Journal of Research High School

Vol. 2025 (11) 889 – 895
ISSN 2688-3651 [online]

Using Design Software in Conjunction with CFD Solvers and Programmatic Automation to Streamline Airplane Wing Design

Aaditya Barbudhe[1] *

[1] Woodbridge High School, Irvine, CA, USA
*Corresponding Author: Aaditya.Barbudhe@cawgcap.org

Advisor: Tanay Topac, tanaytopac@stanford.edu

Received December 5, 2024; Revised June 25, 2025; Accepted July 23, 2025

Abstract

An aspect of aerospace design processes that increases costs and reduces efficiency is the laborious, iterative element, where designs must be repeatedly tested and edited until an acceptable design is found. To solve this, this study investigated the utility and impact of a software workflow that automates the iterative aspect of aerospace design. The study introduced a Python program to transform user-inputted values into a testable 3D model, which is automatically simulated. Results get delivered to the user via the program and in simulation report files. Preliminary results from these simulations have shown expected patterns in aerodynamic performance regarding the relation of angle of attack(AoA) to lift and drag coefficients, showcasing the effectiveness of the software workflow. By automating the process of creating new designs and simulating them, the process of searching for the optimal aerodynamic profile was accelerated by an order of magnitude, illustrating the potential of incorporating CAD (computer-aided design), CFD (computational fluid dynamics), and automation in advancing efficiency in aerospace design processes.

Keywords: Computational fluid dynamics, Parametric design, Flight aerodynamics, Automation, Computer-aided design

1. Introduction

In the realm of modern aerospace engineering, the pursuit of efficient and innovative aircraft design is a continuous endeavor. Advancements in such pursuits are driven by rapid technological advancement, especially in the design phases of the industry, where a great deal of time and money is spent in iterative design processes to perfect aspects of an aerospace vehicle (Weiss and Amir, 2023). Egbert Torenbeek (1976) described the design process as starting in the conceptual phase, in which designers brainstorm and sketch possible designs for the aerospace vehicle. Although hundreds, possibly thousands, of concepts may appear, only some will make it to the configuration phase. During this phase, the design concepts are narrowed down to various promising configurations, and designers begin to run comparative tests between configurations and assess individual configurations to narrow down the design. Wind tunnel tests are carried out, and mock-ups are made and iterated over until the project is deemed mature enough for secondary review by clients. To provide a sense of scale for this stage of the design process, it took the design team of the Lockheel L-1011, a widely used widebody commercial airliner from 1972, a time without any automation or digitization of design processes, over two years and two million man-hours as well as ten thousand hours of wind tunnel testing to achieve the optimum design (Torenbeek, 1976).

This conventional approach has inherent challenges, including time-intensive procedures, resource constraints, and substantial financial investments. Although much of this process has been digitized and therefore expedited with tools such as CAD and CFD, the problem of manually iterating through designs and simulations remains, and has not been addressed thus far. Their lack of integration with other components of the design process makes them unnecessarily cumbersome and tedious to use. A critical tool is required to address these limitations: an integrated

Vol. 2025 (11) 889 – 895
ISSN 2688-3651 [online]

methodology that seamlessly combines generative CAD with high-fidelity CFD simulations and optimization tools in order to automate iterative processes. The gap in the research of such a tool or workflow forms the core of this research, which aimed to transform aerospace design by offering an efficient and cost-effective means of crafting optimal airplane wings, expediting the inefficient methods of traditional initial configuration design, assessing configuration variations, and baseline configuration development.

The central thesis of this study delved into how programmatic automation, generative CAD, and high-fidelity CFD simulations can synergistically assist aerospace engineers in crafting optimal airplane wings that cater to specific flight conditions and constraints. This was studied by creating and utilizing a software workflow containing the aforementioned software to create and test an airplane wing, upon which the benefits for engineers of using the workflow compared to traditional development and testing methods were analyzed.

2. Methods

2.1 Overview

To understand the usefulness of a design software workflow to aerospace engineers, a Python script was created to take in user input for the design of a wing in the form of a NACA (National Advisory Committee for Aeronautics) 4-point profile for the airfoil and floating point values for the parameters of the angle of attack (AoA), taper, and wing length (Cantwell, 2013). Once the user has decided upon the parameters, the script uses a series of functions to compute the coordinates for each point on the airfoil. It starts PTC Creo (a CAD software) and tells PTC Creo to generate a model of a wing based on the coordinates generated and the parameters provided. This gets exported to GMSH (a mesh creation software), where the wing model is turned into a mesh file that SU2 (detailed simulation software) can test. After testing, results are delivered numerically through the command line and visually through flow visualization files.

2.2 PTC Creo

Before the Python Script was rendered functional, a template *.prt file had to be created manually within the PTC Creo software. This template file has to be used as a base for any airfoil that an engineer designs with the Python script because the script takes advantage of the parametric design aspect of PTC Creo, meaning that previously created points and object parameters could be programmatically altered, but could not programmatically created. The points of the airfoil and the relations between dimensions and parameters must be manually defined.

Once these parameter values and dimension-parameter relations have been defined manually, the script could programmatically alter the parameters to generate a model of a wing based on any valid NACA 4-point profile.

2.3 Python Script/Programmatic Automation

Several external modules were utilized to communicate with PTC Creo, perform complex operations, and carry out other vital tasks in the Python script used in the generation of the wing model and the communication between the aspects of the software workflow. The creopyson library was heavily used within the script due to its functionality as a communicator between the Python program and PTC Creo. This module's functions allow the Python script to command PTC Creo's various features, such as parametrically editing the coordinates of points or configuring design variables, such as taper, to create a wing model that matches the requirements of the designer.

The script's key logic lies in the naca4 function, which creates the wing contour coordinates based on the parameters and NACA four-point profile the user gives. It also formats the list of points so that PTC Creo can read them when it is generating the wing model.

The Naca 4-digit series airfoils were the first family of airfoils that utilized the findings of NACA engineers, which showed that the most successful airfoils shared many similarities, mainly the slope of the mean camber line (Figure 1) and the thickness distribution concerning the mean camber line. In NACA 4-digit series, the first digit specifies the maximum camber (m) in the percentage of the chord (airfoil length), the second indicates the position of

Vol. 2025 (11) 889 – 895
ISSN 2688-3651 [online]

the maximum camber (p) in tenths of the chord, and the last two numbers provide the maximum thickness (t) of the airfoil in the percentage of the chord (Cantwell, 2013).

Figure 1 provides a diagram of a NACA airfoil with the chord and mean line shown. The thickness distribution about the mean camber line, which was referred to earlier, is shown in the diagram through the varying distance of the profile's border from the mean line.

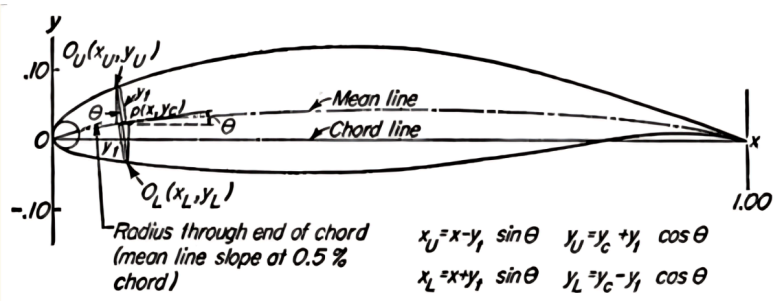

Figure 1. A representation of a NACA 4 point wing profile with its mean line illustrated and the relations needed to graph the wing profile (Cantwell, 2013)

The user enters four digits into the program that get parsed into the m, p, and t values. Next, the NumPy linspace function is used to create an array of evenly spaced floating-point values from 0 to 1, stored in the array labeled “x”. This is useful because these will be used for the abscissae of the wing contour coordinates. The mean camber line coordinates are attained by plugging in values x, m, and p into Equations 1 and 2 (Cantwell, 2013). Different equations are needed from zero to p and from p to 1 to generate the asymmetric shape of the airfoil.

$$y = \frac{m}{p^2}(2px - x^2) \; from \; x = 0 \; to \; x = p \tag{1}$$

$$y = \frac{m}{(1-p)^2}[(1-2p) + 2px - x^2] \; from \; x = p \; to \; x = c \tag{2}$$

Following this, the function titled "theta_function" takes in the parameters x, m, and p to determine the mean camber line angle, which is a metric that refers to the angle of the mean camber line about the airfoil's chord line, representing its deviation from the chord.

The script then calculates the thickness distribution above and below the camber line using the relation detailed by Equation 3 (Cantwell, 2013).

$$\pm y = \frac{t}{0.2}(0.2969\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4) \tag{3}$$

Finally, the final coordinates for x and y on the top and bottom of the camber line are calculated using the relations in Equations 4 to 7 (Cantwell, 2013).

$$x_U = x - y_t \sin\theta \tag{4}$$

$$y_U = y_c + y_t \cos\theta \tag{5}$$

$$x_L = x + y_t \sin\theta \tag{6}$$

$$y_L = y_c - y_t \cos\theta \tag{7}$$

To take the data produced from calculating the coordinates of the points and parse it into a format that is readable by PTC Creo, the script must first store the coordinates of the top contour line and bottom line in separate arrays. These arrays store points in the format [x,y,z], where the z coordinate is constantly set to zero, since a 2D model is initially desired.

2.4 Starting PTC Creo and Creating a Wing Model

For the Python script to run PTC Creo and give it commands, the CREOSON server, a locally hosted server made

Vol. 2025 (11) 889 – 895
ISSN 2688-3651 [online]

by the publishers of creopyson, must be started manually by running the file outlined in this Windows system path: C:\working\CREOSON\CreosonServerWithSetup-2.8.1-win64\CreosonSetup.exe.

This opens a window to start the local server, which creopyson relies on and facilitates communication between the Python Script and PTC Creo. Once this local server has been launched, the program uses the creopyson modules function "start_creo()" to initialize the creopyson client and then start PTC Creo. Once this has been done, the script polls to check whether PTC Creo is running. Once it senses that PTC Creo is running, the connect() function is used, which connects the script to the CREOSON server, and immediately after, "file_open" is used to open a template file in Creo that has 100 points already created along an upper and lower spline with each point having parameterized coordinate. Once the file is loaded, the script will begin to iterate through the upper wing contour coordinates, take the values stored earlier for each point, and write the x and y coordinates into the coordinate parameters for each point along the splines within PTC Creo. Along with the parameters for the top and bottom points' coordinates, The script also edits the parameters that control taper and wing length and alters the coordinates of the points to account for the AoA.

2.5 GMSH

Once the user determines an ideal testing range based on their design requirements, the PTC Model must be manually exported as a .stl file to GMSH, a mesh generator, to create a .su2 file that SU2 can read. Within GMSH, the model is given boundary and marker definitions and point relations for the SU2 file. The software can be controlled with text files written in GMSH's scripting language (.geo files), which were used to assign marker names and merge the model to create a mesh (Geuzaine and Remacle, 2009). An example of a finished mesh based on a NACA 2412 profile can be seen in Figure 2.

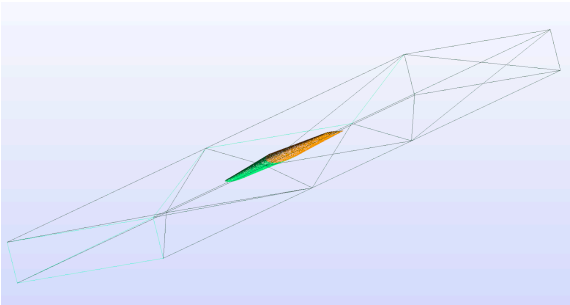

Figure 2. The completed mesh of a flying wing design in GMSH, based on a single wing model generated in PTC Creo

Once the mesh has undergone 3D optimization, it can be exported as a .su2 file and used for testing within SU2.

2.5 SU2

Before the mesh file was ready to be tested in SU2, creating a configuration file was required to define the problem SU2 would solve. Once the configuration file was created, the command to execute the solver was run from the Windows Powershell, the results were printed on the screen, and the flow files were stored in the directory where the PowerShell was opened.

A configuration file in SU2 is a text file that defines the conditions of the simulation to be run, such as the airspeed, the physical model to be used, boundary conditions, etc. Due to the complexity of creating an SU2 configuration file, an open-source configuration file was downloaded that matched the type of mesh and problem that was being solved from the SU2 library. Some modifications to the file were required to fit the requirements for this research, including setting the airspeed to subsonic (Mach 0.8), changing the reference length and Reynolds length to fit the wing’s chord of 1, renaming the boundary markers listed in the file, and changing the name of the input mesh file. After this, the mesh file was ready to be tested.

Once the mesh and configuration files are ready, they must be in the same directory as the SU2_CFD program for the simulation to work if SU2 is not added to the Windows system path. If it is added to the path, the files can be in any directory as long as they are together. After the files were in the correct location, Windows Powershell was run in the directory with the files and used the SU2_CFD command, followed by the name of the configuration file. Upon execution, the program provided tables of analytical data within the shell and produced history, flow, surface flow, and restart files in the directory where SU2 was located.

Vol. 2025 (11) 889 – 895
ISSN 2688-3651 [online]

3. Results

The significant products of the SU2 simulation included numerical values for the total lift coefficient and drag coefficient, as well as general and surface flow files, which show numerous elements around the wing, such as pressure and airspeed.

3.1 Drag and Lift Coefficients

For the tests conducted, the wing was tested at zero, five, ten, and fifteen degrees of AoA, all at Mach 0.8. The total drag coefficient (CD) and the lift coefficient (CL) for each test are seen in Figure 3 and Table 1.

Table 1: CD/CL in relation to AoA

AoA	CL	CD
	-0.135	0.358
	0.157	0.366
	0.454	0.424
	1.31	0.583

In the graph shown in Figure 3, the blue line shows the lift coefficient and the red line shows the drag coefficient. Both lines are in relation to angles of attack from 0 to 15 degrees. This graph shows that, until an angle of attack of 10 degrees, the wing produces more drag than lift and past 10 degrees, the lift appears to grow exponentially, with the drag showing linear growth, indicating that, within the AoA range of 0 to 15 degrees, 15 degrees of AoA is the most efficient. The same results can been seen in Table 1 Although a trend that a higher angle of attack may always be more efficient might be apparent in the graph, it should be considered that after a certain point, the behavior of these graphs will change, and the drag coefficient will overtake the lift coefficient, similar to a plane's behavior while stalling.

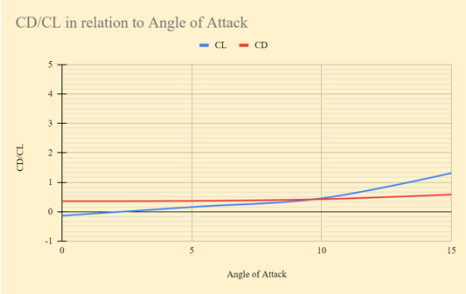

Figure 3. A graph of the lift and drag coefficient for a 2412 wing in relation to AoA ranging from 0 to 15 degrees

3.2 Flow Visualization

In Figures 4 and 5, the numeric data of the lift and drag coefficients is reflected in the flow visualization files produced by the simulation in which cooler colors indicate lower pressures while warmer colors indicate higher pressures.

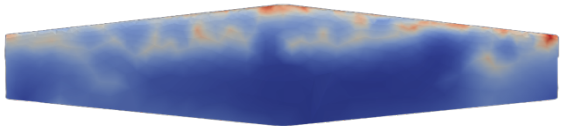

Figure 4. A pressure heat map of the bottom of the wing at AoA of 0 degrees, with warmer colors indicating higher pressures

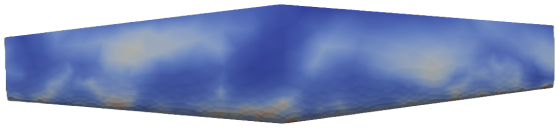

Figure 5. A pressure heat map of the top of the wing at AoA of 0 degrees, with warmer colors indicating higher pressures

In Figures 4 and 5, it can be seen that when at an AoA of zero degrees, lower pressures are most widely seen on the bottom as compared to the top, which would cause the wing to be pulled downwards, as is exemplified in Figure 4 where, during this test, it can be seen that the lift coefficient was negative.

In Figures 6 and 7, the numeric data is represented by the same color mapping.

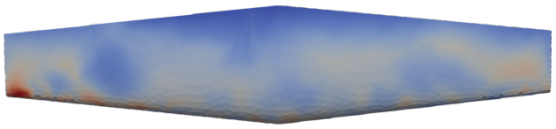

Figure 6. A pressure heat map of the bottom of the wing at AoA of 15 degrees, with warmer colors indicating higher pressures

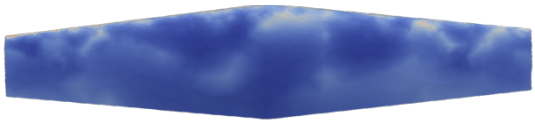

Figure 7. A pressure heat map of the top of the wing at AoA of 15 degrees, with warmer colors indicating higher pressures

In figures 6 and 7, it is revealed that significantly higher pressures are present on the bottom of the wing than on the top, which would result in the wing being pushed upwards, as can be seen in figure 6.1 where, at fifteen degrees of AoA, the coefficient of lift is at its greatest along the range of zero to fifteen degrees AoA.

Vol. 2025 (11) 889 – 895
ISSN 2688-3651 [online]

4. Discussion

4.1 Results Analysis

The testing results showed that the software workflow described in the methods section will allow aerospace engineers to quickly and easily design and test a wing using CAD, programmatic automation, and CFD. The software workflow could cheaply and rapidly provide definitive and detailed results for the ideal AoA(between zero and 15 degrees) for a NACA 2412 wing traveling at Mach 0.8. They can certainly be compared to running actual tests using real mock-up wings and wind tunnels because the wing mesh generated by using the Python Script, PTC Creo, and GMSH was very detailed, and the simulation technique applied by SU2 used highly sophisticated solving methods and took into account a great variety of different factors to produce the most accurate results possible from a digital testing solution.

This testing showed that the Python script and PTC Creo vastly lower the cost and effort of creating a tangible wing model by feasibly replacing such tedious processes with computer-aided design techniques that can automatically create and store wing designs electronically at a meager cost with very little space. The automation script developed in this study allows designers to not only quickly create wing models parametrically but also change them to their specifications in only a matter of seconds.

Further, this testing illustrated how simulation software like SU2 can eventually completely replace wind tunnel testing as its fidelity continues to increase with improved computing machines and better mathematical models. This will save thousands of hours and millions to tens of millions of dollars and resources such as power, space, and workers if a large-scale project uses the methods outlined in this paper.

4.2 Future Work

Although the design tool outlined in this paper greatly quickens the process of designing and testing wings, its functionality could be further optimized through iterative shape optimization, where the software workflow would be used iteratively in conjunction with genetic optimization algorithms to specify wing designs to preferences set by engineers automatically.

Genetic algorithms must be applied to the multi-objective shape optimization problem to create a functional shape optimizer. Genetic algorithms are a type of optimization algorithm, part of a wider group of algorithms called evolutionary computation since they imitate processes of natural selection and evolution found in nature (Goldberg, 1987). This algorithm's level of randomization can be set by the creator, which allows for a more robust and accurate optimization process (Carr, 2014). However, the random nature of genetic optimization problems that allows them to optimize accurately is the same reason why they may be slower to converge on a global optimum (Keane et al., 2007; Holst, 2012).

Although infrequent, approaches involving automatic optimization methods similar to that of this project have been discussed in the past in which the combined use of CAD, mesh generation, CFD, and genetic optimization algorithms are provided as separate elements that can be used together to achieve a system for aerodynamic shape optimization (Jameson et al., 2002; Keane et al., 2007). In one example, aerodynamic optimization requires the combination of automatic search and optimization procedures, where genetic algorithms are briefly described in the application of fully automatic design methods (Jameson et al., 2002). In another, the steps of CAD design, mesh generation, CFD testing, and multiple optimization methods (not just genetic) are listed but not implemented (Keane et al., 2007).

The main gap in these solutions and descriptions is that none mention integrating these tools for aerodynamic shape optimization through a software pipeline similar to the one discussed in this paper, which automates the entire optimization process, from creating the initial mesh to a final product. Such a tool could further improve the speed of wing design processes in the aerospace industry.

5. Conclusion

In conclusion, the data collected from this research exhibited the ability of generative CAD, high-fidelity CFD

Vol. 2025 (11) 889 – 895
ISSN 2688-3651 [online]

simulations, and programmatic automation to synergistically assist aerospace engineers in crafting optimal airplane wings catered to specific flight conditions and constraints because the software workflow allowed the configuration phase of aircraft design to be automated and digitized, therefore reducing costs and engineers' effort while also increasing efficiency.

References

Cantwell, B. J. (2013, May 12). The NACA Airfoil Series. https://web.stanford.edu/~cantwell/AA200_Course_Material/The%20NACA%20airfoil%20series.pdf

Carr, J. (2014). An introduction to genetic algorithms. *Senior Project*, 1(40), 7.

Creopyson. (n.d.). *Creopyson Documentation*. https://creopyson.readthedocs.io/en/latest/modules.html

Econmon. (2014, June 14). SU2 Config File. *SU2*. Retrieved September 10, 2023, from https://github.com/su2code/Tutorials/tree/master/compressible_flow/Turbulent_ONERAM6/mesh_ONERAM6_turb hexa_43008.su2

Geuzaine, & Remacle. (2009). GMSH. *Gmsh*. Retrieved September 10, 2023, from https://gmsh.info/ Goldberg, D. E., & Richardson, J. (1987, July). Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on

Genetic Algorithms (Vol. 4149). Hillsdale, NJ: Lawrence Erlbaum.

Jameson, A., Martinelli, L., & Vassberg, J. (2002, September). Using computational fluid dynamics for aerodynamics - a critical assessment. In *Proceedings of ICAS* (pp. 2002-1).

Keane, A. J., & Scanlan, J. P. (2007). Design search and optimization in aerospace engineering. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 365(1859), 2501-2529.

Litherland, B. (2015, July 1). Using VSPAero. *OpenVSP*. https://openvsp.org/wiki/doku.php?id=vspaerotutorial NumPy. (n.d.). *NumPy Documentation*. https://numpy.org/doc/stable/

Torenbeek, E. (1976). Synthesis of Subsonic Airplane Design: An Introduction to the Preliminary Design of Subsonic General Aviation and Transport Aircraft, with Emphasis on Layout, Aerodynamic Design, Propulsion and Performance (pp. 3-9). Springer.

Various Authors. (2023). SU2. *SU2*. Retrieved September 10, 2023, from https://su2code.github.io/docs_v7/home/

Weiss, S. I., & Amir, R. (2023, June 24). Aerospace industry. *Encyclopedia Britannica*. https://www.britannica.com/technology/aerospace-industry