



Research in Engineering Education Symposium (REES)
Regional Conference in Engineering Education (RCEE)
& Research in Higher Education (RHEd)
September 29 – October 3, 2025
Johor, Malaysia



Combining Simulation and Coding to teach the Next Generation of Engineers and Researchers

Gautham Varma and Kaitlin Tyler
Ansys India Pvt Ltd.
Corresponding Author's Email: gautham.varma@ansys.com

ABSTRACT

CONTEXT

A skill gap is forming between academia and industry, leaving new graduates ill-prepared for the challenges of working on real world problems (Spang, 2014). These skills are both technical and professional and include areas such as complex problem solving, critical thinking, and quickly adapting to ever-evolving technologies (*i.e.* simulation and coding). The rapidly changing technical skills pose a challenge due to industry's need to always evolve while academia needs to balance teaching theoretical skills and preparing students for work.

PURPOSE

Ansys, part of Synopsys looks to reduce the industry-academia skill gap by empowering educators to include simulation-based learning in their curriculum.

APPROACH

One approach Ansys is taking is providing Educational Resources that directly plug into the curriculum including simulation examples, case studies, and more. For this work, we focus on leveraging open-source Python libraries (Kaszynski, 2020) (PyAnsys) with Ansys simulation solvers to create Project-Based Learning (Bayat & Tarmizi, 2012) (PBL) classroom activities. By combining simulation and coding in a project-based format, students gain experience in applying theoretical knowledge to solve real-world problems using industry-level tools.

ANTICIPATED OUTCOMES

We anticipate that these educational resources will help reduce this skill gap from multiple perspectives. From the educator perspective, we hope to reduce energy barrier to curriculum change, allowing for quick adaptation to industrial needs. From the student perspective, we hope to provide them with the opportunity to learn fundamental theory in the context of industrial applications, gaining critical thinking and problem-solving skills in the process.

CONCLUSIONS

We in the Ansys are uniquely positioned to understand the challenges of graduates when transitioning from academia to the workforce. Hence, we are committed to supporting educators looking to upskill their students before graduation. In this paper, we present our method of combining PBL, simulation, and Python in educational resources designed to teach both theoretical knowledge and practical skills.

KEYWORDS

Problem-Based Learning, Simulation, Python, Industry-Academia Skill Gap.

Introduction

Traditional fluid dynamics education typically introduces core concepts—such as drag, vortex shedding, and flow separation—through theoretical instruction (Kundu, Cohen, Dowling, & Capecelatro, 2024). While these principles are foundational to fluid mechanics, students often struggle to relate them to practical engineering scenarios (Palmer & Hall, 2011). Industrial Computational Fluid Dynamics (CFD) simulation platforms, such as Ansys Fluent, can help connect theory with practice. The interface, however, can be overwhelming; novice users often find themselves struggling with procedural tasks such as geometry creation, meshing, boundary condition setup, and solver configuration. As a result, their focus is diverted from understanding physical phenomena to managing software complexity.

To bridge this gap, we propose an integrated, hands-on approach that combines Python scripting with the Ansys Fluent solver, using the PyAnsys (Ansys, 2025) (Virtanen, et al., 2020) ecosystem using Problem-Based Learning (PBL) (Kokotsaki, Menzies, & Wiggins, 2016). This approach empowers students to explore fluid dynamic behavior while simultaneously developing essential computational skills. By automating repetitive and error-prone processes, students can dedicate more time to applying their theoretical knowledge to solve the problems scaffolded in the activity. In doing so, they cultivate a deeper understanding of both the governing physics and modern CFD workflows, while also acquiring valuable programming capabilities increasingly demanded in contemporary engineering practice.

Project Objectives

The goal of this project was to create a PBL activity that covers the following three learning objectives:

1. Relate theoretical fluid dynamics concepts to real world applications
2. Write Python code to solve fluid dynamics problems
3. Recognize the components of a basic CFD workflow

The core theoretical question used to create the PBL activity: "At what velocity does flow separation and vortex shedding occur around a 2D bluff body, and how does this affect the resulting drag force?"

This problem invites students to explore the behavior of fluid flow around bluff bodies—such as cylinders or square prisms—where sharp geometrical features cause early boundary layer separation and unsteady wake formation. These phenomena are central to many engineering applications, including bridge aerodynamics, offshore structures, and automotive drag optimization.

Project Implementation

This project is designed around a structured six-step workflow that intertwines core fluid dynamics concepts with hands-on computational fluid dynamics (CFD) (Anderson & Wendt, 1995) practice. Each stage provides a dual learning outcome: reinforcing key physics principles while developing modern engineering skills such as scripting, automation, simulation, and data analysis. The approach enables students to actively experiment with physical parameters while leveraging Python-based tools for efficient, reproducible simulation.

Step 1: Parametric Bluff Body Geometry Creation

The first step introduces students to the importance of geometry in dictating fluid flow behavior around bluff bodies. Here, they explore how blunt shapes such as cylinders and square prisms disrupt the oncoming flow, resulting in early boundary layer separation and the formation of large wakes. By scripting the geometry creation process in Python, students develop a parametric approach to modeling. This enables them to modify geometric parameters like diameter, length, and corner radius with ease, fostering an appreciation for how even small

changes in shape can significantly influence aerodynamic performance. Beyond the physics, students gain technical experience in using Python and PyAnsys tools to create and export CAD-compatible geometries, building a foundation for automation in engineering design workflows.

Step 2: Meshing with Automated Refinement

In the third step, the focus shifts to mesh generation—a crucial phase that balances simulation fidelity with computational efficiency. Students delve into the physical need for refined meshes near solid surfaces to resolve boundary layers and capture flow detachment. They learn how mesh density directly impacts the resolution of small-scale flow structures, such as eddies and vortices, in the wake. Using PyAnsys meshing utilities, students script the creation of both structured and unstructured meshes and apply local refinements near the bluff body where gradients are steepest. This hands-on meshing experience deepens their understanding of grid independence and mesh quality criteria, while also demonstrating how scripting facilitates repeatability and scalability across multiple design variations.

Step 3: Computational Domain and Boundary Definition

Once the bluff body geometry is generated, students proceed to define the computational domain in which the simulation will be conducted. From a physics perspective, they learn how domain size and boundary placement affect the ability to accurately capture flow features such as separation points, recirculating wakes, and vortex shedding. An undersized domain can introduce artificial effects like boundary reflections or prematurely dampen unsteady vortices, leading to unrealistic results. Through scripting, students automate the definition of domain extents and assign appropriate boundary conditions—including velocity inlets, pressure outlets, symmetrical planes, and no-slip walls. This step reinforces the importance of boundary condition selection in governing the flow field and helps students understand how computational assumptions must align with physical reality.

```
# Applying velocity as a boundary condition for the simulation
solver.setup.boundary_conditions.velocity_inlet["inlet"].vmag = U_inf

# Activating residuals to check simulation convergence
solver.tui.solve.monitors.residual.plot("yes")

# Initializing the solution
solver.solution.initialization.hybrid_initialize()

# Setting a maximum number of iterations:
solver.solution.run_calculation.iterate.get_attr("arguments")
if solver.get_fluent_version() >= "23.1.0":
    solver.solution.run_calculation.iterate(iter_count=150)
else:
    solver.solution.run_calculation.iterate(number_of_iterations=150)
```

Figure 1: A demo script to define the Boundary and initial condition

Step 4: Simulation Setup with Varying Reynolds Numbers

With the domain and mesh defined, students begin their simulations, systematically varying the inlet velocity to explore different flow regimes. This step emphasizes the role of the Reynolds number in characterizing flow behavior—highlighting the transition from steady, symmetric flow to unsteady, asymmetric vortex shedding. Students observe how increasing flow velocity alters the pressure distribution and wake dynamics and identify the critical velocity at which unsteady shedding begins. By automating these simulations through parameterized scripts, they develop an efficient way to explore a wide range of physical scenarios. This reinforces the connection between theoretical fluid dynamics and real-world CFD, while simultaneously training students in best practices for batch processing and solver configuration.

Step 5: Post-Processing and Drag Analysis

After the simulations are completed, students turn to the critical task of post-processing. The physics objective here is to quantify how changes in flow structure affect aerodynamic performance, particularly the drag force and drag coefficient (C_d). Students analyze time-resolved pressure fields, identify flow separation points, and visualize vortex patterns using contour plots and streamline animations. With tools like PyFluent and Python plotting libraries (e.g., Matplotlib) (Tosi, 2009), they write scripts to extract surface forces, compute average and fluctuating drag values, and generate graphical representations of key flow features. This stage underscores the value of computational tools in transforming raw simulation data into actionable engineering insights and instills the importance of automated, reproducible analysis pipelines. An example of vortex shedding behind the bluff body can be seen in the Figure 2.

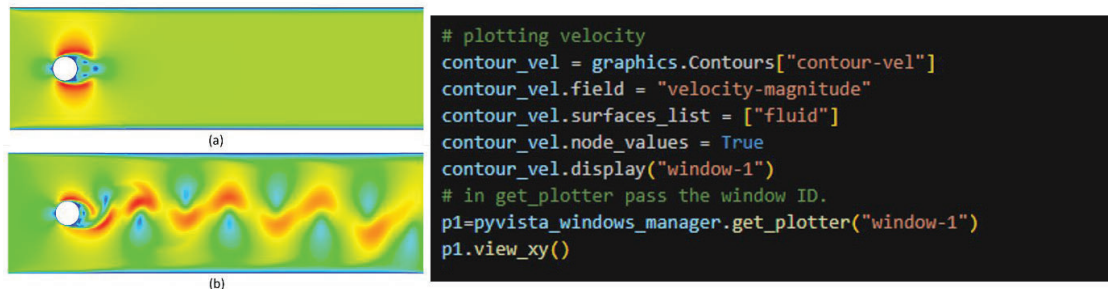


Figure 2 : Contour plot explaining the (a) flow separation behind the cylinder and (b) the vortex shedding observed and a demo script to get the velocity plots

Step 6: Reflection and Skill Integration

In the final step, students are encouraged to synthesize the knowledge and skills they have acquired throughout the workflow. They reflect on how each decision—geometry, domain setup, mesh resolution, or inlet velocity—affects the flow outcome, providing a holistic view of CFD as a physics-informed design process. By comparing simulation results against theoretical expectations or literature benchmarks, students learn to validate their work and develop critical judgment. More importantly, this step helps them recognize the value of scripting and automation in modern engineering practice, where multidisciplinary skills in physics, programming, and digital tools are essential for solving complex design problems efficiently and accurately.

Educational Impact

We believe this work is impactful because it makes simulation accessible in courses where it may have otherwise been challenging to include. By leveraging scripting, the project workflow encourages learners to engage more actively with simulation processes, not just as procedural tasks, but as a means for testing hypotheses and observing physical phenomena. As a result, students develop both a deeper grasp of fundamental fluid dynamics and practical proficiency in modern CFD tools, effectively bridging the gap between theoretical instruction and real-world engineering applications.

Next Steps and Limitations

The next step for this work is to test the activities within a classroom and collect student and instructor feedback. This is a challenge for companies like Ansys, requiring university collaboration to complete. Minimum three universities from around the world will be chosen to ensure the applicability of the topic and strategy across different curriculum.

Conclusion

In this paper, we have shown part of our efforts in closing the industry-academia skill gap through educational content by focusing on bringing technical skills like coding and simulation into fundamental courses. By utilizing a combination of open-source Python libraries, industrial software, and a Problem-Based Learning approach to content creation, we have shown how a fundamental concept in fluid dynamics can be explored in a more approachable and exploratory experience for students. It is our hope that this approach will assist in reducing the barrier to including industry-relevant technical skills across the engineering curriculum.

References

- Anderson, J. D., & Wendt, J. (1995). *Computational fluid dynamics*. New York: McGraw-Hill.
- Ansys. (2025, May 8). *PyAnsys*. (Ansys, Inc) Retrieved 2025, from <https://docs.pyansys.com/>
- Bayat, S., & Tarmizi, R. A. (2012). Effects of Problem-based Learning Approach on Cognitive Variables of University Students. *Procedia-Social and Behavioral Sciences*. doi:<https://doi.org/10.1016/j.sbspro.2012.06.027>
- Kaszynski, A. (2020). Pyansys: Python interface to MAPDL and associated binary and ASCII files. *Zenodo*, 10.
- Kokotsaki, D., Menzies, V., & Wiggins, A. (2016). Project-based learning: A review of the literature. *Improving schools*, 19(3), 267-277.
- Kundu, P. K., Cohen, I. M., Dowling, D. R., & Capecelatro, J. (2024). *Fluid Mechanics*. Elsevier.
- Palmer, S., & Hall, W. (2011). An evaluation of a project-based learning initiative in engineering education. *European journal of engineering education*, 36(4), 357-365.
- Spang, D. (2014). Curriculum design and assessment to address the industry skills gap. *2014 ASEE Annual Conference & Exposition*, 24-345.
- Tosi, S. (2009). *Matplotlib for Python developers*. Packt Publishings Ltd.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., . . . al, e. (2020). SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature methods*, 17(3), 261-272.

Copyright Statement

Copyright © 2025 Gautham Varma and Kaitlin Tyler: The authors assign to the Research in Engineering Education Network (REEN) and educational non-profit institutions a non-exclusive license to use this document for personal use and in courses of instruction provided that the article is used in full, and this copyright statement is reproduced. The authors also grant a non-exclusive license to REEN to publish this document in full on the World Wide Web (prime sites and mirrors), on portable media and in printed form within the REES and RCEE & RHEd 2025 proceedings. Any other usage is prohibited without the express permission of the authors.