From calculations by hand to hands-on explorations: Re-thinking structural design pedagogy through interactive computational tools

Juney Lee

Carnegie Mellon University

Abstract

Traditional structural design education relies on hand calculations and analytical methods, reinforcing a longstanding divide between architecture and engineering disciplines in practice. While engineers emphasize numerical results and data-driven analysis, architects engage in more intuitive, form-driven explorations. This disciplinary split extends to academia, where differing accreditation standards and pedagogical approaches further entrench these distinctions. However, emerging computational tools offer new opportunities to bridge this gap by fostering interactive, hands-on learning experiences that enhance the intuition of architecture students learning structural design.

This paper proposes a new pedagogy for structural design in architecture education that integrates computational tools as the primary medium, shifting the focus from purely analytical methods to more interactive and exploratory learning. By leveraging digital platforms such as computational graphic statics, parametric structural modeling, and algorithmic form-finding methods, students can engage with structures dynamically with real-time feedback that reinforces both conceptual intuition and computational literacy.

The proposed curriculum consists of three structural design courses, each focusing on a distinct aspect of comprehensive structural education. These courses integrate various computational tools progressively, increasing in depth and complexity as students advance. The sequence is designed to build a strong foundation of knowledge and skills, preparing students for integrated design studios in the later stages of their education. In these studios, students apply the technical skills acquired in the structural design sequence within a holistic design context, demonstrating their ability to integrate structural design into their architectural projects in innovative ways as required by the *National Architectural Accrediting Board* (NAAB).

Exemplary student work from the first implementation of this new curriculum is presented, illustrating how computational approaches can transform how students engage with technical subject matter and leverage it to explore new design possibilities that were previously unattainable. The primary objective is to equip architecture students with the skills needed to meet the rapidly evolving demands of the *architecture, engineering, and construction* (AEC) industry. By shifting from static calculations to interactive digital explorations, we can cultivate a new generation of architects capable of innovating across and beyond traditional disciplinary boundaries. The paper concludes with a discussion on current developments and future efforts to further integrate emerging computational technologies, with the overarching aim of discovering new methods of teaching and learning structural design.

Keywords: Teaching, pedagogy, structural design, computational structural design, integrated design, graphic statics, parametric structural design, optimization, form finding, augmented reality, virtual reality

1. The pedagogical and disciplinary divide

The divide between architecture and structural engineering in academia is largely shaped by distinct professional standards, accreditation regulations, and licensure processes. Architectural education is deeply rooted in qualitative evaluation, where student work is assessed through design reviews that emphasize conceptual clarity, spatial exploration, formal articulation, and graphical representation. In contrast, structural engineering education relies on quantitative evaluation, where students are assessed through problem sets and exams that require precise numerical solutions (Fig. 1.). This fundamental difference in pedagogical approach reflects the varying professional expectations for architects and engineers, reinforcing the gap between the two fields in both academia and practice.

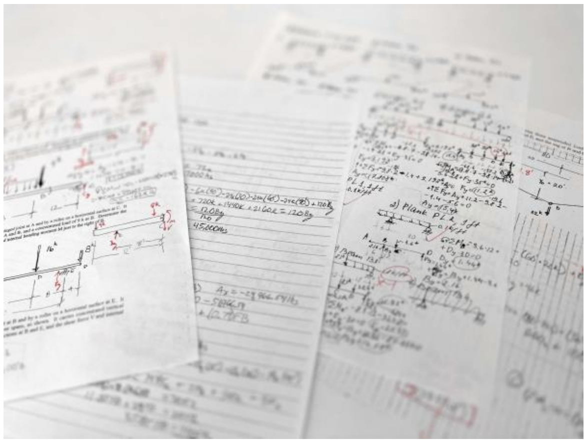


Fig. 1. Problem-set-based homework assignments featuring manual hand calculations from structural design courses at Carnegie Mellon University prior to the 2023–24 academic year.

This divergence raises an important question: when teaching structural design in an architecture school, which approach is more appropriate? Should it be a simplified version of what structural engineering students learn, or should it take a fundamentally different approach? Since architecture students will not be responsible for sizing columns or drawing moment diagrams in their professional careers—tasks that structural engineers are better trained to handle—structural education for architects must be rethought. Instead of training architecture students to perform engineering calculations in a rudimentary manner, structural design education in architecture should focus on enhancing their intuition and sensibilities as designers. As future orchestrators of complex projects involving multiple disciplines including structural engineering, architecture students must develop a deeper understanding of structural principles to facilitate informed, interdisciplinary collaboration.

As technology advances, computers will continue to outperform humans in speed and accuracy for structural calculations. The objective, therefore, should not be to teach computational tools merely as a means to optimize traditional workflows or automating known tasks within independent disciplines. Instead, digital technologies should be leveraged to revolutionize how students design and learn. As Richard Hamming once stated, “the purpose of computing is insight, not numbers.” The true value of computation lies in its ability to generate new insights, allowing human designers to move beyond calculation and focus on exploring new frontiers of exclusively innate human skills: discovering original ideas, creative improvisation, and imagining new possibilities.

2. Towards a more computational approach

The pedagogical separation between architecture and structural engineering is further reinforced by the inherent differences in the computational tools used in each field. These tools have been developed to meet the specific needs of their respective disciplines, optimizing workflows for efficiency within their domains. Consequently, they shape not only how architects and engineers work but also how they think about design and problem-solving, further entrenching disciplinary boundaries.

Architectural computational tools, such as Rhinoceros (McNeel, 2025), prioritize generative geometric capabilities, enabling the rapid creation and iteration of complex three-dimensional (3D) models in tandem with parametric modeling plugins such as Grasshopper. These tools allow architects to explore design options quickly, often detached from real-world constraints such as physics and material behavior. While this geometric freedom fosters unhindered exploration of the design space, it also leads to a disconnect between architectural possibility and structural feasibility.

In contrast, engineering computational tools, primarily based on the *finite element method* (FEM), solve complex equations simultaneously to provide precise analytical results. These tools evaluate structural behavior, assess whether a design is functional, and quantify performance metrics. The fundamental difference is that while architectural software is generative and qualitative, engineering software is analytical and confirmatory. These analytical tools require predefined geometry as input before any analysis can be performed, making them inherently non-creative by design.

Building Information Modeling (BIM) software, such as Revit (Autodesk, 2025), is widely used in practice today as a common platform for architects and engineers. However, BIM serves primarily as a documentation and coordination tool and does not inherently promote innovative thinking that crosses disciplinary boundaries; instead, it streamlines established methodologies rather than encouraging creative disruption.

The adoption of standardized tools that cater to each discipline separately—and the emergence of BIM as a divisive rather than unifying digital language—can be seen as a major obstacle to truly innovative structural design thinking. As buildings become more operationally efficient in the coming decades, attention will shift toward reducing embodied energy, which accounts for a significant portion of a building's environmental impact. Since structural mass contributes substantially to the total embodied energy of a building, it is imperative that we rethink not only how structures are designed but also how structural design is taught to architecture students, who will ultimately shape the buildings of the future.

To address these greater disciplinary and environmental challenges, we must reconsider how computational tools are used, not merely as efficiency-enhancing instruments but as catalysts for innovative thinking. If computation is only used to optimize conventional workflows, the environmental crisis that the building industry significantly contributes to will not improve. New approaches must be explored, whether by repurposing existing digital tools in dramatically different ways or inventing entirely new ones, to challenge the status quo and develop alternative ways of learning, thinking, and making.

3. New curriculum

To address these challenges, a structured curriculum that integrates computational tools at different levels of architectural education is proposed. This new curriculum was introduced at Carnegie Mellon University's School of Architecture during the 2023-2024 academic year. The curriculum consists of four courses:

- Structural Design 1: Form and Forces (6 units, 2nd-year Bachelor of Architecture)
- Structural Design 2: Materials and Analysis (9 units, 3rd-year Bachelor of Architecture and 1st-year Master of Architecture)
- Structural Design 3: Computational explorations (9 units, elective)
- Integrated design studios (18 units, 4th-year Bachelor of Architecture and 1st-year Master of Architecture)

3.1 Structural Design 1: Form and Forces

Structural Design 1 (SD1) is the first course in the structural design sequence, introduced to second-year

Bachelor of Architecture students. At this early stage, the focus is on understanding the relationship between geometry of structural form and the equilibrium of forces, independent of material properties and internal stresses. This foundational knowledge allows students to grasp structural behavior conceptually before engaging with more complex analytical methods.

Graphic statics serves as an ideal tool for teaching and learning structures at this stage, as it enables students to understand static equilibrium through drawing the force vectors rather than through solving equations (Allen and Zalewski, 2010). By visually constructing form and force diagrams, students develop an intuitive understanding of how forces interact within a structure. This approach encourages them to think visually and creatively about structural behavior, reinforcing insights that are often more accessible at this stage than purely analytical methods (Van Mele et al., 2012). It has been shown that graphic statics improves comprehension for students in the early stages of education, as their visual reasoning skills are typically more developed than their analytical thinking abilities (Enrique et al., 2019).

Although graphic statics is a powerful tool for understanding structures, manually drawing form and force diagrams by hand can become a tedious and time-consuming process, especially as structural systems grow in complexity. Traditional methods often lead to disengagement when the manual effort outweighs the conceptual insights gained. However, by integrating the principles of graphic statics with the parametric capabilities of modern *Computer-Aided Design* (CAD) software, students can overcome this challenge. Interactive tools allow for the automatic generation of form and force diagrams, enabling real-time visualization and modification of structural geometries (Van Mele and Block, 2014). This dynamic feedback mechanism keeps students engaged, fostering curiosity about structural behavior while eliminating the mechanical burden of hand-drawing each diagram (Lee et al., 2021).

Fig. 2. Students using interactive graphic statics drawings to study the form and force diagrams of local bridges.

To enhance this learning experience, interactive graphic statics tools developed by the author in collaboration with colleagues from research community are used in the course (Avelino et al., 2021). These tools provide students with an interactive and exploratory introduction to fundamental structural concepts, covering funicular structures such as cables, arches, and trusses. Through a series of exercises and design projects, students are encouraged to think about structures without being constrained by abstract equations and numerical calculations (Fig. 2).

One example of an exercise used in the course involves analyzing local bridge structures using computational graphic statics plugins in Rhinoceros. This exercise allows students to explore how various stiffening mechanisms in funicular bridge structures play a critical role in addressing asymmetric loading conditions (Fig. 3). By engaging with these computational tools in a hands-on manner, students not only develop a deeper understanding of structural behavior but also gain confidence in applying structural principles creatively within their architectural designs. The exploratory nature of SD1 is also paired with their “second year option” studios, where topics are much more broad and experimental than practice-based.

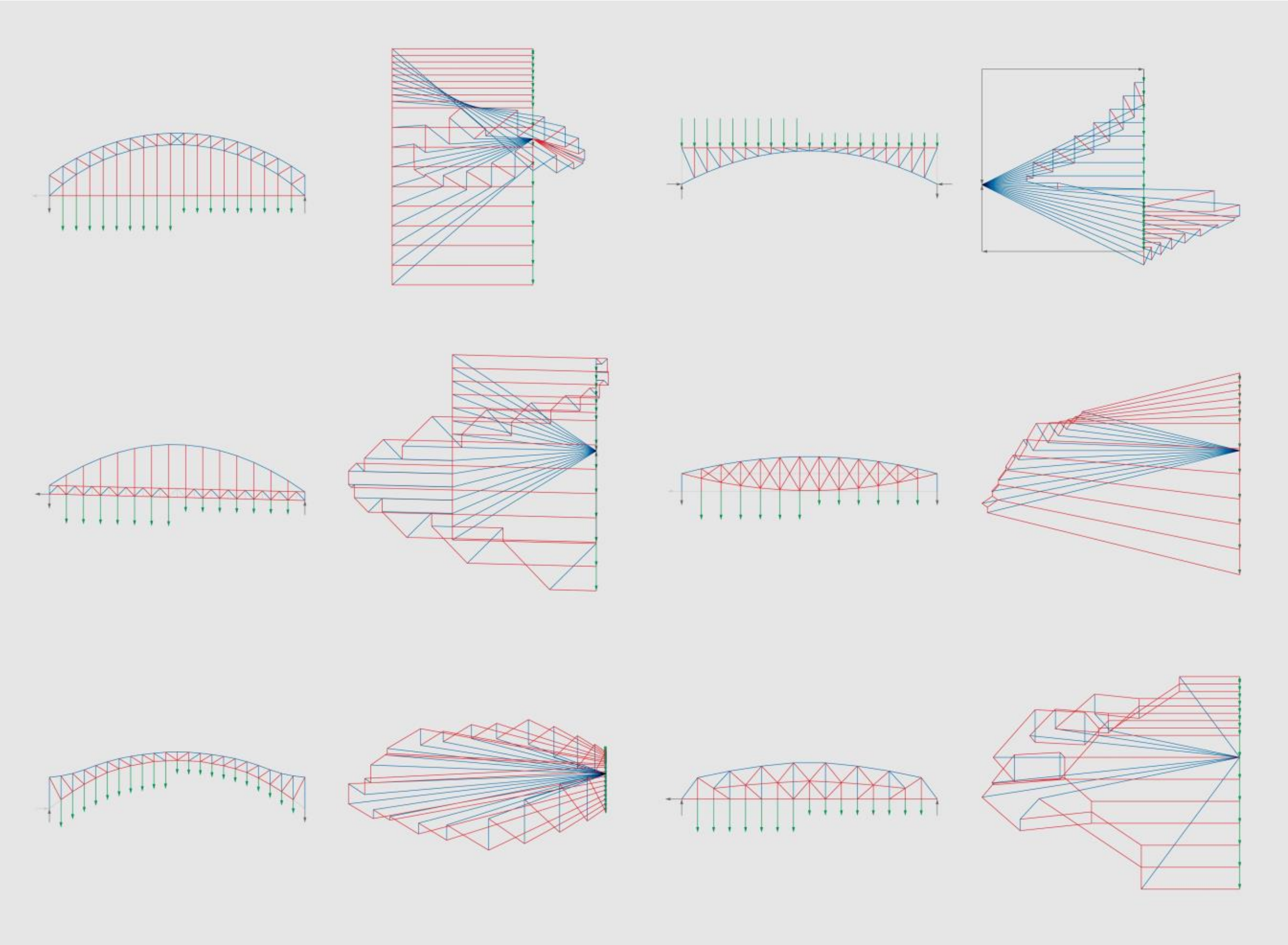


Fig. 3. Form and force diagrams of Pittsburgh bridges under asymmetric loads, generated by students using computational graphic statics tool in Rhinoceros (work by students from Bachelor of Architecture class of 2026).

3.2 Structural Design 2: Materials and Analysis

Structural Design 2 (SD2) is the second course in the sequence, focusing on the application of fundamental structural design principles learned in SD1 to the design of various structural components within a building. This course, taught to third-year Bachelor of Architecture students, shifts from the conceptual understanding of forces and geometry to the materialization of forces and the necessary analysis required to determine the appropriate size and performance of structural members, including columns, beams, floors, and lateral bracing.

Whereas SD1 emphasized the relationship between form and force independent of material properties, SD2 introduces students to the structural implications of materials. The course explores how different materials influence structural behavior, requiring students to analyze how large or small a given element must be under specific loads. While students inevitably learn fundamental equations—such as those governing the buckling of columns and the bending of beams—these concepts are not taught in isolation. Instead of relying solely on quizzes or static problem sets, students engage with computational tools that allow them to experiment, test, and visualize how structural systems respond to various conditions. This hands-on, exploratory approach reinforces learning through interactive engagement rather than memorization of absolute solutions.

A key computational tool used in SD2 is Karamba (Preisinger, 2013), a parametric structural modeling and analysis plugin for Grasshopper in Rhinoceros. The parametric nature of Karamba makes it particularly well-suited for students at this stage, as they will have already taken required computational design courses and are actively integrating digital tools into their design studios. The combination of parametric modeling and physical prototyping allows students to directly compare physical test models with digital simulations, helping them understand both the capabilities and limitations of computational tools. This comparative approach teaches students not only how to use digital tools effectively but also how to critically assess whether the analysis results make sense based on their foundational knowledge of material behavior.

One example of an exercise in the course involves using Karamba to size three structurally equivalent beams, each spanning the same distance but made of different materials: steel, mass timber, and concrete. Instead of performing tedious hand calculations, students use Karamba's automatic sizing feature to quickly determine beam dimensions under applied loads. More importantly, they then link these parametric models to estimate the overall volume of each beam and evaluate its *global warming potential* (GWP). This exercise shifts the focus from purely structural sizing to the environmental impact of material choices, helping students understand how sustainability considerations influence structural design decisions (Fig. 4).

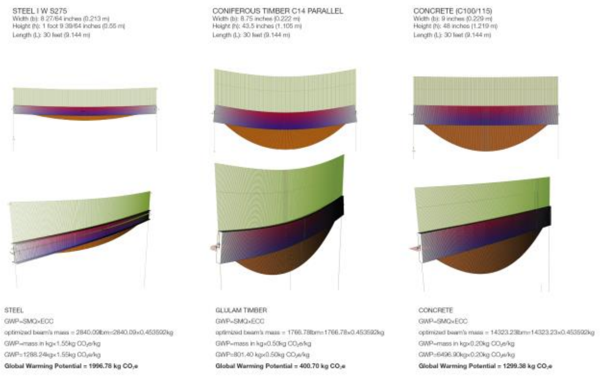


Fig. 4. Beam sizing exercise using Karamba, where students use the time that would have been spent for calculating beam sizes by hand to think about larger environmental implications of their material choices by comparing the global warming potential per unit weight of structural mass (work by Kaiwen Serena Sun).

The primary takeaway for students in SD2 is that while they must understand how structural tools work, they do not need to rely on manual calculations for every design decision. Instead, these tools allow for rapid, accurate calculations, enabling students to dedicate more time to critical decision-making and design exploration. By recognizing the strengths and limitations of computational methods, students learn to balance technical rigor with creative problem-solving, ensuring that digital tools serve as aids rather than substitutes for structural intuition.

SD2 is taught in parallel with the "Praxis II" design studio, a core part of the third-year Bachelor of Architecture curriculum. This studio emphasizes the integration of multiple disciplines into a coherent building design, with a particular focus on structural systems. Through this dual engagement with computational analysis and studio-based application, students develop the technical proficiency and design sensibility necessary to synthesize structure, materiality, and environmental impact into a coherent architectural solution.

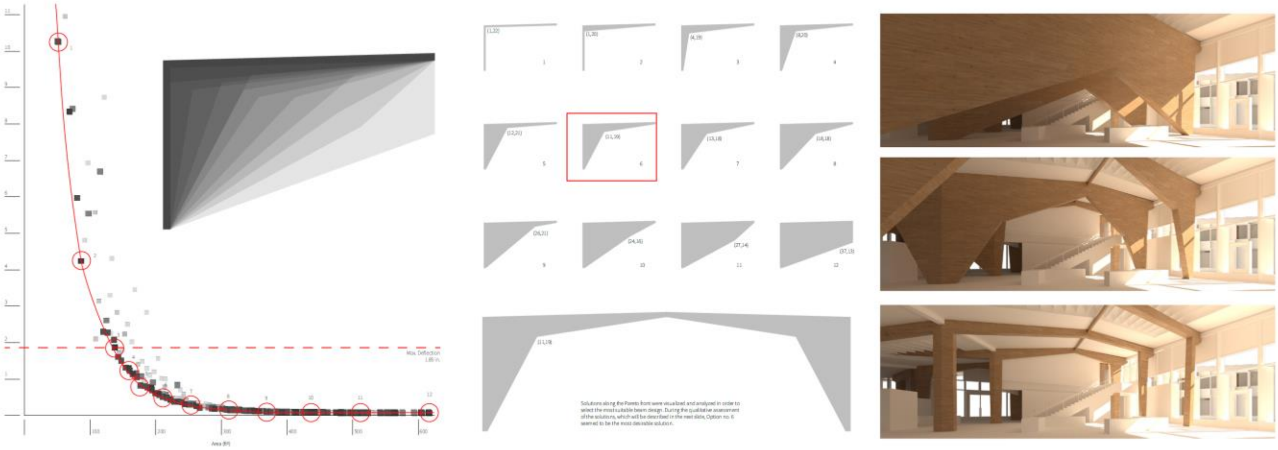


Fig. 5. Multi-objective structural optimization of a student's project from a prior studio, which forces students to challenge or confirm their initial design intentions with newly gained structural insights (work by Hazel Froling).

3.3 Structural Design 3: Computational Explorations

Structural Design 3 (SD3) is the third course in the structural design sequence, introducing students to state-of-the-art computational techniques for designing, shaping, and optimizing complex spatial structures. Unlike previous courses that focus on conceptual understanding (SD1) and hands-on application in building components (SD2), SD3 provides a platform for students to explore structural design more freely, using advanced optimization techniques and form-finding tools such as RhinoVAULT developed by the author in collaboration with various collaborators in the field (Rippmann et al., 2012; Van Mele and Lee, 2024) based on the COMPAS framework (Van Mele et al., 2019).

As an elective course, SD3 is available to Bachelor of Architecture students who have completed SD1 and SD2, as well as graduate students who have met the structural requirements at their previous institutions. The course builds on the iterative testing and real-time feedback capabilities of Karamba, expanding its potential when combined with other plugins from the Grasshopper ecosystem, particularly optimization tools such as Goat (Rechenraum, 2016) and Galapagos (Rutten, 2013). By integrating these computational tools, students experience firsthand how structural design can play a transformative role in the overall architectural design process.

One of the core design projects in the course challenges students to revisit and optimize a studio project from a previous semester using Karamba and optimization tools (Fig. 5). By working with a building design they are already deeply familiar with, students bring a heightened level of engagement and care to the optimization process. As they explore structural refinements aligned with their original design intent, they gain valuable insights into the balance between structural performance and architectural vision. Even when the optimization results diverge from their initial aesthetic expectations, students quickly recognize the impact of structurally-informed design tools—not only for performance-based decision-making but also for expanding their creative possibilities. This iterative process helps them develop stronger design intuition and become more confident decision-makers in future projects.

The open-ended final project in SD3 offers students an opportunity to pursue a computational structural design topic or theme of personal interest. Whether exploring form-finding techniques for funicular shell structures (Fig. 6), topology optimization of spatial systems, or data-driven structural analysis and algorithms (Fig. 7),

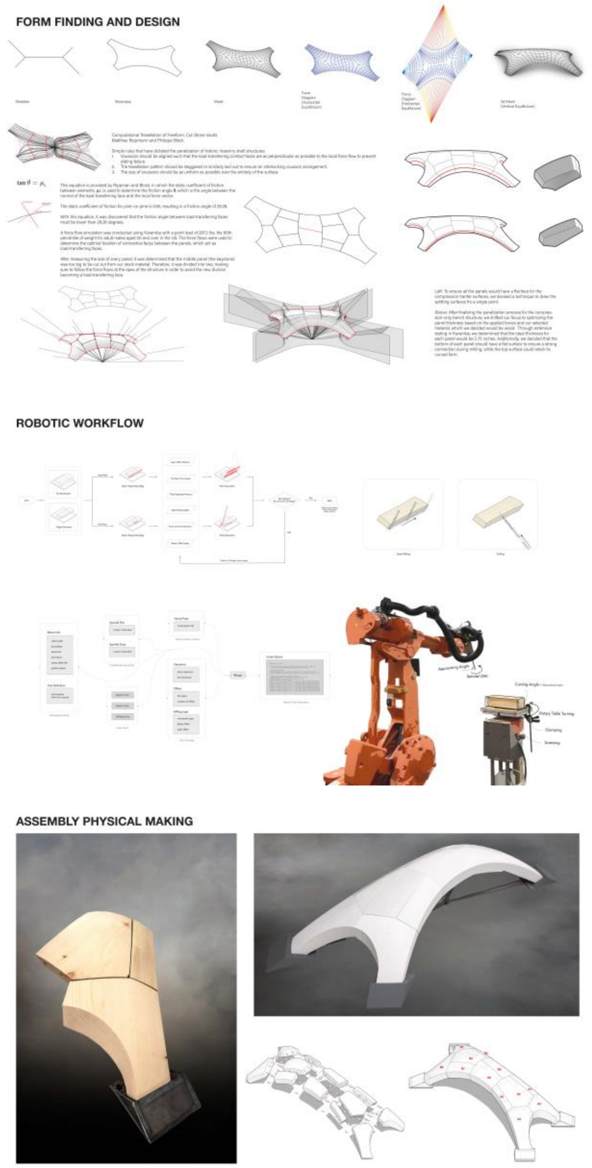


Fig. 6. Second design project from SD3 where RhinoVAULT was used to design a compression-only bench that is robotically milled from solid timber blocks (work by Hazel Foling, Darin Kim, Kaiwen Serena Sun, Eric Yu).

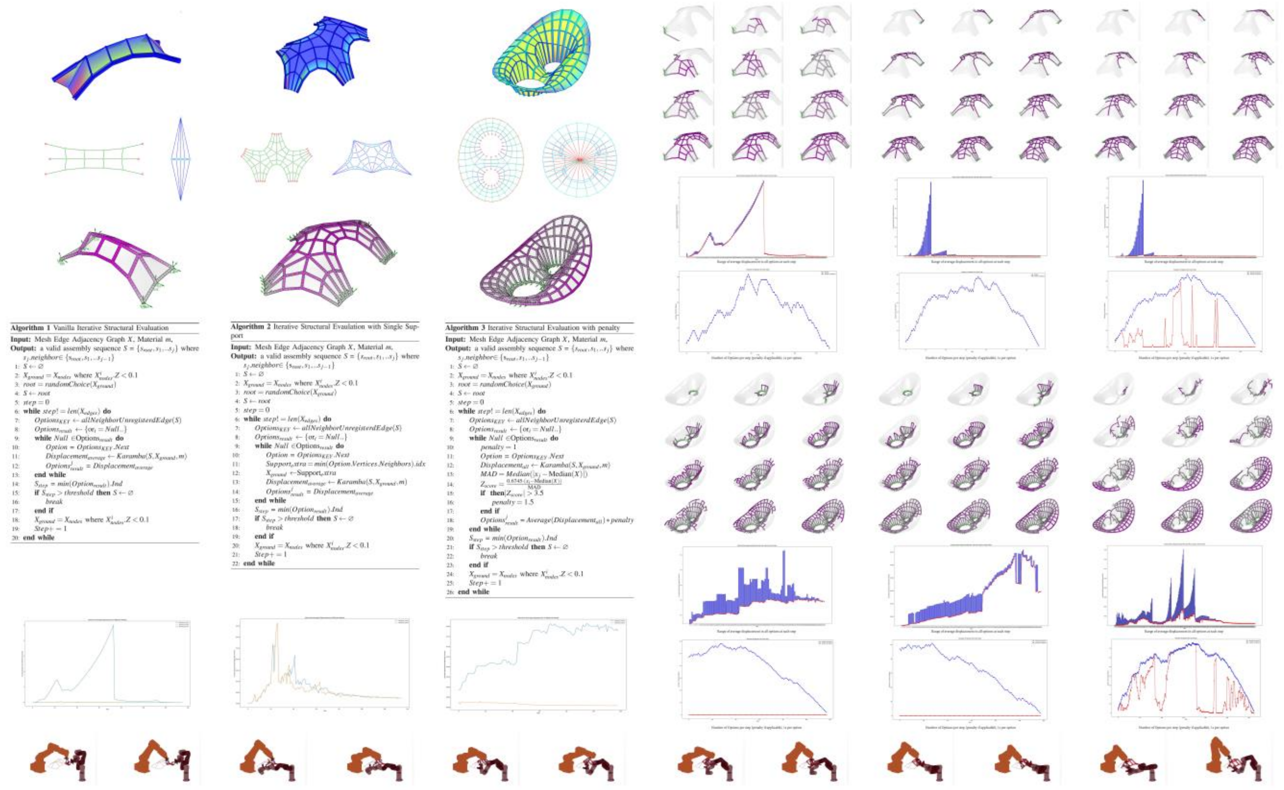
students are encouraged to develop their own computational workflows to achieve their unique design goals. This self-directed exploration fosters critical thinking, creativity, and technical fluency, equipping students with the skills to push the boundaries of structural innovation in architectural design.

3.4 Integrated Design Studios

The final component of the structural design sequence is the integrated design studio, where students apply the skills and knowledge acquired throughout the Structural Design courses to the comprehensive design of a building. This studio serves as a culminating experience, allowing students to synthesize their understanding of structural behavior, computational analysis, and optimization techniques within a fully developed architectural project.

Equipped with a strong conceptual foundation, the ability to create parametric structural models, and proficiency in optimization strategies, students move beyond structural analysis as a mere technical requirement. Instead, they use structural design as a creative driver, exploring how structural systems can inform and transform architectural form. By integrating these methodologies, students are able to challenge conventional building design paradigms, leveraging computational tools to push the boundaries of what is possible.

The work produced by students who have completed the structural design course sequence has demonstrated remarkable improvements in computational and geometric dexterity (Fig. 8, 9). They are able to generate, evaluate, and iterate through various complex geometries while making intelligent, well-informed design decisions that align with their larger architectural vision. This integration of structural thinking into architectural design empowers students to approach building design not just as an assembly of components, but as a holistic, performance-driven process where structure, form, and materiality work in harmony.



```
Algorithm 1 Vanilla Iterative Structural Evaluation
Input: Mesh Edge Adjacency Graph $X$, Material $m$,
Output: a valid assembly sequence $S = \{s_{root}, s_1, .. s_j\}$ where
    $s_j.neighbor \in \{s_{root}, s_1, .. s_{j-1}\}$
$S \leftarrow \varnothing$
$X_{ground} = X_{nodes}$ where $X^i_{nodes}.Z < 0.1$
$root = randomChoice(X_{ground})$
$S \leftarrow root$
$step = 0$
while $step! = len(X_{edges})$ do
    $Options_{KEY} \leftarrow allNeighborUnregistedEdge(S)$
    $Options_{result} \leftarrow \{or_i = Null..\}$
    while $Null \in Options_{result}$ do
        $Option = Options_{KEY}.Next$
        $Displacement_{average} \leftarrow Karamba(S, X_{ground}, m)$
        $Options^i_{result} = Displacement_{average}$
    end while
    $S_{step} = min(Option_{result}).Ind$
    if $S_{step} > threshold$ then $S \leftarrow \varnothing$
        break
    end if
    $X_{ground} = X_{nodes}$ where $X^i_{nodes}.Z < 0.1$
    $Step+ = 1$
end while
```

```
Algorithm 2 Iterative Structural Evaulation with Single Support
Input: Mesh Edge Adjacency Graph $X$, Material $m$,
Output: a valid assembly sequence $S = \{s_{root}, s_1, .. s_j\}$ where
    $s_j.neighbor \in \{s_{root}, s_1, .. s_{j-1}\}$
$S \leftarrow \varnothing$
$X_{ground} = X_{nodes}$ where $X^i_{nodes}.Z < 0.1$
$root = randomChoice(X_{ground})$
$S \leftarrow root$
$step = 0$
while $step! = len(X_{edges})$ do
    $Options_{KEY} \leftarrow allNeighborUnregisterdEdge(S)$
    $Options_{result} \leftarrow \{or_i = Null..\}$
    while $Null \in Options_{result}$ do
        $Option = Options_{KEY}.Next$
        $Support_extra = min(Option.Vertices.Neighbors).idx$
        $X_{ground} \leftarrow Support_extra$
        $Displacement_{average} \leftarrow Karamba(S, X_{ground}, m)$
        $Options^i_{result} = Displacement_{average}$
    end while
    $S_{step} = min(Option_{result}).Ind$
    if $S_{step} > threshold$ then $S \leftarrow \varnothing$
        break
    end if
    $X_{ground} = X_{nodes}$ where $X^i_{nodes}.Z < 0.1$
    $Step+ = 1$
end while
```

```
Algorithm 3 Iterative Structural Evaluation with penalty
Input: Mesh Edge Adjacency Graph $X$, Material $m$,
Output: a valid assembly sequence $S = \{s_{root}, s_1, .. s_j\}$ where
    $s_j.neighbor \in \{s_{root}, s_1, .. s_{j-1}\}$
$S \leftarrow \varnothing$
$X_{ground} = X_{nodes}$ where $X^i_{nodes}.Z < 0.1$
$root = randomChoice(X_{ground})$
$S \leftarrow root$
$step = 0$
while $step! = len(X_{edges})$ do
    $Options_{KEY} \leftarrow allNeighborUnregisterdEdge(S)$
    $Options_{result} \leftarrow \{or_i = Null..\}$
    while $Null \in Options_{result}$ do
        $penalty = 1$
        $Option = Options_{KEY}.Next$
        $Displacement_{all} \leftarrow Karamba(S, X_{ground}, m)$
        $MAD = Median(|x_i - \mathrm{Median}(X)|)$
        $Z_{score} = \frac{0.6745 \cdot (x_i - \mathrm{Median}(X))}{MAD}$
        if then $|Z_{score}| > 3.5$
            $penalty = 1.5$
        end if
        $Options^i_{result} = Average(Displacement_{all}) * penalty$
    end while
    $S_{step} = min(Option_{result}).Ind$
    if $S_{step} > threshold$ then $S \leftarrow \varnothing$
        break
    end if
    $X_{ground} = X_{nodes}$ where $X^i_{nodes}.Z < 0.1$
    $Step+ = 1$
end while
```

Fig. 7. Example of PhD student work from SD3 that synthesizes form-finding techniques of RhinoVAULT and machine-learning techniques to develop a method for constructing compression-only vaults without scaffolding (work by Vina Wei).

4. Current developments and future work

The proposed curriculum was fully implemented for the first time in the Spring 2024 and Fall 2024 semesters. While the introduction of radically new course content demonstrated new potential, several challenges remain. One of the primary concerns is ensuring that the skills students acquire translate effectively into professional practice, where standardized workflows and conventional digital tools dominate. The reality of contemporary practice—where efficiency, liability, and conventional methods often take precedence over experimentation—poses a significant challenge to embedding computationally driven structural intuition in real-world design processes and practices. Bridging the gap between academia and industry remains critical for making a lasting impact, emphasizing the need to disseminate this knowledge beyond the university setting.

One of the key areas for future expansion involves incorporating *Augmented Reality* (AR) and *Virtual Reality* (VR) technologies into structural design education. Despite the increasing geometric and data complexity of contemporary architectural design, the interface through which students interact with these models remains largely unchanged, relying on flat screens and mouse-based interactions. Current developments are experimenting with immersive design environments, where AR and VR platforms allow students to manipulate and experience structural systems at full scale, removing the constraints of traditional two-dimensional representations.

Recognizing this potential, the author, in collaboration with colleagues specializing in AR/VR technologies at the University, is developing a new module for SD3, scheduled to launch in the Fall 2025 semester. This

Fig. 8. Example of student work from Master of Architecture integrated design studio, incorporating structural design of a key architectural feature (work by Clara Martucci, Ben Hao, Siddhant Salvi).

module aims to redefine how students engage with spatial structures, using gesture-based interactions, real-time simulations, and immersive feedback loops to explore form-finding and optimization in a more intuitive and embodied manner.

Additionally, the overall Structural Design curriculum is being refined to integrate machine learning and AI-driven optimization techniques into structural exploration. As AI-generated design proposals become more sophisticated, there is an increasing need to train students not only to use these tools but to critically engage with them, understanding their biases, limitations, and creative potential. Future coursework will incorporate adaptive AI-assisted structural design models, allowing students to leverage data-driven insights while maintaining creative control over their architectural visions.

Fig. 9. Example of student work from BArch integrated design studio, where structural design played a key role in the development of the architectural form (work by Chengming Jia, John Ploeger, Henry von Rintelen, David Warfel).

Finally, another critical aspect of ongoing development is enhancing collaborative learning experiences between architecture and structural engineering students. While interdisciplinary collaboration is a central theme of professional practice, it is often introduced too late in architectural education. New initiatives, such as co-taught seminars and cross-disciplinary workshops, will focus on bridging the cultural and methodological divide between architects and engineers, fostering a more integrated approach to design thinking from the outset. By expanding the role of interactive tools, immersive technologies, and AI-driven design methodologies, we can better prepare architecture students for a rapidly evolving professional landscape, where computation is not just a tool for optimization but a medium for creative exploration.

5. Conclusion

The introduction of computational tools into structural design pedagogy represents a fundamental shift in how architecture students engage with structural concepts. By moving beyond traditional methods that rely solely on analytical equations and static calculations, the proposed curriculum fosters interactive, exploratory learning, allowing students to develop structural intuition through real-time interaction and feedback. The integration of computational graphic statics, parametric structural modeling, and form-finding techniques provide students with both the conceptual understanding and technical fluency needed to navigate contemporary architectural practice.

Through a carefully structured sequence of courses—SD1, SD2, and SD3—students gain progressively deeper insights into structural behavior, materialization, and computational analysis. The final integration of these skills in design studios further reinforces their ability to apply structural thinking within comprehensive building design projects. Early results from student work indicate a notable improvement in computational literacy, geometric dexterity, and the ability to make informed design decisions, demonstrating the success of this hands-on, digitally driven approach.

Despite its successes, this curriculum remains a work in progress, with several challenges that must be addressed. A key concern is the translation of these skills into professional practice, where standardized tools and workflows still dominate. Bridging the gap between academia and industry requires disseminating these innovative approaches beyond the university setting, ensuring that computationally-driven structural thinking is recognized and valued in professional practice.

Looking forward, the integration of AR and VR technologies into structural education presents a promising avenue for further innovation. By enhancing spatial perception and immersive interaction with structural systems, these technologies can fundamentally change how students visualize, manipulate, and understand structural design. The development of a new AR/VR module for SD3, set to launch in Fall 2025, represents the next step in this ongoing effort to expand the role of computation in architectural education.

Ultimately, the goal of this evolving pedagogy is to equip the next generation of architects with the tools, methodologies, and mindsets needed to redefine the role of structure in architectural design. By embracing computational tools as a medium for exploration rather than just optimization, we can foster a new era of design innovation—one that transcends disciplinary boundaries and reimagines the intersection of architecture, structural design, and digital technologies.

References

Allen, E., & Zalewski, W. (2010). *Form and forces: Designing efficient, expressive structures.* John Wiley & Sons.

Autodesk. (2025). *Revit* [Computer software]. Autodesk. https://www.autodesk.com/products/revit

Avelino, R. M., Lee, J., Van Mele, T., & Block, P. (2021). An interactive implementation of algebraic graphic statics for geometry-based teaching and design of structures. *Proceedings of the International fib Symposium on the Conceptual Design of Structures 2021*.

Enrique, L., Tanadini, D., Block, P., & Schwartz, J. (2018). Design-oriented approach to teach structures in architecture based on graphic statics. In *Proceedings of the 2018 IASS Symposium*.

Lee, J., Enrique, L., Van Mele, T., & Block, P. (2021). Geometry-based teaching of structures through computational graphic statics. In *IASS2020/21-SURREY7: Proceedings of the Annual Symposium of the International Association for Shell and Spatial Structures (IASS2020) and the 7th Surrey International Conference on Spatial Structures*.

López López, D., Domènech Rodríguez, M., & Guirao Costas, S. (2022). Intuition and experimentation as teaching tools: Physical and interactive computational models. In *INTED2022 Proceedings: 16th International Technology, Education and Development Conference* (pp. 9727-9734). IATED. https://doi.org/10.21125/inted.2022.2552

McNeel, R. (2025). *Rhinoceros 8* [Computer software]. Robert McNeel & Associates. https://www.rhino3d.com/

Preisinger, C. (2013). Linking structure and parametric geometry. *Architectural Design, 83*(4), 110-113.

Rechenraum GmbH. (2016). *GOAT* (Version 3.0) [Computer software]. https://www.rechenraum.com/en/goat.html

Rippmann, M., Lachauer, L., & Block, P. (2012). Interactive vault design. *International Journal of Space Structures, 27*(4), 219-230.

Rutten, D. (2013). Galapagos: On the Logic and Limitations of Generic Solvers. *Architectural Design, 83*, 132-135.

Solís, M., Romero, A., & Galvín, P. (2012). Teaching structural analysis through design, building, and testing. *Journal of Professional Issues in Engineering Education and Practice, 138*(3), 246–253. https://doi.org/10.1061/(ASCE)EI.1943-5541.0000097

Van Mele, T., Lachauer, L., Rippmann, M., & Block, P. (2012). Geometry-based understanding of structures. *Journal of the International Association of Shell and Spatial Structures*, 53(4), 285-295.

Van Mele, T., & Block, P. (2014). Algebraic graph statics. *Computer-Aided Design, 53*, 104-116.

Van Mele, T., & others. (2017–2019). *COMPAS: A framework for computational research in architecture and structures* [Computer software]. https://doi.org/10.5281/zenodo.2594510

Van Mele, T., & Lee, J. (2024). *COMPAS RhinoVAULT: Funicular form finding for Rhinoceros* [Computer software]. Retrieved from https://github.com/BlockResearchGroup/compas-RV