2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #51554

To Teach, To Leverage or To Exclude: Structural Software in Analysis and Design Classes

Dr. Ryan Solnosky P.E., The Pennsylvania State University

Ryan Solnosky is an Teaching Professor in the Department of Architectural Engineering at The Pennsylvania State University at University Park. Dr. Solnosky has taught courses for Architectural Engineering, Civil Engineering, and Pre-Major Freshman Engineering. His research interests are in Capstone Design Settings and Projects, Active Learning with Technology, and Early Heuristic Techniques used in Structural Design.

To Teach, To Leverage or To Exclude: Structural Software in Analysis and Design Classes

Abstract

Within structural engineering education, teaching fundamental behavior and theory has always been paramount to academic educators. This thinking has resonated across both analysis and design courses, where the intent is to instill core fundamentals and the basis behind design standard provisions before students apply them to contexts that are more practical. While learning does not end with graduation, a question frequently asked is: "should educators teach more software in their classes?" Arguments against software in courses include: software diminishes learning the fundamentals, students should first learn techniques by hand, we should not teach students to be "black box" thinkers, or even that there is no room in the curriculum for these opportunities. Arguments supporting software in the classroom include: educators can demonstrate concepts at a deeper level, permit more contextualized scenarios, and leverage software to showcase fundamental behavior.

This body of work provides perspective insights (from educators and industry professionals) regarding what topics are taught or should be taught. By knowing what educators currently do or do not do and why, we can benchmark the status quo. At the same time, knowing what industry values regarding modeling vs fundamental skills provides an opportunity to update their teaching practices. Having this benchmark, educators can tie practices to the ASCE BoK and the NCSEA recommended curriculum guidelines. Presented in this paper is the resulting survey data that begin to answer these research questions in the form of curricula change recommendations: "*To what extent do industry vs. academic perspectives align or diverge regarding software education?*" And "*Where and to what extent could software be taught in structural classes and how can we leverage it?"*

From the data, there was clear cohort agreement that fundamentals should not be abandoned and that not all analysis methods need to be taught. Software should be taught but how varied and for what intent. Professionals put more emphasis on integrating software throughout the curriculum for regular repeated exposure while faculty supported focused application to projects and capstones. Additionally, professionals advocated for more time building structural intuition around results and teaching techniques for software validation. Recommendations are provided to promote and push for software integration that balances these skills by rethinking analysis and design course delivery.

Introduction

There is a continuous demand for structural engineers who possess professional and specialized technical skills [1] that extend beyond the discipline-specific fundamentals [2] typically embodied in undergraduate structural curricula within architectural (AE) and civil engineering (CE) programs [3]. Adding to this, engineering students entering college since at least the early 2010s tend to develop computational literacy quickly [4] and increasingly expect formal instruction on technical tools [5-6]. Given their frequent exposure to interactive, on-demand technologies, students commonly expect to be introduced to the technology skills required by our profession during their college education [7]. These expectations intersect with long-standing debates in structural engineering education regarding the role of computational tools [8-9]. Such debates have persisted since the rise of commercial structural software in the early 1980s [10].

At the core of this debate is whether instruction should prioritize computational software or fundamental knowledge, often taught through hand calculations [11]. Traditional approaches to teaching structural analysis and design are widely viewed as foundational because they emphasize theory and application [12]; yet they typically rely on hand calculations applied to isolated or closed-form scenarios or to highly idealized open-ended projects that remain manageable by hand [13]. Conceptual understanding is central to this debate, with concerns that software may function as a "black box" and encourage over reliance on "plug and chug" behavior [14]. Despite advances in active learning and its documented success in structural engineering courses [11], instructional practices continue to fail to generate sufficient

conceptual understanding [3; 15]. Barner et al. [16] further state that students' ability to apply fundamentals to complex, ill-defined problems encountered in professional practice is consistently lacking [17]. In contrast, proponents of software integration into the curriculum argue that, when implemented appropriately, computational tools can serve as effective mechanisms for developing conceptual understanding and critical thinking [18].

This debate is often framed around two prominent and opposite stances: 1) teaching content desired by students and, in some cases, industry [8], and 2) teaching content deemed most appropriate by academics [3]. These raise additional unresolved questions regarding whether software should be taught, to what extent it should be incorporated, and which topics are best addressed through computational modeling versus traditional hand-calculation methods—particularly in an era when computers perform much of the technical work for practicing engineers [2]. Related but distinct, artificial intelligence (AI) in engineering education has recently gained significant attention due to its widespread adoption and transformative potential [19]. AI education [19] further underscores the need to reconsider how structural engineers are trained to both use and critically evaluate computational tools.

To better understand whether, how, and to what extent software should be incorporated into the classroom relative to traditional methods, it is necessary to examine current perceptions and practices among key stakeholders. Understanding both industry and faculty perspectives is important, as educators are tasked with preparing students for long-term professional success. With such a benchmark comparison, educators can make more informed decisions that align instructional practices with the ASCE Body of Knowledge (BoK) [20] and the National Council of Structural Engineers Associations (NCSEA) curriculum guidelines [21]. Specific research questions that guided this study include:

- To what extent do industry and faculty perspectives align or diverge regarding software education?
- What software-related opportunities or skills should educators be teaching?
- Where and to what extent should software be taught in structural engineering courses?

ABET, ASCE BoK, & NCSEA: Topics, Ties, and Positions on Software Integration

A common goal of professional organizations is to guide or shape course scope priorities by defining practical skill expectations for entry-level engineers [1,18]. ABET establishes a foundational framework for producing competent professional engineers [21], while the ASCE BoK V3 articulates the knowledge, skills, and attitudes required for entry into civil engineering practice [20,23]. Diving deeper into industry, The NCSEA provides recommendations specific to structural engineering curricula [21]. Although all three organizations seek to develop robust entry-level engineers [23], conflicting opinions remain between the emphasis on fundamental theory and practical application [24]. Because of the influence of these bodies, academic programs attempt—or are mandated, in the case of ABET—to adopt their recommendations [23], often resulting in course revisions, additions or removals of material, or course consolidation [23,25].

While the ASCE BoK acknowledges technological advancement, its guidance remains vague with respect to specific technology-related skills [20]. ABET does not explicitly address computational software, though interaction with technology may be indirectly linked to outcomes such as lifelong learning (Outcome 7), applying engineering knowledge (Outcome 2), and engineering judgement and drawing conclusions (Outcome 6). NCSEA does provide the most directly applicable guidance for this study (i.e.: coverage topics), with its most recent surveys conducted in 2016 [26], 2019 [27], and 2021 [21]. Although NCSEA emphasizes critical course content, the framing of these topics may limit opportunities for curricular change in regard to building conceptual understanding. The two recommended analysis courses include limited qualitative or approximate quantitative methods for understanding behavior, particularly in interpreting software output. Validation is only mentioned once, in the second analysis course, and matrix methods are recommended specifically within the context of structural software [21,28]. Secondary recommendations include the use of real structural examples to illustrate behavior and reinforce foundational knowledge.

As a result, several industry organizations discuss skill sets related to modeling, interpretation, and contextual understanding appear as secondary recommendations rather than core objectives. Design course guidance is similarly technical, with limited discussion of design software, early-stage design heuristics, or the integration of broader design considerations beyond specific applications. Although the 2021 survey did expand recommended topics to include: load paths and load flow, finite elements, performance-based design, and building information modeling (BIM), it provides limited guidance on curricular placement, instructional time, or depth of coverage [21].

A Historical Perspective: To Teach or Not to Teach: Software

Digging deeper into faculty perceptions, a reoccurring theme has been: fundamentals vs computational software. Arguments opposing software use include concerns that it diminishes students' understanding of fundamentals that are essential to effective modeling [3], that students should first learn concepts via hand calculations [29], and that software exposure may lead to "black box" mindsets [30]. Proponents, however, argue that software enables exploration of variables and constraints not easily addressed by hand [31], allows educators to demonstrate complex concepts at greater detail [32], supports contextualized scenarios aligned with professional practice [33], and can reinforce fundamental structural behavior within traditional courses. More broadly, challenges related to representation, idealization, and contextual isolation in teaching fundamentals [34], instructional constraints imposed by dense curricula [21], difficulty in knowing not only a concept but when to apply it [35], and reduced student interest due to abstract presentations of fundamentals [36] persist across both sides of the computational modeling debate.

Concerns remain that software may create the impression that students across skill levels can accurately analyze structures but realistically lack sufficient foundational knowledge [30,39]. Fernandez-Sanchez [35] documented instances in which reliance on tools led to a loss of mindfulness of actual structural behavior. McDaniel and Archer [40] similarly found that many structurally focused programs are susceptible to "black box" syndrome due largely to insufficient instructional rigor. Consequently, computer use in analysis and design courses is often minimal or simplistic, frequently limited to subsets of homework assignments, projects, or capstone experiences [14,32-33,37,39]. To overcome this, Hanson [40] explored instructional approaches aimed at reducing blind trust in software and strengthening students' ability to evaluate the reasonableness of results.

Shaikh [42] argues that when software is implemented correctly, modeling real problems can be grasped relatively easily by students. Prior studies have examined the role of modeling in improving student motivation [36], refining problem-solving approaches [Ahmed et al.], visualizing structural behavior not easily understood through traditional methods [33,35], connecting failure modes critical to conceptual understanding [31, 32], developing engineering "common sense" [43-44], and helping students overcome the abstract nature of structural concepts [45]. Baker [46] adds that digital "models" allow students to explore the limits of their conceptual understanding by testing assumptions and responses (through manipulation), mirroring professional practice in which engineers routinely conduct "what-if" analyses [40,42]. To start, time needs to be spent on how to approach and manipulate simpler problems governed by the same fundamental principles.

Collectively, these ongoing debates, evolving professional expectations, and, at times, unclear guidance across stakeholder groups motivate the need to better understand current perceptions and practices regarding software education in structural engineering.

Research Study Design

At the start of this study, it was hypothesized that integrating a mixture theory and computer modeling applications can provide a balanced approach that enables students to work towards the leading edge of structural practice when they enter and contribute to professional practice. Accordingly, it was also hypothesized that, when implemented appropriately, computer-modeling integration can maintain or improve conceptual understanding and critical thinking because of the possibilities software can do.

To address the three research questions driving this study (see Introduction), the study design was informed by literature examining: 1) hand calculation methods and design education status quo, 2) the

possible applications of software in education, and 3) the status quo of structural engineering education. These prior contexts motivated the collection of new data from both industry professionals and faculty educators to develop a refreshed perspective and set of recommendations for more evidence-based research to be conducted.

Data were collected using a qualitative–quantitative mixed-methods approach [47] based on surveys [48]. In addition, the authors, who either teach or practice structural engineering, provide perspectives at key points in the paper to enrich the resulting recommendations. The following subsections describe the survey development, deployment, and participant demographics.

Survey Development

Two similar surveys were sent to structural engineering educators (Survey 1), and industry professional structural engineers (Survey 2). Both surveys were administered online in Qualtrics after an IRB review.

Survey 1 focused on teaching practices related to traditional design and analysis methods, computational software (within the structural domain), and how faculty go about selecting. In total, Survey 1 had 20 questions broken into five categories. Survey 2 asked a similar series of questions on their work practices related to traditional analysis and design methods, computational software usage, and their perceptions on if, and to what extent, software should be taught in educational settings. In total, 18 questions were broken into five categories. For brevity, individual questions are not documented here in detail except as needed when reporting the results. Readers may contact the authors to obtain copies of the survey instruments.

Survey Deployment

Eligible academic participants included full- and part-time faculty of any rank or track who taught structural engineering courses within U.S. civil, architectural, or structural engineering programs. Academics were recruited through ASEE Civil and Architectural Engineering divisions and an AEI listserv. Exclusion criteria were established to exclude the following participants: graders, graduate students, undergraduate TAs, etc.

Eligible industry participants included U.S.-based professionals practicing structural engineering on building-centered projects. Participants ranged from entry-level designers to senior personnel working in firms of varying sizes. Industry participants were recruited through the authors' alumni network and career-fair partner companies (120+ companies) to improve response rates relative to random postings. Invited alumni represented firms ranging from small (<20 people) to large (100+ people), with local to national recognition. Approximately 15% of industry participants were not graduates of the authors' institution.

The target cohort size was approximately 40 academics and 40 professionals. Surveys remained open for two weeks, followed by a reminder. After an additional two weeks, surveys were closed and data were downloaded. Response rates could not be reliably tracked due to listserv-based distribution.

Participants and Demographics

Faculty participants represented diverse backgrounds, with 93.7% being tenure-track and 6.3% being non-tenure-track faculty (Table 1). Faculty reported a mean (m) of 10.3 years of teaching experience and m=2.5 years of industry experience. The industry cohort reported substantially greater professional experience in structural building design, with a m=19 years. This difference aligns with typical academic career pathways, particularly for tenure-track faculty. Professional licensure and certifications were also recorded (Table 1). Of primary interest were the PE and SE licenses as a benchmark indicator of structural engineering competency. 94% of industry participants and 62.5% of academic participants held a PE license, while 7.3% and 23.5%, respectively, held an SE license.

Table 1: Cohort Descriptive Statistics

Cohort Facts	Academic Cohort	Industry Cohort
Total Participants (n)	16 total	35 total
Years of industry experience	Min: 0 yrs. Max: 8 yrs. Avg: 2.5 yrs.	Min: 2 yrs. Max: 47 yrs. Avg: 19 yrs.
Years of academic experience	Min: 1 yr. Max: 28 yrs. Avg: 10.3 yrs.	N/A*
Structural Licensure	PE: 62.5% (n=10) SE: 7.3% (n=1)	PE: 94% (n=32) SE: 23.5% (n=8)
Type of Academic Position	Tenure Track: 15 Non-Tenure Tack: 1	Adjuncts: 5

*Adjuncts were excluded here

These demographics indicate that the data reflect multiple informed perspectives and that participants possessed enough familiarity with the topics to have validity to the recommendations pulled from their data.

Survey Results

Of the 59 total survey responses received, 8 incomplete surveys were discarded, as incomplete responses were interpreted as consent withdrawn. The remaining 51 valid responses are presented here and grouped by key topics to frame the discussion. Where possible, results are compared directly between faculty and industry cohorts. Due to the relatively small number of faculty responses (n = 16), comparisons primarily rely on descriptive statistics. Additional statistical measures, scales, and question background are discussed within the relevant sections as applicable.

Hand Methods for Analysis: Industry vs Academia Current Practices

To meaningfully discuss whether structural engineering curricula should emphasize computer software or hand-based methods, it is first necessary to examine perceptions and practices related to hand methods of analysis. Analysis methods are central to this discussion, as a central theme of the published literature on software education focuses on structural analysis performed using computational tools [3,10,21]. Historically, hand methods have been preferred because they are viewed as essential to developing critical thinking and understanding structural behavior [33,36]. Of the analysis topics regularly taught in structural engineering programs [21,49], 12 methods were picked for inclusion (Fig. 1). Faculty responses (Fig. 1) indicate that, of these 12 methods, six are either taught at the undergraduate (UG) level or not taught at all. Eleven of the methods are taught more frequently at the UG level than at the graduate (GR) level, with the exception of plastic analysis. Additionally, six methods have a majority response (greater than 50%) indicating that they are not taught at any level.

Fig. 1 further shows that slope deflection, direct integration, virtual work, moment area, and matrix methods are the most taught hand analysis techniques. Open-ended responses suggest that faculty do not necessarily teach all these methods but instead select those that can reasonably fit within a single course (excluding matrix methods) [21]. As expected, AISC Tables 3-21 and 3-22 and ACI Coefficients are typically taught at the UG level and are nearly always covered in their respective design courses. One notable outcome is the comparatively low emphasis (60% not taught) on approximate methods that can be used to quickly assess the reasonableness or correctness of analysis results.

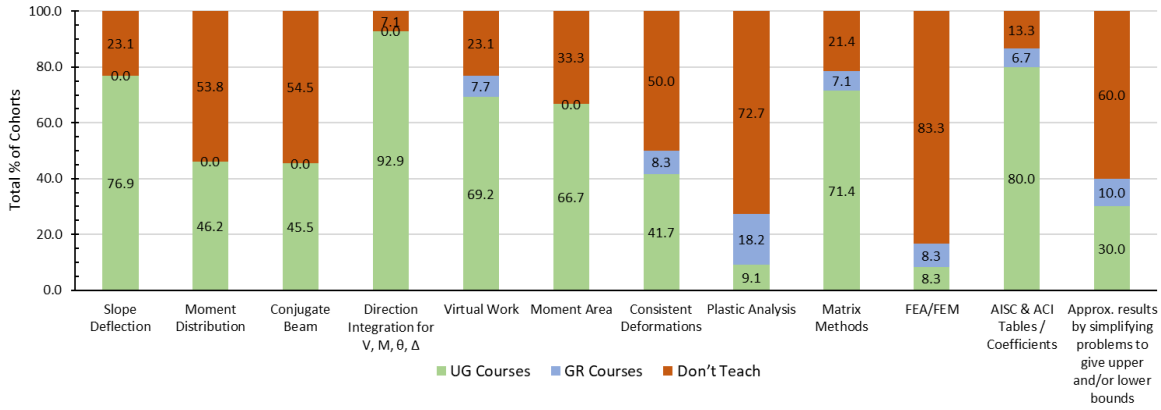


Figure 1: Faculty on if and where different analysis methods are taught.

A similar question was posed to industry respondents, but was framed as: *"How frequently do you use these analysis method types in practice?"* A 6-Point Likert Scale from Never to Daily was adopted. Fig. 2 presents these results. The data show that many mathematically intensive analysis methods are rarely or never used in practice. The methods with the highest reported use frequency are moment distribution, virtual work, and plastic analysis. In terms of regularly leveraged methods, simplistic or already developed methods like AISC Tables or ACI Coeficcents is over 64% used frequently. Industry respondents also report frequent use of approximate or conservative methods to obtain quick answers that can serve as reference checks (far right of Fig. 2).

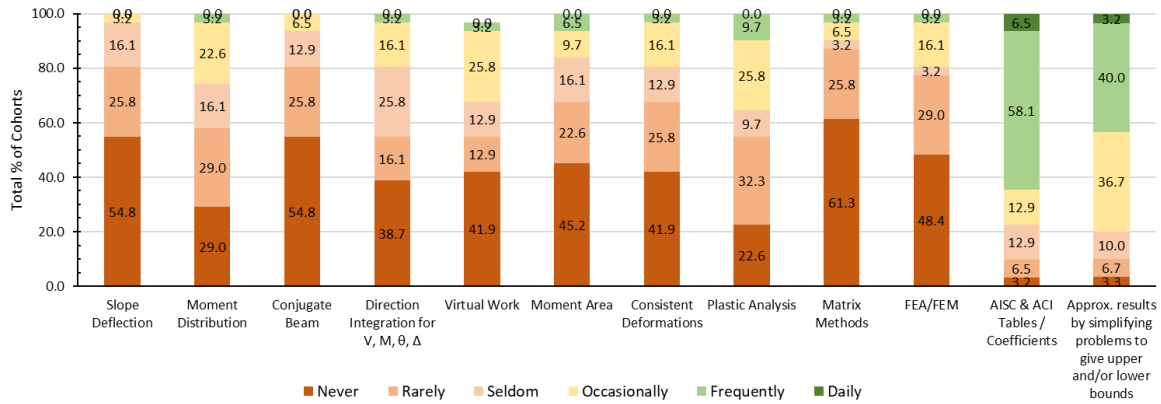


Figure 2: Industry usage frequency of different taught analysis methods.

Given that industry respondents report using approximate methods ~43% of the time, both cohorts were asked which approximate analysis methods should be taught (industry) or are currently taught (faculty), as shown in Fig. 3. Based on available instructional resources that describe approximate analysis techniques [49,50], 10 methods were included. Prior literature by Hanson [49] and Nielson [50] promote these techniques due to their qualitative underpinnings, their role in helping students conceptualize structural behavior, their emphasis on critical interpretation of results rather than procedural detail, and their usefulness in judgement checking software-generated results [41]. Overall, agreement levels across methods are relatively similar between cohorts (Fig. 3), although industry responses are slightly higher. The largest differences occur for approximate lateral movement methods (37.8% higher among industry) and methods for estimating upper and lower bound solutions (also 37.8% higher for industry). One unexpected outcome is that faculty respondents indicated no instruction on qualitatively assessing loaded trusses to estimate whether members are in tension or compression.

A key takeaway from Fig. 3 is that, for most approximate methods, industry reports over 50% agreement that these techniques should be introduced in structural engineering courses (in some form). This finding aligns with earlier results in Fig. 2 and suggests potential opportunities to better prepare students for industry.

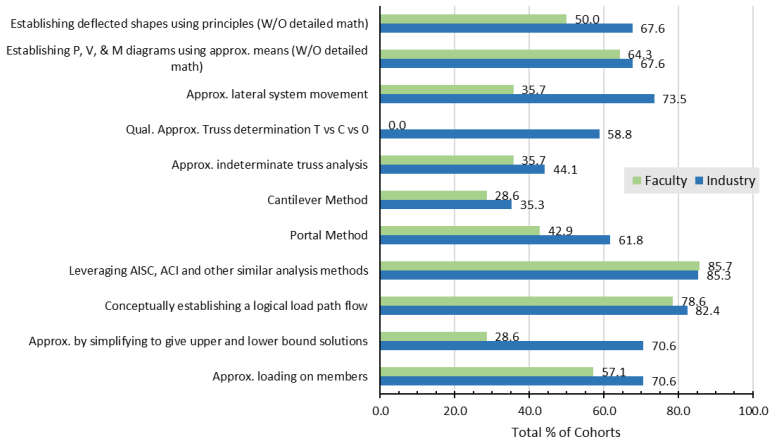
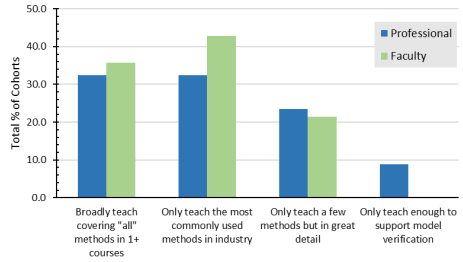


Figure 3: Perceptions on yes to teaching different approximate analysis methods.

Given the recorded differences between what is taught vs. what is frequently used in practice, both cohorts were asked how analysis methods should be selected for instruction (Fig. 4a). Data indicates mutual perspective alignment, but no single approach achieved a clear majority. Notably, fewer than 25% of respondents supported the approach of selecting only a few methods and teaching them in extensive detail. The greatest agreement (by a substantial margin) is to focus on what industry is using. If this approach is adopted, the results from Fig. 2 suggest an increased emphasis on plastic and approximate analysis methods. A follow-up question identified which specific methods faculty should prioritize. Figure 4b presents the ranked methods.



a) Approach to selecting methods

Methods to Teach	Yes to Teach
Moment Distribution	85.3%
Direct Integration for V, M, θ and Δ	73.5%
Virtual Work	58.8%
Moment Area	64.7%
Plastic Analysis	52.9%
AISC Tables and ACI Coefficients	82.3%
Approximating Results	70.6%

b) Top industry agreements on methods

Figure 4: Best approach to teach hand analysis methods from an industry perspective.

Perceptions on Software in Academia

To understand software integration in academia, a concise dive into perceptions of both faculty and industry cohorts were surveyed. The first question asked respondents for their opinions on how software should be introduced in academic settings. Respondents were provided with six options (Fig. 5) and asked to choose the approach they felt would be most effective for department undergraduate (UG) programs. The majority agreed that software should either be introduced at key stages or integrated throughout the curriculum. In contrast, professionals exhibited a broader distribution of responses, though the most common preference was still introducing key software skills at key stages.

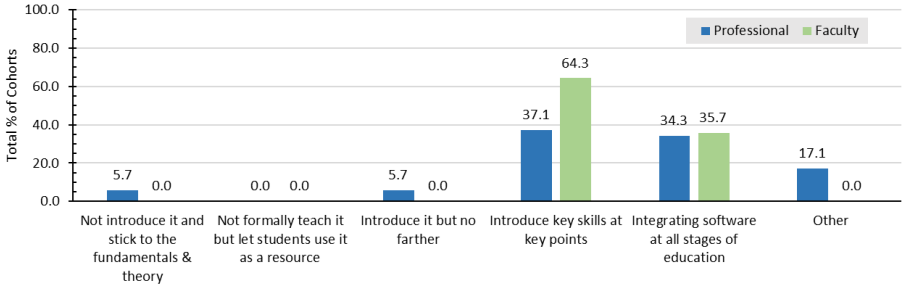


Figure 5: Opinions on implementing structural software in academic settings.

The second question asked respondents to describe into which software skills they believe are warranted to be taught in structural engineering courses (Fig. 6). This question was informed by literature [10,37,39] suggesting a variety of software applications could be leveraged, as well as recent trends such as data science [19] and parametric modeling [51]. Respondents were able to select multiple options across four different curriculum locations. Fig. 6 indicates a lack of consensus between faculty and industry on which software applications are essential and on where to place them. Faculty respondents showed more reluctance toward four types of software (>50% selecting "Don't Teach") while in contrast, industry only had much lower "Don't Teach" responses (~34%).

A notable trend in industry responses was the strong support for including commercial design and analysis software in both introductory and advanced courses, spanning both UG and graduate (GR) levels. This suggests an integrated approach where software is taught in conjunction with relevant structural topics. Conversely, faculty were most opposed (~71%) to teaching commercial design software, with reasons not known from this survey (possibly due to unfamiliarity). Faculty also expressed hesitation toward incorporating data science, this counters the increasing published literature emphasis on these skills in civil engineering education [52]. Interestingly, both faculty and industry respondents agreed on the importance of teaching spreadsheet development and "traditional" analysis software at the UG level.

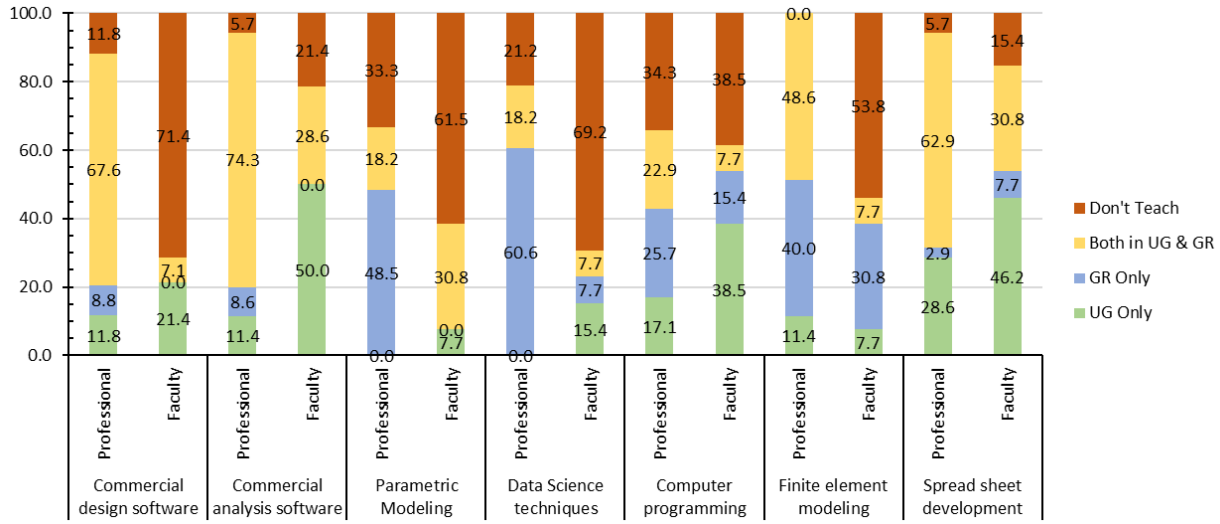


Figure 6: Types of software to teach and where in a curriculum.

Current Teaching of Software:

To provide broad recommendations on software integration, faculty were surveyed about their current practices. Here, questions aimed to understand the status quo and the reasoning behind existing practices. Of the six questions utilized, two were also asked to industry professionals to obtain data on recommendations for faculty to consider (Figs. 7-8).

The first benchmark was a list of 12 common areas often discussed in the context of structural engineering software, categorized into three "how to" domains: 1) model building and software usage, 2) result validation and debugging/engineering judgment, and 3) modeling type selection and application approaches (Fig. 7).

Faculty were asked to select the topics they currently teach, while industry professionals selected the topics they felt were important that programs teach. Industry respondents reported a higher percentage of recommended coverage across all areas compared to faculty. For result validation, industry scored slightly higher, but faculty responses were close. Notably, only ~64% of faculty reported comparing hand calculations with software results, while ~97% of industry professionals viewed this as highly desirable. In the model-building category, faculty led in three out of four subcategories. However, it was surprising that no faculty members reported teaching proper modeling techniques (e.g., meshing, selecting solvers, modifying properties, etc.).

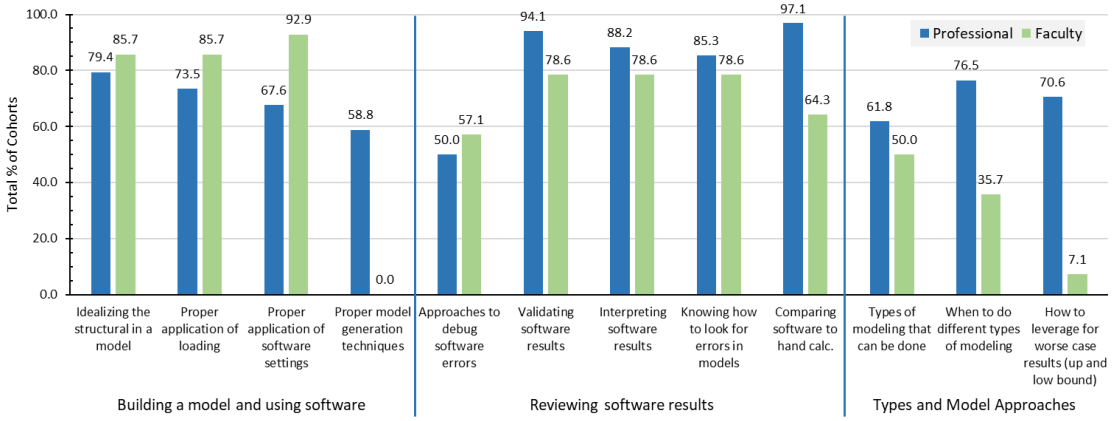


Figure 7: Frequency of which topics are typically taught in courses.

Faculty were also asked about the amount of time they dedicate to teaching software and modeling-related topics, with industry providing recommendations for appropriate durations (Fig. 8). Respondents were given eight duration options (including "none"), which were categorized by course type (analysis, design, and capstone).

In analysis courses (Fig. 8), 40% of faculty spend 2-3 weeks on software, while 26% of industry professionals recommend 3-4 weeks. Industry responses showed a broader variation, ranging from 1 week to 3-4 weeks. In design courses, faculty responses varied more, with 18% indicating "none" and 55% selecting 1 week. Industry responses for design courses were also spread out, but the two most favored durations were 1 week (23%) and 2-3 weeks (20%). In capstones, software use was deemed all-encompassing, with 44% of faculty dedicating 50% of the semester to it (though it is unclear on if it was taught or just leveraged). For the other faculty, software was used for 10-25%. A surprising result was the limited number of high adoption recommendations for software in capstone projects, despite research [53] indicating that capstones are an ideal area for deeper software integration.

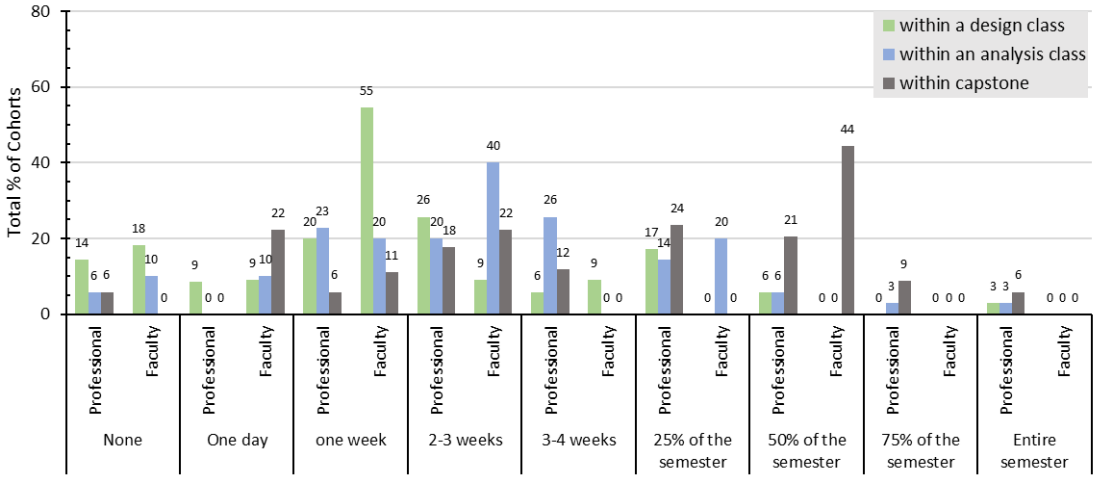


Figure 8: Approx. dedicated time (or suggested time be spent) on model-related topics and activities.

Shifting focus to just faculty, instructors were asked about the types of studies, simulations, or problems they assign to students when software as a key component. Faculty had seven options plus an "other" category. Respondents who do not teach software (n=2, 13%), skipped this question. Fig. 9 shows the “type methods” taught frequency. The most common was validating the accuracy of hand calculations (~91%). The least common (~27%) was strengthening students’ understanding of how structural features influence results. Interestingly, study types that foster critical thinking or conceptual understanding [41, 43, 49] received lower adoption rates (Fig. 9). This suggests an opportunity to incorporate more studies that strengthen students' conceptual grasp of structural behavior.

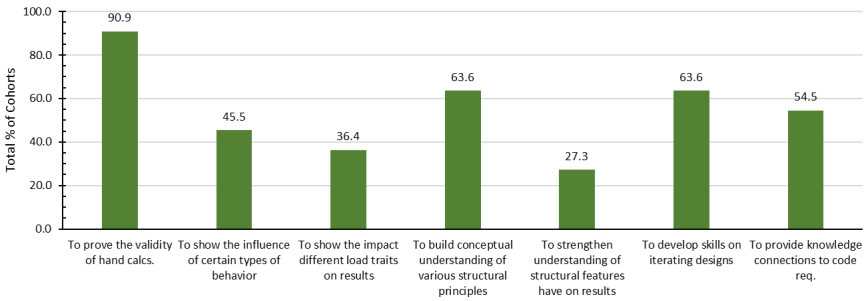


Figure 9: Adopted types of studies that are given to students with a software component.

Faculty were also asked about the types of structural representations being modeled and the context in which these studies were applied (Fig. 10). Most faculty responses indicated less than 50% engagement with certain structural types. Although, there was a good distribution across various representations. Trusses (61.5%) and full buildings (53.8%) were the most commonly modeled structures, with capstone encompassing the full building perspective. In terms of problem contexts, no context received over 50% utilization, with the lowest being evaluating results from software (30.8%), and the highest was problem-solving approaches (46.2%). This suggests a disconnect between teaching software and applying it to broader structural processes [32,38].

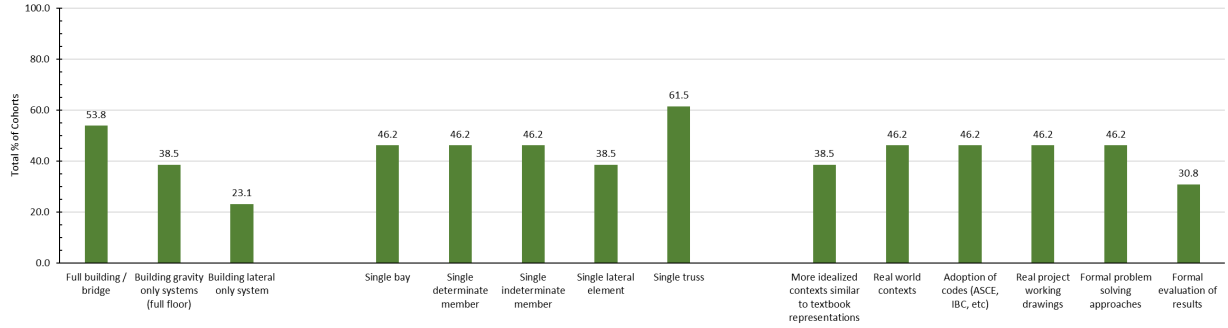


Figure 10: Different structural representations and contexts being utilized in the classroom.

Faculty who incorporate software into assessments were asked to describe the balance between theoretical and practical components (Fig. 11). No faculty ignored practical application, but 38.5% favored theory-heavy assessments with limited practical components. Of the rest, 46.2% attempted to balance both aspects. This trend may reflect broader challenges in computational assessments, where dense theoretical content, such as Finite Element Methods (FEM), may not always connect well with real-world career applications [42].

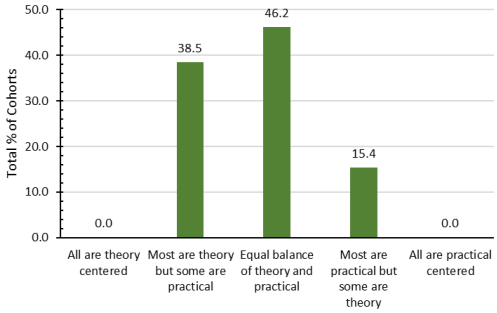


Figure 11: Balance of theory vs practical scoping in faculty assessments.

Given earlier insights (Figs. 9 and 10) into adoption gaps around critical thinking focused studies, faculty were asked to document conceptual and fundamental knowledge struggles they observe in students when using software. Results were grouped into three categories (each with three specific items) (Fig. 12). Faculty reported high frequencies of observed struggles across all categories. The lowest difficulty was

applying knowledge to new situations, while the greatest challenges occurred in interpreting results and drawing conclusions (Fig. 12, middle set). This suggests that curriculum improvements focused on result interpretation, either through software or conceptual hand means, would have a significant impact.

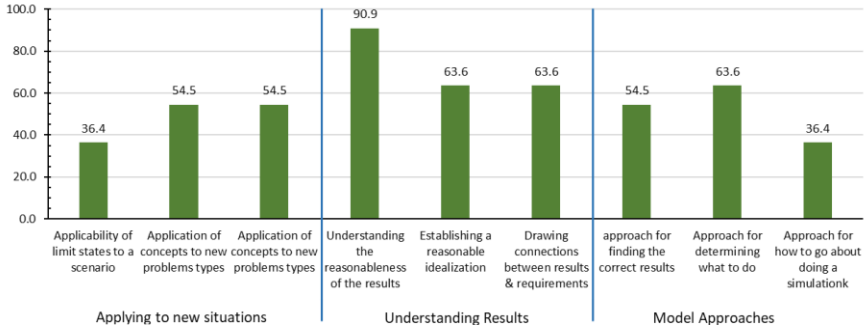
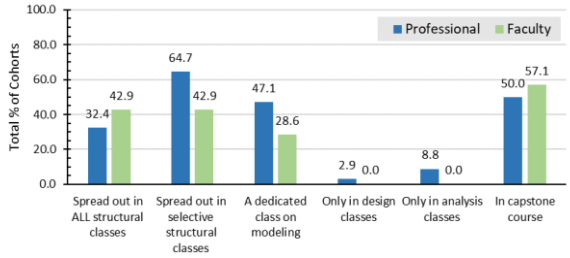


Figure 12: Observed knowledge struggles in students.

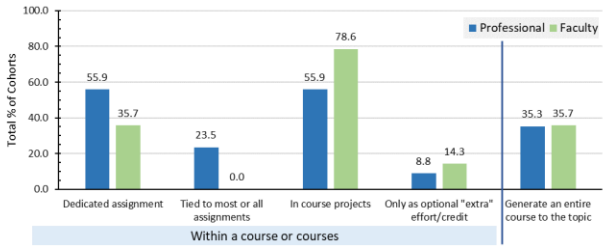
Cohort Perceptions on Software Integration Possibilities

Knowing the current status of software modeling education, hand calculation trends, and perceptions from faculty and industry, next step was to examine respondent-generated possibilities for future software integration. Earlier results provided industry insights on topics (Figs. 3–7) and methods (Fig. 8). This section builds on those findings. Three survey questions addressed software integration, with data in presented in Figs. 13a–b and 14.

Respondents indicated, based on their expertise, where software should be integrated within courses or curricula (Fig. 13a). Industry most frequently indicated that software should be spread across selective structural classes (64.7%), followed by capstone at 50% with a close third (47.1%) being to create or dedicate a unique class focused on modeling. Faculty favored the capstone course (57.1%) with considerably lower support for the other options (Fig. 13a). Both cohorts reported similar support (33–43%) for spreading software instruction across all structural classes. Few respondents in either group (<10%) indicated that software should be taught only in design classes or only in analysis classes.



a) Where to put software



b) What technique to engage with

Figure 13: Perceptions on where to place and what techniques to engage with.

Knowing class level integration perceptions, Fig 13b provides new insights into strategies for software interaction and exposure. Faculty respondents revealed strong agreement that software should be integrated through course projects (78.6%), with dedicated assignments being next (35.7%). Among industry respondents, the most frequently reported approaches were equally divided between dedicated assignments and course projects (55.9% each). Connecting software to multiple or many assignments was reported by only 23.5% of industry respondents, while 0% of faculty reported this option. Both groups reported relatively poorly to optional “extra” credit activities, likely because they are not priority learning opportunities for students.

The final broad item to consider in software integration is the selection of software platforms or package (e.g.: SAP 2000, RAM SS, RISA, spColumn, Tekla, etc.) is important for two reasons. First, software capabilities must align with the skills being reinforced [3,8] and the second is financial resources availability within the department. Among industry (Fig. 14), the majority (57.1%) favored the integration

of multiple software applications across different courses, while faculty most often preferred a single platform supporting both analysis and design (50.0%). Neither cohort indicated support for allowing students to independently select their software platform. Industry recommendations align with connecting topics to the most suitable platform, while faculty prefer a single-platform approach to simplify instruction (to avoid multiple interfaces). That said, no single software necessarily does all types of analysis and/or design in a manner that likely best matches course objectives [53].

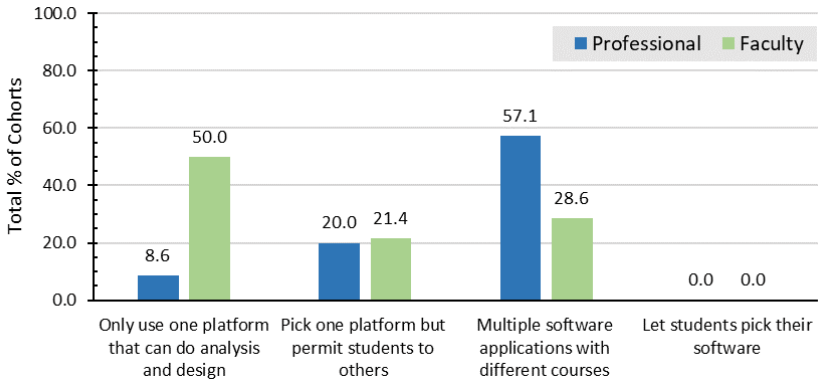


Figure 14: Perceptions on what approach to pick software for integration.

Discussions on Software Integration

The following discussion draws on qualitative open-ended responses within the survey data to provide richer context from the previously identified patterns and to situate the results for a broader curricular set of recommendations to come.

Both faculty and industry respondents generally agree that traditional hand analysis methods have their place in the structural curriculum as a foundational knowledge base. Differences emerge in how many and which methods to teach. Faculty regularly emphasize traditional methods to best suit a problem type (e.g. truss deflections) or those that have been historically taught. Some industry, though, recommend teaching just enough to instill the core fundamentals or to prepare them for the FE/PE/SE exam. Industry further stressed the importance of equipping students with the proper tools and, equally important, the processes routinely conducted to evaluate results (e.g.: approximate methods and benchmarking studies). A shared priority is developing conceptual knowledge of structural behavior to enable interpretation and verification of results.

Varying levels of agreement exist on whether software can serve a complementary—or even concurrent—role in critical skill building. While faculty see software as a more pronounced as a way to introduce industry opportunities to connect hand calculation validity, industry respondents stressed instruction must recognize the limitations of software modeling, teach how to properly validate output, and establish student leveraging software that avoids treating software as a "black box." One industry professional noted the importance of avoiding the misconception that general modeling is "easy," because *"…seemingly harmless defaults that students / professionals can click can drastically affect results…"* Several more industry professionals advocated for more education on 1) how to spot when the results appear to be incorrect and 2) how to prompt students to question if results are correct.

Both groups support selective, structured integration of software. Faculty typically introduce software in later or elective courses, often in conjunction with capstone projects, to reinforce concepts and provide exposure to modeling tools as capstone is a natural location in a cramped curriculum. Industry indicates more flexibility by spreading knowledge out, supporting repeated exposure for verification and practical application can help students develop structural intuition and engineering judgment. Both groups also agree that software package selection should be deliberate and that fewer tool platforms should be used. Instead, more mastery exploration of features and limitations are of value (transferable skills if done right). Both groups caution against teaching multiple platforms superficially and allowing unrestricted student choice. In all cases, software is framed as a tool for enhancing and reinforcing understanding and critical thinking rather than a replacement.

Recommendations to Educators Moving forward to Balance Traditional and Software

In reflecting back on these findings, coupled with already explored literature mechanisms and the authors' own experience with integration, several practical recommendations emerge. While the results above seek to strike a balance between software and hand methods, integration remains. Adding to this, conceptual understanding and being able to build structural intuition or engineering judgement was also a trend. As such, provided here are recommendations to advance our teaching status quo to balance traditional methods and computational modeling, including guidance on selecting hand-calculation topics, choosing appropriate platforms strategically, and reinforcing conceptual understanding through parallel exercises in both hand calculations and software.

While the results above seek to strike a balance between software and hand methods, integration remains central. Additionally, conceptual understanding and the development of structural intuition or engineering judgment were recurring themes. Accordingly, the recommendations below aim to advance our teaching status quo to balance traditional methods and computational modeling, including guidance on selecting hand-calculation topics, choosing appropriate platforms strategically, and reinforcing conceptual understanding through parallel exercises in both hand calculations and software.

Shift in Hand Calculation Topics and Methods

Earlier results indicated that while hand calculations remain critical for developing fundamental understanding, not all traditional methods are needed. Key software themes to prioritize are: allow students to verify computational results and emphasize qualitative approaches more directly aligned with structural engineering practice [41]. Based on these findings, the following recommendations are offered.

- Reduce the number of hand calculation methods taught to the most leveraged methods industry [41]:
  - Method of sections, moment distribution, virtual work, direct integration and plastic analysis.
- Strongly emphasize and deepen discussions on qualitative to semi-quantitative methods to connect to intuition [41, 49]:
  - Allocate more time (or add if none exists) on how to predict results for trusses, beams and frames prior to formal modeling to hand calcs.
  - Cover approximating truss behavior (movement and T/C/Zero force) qualitatively.
  - Emphasize deeper exploration of internal force diagram construction using EoE and fundamental shape assumptions by deduction (particularly for indeterminate)
  - Strengthen instruction on deformation sketching with connections to stiffness and support impacts.
- Expand approximate method instruction and tie them in parallel with traditional methods [51]:
  - Spend time on the various approximate methods in Fig. 3 to the extent students become comfortable and confident with them. These can be spread across courses.
  - Avoid isolating approximation techniques into a single module; instead, introduce them when they are most relevant alongside other core content.
  - For parallel learning, do comparisons. An example could be approximate frame drift vs that of virtual work.
  - Teach design heuristics (approx. techniques) in design classes improve Rule of Thumb (RoT) sizing skills and to better interpret design results.
- Expand load path discussions for gravity and lateral systems using both idealized (textbook) and more real-life examples (from industry).
  - Include: sketching, load flow, transfer systems, support conditions, and both gravity and lateral systems.

Software Integration into Activities, Assessments, and the Curriculum

Software integration can be done on different scales of economy depending on the program size and the composition of courses. Integration should be done both analysis and design courses. It is important that instructional examples be explicitly demonstrated and discussed and not just attached to assignments [3]. Based on these considerations, the following recommendations are proposed.

- Periodically incorporate examples or homework problems that explicitly demonstrate software vs. hand calculations results. This solidifies whether methods can: 1) produce exact results (for frames elements) or 2) to showcase slight differences.
  - Adopt one comparison example per method (as a start). Consider beams, frames and walls.
  - Adopt one software integrated example per unique topical homework set.
  - Deliver demonstrations via live, recorded, or written summarized and paired with in-class discussion.
- For class projects (particularly design and capstone), allow software utilization.
  - Focus on iteration, optimization, and evaluation of design alternatives.
  - Require spot-checking of results using hand calculations, approximate methods, or qualitative reasoning.
- In design courses, have students develop spreadsheets for capacity calculations or sizing, as this is a core practice skill.
- Strategically integrate software at the curriculum level.
  - Adopt at least one platform that is reinforced across courses and assignments.
  - For design exploration, consider a more robust software capable of gravity/lateral design capabilities for projects and capstones. RAM SS and RISA are good resources.
  - For design teaching, consider isolated software like spSuite (e.g. beam, column, mat, etc.)
  - Pick software that is viable and helpful for capstone studies. Alumni of your program should be using these platforms as it's a way to get instructional support.
- Leverage software to better visualize structural behavior.
  - Use software-generated animations and visual outputs to illustrate complex or commonly misunderstood behaviors [32]. Examples can be: stress profile change as loading changes, seeing building lateral periods, and many more can be done.

Iterative and Interactive Studies with Software to Learn Behavior Impacts

To develop engineering judgement, students need to actively explore critical thinking [12,36]. When coupled with educational psychology techniques [41, 44], iterative and interactive learning strategies can start developing this mindset. The goal here is to leverage technology by allowing software to streamline computation and to focus on interpretation and comparison reasoning. Given this, the following recommendations are proposed.

- To build intuition, require students to first guess or approximate an answer before doing hand (or software) analysis [49].
- Have students compare and contrast similarities and differences between problems within a set to reinforce topics connected to scenarios [54].
- Incorporate more exploratory problem sets where variables are systematically adjusted and students to summarize key trends [50].
  - Use software to cut down on the work. Vary parameters such as: stiffness, support conditions, loading, materials, geometric characteristics, etc.
  - Apply these explorations to single members, frames, and trusses, as well as larger structural systems such as bays, floor systems, and lateral-force-resisting systems.
  - Further implement in design courses (e.g., in design, vary materials, unbraced lengths, or key design properties, et.).

- Incorporate interactive media (e.g., GeoGebra [31]) to permit real-time exploration that builds a sense of the physical meaning of different analysis / design parameters.
- Adopt scenario-based learning when creating homework, examples, and assessments to better reflect professional practice context over "textbook language" and often better engage students [18].

Software Scope: Teaching about Settings, Mechanics, and Validation

To support effective and responsible software use, instruction needs to extend beyond procedural interaction to include an understanding of settings, underlying assumptions, and validation practices. Emphasis should be placed on guiding students to interpret software output, recognize modeling implications, and develop strong verification and validation habits. Based on these needs, the following recommendations are proposed.

- Provide tutorials to help start software knowledge acquisition.
  - Tutorials must explain why specific steps and settings are done/used, not merely what to click.
  - Leverage vendor documentation and industry resources to support these explanations.
  - Connect key software mechanisms with user interface "options" (defaults vs others) and what they mean to structural behavior and / or design.
- Deemphasize the mathematics of the direct stiffness method and finite element method at the UG level. Maintain context though.
  - Provide an introduction into why the theory is important and how changing software settings relate.
  - Provide important context on what changing settings might/will do. (ex. switching between meshing options in Etabs for walls).
- Develop validation skills explicitly.
  - Focus on evaluation skills related to features of a solution, establishing upper and lower bounds, etc.
  - Provide context into common modeling errors for different situations. This includes why they exist and how to find them.
- Leverage a mixture of correct and intentionally incorrect computer models and their results. Ask students to explore and explain discrepancies.
  - Vary the level of guidance (hints) and assumptions that adopt scaffold learning. That said, grounded scenarios need described.

Conclusion

In summary, both faculty and industry perspectives converge on the importance of grounding software use with traditional and approximate methods by focusing on selective assignments and adopting deliberate platform choices. Differences primarily relate to the timing and scope of software exposure, with industry slightly favoring earlier engagement for long-term applied skill development. Together with tested practices in literature, the recommendations set forth here can start to strategically integrate software in a structural curriculum. A core theme to the recommendations is software complements, not replace, foundational instruction, preparing students to interpret computational results critically. If an approach can be taken to not teach the "black box" mindset, or not only teach hand methods for the sake exposure, then classroom takeaways can be more concept centered.

References

[1] Plata, A. (2017). "Origins of the Profession: The Role of History and Design in the Education of Structural Engineers" *Structures Congress 2017* April 6–8, 2017 | Denver, Colorado

[2] Bhatia, A. and Chen, C. (2015). "Active Learning Pedagogies Promoting the Art of Structural and Civil Engineering" *Proceedings of the ASEE Annual Conference & Exposition*, 2015

[3] Aparicio, A.C. and Ruiz-Teran, A.M. (2007). "Tradition and Innovation in Teaching Structural Design in Civil Engineering", *Journal of Professional Issues in Engineering Education and Practice*, 133(4), 340–349

[4] Grant, D.M., Malloy, A.D. & Murphy, M.C. (2009). "A Comparison of Student Perceptions of their Computer Skills to their Actual Abilities." *Journal of Information Technology Education: Research*, 8, 141-160. Informing Science Institute.

[5] Glover, D., Miller, D., Averis, D., and Door, V. (2007). "The evolution of an effective pedagogy for teachers using the interactive whiteboard in mathematics and modern languages: an empirical analysis from the secondary sector," *Learning, Media and Technology*, 32:1, 5-20.

[6] Horne, M., & Hamza, N. (2006). "Integration of virtual reality within the built environment curriculum". *Journal of Information Technology in Construction (ITcon)*, *11*(23), 311-324.

[7] Fox, A.B., Jonathan Rosen, and Mary Crawford. (2009). "Distractions, distractions: does instant messaging affect college students' performance on a concurrent reading comprehension task?". *CyberPsychology & Behavior*. 12(1): 51-53.

[8] El-Sawy, K. M., & Sweedan, A. M. I. (2010). "Innovative use of computer tools in teaching structural engineering applications." *Australasian Journal of Engineering Education*, *16*(1), 35-54.

[9] Bishay, P. L. (2020). "Teaching the finite element method fundamentals to undergraduate students through truss builder and truss analyzer computational tools and student-generated assignments mini-projects." *Computer Applications in Engineering Education*, *28*(4), 1007-1027.

[10] Vázquez-Boza, M., Justo, E., & Delgado, A. (2015). "The evolution of structural engineering education in the era of computer." *In 3rd International Conference on Mechanical Models in Structural Engineering* (pp. 712-729).

[11] Davalos, J. (2003). "Neoclassical active learning approach for structural analysis." In *2003 Annual Conference* (pp. 8-873).

[12] Martino, N., & Ghanem, A. (2016), "*Innovative Approach to Teaching Applied Structures Courses". 2016 ASEE Annual Conference & Exposition*, New Orleans, Louisiana. 10.18260/p.25717

[13] Forcael, E., Garcés, G., Bastías, E., & Friz, M. (2019). "Theory of teaching techniques used in civil engineering programs." *Journal of Professional Issues in Engineering Education and Practice,* 145(2), 02518008.

[14] Lanning, J., & Roberts, M. (2019). "Fighting "Plug and Chug" Structural Design through Effective and Experiential Demonstrations." *2019 ASEE Annual Conference & Exposition Proceedings*, Tampa, Florida, 2019/06/15,

[15] McCoy, C., Phillips, J. J., Ramming, C. H., & Guyer, C. (2021). "Development of a Structural Loadings Course for Architectural Engineering Students." *2021 ASEE Virtual Annual Conference Content*.

[16] Barner, M. S., Brown, S. A., & Linton, D. (2021). "Structural engineering heuristics in an engineering workplace and academic environments." *Journal of Civil Engineering Education*, *147*(2), 04020014.

[17] MacRobert, C. J. (2018). "Introducing engineering judgment through active learning." *Journal of Professional Issues in Engineering Education and Practice*, *144*(4), 05018009.

[18] Barner, M. S., Adam Brown, S., Bornasal, F., & Linton, D. (2022). "Tangibility of representations in engineering courses and the workplace." *Journal of Engineering Education*, *111*(1), 162-184.

[19] Plevris, V. (2025). "A Glimpse into the Future of Civil Engineering Education: The New Era of Artificial Intelligence, Machine Learning, and Large Language Models." *Journal of Civil Engineering Education*, *151*(2), 02525001.

[20] ASCE (2019). *Civil Engineering Body of Knowledge (CEBOK) for the 21st Century: Preparing the Civil Engineer for the Future*, 3rd Edition. American Society of Civil Engineers, Reston, VA.

[21] Kam-Biron, B. Perkins, S. Francis. (2022). "NCSEA 2021 Practitioner Survey." *STRUCTURE.*

[22] ABET (2023). "Criteria for Accrediting Engineering Programs, 2023 – 2024" https://www.abet.org/wp-content/uploads/2023/03/23-24-EAC-Criteria_FINAL2.pdf

[23] Swenty, M.K.; and Swenty, B.J. (2023). "Alignment of civil engineering programs with the ASCE body of knowledge." *Journal of Civil Engineering Education*, 149(4), 04023006.

[24] Powell, G. H. (2008). "Structural analysis: Are we relying too much on computers? Part 1: The problem." *Structure Magazine*, Nov., 50-52.

[25] Fowler, D., N. Poling, A. Whitney, J. Morgan, and K. Brumbelow. (2015). "Data-driven curriculum redesign in civil engineering." *In Proc., Frontiers in Education Conf.*, FIE. New York: IEEE.

[26] B. Perkins. (2016). "2016 NCSEA Structural Engineering Curriculum Survey." *STRUCTURE*. https://www.structuremag.org/?p=10449

[27] S. M. Francis. (2021). "2019 NCSEA Structural Engineering Curriculum Survey Results." *STRUCTURE.* https://www.structuremag.org/?p=17717

[28] Lanning, J., Roberts, M. W., & Wiggins, B. K. (2024). "Exploring Educational Needs and Practices in Structural Analysis." *In 2024 ASEE Annual Conference & Exposition*.

[29] Brown, S., B. Lutz, N. Perova-Mello, and O. Ha. (2019). "Exploring differences in statics concept inventory scores among students and practitioners." *Journal of Engineering Education*, 108(1), 119-135.

[30] Cuadra, C. (2010). "Challenges in building structure engineering education." *Proc., Int. Conf. on Education and Educational Technology, World Scientific and Engineering Academy and Society*, Stevens Point, WI, 123–125.

[31] Lanning, J., & Badrya, J. (2021). "Interactive Online Figures for the Core Concepts in Structural Steel Design." *2021 ASEE Virtual Annual Conference Content Access*.

[32] Dittenber, D., & Fredette, L. (2022). “Forming Cognitive Connections: Desktop Learning Modules, Structural Analysis Software, and Full-Scale Structures.” *2022 ASEE Annual Conference & Exposition*.
[33] Kitipornchai, S., Lam, H. F., and Reichl, T. (2009). “A new approach to teaching and learning structural analysis.” *Proc. Int. Conf. on Computer Supported Education*, Setúbal, Portugal, 379–383.
[34] Ali, M. (2015). “An overview of research on experts and novice problem solvers in physics.” *Bulletin Persatuan Pendidikan Sains dan Matematik Johor* 25 (1): 70–75.
[35] Fernández-Sánchez, G., & Millán, M. Á. (2013). “Structural analysis education: Learning by hands-on projects and calculating structures.” *Journal of Professional Issues in Engineering Education and Practice*, *139*(3), 244-247.
[36] De Justo, E., & Delgado, A. (2015). “Change to competence-based education in structural engineering.” *Journal of Professional Issues in Engineering Education and Practice*, *141*(3), 05014005.
[37] Romero, M., and Museros, P. (2002). “Structural analysis education through model experiments and computer simulation.” *Journal of Professional issues in engineering education and practice*, 128(4), 170-175.
[38] Nelson, J., & Yehia, S. (2004). “Structural Analysis Courses: Computers or Fundamentals.” In *2004 Annual Conference* (pp. 9-1124).
[39] Mtenga, P., and Spainhour, L. (2000). “Applications of mathematical software packages in structural engineering education and practice.” *J. Comput. Civ. Eng*., 14(4), 273–278.
[40] McDaniel, C., & Archer, G. (2009). “Developing a Feel for Structural Behavior.” In *2009 Annual Conference & Exposition* (pp. 14-441).
[41] Hanson, J. (2022). “Teaching students how to evaluate the reasonableness of structural analysis results.” *Journal of Civil Engineering Education*, 148(1), 04021013.
[42] Shaikh, F. (2012). “Role of commercial software in teaching finite element analysis at undergraduate level: a case study.” *Journal of engineering education*, 7(2), 2-6.
[43] Simonen, K. (2014). "Iterating structures: Teaching engineering as design." *Journal of Architectural Engineering* 20.3 (2014): 05014003.
[44] Gilbert, J. K., Bulte, A. M., & Pilot, A. (2011). “Concept development and transfer in context-based science education.” *International Journal of Science Education*, 33(6), 817-837.
[45] Dollár, Anna, and Paul Steif. "Understanding Internal Loading through Hands On Experiences." *2002 Annual Conference*.
[46] Baker, D. W. (2018). “The use of geogebra virtual interactives in statics to increase conceptual understanding.” In *2018 ASEE Annual Conference & Exposition*.
[47] Creswell, J. W. (1999). *Mixed-method research: Introduction and application.* In Handbook of educational policy (pp. 455-472). Academic press.
[48] Berends, M. (2012). *Survey methods in educational research.* In Handbook of complementary methods in education research (pp. 623-640). Routledge.
[49] Hanson, J. H. (2019.) *Structural analysis: Skills for practice*. Hoboken, NJ: Pearson.
[50] Nielson, B. G. (2022). *Structural analysis: understanding behavior*. John Wiley & Sons.
[51] Vrouwe, I., Dissaux, T., Jancart, S., Stals, A., Werner, L. C., & Koering, D. (2020). “Concept Learning Through Parametric Design; A Learning Situation Design for Parametric Design in Architectural Studio Education.” *ECAADE 2020: Anthropologic-Architecture and Fabrication In The Cognitive Age*, VOL 2, 135-144.
[52] Grajdura, S., & Niemeier, D. (2023). “State of programming and data science preparation in civil engineering undergraduate curricula.” *Journal of Civil Engineering Education*, 149(2), 04022010.
[53] Solnosky, R. (2025). “Examining the Effectiveness of Software Strategies and Student Perceptions of Software in Building-Centered Capstones.” Journal of Architectural Engineering, 31(4), 04025040.
[54] Roelle, J., & Berthold, K. (2015). “Effects of comparing contrasting cases on learning from subsequent explanations.” *Cognition and Instruction*, 33(3), 199-225.