2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
Paper ID #50012

Application of Two CFD Packages to Savonius Wind Turbine Simulation: Perspectives on Suitability for Undergraduate Research and Instruction

Dr. Robert T. Bailey P.E., Loyola University Maryland

Dr. Robert T. Bailey is currently a Professor of Mechanical Engineering in the Department of Engineering at Loyola University Maryland. He received his B.S., M.S., and Ph.D. degrees in Mechanical Engineering from the University of Florida. He worked in industry for Westinghouse and Science Applications International Corporation, served as a senior program officer at the National Research Council, and taught previously at the University of Tennessee at Chattanooga. His research interests include mechanistic engineering analyses to support risk and safety assessment of industrial processes; application of computational fluid dynamics to heating, ventilating, and air conditioning systems and wind turbines; and improvements in engineering education. Dr. Bailey is a member of the American Society of Mechanical Engineers and the American Society for Engineering Education and is a registered professional engineer in the state of Maryland.

Application of Two CFD Packages to Savonius Wind Turbine Simulation: Perspectives on Suitability for Undergraduate Research and Instruction

Abstract

In undergraduate research projects that involve the application of Computational Fluid Dynamics (CFD), the selection of an appropriate software package can be an important consideration. Commercial packages vary in both complexity and capabilities. Project goals can also differ, but if publication-level results are desired, then solution accuracy becomes critical. More complicated codes often yield more accurate solutions, yet they can also overwhelm undergraduates with an unmanageable number of options and level of complexity. In this paper, we explore the application of two commercially available CFD packages—SOLIDWORKS Flow Simulation (SWFS) and Ansys Fluent—and compare them based on solution accuracy and ease of implementation by undergraduates. The scenario investigated involves the behavior of a Savonius vertical axis wind turbine for which experimental performance data are available. It was found that both SWFS and Ansys Fluent were able to accurately calculate the turbine power coefficient (C_P) for lower tip speed ratios (TSRs). But past the peak C_P, SWFS predicted higher C_P values that were not supported by experimental data, while Ansys Fluent did not. Examination of the numerically predicted flow fields in this regime revealed relatively small but significant differences in blade pressure distribution and vortex intensity that led to higher predicted turbine torque and power by SWFS. This suggests that SWFS should be used with care when applied to such rotating machinery problems. Student ease of use was assessed qualitatively based on feedback and reflections from undergraduate researchers. The CFD application process was divided into five steps, and a structured evaluation considering task complexity, number of activities required, and setup time needed for each step revealed that SWFS was significantly easier for the students to use than Ansys Fluent. Student feedback also aligned with educational theories involving cognitive load and spiral curricula, leading to a proposed framework for integration of CFD into undergraduate mechanical engineering curricula.

Introduction

Finite Element Analysis (FEA) is a key component of modern commercial design processes, and most employers now expect that newly graduated bachelor's level mechanical engineers should have some experience with FEA for stress and displacement analysis in their undergraduate studies [1]. Because of this, and due to the development and availability of user-friendly commercial software packages, FEA has become a common element within undergraduate mechanical engineering curricula [2]. FEA is also applied within the realm of computational fluid dynamics (CFD) and thermal-fluids engineering, but the more widely used computer codes in this area are generally based on a finite volume (FV) approach [3]. In some ways, the mathematics associated with FV methods is less complex, at least at the introductory level, yet their inclusion in undergraduate curricula is less widespread. One reason for this situation is that the discretization process for FV methods (*i.e.*, mesh or grid generation) is notoriously more complex in implementation than that for solid-mechanics-based FEA. Another reason is the myriad of options, including spatial and time discretization schemes, solver algorithms, and

turbulence models. This makes it more difficult for undergraduate students to use FV computer codes effectively without extensive training and instruction.

One way that software developers have tried to mitigate these challenges is to directly incorporate CFD capabilities into existing computer-aided design (CAD) packages while reducing the available user options and simplifying the grid generation process. This is the approach taken in SOLIDWORKS Flow Simulation (SWFS), an "add-in" module that integrates seamlessly with the popular SOLIDWORKS three-dimensional (3D) CAD package [4]. A student can create complex geometries with SOLIDWORKS—a tool with which many are already familiar—and then use SWFS to perform thermal-fluid simulations with that geometry. The immersed-body, structured, Cartesian approach to mesh generation built into the software has the potential to simplify the process of setting up a simulation, making CFD more accessible to undergraduates [5]. But the reduction in features and options relative to other more complex codes (*e.g.*, Ansys Fluent or STAR CCM+) could also lead to less accurate results.

Many different approaches have been taken for CFD instruction at the undergraduate level, but there are common themes. When students are first introduced to the process of analyzing a fluid flow problem using a CFD tool, solution accuracy is generally not the primary concern. Instead, the focus is on learning how the theoretical concepts and steps involved can be executed with the software. But once the framework and basic options are understood and simulations are performed, the focus turns to interpreting results, recognizing when solutions may be inaccurate, and devising strategies to test and improve the solutions. This is particularly true when undergraduates participate in research projects where the quality of the numerical solution is critical. Setting aside classroom instruction for the moment, it is reasonable to specify three objectives for such one-on-one (professor-student) projects.

1. The undergraduate student researcher will gain a better understanding of CFD theory and practice via application of a commercially available code to a realistic and important research problem.
2. The undergraduate student researcher will hone troubleshooting skills to eliminate problem setup errors, identify questionable solution attributes, and improve solution accuracy.
3. The results generated by the undergraduate student will be accurate enough to contribute meaningfully to the project and support peer-reviewed publication.

These objectives can be both complementary and competing, particularly when project time constraints are considered. That is, a slow and methodical instructional approach that incrementally leads the student to a better overall understanding of CFD will almost certainly yield better solutions, but it may take so long as to be at odds with the project schedule. It may also require more support from the instructor than is feasible in the given timeframe. This is particularly true if the software is complex with an extensive list of features designed for the experienced CFD analyst.

So, for research projects where undergraduate students are being asked to use commercially available CFD software, the impact of the particular package on achieving these objectives is of interest. This leads to two important questions:

- How does the accuracy of CFD packages where the modeling options have been intentionally limited compare to more complex software where such limitations are not present?
- How do differences in CFD package complexity affect the level of student time and effort required to create and conduct effective simulations?

It is recognized that answers to these questions can be student, problem, and software dependent. Nevertheless, understanding is built incrementally, and the examination of specific cases can lead to important insights and serve as a building block toward a more comprehensive meta-analysis.

The SWFS CFD module within SOLIDWORKS has already been presented as an example of software that has been intentionally streamlined for both student and design engineer use. In fact, SWFS is marketed as a tool that can be used by design engineers (who are not dedicated CFD analysts) to guide evolving product designs [6]. (Autodesk has a similar capability available for use with its Fusion 360 product [7, 8].) Ansys, on the other hand, is a multiphysics numerical analysis package whose CFD capabilities have generally not been simplified in this way. Ansys can simulate a variety of physical phenomena including deformation of solids, heat transfer, acoustics, electromagnetics, and fluid flow [9]. Ansys Fluent exists under the Ansys umbrella, and it is a general purpose CFD program capable of simulating very complex fluid mechanics problems [10].

SWFS and Ansys Fluent each include features designed to facilitate the analysis of fluid flow through rotating machinery including the designation of rotating regions and the automated calculation of torque from numerical solutions of velocity and pressure. Both codes have been applied to wind turbine flows with success [11,12]. However, limitations in SWFS accuracy for flows involving rotation have been reported [13,14].

With faculty guidance, undergraduate researchers at Loyola University Maryland have used both SWFS and Ansys Fluent to simulate two-dimensional (2D) airflow over vertical axis wind turbines with the goal of identifying optimal spacing configurations that maximize power harvesting. These physical situations contain flow features that are common to rotating turbomachinery, in general. In response to the two questions posed above, we present and interpret the results of a scenario involving a single wind turbine with a focus on both solution accuracy and ease of implementation by undergraduates. Comparisons with existing experimental data are made, and flow field characteristics and other results are used to identify where SWFS and Ansys Fluent perform well and where they are less accurate. Conclusions are also offered about the relative level of difficulty between the tools for undergraduates when setting up and performing the simulations and the impact this can have on research projects and instruction. Finally, a broader framework for the inclusion of CFD at the undergraduate level is proposed, based on cognitive load theory [15, 16] and the spiral curriculum [17,18].

Physical Problem Description

The physical scenario considered in this paper involves a single Savonius wind turbine subjected to a uniform crosswind. The Savonius, pictured in Figure 1, is one of the simplest vertical axis

wind turbines. This particular design is a two-bladed rotor consisting of two semicircular blades with top and bottom end plates to limit vertical tip losses.

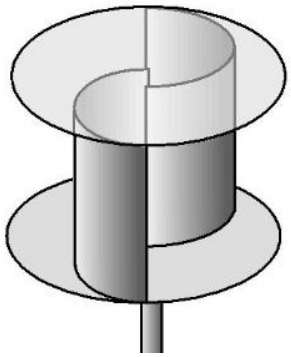


Figure 1. A basic Savonius wind turbine with end plates (top plate transparent).

The (mostly) drag-based Savonius is self-starting at low wind speeds, able to accept wind from any horizontal direction, and relatively simple to fabricate. It does, however, suffer from low power harvesting efficiency when compared to other wind turbine designs.

Perhaps the most important measure of wind turbine performance is the dimensionless *power coefficient* (C_P), defined as

$$C_P = \frac{P_{\text{turbine}}}{P_{\text{wind}}} = \frac{P_{\text{turbine}}}{\frac{1}{2}\rho V^3 A} = \frac{T\omega}{\frac{1}{2}\rho V^3 A}$$

where P_{turbine} = mechanical power produced by the turbine
P_{wind} = power available due to the wind's kinetic energy
ρ = density of air
V = freestream velocity of the wind approaching the turbine
A = capture area of the turbine
T = torque on the turbine shaft due to wind passing over the moving rotor
ω = turbine rotor angular velocity

The power coefficient represents the fraction of available wind power that is harvested by the turbine. For a given turbine geometry, dimensional analysis and experimentation have shown that the C_P is a function of the dimensionless *tip speed ratio* (TSR), $\omega R/V$, where R is the turbine rotor radius. In this problem, we are interested in accurately predicting the variation of average C_P with TSR and in identifying the maximum C_P for a given Savonius turbine geometry.

Simulation Software and Numerical Models

Both SWFS (2025 SP5.0) and Ansys (2025/R2) Fluent were used to solve the Reynolds-averaged Navier Stokes equations for the velocity and pressure fields surrounding a Savonius turbine operating at a constant rotational speed and to calculate the resultant torque on the rotor and the associated power coefficient. Prior studies by other researchers have shown that the

dynamic performance of simple Savonius turbines can be predicted accurately by two-dimensional (2D) numerical simulations [12,19,20].

The 2D Savonius rotor geometry is shown in Figure 2. The center of rotation is point O. The two blades are semicircles, each with radius r and diameter d. The overall diameter of the turbine rotor is D, and it depends on r and on the symmetric overlap distance S_0 as shown. For the simulations reported in this paper, $r = 0.25$ m, $D = 0.9$ m, and $S_0 = 0.1$ m. The blades are taken to be 2 mm thick. Shear and pressure forces on the moving blades (and the resultant torque and power) were calculated for a 1 m vertical blade height.

(We note that other cases involving different dimensions, blade shape modifications, and more than one Savonius turbine were investigated as part of our broader research, but in this paper, we focus on the single turbine case just described because it allows us to compare our numerical results with existing, high-quality experimental data.)

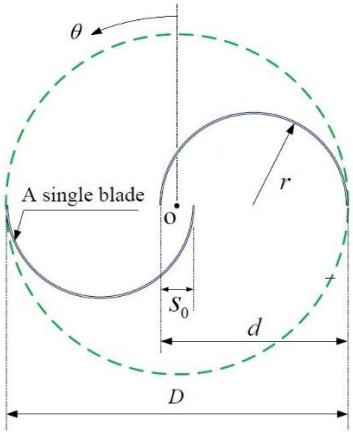


Figure 2. Savonius turbine rotor geometry (rotor orientation shown corresponds to $\theta = 90°$).

The computational domain for the simulations is shown in Figure 3. Wind (air; $\rho = 1.17$ kg/m^3; $\mu = 1.89 \times 10^{-5}$ N·s/m^2) enters the domain from the left at a specified uniform velocity (7 m/s; inlet boundary condition) and exits the right side of the domain after interacting with the turbine. The outlet is placed 20 rotor diameters from the center of rotation in order for a constant pressure outlet boundary condition to be physically reasonable. The inlet is placed 10 diameters upstream to avoid perturbations to the inlet flow from the presence of the turbine. These relative dimensions have been shown by other researchers to be appropriate for obtaining accurate solutions [21]. Rather than coupling the fluid-rotor dynamics, a simplified approach is adopted whereby the rotor kinematics is prescribed. The 2D cross section of the Savonius blades is surrounded by a rotating region [10] where the mesh rotates with the blades. A sliding interface is used between the edge of the rotating region and the remainder of the domain where the mesh is stationary. Within the rotating region, a no-slip condition is applied at the rotating blade surfaces. By setting the turbine rotational speed to different values, the torque imparted by the moving fluid on the rotor at each speed can be determined. Counterclockwise (CCW) rotor rotation is imposed and taken as positive, and CCW torque is also reported as positive.

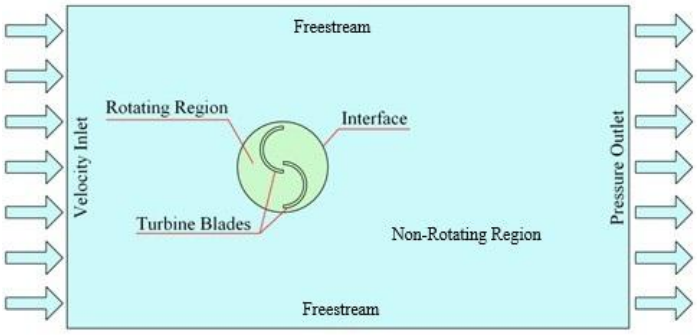


Figure 3. The 2D computational domain (not to scale).

Freestream boundary conditions are implemented at the remaining two (upper and lower) boundaries which are 10 diameters from the center of rotation [21]. At the wind and rotor velocities investigated, the flow is considered to be incompressible. Time-dependent simulations were conducted after an impulsive start and covered 10 complete rotations of the turbine rotor. Two models were created—one using SWFS and the other using Ansys Fluent.

SWFS Model

SWFS solves the governing partial differential equations for fluid flow using a SIMPLE-like pressure-based, finite volume (FV) method formulated on a Cartesian (rectangular) coordinate system [4]. Users can control some solution process parameters, but only one basic solution algorithm is available. The time (first-order) and spatial (second-order) derivative approximations are fixed. A modified version of the k-ε turbulence model that includes special wall functions and other enhancements is the only turbulence model included [22,23].

Rather than using a body-fitted approach, SWFS makes use of an immersed-body, structured, Cartesian mesh [4]. An initial rectangular mesh is first generated over the computational domain, which can include both solid and fluid regions. Cells that contain both solid and fluid materials are subsequently split into "partial cells" as defined by the intersection of the cell and the solid CAD geometry (see Figure 4a). These partial cells then receive special treatment in the solution algorithm to account for their shape and contents. Mesh refinement is accomplished using clustering functions or by cell splitting, where (in 2D) a rectangular cell is split into four smaller, geometrically similar rectangles (see Figure 4b). The user can control this refinement.

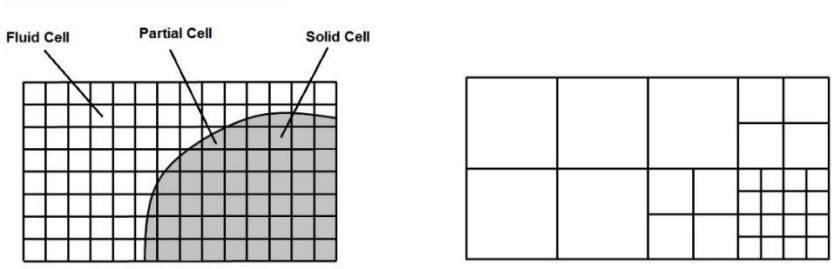
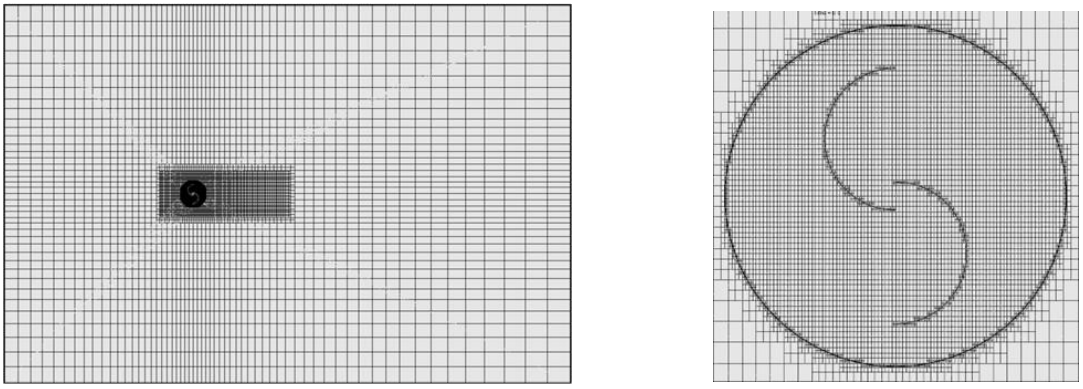


Figure 4. (a) Types of mesh cells in SWFS; (b) Cartesian mesh refinement in 2D.

SOLIDWORKS is, first and foremost, a CAD tool, so the Savonius geometry was created using the standard CAD features in SOLIDWORKS. Mesh and time step sensitivity studies were conducted, and it was found that a mesh density of 184,246 cells and a time step size corresponding to a 1° rotation of the turbine rotor produced solutions that did not change significantly with further spatial or temporal refinement. The SWFS computational meshes used are shown in Figure 5.



(a) (b)

Figure 5. SWFS mesh: (a) entire domain; (b) near the rotors and the rotating/non-rotating region interface.

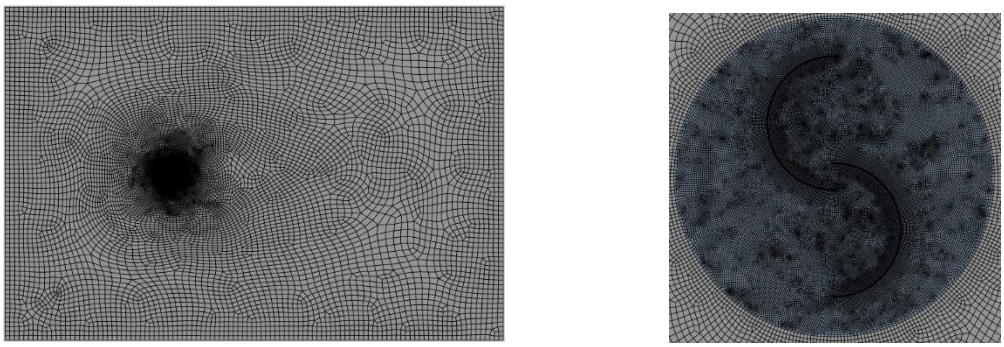
Ansys Fluent Model

Ansys Fluent also uses an FV approach to solve the governing mathematical equations, but it includes multiple options for different algorithms, different solvers, different spatial differencing schemes, different time differencing schemes, and different turbulence models [8]. This makes the software both exceedingly capable and complex to learn.

Unlike SWFS, Ansys Fluent favors a hybrid meshing approach consisting of unstructured polyhedral volumes surrounding body-fitted hexahedral or rectangular mesh (inflation) layers near solid surfaces. Meshes can be generated within the Fluent module or by using more general Ansys meshing tools. The user has an exceptional amount of flexibility in creating the overall mesh, which again makes the software simultaneously powerful and challenging to master.

Ansys is primarily an advanced engineering analysis tool and not a CAD program, but it does contain modules that allow the user to create 2D and 3D geometric models. For the problem analyzed here, the Savonius geometry was created using the DesignModeler tool.

Mesh and time step sensitivity studies were also conducted using Ansys Fluent, and it was found that a mesh density of 64,289 cells and a time step size corresponding to a 1° rotation of the turbine rotor produced solutions that did not change significantly with further spatial or temporal refinement. The Ansys Fluent computational meshes used are shown in Figure 6. Structured inflation layers were added around the blades to properly resolve the boundary layer. As with SWFS, the region surrounding the rotor was modeled as a rotating zone with a sliding mesh interface. The remainder of the domain was non-rotating.



(a) (b)

Figure 6. Ansys Fluent mesh: (a) entire domain; (b) near the rotors and the rotating/non-rotating region interface.

The Pressure-Implicit with Splitting of Operators (PISO) pressure-velocity coupling scheme was selected as the primary solution algorithm with second-order discretization in space and time. Multiple turbulence models were investigated (including k-ω SST) [24], but the k-ε model with enhanced wall treatment was ultimately chosen as it yielded the closest agreement with experimental data. (More on this later.)

Experimental Data

Results from extensively documented tests conducted by Sandia National Laboratories [25,26] were used for comparison with the numerical results. These tests were run on a physical prototype Savonius wind turbine fitted with instrumentation to measure windspeed, rotational speed, and rotor torque. The physical dimensions of this turbine are identical to those previously listed for the simulations reported in this paper. The results for the power coefficient (C_P) as a function of tip speed ratio (TSR) are shown in Figure 7 for a wind velocity of 7 m/s. Error bars represent the 95 percent confidence intervals for each data point.

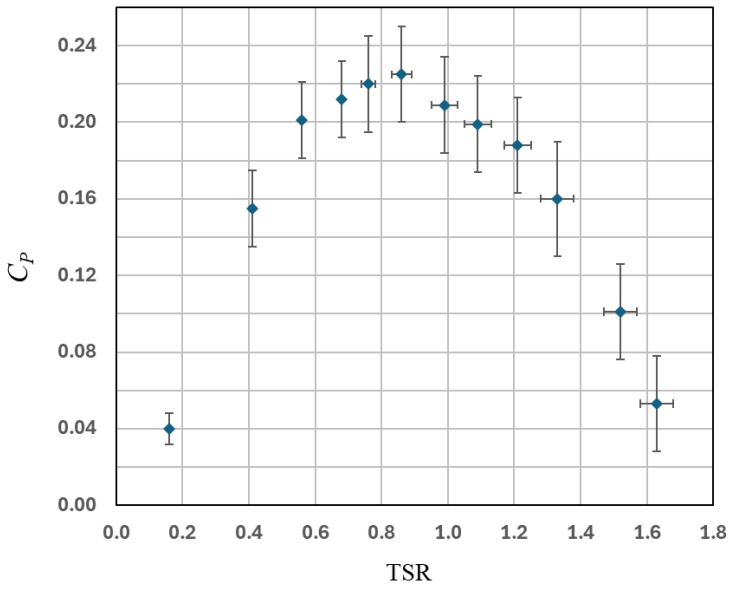


Figure 7. Experimental results from [26].

Numerical Results and Discussion

Numerical calculations were performed using a Dell Microsoft-Windows-based PC equipped with an Intel Core i7-8700 CPU running at 3.18 GHz and 32 GB of RAM. Run times for 10 revolutions of the rotor were approximately 1.3 hours for SWFS and 5.5 hours for Ansys Fluent.

Both SWFS and Ansys Fluent have the ability to calculate pressure and shear forces on surfaces using field solution results. Torques associated with those forces about the turbine shaft can also be determined. This, in turn, allows calculation of C_P. Figure 8 presents the C_P results of the SWFS and Ansys Fluent simulations together with the experimental results over the range of TSR values examined.

In the numerical simulations, the C_P values represent the average over the last two full rotations of the rotor, when periodically repeatable behavior has been established. Both SWFS and Ansys Fluent show good agreement with the experimental data up to a TSR of approximately 0.9, falling well within the uncertainty bounds shown. This point corresponds to the experimentally observed peak C_P of 0.22. For TSR values of 1.0 and greater, Ansys Fluent tracked the experimental results much more closely than SWFS. In fact, SWFS predicts a peak C_P of 0.235 at a TSR of 1.2, a clearly non-physical result. This error leads to an unrealistic expectation of higher performance. In addition, wind turbines generally include gearing and control systems that attempt to keep the turbine rotating near its optimal efficiency point. An error in predicting the TSR value associated with that peak efficiency could lead to a poor system design that tries to operate at the wrong rotational speed.

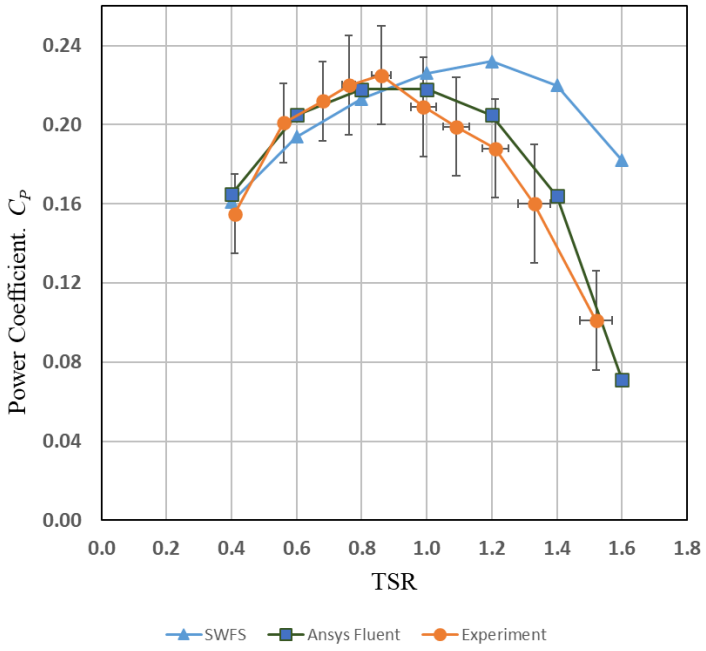
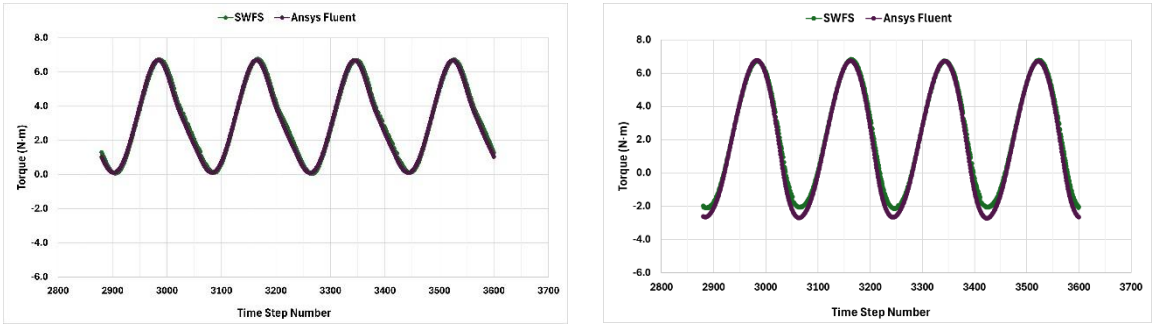


Figure 8. Numerical simulation and experimental results for power coefficient.

Plots of instantaneous torque versus time step number (Fig. 9) provide some insight into why SWFS is overestimating the peak C_P. As the turbine rotates, the aerodynamic forces on the blade surfaces change, yielding a cyclic variation in the resulting torque. This torque can be positive (CCW) during some parts of the rotational cycle and negative (clockwise; CW) during other parts. The particulars of the variation generally depend on the blade shape, the overlap distance, and the TSR. At a constant rotational speed, power is directly proportional to torque, so the times (and rotor positions) with the highest instantaneous torque generate the highest instantaneous power. Plots are shown for TSR = 0.8, where both SWFS and Ansys Fluent match the experimental C_P data well, and for TSR = 1.2, where SWFS produces an unreasonably high time-averaged C_P.



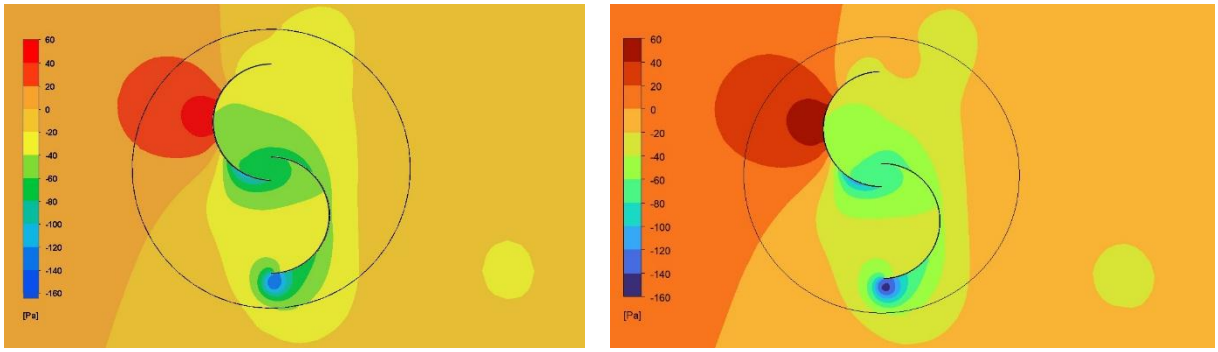
(a) TSR = 0.8 (b) TSR = 1.2

Figure 9. Cyclic turbine rotor torque.

It can be seen in Fig. 9a (TSR = 0.8) that both Ansys Fluent and SWFS produce relatively smooth torque curves that track closely and, therefore, yield very similar average torque values. In Fig. 9b (TSR = 1.2), both torque curves are once again relatively smooth, but the peak negative (CW) torque calculated by SWFS is consistently smaller in magnitude. This results in a 16% higher average torque for SWFS over the two full rotations shown.

The underprediction of negative torque during part of the rotational cycle by SWFS is the primary driver for the overestimation of C_P. For our Savonius geometry and TSRs, the maximum negative torque occurs when the turbine blades have rotated just over 90° CCW from the position shown in Figure 2. To gain some insight into why this numerical underprediction occurred, the relative pressure distribution near that rotor position was examined and is shown in Figure 10 for both SWFS and Ansys Fluent. In these contour plots, the black circles represent the interface between the rotating and non-rotating portions of the domain (not a physical boundary). Also, because the plots were generated with two different programs, the color scales are very similar, but it was not possible to make them identical.

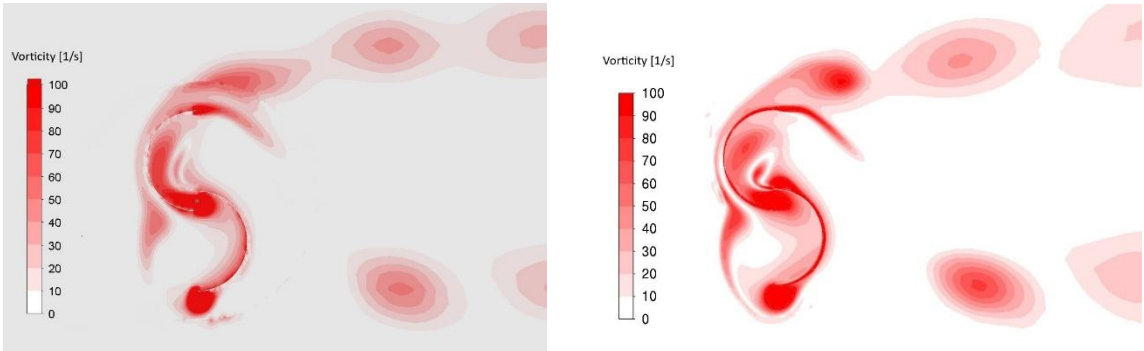
The *instantaneous* torque on the two-bladed rotor in the position shown in Figure 10 is negative (CW) in both simulations and is 23% smaller in SWFS than in Ansys Fluent (−2.04 N·m versus −2.66 N·m). Still, the two calculated pressure fields appear quite similar. The stagnation point position on the upper (returning) blade is essentially the same in both plots, as is the pressure value there. Both simulations also show vortex formation at the bottom of the lower (advancing) blade, but the Ansys Fluent vortex is a bit more intense.



(a) SWFS (b) Ansys Fluent

Figure 10. Relative pressure contours at TSR = 1.2 for (a) SWFS and (b) Ansys Fluent.

For Savonius wind turbines, the typical drop in C_P with increasing TSR once a peak has been achieved is generally understood to be the result of separation along the blades and/or stronger tip vortex formation, both of which can negatively affect blade-wake interactions relative to producing positive torque [27]. To gain additional insight into differences in the simulations, plots of vorticity were also generated and are shown in Figure 11. Here, the greater intensity of the advancing and returning blade vortices being generated and shed in the Ansys Fluent results is more clear. Additional analyses, including detailed comparisons of numerical values of velocity, pressure, shear stress, vorticity, and turbulent kinetic energy were conducted, but rather than revealing dramatic differences in predicted flow behavior, what was observed were multiple relatively modest differences that combined to reduce the negative torque on the returning blade. Ultimately, SWFS is able to capture the quantitative variation in C_P up to TSR = 0.9, and it also predicts an eventual reduction in C_P at higher TSR values, but it does not accurately calculate the peak C_P or the corresponding TSR.



(a) SWFS (b) Ansys Fluent

Figure 11. Vorticity contours at TSR = 1.2 for (a) SWFS and (b) Ansys Fluent, showing stronger vortex intensity predicted by Fluent.

There are several differences between the SWFS and Ansys Fluent numerical models for this problem, but it is postulated that the single turbulence model (and wall treatment) in SWFS is contributing to this behavior. This is consistent with results reported in [14] for other rotational flows; however, we have yet to identify definitive evidence to confirm this hypothesis.

It should be noted that prior studies by other researchers have generally reported that the most accurate, two-equation, turbulence model for Reynolds-averaged wind turbine simulations (including the Savonius) is the k-ω SST model [27]. However, in this study, the k-ε model with enhanced wall treatment outperformed the k-ω SST model in Ansys Fluent simulations. This was true even when care was taken to limit y^+ (the dimensionless wall distance) to less than 2 to ensure that the first mesh point was located well within the viscous sublayer. This is being investigated further, but not in this paper.

Ease of Use by Students

The author has significant experience teaching undergraduate students to use FEA for displacement and stress analysis of static structures within a computer-aided design and simulation course. But because of the size of the liberal arts "core" at our institution (taken by all majors, including engineering) and the inclusion of other engineering content that the faculty agrees is critical, we have not made room in our curriculum for similar CFD instruction within our courses. (Students are required to create a finite-difference-based code of their own as a project in our Heat Transfer course, but we have not covered CFD nor introduced the use of a commercial code to numerically simulate fluid flow in that course or any other—at least not yet.)

What this means is that the observations in this section constitute a qualitative case study based on the use of SWFS and Ansys Fluent by a small number of students (three) on research projects rather than within traditional courses. In each case, the author spent several hours up front explaining the theory behind CFD and the general (non-software-specific) procedure for conducting a numerical simulation. He also provided written materials for the students to review. The observations reported here are based on the author's discussions and correspondence with the students, as well as entries and reflections in their research journals and the times reported to complete the various steps in their analyses. Student quotes come from written materials or recollections and notes from one-on-one discussions. We have attempted to be quantitative, where possible, but many of our observations are necessarily qualitative. Also, because of the specificity of the scenario and the small number of students, the results should be viewed as illustrative rather than generalizable.

To examine ease of use, we have broken down the CFD model development and solution process into five steps that were executed on the Savonius problem:

1. *Geometry* – Using CAD tools, create a model of the Savonius blade geometry and the associated computational domain boundaries.
2. *Meshing* – Create an appropriate 2D mesh using the tools that are available in the software.
3. *Properties, Boundary Conditions, and Regions* – Specify the fluid properties that will be used and set up the boundary conditions (BCs) that define the problem (inlet, outlet, freestream, fluid-solid interface) as well as the zone that accounts for rotation.
4. *Physics Models and Solution Methods/Controls* – Select from among the available physics models and specify the solution methods and controls to be used for the problem.
5. *Post-Processing* – Extract, format, and interpret both numerical and graphical results.

Three factors were examined for each step:

- *Task Complexity* – The overall complexity involved in completing the step (simple, moderate, complex). This includes identifying the elements that need to be set or specified, navigating the user interface to do so, and reading documentation or watching videos to inform the user's choices.
- *Separate Activities* – A count of the main activities involved in the step. Rather than counting each mouse click or value entered, an activity represents grouped actions that complete a significant aspect of the step (*e.g.*, creating a 2D CAD sketch of the turbine blades, specifying a rotating region, imposing a specific boundary condition, or selecting a particular solution algorithm and its related settings).
- *Time* – The estimated total time to (1) read the associated software documentation and watch videos to gain the minimum level of understanding needed for the Savonius problem, and (2) execute the step. This does not include iteration to refine the solution in any way—only the effort required to obtain a first cut.

The results are summarized in Table 1.

An additional comment about the lengths of time listed in the table is in order. These durations indicate that a functional, Savonius, 2D, numerical model could be ready for execution in under 2 hours, where a similar Ansys Fluent model would take most of a workday. The author solicited time estimates from the students later in their projects, asking them to think back to when they put their initial model together. The reality is that, when student researchers were developing their first models with the software, longer times than these were observed and recorded. A SWFS model was up and running in about a day. A complete Ansys Fluent model took on the order of a work week. This reflects the reality of research and of learning a new tool. It also reflects how much their advisor allowed them to struggle and how quickly they sought assistance. Blind alleys are sometimes followed, and processes that become routine are, at first, confusing. What can be said is that the *ratio* of time spent with SWFS versus with Ansys Fluent was close to that listed in the table, though the actual times were longer.

Table 1. SWFS vs. Ansys Fluent: Ease of Use by Students with Prior SOLIDWORKS CAD Experience.

	SWFS			Ansys Fluent		
Step	**Task Complexity**	**Separate Activities**	**Time**	**Task Complexity**	**Separate Activities**	**Time**
Geometry	Simple	2	15 min.**	Simple	2	1 hr.**
Meshing	Moderate	2	1 hr.	Complex	9	4 hrs.
Properties, BCs, and Regions	Simple	4	15 min.	Moderate	7	1 hr.
Physics Models and Solution Methods/Controls	Simple	3	15 min.	Complex	7	2 hrs.
Post-Processing	Moderate	varies*	varies*	Moderate	varies*	varies*

* The number of separate activities and time for *Post-Processing* was not quantified as it depends completely on the number of artifacts created and examined. The *Post-Processing* ease of use was assessed to be roughly equivalent between the two codes.

** The students in this study were already familiar with SOLIDWORKS for CAD. This is the primary reason for the time difference listed under *Geometry*.

Student comments on the various steps provide additional insight into the analysis. With regard to geometry creation, prior familiarity with SOLIDWORKS for basic CAD was understandably helpful. One student put it this way:

> "Because I already knew how to use SOLIDWORKS, making the turbine blades was simple. In Ansys [DesignModeler], I mostly just had to figure out what it took to do the same thing. The interface was less intuitive, but example videos I found online helped a lot."

In terms of meshing, the use of immersed body, structured, Cartesian meshing in SWFS greatly simplifies the mesh generation process for the student. Decisions still need to be made about breaking up the domain to accommodate different mesh densities and clustering in different regions, including rotating regions. But choices about element type, face meshing, and wrapping (inflation) are eliminated. In contrast, meshing in Ansys Fluent is more flexible and more complex. The user has two options—use the general Ansys meshing tool (which is shared with other Ansys modules) or use the meshing capabilities built directly into Ansys Fluent. The former option was exercised in this study. (The user could also import a third-party grid directly, but we do not consider that option here.) As a result of this complexity, mesh generation in Ansys Fluent required significantly more time and effort than with SWFS. According to one student:

> "Creating a mesh with SOLIDWORKS [Flow Simulation] was pretty easy. The settings mostly made sense and were easy to figure out. [Ansys] Fluent was different. I didn't really understand the process at all at first, and it took me a while to get a mesh that worked. What helped the most was a video I found online that was pretty close to what I needed."

Specification of fluid properties is uncomplicated in both codes, but the choice and implementation of physics models and solution methods/controls differs in complexity. SWFS is based around one set of algorithms and one turbulence model. The main choices are whether the simulation is to be internal or external, steady-state or transient, and whether there is any reason to anticipate only laminar flow conditions. In contrast, Ansys Fluent has many options for physics models and solution methods/controls to choose from. One important choice is the turbulence model and associated near-wall treatment functions. There are eight separate turbulence models available, with options for each. Guidance on the strengths and weaknesses of each model is available in the Fluent documentation and from other sources. For a new user who is just learning the software and is not too concerned about solution accuracy just yet, specification of the widely applied standard k-ε model is not difficult (though it may not be a good selection). Another choice is the solution method. Ansys Fluent includes both pressure-based and density-based solvers for steady and transient flow. Within the pressure-based category, algorithm options include SIMPLE, SIMPLEC, PISO, and Coupled [10]. Under each option, the user selects the spatial and temporal discretization formulations, as well as other specialized options. General guidance regarding the selection of an appropriate solution method is available in the Fluent documentation and from other sources, but the sheer number of options can be daunting to the new user. A student described their experience with this step in this way:

> "When you told me exactly what to pick in [Ansys] Fluent, that was simple, but I wanted to know why. I spent a lot of time trying to figure out what each option did. The more I read, the more complicated everything seemed. It was hard to tell what applied to my problem. By the end, I kind of knew what I was doing, but I can't say I'm confident."

Both programs contain ways to save and visualize the results at different time levels in a transient simulation. Students found the post-processing interfaces associated with SWFS and Ansys Fluent to be comparable and generally easy to use. For example:

> "Once SOLIDWORKS [Flow Simulation] was done, I was able to create color plots that let me see what was happening. I did watch a few short YouTube videos for help. It wasn't hard to pick up. Ansys [CFD-Post] might have been a little harder to get started with, but it was pretty much the same, once I got the hang of it."

Two more-general student comments were not specific to particular steps but struck the author as particularly relevant to the larger picture concerning both ease of use and effective use of the software:

> "I liked that I could just keep going with SOLIDWORKS to model fluid flow and get an answer. When I couldn't get it to match the [experimental] data, that was frustrating. I learned a lot more when I changed different settings in [Ansys] Fluent to try to get closer to what was measured."

> "Trying to solve the problem first with SOLIDWORKS [Flow Simulation] was helpful even if the final answers weren't great. When I moved over to [Ansys] Fluent, I already knew the basic steps. I just needed to learn how to do them another way. I think that would have been harder without starting with SOLIDWORKS [Flow Simulation]."

These comments led the author to examine and consider more broadly CFD instruction at the undergraduate level.

Framework for Undergraduate CFD

Student experiences on research projects showed that initial familiarity with SOLIDWORKS helped significantly with the geometry creation part of the CFD solution process, as expected. In addition, the structured mesh generation method and the intentional limitation of options for the user in SWFS significantly reduced the time needed to create a numerical model and run it successfully. This was most evident during mesh generation and in the selection of physics models and solution methods. But two perspectives on this emerged:

- Getting a model up and running relatively quickly using SWFS resulted in less initial student frustration and significant engagement because they were using and building on what they knew, could see rapid progress, didn't have to stop to decode complex setup procedures, and didn't have to research so many options to make decisions during setup.
- Using a more advanced tool (Ansys Fluent) was sometimes frustrating due to more complex setup procedures and the sheer number of options and settings available. However, it forced the students to consider their choices within the CFD model development process and led them to dig in more deeply to understand what each option did and how it might affect the solution. This meant stopping to review written materials and/or searching for helpful videos, which eliminated the immediate gratification of moving forward quickly. Ultimately, though, it led to greater understanding of what it takes to generate an accurate CFD solution.

These two perspectives are not at odds. In fact, both are helpful in understanding what an effective framework for CFD instruction at the undergraduate level could look like. Like any complex concept, CFD is challenging to master, particularly for undergraduates. Universities have taken different approaches for including CFD in their curricula, which have been categorized in [28] as

- *CFD Light* – An instructor of an introductory or intermediate fluid mechanics course uses pre-computed CFD simulations to reinforce fundamental fluid flow concepts.
- *CFD Moderate* – CFD modules are used to augment course material by substituting the module into an existing lecture-form course or into the accompanying lab unit of the course.
- *CFD Heavy* - CFD is taught as a stand-alone course.

This is rational way to perform this classification, but student comments reported in our study suggest that two well-established and related educational theories could additionally inform a framework for CFD instruction in an undergraduate mechanical engineering curriculum:

- *Cognitive Load Theory* – The idea that working memory can only handle a certain amount of information at one time. When that limit is exceeded, information processing breaks down, and learning suffers [15,16].
- *The Spiral Curriculum* – An educational approach where key topics are revisited multiple times throughout a student's education, with each encounter building on previous understanding and introducing greater complexity [17,18].

The spiral curriculum approach can be used to avoid cognitive overload. In terms of CFD, this means that the instructor should begin with simple versions of key concepts that can be grouped into cognitively manageable "chunks." These concepts are then periodically revisited with increasing levels of complexity so that students can build, draw upon, and grow effective schemas associated with CFD and its effective application. This can happen in a single course, but ideally, it happens in several courses over an extended period.

This approach also has relevance when the use of commercial codes is included in CFD instruction, and it suggests (1) starting with a code that limits user choices and avoids many of the complexities of mesh generation at first, and then (2) gradually moving on to a more complex code that expands user choices and corresponding simulation capabilities. For programs that already use SOLIDWORKS for CAD instruction, SWFS is a natural choice for the former. It has the additional benefit of leveraging what students have already learned and extending it into new territory. In this regard, one additional consideration is relevant. Students often demonstrate more engagement with the material when they perceive that it is applicable to "real-world" problems that have meaning for them. Often, this translates into consideration of scenarios involving complex geometries. By using a product that directly links 3D CAD with CFD, new users can proceed to examine realistic geometries with greater ease and less frustration, increasing student interest, though possibly at the expense of high accuracy. After this "honeymoon period," instruction would circle back around to identify shortcomings and introduce more complex models and software to address them. The focus would shift to

interpretation of results, troubleshooting, and more advanced modeling, opening the door for even more interesting problems.

Alternatively, an instructor could just use an advanced CFD code up front and prescribe or provide certain settings for student use. A drawback with this approach is that the myriad of settings and user choices is still visible to the new user, and this can create a kind of "black box" feel for the student if this still visible complexity remains a mystery. It can also provide opportunities for the new user to get lost during problem set up without careful guidance.

There is another balance to be considered in undergraduate CFD instruction, namely theory versus application. Instructors may reasonably disagree on what an appropriate balance should look like, as employers and graduate schools can have very different expectations in this area. Nevertheless, in keeping with the philosophy expressed by Tu *et al.* [29], we propose that an effective undergraduate curriculum should ideally include a healthy dose of each. There are also considerations about how much computer programming should be required. Our position is that having the students write their own small-scale code creates opportunities to learn how CFD algorithms and numerical calculations actually work that cannot be addressed any other way. This knowledge will ultimately make them better code users. But the main focus for undergraduate mechanical engineering students should not be on writing their own CFD code. That is best left to graduate-level courses. (We acknowledge that this paragraph reflects more opinion than fact, and that we have bias in this regard. We, therefore, identify it as such and offer it here for the reader's informed consideration.)

A curriculum based on the elements discussed above would include the following:

- *First-Year*: A required course that introduces the student to a commercial CAD program and focuses on 3D modeling as well as creation of design drawings for fabrication. Ideally, the CAD program also has structural FEA and CFD capabilities, though they are not covered in this course. Here, the students develop strong CAD skills that underpin the subsequent introduction of CFD in later courses.
- *Sophomore/Junior Years*: Include modules within existing courses in fluid mechanics and heat transfer that present the basics of CFD theory and application and also require students to use the CFD capabilities within the already familiar CAD program on small but relevant projects. These modules introduce the students to CFD at a level that avoids cognitive overload and enables them to understand the essentials of the field.
- *Senior Year*: An elective course dedicated to CFD. Expand on CFD theory and transition from the CAD/CFD package to more advanced software that allows students to explore more complex concepts. Re-visit ideas from prior modules and link them to new concepts and assignments. Include simple FV-based programming, but don't place a major focus on code development. Rather, use programming as a way to understand concepts such as approximation of derivatives, time-stepping, and stability in depth. (Transient heat conduction is a good candidate for this.)

As discussed in [28], some institutions integrate CFD into existing courses in their undergraduate curricula [30,31], while others include a separate CFD course (often as an elective) [32,33]. But

the linked, spiral approach described above is often missing, as is the concept of including the institution's CAD software as a common thread throughout the sequence.

Conclusions

Earlier in this paper, two questions were posed, the first of which considered the accuracy of numerical solutions generated by different types of commercial CFD software and the second of which dealt with ease of use by students. Based on the scenario investigated in this study and the student feedback received, conclusions can be drawn related to those questions.

Both SWFS and Ansys Fluent were able to accurately calculate Savonius wind turbine power coefficients for lower TSR values up to the experimentally determined maximum C_P. Past that point, SWFS predicted higher non-physical C_P values that were not supported by experimental data, while Ansys Fluent did not. Examination of the numerically predicted flow fields in this regime revealed relatively small but significant differences in blade pressure distribution and vortex intensity that lead to higher predicted turbine torque and power by SWFS. Mesh refinement and time step reduction sensitivity studies were unable to rectify this shortcoming. Ours is a very specific scenario, but because fluid flow through different types of rotating machinery (*e.g.*, turbines, pumps, and compressors) involves these kinds of flow phenomena, our study suggests that SWFS should be used with care when applied to such problems. This is particularly the case for undergraduate student research projects that require accurate results to support peer-reviewed publication.

Examination of student experiences during research projects revealed that SWFS was significantly easier for the students to learn and use than Ansys Fluent. Student observations motivated the author to formulate a framework for including CFD in undergraduate curricula. The suggested approach is not without challenges. It can mean that multiple instructors (potentially including adjunct or affiliate faculty) need to be familiar with the engineering program's CAD package and its CFD capabilities. It requires coordination of assignments among different courses and instructors. It also contains a course dedicated to CFD, and we have already mentioned that some programs (ours included) struggle to make room for a dedicated CFD course in their curriculum, even as an elective. Nevertheless, we believe this approach has the potential to encourage deeper learning and to better prepare students to effectively apply CFD upon graduation.

We close with a final observation. Faculty who are mentoring undergraduate researchers or teaching CFD content in courses should anticipate that advanced CFD tools demand substantial time for guided learning, particularly in mesh generation as well as solution algorithm and physics model selection. Beginning the process with a CAD-embedded tool can help, as can clear initial guidance on critical software settings. But in the end, allocating structured time for exploration, troubleshooting, interpretation of results, and reflection can transform student frustration into productive learning. In this way, CFD becomes not only a computational technique, but a vehicle for developing engineering judgment, an outcome that aligns strongly with both undergraduate research and professional preparation.

References

[1] M. May, "Top Skills Employers Look for In Mechanical Engineers," https://solidprofessor.com/blog/top-skills-employers-look-for-in-mechanical-engineers/, posted April 8, 2023. [Accessed December 23, 2025].

[2] N. Smith and J. Davis, "Connecting Theory and Software: Experience with an Undergraduate Finite Element Course," in *Proceedings of the 2015 ASEE Annual Conference and Exposition*, Seattle, WA, USA, June 14-17, 2015, Washington, D.C.: American Society for Engineering Education, 2015.

[3] F. Moukalled, L. Mangani, and M. Darwish, *The Finite Volume Method in Computational Fluid Dynamics: An Advanced Introduction with OpenFOAM® and Matlab*, New York, NY: Springer, 2016.

[4] Dassault Systémes, "Technical Reference: SOLIDWORKS Flow Simulation 2025," Boston, MA, 2025.

[5] A. Sobachkin, and G. Dumnov, "Numerical Basis of CAD-Embedded CFD," Wilsonville, OR: Mentor Graphics Corporation, 2015.

[6] R. Mackay, "10 Reasons Why Every Design Engineer Needs SOLIDWORKS Flow Simulation," https://www.javelin-tech.com/blog/2015/09/every-design-engineer-needs-solidworks-flow-simulation/. [Accessed January 2, 2026].

[7] Autodesk, "Autodesk Fusion: More than CAD, it's the future of design and manufacturing," https://www.autodesk.com/products/fusion-360/overview. [Accessed January 3, 2026].

[8] Autodesk, "Autodesk CFD: Simulation software for engineering complex liquid, gas, and air systems" https://www.autodesk.com/products/cfd/overview. [Accessed January 3, 2026].

[9] Ansys, "Ansys: Part of Synopsis," https://www.ansys.com/, [Accessed January 3, 2026].

[10] Ansys, "Ansys 2025/R2 Fluent Theory Guide," Canonsburg, PA, 2025.

[11] Z. Driss, O. Mlayeh, D. Driss, M. Maaloul, and M. Abid, "Numerical simulation and experimental validation of the turbulent flow around a small incurved Savonius wind rotor," *Energy*, vol. 74, pp. 506-817, 2014.

[12] W. Tian, Z. Mao, B. Zhang, and Y. Li, "Shape optimization of a Savonius wind rotor with different convex and concave sides," *Renewable Energy*, vol. 117, pp. 287-299, 2018.

[13] S. Bell, A. Blair, L. Wagner, V. Zou, A. Buendia, and F. Ashrafzadeh, "Survey of Computational Fluid Dynamics for Rotational Purposes," in *Proceedings of the ASME 2019 International Mechanical Engineering Congress and Exposition,* Salt Lake City, UT, USA, November 11-14, 2019, New York, NY: American Society of Mechanical Engineers, 2019.

[14] G. Talamantes and B. Maicke, "Evaluation of CFD Codes for Swirl-Driven Combustors," in *Proceedings of the 46th AIAA Fluid Dynamics Conference 2016*, Washington, DC, June 13-17, 2016, Reston, VA: American Institute of Aeronautics and Astronautics, 2016, pp. 4229-4239.

[15] J. Sweller, "Cognitive load during problem solving: Effects on learning," *Cognitive Science*, vol. 12, no. 2, pp. 257-285, 1988.

[16] Centre for Education Statistics and Evaluation, "Cognitive load theory: Research that teachers really need to understand," New South Wales Department of Education, Sydney, Australia, 2017

[17] J. S. Bruner, *The Process of Education.* Cambridge, MA: Harvard University Press, 1960.
[18] H. Johnston, "The Spiral Curriculum," Education Partnerships, Inc., 2012.
[19] K. Kacprzak, G. Liskiewicz, and K. Sobczak, "Numerical investigation of conventional and modified Savonius wind turbines," *Renewable Energy*, vol. 60, no. 4, pp. 578-585, 2013.
[20] M. Shaheen, M. El-Sayed, and S. Abdallah, "Numerical study of wo-bucket Savonius wind turbine cluster," *J. Wind Eng. Ind. Aerodyn.*, vol. 137, pp. 78-89, 2015.
[21] B. Zhang, B. Song, Z. Mao, W. Tian, and B. Li, "A Novel Parametric Modeling Method and Optimal Design for Savonius Wind Turbines," *Energies*, vol. 10. no. 3, 2017.
[22] Hawk Ridge Systems, "Enhanced Turbulence Modeling in SOLIDWORKS Flow Simulation," Mountainview, CA, 2015.
[23] C. Lam and K. Bremhorst, "Modified Form of Model for Predicting Wall Turbulence," *ASME Journal of Fluids Engineering*, vol. 103, pp. 456-460, 1981.
[24] D. Wilcox, "Formulation of the *k*-ω Turbulence Model Revisited," *AIAA Journal*, vol. 46, no. 11, 2008.
[25] R. Sheldahl, L. Feltz, and B. Blackwell, "Wind tunnel performance data for two-and three-bucket Savonius rotors*," J. Energy*, vol. 2, pp. 160–164, 1978.
[26] B. Blackwell, R. Sheldahl, L. Feltz, "Wind tunnel performance data for two-and three-bucket Savonius rotors*,"* Report SAND76-0131, Sandia National Laboratories, Albuquerque, NM, 1977.
[27] A. Dewan, S. Tomar, A. Bishnoi, and P. Singh, "Computational fluid dynamics and turbulence modelling in various blades of Savonius turbines for wind and hydro energy: Progress and perspectives," *Ocean Engineering*, vol. 283, 2023.
[28] J. Eldredge, I. Senocak, P. Dawson, J. Canino, W. Liou, R. LeBeau, and D. Hitt, "A best practices guide to CFD education in the undergraduate curriculum," *Int. J. Aerodynamics*, vol. 4, nos. 3/4, 2014.
[29] J. Tu, G. Yeoh, C. Liu, and Y. Tao, *Computational Fluid Dynamics: A Practical Approach*, Cambridge, MA: Butterworth-Heineman, 2024.
[30] S. Wismer, L. Dosse, and M. Berry, "Integration of Computational Fluid Dynamics into an Introductory Fluid Mechanics Course," in *Proceedings of the 7th Thermal and Fluids Engineering Conference*, Las Vegas, NV, May 16-18, 2022. Danbury, CT: American Society of Thermal and Fluids Engineers, 2022.
[31] N. Fumo, "Introduction of Computational Fluids Dynamics om a Thermal-Fluids Laboratory," *Computers in Education Journal*, vol. 8, no. 3, 2017.
[32] M. Sarimurat, "Integrating Theory and Practice: A CFD Education Approach," in *Proceedings of the 2024 ASEE Annual Conference and Exposition*, Portland, OR, USA, June 23-26, 2024. Washington, D.C.: American Society for Engineering Education, 2024.
[33] X. Li and S. Cheung, A learning-centered computational fluid dynamics course for undergraduate engineering students," *International Journal of Mechanical Engineering Education*, vol. 53, no. 2, pp. 256-276, 2024.