AIAA SciTech Forum
January 3-7, 2022, San Diego, CA & Virtual
AIAA SCITECH 2022 Forum

10.2514/6.2022-1350

Life in the Fast Lane: Project-Based Learning of Advanced Aerodynamics Using a Rapid Potential Flow Code

Tyler J. Souders[1]
Arizona State University, Tempe, Arizona 85281
Kevin M. Heitmann[2]
Arizona State University, Tempe, Arizona 85281
Timothy T. Takahashi[3]
Arizona State University, Tempe, Arizona 85281

This paper discusses academic use of the well-known vortex-lattice code, VORLAX, which has been improved at Arizona State University and distributed to upper-division and graduate level Aerospace Engineering students. Recent code upgrades have reduced VORLAX execution time up to 90%. In this paper, we show how near-real-time aerodynamic solutions improves learning outcomes when students partake in highly iterative design projects. Furthermore, improved, more interactive data visualization programs enable more intuitive understanding of how pressure distributions develop about candidate airframes.

I. Introduction

THE typical undergraduate aerospace engineering curriculum focuses on the application of the ideal. Classic texts present many techniques as being "universal" and, hence, adaptable when they are in fact quite far removed from "real world" behavior. One common area where traditional engineering education misrepresents reality involves introductory aerodynamics. Many students latch onto the concepts presented by 2D airfoil theory and fail to comprehend the induced effects associated with three-dimensional flow about a finite wing and body configuration; a real 3D wing does not behave as a succession of 2D airfoils altered by twist and downwash. Many students demonstrate improved understand when the concept is something tangible, i.e., it is possible to bend a rubber rod and watch it deform as a function of the force applied. However, the complexity of solving 3D aerodynamic problems traditionally limit the ability to make "real-time" observations regarding how shapes impact flow fields.

Simple computational fluid dynamics (CFD) programs can improve student understanding by showing the effects of: 1) geometry changes (thickness and camber and twist), 2) the freestream Mach number, and 3) overall angle-of-attack on integrated (i.e. *CL*) or field (*Cp* distributions) properties. To avoid further confusing students, this CFD program needs to be simple. While professionals and researchers can comprehend results from a large-scale program capable of resolving compressibility effects, turbulence, friction, and heat transfer simultaneously, this amount of data becomes overwhelming for students. However, the challenge lies inherently in the condition that the programs must offer data that reasonably reflects real-world behavior, particularly when changing conditions. In this case, two-dimensional examples disregard the development of spanwise flow (and spanwise pressure relief) over an aircraft wing, and there does not exist trivial means to reconcile the deviations from reality, however a program that neglects friction effects may be supplemented by an additional, equally-simple program, thereby offering students a complete picture of aerodynamic configuration effects which aid in building intuition necessary to truly understand flow characteristics and the correlations between configuration changes and performance effects.

[1] M.S. Graduate, Mechanical Engineering, Arizona State University, P.O. Box 876106, Tempe, AZ. Presently Ph.D Student, Mechanical Engineering, University of Colorado, Boulder, CO. AIAA Student Member
[2] B.S. Graduate, Barrett Honors College, Aerospace Engineering, Arizona State University, P.O. Box 876106, Tempe, AZ. AIAA Student Member
[3] Professor of Practice, Aerospace Engineering, Arizona State University, P.O. Box 876106, Tempe, AZ. Associate Fellow AIAA.

This paper describes this process using a recently improved vortex-lattice method program. [1][2][3] The vortex-lattice method is a prime example of a simple program that helps students build intuition via the examination of reasonably approximated real-world cases. The program, VORLAX, has been utilized in upper-division and graduate-level courses at Arizona State University since 2012, offering students the opportunity to better understand flow characteristics using their personal computers as an extension of the education provided in the offered introductory aerodynamics course.

II. VORLAX

VORLAX is one of many applications of the vortex-lattice method in a computer program. [1] The vortex-lattice method is a computational reduction of generalized potential flow theory, which means that the program makes a certain set of assumptions which inherently keep the overall concept simple. This should not be confused, however, with the notion that the program itself is simple. Rather, the program features complex subroutines for geometry generation and supports a variety of features which make it applicable for advanced applications, however when avoiding extra complex features, running the program remains simple. Furthermore, *VORLAX* operates in a manner where it is possible for an educator (such as a professor or teaching assistant) to configure the input files in a "one and done" fashion where the students may take the template and make edits sufficient for an entire semester's worth of education.

The *VORLAX* program applies the vortex-lattice method with the assumptions that the flow is compressible, inviscid, without separation, steady, and shock-free. [1] While these assumptions are substantial and do not reflect real-world behavior, they are so large and substantial that they are understandable to students and are also supplemented by other programs, such as *EDET*, a program dedicated to resolving friction drag over an airframe. [4] These assumptions also provide their own teaching moments. For instance, while it is not possible to resolve shock behavior with a vortex-lattice method, it is perfectly valid to use it to identify locations where a shock is likely to form using critical pressure coefficients. [5][6] Thus, what remains is a program that can provide a lot of information to students without intricacies that further confuse them, which makes for a phenomenal educational tool.

The vortex-lattice method operates by representing a geometry, typically an aircraft, as a combination of flat, two-dimensional panels, shown in FIGURE 1. Each of these panels is broken into a number of discrete sections, partitioned by two global parameters which dictate the number of chordwise and spanwise sections that each panel should be broken into. At each of these sections, a bound vortex element ("horseshoe vortex") is placed at its quarter-chord point, with an associated control point located at the three-quarters point, a spacing deemed optimal for most applications. [7] Thus, for a panel with 20 spanwise sections and five chordwise sections, there will be 100 total sections. At each of these sections, the flow induced normal to the control point is computed, not only from that section's vortex, but also from each of the other vortices. The question to be answered is: what vortex strength are each of these vortices? This is not a trivial solve, and in order to close the equations, a zero-flux boundary condition is applied at the control point. Thus, for N sections, there is a resultant $N \times N$ dense system of equations to solve in order to determine the strength of each vortex.

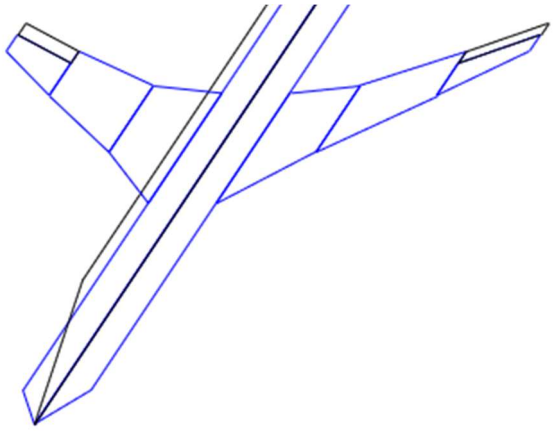


FIGURE 1. Vortex-Lattice Flat Panel Geometry [3]

Recent work [2][3] demonstrated sizable performance improvements to the *VORLAX* program. Having come originally from FORTRAN 66 before being ported to modern Windows personal computers using FORTRAN 77, there were many artifacts that degraded the performance of the program specifically the reliance on "TAPE I/O" to store the computational matrix. These artifacts were impactful to the student learning experience because they limited the number of runs a student could feasibly attempt for a design. For instance, a detailed stability and control run previously took ~8-10 minutes to run, while it is now of the order of 45 seconds to one minute. The "artifacts" in question were namely the reliance on scratch files (thereby relying on thousands of read/write cycles during a run) and poor initial guesses in the iterative

solver. The work demonstrated in [2] showed that the solver itself was quite efficient, benefitting from minor improvements to the initial guess for each iterative solve. The paper also showed that the most influential change to performance was due to the move from scratch files to in-memory operation. The improvements significantly improved accessibility to the program, as there was no longer a harsh "sunk cost" time punishment for pushing the bounds on a given program run, which led to large advancements in the capstone course.

Unlike many "traditional" vortex-lattice implementations, *VORLAX* allows the user to account for effects due to compressibility [1]. This is accomplished by applying the well-known Prandtl-Glauert correction (β, Equation 1) to the velocity perturbations relative to the freestream flow. This is advantageous for advanced aerodynamics courses, as essentially all interesting aerodynamic problems account for compressibility effects. For education applications where compressibility effects are out of the scope of interest, they are disabled by setting the freestream Mach number (M_∞) equal to zero.

$$\beta^2 \equiv 1 - M_\infty^2 \tag{1}$$

From basic potential flow theory, one knows that if the circulation strengths of the vortices are known, then it is a trivial task to determine the pressure distribution about the body. As a result of knowing the entire pressure distribution, it becomes just as trivial to numerically integrate over the surfaces and learn nearly everything about the configuration, both from an aerodynamics point of view as well as for stability and control purposes. One advantage that *VORLAX* has compared to other full-fledged CFD Suites is that the geometry generation and meshing in *VORLAX* is remarkably simple. There are no requirements for convoluted grid generation programs, as each 2D panel is drawn based on a set of principal coordinates. Likewise, the entire act of grid/mesh generation is handled via the aforementioned global parameters dictating the number of points per panel; see FIGURE 2. Thus, *VORLAX* is incredibly simple for a student to begin running, even without any prior CFD experience.

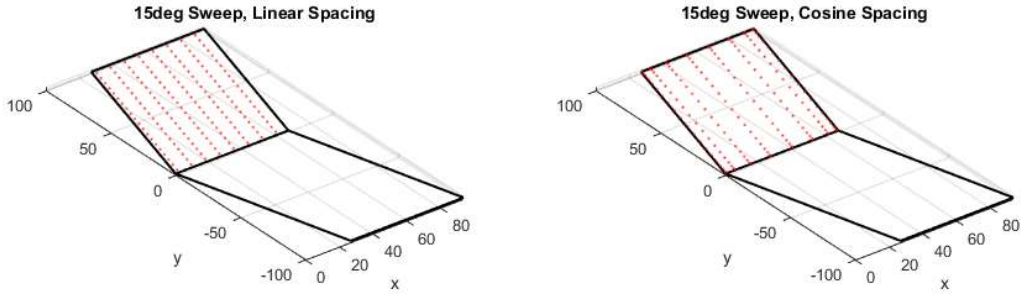


FIGURE 2. Example VORLAX automatic grid generation methods, (a) Linear and (b) Cosine Spacing [2]

Another perk of the *VORLAX* program is that its utilization is computationally very cheap. The program can be run in a matter of seconds for even the most intensive cases in a matter of seconds, and the input and output files are basic text files with a very low storage cost [2]. The entirety of the pre- and post-processing efforts can be fully automated via scripting languages, with common favorites being Visual Basic for Applications (*VBA*), Python, and MATLAB. By generating automated plot outputs and contour plots, it becomes even easier to intuitively understand the action and reaction combination typically seen in an aircraft design process, and thus for the applications of *VORLAX* at Arizona State University, creating the tools necessary to interface with *VORLAX* is an integral portion of the curriculum. One benefit to leaving the students to develop their own tools is that they may choose to do so in whichever scripting environment suits them best. The task at hand involves building an intuition with aerodynamics, so avoiding the extra task of becoming proficient in a new language is fantastic.

While *VORLAX* was always very cheap to run on modern machines, that is not to say that it was always fast to run. The program previously relied on FORTRAN-style file-based TAPE I/O scratch files to save the data instead of running using memory as typical modern programs do. This was remnant from the ages of memory tape style programming, where data would be stored and read from rotating tapes. There was one perk to the memory tape style read/write interface for the program, as it allowed the problem sizes to be incredibly large. However, it was very slow. Moving *VORLAX* to operate on memory led to massive time savings when running the program. To understand the time savings, it is necessary to understand the workings of the *VORLAX* grid generation and solve.

VORLAX operates by solving a dense system of equations, making the act of solving the system the most computationally expensive portion of the program. FIGURE 3 shows the relative cost of the solving subroutine compared to the geometry generation routine and the remainder of the program, in which the proportions of runtime are demonstrated. [8] The solver operates in vector segments, not with the use of any matrices due to memory constraints with the program. The method of solving the system is via a controlled standard over-relaxation method, which can over- or under-relax the solution iteration depending on the computed residual [1]. Each iteration of this solver (which typically converges in a couple hundred of iterations) involves reading the vector data line-by-line to feed into the solving routine. Modern solid-state drives are incredibly fast; however, they remain much slower than RAM, especially for the cumulative runtime of thousands of read and write cycles.

TABLE 1 shows the realizable "wall time" improvements for the 2020 version of *VORLAX* relative to the previous 2014 version of the program. These improvements focus minimally on the exact configuration of the vortex-lattice model, instead focusing on the problem size in order to time the solve routine. Each of the three grid densities were run for three freestream Mach numbers and three angles of attack. At first glance, ~O(20 seconds) does not seem particularly taxing for runtime, however, these benchmark cases are simplified relative to many common usage scenarios. For cases in which *VORLAX* is used to compute stability and control parameters, there will typically be three freestream Mach numbers with 14 different angles of attack, leading to a longer runtime. In addition, this configuration will be run between 4-5 times in order to compute the stability derivatives as a function of control surface deflection. All considering, this leads to 6 -10 minutes in runtime for a single configuration, which must be redone for each incremental change made to the *VORLAX* model.

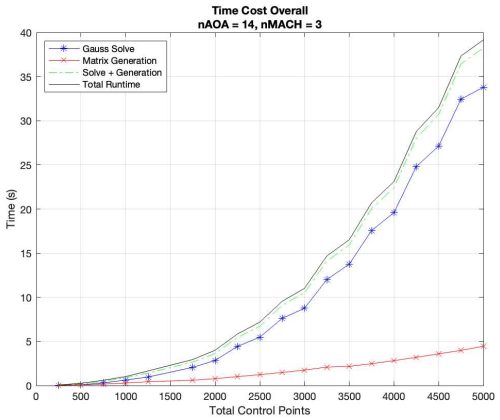


FIGURE 3. VORLAX runtime by subroutine [8]

TABLE 1 - Gauss CSOR Runtime Comparison (sec) [3]

	NVOR = 25, RNCV = 10	NVOR = 50, RNCV = 20	NVOR = 100, RNCV = 40
VORLAX 2014	0.078130	0.984380	20.296880
VORLAX 2020	0.046880	0.328130	5.718750
% Improvement:	**40.0%**	**66.7%**	**71.8%**

With such costly runtime commitments, the realistic number of iterations for control surface design was quite low. With the new version of *VORLAX*, this runtime has been decreased to ~60-80 seconds for a typical stability and control, which has drastically increased the number of runs that a capstone team can perform and has also increased students' freedom to attempt new designs. In practice, the decrease in runtime has had very positive effects for the senior design course, detailed further in Section V.

There have also been improvements to the program unrelated to program runtime. There were incompatibilities discovered inherent to the program's operation that were disabled. For example, the method of computing the leading-edge thrust effect was incompatible with the linear option for chordwise control point spacing, however this was not originally disallowed in the program. Thus, students would run calculations for a given airfoil shape and have incorrect results. This has been blocked via a check in the program in order to prevent students from making the mistake. *VORLAX* was also previously unhelpful with error codes and messages. For instance, if a student attempted to build a model featuring more control points than supported by memory, the program would exit leaving only memory block locations and generic FORTRAN errors. To improve usability, there are now provisions in the program to check aspects such as the grid size and provide meaningful errors in the VORLAX.LOG.

The entire reason to use *VORLAX* is to teach students aerodynamics, not to serve as a course in computational fluid dynamics. *VORLAX* is a reliable CFD program for the cases discussed in the following

sections, so the mentality of its usage is to show students how their learning translates to real behavior, much like a lab experiment, however without the expensive equipment. Thus, with this context in mind, computer errors detract from the learning experience. Much of the inspiration for the recent work done to *VORLAX* was with students in mind, and because *VORLAX* is a tool for learning (as opposed to the goal of learning), it is in the best interest of both the instructor and students for the program to run smoothly.

The manner in which Professor Takahashi teaches his design and aerodynamics courses involves heavily on student participation. Students are encouraged to communicate with each other for help, to come to office hours, to utilize institutional tutoring centers, and ask questions to the teaching assistants. With this large network of resources, quality of life improvements to *VORLAX* help the students almost as much as the runtime improvements. From my experience working extensively with the *VORLAX* program as a student, as an instructional assistant, and as a graduate student, I have witnessed the disruptions to the student learning process associated with program bugs and cryptic error messages. Thus, much of the recent work done with *VORLAX* addressed these issues by means of better error messages, primers designed to assist the students in hitting the ground running with the program and the input/output process, and better documentation of the program's limitations altogether. [8]

III. Working with VORLAX in an Education Setting

VORLAX input and output employs UTF-8 encoded text files; this means that students can program most any computer language to modify and analyze *VORLAX* input, output, and *LOG* files. These files vary considerably in complexity; however, they behave very predictably in a manner which makes development of interfacing tools convenient and manageable for students. The input file focuses primarily on flight configuration and geometry, including fusiform construction for the fuselage and engine nacelles as well as nonplanar wing effects, such as thickness and camber. This is all done in a consistent fashion, with the FORTRAN card formats detailed in reference [1]. The primary takeaway is that, by virtue of *VORLAX* being an old FORTRAN program, the methods of inputting geometry details are concrete and very simple, which is ideal for student introduction.

The *csv* output file (not to be conflated with the *LOG* file) offers a basic breakdown of key aerodynamic performance parameters. This output is the friendliest for beginners to the *VORLAX* program. It provides information relating to factors like the lift and drag coefficients and aerodynamic stability forces (i.e. pitch, roll, and yaw moments). This is given in a comma-separated file that is easy to import and manipulate in programs such as MATLAB or Microsoft Excel, making it ideal for a first step. Using only this file, students may construct lift and drag polar plots as well as dozens of various stability and control plots, commonly used to demonstrate aerodynamic stability and performance for the senior design capstone project each semester.

For other applications, particularly those sought more often by students at the graduate level, there exists the *VORLAX.LOG* file. This file is much more complex than the output file, however it provides the local pressure coefficients at each control point for each flight configuration (angle of attack and Mach number). Additionally, it provides information relating to the local control point slope (for panels with camber) and the circulation strength at each control point. This information is less helpful for the grand scope of performance (after all, the program already integrates the pressures for the user), however it becomes incredibly helpful for advanced applications, both for creating pressure visuals as in FIGURE 4 and for nuanced aerodynamic design. It is with this file where questions pertaining to shock formation, pressure gradients, stall likelihood, and detailed takeoff performance can be evaluated, opening the doors to both educational and professional considerations of a proposed design.

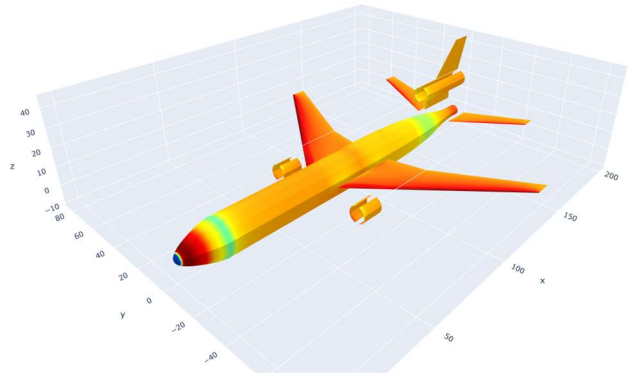
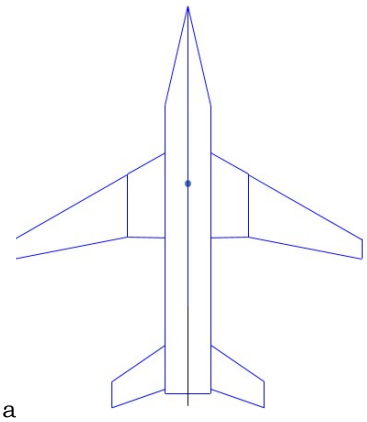
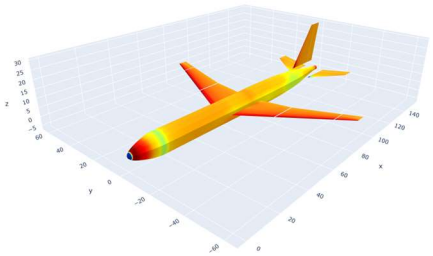


FIGURE 4. VORLAX Representation of an MD-11 [8]

The usage of *VORLAX* in the courses taught at Arizona State University involves all three of the file types. In the undergraduate senior design course, the students construct tools to evaluate the stability and controllability of a proposed design iteration. This typically includes regulatory compliance parameters (such as minimum control airspeed or crosswind performance) pertaining to 14 CFR § 25.149 and 14 § CFR 25.237 respectively, which is possible because *VORLAX* allows the user to model deflected surfaces, such as flaps, slats, ailerons, rudder, and elevator surfaces. In the advanced aerodynamics graduate-level course, students are tasked with evaluating effects of wing design perturbations on the pressure contours over a typical aircraft wing. This is done by altering airfoil shapes, local twist angles, camber magnitudes, and many more – all manifesting in a final design project that involves designing an optimal wing with efficient elliptical loading for a given flight condition using *VORLAX* [9]. These challenges are detailed further in Sections IV and V.



a



b

FIGURE 5. VORLAX Representation of a B757. a) line-art, b) 3D Pressure Field View [8]

One key aspect of a CFD program is the ability to meaningfully interpret the results. This is primarily done via visualization, both with 2D and 3D contour diagrams. While students are tasked with creating some of the visuals (particularly those showing local pressure loading and those showing stability and control derivatives), it is critical in an education application to offer reference visuals to which students may compare their individual results. The challenge in writing visualization and parsing tools is a massive portion of the learning process, and as such the students are tasked with modifying a basic parsing tool in order to complete their tasks. The code is very basic, but will always present all of the techniques necessary for students to develop their own codes.

The only codes that are completed for the students with no requirement that they reverse-engineer them are the codes related to the overall visualization of the VORLAX models. For years, Professor Takahashi has offered a line-vector visualization tool written in VBA to students for use while generating their models, and it has served as a great tool for better understanding the *VORLAX* geometry definition system. This script, as with all scripts, is complete with its own pros and cons. FIGURE 5a shows an output visual from the "old" visualizer. While it runs fast and offers rapid visuals for debugging *VORLAX* input files, it only offers a fixed selection of viewing angles. Furthermore, in some cases with high-resolution screens, the models may appear slightly disfigured. The script does a fine job showing the overall planform shape, and it does so in a manner that is computationally "lightweight" and very easy to implement in larger scripts.

To offer more intuitive imaging to students, although via a less convenient manner, a new visualizer was written in Python to generate a 3D model of the *VORLAX* aircraft, complete with all pressure data, as seen in FIGURE 5b. This form runs slower than the VBA version (though not so much as to discourage its use), however it generates an interactive model using standard plotting libraries which allow the user to zoom, pan, and rotate about the model however they please. Using the *LOG* file, the pressure data is displayed atop the larger scale geometry, making it much easier for students to "feel" the effects of a change in their model. Furthermore, because this tool works using the output *LOG* file, it is capable of displaying the fusiform constructions, complete with their associated pressure coefficients. Thus, a student may use the "old" visualizer during their development to have a rough idea of the shape while simultaneously using personal scripts to verify the data and measure quantities of interest. Upon obtaining a promising result, the student can make use of the "new" visualizer in order to reinforce or change their understanding of the flow behavior about their model. Thus, the combination is very powerful and provides numerous opportunities for students to strengthen their understanding of the material.

One continuous theme revolving around *VORLAX* as an instructional tool is the encouragement for critical thinking skills which supplement the derivation-heavy "classical" instruction that obscures real-world design objections via convoluted plots and rigorous derivations that seldom aim to improve the understanding of a topic. While understanding of the theory is necessary in curriculum, many instructors bury real understanding under layers upon layers of complex mathematics that force students to look only to minute details to see if they are "correct" instead of focusing on the larger picture and debating whether a solution "makes sense".

From several years of working directly with students, it has become apparent that many do not attempt to further their understanding beyond what is taught directly via course lectures and assignments. There is a non-negligible subset of students that are perfectly capable of understanding the big picture but get lost in discussions that serve more to confuse and add a layer of obscurity to the content. Consider solid mechanics, a common second-year engineering course. Some students will become lost with various sign changes, trigonometric functions, and extra "rules" to the point where they are distressed, however they will understand the content almost perfectly when presented a ruler that bends when a force is applied. This is the kind of "real understanding" that simple programs such as *VORLAX* aim to improve alongside the rigorous mathematics.

VORLAX in the classroom provides an opportunity for a feedback loop of understanding. The program runs so quickly that a student can change the model and observe the changes in near real-time. Much like the example of the ruler, this case is no different! When the ruler experiences a force, it bends, and possibly even fractures, and with *VORLAX*, the student can witness the lift coefficient increasing as camber is added to the wing. Aerodynamics is a challenging topic, and as such any tool which improves student understanding should be wholly welcomed.

IV. Casebook Study of Aerodynamics with VORLAX

Professor Takahashi teaches the advanced aerodynamics course at Arizona State University (MAE 564) using a "casebook" method. The course features two primary modes of education – the first encompassing the mathematically dense portions via class discussion and reviews of famous aerodynamics papers, and the second involves heavy usage of the vortex-lattice program *VORLAX* to validate these proposed relations. As a result, the students find themselves in a loop, employing critical thinking techniques in order to reconcile their theoretical understanding of aerodynamics with the "real world" results returned by the vortex-lattice program.

Due to the fast runtime of the vortex-lattice method, the students are able to participate in a near-instantaneous cycle of observing the action and reaction behavior of design perturbations for a connected-flow wing, which enforces critical thinking to answer the question "Does it make sense that this is happening?". Very little focus is given to theory-based assertions of expected behavior, instead having the primary focus on what occurs with the physical system. The final goal of the course is to complete a rigorous project designing an ideal wing for a given critical Mach number for a prescribed aircraft mission type. Each team has a different project; for instance, one team may be designing a small drone to quickly transport medication, while another is working on a widebody aircraft flying from Los Angeles to Tokyo. This variation encourages unique thought amongst the groups, while also allowing the teams to seek assistance from one another while mitigating the risk of a team offering "too much help", as the projects are sufficiently unique.

Prior to the final project, the students spend many weeks of the semester learning to use *VORLAX* and developing scripts to aid in their analysis. While the *VORLAX* input and output format exists as a simple flat-file, students do not find it trivial to write or parse. In the first portion of the semester, students learn how to handle errors in the program, how to read/write the data files, and how to automate this process. Some students find the automation process intimidating due to their unfamiliarity with Microsoft Excel/VBA the preferred tool for working with *VORLAX*. This has been alleviated by offering students primers on Visual Basic for Applications; we try to make the primers encyclopedic leaving the students time to really explore the results obtained from using *VORLAX*.

The assignments leading up to the final project are designed to draw the students' attention to various inconsistencies from the purely theoretical education many have prior to the course. Specifically, the assignments are written in such a way that students will answer a collection of questions. The primary themes and questions are as follows:

1. Document 2-D vs 3-D effects of thin, uncambered wings as a function of angle-of-attack and Mach number
   a. What are the effects of aspect ratio on integrated lift? (see FIGURE 6)
   b. Are the effects incidence pressure fields profoundly influenced by the spanwise position of any "2-D strip" on a wing? (see FIGURE 7)
   c. What do 2-D section cut pressure distributions look like across a thin, uncambered wing? Do they ever look like the pure 2-D idealized pressure distributions discussed in introductory aero?
2. Document 2-D vs 3-D effects of thin, cambered wings as a function of angle-of-attack and Mach number
   a. What are the combined effects of camber and aspect ratio on integrated lift?
   b. What do 2-D section cut pressure distributions look like across a thin, cambered wing? (see FIGURE 8)
   c. Are the effects of camber on pressure fields profoundly influenced by the spanwise position of any "2-D strip" on a wing?

3. Document 2-D vs 3-D effects of thick cambered wings as a function of angle-of-attack and Mach number
   a. What are the effects of thickness on integrated lift?
   b. What are the effects of thickness on the chordwise pressure distribution?
   c. What do 2-D section cut pressure distributions look like across a thick, cambered wing? Do they ever look like the pure 2-D idealized pressure distributions discussed in introductory aero? (see FIGURE 9)
   d. Are the effects of thickness on pressure fields profoundly influenced by the spanwise position of any "2-D strip" on a wing?

4. Document 2-D vs 3-D effects of swept wings as a function of angle-of-attack and Mach number
   a. What are the effects of sweep on integrated lift?
   b. What are the effects of sweep on the chordwise pressure distribution?
   c. Are the effects of sweep on pressure fields profoundly influenced by the spanwise position of any "2-D strip" on a wing?

5. Putting together these effects
   a. Do the effects of thickness, camber and incidence linearly superimpose as predicted with "thin airfoil theory?" (see FIGURE 10)
   b. How do the spanwise effects of thickness, camber and incidence interact on a finite swept wing?

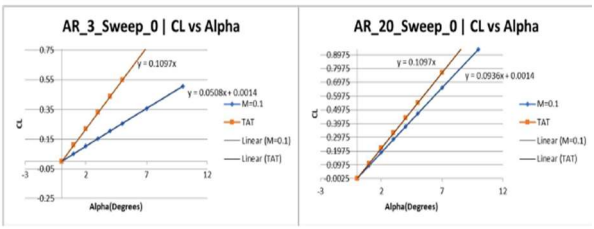


FIGURE 6 – *CL* vs α slopes of real wings compared to 2π per radian [9]

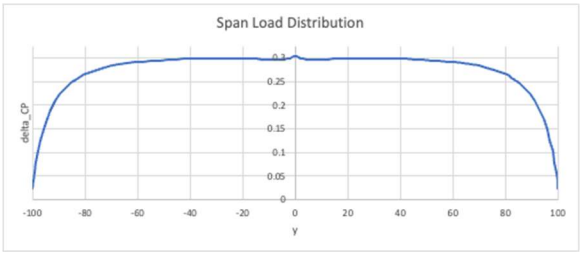


FIGURE 7 – Span Load of an Untwisted, Uncambered AR20 wing [10]

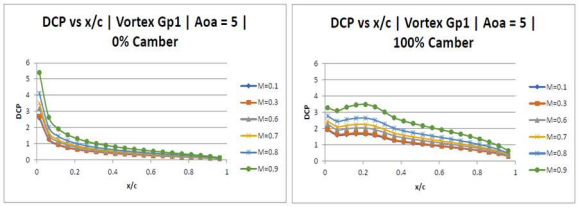


FIGURE 8 – Net Lift (*DCp* vs *x/c*) along the centerline of an AR20 wing at constant angle-of-attack, but at various Mach numbers. A) uncambered, b) NACA camber line [9]

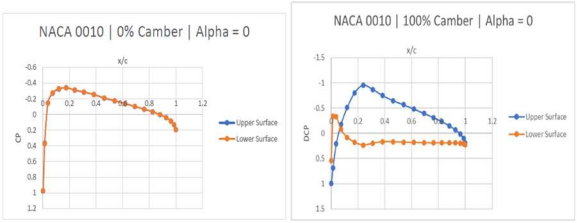


FIGURE 9 – Upper and Lower Surface Pressures (*Cp* vs *x/c*) along the centerline of an AR20 wing at α=0°. a) uncambered, b) NACA camber line [10]

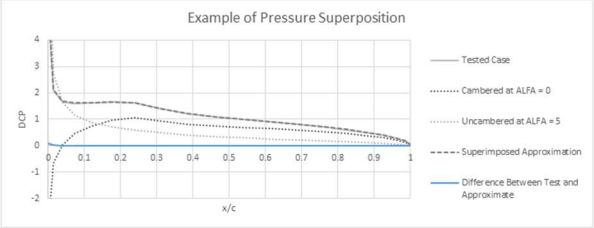


FIGURE 10 –Example of Superposition of pressures due to Thickness, Incidence & Camber [9]

Because these assignments and projects are completed using a single program, the comparisons remain uniform. The flows in question are always inviscid and attached because of the limitations of the *VORLAX* program itself. This forces the class to focus on the results before their eyes. Due to this simplification, the students are able to focus on the visible effects parameters such as wing thickness, camber, and twist have on the flow distribution, without the extra parameters such as Reynolds Number effects or flow separation to use as scapegoats for the effects of three-dimensionality of a finite wing.

The project which follows encompasses all of these topic areas. The students will build analysis spreadsheets that greatly speed up their design iterations. Using their earlier knowledge, they will work to obtain elliptical loading on a wing for their design using *VORLAX* as the primary analysis tool. The process is very iterative, and as such the improved runtime of the *VORLAX 2020* compile [2][3] is massively advantageous.

This method of instruction encompasses two primary modern pedagogical standards: "Project Based Learning" and "Near-Real-Time Feedback". In Project Based Learning, educators have the students "live" the project. The projects occur over a long timeframe, in the case of the advanced aerodynamics project, it begins near the midpoint of the semester. Thus, the students have plenty of time to reach out to the teaching staff (typically Professor Takahashi and a graduate teaching assistant) for feedback and inspiration, all the while continuing to analyze classical aerodynamics papers as a class. Due to the long period of the project, students have sufficient time to explore creative approaches to the problem while simultaneously examining the viability of these ideas, further developing critical thinking skills. [11]

Feedback-based learning is the other critical part to the advanced aerodynamics curriculum. The homework assignments build on each other – not in a manner which punishes those continuously for early mistakes, but in a fashion where feedback on an assignment will directly improve the next assignment. The idea of encompassing immediate feedback takes two forms. The first is due to instructor feedback on student efforts. The timing of the feedback ensures that it occurs in the middle of the learning cycle, as is ideal [12]. The course operates in a continuous manner, where topics organically flow from one to another, and this allows the professor to offer meaningful advice that becomes immediately helpful to the students. In some courses, particularly those at the undergraduate level, the topics are presented in a discrete manner, where the lessons jump from topic to topic without the possibility of making the topics "flow" nicely. In this case, it is difficult to provide feedback in a timely manner, and as such the feedback often goes underutilized.

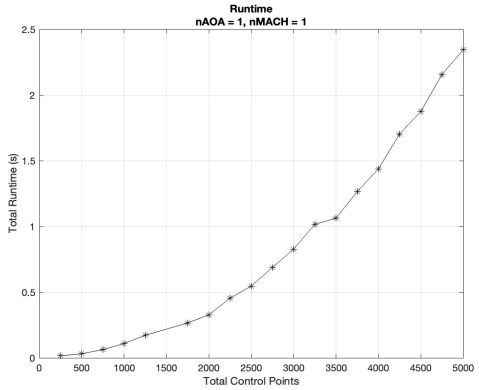


FIGURE 11. VORLAX runtime as a function of problem size [8]

The other form of immediate feedback of interest is that from the program itself. Because *VORLAX* runs in a matter of seconds for the majority of these trial education cases, the students are able to rapidly iterate through dozens of designs; see FIGURE 11. This is one of the biggest perks of the program. A student can vary one single parameter at a time and nearly instantly understand the effect of that change. In other programs, for instance ANSYS Fluent, this change may lead running another 10, 20, or even 60-minute run on a personal computer, which breaks the "flow" of the understanding loop. During a two second run, the student will sit at their PC and look to see what happens. Even with a one-minute run, the student will likely sit an observe. However, when the program runs begin taking more time, students may go make coffee, start scrolling on Facebook, clean their room, or go purchase dinner. This interrupts the learning process and makes it more difficult for students to verify the effects of their changes.

While *VORLAX* makes a few large assumptions, those assumptions are known, and prior work has verified the accuracy of the program [8]. More specifically, there has been a lot of work that shows that the trends shown by the program are correct. For instance, adding camber shows a linear increase to the lift as a function of angle of attack and adding thickness increases the lift while decreasing the pressure coefficients. Thus, in the context of education, the assumptions are forgiven in exchange for an effective learning feedback loop with the program. This is not to say that the numbers the students are working with are exact, rather the students know that there are some discrepancies – particularly those related to friction drag. However, this presents yet another opportunity for critical thinking. The students are informed of the

assumptions of the vortex-lattice method and may ask themselves during the semester whether their results are behaving logically. For example, for a wing of finite thickness, *VORLAX* will tell students that the drag coefficient is negative at zero angle of attack (as is correct for a wing with leading-edge thrust effects). This offers the students the chance to recognize that the addition of friction drag would fix the sign of the drag force while also offering the chance to see the effects of leading-edge thrust in action.

Thus, the advanced aerodynamics course operates by offering constant reinforcement for understanding and development of critical thinking. The course schedule is carefully catered to progress naturally, with in-class discussions, homework assignments, and the final project all complimenting each other. The students are offered all of the knowledge required to build analysis tools, as well as a select few tools which help validate their results.

V. Senior Design Project with VORLAX

A. Senior Design Overview

Professor Takahashi also teaches the senior design course at Arizona State University (AEE 468) during the Spring and Fall terms each year. [13][14] *VORLAX* is heavily relied on in this course, forming the foundation for much of the analysis students perform to determine the overall viability of an aircraft design. While used substantially, *VORLAX* is not the only useful "medium fidelity" code used in the senior design course. A massive part of the course involves performing trade studies that relate design changes to performance impacts, thereby requiring students to use critical thinking in order to weigh compromises between certain factors.

The trade studies performed by the students encompass a wide consideration of variables. The students must form a proper design goal, specifying range, payload, field performance, and fuel burn goals, among other quantities. Due to the complexity of aircraft systems design, the aircraft systems are broken into more manageable categories, and a typical team will have a member working on stability and control, another working on structures, and another working with propulsion, continuing for each main "area" of design, as the team sees fit. It is the job of the team to determine which aspects of design "belong together" – this encourages students to think about which categories make sense to couple. For instance, it does not make sense for the person working on the design of the cabin to also be working on the elevator sizing, nor does it make sense that the person working on the propulsion system be the sole designer of the rudder. The project is rigorous enough that a single member of a team will not find success without working with a team, much like real engineering projects.

The forward portion of the senior design course involves developing a series of tools (typically using VBA within Excel spreadsheets, although the students can technically use whichever language they please, but it is discouraged for compatibility and assistance reasons). The projects have been carefully designed by Professor Takahashi in order to ensure they work seamlessly with *ModelCenter*, the optimization and trade study software of choice. Furthermore, the tools are intended to link together to form a feedback loop, wherein information from sheet "A" directly feeds into sheet "B", but the information from sheet "B" can return to sheet "A" in an iterative design approach. The tools encompass six primary areas: weight, structures, stability and control, aerodynamic performance, "Skymaps" flight performance, and field performance. By coupling these six areas, the students may generate meaningful performance FIGUREs that verify outcome compliance for their given mission.

VORLAX is most relevant to the areas of stability and control and aerodynamic performance. In the former version of *VORLAX* from 2014, the program was too slow to integrate into the *ModelCenter* iterative workflow. It also relied on scratch files that would commonly lead to errors when trying to automate numerous runs in rapid succession. These drawbacks have been fixed in the 2020 version of the program, allowing the students to integrate the program seamlessly with their other tools.

B. Senior Design with VORLAX, a Student's Perspective

The senior design project for the Spring of 2021 was the second term to use the new version of *VORLAX*, and teams had fantastic performance with the software. Groups were tasked with designing a short- to mid-range cargo feeder aircraft for operation in a post-covid environment. Due to COVID-19, passenger air travel went down, while loads aboard dedicated cargo air carriers increased due to a massive decrease in available

FIGURE 12 – Heitmann's Senior Design Airplane [17]

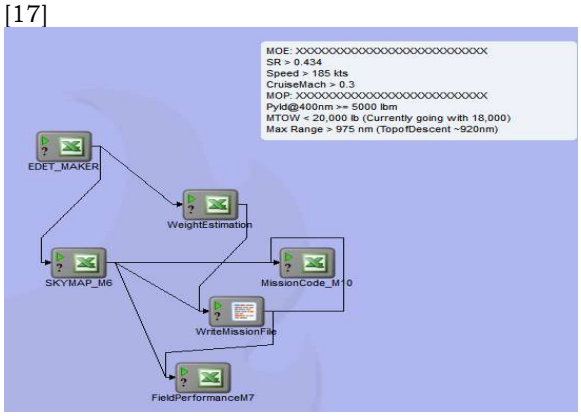


FIGURE 13 – *ModelCenter* environment used to integrate tools and processes into a Model Based Systems Engineering framework

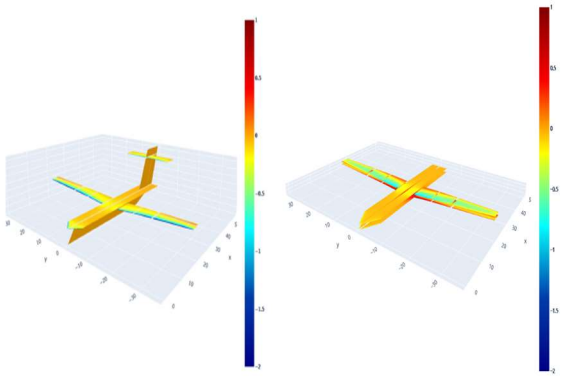


FIGURE 14 – Two Renderings from VORLAX – on the left a flat panel model for S&C computation, on the right a wing/body sandwich panel model. [17]

underbelly storage on typical passenger aircraft [16], leading to the need for a new aircraft that saves both time and money on cargo flights and operating costs from large metropolitan airports to smaller, often rural airports.

Mr. Heitmann's team designed a new cargo aircraft to support these missions; see FIGURE 12. [17]

They used *VORLAX* in their project to create tools for the analysis of aerodynamic performance, stability and control, and for aircraft model visualizations. FIGURE 3 showcases the flowchart used to optimize the design of the wings. This flowchart was then implemented via *ModelCenter*, which is a software package that aids in the design and optimization of trade studies for systems [16]. FIGURE 13 is an example of how the different tools would be input into *ModelCenter*.

For the aerodynamic performance, *VORLAX* was used to determine the optimal twist of the wing. The cutoff for optimal twist was determined by manipulating the twist at control points until the *VORLAX* computed lift coefficient was greater than the lift coefficient found from the dynamic pressure, weight, and reference area during cruising flight. This twist was evaluated in two different configurations, the first was using the flat plate models in *VORLAX*, and the second using sandwich panels. These differences are showcased in FIGURE 14. Flat plate panels are good estimations for lift and drag through *VORLAX*, however they fail to offer information about the local flow behavior. Sandwich panel representations are more accurate to real-world wing design, particularly when considering the implications of the local pressure coefficient [8]. Thus, the sandwich panel configuration was used in the project for detailed design with the flat plate twist used as a starting point. The sandwich panels were used without the control surfaces or tails as they were explicitly focused on finding twist and pressure distribution along the wing.

The sandwich panels allowed for camber, airfoil, and thickness profiles to be added to the wing to find elliptical loading. By varying the three parameters, it is possible to achieve elliptical loading without resorting to the use of winglets, which add considerable friction drag and weight to the aircraft. Using *VORLAX*, changing the profiles during analysis took seconds from start to finish. In addition, the implementation of Excel/VBA allowed for a more seamless transition of data for analysis as shown in FIGURE 15 (overleaf).

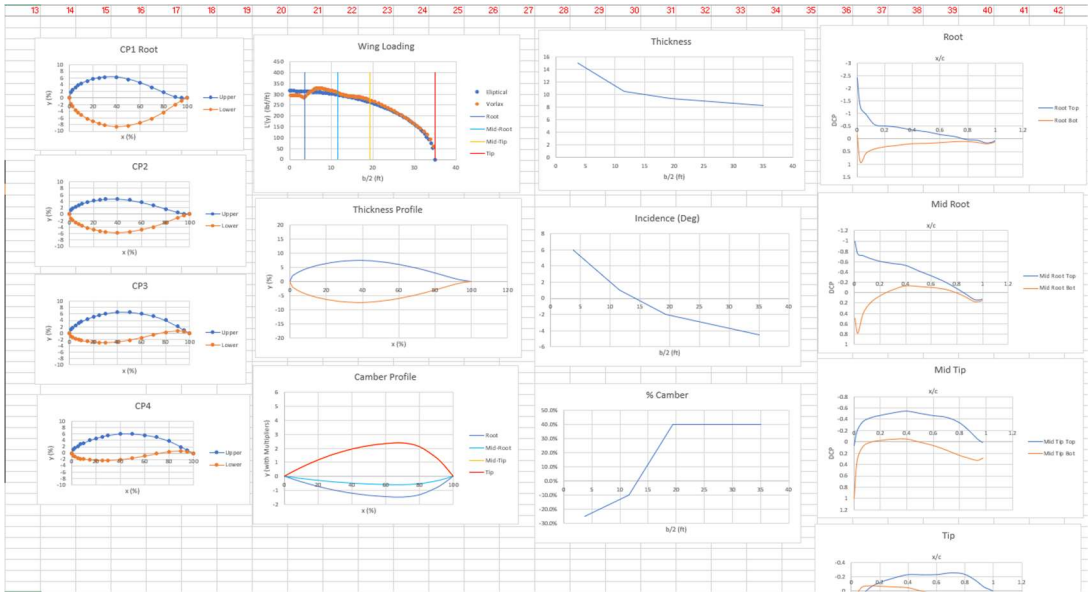


FIGURE 15. VORLAX Aerodynamic Analysis Outputs

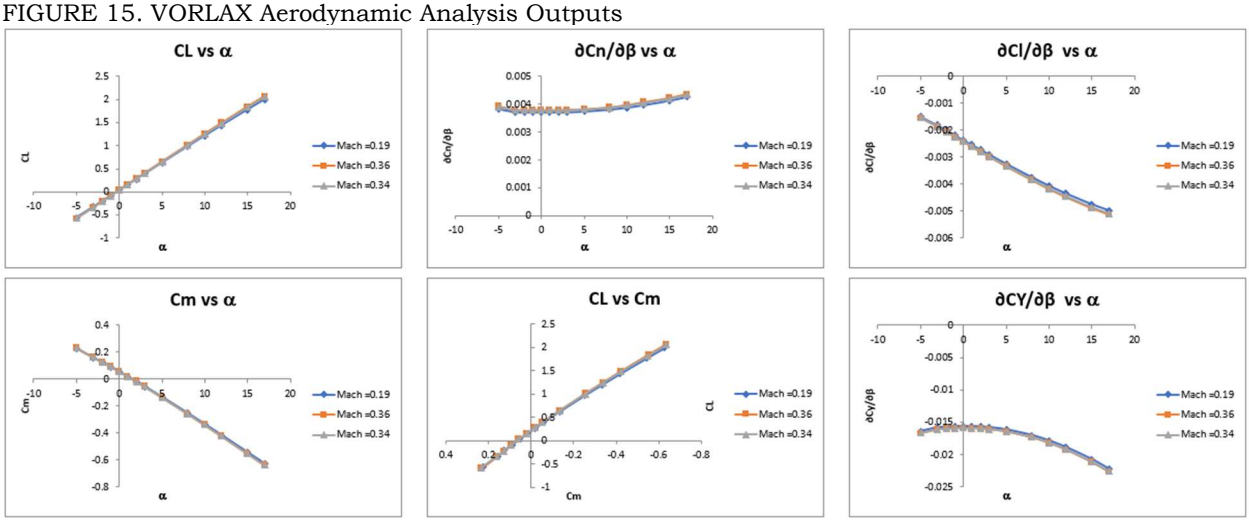
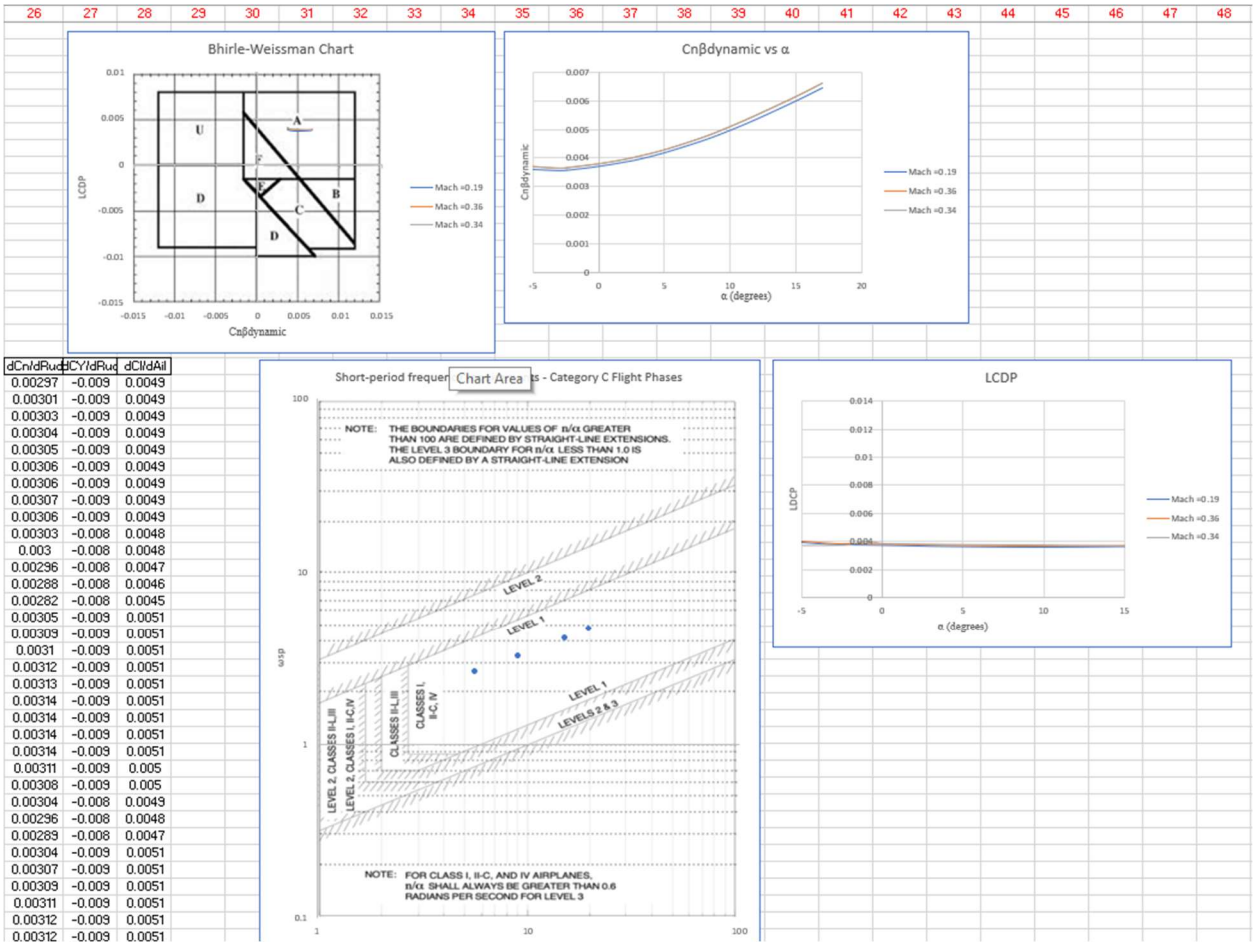


FIGURE 16. VORLAX Stability and Control Output Examples [17]

Heitmann and his student collaborators developed a 3D wing loft with spanwise variation of thickness, camber and incidence to achieve an "elliptical" transverse lift distribution at the design CL and form a favorable pressure distribution for stall characteristics (mute the leading edge suction spike at the leading edge over the outboard wing panel).

The stability and control analysis was done in a similar manner to the twist analysis. Using a more complex governing variables page shown in FIGURE 8 to create the *VORLAX* input file, the output file was able to populate the stability derivatives needed for analysis.

This governing sheet was able to handle the twist, tail size, aerodynamic geometry, and control surface deflection for the inputs. The button shown is a method to run the macro in VBA that does four runs of *VORLAX*. Each run incorporates a different scenario: no control surfaces, aileron deflected, rudder deflected, and elevator deflected. Each scenario took approximately 20 seconds leading to a full run taking 80 seconds to import data and fully populate a stability and control Excel sheet for analysis. From the data in that sheet, the graphs in FIGUREs 16 and 17 were created. The graphs are a visualization of the longitudinal, lateral and directional stability of the aircraft. In addition, the MIL8785C [18] and Bhirle-Weismann [19] charts were created by post-processing *VORLAX* outputs.



dCn/dRud	dCY/dRud	dCl/dAil
0.00297	-0.009	0.0049
0.00301	-0.009	0.0049
0.00303	-0.009	0.0049
0.00304	-0.009	0.0049
0.00305	-0.009	0.0049
0.00306	-0.009	0.0049
0.00306	-0.009	0.0049
0.00307	-0.009	0.0049
0.00306	-0.009	0.0049
0.00303	-0.008	0.0048
0.003	-0.008	0.0048
0.00296	-0.008	0.0047
0.00288	-0.008	0.0046
0.00282	-0.008	0.0045
0.00305	-0.009	0.0051
0.00309	-0.009	0.0051
0.0031	-0.009	0.0051
0.00312	-0.009	0.0051
0.00313	-0.009	0.0051
0.00314	-0.009	0.0051
0.00314	-0.009	0.0051
0.00314	-0.009	0.0051
0.00314	-0.009	0.0051
0.00311	-0.009	0.005
0.00308	-0.009	0.005
0.00304	-0.008	0.0049
0.00296	-0.008	0.0048
0.00289	-0.008	0.0047
0.00304	-0.009	0.0051
0.00307	-0.009	0.0051
0.00309	-0.009	0.0051
0.00311	-0.009	0.0051
0.00312	-0.009	0.0051
0.00312	-0.009	0.0051

FIGURE 17. VORLAX Stability & Control Plots [17][18][19]

As mentioned earlier, a full run of *VORLAX* using Excel VBA took approximately 80 seconds. This short time frame allowed for many more variations of design to be tested. It would take less than 10 minutes to run through and analyze the stability and control of 7 different designs. This is four times faster than the previous iteration of *VORLAX*. Using the same Excel sheet built by the group, the computational time was found to be around 80 seconds per scenario, or 320 seconds per run of VBA with the old version of the program. The lower efficiency would have led to many issues when designing the craft. The benefit to using *VORLAX* for the class was that students could see in real time what changes to design parameters did to for the stability and control. Using these insights, they could then automate hundreds of changes within a reasonable range into *ModelCenter* to find the optimal design for their mission. This was only possible due to the improvements done to *VORLAX*. One of the groups in the design class ran 315 varied designs through *ModelCenter*, taking 7 hours to compute. This method for finding the optimal design was easily run overnight to prevent interference with the schedule of the project. It also gave a reasonable margin of error for rerunning the study in case of an issue in the initial parameters. The older iteration of the program would have presented major time constraints and complications due to the use of scratch files for each individual run. A single change in a design would have needed almost 6 minutes just to see the stability. Running that same 315 optimization case with the older form of *VORLAX* would have led to 28 hours of runtime, during which the computer could not have been used for the project. The integrated master schedule that the students made for the project would have been ruined by the shear runtime required to design the wings, leading to lower quality submissions to meet time constraints. Thanks to the improvements made to *VORLAX*, the students were able to submit professional level designs.

VI. Conclusion

By means of efficient computational fluid dynamics (CFD) programs, students may partake in project-based learning with rapid feedback necessary to reinforce learning via action and reaction associations. Recent work done to improve the vortex-lattice program *VORLAX* has allowed students to seamlessly integrate the program into advanced aerodynamics and full configuration design projects in two courses taught by Professor Takahashi at Arizona State University. Performance improvements to *VORLAX* have allowed students to run the program more rapidly than ever, as well as implement the code into optimization programs, which has contributed substantially to group project performance. Using the improved version of *VORLAX*, students were able to run many more iterations of their designs than previously, leading to better learning due to constant feedback loops that strengthen understanding.

Acknowledgements

This manuscript derives from work Mr. Souders performed in partial fulfillment of the degree requirements for obtaining his M.S. in Mechanical Engineering from Arizona State University. It also derives from work Mr. Heitmann performed in partial fulfillment of the degree requirements for obtaining his B.S. in Aerospace Engineering from Arizona State University. All design analysis on this unfunded project was completed at Arizona State University.

VII. References

[1] Miranda, L. R., Elliot, R. D., and Baker, W. M. "A Generalized Vortex Lattice Method for Subsonic and Supersonic Flow Applications." NASA CR 2865, 1977.
[2] Souders, T. J., & Takahashi, T. T. "VORLAX 2020: Benchmarking Examples of a Modernized Potential Flow Solver." AIAA 2021-2459, 2021.
[3] Souders, T. J., & Takahashi, T. T. VORLAX 2020: Making a Potential Flow Solver Great Again. AIAA 2021-2458, 2021.
[4] Feagin, R.C. & Morrison, W.D. "Delta Method, An Empirical Drag Buildup Technique," NASA CR 151971, December 1978.
[5] Küchemann, D., The Aerodynamic Design of Aircraft, AIAA, Virginia, 2012
[6] Armenta, F.X., Plaban, P, & Takahashi, T.T., "Revisiting Neumark's Critical Pressure Coefficient Rule for Aircraft with Finite Wings," AIAA 2022-1157, 2022.
[7] Kalman, T. P., Giesing, J. P., & Rodden, W. P. "Reply by Authors to G. J. Hancock." Journal of Aircraft, Vol. 8, No. 8, 1971, pp. 681–682.
[8] Souders, T.J., "Modernization of a Vortex-Lattice Method with Aircraft Design Applications." M.S. Thesis, Mechanical Engineering, Arizona State University, 2021.
[9] Souders, T.J. MAE 564 Homework, Fall 2019, Arizona State University, 2019.
[10] Heitmann, K.M. MAE 564 Homework, Fall 2019, Arizona State University, 2020.
[11] PBLWorks. What Is PBL? https://www.pblworks.org/what-is-pbl.
[12] Feedback That's "Just in Time." https://harnessassessment.com/2019/09/22/feedback-thats-just-in-time/. Accessed May 27, 2021.
[13] Takahashi, T.T., Aircraft Performance & Sizing, Vol. I: Fundamentals of Aircraft Performance, Momentum Press, New York, NY, 2016. 200 pages. **ISBN-13:** 978-1606506837
[14] Takahashi, T.T., Aircraft Performance & Sizing, Vol. II: Applied Aerodynamic Design, Momentum Press, New York, NY, 2016. 276 pages. **ISBN-13:** 978-1606509456
[15] Phoenix Integration. ModelCenter.
[16] Polek, G. Air Cargo Saw Record Decline in Demand in 2020. AINonline. https://www.ainonline.com/aviation-news/air-transport/2021-02-03/air-cargo-saw-record-decline-demand-2020#:~:text=Data released data for global air freight markets,to the ongoing Covid pandemic%2C the air . Accessed May 27, 2021.
[17] Ayoub, K., Chaidez, G., Heitmann, K., Hoopes, C., Mulkern, W., Nagpal, P., Olszak, P., & Raganathan, A., "FED EX FEEDERS Design Substantiation Report," AEE 468 Senior Design Project Report, Arizona State University, Tempe, AZ, 2021.
[18] MIL-STD-8785C Military Specification: Flying Qualities of Piloted Airplanes
[19] Bihrle Jr, W. & Barnhart, B., "Design Charts and Boundaries for Identifying Departure Resistant Fighter Configurations," NADC, 1978.