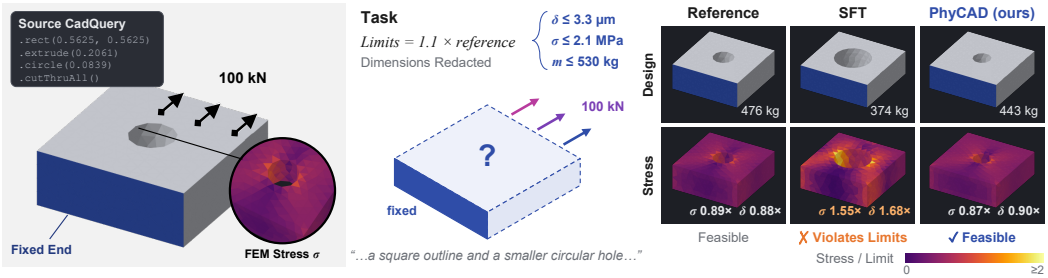
PHYCAD: PHYSICS-AWARE TEXT-TO-CAD GENERATION VIA HIERARCHICAL REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review



(A) FEM-annotated Source CAD **(B)** Task Given to The Policy **(C)** Same Task in SFT & PhyCAD

Figure 1: **Physics-aware CAD optimization.** **(A)** A source CAD model is annotated with FEM conditions and responses. **(B)** Its exact dimensions are redacted to form an inverse-design task with physical constraints. **(C)** While the SFT design achieves lower mass but violates the physical constraints, PhyCAD produces a feasible design with lower mass than the reference.

ABSTRACT

Language-driven computer-aided design (CAD) generation enables the creation of editable parametric models from text, yet geometric correctness alone does not ensure physical feasibility. We present PhyCAD, a framework for physics-aware Text-to-CAD generation that trains language models to produce executable CadQuery programs under geometric and physical constraints. To support this setting, we construct a Text-to-CAD dataset that links parametric CAD models with geometric specifications, loading conditions, and physical constraints. We further develop an automated CAD–CAE verifier that combines program execution, geometric checks, and finite element analysis to provide feedback for model training. A key learning challenge is to prioritize joint satisfaction of geometric and physical constraints while retaining informative feedback on partial progress. A flat aggregation of heterogeneous rewards can obscure this priority, allowing improvements in individual criteria to compensate for unmet design requirements. We therefore introduce a Hierarchical Balanced Advantage for Group Relative Policy Optimization, grouping candidates into levels according to their verification status and combining between-level progress with objective-wise peer ranking within each level. This formulation preserves cross-level priority in the training signal while providing fine-grained feedback for satisfying constraints and improving feasible designs. Experiments across multiple language models show that PhyCAD consistently improves joint geometric and physical constraint satisfaction. These results demonstrate the value of verifiable engineering feedback for advancing language-driven CAD generation beyond geometric reconstruction toward physics-aware design.

1 INTRODUCTION

Computer-aided design (CAD) is a fundamental component of modern engineering, providing editable and parametric representations for downstream simulation, optimization, and manufactur-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ing. Learning-based CAD generation has progressively moved from modeling CAD construction sequences and design histories (Wu et al., 2021; Willis et al., 2021) toward language-driven generation with large language models (LLMs) (Zhang et al., 2026). Recent methods can translate natural-language descriptions into parametric CAD sequences (Khan et al., 2024), generate CAD programs from textual or multimodal inputs (Li et al., 2025c; Sun et al., 2025), or synthesize executable modeling code through iterative or tool-augmented workflows (Badagabettu et al., 2024; Xie & Ju, 2025; Mallis et al., 2025). In particular, CadQuery provides an executable, Python-based parametric representation that aligns naturally with the code-generation capabilities of pretrained LLMs (Xie & Ju, 2025; Guan et al., 2025).

Despite this progress, geometric correctness alone does not ensure physical feasibility. Existing Text-to-CAD systems are commonly optimized or evaluated using geometric similarity or visual consistency (Khan et al., 2024; Alrashedy et al., 2025; Xie & Ju, 2025; Guan et al., 2025). However, natural-language descriptions do not always fully specify the dimensions, topology, and structural details required for engineering design, and even small geometric deviations may substantially affect structural behavior. More importantly, engineering design is ultimately concerned with whether a component meets prescribed dimensional and physical requirements under specified material properties, loads, and boundary conditions, rather than merely whether it resembles a target shape. Bringing an initially generated model into compliance often requires repeated manual measurement, geometric modification, simulation, and re-evaluation, resulting in a costly refinement process. These limitations have motivated increasing attention to functional and physics-based criteria for evaluating and refining generated CAD models, allowing engineering feedback to guide geometry toward designs that satisfy downstream requirements (Berger et al., 2026; Son et al., 2026).

Computer-aided engineering (CAE), particularly simulation based on the finite element method (FEM), provides a natural source of deterministic feedback for evaluating generated designs. Recent studies have demonstrated the use of LLMs in automated FEM-based workflows and the acquisition of verifiable engineering signals (Hou et al., 2025; Qi et al., 2025). Simulation outcomes have also been used to guide iterative CAD generation and optimization (Berger et al., 2026; Son et al., 2026; Deng et al., 2026). Reinforcement learning (RL) has further enabled CAD-generation models to learn from geometric feedback (Guan et al., 2025). Training with both geometric and physical feedback introduces a key challenge: prioritizing joint constraint satisfaction while retaining informative feedback on partial progress. A flat aggregation of heterogeneous rewards can obscure this priority, allowing improvements in individual criteria to compensate for unmet design requirements. This calls for a learning signal that distinguishes progress toward feasibility from improvements within the feasible design space.

As illustrated in Fig. 1, we present PhyCAD, a physics-aware Text-to-CAD framework that generates executable CadQuery programs from geometric specifications and physical constraints under prescribed material properties, loads, and boundary conditions. The task defines a constrained design space with admissible dimensional ranges rather than a unique target geometry. We construct a corresponding dataset and an automated CAD–CAE verifier that provides training feedback through program execution, geometric checks, and FEM-based simulation. Building on supervised fine-tuning, we introduce a Hierarchical Balanced Advantage for Group Relative Policy Optimization (GRPO) (Shao et al., 2024), grouping candidates by verification status and combining between-level progress with objective-wise peer ranking within each level. This formulation preserves cross-level priority in the training signal while providing fine-grained feedback for satisfying constraints and improving feasible designs. The trained model directly generates programs without an iterative simulation-guided refinement loop at inference time. Experiments across multiple language models show that PhyCAD consistently improves joint geometric and physical constraint satisfaction.

Our contributions are summarized as follows:

- We formulate physics-aware Text-to-CAD generation and construct a dataset that jointly captures geometric specifications, loading and boundary conditions, and physics-based constraints.
- We develop an automated CAD–CAE verification pipeline that provides verifiable feedback on program executability, geometric and dimensional compliance, and physical constraint satisfaction to support model training.
- We introduce a Hierarchical Balanced Advantage that combines between-level progress with objective-wise peer ranking within each level, preserving cross-level priority in the training signal while providing fine-grained feedback for satisfying constraints and improving feasible designs.

2 Related Work

Text-to-CAD Generation. Early learning-based CAD methods modeled construction histories, sketch constraints, and assembly relations using datasets such as DeepCAD (Wu et al., 2021), SketchGraphs (Seff et al., 2020), and Fusion 360 Gallery (Willis et al., 2021; 2022). Subsequent approaches map text or engineering drawings to CAD sequences using task-specific transformers (Khan et al., 2024; Qin et al., 2025), or adapt pretrained LLMs and VLMs to structured CAD generation (Li et al., 2025a; Govindarajan et al., 2025; Yuan et al., 2024). CadQuery supports executable program generation (Xie & Ju, 2025), while tool augmentation, self-refinement, and evolutionary search enable iterative refinement (Mallis et al., 2025; Li et al., 2026b; Preintner et al., 2025). PhyCAD studies generation under joint geometric and physical requirements, including global dimensional bounds and FEM-verified physical constraints.

Physics-Aware CAD Generation and Engineering Agents. Recently, (Berger et al., 2026) integrates physics-based verification into multi-agent CAD refinement, while (Son et al., 2026) combines structured blueprints, visual inspection, and finite-element feedback to refine single-part and multi-part STEP models. COSMO-Agent (Deng et al., 2026) trains an LLM through tool-augmented RL to optimize parameterized components through CAE-guided updates to geometry and material choices. Related agents automate computational fluid dynamics (Chen et al., 2024; Xu et al., 2025; Yue et al., 2025), finite-element analysis (Hou et al., 2025; Qi et al., 2025), and multiphysics simulation (Zhang et al., 2025). Complementary to these engineering workflows, PhyCAD addresses a learning challenge in physics-aware CAD generation: prioritizing joint geometric and physical feasibility while retaining informative feedback on partial progress. We introduce a Hierarchical Balanced Advantage that combines progress across verification levels with objective-wise peer ranking within each level. This organizes heterogeneous CAD–CAE outcomes into a training signal that preserves cross-level priority while guiding constraint satisfaction and improvements within the feasible design space.

Reinforcement Learning with Verifiable Feedback. GRPO (Shao et al., 2024) enables policy optimization using group-relative rewards, while InternBootcamp (Li et al., 2025b) provides diverse tasks with programmatic verifiers. In CAD, CAD-Coder (Guan et al., 2025) combines supervised fine-tuning and GRPO using geometric and format rewards, while cadrille (Kolodiazhnyi et al., 2025) uses geometric feedback for multimodal reconstruction. ReCAD (Li et al., 2026a) combines parameterized-code guidance with hierarchical primitive learning, following a curriculum over increasingly complex geometric structures. These methods primarily target geometric fidelity and output validity rather than FEM-verified physical constraints. PhyCAD incorporates CAD–CAE feedback into policy training through a Hierarchical Balanced Advantage. Rather than scheduling a geometric curriculum, it groups candidates by verification status and combines between-level progress with objective-wise peer ranking within each level.

3 Method

Our goal is to train a LLM to generate CAD programs that satisfy both geometric specifications and physical constraints. As illustrated in Fig. 2, our framework takes a physics-aware prompt as input, generates candidate CadQuery programs with an LLM policy, evaluates them through a deterministic CAD–CAE verification pipeline, and converts the verification outcomes into structured reinforcement signals for policy optimization. The resulting training framework integrates CAD executability, geometric compliance, and physics-based feasibility into a unified optimization process.

3.1 Problem Formulation

Let $q \in \mathcal{Q}$ denote a physics-aware design prompt describing geometric requirements, the admissible design domain, material properties, loading and boundary conditions, engineering constraints, and a design objective. An LLM policy π_θ, parameterized by $\theta \in \mathbb{R}^d$, defines a conditional distribution over CadQuery programs $\mathcal{O}$. During training, we sample a group of candidates $o_i \sim \pi_\theta(\cdot \mid q)$, $i = 1, \ldots, G$.

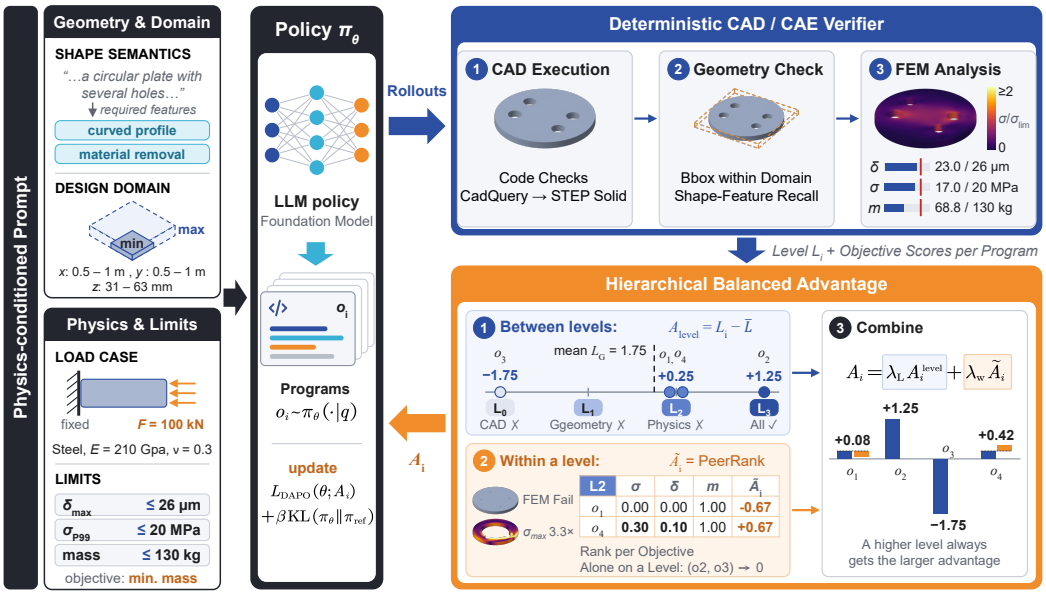


Figure 2: **Overview of our framework.** The LLM policy generates candidate CAD programs from a physics-conditioned prompt containing geometric requirements, design-domain constraints, loads, and physical limits. A deterministic CAD-CAE verifier sequentially evaluates each program for CAD executability, geometric validity, and FEM-based physical feasibility and objectives. These verification results define a Hierarchical Balanced Advantage, which combines between-level progress with within-level peer ranking over objective scores. The resulting advantage A_i provides structured reinforcement signals for updating the CAD-generation policy.

Our setting specifies selected geometric parameters through admissible ranges rather than fixed values, defining a constrained design space rather than a unique target geometry. A program $o \in \mathcal{O}$ is feasible if it executes successfully to produce a valid CAD solid, satisfies the specified geometric requirements, and passes FEM evaluation with all physical constraints satisfied. We denote the feasible set by $\mathcal{F}(q) \subseteq \mathcal{O}$. Among feasible programs, we optimize an engineering objective $J(o; q)$, with smaller values preferred. For the tasks considered here, $J(o; q) = m(o; q)$, where $m(o; q)$ is the mass of the generated solid. The resulting constrained design problem is

$$o^* \in \arg\min_{o \in \mathcal{F}(q)} J(o; q). \tag{1}$$

During training, optimization is performed over the sampled candidate set $\mathcal{O}_\theta(q)$. The CAD–CAE verifier identifies the feasible subset $\mathcal{F}_\theta(q)$ and provides hierarchical feedback for candidates according to their verification status. The policy is optimized to increase the probability of feasible designs while improving $J(o; q)$ among feasible candidates.

3.2 Deterministic CAD–CAE Verifier

For any group of programs o_i sampled from $\pi_\theta(\cdot \mid q)$, we evaluate each program using the same deterministic CAD–CAE pipeline. The verifier proceeds through three stages.

CAD execution. The generated program is checked for syntax and safety, executed, and tested for valid output construction. It must produce a valid CAD solid that can be exported in STEP format and reimported by the downstream geometry kernel. Failure at this stage is recorded as a CAD-construction failure.

Geometry check. For successfully constructed solids, we compare the global bounding-box extents with the prompt's dimensional ranges and check whether the required geometric constructions are present. This checks compliance with the design requirements, separately from the solid-validity check above. Candidates satisfying these requirements proceed to CAE verification.

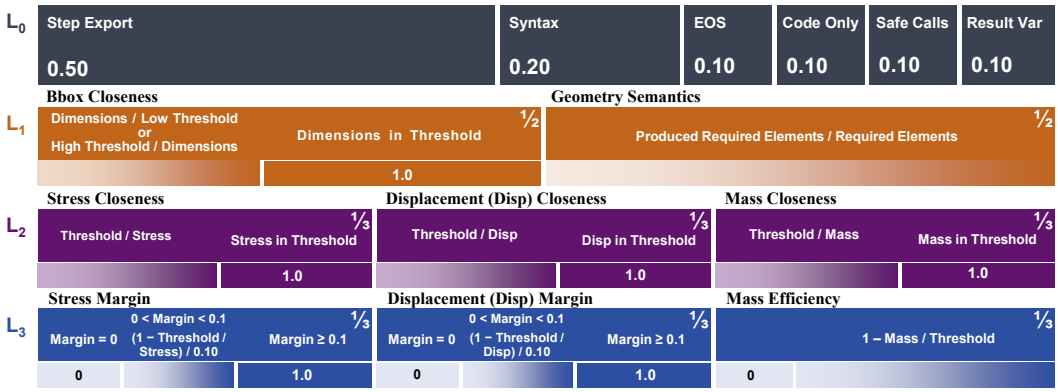


Figure 3: **Reward definitions for within-level PeerRank.** Each block specifies how an evaluated outcome is mapped to an objective reward within its corresponding level. Solid regions denote fixed rewards once a condition is satisfied, while shaded gradients indicate rewards that vary continuously with closeness to the target or the achieved margin; faded regions correspond to outcomes receiving zero or lower reward. The resulting objective rewards are independently peer-ranked and combined using the indicated within-level weights.

CAE verification. The solid is meshed and analyzed using FEM under the prescribed material properties, loads, and boundary conditions. For candidate o_i, we record the volume-weighted 99th-percentile element von Mises stress σ_i, maximum nodal displacement δ_i, and mass $m_i = m(o_i; q)$. Let $\bar{\sigma}$, $\bar{\delta}$, and $\bar{m}$ denote their respective upper bounds in the prompt. Physical verification succeeds when the CAE procedure completes and $\sigma_i \leq \bar{\sigma}$, $\delta_i \leq \bar{\delta}$, and $m_i \leq \bar{m}$.

To summarize these outcomes, let $L_i \in \{0, 1, 2, 3\}$ denote the verification level of candidate o_i:

$$L_i = \begin{cases} 0, & \text{CAD construction fails,} \\ 1, & \text{CAD succeeds but geometric requirements fail,} \\ 2, & \text{geometry passes but CAE or a physical constraint fails,} \\ 3, & \text{all geometric and physical requirements pass.} \end{cases} \tag{2}$$

Thus, the three checks yield four possible outcomes: failure at any one of the checks, or success at all three. A larger L_i indicates further progress through the verification procedure; $L_i = 3$ identifies a jointly feasible candidate.

3.3 Hierarchical Balanced Advantage

The verification levels distinguish coarse progress toward feasibility but do not distinguish candidates with the same status. For example, two physically infeasible candidates may violate their constraints by very different amounts. Conversely, directly aggregating all objective scores can allow improvements in individual criteria to compensate for unmet requirements. We therefore combine a between-level advantage with a within-level comparison, retaining both the verification ordering and feedback on partial progress.

Between levels. For G rollouts generated from the same prompt, we define the level advantage as

$$A_i^{\text{level}} = L_i - \frac{1}{G} \sum_{j=1}^{G} L_j. \tag{3}$$

This term compares each candidate's level with the group mean. Candidates above the mean receive positive level advantages. When all candidates occupy the same level, this term is zero, and the within-level component provides the remaining distinctions.

Within a level. Candidates are compared only with others from the same prompt and the same verification level. We first map the verification outcomes to level-specific objective scores, then rank candidates separately for each objective. Fig. 3 summarizes the score definitions and weights. **(1).** For candidates with $L_i = 0$, CAD-progress feedback reflects output completeness, code-only formatting, syntax validity, safe calls, result construction, and STEP export. This distinguishes partial construction progress even when a valid CAD solid has not been obtained. **(2).** For $L_i = 1$, we assign equal weights to bounding-box closeness and geometric-semantic recall. These scores measure proximity to the dimensional ranges and coverage of the required geometric constructions, respectively. **(3).** For $L_i = 2$, we compare closeness to the stress, displacement, and mass limits, again with equal weights. For an available measurement v with upper bound $\bar{v}$, its score is 1 when $v \le \bar{v}$, and $\bar{v}/v$ otherwise. An unavailable measurement receives a score of zero. These scores distinguish different degrees of constraint violation without rewarding further improvement of a quantity that already satisfies its limit. **(4).** For $L_i = 3$, all constraints are satisfied. We compare mass efficiency $1 - m_i/\bar{m}$, stress margin, and displacement margin with equal weights. The two margin scores are capped at a 10% margin, beyond which additional margin receives no further reward. Feasible-design feedback therefore combines the mass objective with capped margin objectives rather than ranking feasible candidates by mass alone.

For a given prompt and a level $\ell \in \{0, 1, 2, 3\}$, let $\mathcal{I}_\ell = \{j \in \{1, \dots, G\} : L_j = \ell\}$ be the set of candidate indices at that level. Let K_ℓ be its number of objectives. For candidate $i \in \mathcal{I}_\ell$, denote its score for objective $k \in \{1, \dots, K_\ell\}$ by s_{ik}, and its tie-aware centered rank among $\mathcal{I}_\ell$ by $r_{ik} \in [-1, 1]$, with higher ranks indicating better scores. Scores within 10^{-3} are treated as ties. With nonnegative objective weights $w_{\ell k}$ satisfying $\sum_{k=1}^{K_\ell} w_{\ell k} = 1$, the within-level advantage is

$$\widetilde{A}_i = \begin{cases} \displaystyle\sum_{k=1}^{K_{L_i}} w_{L_i k}\, r_{ik}, & |\mathcal{I}_{L_i}| > 1, \\ 0, & |\mathcal{I}_{L_i}| = 1. \end{cases} \tag{4}$$

We refer to this objective-wise ranking and averaging operation as PeerRank. It compares ranks rather than a weighted sum of the original objective scores, and gives $|\widetilde{A}_i| \le 1$.

Combined advantage. The nonnegative weights λ_L and λ_w control the between-level and within-level terms, respectively. We combine them as

$$A_i = \lambda_L A_i^{\text{level}} + \lambda_w \widetilde{A}_i. \tag{5}$$

For candidates i and j from the same prompt group with $L_i > L_j$, the common level baseline cancels, giving

$$A_i - A_j = \lambda_L (L_i - L_j) + \lambda_w (\widetilde{A}_i - \widetilde{A}_j) \ge \lambda_L - 2\lambda_w.$$

Choosing $\lambda_L > 2\lambda_w$ therefore ensures that the higher-level candidate receives a larger advantage regardless of its within-level rank. This is an ordering guarantee for the training signal within each prompt group. Both components are computed together, not introduced in successive training stages.

Policy optimization. We use the hierarchical advantages in GRPO (Shao et al., 2024). The policy loss uses DAPO-style token-level aggregation: generated-token losses are averaged across the batch rather than first averaged separately within each completion. Let $\mathbf{A}$ collect the advantages for the sampled completions, each computed within its own prompt group, and let $\mathcal{L}_{\text{DAPO}}(\theta; \mathbf{A})$ denote the GRPO policy term with this aggregation. With a reference policy π_{ref} and KL regularization weight $\beta \ge 0$, the training objective is

$$\mathcal{L} = \mathcal{L}_{\text{DAPO}}(\theta; \mathbf{A}) + \beta D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{ref}}), \tag{6}$$

where D_{KL} denotes the Kullback–Leibler divergence. The CAD kernel and FEM solver supply verification outcomes, not gradients through geometry construction or simulation. At inference time, the trained policy directly generates a CadQuery program without an iterative simulation-guided refinement loop.

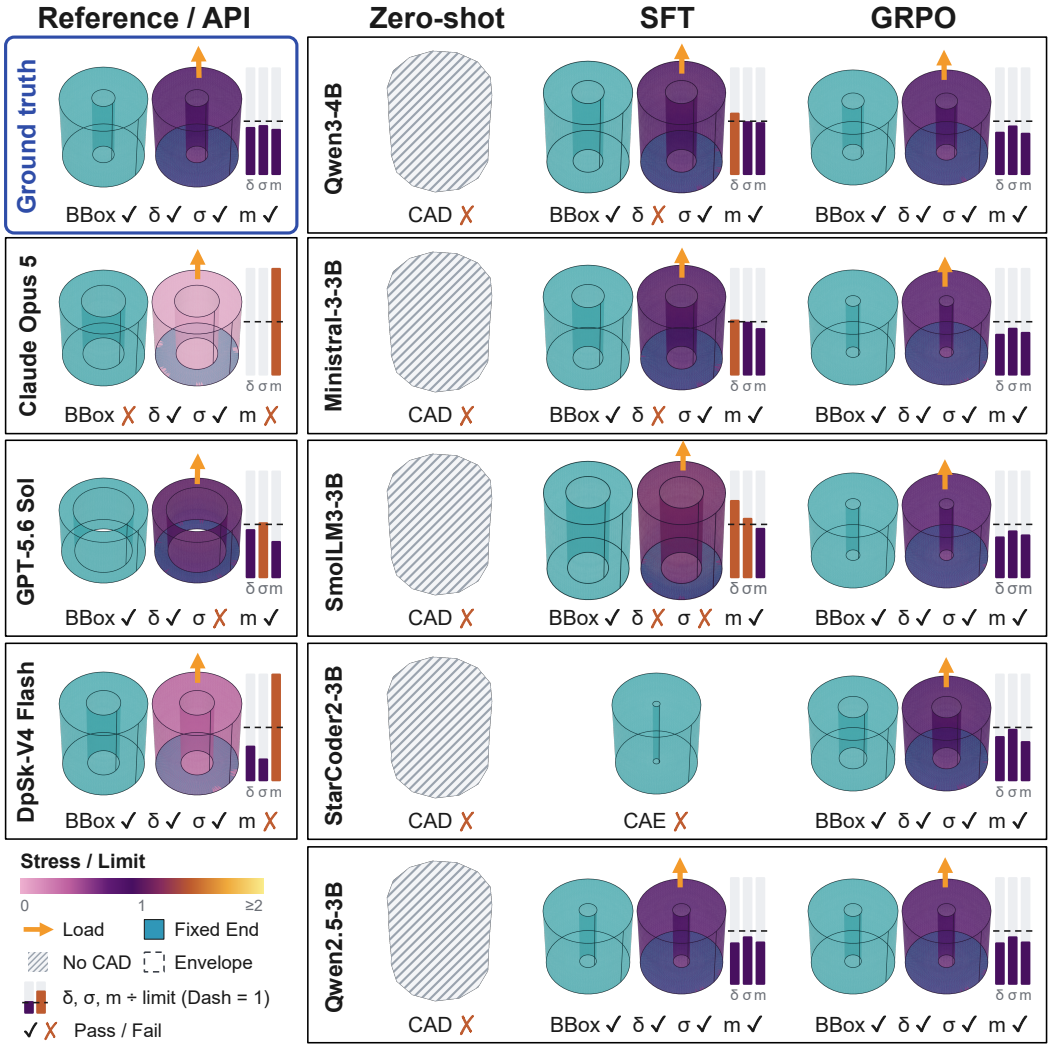


Figure 4: **Qualitative comparison across training stages and model families.** Surface color indicates stress relative to the prescribed limit, while the bars summarize displacement, stress, and mass constraint satisfaction. Across model families, SFT improves CAD generation but can still violate geometric or physical constraints, whereas GRPO more consistently achieves joint feasibility.

4 Experiment

4.1 Dataset Generation

We construct a physics-aware Text-to-CAD dataset from the Text-to-CadQuery corpus (Xie & Ju, 2025), which contains approximately 110k CadQuery programs with geometric and semantic descriptions. Each program serves as a reference design and is processed by a CAD–CAE pipeline. We first derive boundary and loading conditions from the geometry, then perform FEM to obtain the maximum displacement $\delta_{\max}$, 99-th percentile von Mises stress σ_{P99}, and mass m.

Each successfully analyzed reference is converted into an inverse-design task rather than used as a direct reconstruction target. We remove exact dimensions and other numerical specifications that could reveal the reference geometry, while retaining high-level geometric semantics needed to define the intended structure. The reference measurements are then used to derive displacement, stress, and mass limits with a small feasibility margin, ensuring that the reference remains a feasible solution without exposing its exact simulation results. The resulting prompt specifies the geometry semantics, coarse design bounds, boundary and loading conditions, and physical requirements, and asks the

Table 1: **Quantitative comparison on the full evaluation set of 3,373 tasks.** All results are reported as success rates (%). Bold values indicate the best result among different settings of the same model.

Model	Setting	Valid CAD↑	BBox↑	CAE↑	Physical Constraints↑			Joint↑
					Disp.	Stress	Physics	
Claude Opus 5	Direct	73.82	42.31	70.71	46.52	49.87	8.42	7.29
GPT-5.6 Sol	Direct	62.41	35.61	56.69	46.58	35.75	16.87	15.51
DeepSeek-V4-Flash-Ascend	Direct	63.33	50.52	60.45	39.79	38.36	10.85	9.28
Qwen3-4B	Zero-shot	18.56	11.41	17.55	12.24	13.93	1.22	0.74
	SFT	94.25	77.73	91.49	63.09	63.27	39.19	36.64
	GRPO	**99.14**	**91.25**	**96.92**	**84.44**	**81.83**	**57.40**	**55.71**
Ministral-3-3B	Zero-shot	1.27	0.62	1.07	0.68	0.65	0.24	0.12
	SFT	95.55	78.33	92.68	64.63	66.50	40.50	37.36
	GRPO	**98.99**	**94.22**	**97.45**	**84.32**	**81.20**	**59.29**	**58.05**
SmolLM3-3B	Zero-shot	6.91	2.91	6.55	5.37	5.90	0.65	0.18
	SFT	93.92	74.98	90.31	58.85	60.72	31.28	28.73
	GRPO	**98.99**	**91.31**	**96.80**	**77.47**	**76.28**	**45.75**	**44.56**
StarCoder2-3B	Zero-shot	10.94	3.91	10.29	8.81	8.98	1.04	0.56
	SFT	93.74	74.50	90.16	56.95	58.43	28.94	26.33
	GRPO	**98.96**	**92.05**	**97.15**	**76.67**	**74.12**	**43.14**	**41.65**
Qwen2.5-3B	Zero-shot	1.45	0.47	1.36	1.01	0.95	0.21	0.00
	SFT	90.07	65.16	86.45	55.68	57.75	25.32	21.76
	GRPO	**98.55**	**88.50**	**96.32**	**74.47**	**70.53**	**40.94**	**39.49**

model to generate a feasible design while minimizing mass. Detailed construction and sanitization rules are provided in the Appendix.

After CAD–CAE annotation, 77,067 designs successfully pass the complete pipeline. We divide them into disjoint SFT and GRPO subsets. The SFT subset contains 11,024 samples paired with sanitized reference CadQuery programs, whereas the GRPO subset contains 66,043 tasks without target programs and is optimized solely through CAD–CAE verification.

4.2 Implementation Details

From the 66,043 samples allocated to RL, we reserve 3,373 samples as a held-out evaluation set and use the remaining 62,670 prompts for GRPO training. The held-out samples are excluded from all policy updates and are used consistently for evaluating the SFT and GRPO checkpoints. All training and CAD–CAE evaluation experiments are conducted on a NVIDIA H800 GPU.

For GRPO, we use a training batch size of 4 prompts. For each prompt, the policy generates 4 independent completions, resulting in 16 rollout samples per batch. The four completions corresponding to the same prompt form a GRPO group and are evaluated independently by the deterministic CAD–CAE verifier described in Sec. 3.2. Their verification outcomes are then converted into the hierarchical advantages introduced above. We use a learning rate of 5×10^{-7} for policy optimization, with a sampling temperature of 0.9 and top-p of 0.95. For the hierarchical advantage, we instantiate equation 5 with $\lambda_L = 1.0$, $\lambda_w = 0.25$ and $\beta = 0.01$ in equation 6.

4.3 Evaluation Metrics

We evaluate generated designs using seven metrics. **Valid CAD** measures whether the generated program successfully produces a valid CAD solid. **BBox** measures whether all three bounding-box dimensions satisfy the specified design bounds. **CAE** denotes successful finite-element analysis with valid physical measurements. **Displacement** (Disp.) and **Stress** report the pass rates for the prescribed maximum-displacement and stress constraints, respectively. **Physics** measures simultaneous satisfaction of the displacement, stress, and mass constraints. Finally, **Joint** measures whether both the geometric requirements and all physical constraints are satisfied.

Table 2: **Ablation of hierarchical reinforcement learning.** Comparison of direct hierarchical GRPO, non-hierarchical GRPO, and staged training objectives.

Training Strategy	Stage	Valid CAD ↑	BBox ↑	CAE ↑	Physics ↑	Joint ↑	Code Tokens ↓
Baseline	Zero-shot	1.45	0.47	1.36	0.21	0.00	1002.41
	SFT	90.07	65.16	86.45	25.32	21.76	201.06
Direct	Non-hierarchical GRPO	98.40	60.90	83.60	23.40	18.80	199.78
Direct	Hierarchical GRPO	98.55	88.50	96.32	40.94	39.49	195.46
Training Objective	→ Geometry GRPO	95.29	72.37	91.55	25.88	22.86	191.60
	→ Physics GRPO	94.72	75.27	91.40	30.33	27.63	188.03
	→ Hierarchical GRPO	**99.23**	**92.05**	**97.45**	**43.34**	**42.01**	**184.85**

4.4 Comparison with the State of the Art

We compare PhyCAD with three strong proprietary models under direct prompting, including Claude Opus 5, GPT-5.6 Sol, and DeepSeek-V4-Flash-Ascend. We further evaluate five open-source models under SFT and GRPO, with zero-shot results reported where applicable, to examine the effectiveness of our training pipeline across different model families.

Table 1 and Fig. 4 show a consistent progression across training stages. SFT substantially improves CAD validity and CAE success, while GRPO further improves geometric and physical constraint satisfaction across all evaluated models. In particular, Joint success increases from 21.76%–37.36% after SFT to 39.49%–58.05% after GRPO, with corresponding gains in displacement and stress satisfaction. The trained models also achieve substantially higher success on the complete set of design requirements than proprietary models under direct prompting. These results indicate that CAD–CAE verification provides effective supervision beyond program validity and guides generation toward designs that better satisfy geometric and physical constraints.

4.5 Ablation on hierarchical learning.

Table 2 evaluates the effect of the hierarchical objective and staged optimization. Starting from the same SFT model, non-hierarchical GRPO improves CAD validity to 98.40% but reduces joint feasibility from 21.76% to 18.80%. In contrast, direct hierarchical GRPO achieves 98.55% valid CAD and 39.49% joint feasibility, nearly doubling the feasible designs relative to SFT. This indicates the gain comes from explicitly organizing feedback by feasibility level rather than from GRPO alone.

The staged results further show that optimizing geometric or physical objectives independently is insufficient: Geometry and Physics reach only 22.86% and 27.63% joint feasibility, respectively, improving over SFT by only 1.10 and 5.87 percentage points. Joint feasibility rises to 42.01% only when the explicit Hierarchical objective is introduced. This is particularly important for physical feasibility, where satisfying the prescribed stress and displacement thresholds requires the model to prioritize constraint satisfaction before objective refinement. Although staged optimization yields a further 2.52-point gain over direct hierarchical GRPO, it requires additional RL stages and substantially higher training cost.

5 Discussion

Conclusion. Our work establishes a physics-aware Text-to-CAD setting that connects executable CAD generation with deterministic CAE verification. By combining a physics-aware dataset, a CAD–CAE verifier, and hierarchical reinforcement learning, the framework improves geometric and physical compliance across multiple models. These results demonstrate the potential of incorporating verifiable engineering feedback into CAD generation beyond geometric reconstruction.

Limitation and Future Work. The current study has a few scope limitations. First, the prompts mainly encode physical constraints, while broader engineering intent is only weakly represented. Second, the dataset does not systematically cover material variations, though the same verification pipeline can be extended to different material settings. Future work could further introduce local hard dimensional constraints to enforce critical engineering requirements and reduce ambiguity in the generated geometry.

AI USE STATEMENT

Generative AI tools were used to assist with code development and debugging, experimental analysis, and the writing and editing of this manuscript. All AI-assisted code, analyses, and manuscript content were reviewed and verified by the authors. The authors take full responsibility for the methodology, experimental results, claims, and final content of this work.

REPRODUCIBILITY STATEMENT

We provide the implementation and experimental details necessary for reproducing our results in the main paper and Appendix, including the model formulation, optimization settings, dataset configurations, and evaluation protocols. Additional implementation details and experimental results are provided in the Appendix. Anonymous source code is provided in the supplementary material.

REFERENCES

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Haider Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. Generating cad code with vision-language models for 3d designs. In *International Conference on Learning Representations*, volume 2025, pp. 52236–52262, 2025.

Akshay Badagabettu, Sai Sravan Yarlagadda, and Amir Barati Farimani. Query2cad: Generating cad models using natural language queries. *arXiv preprint arXiv:2406.00144*, 2024.

Elias Berger, Muhammad Usama, Jan Mehlstäubl, Bernhard Saske, and Kristin Paetzold-Byhain. Physics-in-the-loop: A hybrid agentic architecture for validated cad engineering design. *arXiv preprint arXiv:2605.19717*, 2026.

Yuxuan Chen, Xu Zhu, Hua Zhou, and Zhuyin Ren. Metaopenfoam: an llm-based multi-agent framework for cfd. *arXiv preprint arXiv:2407.21320*, 2024.

Liyuan Deng, Shujian Deng, Yongkang Chen, Yongkang Dai, Zhihang Zhong, Linyang Li, Xiao Sun, Yilei Shi, and Huaxi Huang. Cosmo-agent: Tool-augmented agent for closed-loop optimization, simulation, and modeling orchestration. *arXiv preprint arXiv:2604.05547*, 2026.

Prashant Govindarajan, Davide Baldelli, Jay Pathak, Quentin Fournier, and Sarath Chandar. Cadmium: Fine-tuning code language models for text-driven sequential cad design. *arXiv preprint arXiv:2507.09792*, 2025.

Yandong Guan, Xilin Wang, Ximing Xing, Jing Zhang, Dong Xu, and Qian Yu. Cad-coder: Text-to-cad generation with chain-of-thought and geometric reward. *Advances in Neural Information Processing Systems*, 2025.

Shifu Hou, Rick Johnson, Ramandeep Makhija, Lingwei Chen, and Yanfang Ye. Autofea: Enhancing ai copilot by integrating finite element analysis using large language models with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 24078–24085, 2025.

Mohammad S Khan, Sankalp Sinha, Talha U Sheikh, Didier Stricker, Sk A Ali, and Muhammad Z Afzal. Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. *Advances in Neural Information Processing Systems*, 37:7552–7579, 2024.

Maksim Kolodiazhnyi, Denis Tarasov, Dmitrii Zhemchuzhnikov, Alexander Nikulin, Ilya Zisman, Anna Vorontsova, Anton Konushin, Vladislav Kurenkov, and Danila Rukhovich. cadrille: Multi-modal cad reconstruction with reinforcement learning. *arXiv preprint arXiv:2505.22914*, 2025.

Jiahao Li, Weijian Ma, Xueyang Li, Yunzhong Lou, Guichun Zhou, and Xiangdong Zhou. Cad-llama: leveraging large language models for computer-aided design parametric 3d model generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18563–18573. IEEE, 2025a.

Jiahao Li, Yusheng Luo, Yunzhong Lou, and Xiangdong Zhou. Recad: Reinforcement learning enhanced parametric cad model generation with vision-language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 6190–6198, 2026a.

Peiji Li, Jiasheng Ye, Yongkang Chen, Yichuan Ma, Zijie Yu, Kedi Chen, Ganqu Cui, Haozhan Li, Jiacheng Chen, Chengqi Lyu, et al. Internbootcamp technical report: Boosting llm reasoning with verifiable task scaling. *arXiv e-prints*, pp. arXiv–2508, 2025b.

Xingang Li, Yuewan Sun, and Zhenghui Sha. Llm4cad: Multimodal large language models for three-dimensional computer-aided design generation. *Journal of Computing and Information Science in Engineering*, 25(2):021005, 2025c.

Xueyang Li, Jiahao Li, Yu Song, Yunzhong Lou, and Xiangdong Zhou. Seek-cad: A self-refined generative modeling for 3d parametric cad using local inference via deepseek. In *International Conference on Learning Representations*, volume 2026, pp. 94278–94298, 2026b.

Dimitrios Mallis, Ahmet Serda Karadeniz, Sebastian Cavada, Danila Rukhovich, Niki Foteinopoulou, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-assistant: Tool-augmented vllms as generic cad task solvers. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7284–7294. IEEE, 2025.

Tobias Preintner, Weixuan Yuan, Adrian König, Thomas Bäck, Elena Raponi, and Niki Van Stein. Evocad: Evolutionary cad code generation with vision language models. In *2025 IEEE 37th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 504–511. IEEE, 2025.

Yupeng Qi, Ran Xu, and Xu Chu. Feagpt: an end-to-end agentic-ai for finite element analysis. *arXiv preprint arXiv:2510.21993*, 2025.

Feiwei Qin, Shichao Lu, Junhao Hou, Changmiao Wang, Meie Fang, and Ligang Liu. Drawing2cad: Sequence-to-sequence learning for cad generation from vector drawings. In *Proceedings of the 33rd ACM international conference on multimedia*, pp. 10573–10582, 2025.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*, 2020.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Guijin Son, Jehyun Park, Seyeon Park, Sunghee Ahn, and Youngjae Yu. Self-improving cad generation agents with finite element analysis as feedback. *arXiv preprint arXiv:2605.17448*, 2026.

Yuewan Sun, Xingang Li, and Zhenghui Sha. Large language models for computer-aided design fine tuned: Dataset and experiments. *Journal of Mechanical Design*, 147(4):041710, 2025.

Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4): 1–24, 2021.

Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. Joinable: Learning bottom-up assembly of parametric cad joints. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15828–15839. IEEE, 2022.

Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6752–6762. IEEE, 2021.

Haoyang Xie and Feng Ju. Text-to-cadquery: A new paradigm for cad generation with scalable large model capabilities. *arXiv preprint arXiv:2505.06507*, 2025.

Zhaoyue Xu, Long Wang, Chunyu Wang, Yixin Chen, Qingyong Luo, Hua-Dong Yao, Shizhao Wang, and Guowei He. Cfdagent: A language-guided, zero-shot multi-agent system for complex flow simulation. *Physics of Fluids*, 37(11), 2025.

Zhe Yuan, Jianqi Shi, and Yanhong Huang. Openecad: An efficient visual language model for editable 3d-cad design. *Computers & Graphics*, 124:104048, 2024.

Ling Yue, Nithin Somasekharan, Tingwen Zhang, Yadi Cao, Zhangze Chen, Shimin Di, and Shaowu Pan. Foam-agent: Towards automated intelligent cfd workflows. *arXiv preprint arXiv:2505.04997*, 2025.

Licheng Zhang, Bach Le, Naveed Akhtar, Siew-Kei Lam, and Duc Ngo. Large language models for computer-aided design: A survey. *ACM Computing Surveys*, 58(9):1–39, 2026.

Tao Zhang, Zhenhai Liu, Yong Xin, and Yongjun Jiao. Mooseagent: A llm based multi-agent framework for automating moose simulation. *arXiv preprint arXiv:2504.08621*, 2025.

A Implementation of Dataset Generation

This appendix provides implementation details for the dataset-generation procedure introduced in Sec. 4.1. We focus on the construction of physical conditions, inverse-design constraints, and dimension-free geometry semantics.

A.1 Reference CAD–CAE Annotation and Physical Conditions

Each source CadQuery program is executed and retained only if it produces exactly one valid connected solid that can be processed by the downstream CAD–CAE pipeline. We analyze all retained designs using three-dimensional small-strain linear elasticity in SI units, with isotropic structural steel ($E = 2.1 \times 10^{11}$ Pa, $\nu = 0.3$, and $\rho = 7850\ \mathrm{kg/m^3}$). For each successfully simulated design, we record the maximum nodal displacement $\delta_{\max}$, the volume-weighted 99-th percentile element von Mises stress σ_{P99}, and the total mass m.

Because the source geometries vary substantially in shape and orientation, the fixture and loading regions are determined automatically. We compute the centroid of every CAD surface and select the pair of surface centroids with the largest Euclidean separation. The normalized direction between this pair defines the functional analysis axis. All surface centroids are projected onto this axis, and surfaces whose projections lie within 5% of either end of the projected range form the two terminal surface groups. This grouping allows multiple coplanar or fragmented terminal surfaces to be handled consistently.

The negative terminal group is fully fixed ($u_x = u_y = u_z = 0$), while a uniform traction is applied to the positive terminal group such that its integrated resultant force is 100 kN along the positive analysis axis. The resulting analysis axis, fixture definition, and loading condition are explicitly included in the model prompt.

A.2 Inverse-Design Task Construction

Each successfully analyzed reference is converted into an inverse-design task while withholding its exact dimensions and simulation measurements.

Coarse design domain. For each global bounding-box extent e, we expose only the enclosing interval between adjacent powers of two,

$$B(e) = \left[2^{\lfloor \log_2 e \rfloor}, 2^{\lfloor \log_2 e \rfloor + 1}\right]. \tag{7}$$

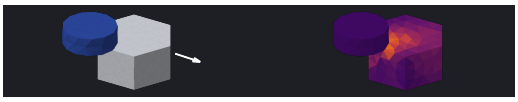
The model therefore receives a coarse admissible range along each axis rather than the dimensions of the reference design.

Performance requirements. The displacement, stress, and mass limits are derived independently from $\delta_{\max}$, σ_{P99}, and m. Each reference measurement is multiplied by 1.10 and rounded upward to two significant digits to form the corresponding upper bound. The margin keeps the reference design feasible under small numerical variations from CAD processing, meshing, and FEM evaluation, while rounding avoids exposing its exact simulation measurements.

A.3 Geometry Semantic Construction

The geometry specification must preserve the intended design structure without revealing the parameters of the reference CAD. We construct this specification from both the source textual description and the CadQuery operation sequence.

Numerical redaction. We remove numerical information that may reveal the reference geometry, including explicit dimensions, angles, quantities with units, and spelled-out numerical values. We retain non-numerical information describing the shape class, topology, feature types and counts, and spatial or compositional relationships. Thus, for example, the existence and relative placement of a cylindrical feature are preserved while its diameter and height are withheld.



Generate executable CadQuery Python code for a physics-conditioned structural design.

GEOMETRY SEMANTICS

The design involves creating a hexagonal prism and a cylindrical top. The hexagonal prism has a flat base and six sides, each with an angle. It measures a value within the design envelope in length, a value within the design envelope in width, and a value within the design envelope in height. The cylindrical top is centered on the hexagonal prism and has a diameter equal to the width of the hexagonal prism's base. The cylinder itself measures a value within the design envelope in both length and width and a value within the design envelope in height. These two parts are combined to form the final shape, which features a hexagonal base and a circular top.
Preserve the described shape class, topology, feature count, and part relationships; choose all exact dimensions within the bounds below.

DESIGN DOMAIN

- Produce exactly one connected solid.
- Global bounding-box extents: x in [0.5, 1] m; y in [0.5, 1] m; z in [0.25, 0.5] m.
- Functional analysis-axis unit vector in global coordinates: [0.702, 0.574, -0.422].

PHYSICAL CONDITIONS

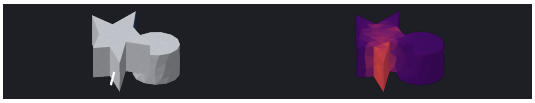
- Analysis: three-dimensional, small-strain linear statics in SI units.
- Material: isotropic linear-elastic structural steel with Young's modulus 2.1e+11 Pa, Poisson ratio 0.3, and density 7850 kg/m^3.
- Fixture: on the negative terminal CAD-surface centroid-projection group along the analysis axis, constrain ux = uy = uz = 0 m.
- Load: on the positive terminal CAD-surface centroid-projection group, apply uniform traction whose integrated total force vector is [70156.8767, 57408.6433, -42216.8252] N.

PERFORMANCE REQUIREMENTS

- Maximum nodal displacement <= 1.8e-05 m.
- Volume-weighted 99th-percentile element Von Mises stress <= 4700000 Pa.
- Total mass <= 570 kg.
- Minimize mass while satisfying every constraint.

OUTPUT

Return only executable Python code using CadQuery. Assign the final solid to a variable named result or assembly.



Generate executable CadQuery Python code for a physics-conditioned structural design.

GEOMETRY SEMANTICS

Construct a star-shaped object and a cylindrical object that will be combined to form the final design. The star-shaped object has a length of a bounded value, a width of a bounded value, and a height of a bounded value. The cylindrical object, which is added to the star-shaped object, has a length of a bounded value, a width of a bounded value, and the same height of a bounded value. The star-shaped object is created by drawing a complex closed loop and extruding it, while the cylindrical object is formed by extruding a circular sketch. Both objects are aligned and combined to create the final composite shape.
Preserve the described shape class, topology, feature count, and part relationships; choose all exact dimensions within the bounds below.

DESIGN DOMAIN

- Produce exactly one connected solid.
- Global bounding-box extents: x in [0.25, 0.5] m; y in [0.25, 0.5] m; z in [0.125, 0.25] m.
- Functional analysis-axis unit vector in global coordinates: [0.388, -0.922, 0].

PHYSICAL CONDITIONS

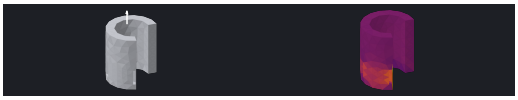
- Analysis: three-dimensional, small-strain linear statics in SI units.
- Material: isotropic linear-elastic structural steel with Young's modulus 2.1e+11 Pa, Poisson ratio 0.3, and density 7850 kg/m^3.
- Fixture: on the negative terminal CAD-surface centroid-projection group along the analysis axis, constrain ux = uy = uz = 0 m.
- Load: on the positive terminal CAD-surface centroid-projection group, apply uniform traction whose integrated total force vector is [38809.806, -92161.8086, 0] N.

PERFORMANCE REQUIREMENTS

- Maximum nodal displacement <= 1.5e-05 m.
- Volume-weighted 99th-percentile element Von Mises stress <= 14000000 Pa.
- Total mass <= 89 kg.
- Minimize mass while satisfying every constraint.

OUTPUT

Return only executable Python code using CadQuery. Assign the final solid to a variable named result or assembly.



Generate executable CadQuery Python code for a physics-conditioned structural design.

GEOMETRY SEMANTICS

The first part involves creating a cylindrical object with a complex curved profile. The object has a length of approximately a bounded value, a width of about a bounded value, and a height of a bounded value. The second part consists of a pair of rectangular prisms with rounded edges. These prisms are positioned adjacent to each other and have a length of around a bounded value, a width of about a bounded value, and a height of a bounded value. Both parts are designed to fit together to form a larger, integrated structure.
Preserve the described shape class, topology, feature count, and part relationships; choose all exact dimensions within the bounds below.

DESIGN DOMAIN

- Produce exactly one connected solid.
- Global bounding-box extents: x in [0.25, 0.5] m; y in [0.25, 0.5] m; z in [0.5, 1] m.
- Functional analysis-axis unit vector in global coordinates: [0, 0, 1].

PHYSICAL CONDITIONS

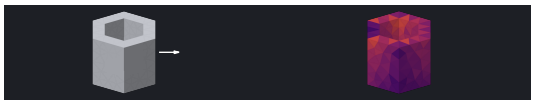
- Analysis: three-dimensional, small-strain linear statics in SI units.
- Material: isotropic linear-elastic structural steel with Young's modulus 2.1e+11 Pa, Poisson ratio 0.3, and density 7850 kg/m^3.
- Fixture: on the negative terminal CAD-surface centroid-projection group along the analysis axis, constrain ux = uy = uz = 0 m.
- Load: on the positive terminal CAD-surface centroid-projection group, apply uniform traction whose integrated total force vector is [0, 0, 100000] N.

PERFORMANCE REQUIREMENTS

- Maximum nodal displacement <= 6.3e-06 m.
- Volume-weighted 99th-percentile element Von Mises stress <= 3900000 Pa.
- Total mass <= 270 kg.
- Minimize mass while satisfying every constraint.

OUTPUT

Return only executable Python code using CadQuery. Assign the final solid to a variable named result or assembly.



Generate executable CadQuery Python code for a physics-conditioned structural design.

GEOMETRY SEMANTICS

The design consists of two main parts: a hexagonal box and a hexagonal plate. Both parts start with a new coordinate system set at the origin with no rotation. The first part involves drawing a hexagonal shape with an inner hexagonal cutout, scaling it down by a bounded proportion, and then extruding it a value within the design envelope to form a solid hexagonal box. This box has a length of approximately a bounded value, a width of a bounded value, and a height of a bounded value. The second part also starts with a hexagonal shape, which is scaled down by a bounded proportion and then extruded a value within the design envelope in the opposite direction to form a thin hexagonal plate. This plate has the same length and width as the box but a much smaller height of a bounded value. The two parts are designed to fit together, with the plate adding a base to the box.
Preserve the described shape class, topology, feature count, and part relationships; choose all exact dimensions within the bounds below.

DESIGN DOMAIN

- Produce exactly one connected solid.
- Global bounding-box extents: x in [0.5, 1] m; y in [0.25, 0.5] m; z in [0.25, 0.5] m.
- Functional analysis-axis unit vector in global coordinates: [0.866, 0.500, 0].

PHYSICAL CONDITIONS

- Analysis: three-dimensional, small-strain linear statics in SI units.
- Material: isotropic linear-elastic structural steel with Young's modulus 2.1e+11 Pa, Poisson ratio 0.3, and density 7850 kg/m^3.
- Fixture: on the negative terminal CAD-surface centroid-projection group along the analysis axis, constrain ux = uy = uz = 0 m.
- Load: on the positive terminal CAD-surface centroid-projection group, apply uniform traction whose integrated total force vector is [86603.1755, 49998.9, 0] N.

PERFORMANCE REQUIREMENTS

- Maximum nodal displacement <= 8.5e-06 m.
- Volume-weighted 99th-percentile element Von Mises stress <= 3000000 Pa.
- Total mass <= 510 kg.
- Minimize mass while satisfying every constraint.

OUTPUT

Return only executable Python code using CadQuery. Assign the final solid to a variable named result or assembly.

Figure 5: **Representative physics-aware Text-to-CAD tasks.** Each example shows the private reference geometry and its FEM stress field together with the corresponding model-visible task. Exact reference dimensions and simulation measurements are withheld; the model receives dimension-free geometry semantics, coarse design bounds, physical conditions, and displacement, stress, and mass limits. It must generate an executable CadQuery design satisfying these requirements while minimizing mass.

Operation-derived semantics. Because textual descriptions do not always capture the CAD construction consistently, we additionally parse the CadQuery program and map its modeling operations to dimension-free geometric features. Numerical operation arguments are discarded. Representative mappings are shown in Table 3.

The sanitized textual description and operation-derived features are combined into the final geometry-semantic specification. It preserves the overall shape class, topology, feature composition, and relationships between components, while leaving their exact dimensions unspecified.

Table 3: Representative CadQuery operation-to-feature mappings used to construct dimension-free geometry semantics. Numerical parameters are not exposed to the model.

CadQuery construction	Extracted geometry semantics
box	rectangular-prismatic feature
cylinder	cylindrical feature
polygonal sketch + extrude	polygonal-prismatic feature
curved sketch + extrude	curved-profile feature
revolve	revolved feature
cut	material-removal relation
hole	hole or through-feature
fillet	rounded-edge feature
shell	hollow or shell construction
union / composition	part-composition relation

Consequently, the reference CAD remains one feasible realization of the task rather than a uniquely specified target geometry.

Representative tasks produced by the complete pipeline are shown in Fig. 5, where the reference CAD and FEM results are included only for illustration and are not provided to the model.

B HIERARCHICAL BALANCED GRPO

Algorithm 1 summarizes the training procedure of our Hierarchical Balanced GRPO.

Algorithm 1 Hierarchical Balanced GRPO.

Require: Policy π_θ, reference policy π_{ref}, prompts $\mathcal{Q}$, group size G, weights $\lambda_L, \lambda_w, \beta$

for each training iteration **do**
  Sample a prompt batch $\mathcal{B} \subset \mathcal{Q}$
  for each prompt $q \in \mathcal{B}$ **do**
    Sample $\{o_i\}_{i=1}^{G} \sim \pi_\theta(\cdot \mid q)$
    Verification: evaluate each o_i with the CAD–CAE verifier to obtain level $L_i \in \{0,1,2,3\}$ and level-specific scores $\{s_{ik}\}_{k=1}^{K_{L_i}}$
      $L_i = 0$: invalid CAD; $L_i = 1$: geometry failure; $L_i = 2$: physics failure; $L_i = 3$: fully feasible
    Between-level advantage: $\bar{L} = \frac{1}{G}\sum_{j=1}^{G} L_j, \quad A_i^{\text{level}} = L_i - \bar{L}$
    Within-level PeerRank: initialize $\widetilde{A}_i = 0$ for all i
    for $\ell = 0, 1, 2, 3$ **do**
      $I_\ell = \{i : L_i = \ell\}$
      if $|I_\ell| > 1$ **then**
        Compute tie-aware centered ranks r_{ik} of $\{s_{ik}\}_{i \in I_\ell}$ for $k = 1, \ldots, K_\ell$
        $\widetilde{A}_i = \sum_{k=1}^{K_\ell} w_{\ell k} r_{ik}, \quad \forall i \in I_\ell$
      end if
    end for
    Hierarchical advantage: $A_i = \lambda_L A_i^{\text{level}} + \lambda_w \widetilde{A}_i$
  end for
  Update θ using DAPO-style GRPO: $\mathcal{L} = \mathcal{L}_{\text{DAPO}}(\theta; A) + \beta D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$
end for
return Optimized policy π_θ