Fireflighter UAV

Professor: Damien Toohey
TA: Alan Marquez

MAE-154A: Preliminary Design of Aircraft

Fall 2019

Mechanical & Aerospace Engineering Department
University of California, Los Angeles

[March 23, 2019]

Table of Contents

Mission and Requirements

Mission Statement

Due to smoky and hazardous flying conditions, wildfires are very difficult to monitor and map. This is problematic for firefighters who need to be constantly updated on the fire's position to ensure public safety as well as their own safety, as they risk being surrounded unexpectedly by the fire. We are seeking a drone that utilizes a thermal infrared imagery and is capable of tracking the movement of wildfires through plumes of smoke. Thermal imaging capabilities are essential for locating the "hot spots" of a wildfire, so that firefighters on the ground and tanker planes know which regions need to be targeted first. By coordinating efforts between drones, ground crews, and tanker planes, fires can be extinguished faster than ever before.

Our mission is to create a UAV capable of locating the regions with the greatest thermal activity and relaying this information to firefighters on the ground. In order to achieve this objective, our UAV must be able to take off and land in areas where there is no landing strip available. Our solution is to use a slingshot to propel our UAV in the air so that it can be launched from virtually anywhere. Once in the air, the Fireflighter drone will map the region affected by the wildfire for an immense amount of time. The Fireflighter will maintain a high altitude in order to more efficiently map the area of concern and avoid encountering the plumes coming off of the wildfire. Once the Fireflighter has completed its mission and begins to return home, it will fly into an apparatus that will catch the UAV by the tip of its wing. This feat will require extra reinforcements on the wing, as the wings must be able to bear more extreme loads.

Mission Requirements

The minimum requirements for our UAV are as follows:

- UAV needs an endurance of at least 8 hours
- Must be equipped with live video transmission
- Must be equipped with a thermal infrared camera
- Must be able to fly in strong winds with low visibility and turbulent air
- Should be capable of flying at an altitude of 15,000 feet in mountainous regions
- UAV needs high lift and stability in order to account for hot and windy air surrounding the flames
- Wings must be able to withstand extreme loads as the UAV will be fired from a slingshot and caught by the tip of its wing upon landing

Payload and Avionics Requirements

If possible, we would like to minimize the weight of our payload while still fulfilling these requirements:

- UAV must have a live video transmitter and battery to power it
- UAV must have autopilot compatible avionics capable of autonomous steering
- UAV requires actuators in conjunction with control surfaces to perform flight maneuvers
- Must have a fuel tank and fuel delivery system capable of keeping the UAV in the air for the entirety of the mission

Layout

Design Overview

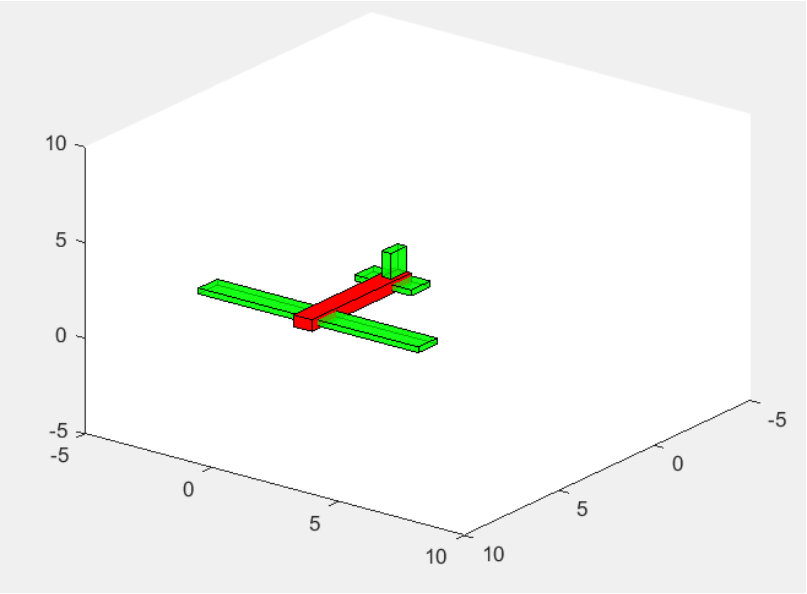
Since this mission requires a UAV specialized in endurance, we decided that a front-mounted engine was necessary for optimal weight distribution and stability. These factors are extremely important to our mission due to the strong winds that often accompany wildfires. Placing the propeller engine on the nose of the aircraft allows the center of gravity to be positioned aft of the neutral point, granting our aircraft the static stability it needs to deal with sudden shifts in wind conditions. Our UAV design is similar to other aircrafts intended to operate in high-wind situations.

Regarding our video feed setup, our UAV utilizes a payload consisting of a MWIR camera module and a wireless range UAV video transmitter. Since our mission is to map an area of concern from a relatively high altitude, the camera will be placed on the bottom of the aircraft such that it faces directly downwards. The camera module will be placed relatively close to the center of the UAV. This enables the sensors and cameras to operate with far less noise than if they were located near the propeller.

Additionally, our UAV must be able to take off and land in areas where there is no landing strip available. Thus, our UAV has no need for landing gear. Instead, our UAV will be flung from a slingshot and then retrieved by an apparatus that will catch the UAV by the tip of its wing. Since this form of retrieval places additional loads on the Fireflighter UAV, we had to take this into account when designing the wings of our UAV and in estimating the weight of our aircraft. Consequently, the wings will be heavier due to extra reinforcements. In our calculations, we used a load factor of 13, a value that is typical for other UAVs that utilize this form of landing.

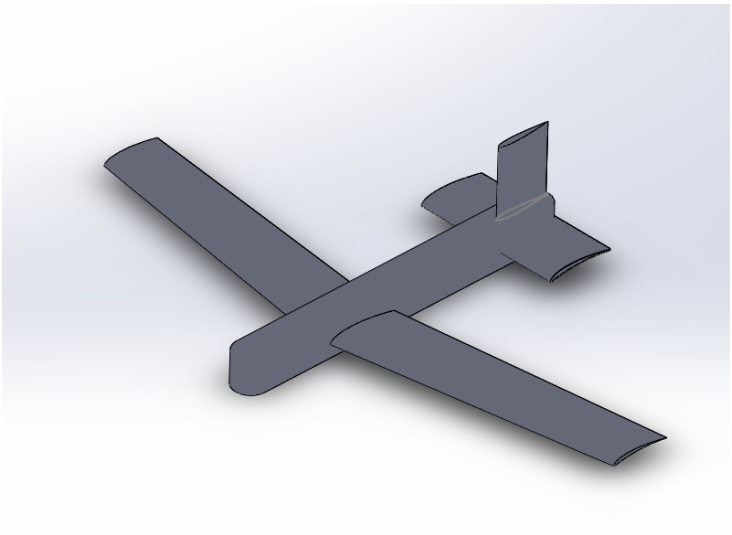
Modeling

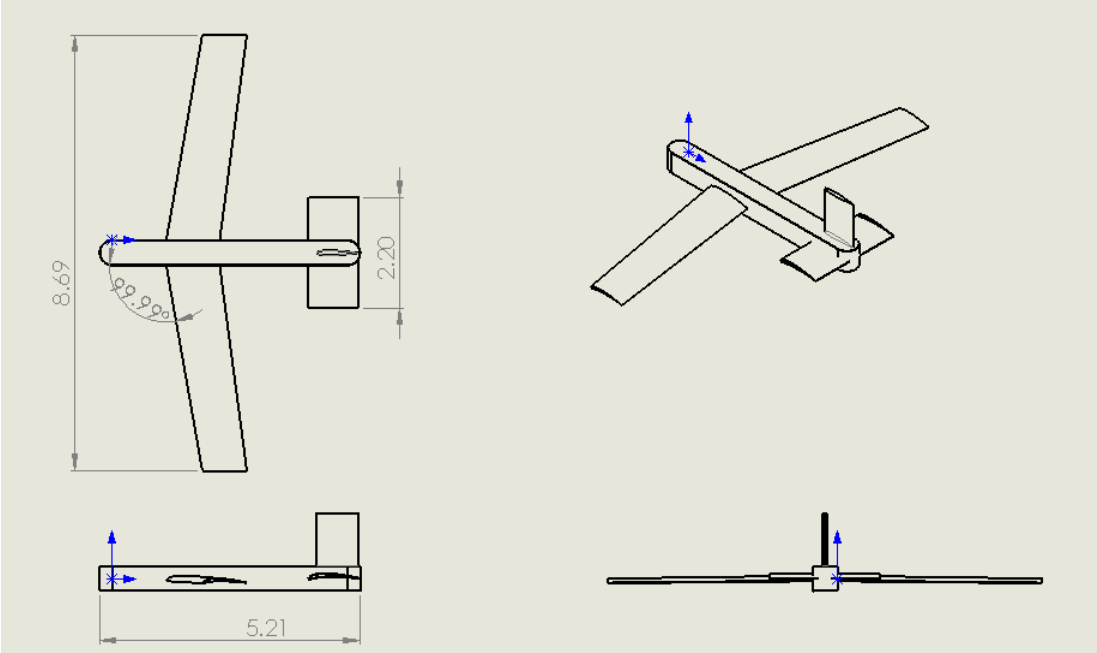
In the time since the final presentation, we have modified our MATLAB optimization code so that it produces a preliminary 3D model of the best aircraft in order to better visualize the dimensions of our plane before making a comprehensive CAD model. Although the code only generates boxes, it is enough to pass the “eyeball check” and helps with debugging. This image shows the 3D output of our modeling function. It takes the basic geometries of the best aircraft and runs them through a modeling function to create the following output:



Note: It does not model wing sweep or taper ratio and assumes rectangular airfoils for the wings and tails.

Once we decided on the final specs for our aircraft and decided that the 3D MATLAB model looked reasonable, we made a CAD model. Shown below is the final CAD model in which we modeled the actual airfoils that were used, along with the appropriate lengths for the tip and root chords. Additionally, all of the edges were filleted in order to reduce parasitic drag.





Note: The dimensions shown are in feet.

It is worth noting that this model is different from the one presented in class. This was because we continued to modify our optimization code. This model depicts the final correct dimensions generated.

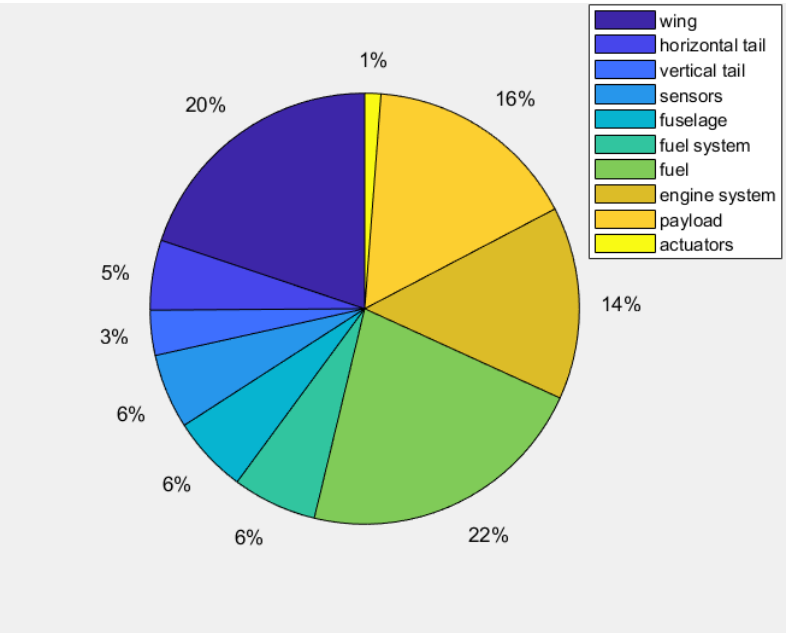
We are aware that sweep should not have a noticeable impact when travelling at subsonic speeds. The sweep angle was randomized from 0 to 30 meaning that it's possible that no sweep was required and the rest of the aircraft dimensions are the optimized parameters. However, it is also possible that the sweep helps with our aircraft's CG location and stability. Also, we think it would look more aesthetically pleasing to the customer.

Aircraft Dimensions

The main dimensions of the final UAV design are tabulated below for ease of access. It should be noted that the sweep angles included in the table correlate to the quarter chord position.

Wings		**Horizontal Tail**	
Area (ft^2)	8.5699	*Area (ft^2)*	2.2538
Span (ft)	8.6905	*Span (ft)*	2.2347
Root Chord (ft)	1.0775	*Root Chord (ft)*	1.0086
Tip Chord (ft)	0.8947	*Tip Chord (ft)*	1.0086
Sweep (deg)	9.9055	*Sweep (deg)*	0.0000
Taper ratio	0.8303	*Taper ratio*	1.0000
Aspect Ratio	8.8128	*Tail arm ($C/4_W$ to $C/4_t$ in ft_t)*	2.9227
Fuselage		**Vertical Tail**	
Length (ft)	5.2097	*Area (ft^2)*	1.1025
Max. Width (ft)	0.7295	*Span (ft)*	1.3389
Max. Depth (ft)	0.6084	*Root Chord (ft)*	0.8235
		Tip Chord (ft)	0.8235
		Sweep (deg)	0.0000
		Taper ratio	1.0000
		Tail arm ($C/4_W$ to $C/4_t$ in ft_t)	2.9227

Weight Breakdown



The pie chart shown above depicts the weight breakdown of our plane by component. The wings make up around 20% of the overall weight due to the additional structural reinforcements put in place to account for the SkyHook Recovery System. The additional factor of safety was reflected in our calculations by increasing the Niccolai safety factor to 13, rather than using 5. Additionally, you can see that the fuel makes up nearly a quarter of our weight as well. This is because our UAV is designed for endurance and long duration surveillance. The fuel weight corresponds to roughly 1 gallon of fuel with the tank placed almost directly above the wings inside the fuselage. It is important to note that the fuel tank must be centered on the wings as much as possible to prevent the center of gravity from shifting too much over the course of the journey.

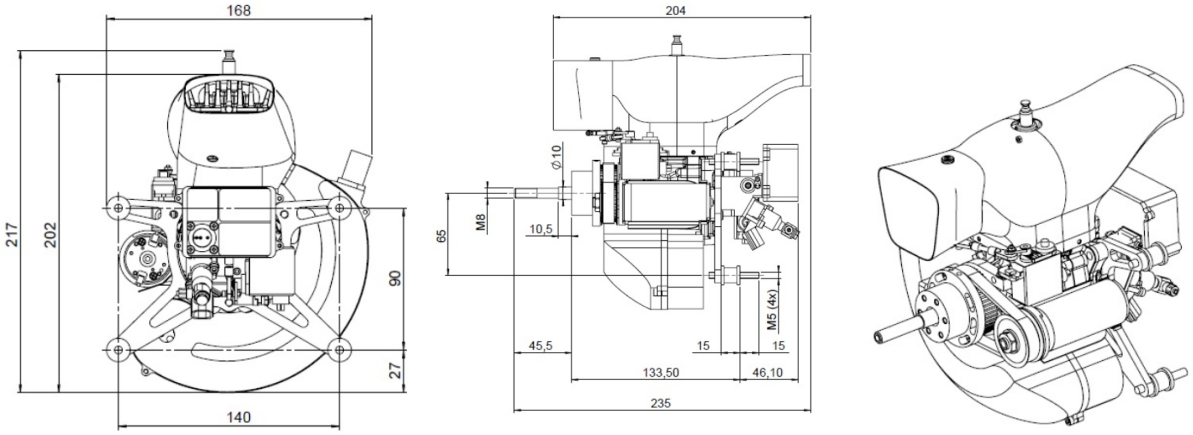
When using the niccolai equations to estimate the weight of the surface controls, the group found that the approximations initially overestimated the weight of the actuators needed to move the control surfaces. This prompted us to change the code so that the surface control weight was based on actual actuators found online. One big change that we made since the final presentation was the choice of actuators. We originally found heavy duty actuators that were around 1 pound each. However, after further research, we discovered that these were more powerful than necessary for our purposes and instead went with a different actuator specifically designed for lightweight UAVs and weighing in at just 40 grams. This significantly reduced the weight and increased the aircraft's endurance.

The weights of the key components are tabulated below. Note that the engine system weight includes the engine's bare weight along with the weight of other components, such as fuel sensors, that are required for the engine to operate.

Component	Weight (lbs)
Wing	7.227
Horizontal Tail	1.912
Vertical Tail	1.2314
Avionics	2.0437
Fuselage	2.0985
Fuel System	2.3137
Fuel	8
Engine system	5.2668
Payload	5.869
Surface Controls	0.4409

Engine Selection

For our engine, we decided upon the UAV28-EFI Engine system [9] because of its high fuel efficiency, compact design, and ability to service up to 55 lbs of takeoff weight. Some of the engine specifications are tabulated below.



Maximum Power Output	3.4 horsepower
Fuel Consumption (Cruise)	400g/kWh
Prop Speed Range	1600-8500 rpm
Weight	4.85 lb
Dimensions	235 x 217 x 168 mm
Fuel Pump Life	2000+ hours
Fuel Pump and ECU Weight	0.52 lb

Fuselage

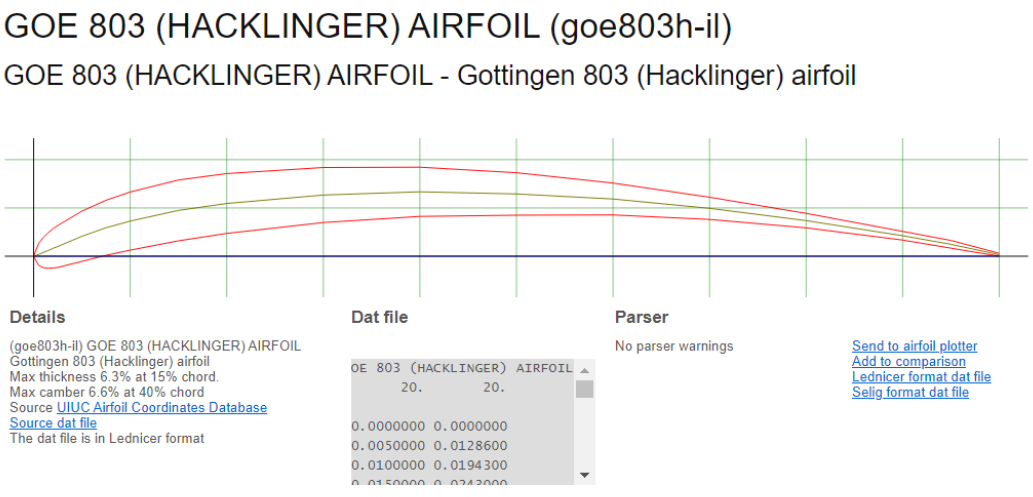
The smallest possible size of the fuselage was set to match the dimensions of the engine in order to adequately house all of the flight components. In our optimization code, we randomized the dimensions (with the minimum being the engine dimensions, and the max being 1 ft greater than the minimum) in order to find the most efficient combination. Overall, we found that narrower fuselages were better in reducing drag. The fuselage was assumed to take a basic and symmetric tube shape with rounded out ends, this was done to simplify the center of gravity calculation for the fuselage by assuming an even mass distribution.

Wing

In order to account for our unique landing style, we adjusted the niccolai equations to have a safety factor of 13, rather than 5. This additional margin of safety accounts for the increased stress placed on the wings when being caught by the Skyhook Recovery System (SRS).

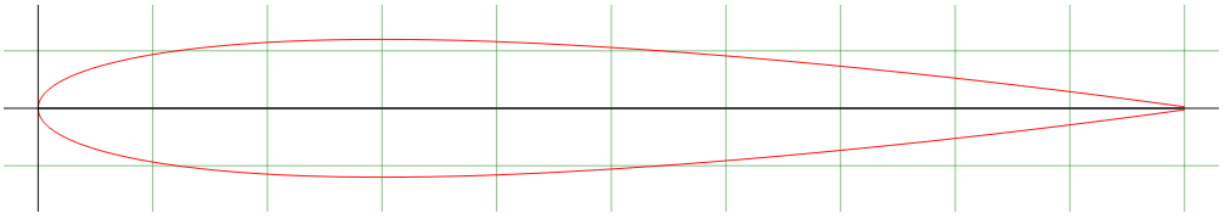
We also decided to use HiTechnology's HLS12 actuators [1] with dimensions of 6 x 0.6 x 0.7 in because they are the perfect size to fit our wing and they also produce an adequate 200N of linear force for our flaps.

We used the GOE-803 airfoil [2] for our wings because it has a high camber profile, ideal for our endurance based mission. The chord length and wingspan were determined by our optimization code. This airfoil was traced in solidworks in order to properly CAD the shape.



Tail

For our vertical tail we used the NACA-0012 airfoil [3] because of it's neutral profile and frequent use in vertical tail wings. Again, the chord length and wingspan were determined by our optimization code.



Details

(n0012-il) NACA 0012 AIRFOILS
NACA 0012 airfoil
Max thickness 12% at 30% chord.
Max camber 0% at 0% chord
Source UIUC Airfoil Coordinates Database
Source dat file
The dat file is in Lednicer format

Dat file

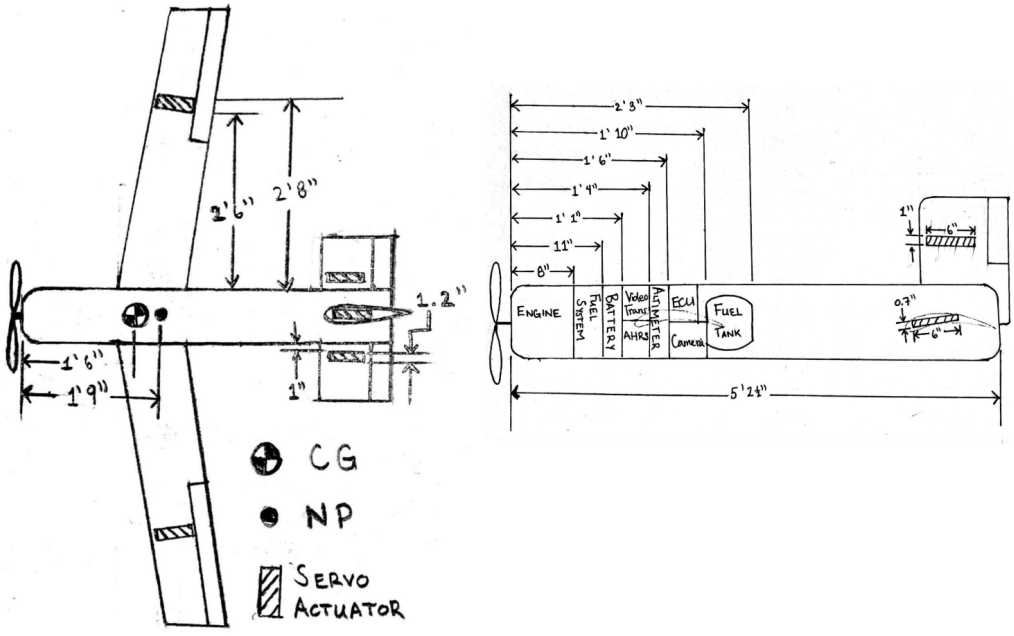
```
NACA 0012 AIRFOILS
     66.       66.

0.0000000 0.0000000
0.0005839 0.0042603
0.0023342 0.0084289
0.0052468 0.0125011
```

Parser

No parser warnings

Send to airfoil plotter
Add to comparison
Lednicer format dat file
Selig format dat file

Payload and Avionics Layout

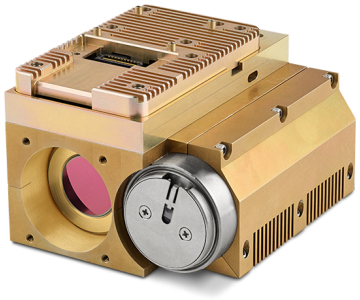


The internal components of our UAV were configured in such a way as to maximize the limited space provided by the fuselage. Components were placed forward as much as possible to shift

the center of gravity to a position that would increase the static stability of the aircraft. The final configuration is shown in the above figure.

Payload and Avionics Specs

The most important payload module is our infrared camera. For this, we are using the Neutrino QX Performance Series Camera, a MWIR Camera Module with a weight of 4.343 lb. This module has the proper specs for our mission and is designed for use in Recon and Surveillance. It has a large operating range of temperatures (-54-80 degrees Celsius) which is perfect for operating in the mountains and close to fires. It has dimensions of 12 x 7.1 x 11.6 cm and fits well within our fuselage.



As for the Live Video Transmitter, we are using the 30km Wireless Range UAV Video Transmitter since it is specifically designed for drones and UAVs with weight conscious components. The Device is long range and low power and can transmit a bandwidth of 8 MHz for high quality live feed. The dimensions of this device are 72x73x24mm which means it will also fit well within our fuselage.

To power the internal components, along with our actuators, we chose the Hercules 12V Lithium Ion battery with a weight of 1.2 lb. Lithium Ion batteries are known for being the most efficient and weight conscious batteries on the market and are capable of holding a high amount of Amp Hours, making them perfect for powering onboard electronic components.

Note that the sources listing the specifications of the payload are attached in sources 10-12 respectively.

In terms of avionics, the UAV was fitted with an AHRS, radar altimeter, an auto control system, and a fuel sensor to ensure that the aircraft would be able to have the data necessary to complete a mission autonomously. Sources 13-16 can be accessed to look at the individual specifications of each component.

Performance and Specifications

The performance and specifications of the final UAV design are discussed in this section.

Performance

The main performance goal for our mission requirements was to achieve 8 hours of flight time while cruising at altitudes of up to 15,000 feet. The following performance metrics are based on a cruising altitude of 15,000 feet.

Vmax - 247 ft/sec
Cruise Speed - 73 ft/sec
Stall Speed - 56.17 ft/sec
Range - 1767.9 miles
Endurance - 29 hours

As can be seen, the endurance goal was surpassed as a result of optimizing for maximum endurance (this is further discussed later on). The UAVs range should be enough to cover a sweep across a wildfire as well. It is important to note that the optimization process involved ranking aircrafts based on their endurance and range while giving other metrics, such as top speed, less of a priority.

Specifications

The UAVs maximum takeoff weight was first estimated to be around 60 pounds. The initial weight guess was based on weights of other UAVs that perform similar tasks. Results from the optimization process show that the initial weight guess was very much an overestimate.

Aircraft Max Takeoff Weight - 36.4031 lbs
Fuel Weight - 8 lbs
Wing Area - 8.5699 ft^2

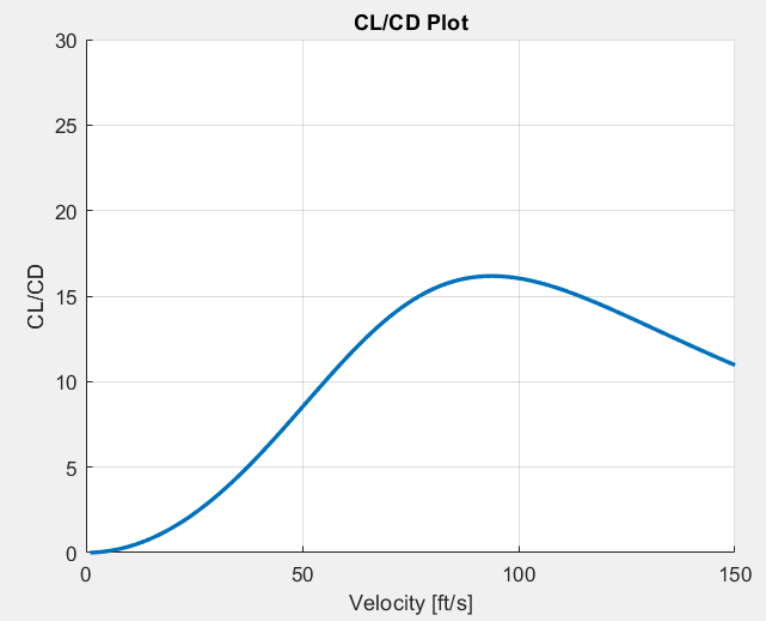
It is important to note that eight pounds of fuel corresponds to roughly one gallon of fuel. Although this amount turned out to be more than enough to meet the endurance requirement of 8 hours, the team decided to carry extra fuel to prolong the amount of time the UAV could remain airborne.

Lift

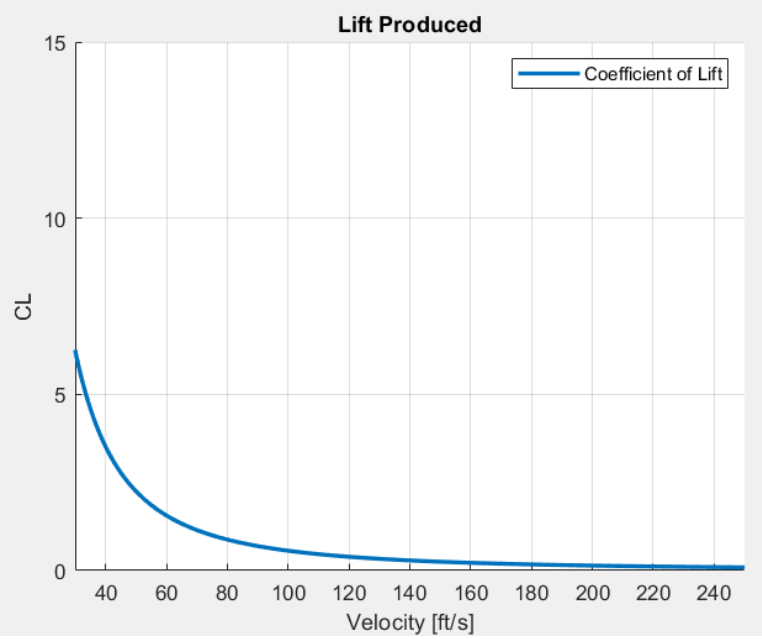
The velocity dependent coefficient of lift was calculated using the following equation assuming steady flight conditions and constant weight.

$$C_L = \frac{W}{\frac{1}{2}\rho V^2 S}$$

Lift coefficient values calculated for different speeds were then divided by predetermined drag coefficient values to obtain lift to drag ratios for a range of aircraft speeds.



The above curve shows the lift to drag ratio as a function of velocity. The maximum lift-to-drag ratio obtained was 16.74 which falls closely to the lift-to-drag ratio seen in a variety of subsonic aircraft. Though the max cl/cd occurs at 95 ft/s, the cruising speed of our aircraft had to be reduced to 73 ft/s when taking into account high altitudes and to match the power required to maintain cruising altitude during our mission. This resulted in a lift-to-drag ratio of 14.54 at cruising conditions.



The above plot shows the relationship between lift generation and aircraft velocity. As expected, the amount of lift the wings can produce drops off at higher speeds. We wanted to make sure that the cruising speed of our aircraft was optimal to maximize the lift generation of the wings while minimizing the resulting induced drag.

Drag Estimation

Without the availability of wind tunnels or a physical UAV prototype, the project had to estimate the drag of the aircraft using empirical equations that provided drag estimates of each aircraft component separately. The component build-up method states that the total parasitic drag of the aircraft could be estimated using the form factor (FF), interference factor(Q), skin friction coefficient (C_f), and Wetted area (S_{Wet}) of each component [5]:

$$C_{D0} = \sum_{c=1}^{n} C_{f,c} \cdot FF_c \cdot Q_c \cdot \frac{S_{wet,c}}{S_{ref}} + C_{D,misc} + C_{D,L+P} \quad .$$

The parasitic drag coefficient was nondimensionalized using the area of the wings as a reference area (S_{ref}). The skin friction coefficient was assumed to be 0.0043 for all components. The wetted area of the fuselage was estimated using the following equation, where d_F represents the fuselage's diameter, l_F is the fuselage's longitudinal length, and λ_F is the fuselage fineness ratio (l_F/d_F).

$$S_{wet,F} = \pi \cdot d_F \cdot l_F \cdot \left(1 - \frac{2}{\lambda_F}\right)^{2/3} \left(1 + \frac{1}{\lambda_F{}^2}\right)$$

The wetted area of the main wing was calculated using the following equation, where $(t/c)_r$ is the thickness to chord ratio at the root of the wing (this was deemed to be 0.063 for the GOE 803 airfoil used in the main wing). S_{exp} is the exposed wing area which was determined by finding the wing area being eclipsed by the fuselage and subtracting it from the standalone wing area. The ratio of relative airfoil thickness, τ, is found by dividing the thickness-to-chord ratio at the wing tip by that found at the wing root (this was deemed to have a value of one since the same airfoil was used throughout the spanwise direction of the wing). Note that λ is the taper ratio of the wing (wing tip chord / wing root chord). This equation was applied similarly to the horizontal and vertical tail of the UAV due to their similar design.

$$S_{wet,W} = 2 \cdot S_{exp} \cdot \left(1 + 0.25 \cdot (t/c)_r \cdot \frac{1 + \tau \cdot \lambda}{1 + \lambda}\right)$$

The form factor of the wing was found using the following empirical equation (again, this was used to calculate the values for the horizontal and vertical tail as well).

$$FF = \left[1 + \frac{0.6}{x_t}\left(\frac{t}{c}\right) + 100\left(\frac{t}{c}\right)^4\right] \cdot \left[1.34 \cdot M^{0.18} \cdot \left(\cos\varphi_m\right)^{0.28}\right]$$

Here, x_t is the location of maximum thickness along the airfoil , M is the mach number of the flow, and φ_m is the sweep angle at the %-line of maximum thickness. Similarly, the form factor of the fuselage was found using the following empirical equation:

$$FF_F = 1 + \frac{60}{\left(l_F / d_F\right)^3} + \frac{\left(l_F / d_F\right)}{400}$$

The interference factor of the main wing and fuselage were approximated to be equal to one, while the interference factor of the horizontal and vertical tail were approximated to be equal to 1.05 and 1.03 respectively.

Once the parasitic drag was estimated and added up for each component, the leakage and protuberance drag of the aircraft ($C_{D,\,L+P}$) was estimated to be 10% of the parasitic drag using values of what was seen for other propeller aircraft. All other forms of zero-lift drag were deemed negligible to the drag calculations.

In order to get the total coefficient of drag of the UAV, the induced drag of the aircraft would also need to be estimated. The induced drag coefficient was deemed to be proportional to the coefficient of lift.

$$C_{Di} = KC_L{}^2$$

Where k is the form factor given by the following equation:

$$k = \frac{1}{\pi Ae}$$

Here, A is the aspect ratio of the wing and e is the oswald's efficiency factor. The oswald efficiency factor was estimated using two different equations depending on the sweep angle of the wing. For wings with sweep angles less than thirty degrees the following approximation can be used [6]:

$$e = e_{theo} \cdot k_{e,F} \cdot k_{e,D_0} \cdot k_{e,M}$$

Here, the efficiency factor of the aircraft is based on the theoretical efficiency factor multiplied by correction factors due to the fuselage ($k_{e,F}$), Zero lift drag ($k_{e,D0}$), and mach number ($k_{e,M}$). The theoretical efficiency factor was calculated using the following:

$$e_{theo} = \frac{1}{1 + f(\lambda - \Delta\lambda) \cdot A}$$

The variable f(λ - Δλ) is a function of taper ratio (λ) given by the polynomial below, the equation is meant to use an input taper ratio of the given wing and calculate the theoretical efficiency factor of the optimum taper ratio to achieve elliptical loading.

$$f(\lambda) = 0.0524\,\lambda^4 - 0.15\lambda^3 + 0.1659\lambda^2 - 0.0706\lambda + 0.0119$$

In order to shift the input taper ratio to the optimal taper ratio the following shift has to be applied:

$$\Delta\lambda = -0.357 + 0.45 \cdot e^{0.0375\varphi_{25}}$$

The correction factor due to the fuselage can be found using the following equation:

$$k_{e,F} = 1 - 2\left(\frac{d_F}{b}\right)^2$$

The correction factor due to mach number was assumed to be equal to one due to the fact that the UAV would be flying at low speeds. The correction factor due to zero lift drag was given a value of 0.804 which was a value based on other propeller aircraft.

If the sweep angle of the wing was equal to or greater than thirty degrees, the following empirical equation would be used to approximate the oswald efficiency factor:

$$\text{Swept-Wing Aircraft: } e = 4.61(1 - 0.045A^{0.68})(\cos\Lambda_{LE})^{0.15} - 3.1$$

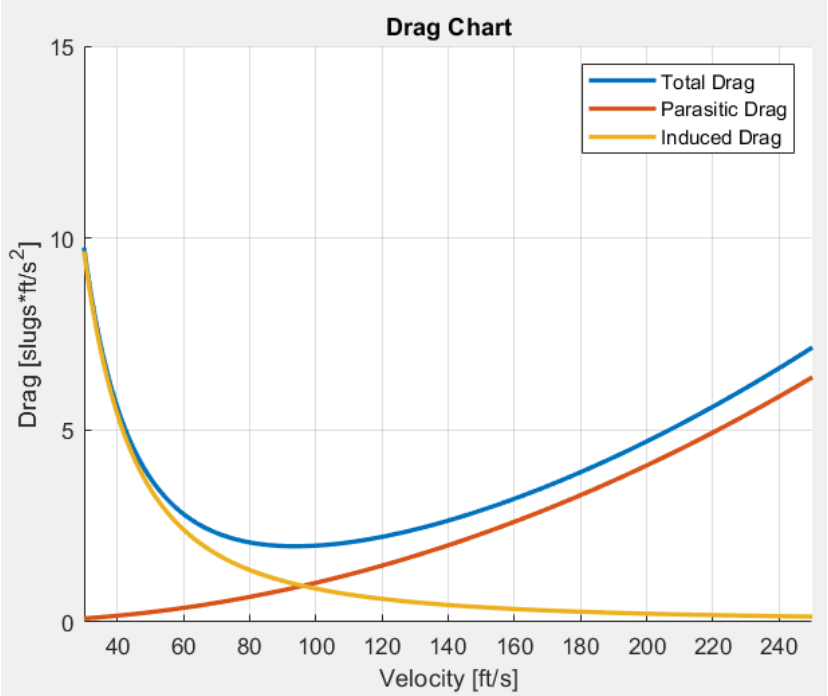
$$(\Lambda_{LE} > 30 \text{ deg})$$

Using this process we were able to find reasonable values for the oswald efficiency factor. Once the induced drag coefficient was calculated, the drag polar equation could then be used to find the total drag coefficient for the UAV.

$$C_D = C_{D_0} + C_{D_i}$$

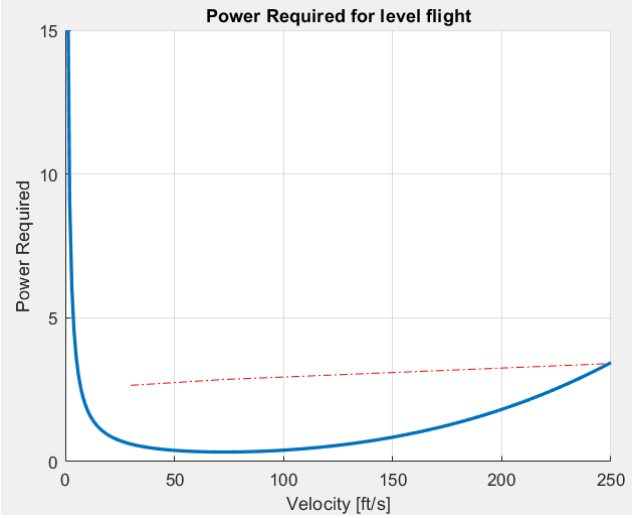
The different drag values of the final UAV configuration were plotted for different velocities by multiplying the respective drag coefficients by dynamic pressure and the reference wing area.

$$D = \tfrac{1}{2}\rho v^2 S C_D$$



Above is the results of our drag equations. Drag is a minimum at our cruising speed.

Propulsion



Above is the power required chart for level flight. The red line is power available. Our engine has a maximum power output of 3.4 HorsePower which corresponds to a top speed of 247 ft/s. For our purposes, this is more than necessary since the UAV is meant to operate at cruising

conditions. However, a high top speed may be important in the case of an emergency where we may need to travel quickly to another area.

The power available line was plotted based on our propeller efficiencies at stall, cruise, and max speeds. The line is supposed to be more curved but we made this plot using only 3 points as this is enough to see where the power available line intersects with the power required line. We looked into generating a plot for power available but it proved to be difficult due to the use of charts for calculating propeller efficiency.

Propeller Design

In order to compute the power required for the cruising, climbing, top speed, and takeoff conditions of our aircraft, we had to first calculate the velocities for each condition. In our optimization code, V_{stall} is calculated using the assumption that $L = W$. Thus, the following expression can be used:

$$V_{stall}^{2} = \frac{W}{\frac{1}{2}\rho S C_{L_{max}}}$$

By determining where the power available curve for our engine intersects the power required curve for level flight, we were able to obtain our top speed, V_{max}. The power associated with V_{max} is the P_{av} specific to our engine.

The speed of the aircraft at takeoff was determined using the following relation:

$$V_{takeoff} = 1.2V_{min} = 1.2V_{stall}$$

Using the assumption that $L = W$, we derived the following expression:

$$C_L = \frac{W}{\frac{1}{2}\rho v^2 S}$$

Using this C_L value, and assuming that $T = D$, we were then able to determine the drag at takeoff using the expression,

$$T = D = \frac{1}{2}\rho v^2 S(C_{D_0} + kC_L^{2}) .$$

And, finally, we were able to determine the power required at takeoff using the following relation:

$$P_{req} = DV$$

The cruise speed of our aircraft, V_{cruise}, was determined within our optimization code, and using the above relation, we were able to determine the power required to achieve V_{cruise}.

In order to find the speed of our aircraft during its climb, we needed to obtain a value for C_L first. Using $C_D = C_{D_0} + kC_L{}^2$, we were able to derive the following expression for C_L:

$$C_L = \sqrt{3C_{D_0}\pi Ae} = \sqrt{\frac{3C_{D_0}}{k}}$$

Then, using the assumption $L = W$, we derived an expression for the lift of the aircraft during its climb:

$$L = C_L(\tfrac{1}{2}\rho v^2 S)$$

And, finally, assuming that $V_{climb} \approx V_{min\ power}$, we obtain the following expression for V_{climb}:

$$V_{climb}{}^2 = \frac{W}{\frac{1}{2}\rho S C_L}$$

The power required to achieve our speed for V_{climb} can then be determined by obtaining the drag during the aircraft's climb, and using $P_{req} = DV$.

The following table depicts the speed and power required for the cruising, climbing, top speed, and takeoff conditions of our aircraft:

Cruise		**Climb**	
V (ft/s)	73.00	*V (ft/s)*	74.30
Preq (HP)	0.2736	*Preq (HP)*	0.3250
Top Speed		**Takeoff**	
V (ft/s)	247.00	*V (ft/s)*	67.40
Preq (HP)	3.4000	*Preq (HP)*	0.3294

Next, we were able to compute C_S values for each condition using the expression

$$C_S = V\left(\frac{\rho}{Pn^2}\right)^{\frac{1}{2}}.$$

Using these C_S values and the design chart for a propeller with 2 blades included in NACA Report 640 (*See Reference 1*), we were able to determine a value for J and the blade angle for each of the conditions our aircraft will experience throughout its flight.

Using these J values, the blade angle values, and the efficiency curves for a propeller with 2 blades included in NACA Report 640 (*See Reference 2*), we were able to determine η as well as the optimal diameter for each condition.

The tip speeds were calculated using the following expression:

$$M_t = \frac{V}{a}\sqrt{1 + \left(\frac{\pi}{J}\right)^2}$$

The following table depicts the optimal diameters for each condition that our aircraft will experience during its flight:

Cruise		**Climb**	
C_S	1.0053	C_S	0.9885
Number of Blades	2	*Number of Blades*	2
Pitch Rate (Beta)	15	*Pitch Rate (Beta)*	15
Advance Ratio (J)	0.500	*Advance Ratio (J)*	0.490
Efficiency (Eta)	0.765	*Efficiency (Eta)*	0.760
Prop Diameter (D)	1.0306	*Prop Diameter (D)*	1.0703
Tip Speed	0.4127	*Tip Speed*	0.6085
Top Speed		**Takeoff**	
C_S	2.0550	C_S	0.8944
Number of Blades	2	*Number of Blades*	2
Pitch Rate (Beta)	27.5	*Pitch Rate (Beta)*	15
Advance Ratio (J)	1.140	*Advance Ratio (J)*	0.450
Efficiency (Eta)	0.8500	*Efficiency (Eta)*	0.730
Prop Diameter (D)	1.5294	*Prop Diameter (D)*	1.0573
Tip Speed	0.4768	*Tip Speed*	0.4224

For our aircraft, we selected a diameter of 1.05 ft, as this is the optimal diameter for cruising, and our aircraft is heavily tailored towards endurance.

Using this diameter, we were able to compute new values for J. Then, using the design chart for a propeller with 2 blades included in NACA Report 640 (*See Reference 1*), as well as the previously calculated C_S values, we were able to determine a new blade angle for each of the flight conditions. Thus, our UAV will utilize a variable pitch propeller to adjust the blade angle for different flight conditions.

The efficiency curves for a propeller with 2 blades included in NACA Report 640 (*See Reference 2*) were once again used to determine η. The tip speeds for each of the different flight conditions were recalculated using the new diameter as well.

The following equations were used to determine the thrust power coefficients:

$$C_P = \frac{P}{\rho n^3 D^5}$$

$$C_T = \frac{T}{\rho n^2 D^4}$$

	C_S	C_T	C_P	J	η	M_{tip}	$\beta\ (deg)$
Cruise	1.0053	0.0565	0.0337	0.4908	0.7600	0.4127	15
Top Speed	2.0550	0.2075	0.4187	1.6605	0.8500	0.4697	40
Climb	0.9885	0.6897	0.0400	0.4995	0.7650	0.4205	15
Takeoff	0.8944	0.7602	0.0406	0.4531	0.7350	0.4196	15

Stability

Static Stability

The static margin of the aircraft is dependent on the location of the aircraft's center of gravity and neutral point. The static margin of the UAV is important to provide stability to the aircraft. A

static margin range of 0.05 to 0.3 was chosen as that of a stable aircraft. The static margin was determined for the UAV at max takeoff weight (full fuel tank) and when the fuel tank was empty. These were the two conditions tested, as the aircraft would have a set payload.

The UAV was split into different components in order to approximate the aircraft's center of gravity location. The UAV was split into the wing, fuselage, horizontal tail, vertical tail, surface controls, propulsion system, fuel system, and fuel. The moment arms were calculated for each component based on the weight of the components and their respective centroids (an even mass distribution was assumed for each component). The moment arms of internal components such as avionics, payload, and auto control system were computed. The overall center of gravity of the aircraft could be then computed using a sum of the moment arms divided by the weight of the aircraft [4]:

$$X_{cg} = \frac{\sum_{i=1}^{c} M_c}{W}$$

This was again computed for the aircraft when it was full of fuel ($W = W_{TO}$) and when it was depleted of all fuel to ensure that the aircraft was still stable ($W = W_{TO} - W_f$). Note, W_{TO} is the maximum take-off weight of the aircraft and W_f is the weight of fuel. This location was nondimensionalized by dividing it by the chord of the wing.

$$h_{cg} = \frac{x_{cg}}{c}$$

The neutral point was then computed using the following equation:

$$h_n = \frac{h_{ac_w} + h_{ac_t} \frac{S_t}{S_w} \frac{a_t}{a_w} (1 - \varepsilon_\alpha)}{1 + \frac{S_t}{S_w} \frac{a_t}{a_w} (1 - \varepsilon_\alpha)}$$

Here, $h_{ac,w}$ is a constant value based on the airfoil of the wing. Similarly, $h_{ac,t}$ is a value based on the airfoil of the horizontal tail. The term ε_α is the downwash angle. The terms, S_t and S_w are the areas of the horizontal tail and wing respectively. The 3d lift curve slopes of the tail and wing (a_t and a_w) were estimated using the 2d lift curve slopes of the airfoil and applying an equation. The 2d lift curve slope was determined from the slope of the C_L vs α graph of the airfoils. The parameter was approximated for the 3d wing by applying the equation [7]:

$$a_t = \frac{a_t'}{(1 + k a_t')}$$

Here, k is the form factor of the wing or tail using the equation described earlier:

$$k = \frac{1}{\pi Ae}$$

Note, a_t' denotes the lift curve slope for the 2d case. The downwash angle was then approximated using the equation, where K is again the form factor as described above:

$$\epsilon_\alpha = 2ka_w$$

Once the nondimensionalized neutral point was determined, the static margin was calculated by using the equation:

$$SM = h_n - h_{cg}$$

The finalized UAV had the following static margin values:

Static Margin for Max Take-off Weight	0.2541
Static Margin with no Fuel	0.2237

The positive static margin indicates that the aircraft is statically stable in the longitudinal direction. The static margin of the UAV does not vary much between both extreme loading conditions, this ensures good stability throughout the aircraft's flight. The code for this process can be found in the cg.m file.

The moment coefficient about the aerodynamic center of the wing (C_{Mac}) was approximated using data points provided by the Airfoil Tools database for the GEO 803 airfoil and Equation 3.28 from McCormick.

$$C_{M_{ac}} = -\frac{\pi}{4}\left(A_1 - A_2\right)$$

The coefficients A_1 and A_2 were determined by numerically integrating Equation 3.25b from McCormick.

$$A_n = \frac{2}{\pi}\int_0^{\pi} \frac{dz}{dx}\cos(n\theta)\,\mathrm{d}\theta$$

Here, dz/dx depends on the points that lie on the mean camber line of the airfoil. Doing the above calculations yielded a moment coefficient of C_{Mac}= -0.205. Comparing this value to the pitching moment plots in the NACA 824 Report shows that our C_{Mac} value falls in line with the

provided data. The data point for the NACA 1408 with a simulated 60° simulated split flap[2] ($C_{M,c/4}$= -0.22) was used for reference since it is geometrically similar to the airfoil we used. Our C_{Mac} value was essential for calculating the incidence angle for trimmed flight.

Stability Derivatives

Stability derivative coefficients had to be calculated to use in our flight simulation code. Most of the linear stability coefficients were calculated using the following equations:

$$C_{L_0} = -a_t \frac{S_t}{S_w} i_t$$

$$C_{L_\alpha} = a_w + a_t \frac{S_t}{S_w}\left(1 - \varepsilon_\alpha\right)$$

$$C_{L_{\dot{\alpha}}} = -2\eta V_H a_t \frac{d\varepsilon}{d\alpha}$$

$$C_{L_q} = 2\eta V_H a_t$$

$$C_{L_{\delta e}} = \tau a_t \frac{S_t}{S_w}$$

$$C_{D_\alpha} = \frac{2 C_L}{\pi e A} C_{L_\alpha}$$

$$C_{D_{\delta e}} = \frac{2 C_L}{\pi e A} C_{L_{\delta e}}$$

$$C_{M_0} = C_{M_{ac}} + V_H a_t i$$

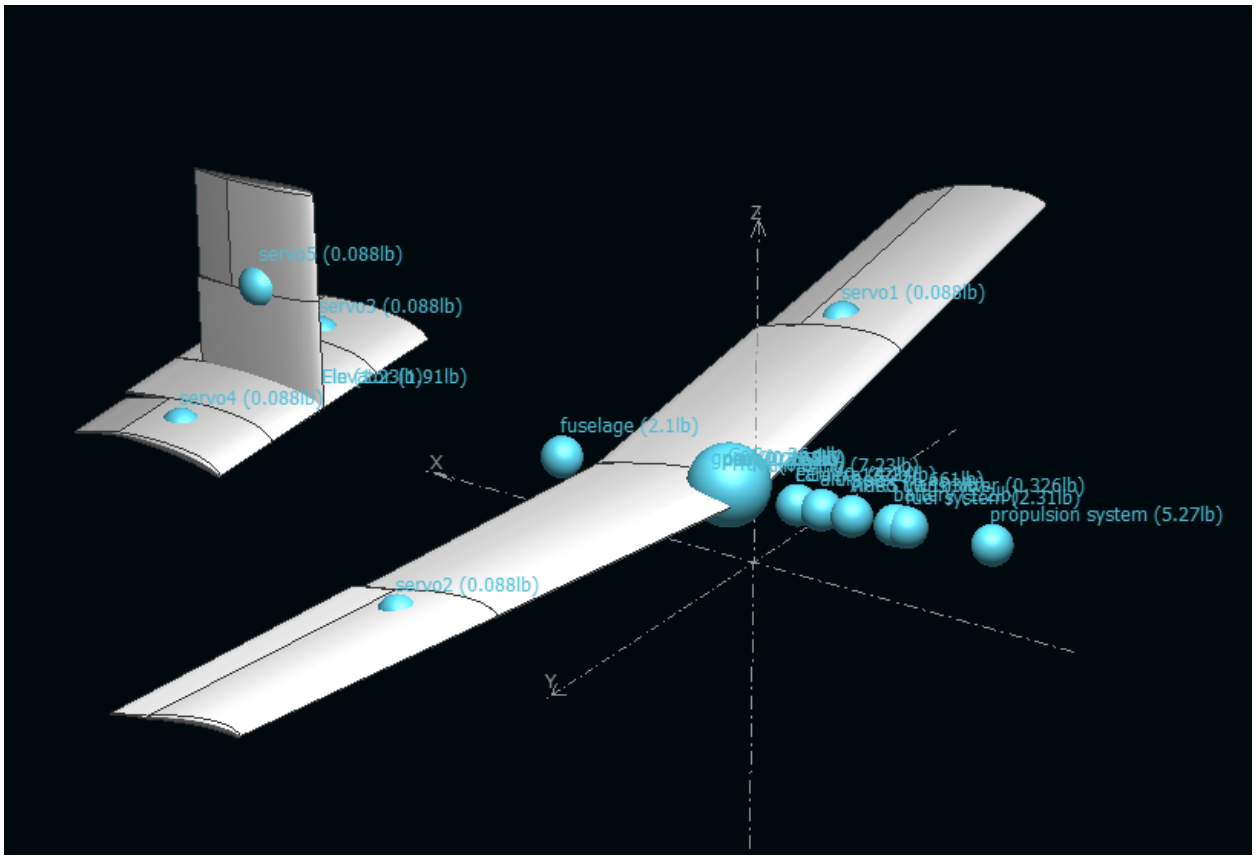
$$C_{M_{\dot{\alpha}}} = -2 C_{L_\alpha} \eta V_H \frac{l_t}{c} \frac{d\varepsilon}{d\alpha}$$

$$C_{M_{\delta e}} = -\tau a_t V_H$$

C_{Do} was taken from the parasite drag calculations while the remaining stability coefficients were calculated using the XFLR5 software.

The XFLR5 software uses airfoil data and aircraft dimensions to approximate values for the linear and lateral/directional stability derivatives. The aircraft's main structural components were modelled and all other components were simulated using their weight and center of gravity location. The XFLR5 model of the UAV is shown below, note that the fuselage is omitted for analysis per recommendation from people more familiar with the program's analysis methods.

The derivatives were computed by completing a stability analysis that iterates through different flap deflections. The flaps on the main wing began at 3° deflections (different direction on opposite ends of the wing). The elevators were set to have an initial deflection of -3° while the flap on the vertical tail began at a deflection of -5°. The stability analysis was performed at step sizes of 0.1° and stopped after each flap was moved by 1°. The plane analysis was performed at a fixed lift (the program varied speed to achieve constant lift). The log file stating the results of the program's analysis was attached in the *stabilityderivatives.txt* file.



Linear Stability	
$C_{L_0} = 0.375$	$C_{D_{\delta e}} = 0.0027$
$C_{L_\alpha} = 5.31879$	$C_{M_0} = 0.2987$
$C_{L_{\dot{\alpha}}} = 1.4251$	$C_{M_\alpha} = -3.54247$
$C_{L_q} = 13.94164$	$C_{M_{\dot{\alpha}}} = -10.0192$
$C_{L_{\delta e}} = 0.302$	$C_{M_q} = -20.1873$
$C_{D_0} = 0.0170$	$C_{M_{\delta e}} = -1.3836$
$C_{D_\alpha} = 0.0473$	

Lateral/Directional Stability	
$C_{Y_\beta} = -0.05416$	$C_{l_{\delta r}} = -0.00229$
$C_{Y_{\delta r}} = 0.191$	$C_{n_\beta} = 0.02910$
$C_{l_\beta} = -0.02239$	$C_{n_p} = -0.02973$
$C_{l_p} = -0.55042$	$C_{n_r} = -0.02128$
$C_{l_r} = 0.07009$	$C_{n_{\delta a}} = 0.02$
$C_{l_{\delta a}} = -0.161$	$C_{n_{\delta r}} = -0.0917$

The only modifications made to our stability derivatives when inputting them into the autopilot simulation were changing the signs of C_{L_adot} , C_{Yb} , C_{nb}, and C_{nr}. This was done so that our values could fall more in line with those used in the Pioneer flight simulation and to ensure that our aircraft would remain stable during flight. We were unable to use the XFLR5 software to compute the stability derivatives due to deflections in the aileron and rudder, as a result, the values given by the pioneer were used for simulation purposes. The values for the moments of inertia (I_{XX}, I_{YY}, I_{ZZ}, and I_{XZ}) of the UAV were also taken from outputs from the XFLR5 program. The inertia values estimated by the program are tabulated below:

Inertia	**Estimate [lbs*ft^2]**
I_{XX}	43.94
I_{YY}	63.44
I_{ZZ}	105.9
I_{XZ}	-3.553

Trimmed flight

The horizontal tail incidence angle required for trimmed flight was calculated for aircraft speeds ranging from V_{stall} to V_{max}. These calculations were made using the following equation:

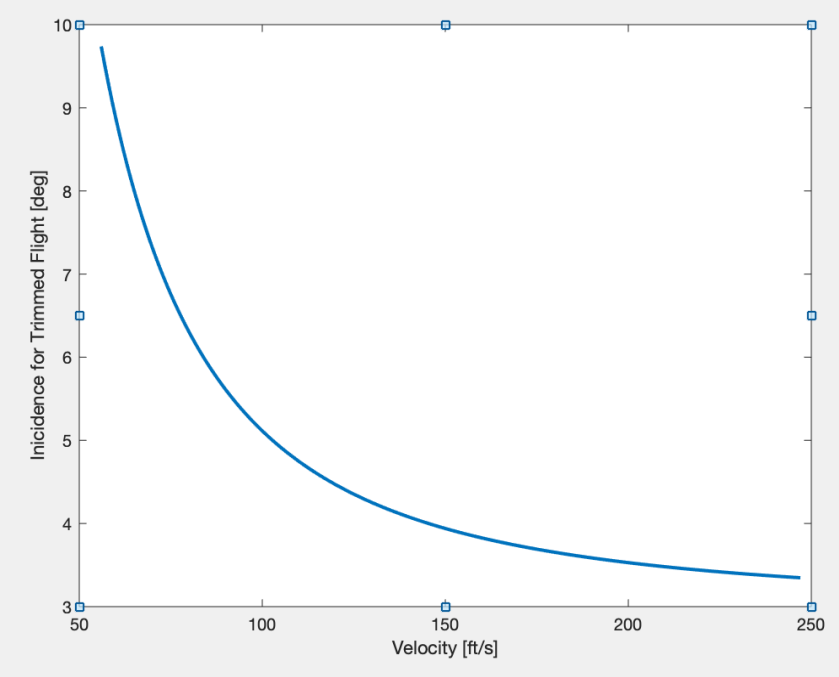
$$i_t = -\frac{C_{M_{ac}} C_{L_\alpha} + C_{M_\alpha} C_L}{C_{L_\alpha} C_{M_i} - C_{M_\alpha} C_{L_i}}$$

The parameters used for this equation are predetermined values in which the C_L term takes into account the variations in velocity. Following are some of the equations used to find the remaining coefficients needed to calculate the incidence angle.

$$C_{L_i} = -a_t \frac{S_t}{S_w}$$

$$C_{M_i} = a_t V_H$$

$$C_{M_\alpha} = C_{L_\alpha}\left(h_{cg} - h_n\right)$$



For our mission, this was the optimal incidence angle curve. For a fixed wing mission, we would pick an incidence angle of around 6.5° which corresponds to the cruising speed of our aircraft.

Optimization Process

Algorithm and Functions

Our Optimization Code consisted of 5 main functions:

1. *UAV.m* - This is the actual script which runs the (N=100,000) iterations and outputs the specs of the best aircraft. This code randomly generates most of the dimensions of our aircraft and then calls the other functions to calculate the specs and generate the graphs. This function essentially compiles the specs for endurance and range and then sorts out whether or not the aircraft is “valid” based on if it meets our specs for stability and endurance, along with a few other requirements. If the aircraft is valid, it stores the specs of the aircraft, along with it's geometries, in vectors. This is useful for plotting data later. Once the N iterations have been completed, the code identifies the best aircraft as the one with the longest endurance. *UAV.m* then contains the propeller calculations and prints the key performance info about the best identified aircraft.

2. *Niccolai.m* - This function takes in several geometries and outputs the Niccolai equation generated weights of various structures (wings, fuselage, tail, etc) along with a few other parameters like S that the Niccolai equations converge onto. Once the weights converge, they are all returned back to the main UAV script.
   Inputs: W,A,Sh,Sv,bh,bv,hac,c,ch,cv,hach,hacv,space
   Outputs: W,Ww,Wau,Weng,Wf,WF,Wfs,Wfu,Wht,Wvt,Wlg,Wp,Wpl,Wsc,Delta,tr,S,lf,lh,tc D,N

3. *CG.m* - This function takes in several weights and geometries and outputs the static margins along with a few CG locations. In this code we hard coded the weights of our payloads and avionics. The function uses these to then find the moments created by each structure in our aircraft. The placement of these avionics is specifically coded to not overlap. Refer to the internal drawing to see the layout of payload materials inside the fuselage.
   Inputs:W,Ww,Wau,Wf,Wfs,Wfu,Wht,Wvt,Wlg,Wp,Wpl,Wsc,c,ch,cv,space,Delta,tr,c_root,c_tip,lf,hac,hach,hacv,lh,b,Sh,S,e,A,bh
   Outputs: cg,cgwing,hn,hcg,sm,sm1,a_w,ep,a_t,hachtemp

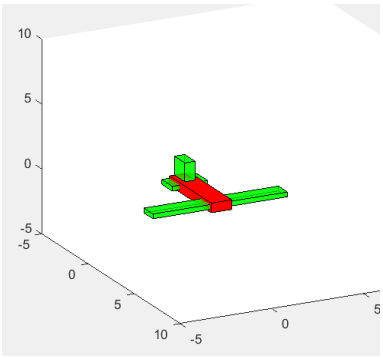
4. *Calcs.m* - This function takes in almost all of our generated aircraft’s parameters and calculates the drag components, the lift, and the power required at each velocity. These values are then used to calculate the performance specs of the input aircraft including CL max, CL Cruise, V stall, V cruise, Range, and Efficiency. The drag is computed for each individual component, including wings, fuselage, and tails.
   Inputs:WF,b,c_root,c_tip,S,tc,tr,c,D,lf,ch,cv,Sh,Sv,Delta,W,density,A,Wfinal,eff,fc,Pav
   Outputs: cd,clmax,Vstall,K,Vcruise,Vdmin,R,E,Max,e,delta_tr,vmax

5. *Plotcalcs.m* - Now that we have almost everything we could need for our aircraft, the only thing left is to generate our graphs. This function plots a variety of graphs and can be suppressed during debugging. We used this function for all of our report’s plots including CL/CD, Power Required, Drag, and histograms of Valid Aircraft Weights and Reasons for Failure. The graphs shown in our report come from this function.

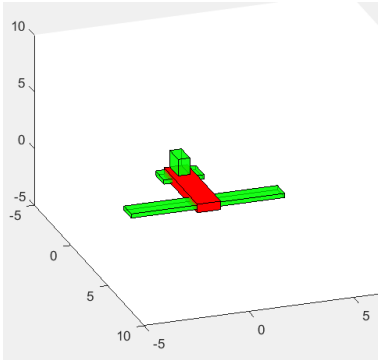
Inputs: WF,b,c_root,c_tip,S,tc,tr,c,D,lf,ch,cv,Sh,Sv,Delta,W,density,A,Wfinal,eff,fc,Pav
Outputs: No outputs used in UAV

6. *Modelaircraft.m* - This function creates a 3D model of the best aircraft identified by the rest of the optimization code. It is called at the very end and takes in the basic geometries of the best aircraft in order to output a 3D model. It works by using voxel.m in order to plot boxes with the size and location of UAV components. There is one red box with the dimensions of the fuselage, and 3 green boxes with the sizes and locations of the wings. *Modelaircraft.m* takes in the sizes and locations as inputs to the function and then calls Voxel to generate the boxes. To rotate the view in the MATLAB figure you have to click the little hand and drag it around to rotate the view.

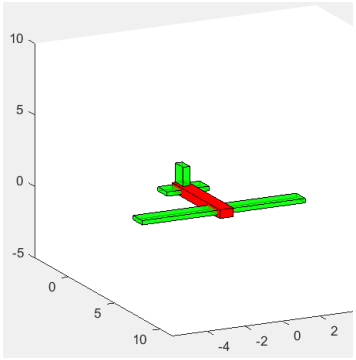
7. *Voxel.m* - this is a function which I found online and it models boxes based on input sizes and locations. The location, size, opacity, and color are input. *Modelaircraft.m* calls this function with the appropriate geometries of the fuselage, wings, tail, and vertical tail and each is merged to create an elementary aircraft like shown below:



N=5000



N=10,000



N=50,000

Voxel.m Inputs: start,size,color,alpha
start is a three element vector [x,y,z]
size the a three element vector [dx,dy,dz]
color is a character string to specify color
Alpha is the opacity
Voxel.m Outputs: No outputs besides plotting the boxes

Modelaircraft.m Inputs: lf,wf,D,space,c,b,ch,bh,cv,bv (aircraft dimensions)
Modelaircraft.m Outputs: None

Now that we have a general idea of how the optimization code works, we can discuss our process for optimizing the code. Naturally we had a lot of debugging to do in order to get all of these functions to work together seamlessly. Once the debugging was done though, we got a

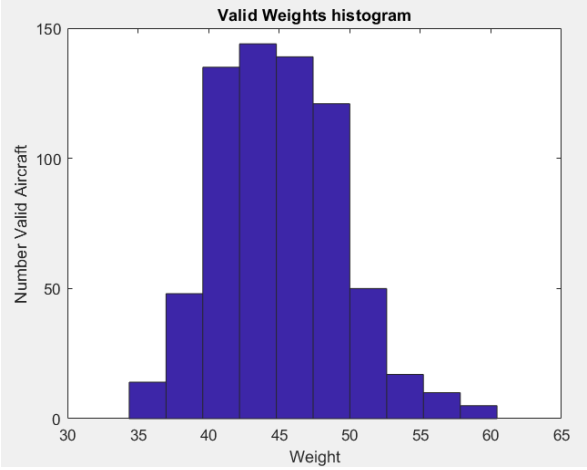
very wide range of weights and also a large amount of aircraft that were unstable or clearly wrong looking. The following strategies were utilized in order to fine tune our optimization code.

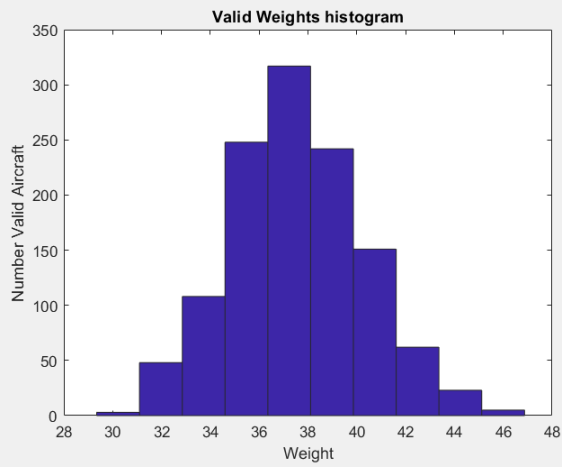
Optimization Strategy

Our first iteration of optimization code successfully generated an “ideal” aircraft but we felt that this needed some more fine tuning. Although the code identified the ideal dimensions, we felt that the ranges of randomized geometries, as well as requirements for being a “valid aircraft” could be adjusted for better optimization.

Firstly, we noticed that some of the ideal aircraft returned by our code had some invalid geometries such as the horizontal tail being bigger than the main wings, or the vertical tail having a wingspan bigger than the fuselage. To address this, we added in some conditions for being a valid aircraft which prevent these anomalies. In order to quickly identify aircraft with invalid geometries and refine our code, we wrote a function called ‘modelaircraft’ which generates a 3D model of the best aircraft. This function was discussed earlier.

Secondly, the histogram of valid weights was skewed heavier than we anticipated. We knew the weights of similar aircraft fell within the 40-45 range, yet we were hoping to make ours lighter for longer endurance. After fixing our servos, we saw the histogram shift lighter.



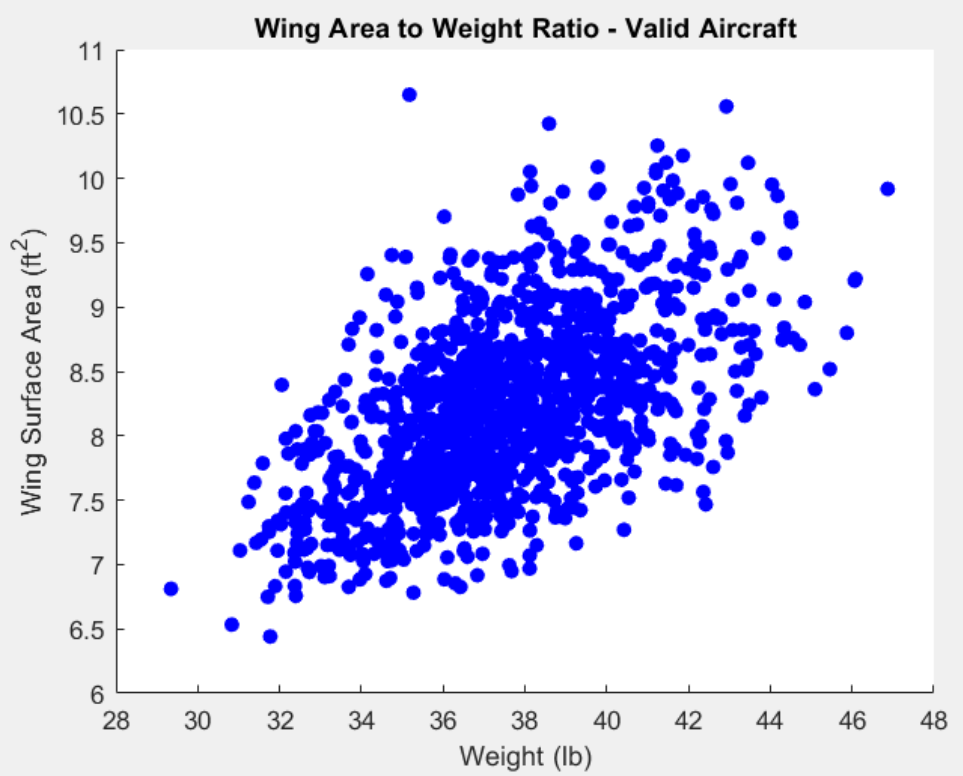


Additionally, we saw that some of the optimized parameters tended to skew towards the extremes of the ranges. Additionally, some of our functions returned errors if we fed in invalid geometries like 0. If this was the case, we tried adjusting the range and tried seeing what happened. This fortunately fixed any errors and also increased the endurance by a several hours. Below is a table depicting all of the major ranges that we changed from our first iteration to our final iteration.

Wings			**Horizontal Tail**		
	Initial Range	*Final Range*		*Initial Range*	*Final Range*
Area (ft^2)	3.9 - 74	3.9 - 74	*Area (ft^2)*	0 - 5	1 - 3
Span (ft)	1.9 - 28.5	2 - 28.5	*Span (ft)*	0 - 6	1 - 4
Sweep (deg)	0 - 30	0 - 30	*Sweep (deg)*	0	0
Taper ratio	0 - 1	0.2 - 1	*Taper ratio*	1	1
Aspect Ratio	0 - 5	1 - 11			
Fuselage			**Vertical Tail**		
Length (ft)	3 - 15	Space+c+ ch + (0-6)	*Area (ft^2)*	0 - 5	1 - 2
Max. Width (ft)	0.5 - 2.5	0.5 - 2.5	*Span (ft)*	0 - 6	0.5 - 1.5
Max. Depth (ft)	0.5 - 2.5	0.5 - 2.5	*Sweep (deg)*	0	0
LE Fuselage to LE wing (ft)	0 - 5	1 - 4	*Taper ratio*	1	1

Having ranges start at 0 sometimes caused errors in our calculations and also didn't make sense geometrically. We adjusted almost all of the ranges to not start at 0. Furthermore, if we noticed that a certain parameter always wanted to be on a certain extreme, we shift the range up or down accordingly. One example of this was Aspect Ratio. Initially, every ideal plane had an A value of 4.9. Once we changed the range to be 1-11, the ideal aircraft settled around 8 or 9.

Additionally, we originally defined the length of the fuselage to be an arbitrary length. However, this would sometimes result in invalid geometries such as the tails not being on the fuselage due to space (LE Fuselage to LE wing) also being randomized. We realized it made more sense to have the length of the fuselage be a function of the other measurements which are always changing. In our final code, this is at a minimum, the wing's chords plus the distance from the leading edge. At a maximum, it is this length plus 6.



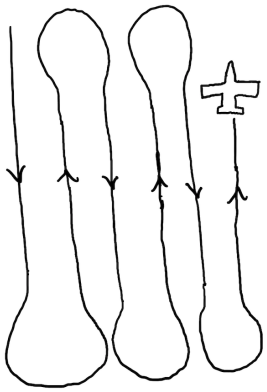
Overall, this was our distribution for Wing Surface Area vs Weight of valid aircraft for N=25,000. We found that generally, the ones in the left half had better endurance due to the reduced structural weight. Additionally, greater wing area also benefited the endurance so the most successful aircraft tended to fall towards the upper left boundary of this shape. Our final aircraft had an S value of 8.6 ft^2 and a weight of 36 lb.

Autonomous Strategy

Mission

Our UAV's mission involves taking off from virtually any location, flying directly to the wildfire, generating a thermal map of an area of concern while simultaneously locating the "hot spots" of the wildfire, and then returning to its takeoff point once the mission has been completed. To integrate the thermal imaging with autonomous strategy, we modified the autopilot to visit the waypoint with the most thermal activity on the way back to the start point.

In order to accomplish this feat in the most efficient manner possible, we decided that the UAV would survey the area of concern in a sweeping motion with the distance between each lap being equivalent to the peripheral vision of our camera. Thus, the flight path was set up as follows:



By modifying the function labeled *wayguidcopy.m*, we were able to generate an array of way points that our aircraft would follow during its mission. The inputs of this function consist primarily of the UAVs takeoff location, the location of the wildfire, and the radius of the wildfire. The area that our UAV will fly over will be centered around the location of the wildfire, and the side length of this region will be equivalent to the diameter of the wildfire. Using the following section of code, we were able to generate the number of way points necessary to complete a scan of the area determined by the user's inputs:

```
for i = 2:4:2*N-2
    waypoints(i+1,1) = waypoints(i,1) + 0;
    waypoints(i+1,2) = waypoints(i,2) + camscan;
    waypoints(i+2,1) = waypoints(i+1,1) - y;
    waypoints(i+2,2) = waypoints(i+1,2) + 0;
    waypoints(i+3,1) = waypoints(i+2,1) + 0;
    waypoints(i+3,2) = waypoints(i+2,2) + camscan;
    waypoints(i+4,1) = waypoints(i+3,1) + y;
    waypoints(i+4,2) = waypoints(i+3,2) + 0;
end
```

Since our UAV must be capable of taking off from virtually anywhere, we had to account for four different scenarios depending on whether the takeoff point of the UAV was north, east, south, or west of the fire's location. In each scenario, the first way point of the flight path is set up to be the closest corner of the area of concern to the takeoff location. Then, the UAV follows a similar, but different flight path depending on which corner it arrives at first.

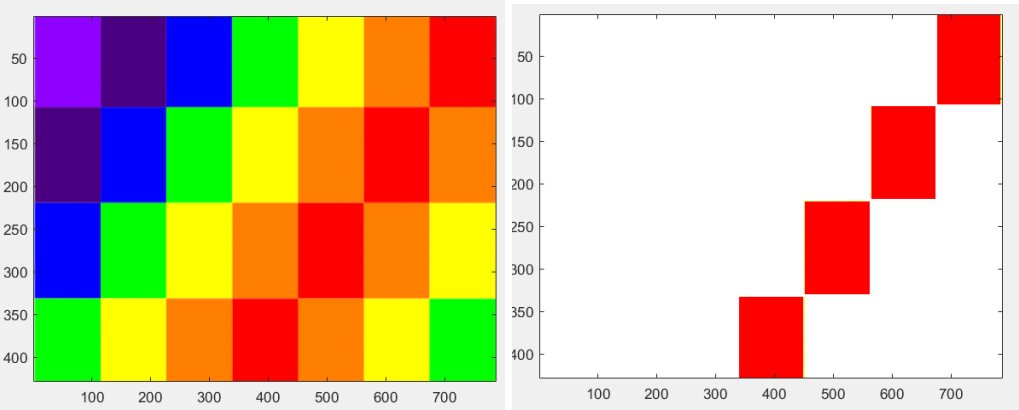
At each waypoint, the code reads in an image and indexes how much thermal activity there is by using the *HeatMapCode.m* function. Presumably, the image would be an infrared scan taken in by the camera. However, for our purposes we just had it use the same image every time. The wayguide function keeps track of which waypoint was the hottest one and makes it the second to last waypoint, meaning that the UAV will circle back past the hottest waypoint before heading back to the home base. Since we used the same image every time, the waypoints are all assigned the same "hotness" value. If this is the case, we programmed the autopilot to circle

back to the second waypoint (since the first one was already on the path back to the launch point). To summarize, our UAV:

1. Flies from it's launch point to the nearest corner of the fire.
2. Completes a sweep surveying the fire, with each scan being one camera angle away from the last
3. Takes a thermal snapshot at each waypoint
4. Loops back to the hottest identified waypoint after completely surveying the entire fire radius
5. Returns to launch point

Heat Mapping

In order to make our UAV truly autonomous, we had to create the function *HeatMapCode.m*. This function utilizes MATLAB image indexing to isolate the pixels in any given image that are a certain color.

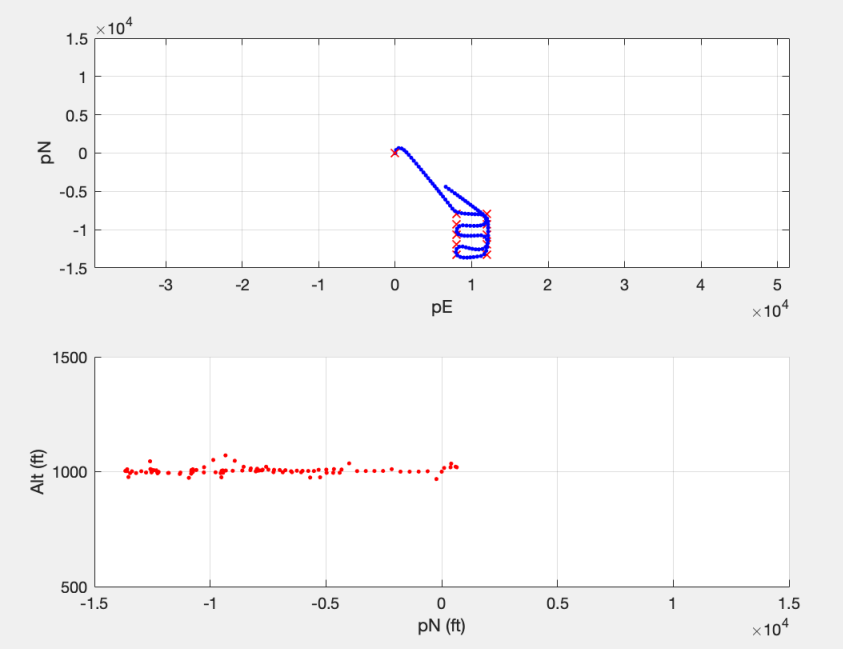


The function then returns the number of pixels in the image that were identified. And, since our UAV is capable of thermal imagery, this function allows us to determine the amount of thermal activity at every way point along its route.

The function works by taking in an image. In this case, it uses HeatMap.png. It then looks at every pixel's RGB value. If the Red value falls above a certain threshold while the Green and Blue stay below a certain threshold, the pixel is identified as red and the count of hotpixels is increased by one. Once the code has looked through every pixel, the count of hotpixels is returned. For reference, the amount of hotpixels identified by the code in this image is 48,474.

Simulation

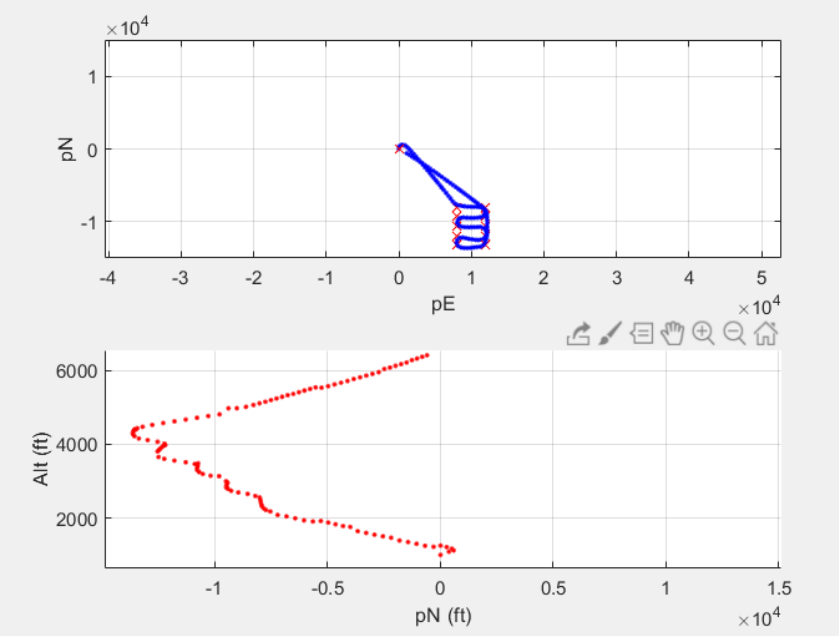
Once the UAV has completed mapping the area of concern, it then returns to the way point with the greatest amount of thermal activity before returning home. Utilizing *init_dynamics.m, aero_dyn.m, get_coeffs.m, and show_map.m*, in which we inputted our aircraft's specifications, we were able to obtain the following sample flight path from Simulink:



The function *wayguidcopy.m* is set up to only update the way point with the greatest amount of thermal activity if *HeatMapCode.m* returns a value that is greater than it did previously. However, since we are using the same image every time, they all have the same index. If this is the case, we programmed the autopilot to return to the second way point before returning home. Initially, we had it return to the first one, but this was already on the way back to the launch point so we changed it to the second waypoint as a proof of concept that it is capable of circling back to the hottest waypoint. Since a standard return value of the heat mapping function is around 48,000 it is extremely improbable that any real scenario would return all snapshots with the same hotness value.

It is important to note that this simulation utilized a few pioneer stability derivatives. We were able to calculate all of our stability derivatives and integrate them in the code we submitted, but this change results in the altitude graph becoming unstable and we were unable to identify the reason. More importantly though, the simulation still successfully works with our *HeatMapCode.m* and loops back to the hottest identified waypoint before returning to base.

The output of the autopilot simulation that used the stability derivatives found using XFLR5 (with the exception of a couple of signs changed to avoid errors) is shown below. As stated before, the altitude increases as the UAV completes its path when it was expected to hold, a reason for this could not be found. The magnitudes of some of the stability derivatives were altered to mimic those given for the pioneer UAV but resulted in the same issue.



Limitations

Currently, our UAV is only capable of one autonomous mission. Upon its launch, the Fireflighter is able to fly directly to the wildfire, scan over the area of concern, and fly to the hottest spot of the wildfire before returning home. Since our goal should be to encompass as wide of a range of missions as possible, we must equip our UAV with more autonomous capabilities in the future.

Another limitation at the moment relates to how we are currently determining which locations are “hot spots.” Currently, our autonomous function only calls upon *HeatMapCode.m* to analyze an image at every way point. Ideally, we want to be able to call upon this function every time an image is taken rather than solely at the way points since in reality, an immense amount of images will be taken in between way points. Additionally, we would need to know the coordinates associated with each image. One idea we had to address this concern is to add in even more waypoints along the way. The UAV would still take snapshots at every waypoint, but there are more snapshots being taken per journey. We had another idea in mind but it fell outside the scope of the class so we decided on the current waypoint snapshot method. Our alternate idea was to have the UAV send a GPS ping to a base whenever the thermal activity identified by the camera fell above a certain threshold. This way, the base would have a map of

all the hotspots, rather than just the location of the hottest one. However, without research and development, we wouldn't know what thermal activity threshold to set or how to best construct a map of real time coordinates identified by the UAV.

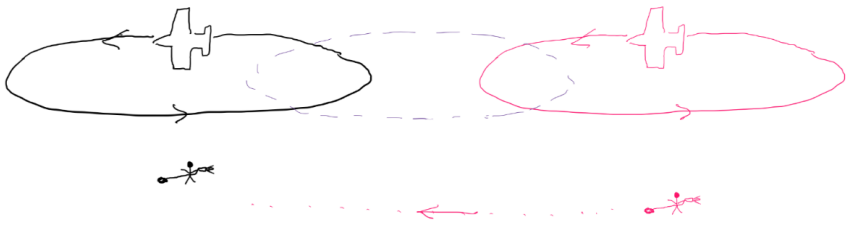
Future Work

We are not sure what caused the altitude plot in Simulink to become unstable, but this would be the first thing to address in the future. Since the Simulink worked with the derivatives of previous aircraft iterations, it can be inferred that the error lies somewhere in the most recently implemented values. There are some small discrepancies between the dimensions and values of the plane design in XFLR5 with those outputted by the optimization code although we could not find the reasoning behind this, this could be a reason for inaccurate stability derivatives.

Our UAV is capable of identifying “hot spots,” a feat we believe will prove very useful for any wildfire application, our future work will primarily involve creating new autonomous capabilities related to this topic.

An additional capability that we would like to pursue in the near future is that of tracking. Since wildfires are constantly moving, the location that is determined to have the greatest thermal activity may have moved by the time our UAV returns to it. Thus, once our UAV flies back to the location it initially stored as the hottest spot, it should conduct a sweep of the area surrounding this location to find its new location. Once this new location has been found, the UAV would track the movement of the “hot spot” and circle over it until an Air Tanker drops its payload.

Another application that we believe would be useful to firefighters involves tracking as well. This time, however, the tracking feature would be used to follow around firefighter crews and circle above them as they encounter areas that they have little information about. Utilizing our heat mapping capabilities, the UAV would be able to determine the thermal activity of the area surrounding the firefighter crew and relay that information to firefighters on the ground. This would ensure firefighter safety, as these crews would be notified immediately if the wildfire comes within a dangerously close distance of them or if the wildfire is about to surround them. The flight path for this scenario would be as follows:



Another feature that we would like to modify in the future is heat mapping. As we mentioned earlier, we want to be able to call upon *HeatMapCode.m* every time an image is taken rather than solely at the way points. In order to accomplish this with the functions we currently have, we would simply need to update our way points array so that it generates a new way point every time an image is taken, even if this does not alter the UAVs flight path. Even though our aircraft satisfies our initial mission requirements, a few more modifications are necessary to make our UAV adaptable to any mission involving wildfires.

As discussed earlier, the XFLR5 program was used to calculate some of the stability derivatives of the aircraft. The program uses 2d airfoil data and uses approximations to convert them into 3d cases. The program uses various methods such as the panel method to simulate how the plane and surrounding air react due to flight. The program's ability to obtain accurate stability derivatives was never tested against that of other programs such as DATCOM. Since a wind tunnel cannot be used to get the values for the UAV, stability derivative values could be averaged based on the results of multiple programs.

The basis of our UAV design is built upon weight estimates found using the Niccolai equations, these equations may not necessarily be accurate for small aircraft such as the proposed UAV. There are other methods available to estimate the weight of components which may be more accurate such as the Torenbeek method [8]. It would be interesting to use different methods to estimate the aircraft's weight and see how they affect the optimization process and final UAV result.

The center of gravity calculations used to determine the static stability of the aircraft also use many different approximations and assume an even mass distribution for each component. In the future, this process could be made more accurate to produce better statically stable aircraft.

Appendix

References

Figure A.1 Design Chart for a Propeller with 2 Blades

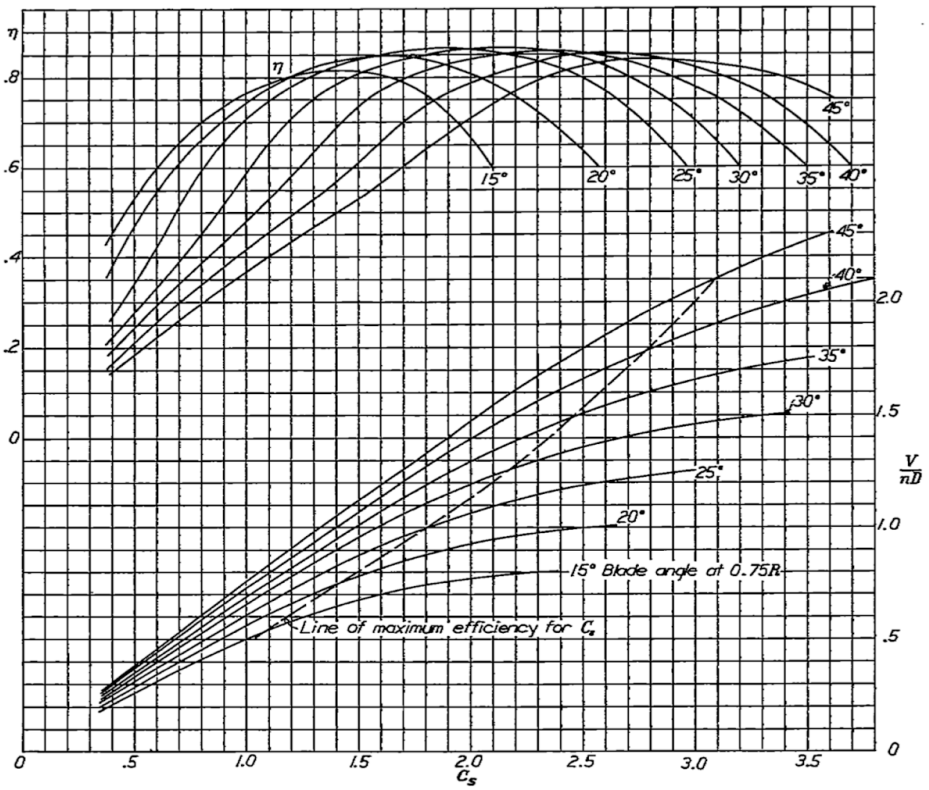


FIGURE 7.—Design chart for propeller 5868-9, Clark Y section, 2 blades.

Figure A.2 Efficiency curves for a Propeller with 2 Blades

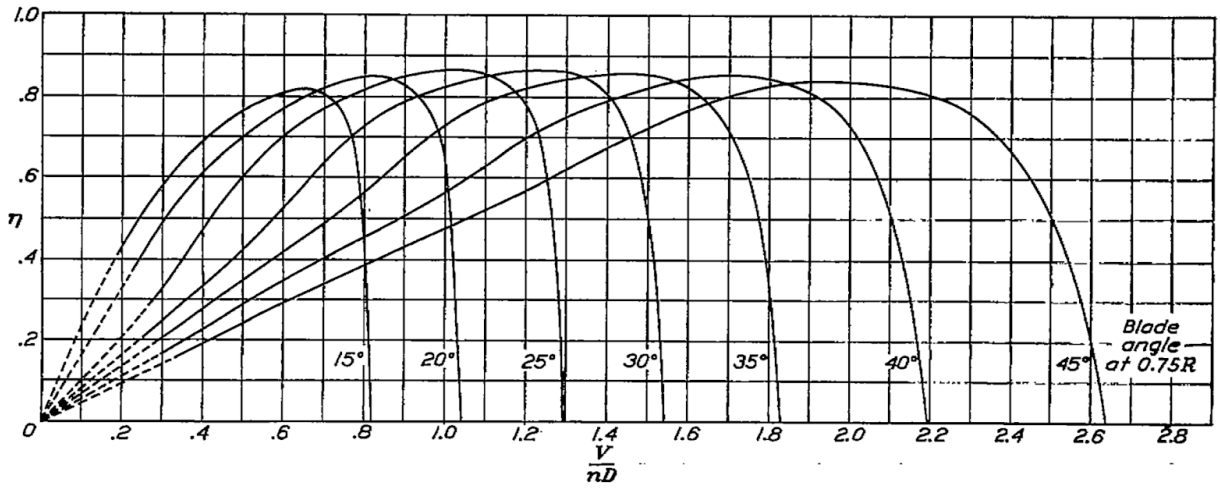
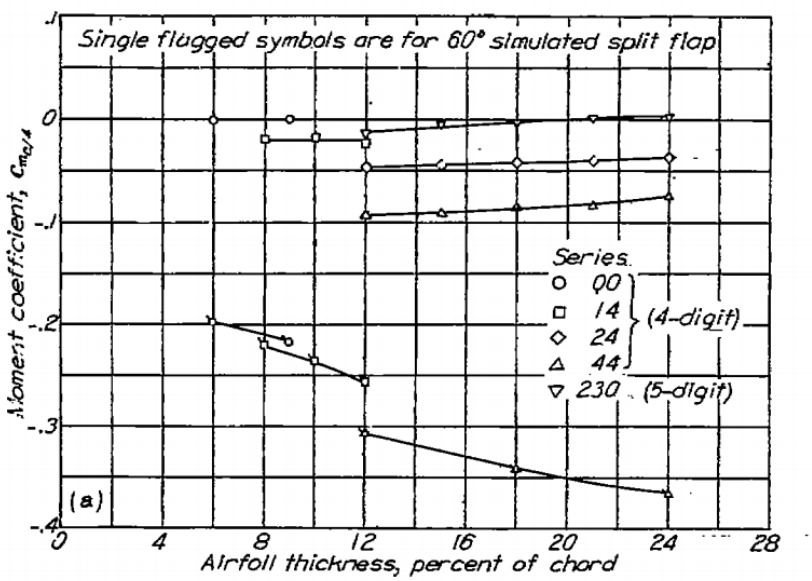


FIGURE 5.—Efficiency curves for propeller 5868-9, Clark Y section, 2 blades.

Figure A.3 NACA Four and Five-Digit Series



(a) NACA four- and five-digit series.

Online Sources

[1] https://hitecnology.com/actuators/linear-series/hls12/product
[2] http://airfoiltools.com/airfoil/details?airfoil=goe803h-il
[3] http://airfoiltools.com/airfoil/details?airfoil=n0012-il
[4] https://www.grc.nasa.gov/www/k-12/VirtualAero/BottleRocket/airplane/acg.html

[5] https://www.fzt.haw-hamburg.de/pers/Scholz/HOOU/AircraftDesign_13_Drag.pdf
[6] https://www.fzt.haw-hamburg.de/pers/Scholz/OPerA/OPerA_PUB_DLRK_12-09-10.pdf
[7] https://www.mh-aerotools.de/airfoils/hdi_aoawing.htm
[8] http://www.ijemr.net/DOC/AircraftMassEstimationMethods(170-178).pdf
[9] https://www.uavfactory.com/product/77
[10] https://www.flir.com/products/neutrino-performance-series/
[11] https://www.cofdm-transmitter.com/sale-11186236-30km-wireless-range-uav-video-transmitter-for-pixhawk-telemetry-and-ptz-signal.html
[12] https://www.harborfreight.com/12V-Lithium-Ion-20-Ah-Compact-Lightweight-Battery-56566.html
[13] https://www.sbg-systems.com/products/ellipse-2-series/#ellipse2-n-miniature-ins-gnss
[14] https://ainstein.ai/drone-makers-drone-service-providers/lr-d1/
[15] https://www.uavos.com/products/autopilots/ap10-2-automatic-control-system-for-uav-mini
[16] https://www.uavfactory.com/product/53