Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025
Paper ID #45635

Bridging Theory and Practice: Integrating Python Programming in Introductory Power Systems Analysis

Dr. Robert J Kerestes, University of Pittsburgh

Robert Kerestes, PhD, is an Associate Professor of Electrical and Computer Engineering at the University of Pittsburgh's Swanson School of Engineering. A native of the Mount Washington neighborhood in Pittsburgh, Pennsylvania, Bob earned his B.S. (2010), M.S. (2012), and PhD (2014) from the University of Pittsburgh, with a concentration in electric power systems.

Bob's research interests include modeling and analyzing electric power distribution systems, intelligent grid technologies, distribution systems dynamics analysis, co-simulation of electric power systems, and integrating distributed energy resources and electric vehicles. He is dedicated to advancing evidence-based teaching practices in power systems education. He currently serves as the Chair of the IEEE Test Feeder Working Group, where he helps develop standardized models for electric distribution system analysis.

In addition to his work with the IEEE Test Feeder Working Group, Bob is an active member of the IEEE PES Analytic Methods for Power Systems (AMPS) Technical Committee and the IEEE PES Distribution Systems Analysis Subcommittee of AMPS.

Bob co-authored the Fifth Edition of Distribution System Modeling and Analysis with Bill Kersting and is leading the efforts for the Sixth Edition. Before joining academia, Bob worked as a mathematical modeler for Emerson Process Management, focusing on electric power applications for Emerson's Ovation Embedded Simulator.

He also served in the United States Navy as an Interior Communications Electrician from 1998 to 2002 on active duty and from 2002 to 2006 in the U.S. Naval Reserves.

Jack Thomas Carnovale, University of Pittsburgh

Jack Carnovale is a Master's Student studying Electrical Engineering at the University of Pittsburgh. He has received a Bachelor's degree in Electrical Engineering from the same institution (2023). His research interests include fault response of DER in electric power distribution systems, motor control, and engineering education. To stay involved with engineering education, Jack has served as a teaching assistant for a variety of classes in his university's ECE department including lecture and lab based classes for power system analysis, mechatronics and motor control, microelectronic circuits, and electronic prototype design.

Jack will defend his Master's Thesis (May 2025) and begin work with Eaton as a Power Electronics LDP.

Paulo Radatz

Bridging Theory and Practice: Integrating Python Programming in Introductory Power Systems Analysis

Abstract

This paper explores the integration of Python programming into an introductory course in power systems analysis, highlighting its critical role in enhancing students' technical and problem-solving skills. By introducing Python early, the course provides students with a solid foundation in essential programming concepts such as functions and basic problem-solving techniques. This foundation equips students with the tools to confidently approach complex power systems analysis problems, preparing them for more advanced coursework. Students build upon this base in later courses, expanding into object-oriented programming (OOP) to further develop more scalable and sophisticated Python programs.

The curriculum bridges theoretical learning with practical application by coupling Python with PowerWorld, a commercial power systems analysis tool. This dual approach allows students to engage in hands-on problem-solving using both custom Python code and industry-standard software, enhancing their understanding of power systems analysis in a real-world context. By streamlining the problem-solving process, this integration fosters deeper comprehension and makes learning more relevant to their future careers.

By the end of the course, students will have gained practical experience, improved computational skills, and the confidence needed to address real-world challenges in power systems engineering. This early introduction to Python, paired with the development of advanced programming skills, positions students for success in both academic and professional environments.

1 Introduction

The growing complexity of power systems in today's energy landscape demands engineers well-versed in theoretical knowledge and practical problem-solving. In response, engineering education must evolve to equip students with the tools and confidence to tackle real-world challenges. This paper explores a curriculum innovation aimed at enhancing students' technical and computational skills through the integration of Python programming into an introductory power systems analysis course.

Introducing Python programming early in the curriculum gives students a solid foundation in essential programming concepts, such as functions and basic problem-solving techniques. While object-oriented programming (OOP) is not directly covered in this introductory course, the knowledge gained here serves as a critical stepping stone, motivating the use of OOP in a subsequent course on the computer analysis of power systems. This approach ensures that students are well-prepared to tackle more advanced problems and develop scalable solutions in their later coursework.

Throughout the course, students complete three hands-on coding assignments using Python to analyze power systems, culminating in a final power flow project that integrates the skills they develop. The final project couples Python with the commercial power systems simulator, PowerWorld, providing students with a comprehensive, real-world application of the concepts they learn. In addition, students take a practicum exam where Python serves as a vital component of the assessment. While generative AI tools are not required or explicitly taught for the Python programming assignments, students are encouraged to use them for validation, debugging, and improving code efficiency. Students also engage in a dedicated ethics assignment to explore the ethical considerations surrounding the use of generative AI in power systems analysis and design.

To measure the perceived effectiveness of these hands-on Python exercises, the course includes an indirect assessment in the form of midterm and end-of-term surveys, gathering feedback on the students' learning experiences.

A vital feature of this approach is the coupling of Python with PowerWorld, a widely used commercial tool for power systems analysis. This integration provides a balanced learning experience that combines custom Python programming with industry-standard software. Through hands-on exercises and problem-solving, students enhance their understanding of power systems theory and apply these concepts in real-world contexts.

This initiative streamlines the problem-solving process, making power systems analysis more accessible and relevant to students. By the end of the course, students demonstrate improved computational skills, practical experience, and an ability to bridge theoretical knowledge with industry application. The early introduction of Python, alongside the development of more advanced programming capabilities, positions students for success in subsequent courses and prepares them for professional careers in power systems engineering.

2 Background

Computer modeling of electric power systems has been the cornerstone of power system analysis since the 1970s. Accurate models and algorithms are essential for enabling quick and precise solutions in power systems engineering[1]. Power system simulators, such as PowerWorld, are widely used for solving complex power system analysis problems and for educational purposes[2]. The textbook used in this course[3] provides a student version of PowerWorld. However, several analytical calculations must be performed to parameterize a system in PowerWorld. Computer programming is an effective method for performing these analyses[4], benefiting both power system engineers and students.

In recent years, many online courses have focused on teaching Python programming to solve power systems analysis problems. These courses often leverage Python-based packages and tools such as PYPSA[5] and Pandapower[6] for tasks like load flow analysis, short circuit calculations, and optimization studies. These resources provide learners with hands-on experience applying Python to power systems, offering practical skills for students and professionals in the field. For example, the course ”Power System Analysis with Python,” available on Udemy, emphasizes using Python to model and simulate complex power systems, incorporating open-source libraries specifically designed for this purpose[7].

In[8], the authors present a course where students build a power systems simulator from scratch. Additionally, the Open Distribution System Simulator (OpenDSS) is a popular research tool in the industry and can be controlled using Python[9]. Similarly, PowerWorld Simulator integrates Python scripting, enabling advanced analysis and automation.

Python has become an essential tool in the power systems field. In this paper, the authors explore enhancing a fundamental course in power systems analysis with Python programming exercises.

3 Python Exercises

In this section, the three Python assignments used in the course are discussed. The first assignment focuses on using the per-unit system to analyze a multizone system, where students implement a power factor correction solution. In the second assignment, students explore the impact of varying the bundling configuration of a three-phase transmission line on its impedance and admittance. The third assignment examines different line models and their influence on traveling waves as the distance increases.

3.1 Python Exercise 1: Per-Unit System Analysis and Power Factor Correction

In this exercise, students develop a Python program to analyze a power system composed of transmission lines, transformers, and loads. The system configuration, as shown in the provided diagram, includes two transmission lines and two loads interconnected by transformers. This project is designed to enhance students’ computational skills in power systems analysis by guiding them through the calculation of per-unit values, voltages, currents, and the effects of power factor correction. This system is shown in the following figure.

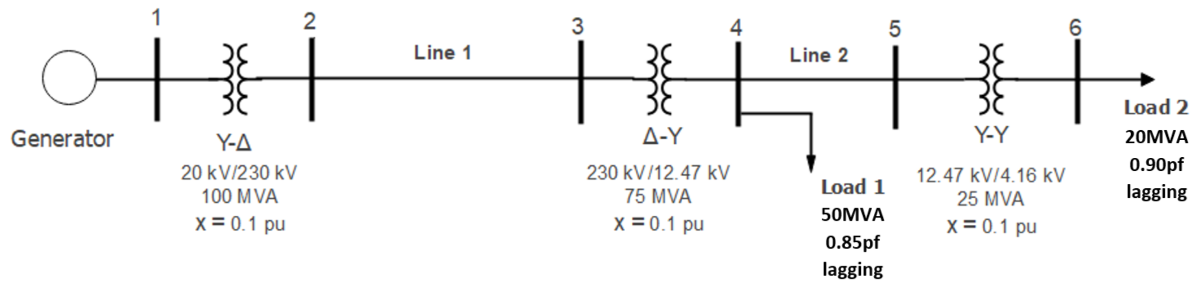


Figure 1: System for programming assignment 1

Step 1: Establishing the Per-Unit System

The first step involves setting up the per-unit system using a base power of 100 MVA and a base voltage of 20 kV on the generator side. Students are tasked with calculating the corresponding base quantities for current and impedance. The calculations must be adaptable so that any changes to the power base or voltage base at the generator automatically update all derived base quantities across the system. This modularity ensures that the script remains flexible and easily adjustable for different power system scenarios.

The program should also calculate and display the per-unit impedance values for the two transmission lines. Line 1 has an impedance of $Z = 0.0376 + j0.5277\,\Omega/$mi, with a length of 100 miles, while Line 2 has an impedance of $Z = 0.4576 + j1.0780\,\Omega/$mi, extending over 2000 feet. After calculating the per-unit values, students draw a per-unit impedance diagram of the entire system, incorporating the phase shifts introduced by the transformers. This diagram serves as a foundation for subsequent calculations, providing a visual representation of the system's parameters in the per-unit domain.

Step 2: Calculating Per-Unit Voltages and Currents

Following the establishment of the per-unit system, students calculate the per-unit voltages and currents at each of the six buses in the system, starting from the generator and moving toward the loads. An operating voltage of 4 kV is assumed at Load 2. The program computes the per-unit voltage at each bus (e.g., $V_{6,pu}$ at Bus 6) and the per-unit current flowing into each bus (e.g., $I_{6,pu}$ at Bus 6). This step involves applying basic power systems principles to calculate voltage drops along the transmission lines and the corresponding current flows based on the load data.

Step 3: Conversion to Physical Quantities

Once the per-unit voltages and currents have been determined, students convert these values back into physical quantities using the corresponding base values. This involves scaling the per-unit results by the appropriate base quantities to obtain the actual voltages and currents at each bus. The program outputs these physical values, enabling students to observe the relationship between per-unit and physical quantities and to validate their calculations.

Step 4: Power Factor Correction

An essential component of this exercise is the implementation of a power factor correction mechanism. Students add a parameter to their program, called `correction`, that allows for adjusting the power factor of the loads. The parameter represents the degree of correction applied, with a value of 1.0 indicating full correction to unity power factor and 0.0 indicating no correction. Partial corrections, such as `correction` = 0.5, allow students to observe the effects of varying levels of power factor adjustment. After implementing the correction, the program recalculates the voltages and currents, displaying the updated results to illustrate how the power factor correction impacts the system.

Step 5: Analysis of Voltage Levels

Power systems require that voltage levels remain within 95% to 105% of their rated values to ensure proper operation. In this step, students analyze the system voltages before and after applying power factor correction to determine if they fall within the acceptable range. This analysis provides insight into how power factor correction can influence voltage stability and overall system performance, especially in terms of maintaining voltage levels within prescribed limits.

Step 6: Exploring the Effects of Changing Base Values

The final task in the exercise is to modify the base values for voltage and current to a new set of values. If the per-unit system is implemented correctly, changing the base values should automatically update all the per-unit quantities while leaving the physical quantities unchanged. This consistency demonstrates the utility and flexibility of the per-unit system in power systems analysis. Students are encouraged to experiment with different base values and reflect on the results, discussing any patterns or observations that arise from changing the base quantities.

3.2 Python Exercise 2: Transmission Line Bundling and Power Factor Correction

In this exercise, students develop a Python program to analyze the impact of conductor bundling on a power system. The line data provided includes fully transposed lines made with Cardinal ACSR conductors. This analysis compares unbundled phases to phases bundled with 2, 3, and 4 conductors per phase. The system is shown below in 2

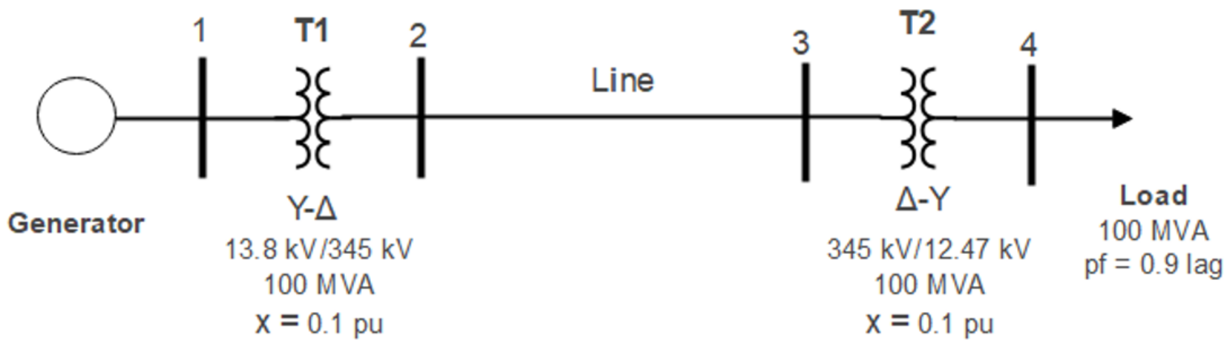


Figure 2: System for programming assignment 2

Figure 3 shows the spacing for each of the three-phase conductors. This figure is used so that the students can algorithmically calculate the geometric mean distance between the three phases which is needed for impedance and admittance calculations.

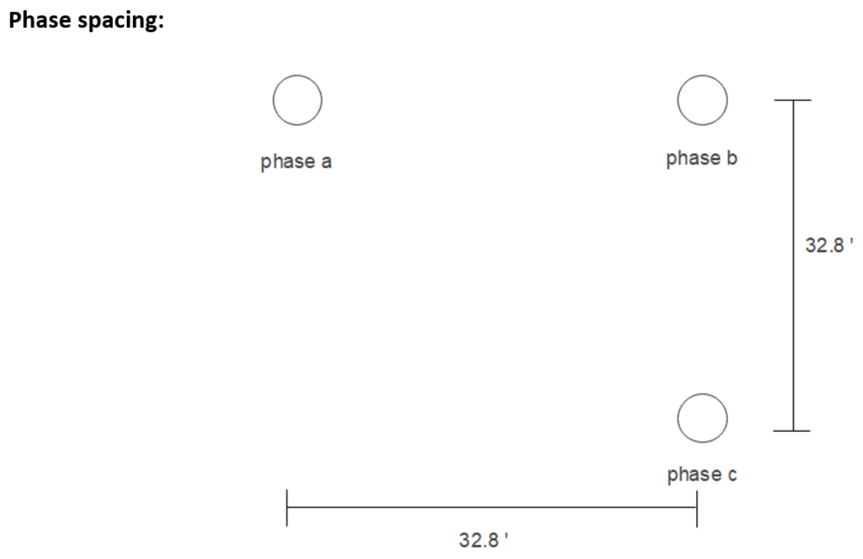


Figure 3: Conductor spacing for project 2

Figure 4 shows the three different configurations for conductor bundling. The number of bundled subconductors and their respective spacing has a direct effect on the phase conductor's geometric mean radius and equivalent physical radius which affects inductance and capacitance calculations.

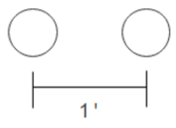
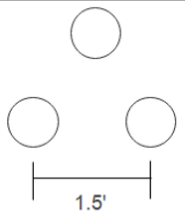
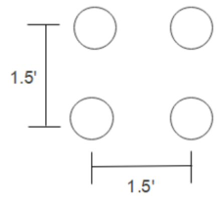
N = 2	N = 3	N = 4
		

Figure 4: Bundle spacing for project 2

Step 1: Determine Line Impedance and Admittance for Different Bundling Configurations

Write a Python program which determines the line's series resistance and reactance and the line's shunt admittance for each bundling configuration. The program should have a parameter called bundle_num which determines the number of bundled subconductors. In your report, please show the per-unit and actual resistance, reactance, and admittance.

Step 2: Powerworld Simulation

Model the system in Powerworld and run the simulation for each bundling configuration. In your report, create a table showing the per-unit voltage and angle at the generator and load bus, generator currents, and power loss.

Step 3: Reflection on Bundling Effects

Write a paragraph reflecting on the effects of conductor bundling. Discuss how bundling positively impacts the system and note any potential negative impacts.

Step 4: Power Factor Correction

If the bus voltages fall outside the range of nominal values, implement a power factor correction solution. Update your Python program and Powerworld simulations accordingly to apply this correction and rerun the analysis.

Step 5: Reflection on Power Factor Correction

Compare this power factor correction solution to others studied in the course. Explain any differences in methodology and outcomes.

3.3 Python Exercise 3: Traveling Waves in Power Systems

In this exercise, students plan to study the effect of short, medium, and long transmission lines and determine the error in the models for different distances. In addition, students explore voltage profiles as a function of loading and distance and reactive power compensation techniques. Due to space limitations for this paper, this project can be found at the GitHub page in [reference left out for double-blind purposes].

4 Analysis of Results

Two surveys were used to evaluate the effectiveness of the course teaching: a midterm survey and an end-of-term survey, both administered by the University Center for Teaching and Learning. These surveys provided an indirect assessment of student learning.

4.1 Midterm Survey

The midterm survey was designed to assess students' preexisting perceptions of their Python programming skills and their understanding of programming's role in power systems analysis. Administered during the fourth week of the class, this survey combined Likert scale and open-ended formats to capture a range of insights.

Two questions (Q1 and Q2) used a five-point Likert scale. Q1 asked, ”I understand the role of programming (i.e., Python) in Power Systems Analysis,” while Q2 asked, ”I feel confident in my ability to use Python for solving basic engineering problems.” The Likert scale ranged from ”Strongly Disagree” (1) to ”Strongly Agree” (5) for both questions.

An open-ended question (Q3) invited students to provide qualitative feedback on potential course improvements. This question asked, ”Do you have any recommendations to improve the course?” Responses to this question offered valuable insights into students' perceptions of course structure and teaching methods, particularly regarding the Python section.

The results from Q1, *I understand the role of programming (i.e., Python) in Power Systems Analysis* are shown in Figure 5 and results from Q2, *I feel confident in my ability to use Python for solving basic engineering problems* are shown in Figure 6

The results from Q3, *Do you have any recommendations to improve the course?*, varied widely. Many students did not provide specific input on the programming aspect of the course but offered general feedback such as, ”I have been really appreciating the office hours, and they have really helped with my comprehension of the material,” and, ”I like the course so far; it has been manageable and interesting.

I also appreciate some of the explanation of the history of power systems.” However, some comments directly addressed programming-related aspects. For instance, one student suggested, ”Would we be able to do a crash course sort of thing for Python?” while another remarked, ”I would prefer if Python were introduced in more depth in class.” Interestingly, one comment revealed resistance to the use of generative AI tools: ”I am coming into this course with zero Python experience, nor do I really support the use of AI and don't intend on using it.” Overall, aside from these specific examples, most students appeared satisfied with the course.

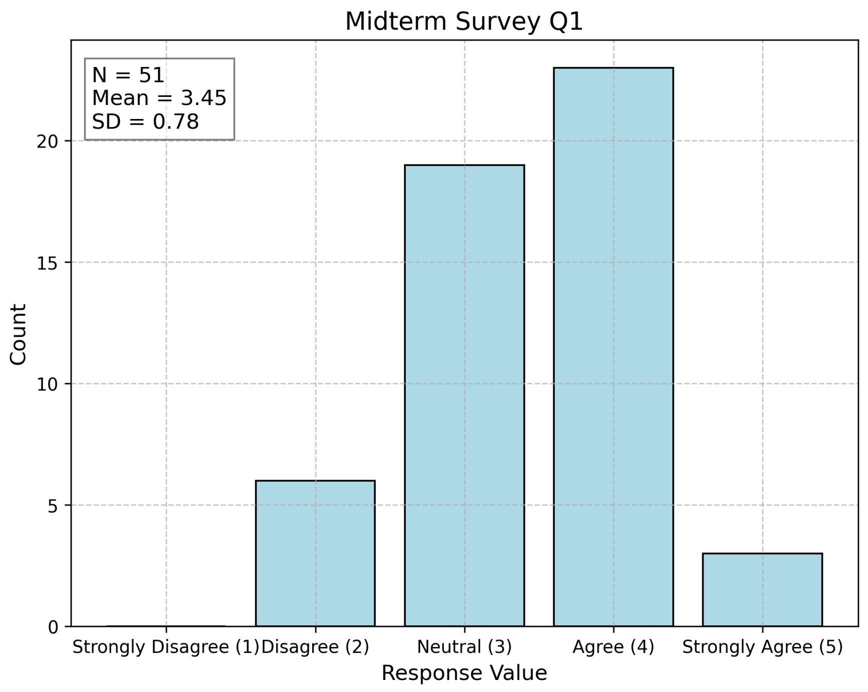


Figure 5: Midterm Survey Q1 - ”I understand the role of programming (i.e., Python) in Power Systems Analysis”

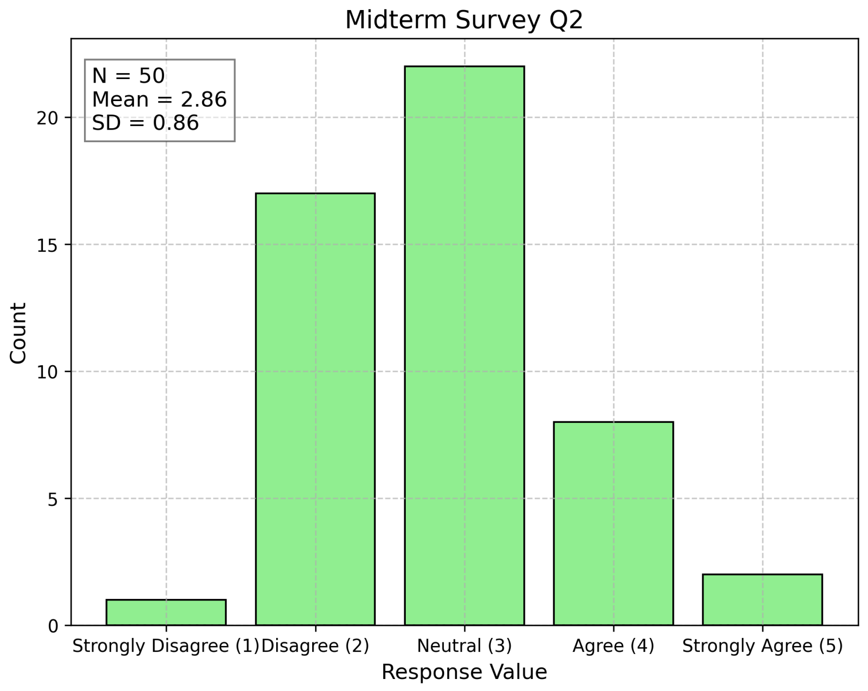


Figure 6: Midterm Survey Q2 - ”I feel confident in my ability to use Python for solving basic engineering problems”

4.2 End-of-Term Survey

The end-of-term survey was designed to evaluate students' perceptions of their Python programming skills, their understanding of Python's role in Power Systems Analysis, and their use of generative AI tools in learning. Administered during the final week of class, the survey included five questions that combined Likert-scale and open-ended formats, aiming to capture students' reflections on their learning experiences.

Three questions (Q1, Q2, and Q4) used a five-point Likert scale. Q1 asked, ”I understand the role of programming (i.e., Python) in Power Systems Analysis,” while Q2 asked, ”I feel confident in my ability to use Python for solving basic engineering problems.” Q4 focused on the use of generative AI tools with the question, ”To what extent did you use generative AI tools (e.g., ChatGPT, GitHub Copilot) to assist you in learning or completing Python assignments?” The Likert scale ranged from ”Strongly Disagree” (1) to ”Strongly Agree” (5) for Q1 and Q2, and from ”Never” (1) to ”Always” (5) for Q4.

Two open-ended questions (Q3 and Q5) provided opportunities for students to share qualitative feedback. Q3 asked, ”Is there anything you would like the instructor to know, or do you have any recommendations for improving the Python section of the course?” Q5 invited students to reflect on their experiences with generative AI tools by asking, ”Is there anything you would like to share about your experience with generative AI tools (e.g., ChatGPT, GitHub Copilot) in this course?”

These surveys offered an efficient, anonymous method to gather both quantitative and qualitative data, enabling an indirect assessment of teaching effectiveness and student engagement.

The results from Q1, *I understand the role of programming (i.e., Python) in Power Systems Analysis* are shown in Figure 7. The results from Q2, *I feel confident in my ability to use Python for solving basic engineering problems* are shown in Figure 8. The results from Q4, *To what extent did you use generative AI tools (e.g., ChatGPT, GitHub Copilot) to assist you in learning or completing Python assignments?* are shown in Figure 9

The results from Q3, *Is there anything you would like the instructor to know, or do you have any recommendations for improving the Python section of the course?* are summarized below.

Student feedback on the Python section of the course highlighted several key areas for improvement and appreciation. A common theme was the need for an earlier introduction to Python. Many students suggested that starting the course with a basic introduction or a simple example assignment before larger projects would help those with no prior experience in Python. For instance, one student remarked, “Starting the course with an intro to Python would be helpful, as we haven't used it in school prior to this class.”

Another frequently mentioned point was the desire for more examples and guidance during class. Students emphasized the value of in-depth examples and instructions to better prepare for assignments, with one noting, “Provide more examples during class time and more in-depth instructions.” Additionally, several students requested clearer and more concise assignment instructions, as they occasionally found the requirements ambiguous.

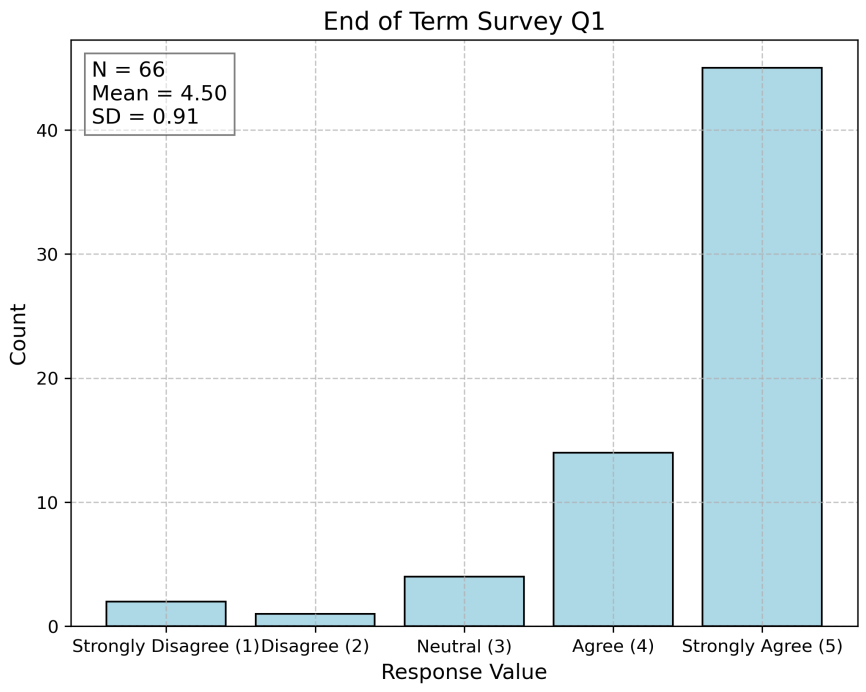


Figure 7: End of Term Survey Q1 - ”I understand the role of programming (i.e., Python) in Power Systems Analysis”

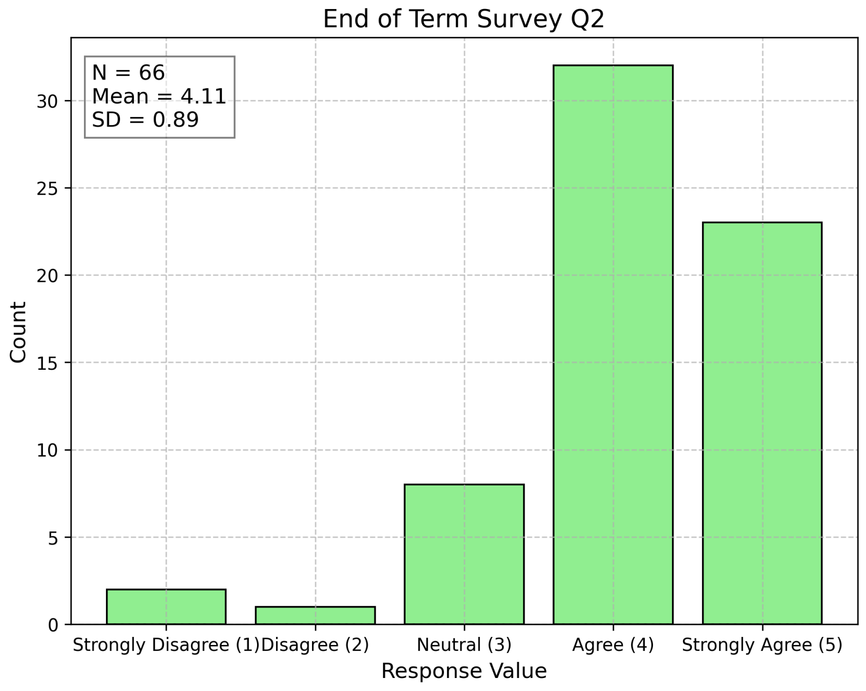


Figure 8: End of Term Survey Q2 - ”I feel confident in my ability to use Python for solving basic engineering problems”

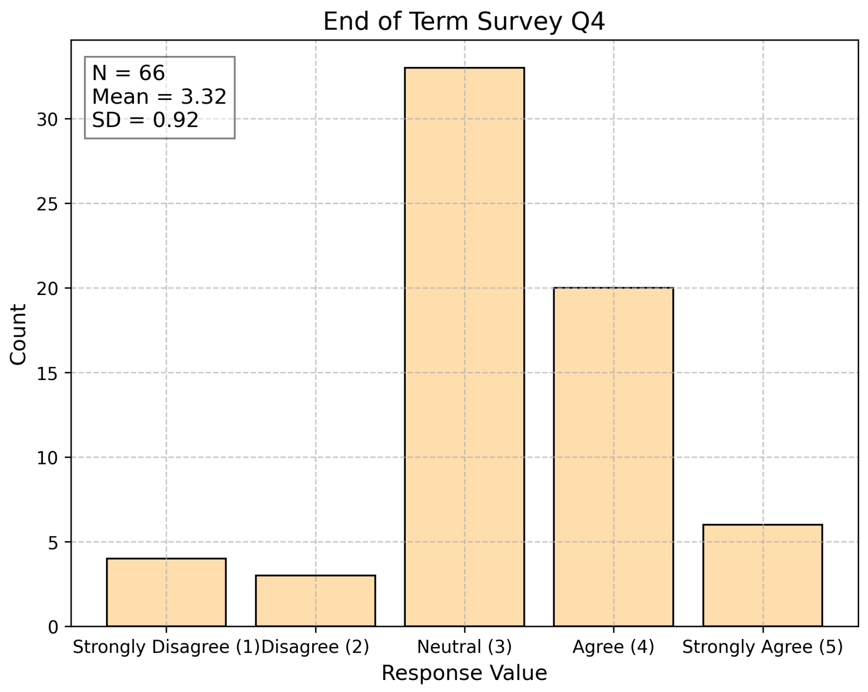


Figure 9: End of Term Survey Q4 - ”To what extent did you use generative AI tools (e.g., ChatGPT, GitHub Copilot”

Despite these challenges, students generally appreciated the integration of Python into the course and its role in reinforcing key concepts in power systems analysis. They acknowledged that the hands-on coding assignments helped bridge theoretical and practical knowledge, with one student stating, “The Python code helped reinforce the concepts learned in class. I felt like doing the code made it easier to do the problems in class.” While some mentioned the challenges of adapting to a new language and incorporating generative AI tools, they valued Python as a useful tool for learning and solving engineering problems.

The results from Q5, *Is there anything you would like to share about your experience with generative AI tools (e.g., ChatGPT, GitHub Copilot) in this course?* are summarized below.

Student feedback on the use of generative AI tools (e.g., ChatGPT, GitHub Copilot) in the course revealed several insights into their utility and limitations. A recurring theme was the effectiveness of these tools for debugging, learning syntax, and simplifying monotonous tasks. One student noted, “AI tools were very helpful in debugging my code and helping me understand some of the syntax.” Many students found AI useful in the early stages of learning Python but relied less on it as they became more familiar with the language.

Another common observation was the limitations of AI when applied to engineering-specific tasks. While ChatGPT was useful for outlining code or generating basic functions, students highlighted its inability to consistently handle complex equations or domain-specific problems. For example, one student remarked, “ChatGPT was helpful for debugging, but it almost never got the equations correct when I asked it to assist with the code for calculating inductance.”

Students emphasized the importance of proper usage, noting that AI tools should complement, not replace, foundational learning. Some expressed concern about over-reliance on AI and the temptation to use it without fully understanding Python syntax. As one student stated, “I avoided over-reliance by trying to learn it first and then utilizing AI as a tool. In this way, I found the experience to be more beneficial in developing my skills.”

Overall, while generative AI tools were appreciated for their ability to streamline coding tasks and provide learning support, students stressed the need for careful oversight and foundational knowledge to use them effectively.

5 Discussion

Incorporating Python programming into this course has been a longstanding goal for the instructor. Programming, particularly Python, has become an essential tool for power systems engineers, making its inclusion in the curriculum imperative. The rollout of Python exceeded expectations in terms of its success.

Two primary goals for the course were to increase students’ awareness of the role of programming in power systems analysis and to develop their ability to solve problems using Python. Survey results from both the midterm and end-of-term evaluations indicate that these outcomes were overwhelmingly achieved.

For the question on whether students understood the role of Python in power systems, the mean score increased from 3.45 out of 5 at midterm to 4.5 by the end of the course. By the end of the semester, 45 out of 66 students strongly agreed, and 14 agreed with this statement, leaving only 7 students neutral or in disagreement.

Similarly, for the question on whether students felt confident using Python to solve basic engineering problems, the mean score rose from 2.86 at midterm to 4.11 by the end of the course. By the end of the semester, 23 students strongly agreed, and 32 agreed, leaving only 11 students neutral or in disagreement, with 8 neutral and only 3 in disagreement.

Another noteworthy outcome was students’ use and perception of generative AI tools. For the end-of-term question on the extent to which students used generative AI, the mean score was 3.32, indicating a slight favorability toward its use. Students reported using generative AI primarily for learning Python syntax and debugging but did not rely on it exclusively. Interestingly, 7 out of 66 students stated that they either did not use generative AI or used it rarely. This finding suggests mixed attitudes toward generative AI, likely influenced by students’ prior experiences and familiarity with such tools. It would be valuable to reassess this in a future semester with explicit instruction on how to effectively use generative AI, rather than merely encouraging its use.

6 Conclusion

The integration of Python programming into an introductory power systems analysis course has proven to be a transformative addition to the curriculum. By providing students with hands-on experience and practical tools, this approach bridges the gap between theoretical concepts and their application in real-world engineering challenges. Survey results clearly indicate significant improvements in students' understanding of Python's role in power systems analysis and their confidence in using it to solve complex problems.

The three Python projects, paired with industry-standard tools like PowerWorld, not only deepened students' technical skills but also fostered their ability to think critically about power systems challenges. Additionally, the exploration of generative AI tools as a supplementary resource demonstrated both the potential and the limitations of these technologies in an educational context.

As the demand for computational skills in power systems engineering continues to grow, incorporating programming into foundational courses equips students with essential competencies for both academic progression and professional success. Future iterations of this course will further refine the integration of Python and generative AI tools to enhance learning outcomes and better prepare students for the evolving energy landscape.

References

[1] W. Kersting and R. Kerestes, *Distribution System Modeling and Analysis with MATLAB® and WindMil®*, 5th ed. CRC Press, 2022.

[2] PowerWorld, "Powerworld simulator," 2023, accessed: Apr. 17, 2023.

[3] J. D. Glover, M. S. Sarma, and T. Overbye, *Power System Analysis & Design, SI Version*. Cengage Learning, 2012.

[4] F. Milano, "Experience of unix terminal-based labs for undergraduate modules on power system analysis," in *EDULEARN14 Proceedings*. IATED, 2014, pp. 268–277.

[5] P. Community, "Pypsa: Python for power system analysis," 2024, an open-source Python environment for energy system modeling, analysis, and optimization. Accessed: Jan. 22, 2025. [Online]. Available: https://pypsa.org

[6] pandapower, "pandapower: An easy to use open source tool for power system modeling, analysis and optimization with a high degree of automation," 2024, accessed: Jan. 22, 2025. [Online]. Available: https://www.pandapower.org

[7] Udemy, "Power system analysis with python: Unlock the future of energy," 2025, accessed: Jan. 22, 2025. A comprehensive course on Python-powered power system analysis using PYPSA and Pandapower. [Online]. Available: https://www.udemy.com/course/power-system-analysis-with-python/

[8] P. Radatz and R. Kerestes, "A novel approach to teaching power systems analysis and design using software development," in *ASEE Annual Conference & Exposition*, Baltimore, MD, Jun. 2023.

[9] EPRI, "Opendss," 2019, accessed: Apr. 16, 2023. [Online]. Available: https://www.epri.com/pages/sa/opendss