2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #51885

Computational Tools for Dynamics and Controls: Developing Computational Literacy for Engineering Students

Wayne L Chang, University of Illinois at Urbana - Champaign

Dr. Wayne Chang is a teaching assistant professor in the Aerospace Engineering Department at the University of Illinois at Urbana-Champaign. He received his BS, MS, and Ph.D. in Mechanical and Aerospace Engineering from the University of California, Irvine.

Jairo Martin Rojas Huamaní, University of Illinois at Urbana - Champaign

Jairo M. Rojas is a Ph.D. candidate in the Department of Physics at the University of Illinois at Urbana-Champaign. His research focuses on theoretical soft matter, with emphasis on tissue mechanics and active matter.

Mr. Grayson Kenneth Schaer, University of Illinois at Urbana - Champaign

Grayson is a PH.D. candidate in aerospace engineering in the Department of Aerospace Engineering at the University of Illinois at Urbana-Champaign. His research focuses on the intersection of applied controls and autonomous materials manufacturing. He has additional interests in developing Python-based educational tools with an emphasis on the simulation and role-play pedagogy.

Prof. Timothy Bretl, University of Illinois Urbana-Champaign

Timothy Bretl is a Professor of Aerospace Engineering at the University of Illinois at Urbana-Champaign.

Siegfried Eggl, University of Illinois at Urbana-Champaign

Siegfried Eggl (Ph.D. ’13 Univ. of Vienna, Austria) is Assistant Professor in Aerospace Engineering at the University of Illinois at Urbana-Champaign, where he is leading the Astrodynamics and Planetary EXploration (APEX) group at UIUC conducting research in Space Situational Awareness, Celestial Navigation, Planetary Science and Planetary Defense. As part of the NSF-Simons Foundation-funded SkAI institute, he is engaged in AI research across astronomy and aerospace engineering.

Thomas Golecki, University of Illinois at Urbana - Champaign

I spent 10+ years in industry as an engineer in structural mechanics and structural health monitoring projects, earning professional licensure as PE and SE. My PhD research focused on the structural optimization of dynamic systems including random loading and vehicle-bridge interaction. Now as teaching faculty, I try to connect course concepts to real-world examples in a way that motivates and engages students.

Melkior Ornik, University of Illinois Urbana-Champaign

Melkior Ornik (mornik@illinois.edu) is an Assistant Professor with the Dept. of Aerospace Engineering and the Coordinated Science Laboratory, University of Illinois Urbana-Champaign, Urbana, IL, USA. He received a Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 2017. His technical research interests include developing theory and algorithms for control of autonomous systems operating in uncertain, complex, and changing environments, as well as in scenarios where only limited knowledge of the system is available.

Srinivasa M. Salapaka, University of Illinois at Urbana - Champaign

Dr. Srinivasa M. Salapaka is a faculty member in Mechanical Science and Engineering at the University of Illinois Urbana-Champaign. He has extensive experience teaching undergraduate and graduate courses in dynamics, signal processing, and control systems. His work on integrating computational tools into engineering curricula draws on his expertise in numerical methods and systems analysis. He is a recipient of the NSF CAREER award and a Fellow of ASME.

Prof. Matthew West, University of Illinois at Urbana - Champaign

©American Society for Engineering Education, 2026

Matthew West is an Associate Professor in the Department of Mechanical Science and Engineering at the University of Illinois at Urbana-Champaign. Prior to joining Illinois he was on the faculties of the Department of Aeronautics and Astronautics at Stanfo

Sascha Hilgenfeldt, University of Illinois at Urbana - Champaign

Sascha Hilgenfeldt is a professor in the Mechanical Science and Engineering department of the University of Illinois Urbana-Champaign. His technical research centers on Soft Matter, disordered systems, and microfluidics.

©American Society for Engineering Education, 2026

Computational Tools for Dynamics and Control: Developing Computational Literacy for Engineering Students

Abstract

The integration of computational tools into the engineering education of undergraduate students has been a crucial task in improving the depth and variety of students' grasp of the materials, as well as in preparing them better for real-world challenges in their future careers. The Grainger College of Engineering at University of Illinois at Urbana-Champaign has pursued this approach in recent years by adopting a suite of Python-based computational exercises in a variety of sophomore-level math and engineering classes. Here we report on the next step in this educational program: new computational tools were developed to be integrated into junior-level classes, taking four courses in the field of Dynamics and Control as an initial target for the implementation. While Python-based and building on the previously developed computational elements, these new tools are more advanced and challenge students to tackle more sophisticated numerical and programming tasks. At the same time, they encourage open-ended thinking and creativity in the students, as the new tools are flexible enough to allow for deeper exploration of topics. The goal is to foster true computational literacy, so that students are comfortable using a computer for solving problems as a matter of course. In addition to a description of the tools and their implementation in the curriculum, we report extensive survey data concerning student acceptance of and comfort with these changes. We find that students enter the classes with solid Python backgrounds, and students report that their computational skills have improved with the new tools, that they appreciate the importance of computational literacy for their post-graduation careers, and that they have increased confidence in applying their computational skills. These positive trends are particularly strong with aerospace engineering students.

1 Introduction

The use of computational tools is ubiquitous and critical today in multiple engineering disciplines, and it is therefore crucial to ensure that graduating engineering students have acquired as much familiarity with and confidence in the use of such tools as possible during their education, i.e., to make sure they are computationally literate [1, 2]. To start addressing the need, engineering faculty at University of Illinois at Urbana-Champaign have implemented curriculum changes over the past five years, incorporating computational concepts and computational tools into several classes, from introductory computer science to linear algebra and ordinary differential equations, to introductory mechanics classes (in particular sophomore-level statics courses). These course updates occurred over several semesters, based on analysis of data collected from students, such as surveys and course grades. They included changes in curriculum placement and prerequisites/corequisites, in the material taught, and in the teaching methods. An overarching concept of these changes was to base computational elements on a consistent language in order to

ensure as much continuity for the students as possible [3, 4, 5, 6]. The language selected as this lingua franca was Python.

Other studies have demonstrated that computational tools are effective in higher education when properly incorporated into instruction [7, 8, 9], particularly when students can visualize physical responses to changing system parameters via interactive simulations [10]. Mansbach et al. [11] demonstrate that students' average grades increase with additional computational tools. Kononov et al. [12] show that students desire greater implementation of computational tools over one course and the entire curriculum, including all course levels. Zhang et al. [13] show similar results as students prefer computational content starting in their first year of education. These works have shown that students gained confidence using computational tools after taking the reformed courses. Furthermore, surveys by Lee et al. [14] found that graduating students are not only more comfortable using computational tools to solve materials science engineering problems but also that more than half of the surveyed students believed that computation had provided them with a better understanding of the course content.

In the present paper, we report on implementing computational tools in junior-level courses in the mechanical engineering and aerospace departments of the same major Midwestern university, i.e., classes downstream of those targeted by the previous efforts. We selected four 300-level classes that cover topics from Dynamics, Signal Processing, and Control (two each from the mechanical engineering and aerospace departments). The vast majority of students taking these classes have previously taken the introductory computer science, mathematics (linear algebra and ODEs), and Statics classes that were modified in the previous efforts. Therefore, our approach is well-suited for pursuing two main goals: (i) To assess the success of establishing the sophomore-level background of computational skills in upstream classes, and (ii) To implement new computational tools in the junior-level classes that are more advanced and more sophisticated than at sophomore level, and whose design is specifically meant to further boost student comfort, fluency, and creativity in the use of computation. Thus, our efforts are particularly focused on furnishing students with tools providing real-time, interactive experiences that are visually appealing without compromising the accuracy of the underlying modeling and computation. Furthermore, the projects realized through these tools are flexible enough to enable creative exploration of existing projects by students, as well as to allow for the simple development of new projects by instructors.

In the following, we briefly introduce and illustrate the two different toolkits developed, as well as their use in classes. We then describe the methods of assessing student feedback from surveys held over the course of five semesters, and analyze the results in the light of the aforementioned major goals.

2 Approaches to Computational Tools in Junior-level Courses

The present study focuses on two classes in mechanical engineering (ME 340 and ME 360 in the curriculum of UIUC, which for brevity we will call ME1 and ME2 here) and two in aerospace engineering (AE 352 and AE 353, henceforth AE1 and AE2). In the former case, ME1 is an explicit prerequisite of ME2, and the two classes are almost always taken in subsequent semesters by students. In the latter case, there is no explicit prerequisite requirement, but again students take

the AE classes most typically in subsequent semesters, with the sequence AE1 – AE2 being the most common. All four classes are offered every semester, and our study covers the semesters between Spring 2024 and Fall 2025 including. Average enrollment per semester was 146 (ME1), 151 (ME2), 86 (AE1), and 82 (AE2). Engineering students attending all four of these courses have completed very similar prerequisite classes, in particular mandatory introductory computer science, introductory statics, and introductory dynamics, in addition to a calculus sequence and mathematics classes in linear algebra and ordinary differential equations.

Recognizing the documented desire for interactive and visually appealing computational tools [15, 16], it was natural to choose these four classes, which all concern dynamical processes: motion and dynamics catch students' attention and make real-time interaction very rewarding. ME1 concerns modeling and analysis of dynamical systems, while ME2 covers fundamentals of signal processing and control. AE1 similarly treats dynamical systems, while AE2 explicitly concerns control systems. A significant difference is that AE1/AE2 are explicitly meant to present this material in an aerospace engineering context, while ME1/ME2 aim for general applications. Furthermore, both ME classes contain a laboratory component with analysis of using MATLAB and Simulink, an established tool use for limited purposes that has been deemed successful and was not to be compromised. The newly developed computational tools provide, in contrast, much more versatility, interactivity, and general applicability.

Therefore, two quite different sets of computational tools were developed (though both are ultimately Python-based). In ME1/ME2, flexible Jupyter notebooks were designed including elements of solving equations, visualizing solutions, and interactive changing of parameters. For AE1/AE2, a separate sophisticated simulation and visualization tool was developed that allows for the computational discussion of aerospace-related projects in a way that is strongly appealing visually and may be extended into gamification of instruction.

2.1 Python tools for ME1/ME2/AE1

The Jupyter notebooks for ME1/ME2 were placed in the Colab environment [17], allowing students to exchange and collaborate on contributions. Within these notebooks, simple programming tasks (rather than just running code) were demanded of the students, emphasizing a more advanced level of computational literacy than was implemented in upstream classes [3, 4, 5], while using tools that were developed for those earlier courses as a template, so that students recognize the similarities and are more confident in their use of the tools from the start. Figure 1 shows an example screen of a pendulum dynamics toolkit including simultaneous animation of the pendulum motion, time series plotting, and phase plotting.

Such exercises were used in class and in homework exercises. The tools are very transparent and encourage not only exploration of different parameter values, but hands-on programming changes by students. Instructors can easily copy and modify modules for solution of dynamical systems, visualizations, etc., in order to add to a growing library of problems and projects. Variants of these Jupyter notebooks were also used in AE1 where thematic overlap suggested this (e.g. tuned mass damper, double pendulum, coupled oscillators).

In addition to the explicit Python codes, we have found it important to generate executable versions that run in real time and allow for real-time changes of parameters via GUI sliders or

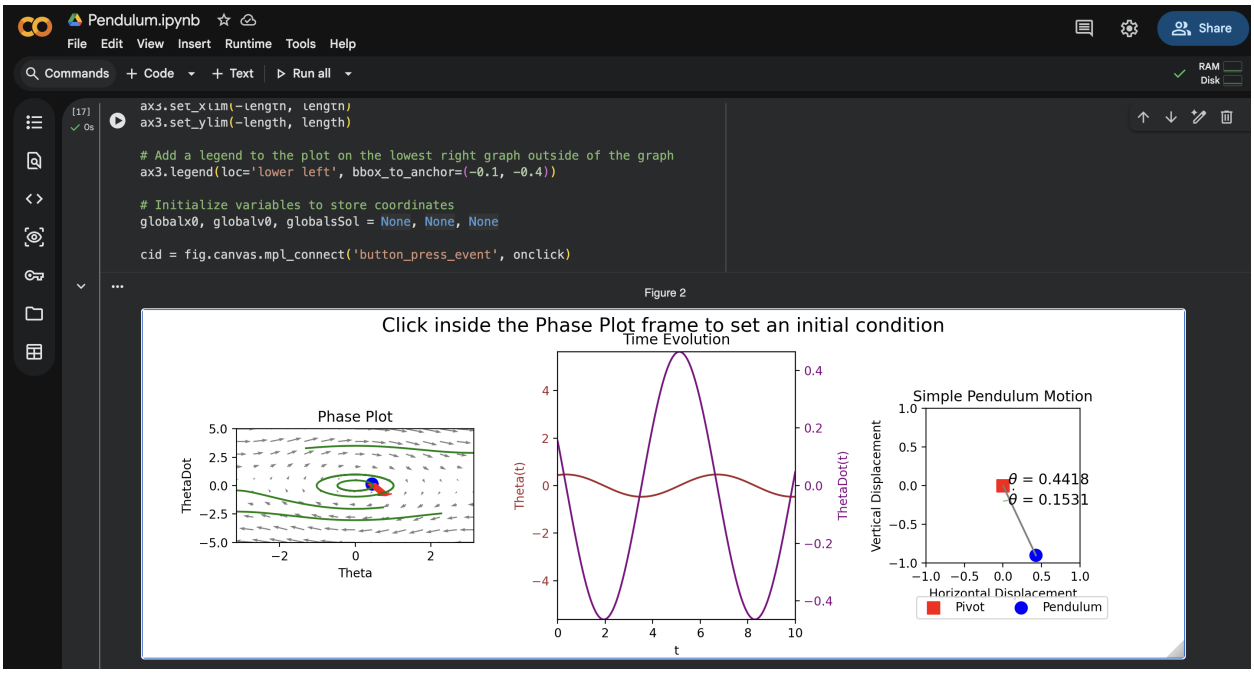

Figure 1: Jupyter notebook environment (with partial Python code visible) for students in the ME1 class to run and modify, with simultaneous visualization of pendulum motion, time series, and phase plot.

similar means. Literature suggests that such interactivity with immediate feedback significantly increases appeal and acceptance of computational tools by students [15, 18]. These executable versions had not, however, been implemented yet in the semesters whose instruction is covered in the present work.

2.2 Simulation and Visualization Environment for AE1/AE2

In the AE classes, particularly the controls class AE2, the aim was to present the students with impressive visuals in a computational tool that establish a very direct connection with specific class projects that are fundamental to aerospace engineering instruction, such as quadcopter control. To this end, we developed a unique platform for both physics-based simulation and sophisticated visualization. This program, dubbed condynsate for **Co**ntrol and **dyn**amics simul**at**or, is described in more detail in Section 3.2 below. Here, we emphasize its complementary character to the Python tools in the ME1/ME2 classes: condynsate fills a need for a fully realized, public-facing tool with extensive documentation that encourages real-time exploration of realistic dynamics of articulated bodies. As such, it is also immediately applicable to controls problems that are central to aerospace engineering education. The emphasis of its implementation has so far been placed on control-based problems in the AE2 class.

3 Description of Computational Tools and their In-class Use

3.1 Colab notebooks

In ME1 and ME2, computational tools were integrated as instructional supports to enhance student learning in dynamics and control. Rather than serving primarily as numerical solvers, these tools were designed to support conceptual understanding, promote active learning, and help students connect mathematical representations to physical system behavior. Their use in both courses was guided by two primary instructional objectives: (1) supporting concept development through visualization and interactive exploration, and (2) reinforcing learning through structured computational exercises used for practice and assessment. The first instructional role of computation focused on visualization as a means of reducing abstraction in traditionally challenging topics. Dynamics and control concepts often require students to reason across multiple representations, such as physical motion, time-domain response, frequency behavior, and phase-space dynamics. Students frequently struggle to integrate these perspectives when they are presented separately through equations or static figures. Interactive computational visualizations allowed these representations to be viewed simultaneously and manipulated in real time, helping students develop a more coherent mental model of system behavior. This approach aligns with established educational practices that emphasize visual learning and interactive engagement as tools for improving conceptual understanding. In ME1, for instance, these visualization tools provided early exposure to system dynamics and stability concepts. By linking phase portraits, time-response plots, and physical animations within a single computational environment, students were able to directly observe how abstract representations relate to real system motion (see Fig. 1). This integrated view aimed at deeper understanding of dynamic behavior and provided a foundation for later topics that rely on interpreting system response qualitatively, rather than solely through formal mathematical analysis. Many of the tools developed in ME1 were also directly used in AE1, as there is strong overlap in the teaching of dynamics concepts between ME1 and AE1. A similar instructional philosophy guided the use of computational tools in ME2, where signal processing concepts are often perceived by students as highly abstract. For example, interactive visualizations that combined time-domain signals, frequency-domain representations, and reconstructed outputs allowed students to explore the consequences of modeling and sampling choices. By enabling students to immediately observe and interpret the effects of these choices, the tools aimed at supporting intuitive understanding of core ideas such as signal representation and distortion (cf. Fig. 2). The inclusion of auditory feedback further reinforced learning by connecting mathematical concepts to perceptible outcomes. The second instructional role of computation involved structured exercises designed to reinforce learning through exploration and analysis. These exercises encouraged students to investigate how changes in system parameters affect behavior, promoting deeper engagement with underlying concepts. Rather than focusing exclusively on obtaining numerical answers, students were asked to interpret results, identify trends, and explain system responses. This approach helped shift emphasis from procedural problem-solving to conceptual reasoning, while simultaneously developing students' computational proficiency. Across both ME1 and ME2, these computational exercises also served as tools for formative and summative assessment. By requiring students to analyze simulation results and justify conclusions, instructors were able to evaluate both conceptual understanding and the ability to apply computational methods. This dual emphasis supports the development of

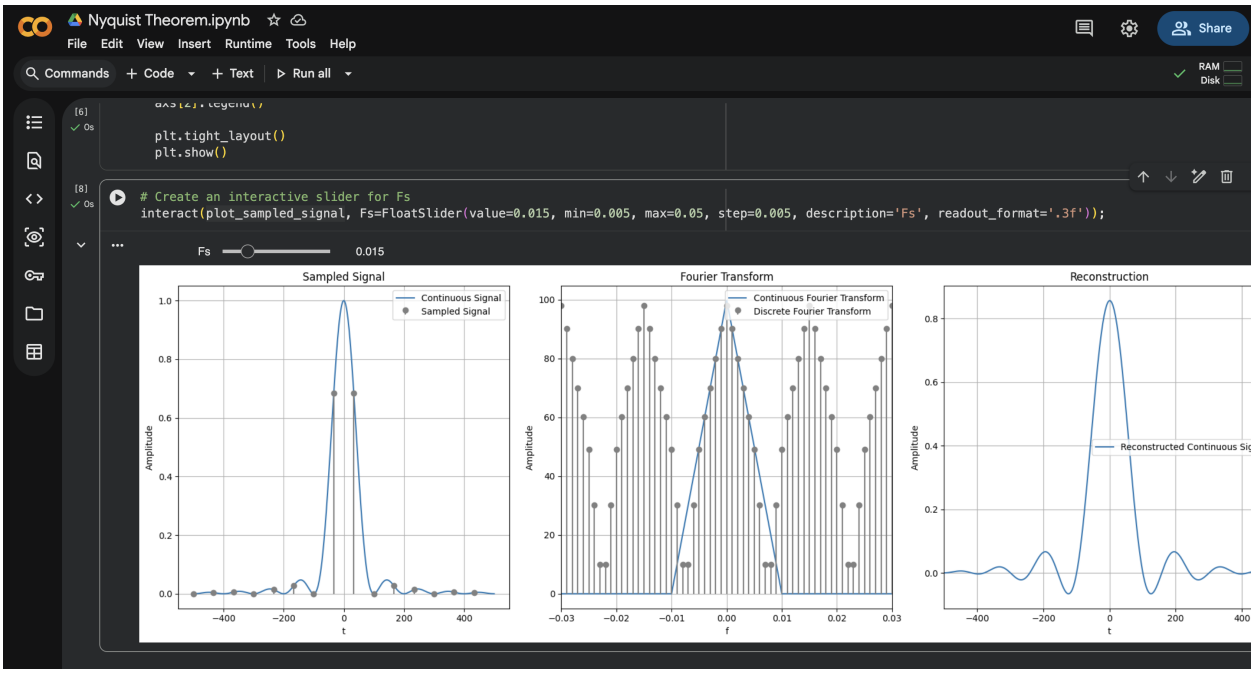

Figure 2: Jupyter notebook environment on signal processing (with partial Python code visible) for students in the ME2 class to run and modify, with simultaneous visualization of sampled signal, Fourier transform, and reconstructed signal. Students can adjust the sampling frequency with a slider.

computational literacy as an integrated engineering skill, rather than an isolated programming requirement. Overall, the instructional use of computational tools in ME1 and ME2 targeted enhanced student engagement, increased conceptual understanding, and promotion of active learning. By embedding computation directly into teaching and assessment, these courses encouraged students to view computational tools as integral to reasoning about dynamic and control systems. This approach aligns with broader educational goals of preparing engineering students to effectively use computation as part of their analytical and problem-solving toolkit.

3.2 Condynsate

For the AE1 and particularly the AE2 classes, students and faculty at major Midwestern university developed the tool condynsate in order to provide real-time simulation and visualization of articulated bodies with nonlinear dynamics based on Python. Built on PyBullet [19] as a simulation engine and MeshCat [20] as a visualization tool, condynsate implements real-time simulation of rigid bodies as well as articulated bodies with a browser-based 3D viewer to visualize simulations, a built-in animator to plot arbitrary simulation states, and a keyboard module which allows detection of key press events. These choices ensure condynsate has a broad scope of applicability, being capable of representing any dynamic system by a .urdf file [21]. Details about condynsate are available in a separate publication [22], and it is available, together with a documentation, tutorial, and application examples, on GitHub [23].

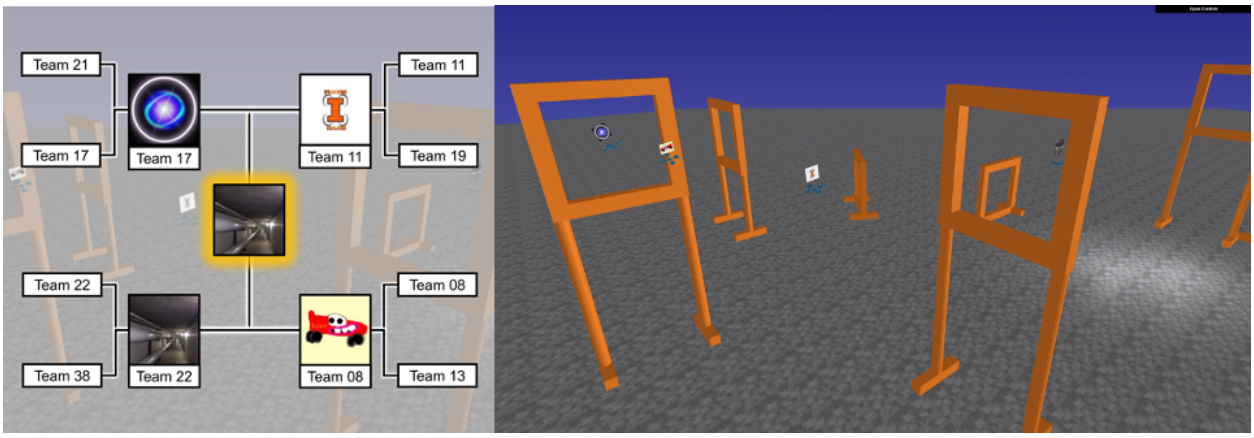

Figure 3: Final student project in the AE2 course using condynsate to control quadcopter drone simulation models and race them in teams. Left: team competition bracket; right: obstacle course for drones.

In the present publication, we briefly describe the use of condynsate in the dynamics and control classes of our concern. The majority of this use occurred in the AE2 (control systems) class, where students were introduced to and worked with a specific sequence of projects implemented via condynsate. Students first worked on a PD torque controller for a wheel on an axle, as an introductory problem to familiarize them with the tool. A second example project featuring a wheeled cart was revisited throughout the course to introduce new concepts (e.g. autonomous balance of an inverted pendulum).

Class projects were given to two-student teams and each involved an instructor-programmed PyScript based backend and a student-facing Jupyter notebook frontend, with basic equations of motion given. Students thus had the opportunity for flexible Python programming to tackle the tasks. Such projects included 2-axis control of a gyroscope, control of a robotic self-balancing Segway-type vehicle, and orientation control of spacecraft. The course culminated in a project on quad rotor control capped by a competitive drone race in which student teams competed for best performance. Figure 3 demonstrates the visualization and student engagement related to this project, including the students' design of team logos.

Over the course of this project (Spring 2024 through Fall 2025) structural improvements were made in the programming and running of condynsate that increased running speed to the point that real-time interaction (no-delay changes of displayed dynamics upon interactive changing of parameters) became possible.

Note also that the implementation of enhanced visuals (e.g. textures and shadows) contributes to a realistic appearance that pulls the student into the simulated system as a realistically modelled phenomenon that can be intuitively grasped. As demonstrated with the drone race project illustrated in Fig. 3, this facilitates game-scenario approaches to solving problems with the aim of keeping student interest and enthusiasm high. In the future, we will further pursue the potential of condynsate to promote educational goals in a gamified environment while maintaining the accuracy of a robust physics-based simulation engine.

4 Methods for assessing student feedback

4.1 Student Survey Design

Anonymous end-of-semester surveys were administered online to students enrolled in all four courses in most of the four semesters of Spring 2024, Fall 2024, Spring 2025, and Fall 2025. Not all instructors participated in these surveys, so that not all semesters are present for every course. The percentage of students returning filled-in surveys also varied between semesters, as different instructors placed different degrees of emphasis on encouraging students to reply. Most semesters of the ME1 and ME2 classes contain data from two parallel sections, as these classes are large enough to be split into two sections each semester. AE1 and AE2, by contrast, are single-section courses.

Each questionnaire contains questions belonging to four distinct groups: (i) questions concerning upstream class prerequisites and previous knowledge; (ii) questions concerning the synergy of computational tools with the material taught in the surveyed class; (iii) questions probing differentiation between the use of computational tools described in the present paper from computational tools encountered in previous classes; (iv) questions concerning the value of computational education for downstream classes and further career of the students.

Between the four classes ME1/ME2/AE1/AE2, variations were kept to a necessary minimum: in (i), prerequisites are slightly different between courses; in (ii), specific questions about class content vary; in (iii), the different characters of the ME and AE tools necessitate different formulations.

In selected classes and semesters, in order to obtain a more fine-grained and immediate picture of student reaction to a semester's class material, relevant survey questions were administered both in the beginning of a semester ("pre-semester survey") and at the end of the same semester ("post-semester survey"). We here present only data from the post-semester surveys available in all classes and semesters; we defer a comparison of pre-semester to post-semester surveys to later work.

4.2 Survey Questions and Answer Key

Survey data are the results of self-reported student replies. While some questions, particularly those concerning whether particular upstream classes were taken, have binary yes/no answers, most questions ask for students assessment of statements that can be agreed or disagreed with. Answers to most of these survey questions are presented on a 5-point Likert scale using the following options and corresponding numeric values: "Strongly disagree" (1), "Disagree" (2), "Neither agree nor disagree" (3), "Agree" (4), "Strongly agree" (5). Where we deviate from this scheme (answers to Q1.1 and Q1.2 below), the explicit answer options are given.

5 Results

In our study, we relied on self-reported survey data from students in the four classes between Spring 2024 and Fall 2025. Surveys were not held in all semesters in all classes, and response rates varied. The surveys contained a number of questions about course-specific prerequisites and

course-specific teaching outcomes, which we will not differentiate here, as our focus is on the overall acceptance and success of computational tools. Here we thus report on those questions that are of equal relevance across all courses, structured along the four themes mentioned above.

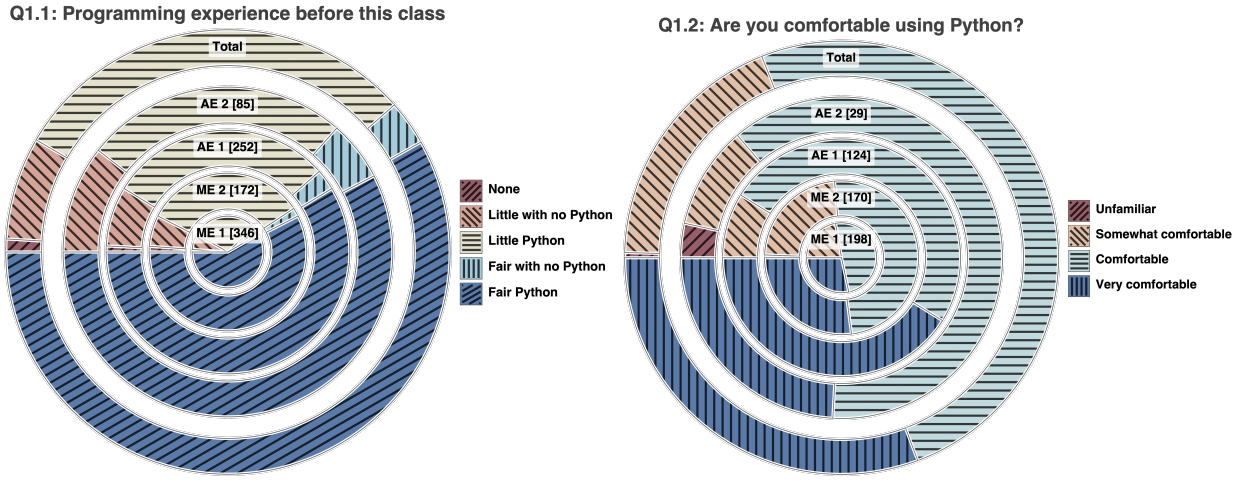

Figure 4: Student survey responses to questions Q1.1 (previous experience with Python) and Q1.2 (comfort with using Python). Familiarity and comfort are consistently high across all four courses. Numbers on each ring are total number of student responses across four semesters in each class.

5.1 Incoming students' knowledge of and comfort with Python

Being a continuation of a broader effort towards computational literacy at major Midwestern university, one aim of our project was to assess whether students who had completed computationally enhanced classes up to sophomore level entered the courses of specific interest with unified backgrounds and knowledge as intended by those broader efforts. In particular, Python as a *lingua franca* should have been established, so that students should have a significant level of comfort with the language entering ME1/ME2/AE1/AE2. We asked the following questions: Q1.1 "Describe your programming experience before this class" – where the answer options were explicitly asking for experience with Python or with other languages; and Q1.2, "How comfortable are you with Python as a programming environment?". The results in Fig. 4 show that students have – as intended – become familiar with Python in upstream classes in their overwhelming majority. Figure 4, as well as the following figures, presents added (summed) data from all survey semesters for the four classes in the inner four rings (numbers in brackets are numbers of survey answers in each course), as well as an overall total for all four classes in the outer ring. Students report familiarity with Python or basic experience with Python (Q1.1) consistently between 84% (AE2) and 91% (ME2). Similarly, the percentage of students comfortable or very comfortable with Python (Q1.2) is consistently high between 76% (ME2) and 91% (AE1). The small percentage of students not entirely comfortable with Python was addressed by offering optional tutorials and training modules.

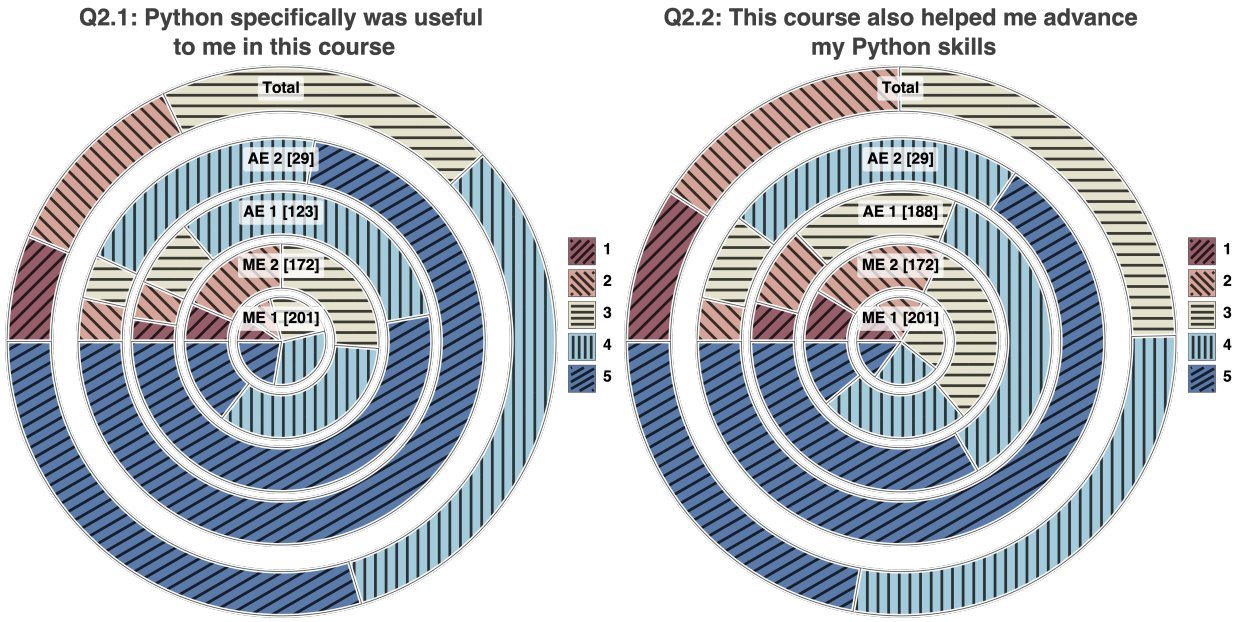

Figure 5: Student survey responses to questions Q2.1 (Python was helpful for this class) and Q2.2 (This class also helped my Python skills). Answers are on a Likert scale of 1-"Strongly Disagree", 2-"Disagree", 3-"Neutral", 4-"Agree", 5-"Strongly Agree".

5.2 Student evaluation of computational tools in current classes

To further tie the programming language Python to the classes of our current interest, we asked two complementary questions (as mentioned before, these are questions in post-semester surveys): Q2.1 "Python specifically was useful to me in this course" and Q2.2 "This course also helped me advance my Python skills". Averaged over all classes, the former question was more positively answered than the latter, perhaps unsurprisingly as none of the classes explicitly contained computation skills in its course material. However, a particularly strong positive response to both questions was registered for AE2 (the class relying on the condynsate tool) and to a lesser extent for AE1 (in which students were sporadically exposed to condynsate). The greater level of sophistication in the presentation and visualization of condynsate (compared with the Colab Python tools) translated into a greater level of student recognition of the program as an advanced tool.

5.3 Student assessment of computational tool sophistication

Continuing this line of questioning, we asked about the students awareness of the degree to which computational tools were used in these classes (Q3.1 "I apply computational tools (as opposed to pen-and-paper) more widely than in earlier classes") and whether the students consider the current computational tools as more advanced (Q3.2 "I feel that I am able to tackle more advanced computational/programming problems than in earlier classes"). Figure 6 gives a picture consistent with that of the previous section: While the overall student response is positive in all classes, students agree significantly more with these statements in the AE classes with exposure to condynsate: Responses of "agree" and "strongly agree" to Q3.1 comprise 69% (AE1) and 86%

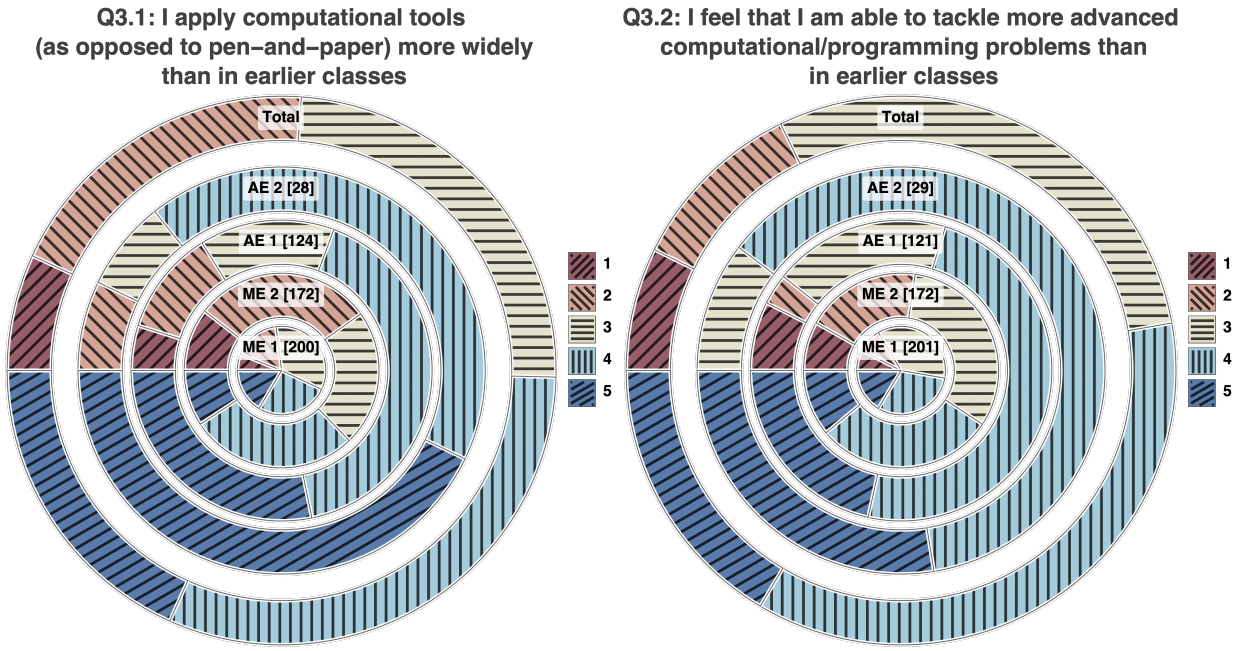

Figure 6: Student survey responses to questions Q3.1 (degree of application of computational skills) and Q3.2 (sophistication of applied computational skills). Answers are on a Likert scale of 1-"Strongly Disagree", 2-"Disagree", 3-"Neutral", 4-"Agree", 5-"Strongly Agree".

(AE2) of answers, to Q3.2 the figures are 70% (AE1) and 90% (AE2), respectively. This is another indication that the condynsate tool is seen as a more advanced and wider application of programming and computation. In addition, student and instructor feedback suggests that aerospace engineering students are aware of the central role of computational tools for their specific careers, and thus appreciate the need for familiarity with advanced tools.

5.4 Student valuation of computational tools for academic and professional careers

Having established that on average students are aware of a progression of their computational knowledge and abilities having followed the Mechanical Engineering and Aerospace Engineering curricula at the major Midwestern university to the junior level, we ask students if this translates into a positive attitude towards future applications of computation. In particular, our surveys contained the questions Q4.1 "I think computational/programming skills are useful for other classes" and Q4.2 "I think computational/programming skills are useful for my career." As shown in Fig. 7, the responses were overwhelmingly positive and support expectations that students will remain continuously engaged and comfortable with the use of computational tools.

5.5 Focus group feedback from Teaching Assistants of AE2

In addition to student surveys, this project held a focus group discussion to obtain feedback from both undergraduate student and graduate student teaching assistants of AE2. The undergraduate teaching assistants had taken this class in a previous semester and were thus familiar with the use of tools (particularly condynsate) as students taking the class as well as in the role of instructors.

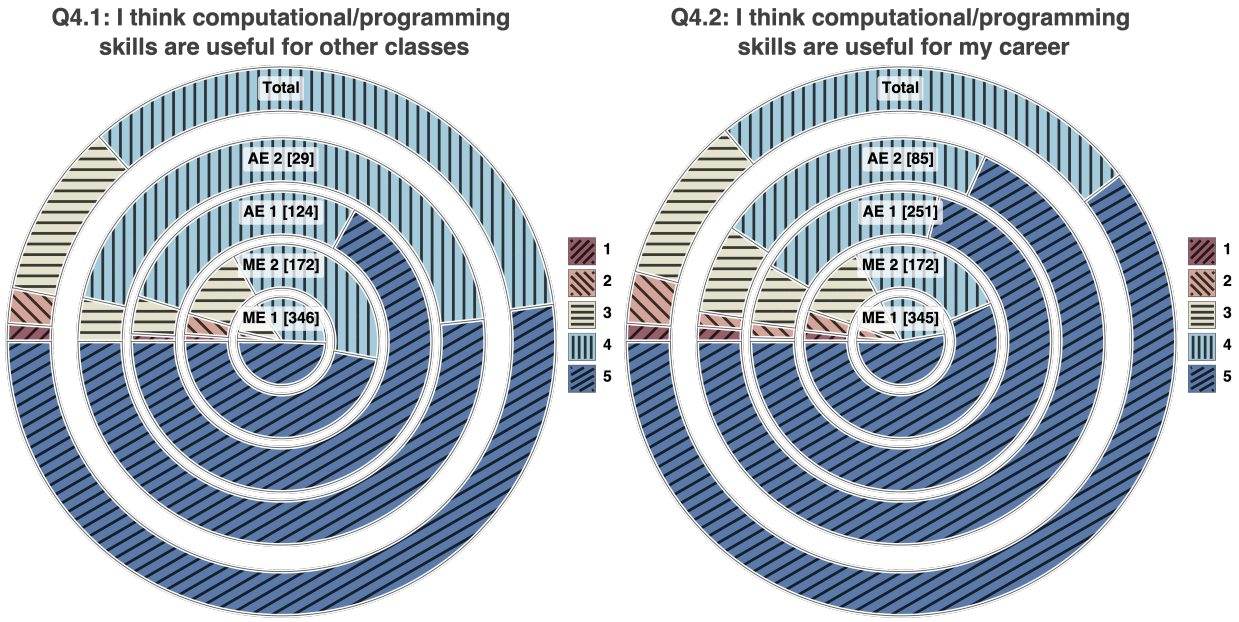

Figure 7: Student survey responses to questions Q4.1 (usefulness of computational skills for other classes) and Q4.2 (usefulness of computational skills for the students' career). Answers are on a Likert scale of 1-"Strongly Disagree", 2-"Disagree", 3-"Neutral", 4-"Agree", 5-"Strongly Agree".

The teaching assistants emphasized the utility of condynsate for improving students' understanding of physical processes as well as its benefits as an interactive visualization tool (sample comments: "a tool that lowers the barrier of entry if you want to [...] create your own dynamical system, simulate it, visualize it"; "you can effectively see the changes as you make them"). Teaching assistant feedback on installation and running of condynsate prompted us to push for a platform- and hardware-independent version of condynsate, which is described in a separate publication [**?**].

6 Conclusions

Computational literacy is recognized as an important skill in most aspects of STEM education, and in the education of engineering students in particular. The activities reported in the present work are the junior-level continuation of changes that had been implemented in freshman- and junior-level classes as part of a long-term overhaul of the engineering curriculum at a major Midwestern university, with the goal of consistently exposing students to computational elements in a sequence of classes. Our findings underscore the positive and significant impact on students' learning experiences resulting from such a strategy. Recognizing that the junior-level classes provide an opportunity to instill greater confidence in the students concerning the depth and spectrum of their computational abilities, we decided to present more advanced tools to the students, with more flexibility and more leeway for free exploration. Flexible modules of Colab Python notebooks were used in the two mechanical engineering classes ME1 and ME2 and (partially) in the aerospace engineering dynamics class AE1. In AE2, and particularly again in AE1, a further level of sophistication was reached with projects based on the simulation and

visualization engine condynsate.

Through the results of our systematic student surveys, we first established that upstream classes had established a common baseline of familiarity with and relative comfort with Python as a programming language. Answers were strongly affirmative in all classes. Students also acknowledged that Python computational tools were helpful for the understanding of the classes, with the AE classes showing a higher level of agreement with this statement. The converse statement (the course helped develop Python skills further) showed this differential more clearly, with the condynsate-exposed students in AE1 and AE2 being significantly more positive than the students in ME1 and ME2.

The trend of students' more positive perception of condynsate continues in the answers to questions about wider use of computational tools and use of more advanced computational skills as compared to upstream classes. While students in ME1 and ME2 are nearly neutral about these statements, the AE students evaluate them positively, and strongly positively in AE2 with its pronounced use of condynsate. Finally, all students overwhelmingly agree that the use of computational and programming tools will be helpful in both downstream classes and in the students' broader career.

In summary, these results show robust acceptance of and identification with the efforts aimed at computational literacy among the student body. They also suggest that in more advanced classes students appreciate more sophisticated tools that are visually appealing and allow for flexible, creative interaction. Incorporating both physically correct simulations and appealing visualizations together appears to have the greatest positive effect. Such tools allow for fun, gamified projects and exercises, particularly in the context of dynamics and control systems, where animation and motion are natural elements. We hope to expand the use of our tools further and adapt them to individual classes further downstream the curriculum, e.g., in robotics courses.

References

[1] S. Kim, S. Han, and H. Kim, "How can we teach computational literacy to all levels of students?" in *2009 Fifth International Joint Conference on INC, IMS and IDC*. IEEE, 2009, pp. 1395–1400.

[2] D. Braun and J. Huwer, "Computational literacy in science education,Äìa systematic review," *Frontiers in Education*, vol. 7, Aug 2022.

[3] M. Silva, P. Hieronymi, M. West, N. Nytko, A. Deshpande, J.-C. Chuang, and S. Hilgenfeldt, "Innovating and modernizing a linear algebra class through teaching computational skills," in *2022 ASEE Annual Conference & Exposition*, 2022.

[4] H. Chen, M. West, S. Hilgenfeldt, and M. Silva, "Measuring the impact of a computational linear algebra course on students' exam performance in a subsequent numerical methods course," in *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 2023, pp. 115–121.

[5] W. Chang, S. W. Ok, M. West, S. Hilgenfeldt, and M. Silva, "Effects of integrating computational tools into an introductory engineering mechanics course," in *2024 ASEE Annual Conference & Exposition*, 2024.

[6] W. L. Chang, M. R. Hoyle, M. D. Opolz, J.-C. Raymond-Bertrand, N. C. Admal, T. Golecki, K. M. Halloran, S. Hutchens, C. Luetkemeyer, B. Mercer *et al.*, "Reflecting on ten years of building a community of practice for teaching innovations in fundamental mechanics courses," in *2025 ASEE Annual Conference & Exposition*, 2025.

[7] G. M. Lu, D. R. Trinkle, A. Schleife, C. Leal, J. Krogstad, R. Maass, P. Bellon, P. Y. Huang, N. H. Perry, M. West, T. Bretl, and G. L. Herman, "Impact of integrating computation into undergraduate curriculum: New modules and long-term trends," in *2020 ASEE Virtual Annual Conference*, 2020.

[8] A. D. Santo, J. C. Farah, M. L. Martinez, A. Moro, K. Bergram, A. K. Purohit, P. Felber, D. Gillet, and A. Holzer, "Promoting computational thinking skills in non-computer-science students: Gamifying computational notebooks to increase student engagement," *IEEE Transactions on Learning Technologies*, vol. 15, no. 3, p. 392–405, 2022.

[9] A. Ciobanu, C. Miron, C. Berlic, and V. Barna, "Integrating computational tools in teaching electromagnetic oscillations," *Romanian Reports in Physics*, vol. 75, no. 4, 2023.

[10] Y. Rodríguez, A. Santana, and L. M. Mendoza, "Physics education through computational tools: the case of geometrical and physical optics," *Physics Education*, vol. 48, no. 5, pp. 621–628, 2013.

[11] R. Mansbach, A. Ferguson, K. Kilian, J. Krogstad, C. Leal, A. Schleife, D. R. Trinkle, M. West, and G. L. Herman, "Reforming an undergraduate materials science curriculum with computational modules," *Journal of Materials Education*, vol. 38, p. 161–174, 2019.

[12] A. Kononov, P. Bellon, T. Bretl, A. L. Ferguson, G. L. Herman, K. A. Kilian, J. A. Krogstad, C. Leal, R. Maass, A. Schleife, J. K. Shang, D. R. Trinkle, and M. West, "Computational curriculum for MatSE undergraduates," in *2017 ASEE Annual Conference & Exposition*, 2017.

[13] X. Zhang, A. Schleife, A. Ferguson, P. Bellon, T. Bretl, G. L. Herman, J. A. Krogstad, R. Maass, C. Leal, D. R. Trinkle, and M. West, "Computational curriculum for MatSE undergraduates and the influence on senior classes," in *2018 ASEE Annual Conference & Exposition*, 2018.

[14] C.-W. Lee, A. Schleife, D. R. Trinkle, J. A. Krogstad, R. Maass, P. Bellon, J. K. Shang, C. Leal, M. West, T. Bretl, G. L. Herman, and S. Tang, "Impact of computational curricular reform on non-participating undergraduate courses: Student and faculty perspective," in *2019 ASEE Annual Conference & Exposition*, 2019.

[15] P. Menghal and A. J. Laxmi, "Real time simulation: A novel approach in engineering education," in *2011 3rd International Conference on Electronics Computer Technology*, vol. 1. IEEE, 2011, pp. 215–219.

[16] D. Makuteniene, R. Kliukas, A. Čeponis, O. Ovtšarenko, and H. Suzuki, "Integrating graphical methods into engineering education: Enhancing learning and visualization skills in university curricula," in *International Conference on Geometry and Graphics*. Springer, 2024, pp. 215–222.

[17] E. Bisong, "Google colaboratory," in *Building machine learning and deep learning models on google cloud platform: a comprehensive guide for beginners*. Springer, 2019, pp. 59–64.

[18] D. Alique and M. Linares, "The importance of rapid and meaningful feedback on computer-aided graphic expression learning," *Education for Chemical Engineers*, vol. 27, pp. 54–60, 2019.

[19] E. Coumans, "Pybullet." [Online]. Available: https://pybullet.org

[20] J. Nimmer, R. Deits, and R. Tedrake, "meshcat-python." [Online]. Available: https://github.com/meshcat-dev/meshcat-python

[21] "Open robotics robot operating system." [Online]. Available: https://www.ros.org/

[22] G. Schaer, W. Chang, S. Eggl, T. Bretl, and S. Hilgenfeldt, "condynsate: A python-based controls and dynamics simulation package," under review (2026).

[23] G. Schaer, "condynsate," GitHub repository, https://github.com/condynsate/condynsate, 2026.