(l)} & A^{(l+1)} \end{bmatrix} + \left[ U^{(l)} \right]^T \tilde{K}^{(l)} U^{(l)}, \\ a_{00}^{(l)} &:= \mathbf{u}^{(l+1)} \cdot K^{(l+1)} \mathbf{u}^{(l+1)} = \mathbf{u}^{(l+1)} \cdot \mathbf{f}^{(l+1)} - \mathbf{u}^{(l+1)} \cdot \mathbf{r}^{(l+1)}, \\ \mathbf{a}^{(l)} &:= \left[ U^{(l+1)} \right]^T K^{(l+1)} \mathbf{u}^{(l+1)} = \left[ U^{(l+1)} \right]^T \mathbf{f}^{(l+1)} - \left[ U^{(l+1)} \right]^T \mathbf{r}^{(l+1)} \\ &= \mathbf{b}^{(l+1)} - \left[ U^{(l+1)} \right]^T \mathbf{r}^{(l+1)}. \end{aligned}$$

We note that  $\mathbf{r}^{(l+1)}$  and  $\mathbf{u}^{(l+1)} \cdot \mathbf{f}^{(l+1)}$  are the residual and compliance of simulation  $(l+1)$  that have already been computed by the PCG solver and when accumulating  $J_{\text{LBL}}$ . All that must be computed are the *sparse* products  $\left[ U^{(l)} \right]^T \tilde{\mathbf{f}}^{(l)}$  and  $\left[ U^{(l)} \right]^T \tilde{K}^{(l)} U^{(l)}$  (which can be done efficiently by iterating over only the fine elements whose densities were voided) as well as  $N$  dense dot products  $\mathbf{u}^{(j)} \cdot \mathbf{r}^{(l+1)}$  for  $l+1 \leq j \leq N$ .





**Figure 5: Cut-cell solver validation: we study the number of PCG iterations on average for each layer simulation at different relative residual tolerances (left) and grid resolutions (right). This analysis is performed for fixed designs: a uniform density  $\rho = 0.6$  design (top) and the non-uniform density design shown in the inset of Section 5.1 that is upscaled to higher resolution.**

#### 4.4 Incremental Gradient Calculation

In order to incorporate  $J_{\text{LBL}}$  in a topology optimization, we need its gradient:

$$\begin{aligned} \left[ \frac{\partial J_{\text{LBL}}}{\partial \rho_{\text{fem}}} \right]_e &= \frac{1}{L} \sum_{l=L_e}^L \left[ \frac{\partial J^{(l)}}{\partial \rho_{\text{fem}}} \right]_e \\ &= \frac{1}{L} \sum_{l=L_e}^L \left( \mathbf{u}_e^{(l)} \cdot \mathbf{f}_0 - \frac{1}{2} s'(\rho_{\text{fem}}[e]) \mathbf{u}_e^{(l)} \cdot \mathbf{K}_0 \mathbf{u}_e^{(l)} \right), \end{aligned}$$

where  $L_e$  is the index of the smallest partial structure containing element  $e$ . We note that density variable  $\rho_{\text{fem}}[e]$  has no influence on the partial structures  $l < L_e$  since it and the corresponding attenuated elasticity tensor are masked out by the cut-cell solver applied for those structures (Section 4.1.1).

We compute this gradient efficiently by interleaving our layer simulations with accumulating terms of the gradient (and objective) sums; immediately after  $\mathbf{u}^{(l)}$  is computed, we evaluate the  $l^{\text{th}}$  term and accumulate it to the running total. This way no additional memory storage is needed for gradient calculation. The computational cost of gradient and objective evaluation is also essentially

negligible with this approach (typically accounting for less than 1% of the full optimization time).

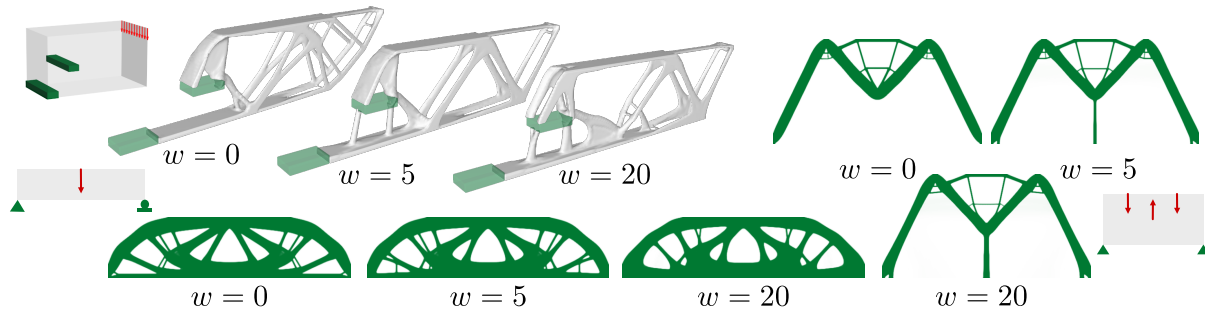
#### 5 EVALUATION

Our accelerated printability term can be included in any topology optimization regardless of the physics or goals involved. However, for simplicity, we evaluate the term in the most popular context of compliance minimization under prescribed loads. We perform the optimization:

$$\begin{aligned} \min_{\mathbf{p}} J_{\text{main}}(\mathcal{F}(\mathbf{p})) + w J_{\text{LBL}}(\mathcal{F}(\mathbf{p})) \\ \text{s.t.} \quad \sum_e \rho_{\text{fem}}[e] \text{Vol}(e) \leq V_{\text{max}}, \end{aligned} \quad (6)$$

where  $w$  is a user-tunable weight trading off between the primary design objective and the goal of ensuring robust layer-by-layer fabricability, and  $J_{\text{main}}$  is the compliance computed by solving (3) with the self-weight boundary conditions replaced by user-specified ones. We use the MMA algorithm for optimization [Svanberg 2007], which is known to scale well to the large numbers of variables involved in topology optimization.

Our solver is implemented in C++ with parallelization via tbb [Intel 2022] and vectorization using Eigen [Guennebaud et al. 2010].

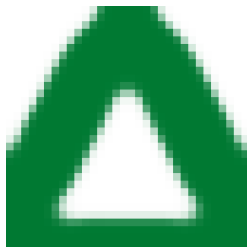


**Figure 6: Dependence of the optimal design on layer-by-layer term weight  $w$  for three different design problems (whose boundary conditions are illustrated by inset schematics). The upper-left design problem was optimized at resolution  $256 \times 64 \times 16$ , right at  $512 \times 256$  and bottom at  $512 \times 128$ . For these examples we used  $N = 3$  and a residual tolerance of  $10^{-5}$ .**

We perform our experiments on a single Linux workstation with an AMD Ryzen 5950X CPU and 128GB RAM, however they could have been achieved with considerably less memory; at  $256 \times 128 \times 128$ , our full solver and topology optimization framework occupies under 4GB RAM.

### 5.1 Cut-Cell MGPCG Validation

To validate the convergence behavior of our cut-cell MGPCG solver, we recorded the number of PCG iterations needed to drive the norm of the residual (force imbalance) below a certain tolerance threshold (relative to the norm of the applied force) in each layer simulation. We did this both for a fixed uniform density design with  $\rho = 0.6$  and for the non-uniform density shown in the inset that we upsampled to each grid tested. We plot in Figure 5 the average number of PCG iterations spent for each layer simulation when initialized from  $\mathbf{u}_{\text{init}}^{(l)} = 0$  and from our subspace search initialization introduced in Section 4.2 with  $N = 3$ . We note that our cut-cell solver is able to drive down the residual significantly faster when greater accuracy is needed, particularly in the non-uniform density case; this is in addition to the fact that *our cut-cell solver also does roughly half the work per iteration* by skipping calculations for detached nodes. We also observe that the number of PCG iterations needed per layer actually *decreases* with increased resolution when using our cut-cell solver paired with our subspace initialization since the layer simulation problems become more and more similar under refinement; this beneficial property is of course lost with a poor initialization (the best we can hope for in general is the number of MGPCG iterations remaining constant under grid refinement), but our cut-cell solver still converges faster from  $\mathbf{u}_{\text{init}}^{(l)} = 0$ .



### 5.2 Weight Dependence

In Figure 6, we demonstrate the effect of changing the weight  $w$  on optimal designs minimizing (6). To prevent the MMA optimization from descending toward a different local optimum in the nonconvex objective landscape and obscuring the trend, we initialize the optimization under each weight choice with the output from the

next-lower weight. We note that even at low weight, the term is effective at inserting support struts for unprintable overhangs in the examples at the top-left and right; these struts are subsequently thickened as the weight is further increased. We also observe that the shallower overhang angles present in the  $w = 0$  design for the MBB beam example (bottom row) is automatically eliminated by the layer-by-layer term without the need for geometric heuristics, and a highly robust, organic design emerges at weight  $w = 20$ .

### 5.3 Initial Guesses in Design Optimization

We study the PCG iteration reduction and speedup achieved by our initial guesses across the constantly-changing designs encountered during topology optimization using 50 design iterations and weight  $w = 10$  for  $J_{\text{LBL}}$ . First we run the optimization using the full-resolution, voxel-level ( $L = N_y$ ) layer-by-layer objective using various initialization strategies and solver accuracies. Representative results are shown in Figure 7; we found essentially identical trends across different settings for  $s_{\text{min}}$  spaced from  $10^{-4}$  to  $10^{-7}$  and different boundary conditions. We note that our initialization achieves a significant speedup over a zero-initialization ( $\approx 8 \times$  fewer PCG iterations at looser tolerances) and maintains a meaningful lead over the “constant” approach of picking  $\mathbf{u}_{\text{init}}^{(l)} = \mathbf{u}^{(l+1)}$  across all residual tolerances. We also demonstrate the trade-off between reducing the residual tolerance used to evaluate  $J_{\text{LBL}}$  and the design’s robustness by plotting  $J_{\text{LBL}}$  re-evaluated at full-resolution and high accuracy for the final design generated by each run on the right. In particular, we observe the relative tolerance has no significant effect on final layer-by-layer objective until dropping below  $10^{-1}$ , suggesting that lowering the residual tolerance is a viable strategy for dramatically reducing computational expense.

One obvious way to accelerate the layer-by-layer simulation is to solve it at a lower resolution than the density field representation and  $J_{\text{main}}$  evaluation, applying (potentially multiple levels of) 2-1 downsampling. We demonstrate that this strategy is mostly orthogonal to our contributions in Figure 8, showing that our initializations provide consistent reduction in solver iterations and computational expense atop the acceleration due to downsampling. We note that the 3D optimization grid started coarse to begin with ( $64 \times 64 \times 64$ ) and coarsening with 6 levels of downsampling would have left no room for improvement by our initializations (having



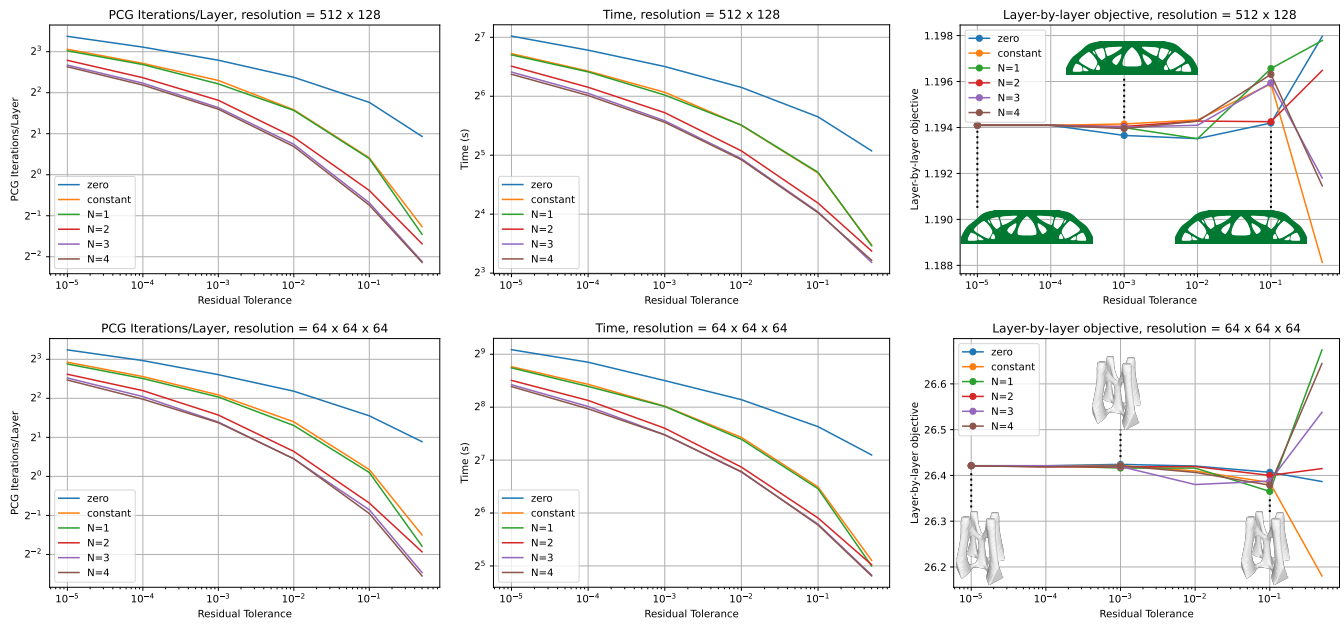
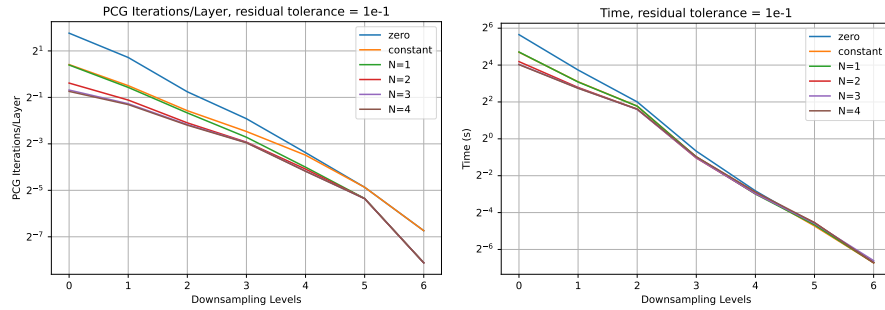


Figure 7: Convergence rate under various initial guesses of the *full-resolution voxel-level layer-by-layer term* incorporated in a 2D (top) and 3D (bottom) topology optimization. On the right, the robustness of the final design is assessed by accurately re-evaluating  $J_{LBL}$  to high accuracy (at CG tolerance  $10^{-12}$ ).

2D Design,  
Resolution  $512 \times 128$



3D Design,  
Resolution  $64 \times 64 \times 64$

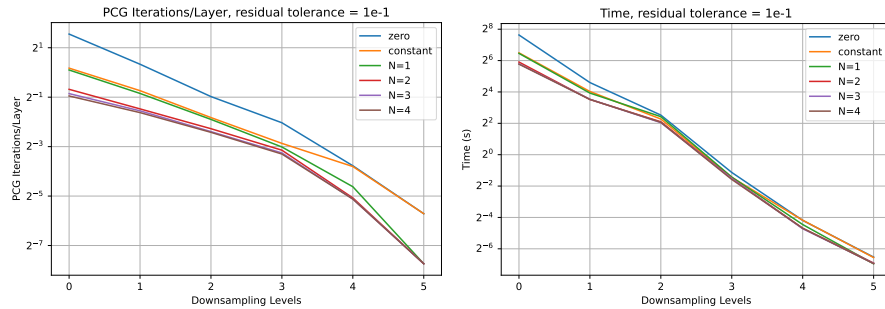


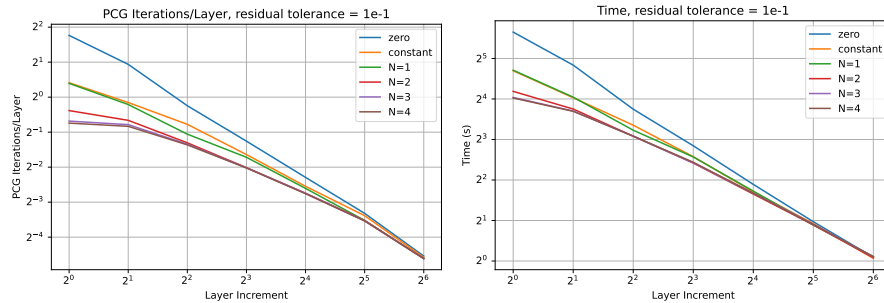
Figure 8: Convergence rate under various initial guesses of the *downsampled voxel-level layer-by-layer term* incorporated in a 2D (top) and 3D (bottom) topology optimization.

reached a resolution of  $1 \times 1 \times 1$ ). Furthermore, well before this coarse of a resolution,  $J_{LBL}$  loses meaning.

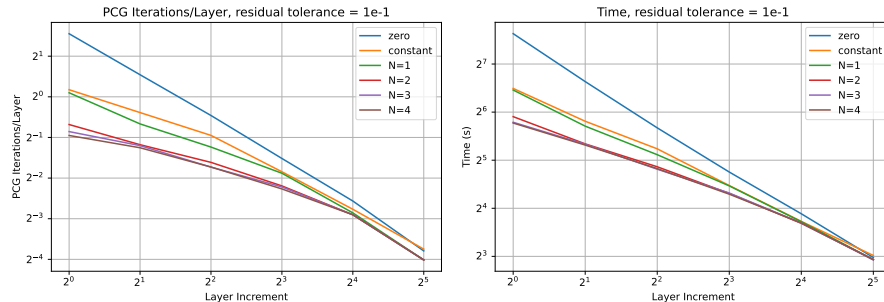
Likewise, one can reduce the number of simulations run by skipping layers, at the risk of potentially missing unprintable features;

this was the strategy used by [Allaire et al. 2017] and [Haveroth et al. 2022]. We demonstrate the performance of our initial guesses in this mode and the final design robustness in Figure 9. Once again our initializations consistently provide improvement, though naturally

**2D Design,  
Resolution  $512 \times 128$**



**3D Design,  
Resolution  $64 \times 64 \times 64$**

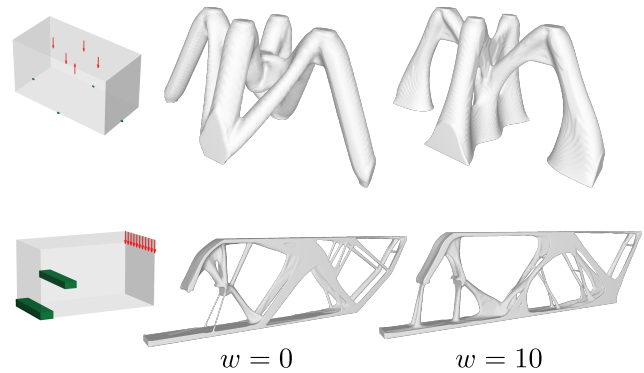


**Figure 9: Convergence rate under various initial guesses of the full-resolution layer-by-layer term with layer skipping incorporated in a 2D (top) and 3D (bottom) topology optimization.**

the benefit is diminished due to the decreased similarity between  $\mathbf{u}^{(l)}$  and  $\mathbf{u}^{(l+i)}$  for large layer increments  $i$ . Of course, when this strategy is employed, unprintable features below the length-scale of the layer increment can become invisible to the simulation.

**5.4 High Resolution Design**

In Figure 10, we demonstrate the resolution of 3D designs that can be achieved in a few hours on a standard desktop workstation thanks to our accelerations. In both cases, we use our  $N = 3$  initialization and run 50 iterations of the design optimization. The design at the top half was generated at resolution  $256 \times 128 \times 128$  and used a residual tolerance of  $10^{-2}$  for the PCG solves. This entire process took 53 minutes. The design on the bottom was generated at resolution  $512 \times 128 \times 32$ . This time a much stricter residual tolerance was used ( $10^{-5}$ ) to illustrate that our framework still has acceptable performance at this accuracy level. This optimization took 171 minutes, but we note that according to the scaling behavior shown in Figure 7 we should expect a runtime of roughly 4x lower at a residual tolerance of  $10^{-2}$ .



**Figure 10: Topology optimizations of designs on grids of size  $256 \times 128 \times 128$  (top) and  $512 \times 128 \times 32$  (bottom). The boundary conditions used in design are shown in the inset schematics. The designs on the left were generated without  $J_{LBL}$  and feature unprintable overhangs that are removed by  $J_{LBL}$ .**

**6 CONCLUSION**

We conclude with some observations on limitations of our solver and opportunities for future work. First, we have noticed that while our initial guesses tend to be particularly excellent for the solid regions, they have difficulty accurately predicting the deformation of void nodes, which our MGPCG solver then takes substantial time to correct. We suspect that this is related to conditioning problems caused by the high contrast ratio between stiffnesses assigned to the solid and void regions ( $1/s_{min}$ ). We plan to study this issue in future work and potentially develop better contrast-aware

coarsening operators that we conjecture will further accelerate the solves. Second, at high weight, the layer-by-layer objective term causes convergence issues in the optimality criteria (OC) method, which otherwise performs excellently for compliance minimization under a volume constraint. For this reason, we used the MMA algorithm for all results in this paper, but we would like to study this phenomenon in more depth and possibly develop a more robust variant of the OC algorithm. Third, we are interested to consider other physics at play during the fabrication process, like the peeling forces applied during stereolithography printing or thermal stresses

in laser sintering. Fourth, we are excited to extend our acceleration strategies to address general staged construction schemes, where incremental parts can feature curved boundaries, and worst-case load formulations. Finally, we would like to explore the behavior of the layer-by-layer objective in more settings: for a greater range of boundary conditions, different primary objectives, and in multiphysics problems.

We believe that with our fast solver, high-quality, inexpensive initialization scheme, and demonstration of resilience to loose residual tolerance, we have shown the layer-by-layer simulation approach is viable for ensuring robust manufacturability in realistic-scale topology optimization problems.

## ACKNOWLEDGMENTS

We thank Joseph Teran for numerous insightful discussions and suggestions throughout the project and Michele Vidulis, Shad Durusel, and Vincent Pollet for their help during the early development stages of our multigrid topology optimization framework.

## REFERENCES

- Grégoire Allaire, Charles Dapogny, Alexis Faure, and Georgios Michailidis. 2017. Shape optimization of a layer by layer mechanical constraint for additive manufacturing. *Comptes Rendus Mathématique* 355, 6 (2017), 699–717. <https://doi.org/10.1016/j.crma.2017.04.008>
- Grégoire Allaire, Françoise Jouve, and Georgios Michailidis. 2016. Thickness control in structural optimization via a level set method. *Structural and Multidisciplinary Optimization* 53, 6 (2016), 1349–1382.
- Grégoire Allaire, Françoise Jouve, and Anca-Maria Toader. 2004. Structural optimization using sensitivity analysis and a level-set method. *J. Comput. Phys.* 194, 1 (2004), 363–393. <https://doi.org/10.1016/j.jcp.2003.09.032>
- Oded Amir, Niels Aage, and Boyan S Lazarov. 2014. On multigrid-CG for efficient topology optimization. *Structural and Multidisciplinary Optimization* 49, 5 (2014), 815–829.
- Oded Amir and Yoram Mass. 2018. Topology optimization for staged construction. *Structural and Multidisciplinary Optimization* 57, 4 (2018), 1679–1694.
- Jernej Barbic and Doug L James. 2005. Real-time subspace integration for St. Venant-Kirchhoff deformable models. *ACM transactions on graphics (TOG)* 24, 3 (2005), 982–990.
- Mohamad Bayat, Wen Dong, Jesper Thorborg, Albert C. To, and Jesper H. Hattel. 2021. A review of multi-scale and multi-physics simulations of metal additive manufacturing processes with focus on modeling strategies. *Additive Manufacturing* 47 (2021), 102278. <https://doi.org/10.1016/j.addma.2021.102278>
- Martin Philip Bendsoe and Ole Sigmund. 2003. *Topology optimization: theory, methods, and applications*. Springer Science & Business Media.
- William Briggs, Van Henson, and Steve McCormick. 2000. *A Multigrid Tutorial, 2nd Edition*.
- Tyler E Bruns and Daniel A Tortorelli. 2001. Topology optimization of non-linear elastic structures and compliant mechanisms. *Computer methods in applied mechanics and engineering* 190, 26-27 (2001), 3443–3459.
- Michael Bruyneel and Pierre Duysinx. 2005. Note on topology optimization of continuum structures including self-weight. *Structural and Multidisciplinary Optimization* 29, 4 (2005), 245–256.
- Xiang Chen, Changxi Zheng, and Kun Zhou. 2016. Example-Based Subspace Stress Analysis for Interactive Shape Design. *PP*, 99 (2016).
- Elena Cherkav and Andrej Cherkav. 2008. Minimax optimization problem of structural design. *Computers & Structures* 86, 13-14 (2008), 1426–1435.
- Charles Dapogny, Cécile Dobrzynski, and Pascal Frey. 2014. Three-dimensional adaptive domain remeshing, implicit domain meshing, and applications to free and moving boundary problems. *Journal of computational physics* 262 (2014), 358–378.
- Nikan Doosti, Julian Panetta, and Vahid Babaei. 2021. Topology Optimization via Frequency Tuning of Neural Design Representations. In *Symposium on Computational Fabrication*. 1–9.
- Tao Du, Kui Wu, Andrew Spielberg, Wojciech Matusik, Bo Zhu, and Eftychios Sifakis. 2020. Functional Optimization of Fluidic Devices with Differentiable Stokes Flow. *ACM Trans. Graph.* 39, 6, Article 197 (nov 2020), 15 pages. <https://doi.org/10.1145/3414685.3417795>
- Carlos Faria, Jaime Fonseca, and Estela Bicho. 2020. FIBR3DEmul—an open-access simulation solution for 3D printing processes of FDM machines with 3+ actuated axes. *The International Journal of Advanced Manufacturing Technology* 106, 7 (2020), 3609–3623.
- Andrew T Gaynor and James K Guest. 2016. Topology optimization considering overhang constraints: Eliminating sacrificial support material in additive manufacturing through design. *Structural and Multidisciplinary Optimization* 54, 5 (2016), 1157–1172.
- Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. <http://eigen.tuxfamily.org>.
- G.A. Haverth, C.-J. Thore, M.R. Correa, R.F. Ausas, S. Jakobsson, J.A. Cuminato, and A. Klarbring. 2022. Topology optimization including a model of the layer-by-layer additive manufacturing process. *Computer Methods in Applied Mechanics and Engineering* 398 (2022), 115203. <https://doi.org/10.1016/j.cma.2022.115203>
- Intel. 2022. Intel® oneAPI Threading Building Blocks. <https://github.com/oneapi-src/oneTBB>
- Hokeun Kim, Yan Zhao, and Lihua Zhao. 2016. Process-level modeling and simulation for HP’s Multi Jet Fusion 3D printing technology. In *2016 1st international workshop on cyber-physical production systems (CPPS)*. IEEE, 1–4.
- Matthijs Langelaar. 2017. An additive manufacturing filter for topology optimization of print-ready designs. *Structural and multidisciplinary optimization* 55, 3 (2017), 871–883.
- Timothy Langlois, Ariel Shamir, Daniel Dror, Wojciech Matusik, and David IW Levin. 2016. Stochastic structural analysis for context-aware design and fabrication. *ACM Transactions on Graphics (TOG)* 35, 6 (2016), 226.
- Haixiang Liu, Yuanming Hu, Bo Zhu, Wojciech Matusik, and Eftychios Sifakis. 2018. Narrow-Band Topology Optimization on a Sparsely Populated Grid. *ACM Trans. Graph.* 37, 6, Article 251 (dec 2018), 14 pages. <https://doi.org/10.1145/3272127.3275012>
- Shinji Nishiwaki, Mary I Frecker, Seungjae Min, and Noboru Kikuchi. 1998. Topology optimization of compliant mechanisms using the homogenization method. *International journal for numerical methods in engineering* 42, 3 (1998), 535–559.
- Timilehin Martins Oyinloye and Won Byong Yoon. 2021. Application of Computational Fluid Dynamics (CFD) Simulation for the Effective Design of Food 3D Printing (A Review). *Processes* 9, 11 (2021). <https://doi.org/10.3390/pr9111867>
- Julian Panetta, Abtin Rahimian, and Denis Zorin. 2017. Worst-Case Stress Relief for Microstructures. *ACM Trans. Graph.* 36, 4, Article 122 (jul 2017), 16 pages. <https://doi.org/10.1145/3072959.3073649>
- Julian Panetta, Qingnan Zhou, Luigi Malomo, Nico Pietroni, Paolo Cignoni, and Denis Zorin. 2015. Elastic textures for additive fabrication. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–12.
- Xiaoping Qian. 2017. Undercut and overhang angle control in topology optimization: a density gradient based integral approach. *Internat. J. Numer. Methods Engrg.* 111, 3 (2017), 247–272.
- Christian Schumacher, Bernd Bickel, Jan Rys, Steve Marschner, Chiara Daraio, and Markus Gross. 2015. Microstructures to control elasticity in 3D printing. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–13.
- Christian Schumacher, Jonas Zehnder, and Moritz Bächer. 2018. Set-in-Stone: Worst-Case Optimization of Structures Weak in Tension. *ACM Trans. Graph.* 37, 6, Article 252 (Dec. 2018), 13 pages. <https://doi.org/10.1145/3272127.3275085>
- Eftychios Sifakis and Jernej Barbic. 2012. FEM simulation of 3D deformable solids: a practitioner’s guide to theory, discretization and model reduction. In *ACM siggraph 2012 courses*. 1–50.
- Ole Sigmund and Joakim Petersson. 1998. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization* 16, 1 (1998), 68–75.
- M. Stolpe and K. Svanberg. 2001. An Alternative Interpolation Scheme for Minimum Compliance Topology Optimization. *Struct. Multidiscip. Optim.* 22, 2 (sep 2001), 116–124. <https://doi.org/10.1007/s0015801580129>
- Krister Svanberg. 2007. MMA and GCMMA-two methods for nonlinear optimization. *vol 1* (2007), 1–15.
- Adrien Treuille, Andrew Lewis, and Zoran Popović. 2006. Model reduction for real-time fluids. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 826–834.
- Nico P Van Dijk, Kurt Maute, Matthijs Langelaar, and Fred Van Keulen. 2013. Level-set methods for structural topology optimization: a review. *Structural and Multidisciplinary Optimization* 48, 3 (2013), 437–472.
- Fengwen Wang, Boyan Stefanov Lazarov, and Ole Sigmund. 2011. On projection methods, convergence and robust formulations in topology optimization. *Structural and Multidisciplinary Optimization* 43, 6 (2011), 767–784.
- Weiming Wang, Dirk Munro, Charlie CL Wang, Fred van Keulen, and Jun Wu. 2020. Space-time topology optimization for additive manufacturing. *Structural and Multidisciplinary Optimization* 61, 1 (2020), 1–18.
- Jun Wu, Christian Dick, and Rüdiger Westermann. 2016. A System for High-Resolution Topology Optimization. *IEEE Transactions on Visualization and Computer Graphics* 22, 3 (2016), 1195–1208. <https://doi.org/10.1109/TVCG.2015.2502588>
- Hongyi Xu, Yijing Li, Yong Chen, and Jernej Barbic. 2015. Interactive material design using model reduction. *ACM Transactions on Graphics (TOG)* 34, 2 (2015), 1–14.
- Suna Yan, Fengwen Wang, Jun Hong, and Ole Sigmund. 2019. Topology optimization of microchannel heat sinks using a two-layer model. *International Journal of Heat and Mass Transfer* 143 (2019), 118462.
- Qingnan Zhou, Julian Panetta, and Denis Zorin. 2013. Worst-case structural analysis. *ACM Trans. Graph.* 32, 4, Article 137 (July 2013), 12 pages.