2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #51434

Computation-Centered Learning for Conceptual Mastery in Mechanical Design

Dr. Zubaer Hossain, Texas A&M University

Dr. Hossain received his PhD degree in Mechanical Engineering from the University of Illinois at Urbana-Champaign (UIUC). Prior to joining Texas A&M as an associate professor of instruction, he was an assistant professor at UD and served as a postdoctoral scholar at California Institute of Technology. He also worked as a visiting scientist at Lawrence Livermore National Laboratory and as a visiting scholar at Johns Hopkins University. His research and teaching interests include mechanics and physics of deformation and fracture in semiconductors, heterogeneous materials, and composites. He explores the role of material and structural anisotropy and asymmetry in controlling effective properties at multiple scales. Improving domestic students' success and participation in higher education and research drives his teaching and mentoring ambitions.

Computation-Centered Learning for Conceptual Mastery in Mechanical Design

Abstract

Teaching mechanical design problems that involve competing stiffness and fatigue requirements challenges students to integrate multiple mechanics concepts such as stress concentration, deflection compatibility, endurance limits, and material sensitivity into a coherent design rationale. These challenges are fundamentally computational in nature, requiring students to reason across coupled relationships rather than apply isolated formulas. Traditional instruction often fragments these ideas, leading to procedural knowledge that does not readily transfer when constraints interact. This work introduces a computation-centered, AI-augmented learning module designed to support integrated reasoning and verification in multi-constraint mechanical design, demonstrated through stepped shaft design as a canonical example.

The module was implemented in two junior-level courses, MEEN 305 Solid Mechanics and MEEN 368 Solid Mechanics in Mechanical Design. Browser-based computational notebooks, such as JupyterLite, form the backbone of the learning environment, allowing students to manipulate geometry and loading parameters, visualize system response, and compute stiffness and fatigue metrics. Generative AI, exemplified by ChatGPT, is used as a complementary computational support that prompts explanation, interpretation of trade-offs, and articulation of assumptions rather than providing authoritative solutions. AI accuracy is addressed through explicit verification practices, including parameter sweeps, cross-checking against computational outputs, and consistency with expected mechanics trends.

Student learning is examined through design artifacts, reflective responses, and guided recitation discussions, with emphasis on integrated reasoning, evidence-based justification, and verification behavior rather than numerical outcomes alone. Preliminary evidence indicates that students more effectively explain competing constraints, interpret design sensitivity, and justify design decisions using both intuition and computational evidence. While demonstrated through stepped shafts in MEEN 368, the computation-centered instructional framework is applicable to a broad class of mechanics and mechanical design problems involving coupled constraints.

1 Introduction

Mechanical design problems that involve competing stiffness and fatigue requirements require students to integrate multiple mechanics concepts into a coherent design rationale. These problems demand reasoning across bending moment distribution, deflection and slope limits, stress concentration at geometric discontinuities, and fatigue behavior under combined loading. Such integration is inherently computational, as changes in geometry or loading propagate simultaneously through multiple governing relationships, a feature that prior work has shown to support deeper conceptual understanding when computation is used for exploration rather than procedural execution [1, 2]. In practice, the engineering question is not merely whether a stress is below a limit or whether a deflection is acceptable, but why a change that improves one constraint can deteriorate another, and how to defend a final design choice under competing requirements.

Stepped shafts provide a canonical example of this challenge because they compress multiple core mechanics ideas into a single, familiar artifact. A typical stepped shaft design task requires students to reason about global system response, such as bending moments and deflections, while simultaneously accounting for local effects such as stress concentrations at shoulders and keyways and fatigue safety under combined bending and torsion. Although the specific geometry is simple, the interaction among constraints is nontrivial, making stepped shafts an effective context for studying how students integrate computation, mechanics reasoning, and design judgment.

This work introduces an instructional module, named Computation-Integrated Learning and Teaching (CILT), intended to strengthen conceptual integration through computation, using stepped shaft design as an illustrative case. The module blends two complementary computational supports. First, browser-based computational notebook, JupyterLite, allows students to manipulate geometry and loading parameters, visualize bending moment and deflection responses, and compute stiffness and fatigue metrics in a transparent and reproducible manner. Second, generative AI is used as a conversational scaffold that prompts students to articulate assumptions, interpret computational outputs, and explain trade-offs in their own words. Generative AI is not positioned as a solver, but as a tool that supports explanation and reflection alongside computation.

The CILT instructional design is motivated by two recurring classroom challenges. The first is that students often treat stiffness checks and fatigue checks as independent requirements rather than as coupled constraints mediated by geometry and loading. The second is that students may over-trust a single tool, whether that tool is a hand-solution template, a computational script, a finite element contour plot, or a generative AI response. The module therefore emphasizes verification as a central learning objective. Students are required to cross-check results across representations, perform parameter sweeps, and defend design decisions using mechanics reasoning rather than tool authority. Student artifacts illustrate both the potential and the need for this scaffolding, as students can construct sophisticated Python workflows that compute reactions, generate moment and deflection curves, and evaluate fatigue criteria, yet those workflows also encode modeling assumptions and implementation choices that must be interpreted and validated.

2 Pedagogical framework and role of computation

The developed instructional module adopts a computation-centered view of learning in which executable models function as primary representations for reasoning rather than as post-solution verification tools. In this framework, code is treated as a medium for conceptual expression. Students must explicitly encode governing relationships, declare assumptions, and operationalize geometry–load interactions in a form that produces inspectable outputs. This requirement transforms abstract relationships into manipulable objects, enabling learners to interrogate scaling behavior, sensitivity, and governing constraints directly through controlled variation. Guided articulation and reflection are used to support this process, but computation remains the central epistemic reference against which reasoning is tested. This framing aligns with computational thinking perspectives in which abstraction, parameterization, and systematic variation are treated as disciplinary practices rather than technical skills [2]. In CILT module, these practices are embedded within mechanics reasoning rather than treated as separate programming objectives.

Stepped shaft design provides a concrete context for developing this expertise because it naturally couples global response with local discontinuity effects within a single artifact. A central conceptual challenge is understanding how geometry mediates competing responses through distinct scaling relationships. For a circular cross-section, bending stress and deflection scale differently with diameter,

$$\sigma_b = \frac{32M}{\pi d^3} \quad \text{and} \quad \delta \propto \frac{1}{d^4}, \tag{1}$$

so a single geometric change can improve one constraint while disproportionately affecting another. The act of writing and modifying Python code introduces a productive cognitive constraint. Variables must be defined, relationships must be explicitly structured, and units and dependencies must be made consistent for the model to execute. This process externalizes reasoning that is often implicit in hand calculations. When students vary diameter, load position, or material properties, the computational model forces immediate reconciliation between intuition and system response. Discrepancies between predicted and computed trends serve as triggers for model revision, promoting deeper structural understanding rather than procedural execution.

In addition, local discontinuities introduce stress concentration effects that can shift the governing location and alter combined-stress measures used in design checks. For stiffness calculation, the governing curvature expression for the simply supported stepped shaft (step at $x = a$, concentrated load F_1 applied at $x = b$) is given by

$$\frac{M(x)}{I(x)} = \frac{R_1}{I_1}\langle x\rangle^1 - \frac{F_1}{I_2}\langle x-b\rangle^1 + \left(\frac{1}{I_1} - \frac{1}{I_2}\right)\left[M_a\langle x-a\rangle^0 + V_a\langle x-a\rangle^1\right] \tag{2}$$

Encoding the inertia change directly within the curvature expression makes geometric discontinuity an explicit computational object rather than an implicit piecewise condition. Students can modify I_1, I_2, or the step location a and immediately observe how curvature, slope, and deflection fields respond. This manipulability reinforces the idea that stiffness behavior is governed by distributed interactions rather than isolated formulas.

The CILT module is designed to force students to reconcile these interactions by pairing symbolic relationships with computational exploration and interpretation. Short computational fragments are used to expose parametric dependence explicitly, including combined normal and shear stresses with stress concentration and von Mises equivalent stress.

The stepped shaft configuration analyzed in this module is illustrated in Figure 1. The assembly includes bearing supports, a gear-mounted load application point, and geometric discontinuities at shaft shoulders. This physical context makes explicit the locations at which reactions, bending moments, deflection limits, and stress concentrations interact. By grounding the computational model in a recognizable mechanical system, students can map abstract curvature and stress expressions to specific geometric features, reinforcing the connection between symbolic representation and physical artifact.

Within this computation-centered framework, generative AI is intentionally constrained to function as an interpretive scaffold rather than a numerical solver. Executable models serve as the primary evidentiary basis, while AI supports articulation and interrogation of that evidence. Students use AI interactions to interpret computational results, articulate assumptions, and reason about trade-

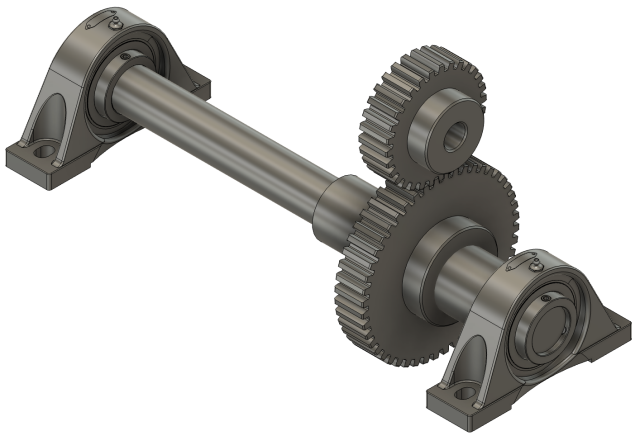

Figure 1: Three-dimensional CAD model of a shaft–gear assembly supported by pillow block bearings. A large spur gear mounted on the primary shaft meshes with a smaller gear, illustrating torque transmission and load transfer through the shaft–bearing system.

offs revealed through parameter variation. For example, students are prompted to explain why the governing location may shift to a shoulder when fatigue stress concentration is introduced through

$$K_f = 1 + q(K_t - 1), \tag{3}$$

and why a separate concentration factor for shear may be needed when torsion or transverse shear contributes to the stress state. This emphasis on articulation and explanation aligns with prior findings showing that eliciting self-explanations improves conceptual understanding and transfer, particularly in complex problem-solving contexts [3]. The AI does not compute concentration factors or select material parameters; instead, it prompts students to justify parameter choices and to connect them to material behavior, geometry, and loading. This design supports epistemic agency, defined here as the student's ability to validate their own computational models and thinking.

In contrast to traditional static problem sets in which geometry and loading are fixed, this module treats variation as a central learning mechanism. Students do not merely compute whether a design satisfies a constraint; they explore how satisfaction emerges or deteriorates under parametric change. This shift from answer-seeking to behavior-seeking reframes mechanical design as the study of interacting functional relationships rather than the application of isolated equations.

A central design requirement of the module is to guard against reliance on AI output as authoritative. This requirement is operationalized through prompt design and explicit verification practices. Prompts are written to elicit assumptions, reasoning steps, and expected qualitative trends rather than final numerical answers. Students are required to validate AI-supported claims by reproducing results in the computational environment, performing parameter sweeps, and checking consistency with expected mechanics trends. When discrepancies arise between AI statements and computational or analytical results, students document the inconsistency and explain how their understanding changed as a result. In this way, computation remains the reference point for credibility, with generative AI supporting articulation, reflection, and verification rather than replacing analytical reasoning. The following code fragment illustrates the type of parametric exploration

used by students to examine combined stress and stress concentration effects.

```
# JupyterLite-style Python fragment for parametric exploration
# Combined bending + torsion at a shoulder
# Purpose: expose parametric trends (not automate a full design)

import numpy as np
import matplotlib.pyplot as plt

# Example loading (students vary these)
M = 300.0       # bending moment at the shoulder, N·m
T = 180.0       # torque at the shoulder, N·m

# Diameter sweep (small diameter at the shoulder)
d_mm = np.linspace(18.0, 60.0, 43)
d = d_mm * 1e-3

# Fillet ratio sweep (students vary these)
rd = np.array([0.02, 0.05, 0.10])  # r/d values (small to moderate fillets)

# Notch sensitivity (illustrative; students justify choices)
q_sigma = 0.85
q_tau   = 0.80

def Kt_bending(rd_val):
    # Monotonic proxy: smaller r/d gives higher Kt
    # Replace with chart-based interpolation in the full notebook.
    return 1.2 + 1.6 * np.exp(-rd_val / 0.03)

def Kts_torsion(rd_val):
    # Monotonic proxy: smaller r/d gives higher Kts (typically < Kt for bending)
    # Replace with chart-based interpolation in the full notebook.
    return 1.1 + 1.2 * np.exp(-rd_val / 0.035)

plt.figure()

for rd_val in rd:
    Kt  = Kt_bending(rd_val)
    Kts = Kts_torsion(rd_val)

    # Fatigue-style concentration factors
    Kf  = 1.0 + q_sigma * (Kt  - 1.0)
    Kfs = 1.0 + q_tau   * (Kts - 1.0)

    # Nominal stresses for a solid circular shaft
    sigma_nom = (32.0 * M) / (np.pi * d**3)   # Pa
    tau_nom   = (16.0 * T) / (np.pi * d**3)   # Pa

    # Concentrated stresses at the shoulder
    sigma = Kf  * sigma_nom
    tau   = Kfs * tau_nom

    # von Mises equivalent stress for combined normal + shear
    sigma_vm = np.sqrt(sigma**2 + 3.0 * tau**2)

    # Normalize to compare trends across cases
    sigma_vm_norm = sigma_vm / sigma_vm[0]

    plt.plot(
        d_mm, sigma_vm_norm,
        label=f"r/d={rd_val:.2f}, Kt={Kt:.2f}, Kts={Kts:.2f}"
    )

plt.xlabel("shoulder diameter d (mm)")
plt.ylabel("normalized von Mises stress at shoulder")
plt.grid(True)
plt.legend()
```

```
plt.show()
```

This fragment illustrates several computation-centered learning features. First, diameter and geometry proxies are treated as manipulable parameters rather than fixed inputs, making sensitivity explicit. Second, stress concentration effects are encoded functionally, allowing students to observe how local discontinuities can shift governing behavior even when nominal stresses decrease. Third, normalization of equivalent stress enables comparative reasoning across geometric configurations, directing attention toward trend interpretation rather than absolute magnitude. The code is intentionally partial: it exposes structural relationships without automating full design selection, preserving the requirement that students interpret, justify, and validate results.

3 Course context and learning objectives

The CILT module was implemented in a junior-level mechanics and mechanical design context, with stepped shaft design serving as a canonical anchor problem. The instructional materials were integrated into an existing design project in which students design a multi-segment transmission shaft supported at bearings, subjected to transverse loading, and transmitting power at a specified rotational speed. The design task requires students to evaluate global system response, including reactions, bending moments, slopes, and deflections, while also accounting for local effects such as stress concentration at geometric discontinuities and combined stress states relevant to fatigue and equivalent stress measures.

The learning objectives for the module emphasize computation-supported integration and verification rather than isolated calculation. Students are expected to explain why stiffness and fatigue constraints compete and how geometry mediates that competition through distinct scaling relationships. Students are expected to identify and justify likely critical locations using both global loading response and local discontinuity effects, including the role of stress concentration and combined bending and shear. Students are expected to use computation to explore parametric sensitivity by varying geometry and loading and to interpret how those changes propagate through stress, deflection, and equivalent stress measures such as von Mises stress. Finally, students are expected to demonstrate verification behaviors, including cross-checking AI-supported explanations against computational results, performing parameter sweeps to confirm expected trends, and using limiting cases to assess the credibility of their conclusions. While stepped shaft design served as the anchor problem in MEEN 368, a different mechanics problem was used to implement the same CILT framework in MEEN 305.

The instructional architecture described next operationalizes these objectives through explicit computational, verification, and articulation practices embedded within the course workflow.

4 System architecture and verification practices

In the CILT module, outputs from all tools are treated as evidence that must be interpreted, interrogated, and verified rather than accepted as authoritative. This framing reflects long-standing views of engineering design as an evidence-based judgment process in which verification, interpretation, and justification are central to competent practice [4, 5]. The instructional design deliberately assigns distinct and complementary roles to the computational environment and to generative AI. Browser-based computation serves as the primary reference for numerical credibility by producing traceable calculations and visualizations that students can reproduce, inspect, and stress test through parametric variation. Generative AI is used to prompt explanation, surface assumptions, and support interpretation of trade-offs revealed through computation. This separation of roles is intended to cultivate engineering judgment by requiring students to justify why a result is credible and what physical mechanisms explain it.

The module uses a browser-based JupyterLite environment so students can run and modify Python notebooks without local installation. The use of browser-based computational notebooks aligns with prior work highlighting their value for interactive exploration, transparency, and reproducibility in engineering and computational science education [6, 7]. The notebooks implement a mechanics workflow aligned with the stepped shaft design project, including statics for support reactions, evaluation of shear and bending moment distributions, computation of slope and deflection, and evaluation of stress and fatigue metrics under combined loading. In addition to nominal stress calculations, students explore the effects of stress concentration and combined bending and shear through equivalent stress measures such as von Mises stress. Students vary geometric parameters such as shaft diameters and step locations, as well as loading parameters such as force magnitude and position, and observe how these changes propagate through global response, local stresses, and governing design checks. The feedback loop is intentionally short so that verification becomes a routine part of exploration rather than a final validation step performed at the end.

Verification is operationalized through three complementary practices. First, prompt design in the generative AI emphasizes assumptions, governing relationships, and expected qualitative trends rather than final numerical answers. Students are encouraged to ask for reasoning pathways, limiting cases, and consistency checks. Second, computational cross-checking is required. Students validate AI-supported claims by reproducing them in the notebook environment, performing parameter sweeps, and confirming that observed trends align with expected mechanics scaling relationships, including the relative sensitivity of stress, deflection, and equivalent stress measures to geometry. Third, triangulation across representations is encouraged. Students compare notebook outputs with simplified hand calculations on limiting cases and, when appropriate, with finite element results. When discrepancies arise, students identify whether the source is a modeling assumption, an implementation choice, or a misinterpretation of results, and they document how resolving the discrepancy altered their understanding.

Student artifacts illustrate why these verification practices are both necessary and instructionally valuable. Students are able to construct sophisticated scripts that compute reactions, generate bending moment and deflection curves, and evaluate stress and fatigue criteria, yet those scripts may also encode assumptions, overwritten parameters, or implicit choices whose implications are not immediately apparent without deliberate checking. The module treats these moments not as

errors to be corrected, but as opportunities to practice validation, interpretation, and judgment, which are central components of design competence.

5 Learning activities and student workflow

Learning activities are organized as a repeated cycle of prediction, computation, interpretation, and justification, with executable models serving as the structural backbone of the reasoning process. Rather than treating computation as a terminal calculation step, the workflow positions executable notebooks as a dynamic environment in which assumptions, parameters, and governing mechanisms can be iteratively tested and refined. Students begin with a baseline design configuration and make qualitative predictions about how changes in geometry or loading will affect stiffness, stress, and fatigue behavior.

For example, students may predict that increasing a segment diameter will reduce nominal bending stress and increase stiffness, while also anticipating that stress concentration at a shoulder may shift the governing location or alter the dominant stress measure. This initial prediction stage externalizes the student's mental model before any computational output is introduced. Because the subsequent computational environment makes parameter variation immediate and reversible, predictions are framed not as single guesses but as hypotheses about directional behavior. Students are encouraged to state expected trends (e.g., "stress should decrease with diameter as $1/d^3$") and anticipated governing transitions before observing outputs. This pre-commitment creates a reference against which computational evidence can challenge or confirm their reasoning. The distinct instructional roles assigned to computation, generative AI, and the student are summarized in Table 1, clarifying how each tool supports reasoning without displacing engineering judgment.

Students then engage with generative AI to support articulation and planning. The AI is prompted to ask clarifying questions, surface assumptions, and suggest verification checks rather than provide numerical results. These interactions are used to help students explain anticipated trade-offs at a conceptual level and to frame what evidence would be required to support or refute a prediction. The AI also supports interpretation of computed outputs by prompting students to explain plots and trends in physical terms, such as what slope or deflection at a bearing implies for alignment, or why a location of maximum bending moment does not necessarily correspond to the governing equivalent stress when stress concentration and combined loading are present.

Next, students use JupyterLite notebooks to compute system response and perform parametric exploration. The notebooks generate bending moment and deflection visualizations and compute stress, equivalent stress, and fatigue-related quantities under combined bending and shear. The executable environment enables students to manipulate geometry and loading while preserving traceability between input assumptions and output responses. Because each cell documents both the symbolic relationship and its numerical realization, students can inspect intermediate quantities such as curvature, stress components, and equivalent stress measures. This visibility reduces the opacity often associated with multi-step analytical derivations and supports causal reasoning about how local modifications propagate through the system. Students vary geometric parameters such as segment diameter and fillet size, as well as loading parameters, and examine how these changes

Table 1: Distribution of instructional roles across tools

Task	JupyterLite	Generative AI	Student role
Computation	Executes traceable calculations for reactions, internal forces, stresses, and deflections.	Provides conceptual guidance without generating numerical solutions.	Defines boundary conditions, parameters, and modeling assumptions.
Sensitivity analysis	Supports parametric exploration by visualizing responses to changes in geometry and loading.	Prompts interpretation of trends and identification of governing mechanisms.	Explains physical causes of observed changes and interactions among constraints.
Verification	Enables parameter sweeps and internal consistency checks across computed quantities.	Challenges assumptions and checks logical coherence of reasoning.	Cross-checks results across representations and defends conclusions using mechanics reasoning.

propagate across multiple constraints.

The activity design emphasizes directional and comparative reasoning. Students are asked not only what a safety factor or equivalent stress value is, but why it changed, which mechanism dominates, and how competing effects interact. A key computational affordance is reversibility: students can modify a parameter, observe system response, and immediately revert or explore alternative configurations without reconstructing the entire analytical solution. This low-friction iteration supports comparative reasoning across design states and makes sensitivity analysis a natural part of the workflow rather than an add-on exercise.

In contrast to traditional static design assignments in which a single configuration is evaluated to completion, this workflow treats variation itself as a learning mechanism. Students encounter governing behavior not as a fixed answer but as an emergent property of interacting relationships. Computation makes these interactions visible and adjustable, transforming design reasoning from equation application to structured exploration.

The cycle concludes with a brief written justification that makes reasoning explicit. Students explain their final design choice in terms of competing constraints and cite computational evidence from notebook outputs. They also describe at least one verification check they performed, such as a simplified hand calculation, a parameter sweep to confirm expected trends, or a consistency check between stress, deflection, and equivalent stress responses. This closing justification reinforces the expectation that computation functions as an evidentiary medium. Executable outputs provide structured evidence, while engineering judgment determines how that evidence is interpreted and defended.

6 Evidence of learning and measurement approach

Evidence of learning is organized around three complementary sources aligned with the module objectives: student-produced design artifacts, short reflective responses, and instructor observation of reasoning during discussion. The measurement approach is appropriate for a course-embedded instructional intervention and emphasizes conceptual integration, verification behavior, and reasoning quality rather than numerical correctness alone. Evidence of learning is examined through three complementary lenses aligned with the instructional objectives:

- **Artifact-based evidence of integrated reasoning.** Student design artifacts are evaluated for evidence of integrated reasoning rather than isolated calculation accuracy. Specifically, artifacts are examined for whether students (i) explicitly connect stiffness and fatigue constraints, (ii) justify critical locations using both global response and local discontinuity effects, and (iii) interpret parametric sensitivity results mechanistically rather than descriptively. Many students submit computational notebooks that calculate reactions, bending moments, deflections, stress concentration effects, and equivalent stress measures. Evaluation focuses on whether students can articulate what computed quantities represent physically, how parameter variation alters governing behavior, and how results support a defensible design decision. This emphasis reflects the computation-centered premise that executable models should function as reasoning tools rather than answer generators.

- **Reflective responses and rubric-based assessment.** Reflective responses are used to assess conceptual articulation and engineering judgment. Students respond to prompts requiring explanation of (i) why stiffness and fatigue constraints may compete, (ii) how geometric discontinuities can shift governing locations, and (iii) how they verified the credibility of their results across symbolic, graphical, and computational representations. Responses are evaluated using a compact rubric with dimensions including clarity of mechanism explanation, evidence-based justification, and verification practice. Each dimension is rated on a five-point scale anchored by descriptive performance criteria. This structured evaluation provides a transparent and repeatable method for assessing conceptual integration within a design context.

- **Formative observation of reasoning behavior.** Classroom discussion serves as formative assessment of reasoning processes. During guided exchanges, students are asked to defend trade-offs, reconcile differences between analytical predictions and computational output, and explain discrepancies between expected and observed trends. The instructor documents recurring misconceptions as well as prompts that successfully elicit deeper explanation. These observations inform iterative refinement of computational scaffolds and AI prompts, with the explicit goal of shifting attention from procedural execution toward mechanism-centered reasoning.

- **Perception data and calculation of reported percentages.** In addition to artifact-based and observational evidence, student perceptions were collected using short end-of-module survey prompts administered anonymously. Survey responses were obtained from 81 students in MEEN 305 and 95 students in MEEN 368, corresponding to response rates of 68% and 72%,

respectively, where the same CILT framework was implemented using different mechanics problems. All respondents completed the same instrument at the conclusion of the module. Each item used a five-point Likert scale (1 = Strongly disagree, 5 = Strongly agree). For each construct shown in Figure 2, the reported value represents the percentage of respondents selecting either 4 (Agree) or 5 (Strongly agree). Formally, if N denotes the total number of valid responses to a given item and $N_{4,5}$ denotes the number selecting 4 or 5, the reported percentage is computed as

$$\text{Positive response rate} = \frac{N_{4,5}}{N} \times 100\%. \tag{4}$$

This aggregation approach emphasizes endorsement strength while maintaining interpretability for course-level analysis.

The plotted constructs correspond directly to computation-centered learning objectives. "Conceptual understanding" captures perceived improvement in linking equations to physical behavior. "Integration of constraints" reflects students' ability to reconcile stiffness and fatigue considerations within a unified model. "Confidence in verification" measures perceived improvement in checking results across representations and parameter variations. "Engagement with design reasoning" reflects active exploration rather than passive calculation. "Usefulness of computation + AI" captures the perceived value of executable notebooks and structured articulation prompts as supports for learning.

- **Role of computation in perceived improvement.** The reported trends in Figure 2 are

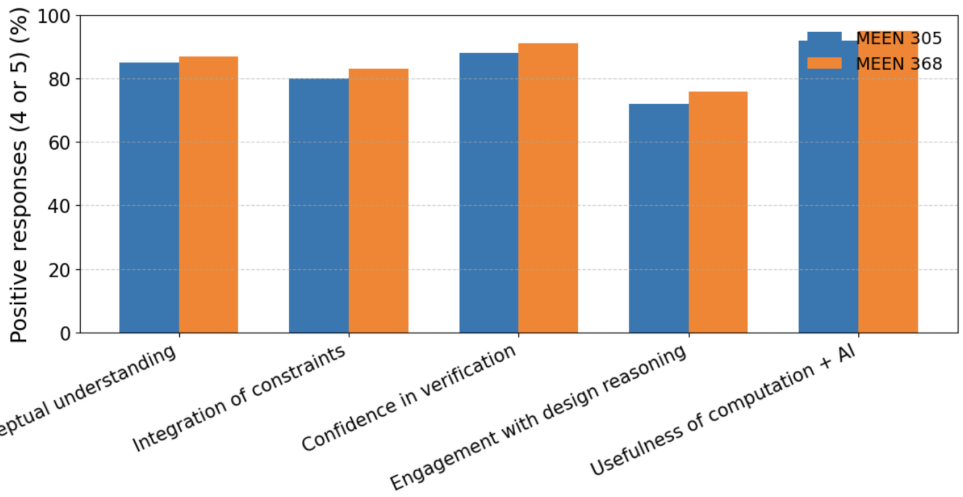

Figure 2: Percentage of students selecting 4 (Agree) or 5 (Strongly agree) on a five-point Likert scale for key constructs associated with the CILT module. MEEN 368 corresponds to the stepped shaft implementation, while MEEN 305 corresponds to a different mechanics application of the same framework.

consistent with observed behavioral shifts during the module. Students frequently performed exploratory parameter sweeps, deliberately compared symbolic derivations with numerical outputs, and normalized stress quantities to interpret governing transitions. The ability to

vary diameter, fillet ratio, load position, and material properties within a single executable environment allowed students to observe sensitivity patterns that are difficult to apprehend through static calculation alone.

In particular, high reported confidence in verification aligns with repeated cross-checking behaviors observed in notebooks, including dimensional validation, comparison of analytical and numerical curvature fields, and reconciliation of stress concentration effects with nominal scaling trends. Similarly, elevated ratings for conceptual understanding and constraint integration correspond to explicit written explanations in which students linked $1/d^3$ stress scaling and $1/d^4$ deflection scaling to governing transitions under geometry modification.

Across both cohorts, positive response rates exceeded 80% for conceptual understanding, integration of constraints, and confidence in verification, with MEEN 368 showing slightly higher endorsement across most constructs. Confidence in verification and perceived usefulness of computation exhibited the highest reported gains, suggesting that repeated parameter sweeps and cross-representation checks strengthened students' trust in evidence-based reasoning. Engagement with design reasoning, while slightly lower than the other constructs, remained above 70% in both courses, indicating that computation-supported exploration promoted active interpretation rather than passive calculation. The consistent pattern across MEEN 305 and MEEN 368 supports the transferability of the computation-centered architecture across related mechanical engineering contexts.

Taken together, these evidence sources suggest that computation functions not merely as a calculation aid but as a structural reasoning medium. While demonstrated in the context of stepped shaft design, the assessment constructs of mechanism articulation, verification practice, and evidence-based justification are broadly applicable to mechanics and mechanical design problems involving competing constraints.

While these findings are encouraging, several limitations should be noted. The perception data reflect self-reported learning gains within a course-embedded intervention and do not establish causal effects. The study does not include a control group using a non-computational workflow, and sample sizes are limited to the participating cohorts. Accordingly, the results should be interpreted as preliminary evidence of alignment between computation-centered design activities and perceived conceptual improvement rather than definitive proof of instructional impact.

7 Conclusion

This work presents a computation-centered instructional framework, CILT, implemented through JupyterLite notebooks in a representative mechanical engineering course, MEEN 305 (Solid Mechanics) and MEEN 368 (Solid Mechanics in Mechanical Design). Using stepped shaft design as a canonical example for the MEEN 368 implementation, the module positions executable computation as the primary epistemic reference through which students explore coupled relationships among geometry, stress, deflection, and fatigue behavior. Rather than treating computation as a terminal calculation step, the framework embeds parametric variation, reversibility, and verification directly into the reasoning process.

The central contribution of this work is the restructuring of the design task around executable models. By making governing transitions observable and manipulable within a browser-based computational environment, JupyterLite enables students to engage in directional prediction, systematic parameter sweeps, and cross-representation verification as integral components of the task. In this structure, mechanism-centered explanation and evidence-based justification become necessary conditions for completing the design activity rather than optional extensions.

Evidence from student artifacts, reflective responses, classroom observations, and perception data in MEEN 305 and MEEN 368 suggests alignment between this computation-centered architecture and improvements in integrated reasoning, verification confidence, and engagement with design trade-offs. While preliminary and course-embedded, these findings indicate that computing-enabled platforms can support the development of engineering judgment while preserving analytical rigor. Although demonstrated in the context of stepped shaft design, the instructional principles emphasized here (parametric exploration, structured verification, and evidence-based justification within an executable environment) are transferable to other mechanics and mechanical design topics where multiple constraints interact. More broadly, this work illustrates how browser-accessible computational platforms such as JupyterLite can serve as scalable infrastructures for embedding computation-centered learning across mechanical engineering curricula.

References

[1] M. D. Koretsky, D. Amatore, C. Barnes, and S. Kimura, "Enhancement of student learning in experimental design using a virtual laboratory," *IEEE Transactions on education*, vol. 51, no. 1, pp. 76–85, 2008.

[2] P. J. Denning and M. Tedre, "Computational thinking: A disciplinary perspective," *Informatics in Education*, vol. 20, no. 3, p. 361, 2021.

[3] M. T. Chi, N. De Leeuw, M.-H. Chiu, and C. LaVancher, "Eliciting self-explanations improves understanding," *Cognitive science*, vol. 18, no. 3, pp. 439–477, 1994.

[4] C. L. Dym, A. M. Agogino, O. Eris, D. D. Frey, and L. J. Leifer, "Engineering design thinking, teaching, and learning," *Journal of engineering education*, vol. 94, no. 1, pp. 103–120, 2005.

[5] J. Turns, C. J. Atman, and R. Adams, "Concept maps for engineering education: A cognitively motivated tool supporting varied assessment functions," *IEEE Transactions on Education*, vol. 43, no. 2, pp. 164–173, 2002.

[6] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay *et al.*, "Jupyter notebooks–a publishing format for reproducible computational workflows," in *Positioning and power in academic publishing: Players, agents and agendas*. IOS press, 2016, pp. 87–90.

[7] L. A. Barba, L. J. Barker, D. S. Blank, J. Brown, A. B. Downey, T. George, L. J. Heagy, K. T. Mandli, J. K. Moore, D. Lippert *et al.*, "Teaching and learning with jupyter," *Recuperado: https://jupyter4edu. github. io/jupyter-edu-book*, pp. 1–77, 2019.