The Journal of Open Source Education

Aero Python: classical aerodynamics of potential flow using Python

Lorena A. Barba[1] **and Olivier Mesnard**[1]

1 The George Washington University

DOI: 10.21105/jose.00043

Software

- Review
- Repository
- Archive

Submitted: 29 January 2019
Published: 29 January 2019

Summary

The **AeroPython** learning module presents a computational approach to an engineering course in classical aerodynamics. The audience of such a course would be advanced undergraduate students in mechanical or aerospace engineering (or master's students), with a fluid mechanics pre-requisite. Classical aerodynamics is a broad theoretical subject—this learning module focuses on one topic: the use of potential flow for aerodynamic analysis, via the panel method.

AeroPython is a collection of Jupyter notebooks: one "lesson zero" introduction to Python and NumPy arrays; 11 lessons in potential flow using Python; three student assignments involving a coding mini-project; and one extra notebook with an exercise deriving the panel-method equations.

The list of lessons is:

- *Python crash course*: quick introduction to Python, NumPy arrays and plotting with Matplotlib.
- *Source & sink*: introduction to potential-flow theory; computing and plotting a source-sink pair.
- *Source & sink in a freestream*: adds a freestream to a source to get a Rankine half-body; then adds a freestream to a source-sink pair to get a Rankine oval; introduces Python functions.
- *Doublet*: develops a doublet singularity from a source-sink pair, at the limit of zero distance; adds a freestream to get flow around a cylinder.
- *Assignment 1: source distribution on an airfoil.*
- *Vortex*: a potential vortex, a vortex and sink; idea of irrotational flow.
- *Infinite row of vortices*: superposition of many vortices to represent a vortex sheet.
- *Lift on a cylinder*: superposition of a doublet, a freestream, and a vortex; computing lift and drag.
- *Assignment 2: the Joukowski transformation.*
- *Method of images*: source near a plane wall; vortex near a wall; vortex pair near a wall; doublet near a wall parallel to a uniform flow. Introduces Python classes.
- *Source sheet*: a finite row of sources, then an infinite row of sources along one line. Introduces SciPy for integration.
- *Flow over a cylinder with source panels*: calculates the source-strength distribution that can produce potential flow around a circular cylinder.
- *Source panel method*: solves for the source-sheet strengths to get flow around a NACA0012 airfoil.
- *Vortex-source panel method*: start with the source panel method of the previous lesson, and add circulation to get a lift force. Introduces the idea of the Kutta condition.

- *Exercise: Derivation of the vortex-source panel method.*
- *Assignment: 2D multi-component airfoil.*

The design of the lessons follows the pattern of the **CFD Python** collection (Barba & Forsyth, 2018), with step-by-step, incrementally worked-out examples, leading to a more complex computational solution. In this case, the final product is a panel-method solution of potential flow around a lifting airfoil.

Statement of need

Classical aerodynamics based on potential theory can be an arid subject when presented in the traditional "pen-and-paper" approach. It is a fact that the mathematical framework of potential flow was the only tractable way to apply theoretical calculations in aeronautics through all the early years of aviation, including the development of commercial aircraft into the 1980s and later. Yet, the only way to exercise the power of potential-flow aerodynamics is through numerical computation. Without computing, the student can explore only the simplest fundamental solutions of the potential equation: point sinks and sources, point vortex, doublet, uniform flow.

The essential tool for applying this theoretical framework to aerodynamics is the panel method, which obtains the strength of a distribution of singularities on a body that makes the body a closed streamline. The addition of vortex singularities to satisfy a Kutta condition allows treating lifting bodies (like airfoils). The AeroPython series begins with simple point-singularity solutions of the potential equation, and applies the principle of superposition to show how to obtain streamline patterns corresponding to flow around objects. Around the half-way point, the module presents the learner with the fundamental relationship between circulation (via a point vortex) and the production of a lift force. Using a distribution of many point singularities on an airfoil, finally, the module shows how we can obtain pressure distributions, and the lift around an airfoil. With this foundation, the student is ready to apply the panel method in authentic engineering situations.

Exercising the power of superposition—the hallmark of the linear potential equation—via numerical computing with Python, and immediately visualizing the resulting flow patterns, the student begins to quickly interpret and imagine the real applications of this theory. At the end of the course, the student is ready to use panel methods for aerodynamic design, and knowingly use other software packages such as XFOIL (http://web.mit.edu/drela/Public/web/xfoil/) (Drela, 1989) and XFLR5 (http://www.xflr5.com).

Pedagogy and instructional design

AeroPython adopts the instructional design patterns used in the **CFD Python** learning module (Barba & Forsyth, 2018). It breaks down a rather complex numerical problem—in this case, the panel method for lifting bodies—into small steps, it chunks these steps together logically, and adds narrative and context. The pedagogical strategy relies on the worked-example effect (Sweller, 2006, Chen, Kalyuga, & Sweller (2015)) and sub-goals (Catrambone, 1998, Margulieux, Catrambone, & Guzdial (2016)). In the classroom, students are guided to re-write the worked examples in their own Jupyter notebooks, and annotate with their own reflections during class discussions. These discussions may take the form of short introductions to a new section of the course, or a debrief after completing a sub-goal. Classroom activity may also be peppered with "pair-share" exercises, and individual inspection of student work by the instructor. Some assigned readings can also be selected; see some examples in the syllabus for a course at the George Washington University using these lessons (Barba, n.d.). Barba has taught the semester-long course there four times since 2014, and Mesnard is currently teaching its fifth iteration.

References

Barba, Lorena A. (n.d.). MAE-6226: Aerodynamics course syllabus. figshare, doi:10.6084/m9.figshare.4584328.v1.

Barba, Lorena A., & Forsyth, G. F. (2018). CFD Python: The 12 steps to Navier-Stokes equations. *Journal of Open Source Education*, *1*(9), 21. doi:10.21105/jose.00021

Catrambone, R. (1998). The subgoal learning model: Creating better examples so that students can solve novel problems. *Journal of Experimental Psychology: General*, *127*(4), 355–376.

Chen, O., Kalyuga, S., & Sweller, J. (2015). The worked example effect, the generation effect, and element interactivity. *Journal of Educational Psychology*, *107*(3), 689. doi:10.1037/edu0000018

Drela, M. (1989). XFOIL: An analysis and design system for low Reynolds number airfoils. In T. Mueller (Ed.), *Low reynolds number aerodynamics*, Lecture notes in engineering (Vol. 54, pp. 1–12). Berlin, Heidelberg: Springer. doi:10.1007/978-3-642-84010-4_1

Margulieux, L. E., Catrambone, R., & Guzdial, M. (2016). Employing subgoals in computer programming education. *Computer Science Education*, *26*(1), 44–67.

Sweller, J. (2006). The worked example effect and human cognition. *Learning and instruction*, *16*(2), 165–169. doi:10.1016/j.learninstruc.2006.02.005