Solid Freeform Fabrication 2023: Proceedings of the 34th Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference
Reviewed Paper

G-WING: A NOVEL SOFTWARE TOOL FOR TOOLPATH-CENTRIC DESIGN OF WINGS FOR MATERIAL EXTRUSION

Justin D. Valenti*‡, Joseph Bartolai†, and Michael A. Yukish*

*Department of Aerospace Engineering
†Applied Research Laboratory
The Pennsylvania State University, University Park, PA

Abstract

A novel software tool for the design of small aircraft wings to be fabricated with material extrusion is presented where the key requirement of the tool is to minimize the time from identified need to realized capability. The tool, named G-Wing, uses rapid design algorithms based on lifting line theory to determine the outer-mold line of the wing based on desired aerodynamic behavior. The resulting wing shape and flight-load distribution are given to a structural design algorithm to determine the internal structure of the wing based on both expected flight loads and manufacturing constraints. Finally, manufacturing instructions in the form of G-Code are created directly from the wing shape and internal structure. This process removes explicit geometric modeling and slicing from the critical design path and directly converts airfoil coordinates to perimeter G-Code points, minimizing the introduction of geometric error. This process has been used to design and fabricate multiple small aircraft wings that have successfully flown. G-Code for an example wing section is shown to be lighter and require less build time compared to G-Code generated by a standard CAD-slicing toolchain.

Introduction

Since 2015, a steady line of research exploring the design of Uncrewed Aerial Vehicles (UAVs) to be fabricated with polymer Material Extrusion (MEX) has existed at Penn State [1–8]. From the beginning, it was obvious that the decision to additively manufacture the aircraft changes the design process [1, 2]. Furthermore, there was a desire to rapidly design and iterate on AM UAVs, which lead to development of the design philosphy "Operational Responsiveness" (OpRes) [1]. The OpRes philosophy states that the time from identified need to realized capability is minimized when the design space is constrained to the intersection of available product architectures, design tools, and manufacturing methods, shown in Figure 1. "Design assist tools" can be developed for a particular problem which guide and constrain a decision maker produce design within this intersection.

The traditional AM workflow involves creating a geometric CAD model, then giving that model to a *slicer* which translates the geometry into instruction for the AM machine. Presumably, prior to the CAD stage, there are design tools, studies, rules of thumb, and designer's intuition that are used to generate the design, who's geometry is then instantiated in CAD. This complete workflow is shown in Figure 2. This process works well for many applications. Aircraft wings are *not* one of these applications.

‡Corresponding Author: jdv5076@psu.edu

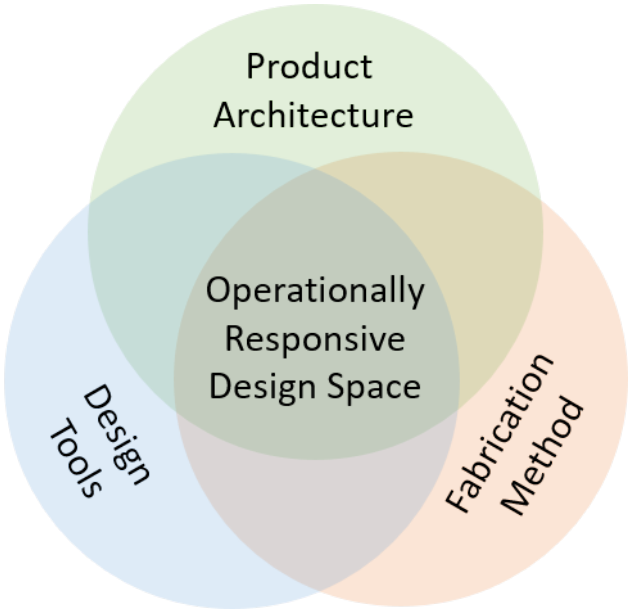


Figure 1: *The idea of designing for operational responsiveness focuses on constraining the design space to the intersection of available manufacturing methods, design tools, and product architectures.*

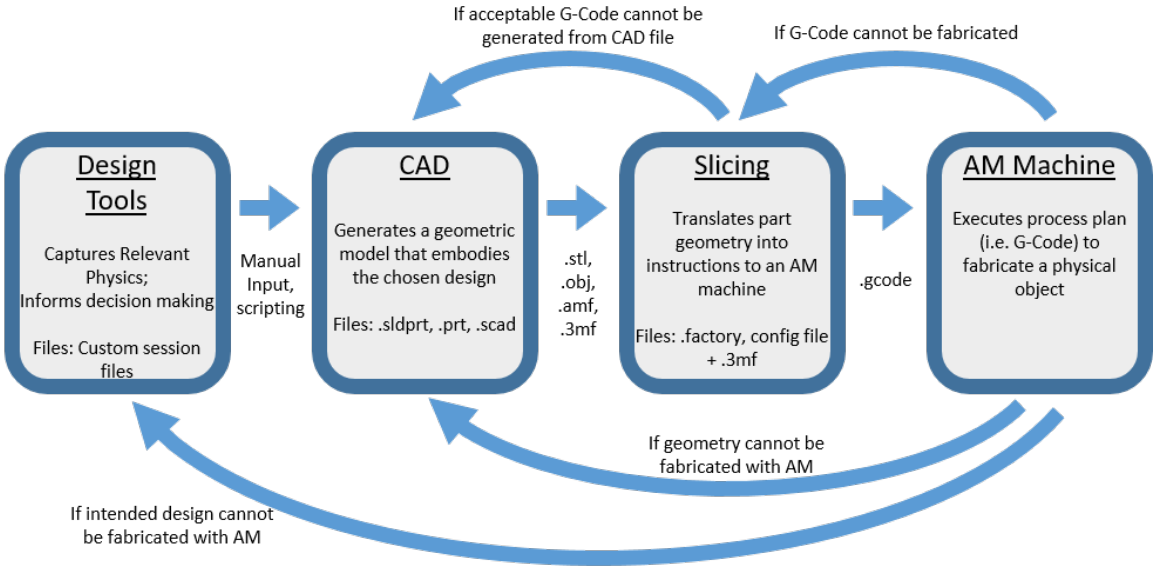


Figure 2: *Traditional AM Workflow*

The strict strength to weight requirements of aircraft structures, combined with the minimum feature size of an MEX machine, tend to lead to single extrusion sparse structures. Not only are these types of structures difficult for MEX machines to fabricate, slicers struggle to generate a "good" path plan for the single extrusion structures. The wing geometry needs to be generated with an understand of how the final G-Code is generated to ensure that the final wing matches the designer's intent. Two publicly available examples demonstrate this. First, the STL files of the freely available Eclipson Airplane [9] have strategically place

voids within the wing structure. This manipulates the slicer into creating a part only with single extrusion walls. The wing is designed to be sliced with 0% infill and for the perimeter extrusions to form both the skin of the wing and the internal structure. Second, the company 3D Lab Prints [10] sells the files to build aircraft on Desktop 3D printers. They provide the geometry in several file types and specific instructions on how to slice the geometry in several available slicers. Both examples show clear engineering thought into not only the shape of the aircraft, but how the aircraft is converted into G-Code.

In aircraft structures, failure to invest in the toolpath design will likely lead to a poor end product. It is important to consider upfront when designing a wing and its internal structural how that structure will be translated into a coherent toolpath. In short, for MEX-produced wings, the wing design problem is really a toolpath design problem. The rest of this paper details a fast, efficient, and holistic solution to that design problem.

Novel Contribution: A holistic toolchain was developed to design UAV wings to be built with MEX. This toolchain was instantiated in a software tool named G-Wing. The software tool connects low-order aerodynamic, structural, and process design algorithms to automate the design process from desired aerodynamic behavior to ready-to-make G-Code that guarantees flight ready wings. This process bypasses the CAD and slicing stages of the tradition AM toolchain, minimizing the introduction of errors, and ensuring that the designers intent is carry all the way to the finished product.

Layout of Paper: This manuscript first provides a detailed description of the G-Wing functionality and underlying design modules. We then provides a comparison of the G-Wing toolchain to a more conventional MEX toolchain. A discussion section explores the broader implications of this work. Finally, a conclusion section summarizes the work.

Description of G-Wing

G-Wing is a custom software tool written in Python 3. G-Wing designs wings using a recently publish design pattern specifically developed for wings build with MEX [11]. The tool consisted of 3 main modules: aerodynamics, structures, and process planning. The tool ingests a text-based configuration (config) file in which the user defines all aerodynamic, structural, and process design parameters and settings (Figure 3). The main output of G-Wing is a collection of .gcode files ready to be given to an MEX machine.

Aerodynamic Design: The main output of aerodynamic design module is the outer mold line of the wing and the spanwise lift distribution across the wing. Lifting line theory [12,13] provides a fast method to analyze wings. The authors have publish several algorithms [14,15] to perform the inverse design of wings using the matrix formulation [16] of lifting line theory. These algorithms are used in G-Wing to allow the user to generate wings to yield a desired aerodynamic behavior. A brief overview of the model will now be provided.

In lifting line theory a 3D wing is modeled as 1D straight "lifting line" which runs in the spanwise (y) direction from the left wingtip to the right wingtip, as shown in Figure 4. This formulation is valid for straight wings with no sweep or dihedral. The outer mold line of the wing is uniquely defined by an airfoil shape*, a continuous distribution of chord as a function of y, $\tilde{c} = \tilde{c}\,(y)$, and a continuous distribution of twist of a function of y, $\tilde{\alpha} = \tilde{\alpha}\,(y)$. In the

*The present work assumes a constant airfoil shape, but the work is easily generalized to allow for a varying airfoil across the span.

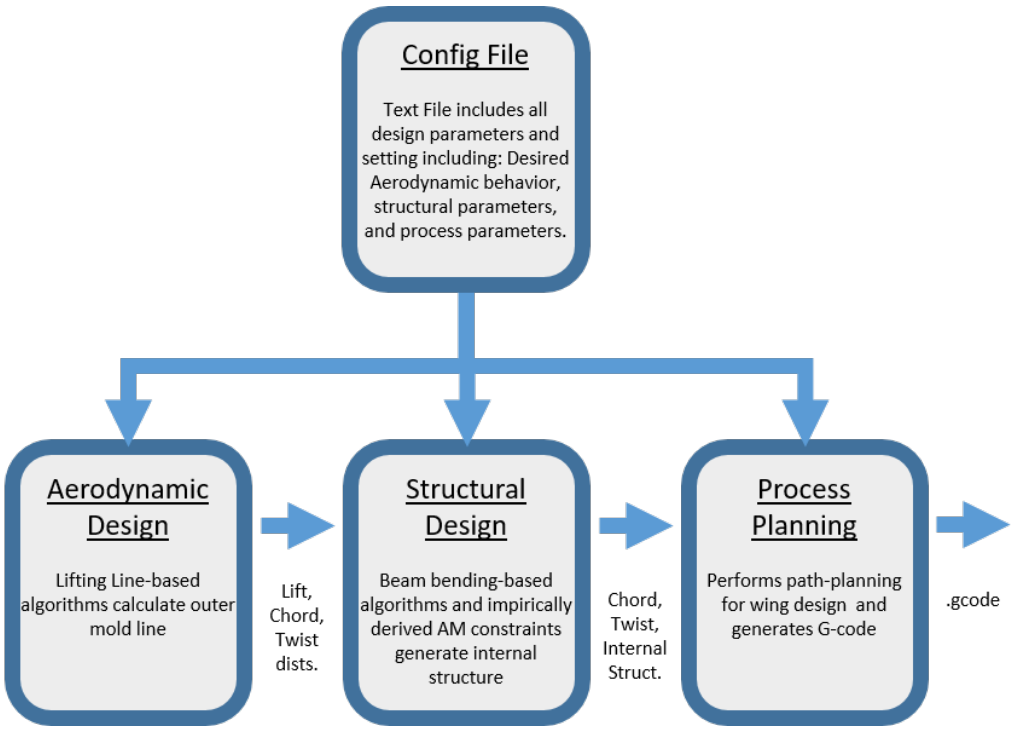


Figure 3: Workflow of G-Wing design modules

matrix formulation, the lifting line is discretized into n equally spaced spanwise stations. The chord and twist distributions are then approximated a $n \times 1$ column vectors where each element of the vector represents of local value of chord or twist: $\tilde{c}(y) \approx \vec{c} = [c_1, c_2, \ldots, c_n]^T$, $\tilde{\alpha}(y) \approx \vec{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_n]^T$. This discretization is shown in Figure 4.

While not pictured, the smooth airfoil shape (i.e. the cross section of the wing) is represented by a list of (x, z) coordinates. This list is stored as a $m \times 2$ matrix, where m is the number of airfoil coordinates: $P_{af} = [x_1, z_1; x_2, z_2; \ldots; x_m, z_m]$. G-Wing does not have the ability to perform airfoil design. The airfoil shape is provided up from by the user.

In the matrix lifting line formulation, the wing is represented by n adjacent "horse shoe vortices". Each horseshoe vortex is fixed to the lifting line, centered about a y location, with tails trailing off infinitely far downstream, as shown in Figure 4. At an arbitrary y location, the local lift ℓ is directly related to the vorticity Γ at that location via the Kutta-Jouskowski theorem, $\ell = \rho V_\infty \Gamma$. Following the discretization scheme of chord and twist, the continuous lift and vorticity distributions are also approximated as $n \times 1$ vectors: $\vec{\ell} = [\ell_1, \ell_2, \ldots, \ell_n]^T$ and $\vec{\Gamma} = [\Gamma_1, \Gamma_2, \ldots, \Gamma_n]^T$, respectfully.

Several design algorithms are available to the user in G-Wing. Of which, the user chooses one to design the wing. This first design algorithm use the basic lifting line calculation [16] which calculates a vorticity distribution $\vec{\Gamma}$ from the wing geometry: $\vec{c}$ and $\vec{\alpha}$. This allows a designer to explicitly define the outer mold line up front. G-Wing then calculates the aerodynamic behavior.

The next two algorithms, formally presented in [14], allow a user to specify a lift distribution $\vec{\ell}$ and either a chord distribution $\vec{c}$ and a twist distribution $\vec{\alpha}$. G-Wing will the calculate either $\vec{c}$ or $\vec{\alpha}$, whichever was not specified.

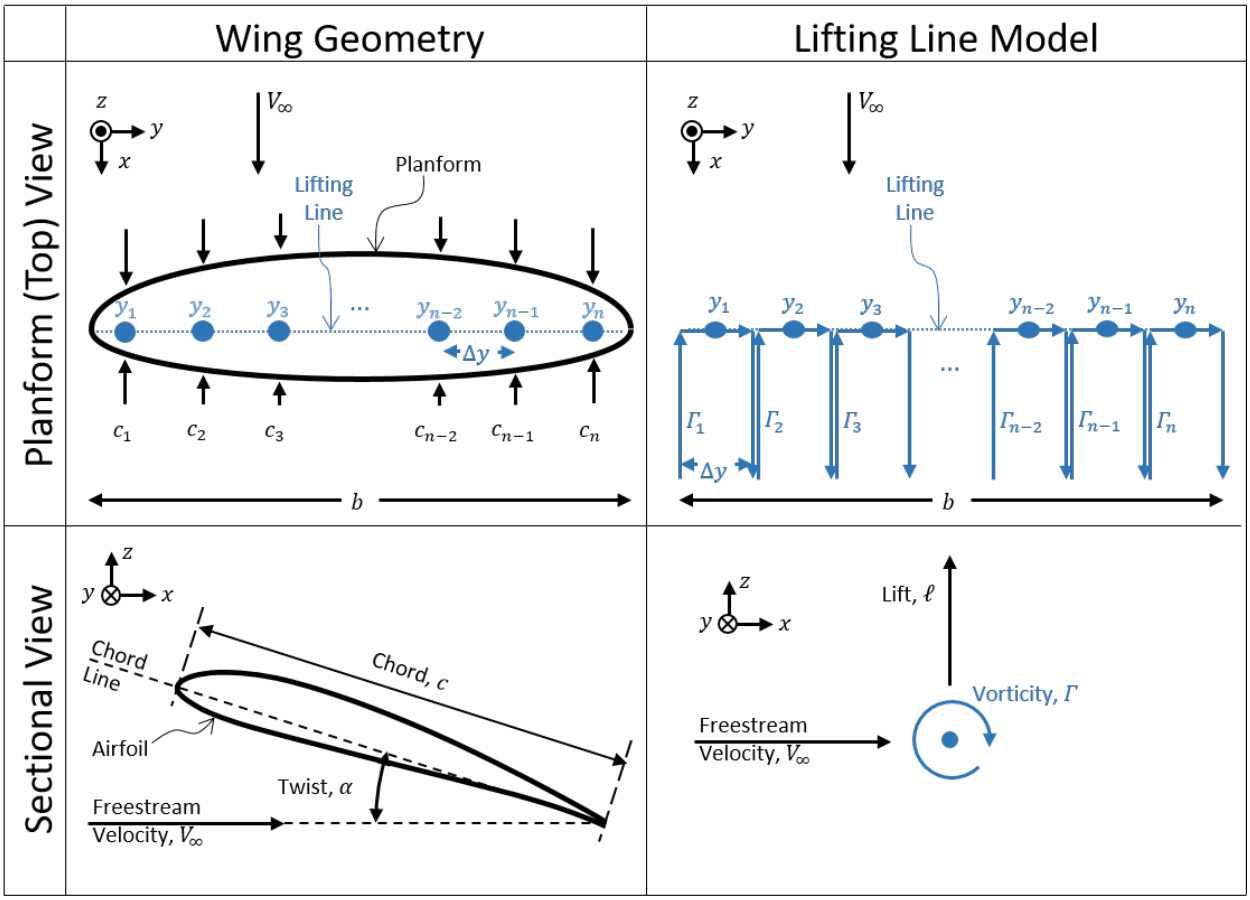


Figure 4: Diagrams of the definition of external wing geometry (left) and how the wing is modeled in lifting line theory (right). Lifting line theory collapsed the 3D wing geometry into a 1D lifting line.

It is worth noting that the lift distribution will change as the wing changes total lift coefficient, defined as $c_L = 2L/\left(\rho_\infty V_\infty^2 S\right)$, where L is the total lift over the entire wing, ρ_∞ is the freestream air density, and S is the wing area. Wings operate over a range of c_L values, typically trading off between V_∞ and c_L. High c_L values correspond to slower flight and vice versa. The prior three designs allow a user to design a wing for the aerodynamics of a single c_L

Finally there are three algorithms that allow the user to design a wing by defining the aerodynamic behavior at two c_L values. The first, presented in [14], allows a user to define two lift distributions, $\vec{\ell}_1$ and $\vec{\ell}_2$, each at a different c_L, and simultaneously calculate the wing chord and twist. The final two algorithms, presented in [15], involve designing a particular spanwise distribution of local 2D lift coefficients $\vec{c}_\ell = [c_{\ell 1}, c_{\ell 2}, \ldots, c_{\ell n}]^T$. At the j^{th} spanwise location, the 2D lift coefficient is defined as $c_{\ell j} = 2\ell_j/\left(\rho V_\infty^2 c_j\right)$. Algorithms are available to calculate chord and twist from (1) two local lift coefficient distributions, $\vec{c}_{\ell 1}$ and $\vec{c}_{\ell 2}$, each at a different c_L and (2) one lift distribution, $\vec{\ell}_1$, and one lift coefficient distribution, $\vec{c}_{\ell 2}$.

A summary of all available aerodynamic design algorithms is provided is in Table 1. Formal mathematical derivations and instructions on how to implement the algorithms are

provided in [14, 15]. Regardless of which algorithm is used, at the end of the aerodynamic design routine, the wing geometry is full defined by the a chord distribution $\vec{c}$, a twist distribution $\vec{\alpha}$, and a user defined airfoil. The flight loads are defined by the lift distribution $\vec{\ell}$. As a note, in the case of designing a wing for two c_ℓ distributions, the results need to be run through the direct design algorithm to obtain a lift distribution.

Table 1: Six design algorithms are available in G-Wing, of which the user would chose one to perform the design. The first three algorithms design a wing for a single c_L, the second three algorithms allow a designer to define the aerodynamic behavior at two different c_L values.

Design Case	User Defined Inputs	Output(s)
Direct Design	$\vec{c}$, $\vec{\alpha}$	$\vec{\ell}$
Single Point Inverse Design #1	$\vec{c}$, $\vec{\ell}$	$\vec{\alpha}$
Single-Point Inverse Design #2	$\vec{\alpha}$, $\vec{\ell}$	$\vec{c}$
Two-Point Inverse Design #1	$\vec{\ell}_1$, $\vec{\ell}_2$	$\vec{c}$, $\vec{\alpha}$
Two-Point Inverse Design #2	$\vec{c}_{\ell 1}$, $\vec{c}_{\ell 2}$	$\vec{c}$, $\vec{\alpha}$
Two-Point Inverse Design #3	$\vec{\ell}_1$, $\vec{c}_{l2}$	$\vec{c}$, $\vec{\alpha}$

Since the lifting line model collapses the wing to a straight line, there is some ambiguity on how non-linear chord and twist distributions are converted into 3D geometry. This is handled in G-Wing by aligning all airfoil sections about a potential hinge line for a control surface. The chordwise location of the this straight line, and thus the resulting hinge-line, is a user-defined parameter. The conversions chord and twist distributions into a wing planform is shown Figure 5.

It is interesting to note that the outer mold line of the wing is fully defined by the chord and twist distributions, both in the form of $n \times 1$ vectors, and an airfoil profile in the form of (x, z) coordinate list. If the wing is discretized into 400 spanwise stations and the airfoil coordinate list contains 40 (x, z) points, both reasonable values, then the entire wing geometry is defined by 880 floating point numbers. At this point there is no explicit geometric modeling or tessellation because it is not needed at any point from design to G-Code.

G-Wing can optionally call Athena Vortex Lattice (AVL) [17] for aerodynamic verification. If the designer opts for AVL comparison in the config file, G-Wing will output the wing geometry to AVL for higher-order aerodynamic analysis. The aerodynamic behavior predicted from the lifting line theory of G-Wing is then automatically compared to the behavior predicted by the vortex lattice theory of AVL.

Structural Design: Given the outer mold line, the structural design model determines the layout of the internal structure. Previous work by the authors' [4] propose a family of structure known as "curvilinear spars" that allow the internal structure to be tailored for a desired bending stiffness. The concept was inspired by work on "spaRibs" from Virginia Tech [18] and incorporated the concept of "cross-sectional analysis" [19] to simplified the design process.

The wing is built "standing up" with the build direction oriented with the spanwise direction. Each layer consists of the airfoil shape and several vertical members, i.e spars, as shown in Figure 6. At any given spanwise location, the number and chordwise (x) locations

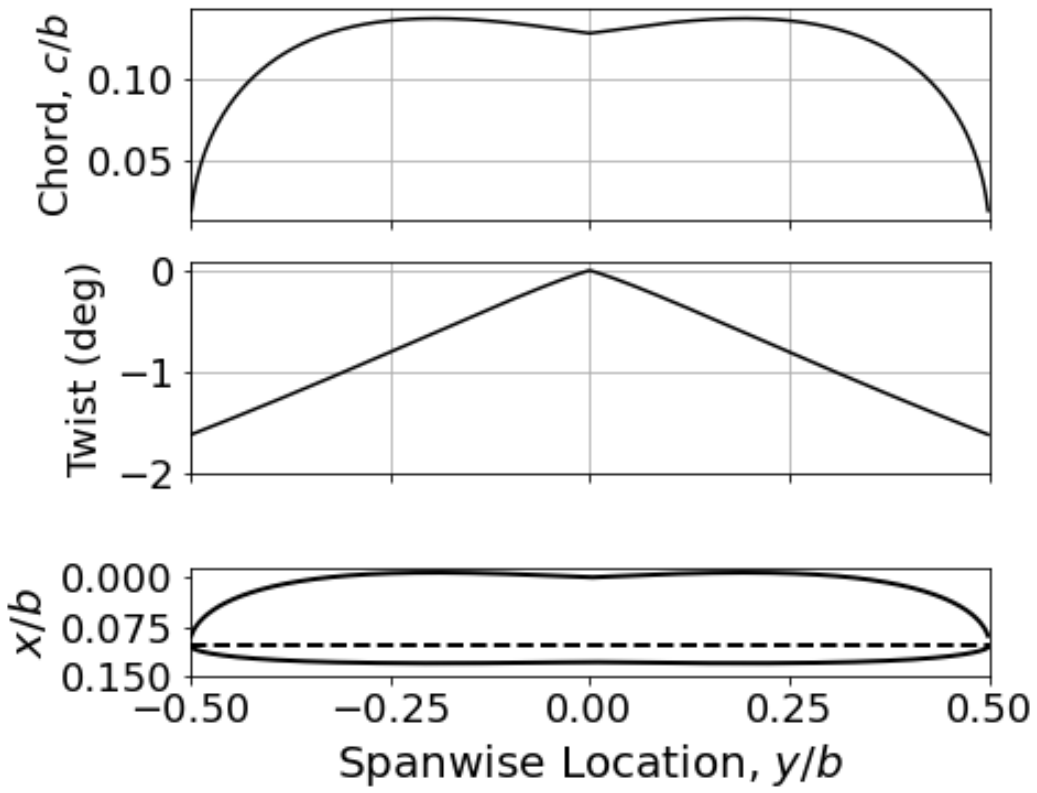


Figure 5: *Example wing (bottom) defined by chord and twist distributions (top two plots). The dashed line in the bottom plot corresponds to a straight hinge line at 80%chord.*

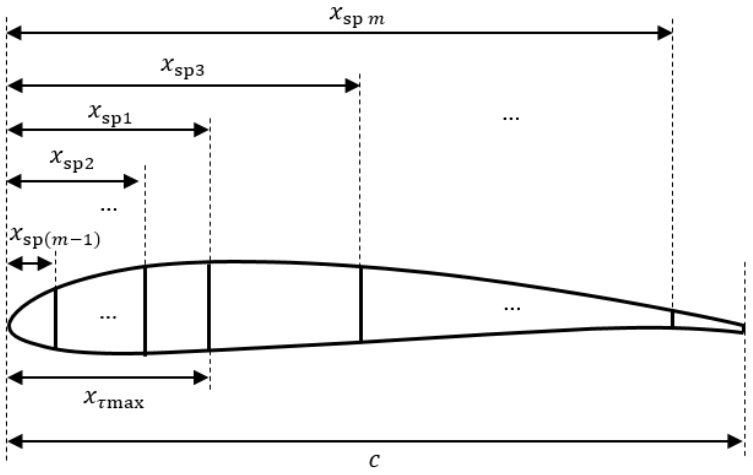


Figure 6: *Sectional view showing an airfoil with several spars at various chordwise (x) locations. The "main" spar is placed at the airfoil max thickness location* $x_{\tau\mathrm{max}}$. The number and locations of the spars fore and aft of the main spar are determined by bending inertia requirements and AM constraints.

of the spars chosen to either increase the bending inertia of the wing to carry the flight

loads [4] or support the skin such that it does not deform during manufacturing [7, 8]. The internal structure of the wing can then be uniquely defined by an $m \times n$ "spar location matrix":

$$X_{sp} = \begin{bmatrix} x_{\text{sp}11} & x_{\text{sp}12} & \dots & x_{\text{sp}1n} \\ x_{\text{sp}21} & x_{\text{sp}22} & \dots & x_{\text{sp}2n} \\ \dots & \dots & \dots & \dots \\ x_{\text{sp}m1} & x_{\text{sp}m2} & \dots & x_{\text{sp}mn} \end{bmatrix}.$$

A single row of the X_{sp} matrix represents the chordwise locations of a single spar across the span. A single column represents the chordwise locations of all spars at a single spanwise location. The reader is directed the previous cited work for the exact details on of the these two design considerations, but a brief pseudo-code of the design process is given in Algorithm 1.

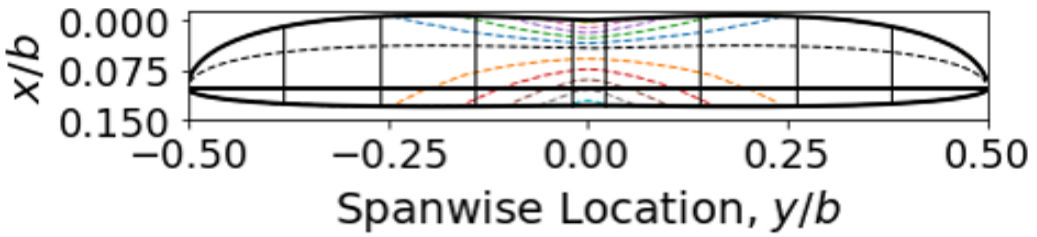
Algorithm 1: Pseudocode describing the steps to generate the spar locations form a wing load, wing geometry, and AM constraints

Input: Wing Design: {Chord distribution $\vec{c}$, Lift distribution $\vec{l}$, Airfoil Coordinates P_{af}}
Machine Settings: {extrusion width, material yield strength σ_y}
Output: Spar Location Matrix X_{sp}

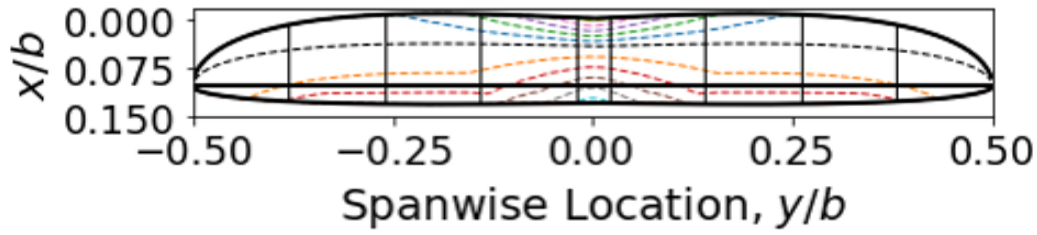
1 Locate the chordwise location of the max airfoil thickness. This becomes the location of the "main spar";
2 Calculate the bending moments on the wing from the lift distribution provided by the aerodynamic module;
3 Calculate the bending inertia required to carry the bending moments;
4 Place spars to provide the required bending inertia according to [4];
5 Limit spar spacing based on AM limits based on [8];
6 Write the spar locations to the matrix X_{sp}

Two example wing structures are provided in Figure 7. The structure shown in Figure 7(a), is example wing structure that was generated to only carry the flight loads with no consideration of supporting the wing skin. This corresponds to the structure that is generated by Step 4 of in Algorithm 1. The structure given in Figure 7(b) shows the type of structure that results from taking the structure shown in 7(a) then limiting the max spar spacing to properly support the wing skin. This corresponds to the structure that results from Step 5 in Algorithm 1. It can be seen in the aft portions of the wing, around spanwise locations of roughly $\pm 0.15 y/b$, that the curvature of the spars distinctly changes. Inboard of this location, spar locations are fully designed to carry flight loads. Outboard of this region, the fore spars are still designed to carry the flight loads, however the aft spars are designed to support the skin during manufacturing.

The main output of the structural design module is the spar spacing matrix X_{sp}. This matrix, along with the chord and twist distributions found in the aerodynamics module fully



(a) *Planform view of curvilinear spar structure designed for a particular bending inertia to carry the expect flight loads*



(b) *Planform view of curvilinear spar structure with the chordwise spar spacing limited by AM constraints*

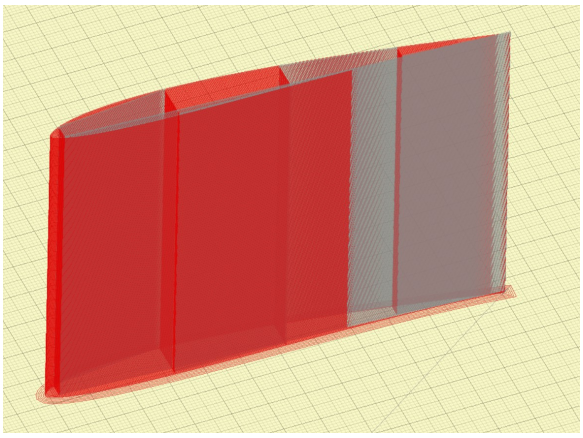
Figure 7: Planform view of example curvilinear spar structures both (a) without and (b) with AM spar spacing constraints. In both structures, the black dashed line corresponds to the "main" spar placed along the max thickness points of the airfoil sections. The remaining dashed lines correspond to the other curvilinear spars. Also visible is the hinge line running between the wingtips (explained in the Aerodynamics section) and the wing section breaks (explained in the Process Planning section).

define the shape of the wing and are provided to the process planning module to generate the G-Code.

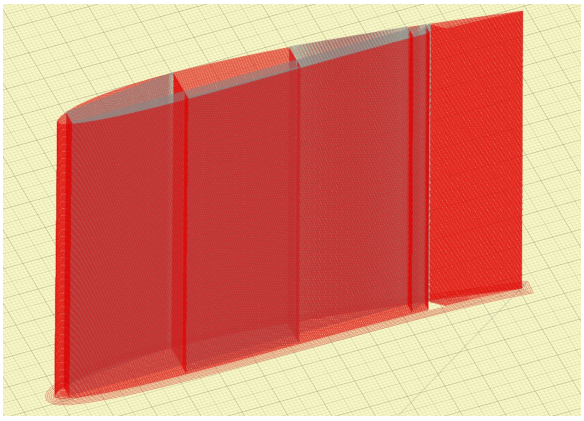
Process Planning: With the outer mold line defined in the aerodynamic module and the internal structure defined in the structural module, the process planning module is tasked with generating the instructions to the MEX machine that represents the final design of the wing. MEX parameters such as build volume dimensions, extrusion width, layer height, build speeds and temperatures, etc. that are user-defined in the config file are used by the process planning routine to translate the airfoil, chord, twist, and spar spacing distributions into build-able G-Code files.

Wingspans are typically too large to be fabricated in one build. The process planning module automatically breaks the wing into smaller sections based on build volume. The wing sections are built "standing up", with the spanwise direction of the wing corresponding to the build direction of the MEX machine[†], as shown in Figure 8. In this orientation, the span of a single wing section is limited by a user-defined max build height,$z_{\max}$. This parameter can either correspond to the height of the build volume itself or, more likely, the max build height that can be achieved before the tall, thin wing section begins to wobble. The first

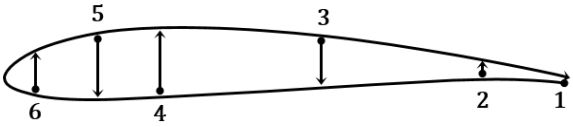
[†]Please note that standard aircraft coordinates define the spanwise direction as the y direction. In AM literature, the build direction is referred to as the z direction. These are aligned in the build orientation used in this work.



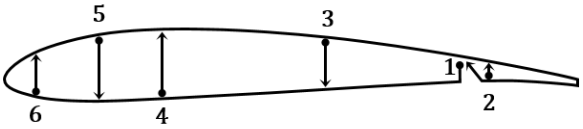
(a) *Example G-Code for a simple wing section*



(b) *Example G-Code for a wing section with a plain flap.*



(c) *Example toolpath for a simple wing section*



(d) *Example toolpath for a wing section with a plain flap.*

Figure 8: *Example wing section and toolpath used to build MEX wings. (a) and (b) Illustrations of wing section printed "standing up" on the build plate, red: extrusion moves, gray: travel moves. (c) and (d) Diagrams of the example toolpaths for a given layer to create either simple wing sections (c) or wing sections with a plain flap (d).*

wing section to be identified is the center section. The user defines a center section width w_c, which could correspond to the width of a fuselage. After the center section is defined, the remain wingspan is broken into wing sections with a span of

$$b_{\text{sect}} = \frac{(b - w_c)}{\text{ceil}\left((b - w_c)/z_{\max}\right)}. \tag{1}$$

An example of these wing section breaks are shown in Figure 7.

With this build orientation, the perimeter points for each layer corresponds to the airfoil coordinates. The size and angle of the airfoil is provided by the chord and twist, which was determined in the aerodynamic design module. The infill locations are determined by the spar spacing found in the structural design module. Example tool paths are shown in Figure 8. One exception to the infill is that solid infill (i.e. spars with no space in between) is used to the first layer of each wing section. This both improves the build plate adhesion and serves as a wing rib in the final structure. Pseudocode for the G-Code generation is provided in Algorithm 2.

The process planning module outputs a series of G-Code files, one for each wing section. These G-Code files represent the final design of the wing. The entire toolchain takes less than 5 minutes to run on a typical laptop.

Algorithm 2: Pseudocode describing the steps to convert the wing design defined by the chord and twist distributions across the span, the chordwise and spanwise locations of the spars, and the airfoil coordinates into .gcode files for the MEX machine.

Input: Wing Design: {Chord distribution $\vec{c}$, twist distribution $\vec{\alpha}$, Spar Locations $\vec{X}_{\text{sp}}$, Airfoil Coordinates P_{af}}
Machine Settings: {extrusion width, layer height, max build height, toolhead speeds, Custom start/end scripts, etc}

Output: .gcode files for the wing

```
Split wing in smaller "wing sections" based on max build height;
for each wing section do
    Open new .gcode file for wing section;
    for each layer do
        Set z-height for layer and find correspond spanwise location on wing;
        Interpolate chord, twist, and spar spacing values;
        Use local chord and twist value to scale and rotate airfoil coordinates,
         translate coordinates to desired location build plate;
        Generate G-Code commands for transformed airfoil coordinates;
        Write airfoil G-Code commands to .gcode file;
        if 1st Layer then
            Solid Infill ;
        else
            Determine spar locations based on local spar spacing;
        end
        Generate Write G-Code commands for internal structure;
    end
    Output G-Code File for wing section;
end
```

CAD File Generation: G-Wing removes a CAD tool from the critical path to G-Code. Nonetheless, CAD modeling can still be useful when incorporating the wing into a broader aircraft. G-Wing can optionally generate an OpenSCAD [20] model of the wing. Notably, the process to generate the .scad file directly from the aerodynamic and structural design results and is *fully independent* of the G-Code generation process.

The independence of CAD and G-Code generation means that both are theoretically exact representations of the intended design. One is not a interpretation of the other. There is no loss of designers intent on the path to G-Code.

OpenSCAD was chosen as the CAD software for two reasons. First, it is an open-source software that is freely available online. Second, the file format is purely text-based. As a result, G-Wing needs to only generate an additional text-based .scad file. The user would then have option to open this .scad file in the OpenSCAD after G-Wing finishes running.

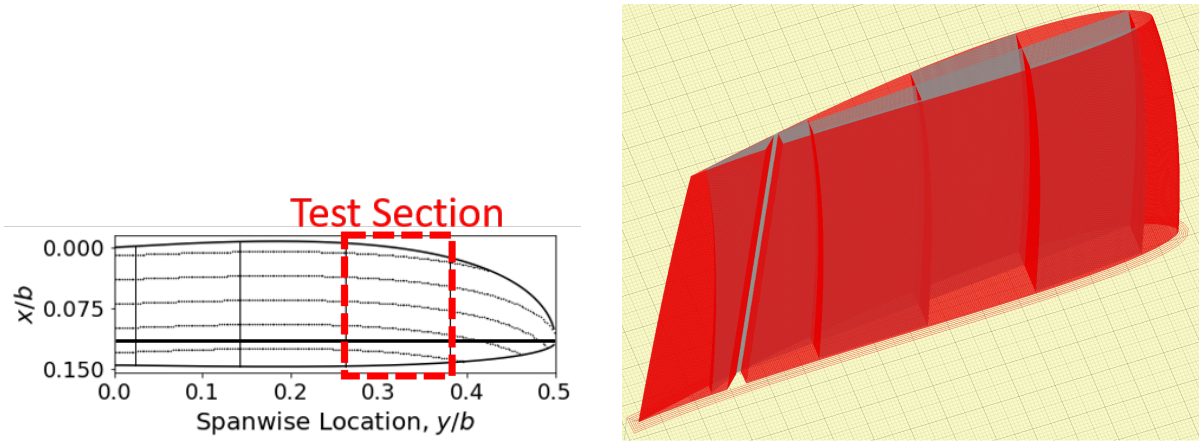
This meaning that that CAD output does not interrupt the design flow.

Comparison to Conventional Tool Chain

A comparison of G-Code generated with G-Wing to that generated with a more conventional AM toolchain was performed to explore the benefits of G-Wing. A wing section was chosen for this comparison from a wing designed for the Kite, shown in Figure 9, a small UAV designed as a test-bed for MEX-built wings. The wing itself has a factor of safety of greater than 20 and the spar spacing is determined by manufacturing constrains everywhere along the span. The wing and selected test section is shown in Figure 10.

(a) *Kite with 3 wings of different designs* (b) *Kite in flight*

Figure 9: *The Kite is an MEX-produced UAV, which was designed as a test-bed for MEX wings.*



(a) *Planform view of wing with test section* (b) *G-Code of test section generated by G-Wing boxed in red*

Figure 10: *Test wing section chosen for study.*

A summary of the wing parameters is provided in Table 2. The MEX parameters used to generate the G-Code is provided in Table 3. An OpenSCAD model of the test section was also generated by G-Wing, which was in turn exported as an STL file. The open-source slicing software Slic3r [22] was used to slice the wing section STL with several common infill settings. G-Code files were generated with 5% and 20% infill fractions for grid, honeycomb,

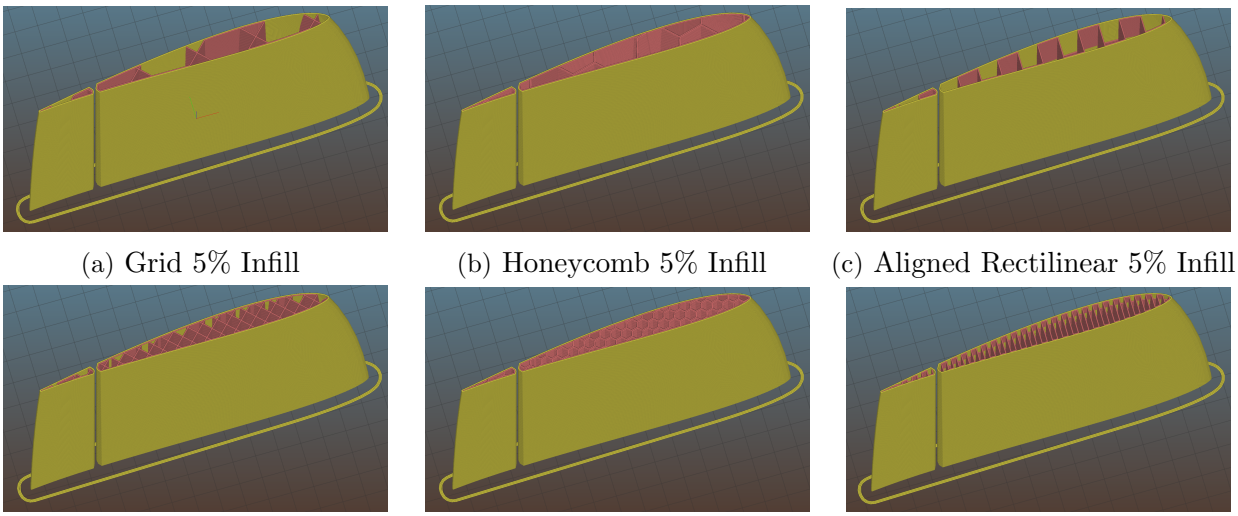
Table 2: Wing parameters

Wing Parameter	Value
Airfoil	AG24 [21]
Wingspan	1.09 m
Aspect Ratio	8
Spar Spacing	30 mm

Table 3: Build Parameters

FFF Parameter	Value
Nozzle Diameter	0.4 mm
Layer Height	0.2 mm
Extrusion Width	0.5 mm
Material Density (PLA)	1240 kg/m^3
Perimeter Speed	40 mm/s
Infill Speed	40 mm/s
Solid Infill Speed	60 mm/s
1st Layer Speed	30 mm/s
Travel Speed	80 mm/s

and aligned rectilinear sparse infill patterns. The build parameters listed in Table 3 were also used for slicing in Slic3r for an apples-to-apples comparision with G-Wing. Screenshots of the 6 G-Codes generated with infills in Slic3r are provided Figure 11. A seventh G-Code



(a) Grid 5% Infill (b) Honeycomb 5% Infill (c) Aligned Rectilinear 5% Infill

(d) Grid 20% Infill (e) Honeycomb 20% Infill (f) Aligned Rectilinear 20% Infill

Figure 11: Wing sections sliced in Slic3r with infill.

file was generated in Slic3r with no infill as a baseline to show the effects of infill on the build.

Both G-Wing and Slic3r predicts part mass when G-Code is exported. The build times of wing sections were predicted using the open-source software Pronterface [23]. The predicted mass and build times of all G-Code files are provided in Table 4. Considering mass first, G-Wing is about halfway between the three 5% infill sections and the 0% infill section. This immediately indicates the how sparse these wing structures are. Slic3r's manual suggests infills of at least 20% for any part where strength is important. Unfortunately, the 20% infill sections over twice the weight of the G-Wing sections.

The build time comparison is also very interesting. The G-Wing section is only marginally longer than the 0% infill section. This highlights the efficiency of the G-Wing toolpath. By

Table 4: Mass and build time values of the wing section generated by G-Wing and wing sections generated by Slic3r

Metric	G-Wing	Slic3r 0% Infill	Slic3r 5% Infill			Slic3r 20% Infill		
			Grid	Honey-Comb	Aligned Rect.	Grid	Honey-Comb	Aligned Rect.
Mass (g)	36.2	24.30	44.38	48.20	43.69	69.76	78.67	70.28
Build Time (h:mm)	2:10	2:07	4:11	4:18	3:14	5:39	6:12	4:53

contrast, the other wing sections take hours longer to print. Notably, the aligned rectilinear sections require significantly less time than the other wing sections of similar infill percentage. This suggests aligned rectilinear may be a good choice of infill if a designer is required to use a conventional tool chain.

Finally, it is worth looking qualitatively at the lattice infills. The G-Wing design is strong enough to carry the flight loads, which suggests infill percentages of any infill design for this section ought to be around 2-3%. At these infill percentages, an individual lattice cell does not fit within the wing section. The sparse lattice infills become a non-uniform collect of inefficient extrusions.

Discussion

From the AM-specific prospective, G-Wing solves several problems that appear in a typical AM toolchain. First, there is no loss in designer intent throughout the entire toolchain. In a tradition CAD-slicer toolchain, there are opportunities for the introduction of geometric error first when modeling the design in CAD, then again when the design is tessellated into an STL, then again when that tessellated shape is translated into G-Code. The development of G-Wing began by asking the question "what does a good toolpath look like?" Our answer to that question is provided in Figure 8. From there, the rest of the design methods were developed and link together in a way maintained the designers intent through the entire process.

A second AM-specific implication of G-Wing is that the process planning is completely deterministic. The same config file with always lead to the same G-Code. During a traditional slicing session, it is often difficult to know and/or decide exactly how infill patterns will be placed and exact what the pathplan will be. G-Wing solves that problem.

The restriction of this work is that G-Wing is application specific - the inevitable result of the focus on extremely short design cycles as the key performance metric. The software will only produce wings within a very specific design space and nothing more. Any additional feature is only achieved through some amount of development effort. For a different application, the decision has to be made whether or not the specific design problem at hand is repeated enough times to warrant a full engineering effort to develop the digital thread. This leads to the broader digital engineering implications of the work.

There is currently a large amount of enthusiasm surrounding "digital engineering" [24, 25]. A major concept in digital engineering is taking ownership of the toolchain of an engineering problem. G-Wing is a small example of a homegrown, fully-owned toolchain from initial design to complete manufacturing process plan. For another application, it

may make more sense to integrate a combination of homegrown and commercial-off-the-shelf tools. Regardless, the building and owning of that toolchain is a digital engineering problem. When this is done properly, the time from identified need to realized capability is minimized. G-Wing is an example of this - the tool itself designs a ready-to-build wing less than 5 minutes that will fly.

Conclusion

G-Wing is a holistic toolchain which automates the design from desired aerodynamic behavior to ready-to-build G-Code. The software tool allows a designer to simultaneously perform aerodynamic, structural, and process design. The resulting G-Code was shown to be significantly better than G-Code created by a more traditional toolchain using standard slicing settings.

At the time of writing this manuscript, the source code itself for G-Wing is currently undergoing public release. A GitHub repository [26] has been set up in anticipation of public release. The code is expected to be published before this manuscript is available online.

References

[1] S. W. Miller, M. A. Yukish, M. E. Hoskins, L. A. Bennett, and E. J. Little, "A Retrospective Analysis of System Engineering Data Collection Metrics for a 3D Printed UAS Design," in *Procedia Computer Science*, vol. 153. Elsevier B.V., 2019, pp. 1–8.

[2] S. W. Miller, M. E. Hoskins, L. A. Bennett, E. J. Little, and M. A. Yukish, "CHALLENGES AND LESSONS LEARNED FROM DEVELOPING AN OPERATIONALLY RESPONSIVE 3D PRINTED UAV," Penn State University, Tech. Rep., 2020. [Online]. Available: https://scholarsphere.psu.edu/resources/80adcde0-b62f-4cfb-8cd4-0a69e196e504

[3] J. D. Valenti and M. A. Yukish, "Minimizing Induced Drag of Rectangular, Additively-Manufactured Wings," in *AIAA AVIATION*, Virtual Event, 2020, pp. 1–12.

[4] ——, "Design of Curvilinear Spars for Wings using 1D Loading Distributions," in *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2022.

[5] S. R. Lamagna Reiter, J. D. Valenti, and M. A. Yukish, "Nonplanar Technique for 3D Printing Wings," in *AIAA Science and Technology Forum and Exposition, AIAA SciTech Forum 2022*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2022.

[6] T. Jones, M. A. Yukish, and S. W. Miller, "Design and Application of Additively Manufactured Compliant Mechanism Ailerons for sUAS," in *AIAA SCITECH 2022 Forum*. Reston, Virginia: American Institute of Aeronautics and Astronautics, 1 2022. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2022-1502

[7] J. D. Valenti, J. Bartolai, J. A. Cole, and M. A. Yukish, "Additive Manufacturing Process-Induced Wing Deformation and Effects on Aerodynamic Performance," in *ASME International Mechanical Engineering Congress & Exposition*, Columbus, 2022.

[8] J. D. Valenti, J. Bartolai, and M. A. Yukish, "Experimental Study of Wing Structure Geometry to Mitigate Process-Induced Deformation," in *Solid Freeform Fabrication Symposium*, Austin, 2022.

[9] Eclipson, "Free RC airplane," 2019. [Online]. Available: https://www.thingiverse.com/thing:3302937

[10] "3D Lab Print." [Online]. Available: https://3dlabprint.com/

[11] J. D. Valenti, J. Bartolai, and M. A. Yukish, "Design Space Bounding of Desktop 3D Printed Wings with Curvilinear Spars," in *AIAA AVIATION 2023 Forum*. Reston, Virginia: American Institute of Aeronautics and Astronautics, 6 2023. [Online]. Available: https://arc.aiaa.org/doi/10.2514/6.2023-4191

[12] L. Prandtl, "Tragflügel-Theorie, 1. u. 2," *Mitteilung. Nachr. von der Kgl. Gesellschaft der Wissenschaften. Math. Phys. Klasse*, vol. 151, 1918.

[13] ——, *Applications of Modern Hydrodynamics to Aeronautics, NACA-TR-116*, 1923. [Online]. Available: https://ntrs.nasa.gov/citations/19930091180

[14] M. A. Yukish and J. D. Valenti, "Simultaneously Deriving Wing Twist and Planform from Two Lift Distributions," *Journal of Aircraft*, pp. 1–5, 3 2020. [Online]. Available: https://arc.aiaa.org/doi/10.2514/1.C035586

[15] J. Valenti and M. Yukish, "Deriving Wing Chord and Twist from Lift and Lift Coefficient Distributions," in *AIAA Scitech Forum*. San Diego: American Institute of Aeronautics and Astronautics, 2022.

[16] M. A. Yukish, "Revisiting the combinatorics of lifting line and 2D vortex lattice theory," *The Aeronautical Journal*, vol. 123, no. 1265, pp. 993–1012, 7 2019. [Online]. Available: https://www.cambridge.org/core/product/identifier/S0001924019000678/type/journal_article

[17] M. Drela and H. Youngren, "Athena Vortex Lattice." [Online]. Available: http://web.mit.edu/drela/Public/web/avl/

[18] D. Locatelli, S. B. Mulani, and R. K. Kapania, "Wing-box weight optimization using curvilinear spars and ribs (SpaRibs)," *Journal of Aircraft*, vol. 48, no. 5, pp. 1671–1684, 9 2011.

[19] N. Umetani and R. Schmidt, "Cross-sectional structural analysis for 3D printing optimization," in *SIGGRAPH Asia 2013 Technical Briefs on - SA '13*, 2013.

[20] "OpenSCAD, The Programmers Solid 3D CAD Modeller." [Online]. Available: https://openscad.org/

[21] M. Drela, "AG24 Bubble Dancer DLG." [Online]. Available: http://airfoiltools.com/airfoil/details?airfoil=ag24-il

[22] G. Hodgson, "Slic3r Manual," 2019. [Online]. Available: https://manual.slic3r.org/

[23] "Printrun." [Online]. Available: http://www.pronterface.com/

[24] W. Roper, "There is no spoon: The new digital acquisition reality," *Defense AR Journal*, vol. 28, no. 4, p. 488, 2021.

[25] ——, "Bending the Spoon: Guidebook for Digital Engineering and e-Series," *Defense AR Journal*, vol. 28, no. 4, pp. 487–488, 2021.

[26] J. D. Valenti, "G-Wing." [Online]. Available: https://github.com/JDValenti/G-Wing