Team Tempest

AA241X: Final Report
June 08, 2010

Amrita Mittal

Gavin MacGarva

Harsh Menon

Kevin Reynolds

Kuldeep Lonkar

TABLE OF CONTENTS

TABLE OF CONTENTS ii
LIST OF FIGURES iv
LIST OF TABLES v
1 INTRODUCTION 1
2 CONCEPTUAL DESIGN 5
2.2 Design Philosophy 7
3 AIRCRAFT DESIGN 8
3.1 Airfoil Design and Selection 8
3.2 Modules in Optimization 9
3.2.1 Weight Estimation Module 9
3.2.2 Aerodynamic Analysis Module 10
3.2.3 Structural Analysis Module 13
3.2.4 Altitude Gain Estimation Module 13
3.3 Optimizer 16
3.4 Optimization Results 17
3.5 Stability Analysis and Tail Design 19
3.5.1 Objectives 19
3.5.2 Analysis Tools 19
3.5.3 Design 20
4 CONSTRUCTION AND CAD MODEL 29
4.1 Construction Methods 29
4.1.1 Fuselage 29
4.1.2 Wing 34
4.1.3 Empennage 37
4.2 CAD Model 38
4.3 Weight Breakdown 39
5 CONTROL LAW AND SIMULATIONS 41
5.1 State Machine 41
5.2 Selection of Control Methodology 42
5.3 Control Law 45
5.4 Control Law Simulations using Simulink 48
5.4.1 Propulsion Model 48
5.4.2 Implementation of Control Law in Simulink 49
5.4.3 Simulation Results 50
5.5 Implementation of Control Law 53
5.5.1 Details of the error formulation 53

5.5.2 Details of C30 implementation and MATLAB simulation 54
6 FLIGHT TESTS 56
6.1 Test Procedures 56
6.1.1 Stephano 57
6.1.2 Prospero 58
6.2 Flight Tests 58
6.3 Fly-off Flight Test Results 64
6.4 Fly-off Flight Test Data Analysis 67
7 LESSONS LEARNED 71
APPENDIX A: Microcontroller C30 Code 73
APPENDIX B: AVL Input Files 84

LIST OF FIGURES

1: Drag Polar at Re = 70000 8
2: Coefficient of drag for SD7043 12
3: Free body diagram of aircraft in climb 14
4: Maximizing altitude gain 15
5: Optimizer 17
6: Optimization results 18
7: Trimmed Stephano dimensional eigenvalues plot at sea level 23
8: Trimmed Prospero dimensional eigenvalues plot at sea level and 10,000 ft MSL 28
9: Fuselage structure with boom attached 31
10: Completed Prospero Fuselage in the field 32
11: Joining the inboard and outboard sections 35
12: Joining the outboard sections 36
13: Prospero 36
14: Control surface hinge (left) and attachment to boom (right) 37
15: Completed empennage (Prospero) 37
16: Front and top view of Prospero 38
17: Isometric view of Prospero 39
18: State machine 42
19: Lateral dynamics block diagram 43
20: Desired control characteristics 44
21: Circular region 46
22: Propulsion block 48
23: Complete Simulink model 49
24: Autopilot block 50
25: Comparison between PD and PID control 51
26: Robustness with respect to launching location 52
27: Robustness with respect to winds 52
28: Rudder deflections before (left) and after (right) 55
29: Flight data from May 13, 2010 60
30: Flight data from May 25, 2010 60
31: Flight data from June 1, 2010 61
32: Climb and descent data from June 1, 2010 62
33: Climb data from June 1, 2010 62
34: Descent data from June 1, 2010 63
35: Variation of climb and descent radii 64
36: Flight data from June 2, 2010 65
37: Climb and descent data from June 2, 2010 65
38: Flight data from June 3, 2010 66
39: Climb and descent data from June 3, 2010 66
40: Total Altitude = $\sum \Delta h_{climb} + \Delta h_{upper\ transition} + \Delta h_{lower\ transition}$ 68
41: Rate of climb for different states 68
42: Estimation of true airspeed 69

LIST OF TABLES

1: Weight breakdown of the wing 10
2: Design breakdown 18
3: Stephano Tail Geometry 22
4: Trimmed Stephano Eigenvalues at Sea Level 23
5: Effects of design changes analyzed on Dutch roll 25
6: Effects of design changes analyzed on spiral mode 26
7: Prospero Tail Geometry 26
8: Trimmed Prospero Eigenvalues at Sea Level 27
9: Comparison of Control Power Derivatives for Stephano and Prospero 28
10: Weight of required items 39
11: Weight breakdown 40
12: Comparison of control characteristics 44
13: Color convention 59
14: Tests results in comparison to theoretical estimates 70

1 INTRODUCTION

The following report summarizes the work of Team Tempest in the *AA241X – Design, Construction, and Testing of Autonomous Aircraft* course offered by the Department of Aeronautics and Astronautics at Stanford University. The course allowed students with expertise in the disciplines of conceptual design, applied aerodynamics, structures, and guidance and control to apply conceptual design techniques from each discipline in the design, construction, testing and evaluation of a small, autonomous aircraft. The necessary electronic components and raw materials were provided, however the design configuration, construction methods, and control law development was left as a design decision for each team. The course was offered during the spring quarter of 2010 which encompassed the dates of March 29th to June 8th. All work was to be completed during this time and teams were expected to deliver a UAV capable of meeting mission requirements by the fly-off day at Lake Lagunita on the Stanford University campus set for June 2nd, 2010.

1.1 Mission Requirements

Author: Gavin MacGarva

The course presented teams the following mission statement:

1. Design, build, and fly an autonomous aircraft.
2. Electric-powered aircraft must climb as high as possible (total vertical altitude increase) under its own control and land back in a designated area, recording its position.
3. Control system hardware and propulsion system will provided and it is not subject to change.

FAA regulations limited the ability of teams to demonstrate the second mission requirement on the fly-off day, restricting the maximum altitude for unmanned aircraft to 400 ft. The total altitude capability was demonstrated by a series of powered climbs and power-off

descents, and the net change in altitude during the climb phase of each segment was considered the total altitude gain. Despite this artificial environment, the teams were still required to present an aircraft and supporting data demonstrating that the aircraft was capable of a direct climb to the highest altitude possible, handling the stability and control issues associated with less-damped dynamic modes and increased wind velocities at altitudes well above 400 ft.

The third mission requirement limited the control system to GPS and actuators/servos. Sensor data available to a control law was therefore limited to data provided by the GPS board only which gave latitude, longitude, and altitude. No sensors to determine the orientation of the aircraft were allowed. Airspeed sensors were available but only if teams demonstrated an autonomous flight without the sensor before the fly-off day. The control system was further constrained to keep the aircraft within the bounds of Lake Lagunita with no pilot input and land within 50 meters of the launch point. The propulsion system was limited to a small electric motor powered by Lithium Polymer batteries, both of which were provided to teams and could not be changed.

1.2 Team Organization

Author: Gavin MacGarva

In order to more efficiently satisfy the mission requirements, Team Tempest divided the work into five disciplines: aerodynamics, stability analysis and empennage design, simulations, control, and structures/CAD modeling. Each team member was given primary responsibility in each of the disciplines, although most team members contributed to other disciplines as well in varying degrees depending on the assigned discipline throughout the course. All team members participated in the construction process. More detail descriptions of the work done by each team member throughout the course is provided below:

Amrtia – *Aerodynamics & Optimization Lead*. At the beginning of the quarter Amrita's efforts were focused on development of an initial wing design. After the first plane was manufactured, she started working on linking different analysis modules together and optimizing the second design to achieve the best climb performance. She also modified the weight model for

wing to get better match with manufactured wing. The aerodynamic analysis code was simultaneously refined to include analysis using Xfoil and Tornado. As the need to build the second plane grew, she and Kuldeep designed the second fuselage and took lead in manufacturing the wing for Prospero. She also analyzed the flight data for the aerodynamic performance after each testing which was used to optimize the elevator settings in flight.

Gavin MacGarva – *Stability Analysis/Empennage Design Lead*. At the beginning of the quarter Gavin's efforts were focused on the development of an initial weight model which would be used to estimate the weight of a given design such that its performance could be predicted. This model would evolve over the course of the quarter and be incorporated into the optimization process. After the development of the initial code was complete his efforts shifted to the primary responsibility of design of the empennage and stability analysis through the use of the Athena Vortex Lattice (AVL) software. He also worked to provide the simulations lead with stability derivative information of the various designs for the 6 DoF simulations and was the R/C pilot during flight testing.

Kuldeep – *Simulations Lead.* At the beginning of the quarter Kuldeep's efforts were focused on how to use AeroSim blockset in MATLAB to simulate 6 DoF simulations. Then Kuldeep and Harsh worked on maximum altitude module, error formulation and control law design. Once error formulation and control laws were finalized Kuldeep implemented it in simulations and ran simulations for different types of controls (PD / PID etc) and gains to understand their effects on performance of aircraft in different conditions like different launching positions & winds etc.

Harsh – *Control Lead.* Harsh wrote maximum altitude gain module along with Kuldeep; and initially worked on error formulation and control law design. Harsh learnt how to program autopilot board and once the error formulation and control laws were finalized, he implemented them in C30 programming language. Harsh also designed the state machine for the mission and evaluated the performance of different control strategies like P, PI, PD and PID, from the transfer

function of the system. Harsh, later, focused on modifying the code, debugging and data analysis (plotting data, turn performance in climb and descent).

Kevin – *Fuselage Manufacturing & CAD Modeling Lead.* Kevin initially worked on manufacturing of first fuselage. He, later, focused on creating a CAD model of Stephano and Prospero in SolidWorks. Inertia and CG location obtained from CAD model were used in 6 DoF simulations in Simulink. Weight estimate obtained from CAD model were also used in the optimizer.

Apart from these specific tasks, everyone participated in manufacturing.

2 CONCEPTUAL DESIGN

2.1 Configuration

Author: Gavin MacGarva

The first decision Team Tempest faced was the selection of a configuration that would allow the team to best satisfy the mission requirements. The fundamental philosophy was to choose a configuration that offered the greatest potential for success with minimal risk. Given that the entire design, construction, testing and evaluation process was limited to the span of one quarter, the key risk metric was time. Lessons learned from previous years indicated that the development of an adequate control law was typically the most challenging issue faced by teams, usually handled late in the quarter when the number of days available to test before the fly-off day were limited. In light of these lessons from previous classes, the team decided to push to have as much flight testing time available as possible to develop the control law. Therefore, Team Tempest needed to choose a design would not take up much time in other areas. Some of the key considerations, in order of importance were:

1. *Stability and Control* – Mission requirements put significant limitations on the control system and the sensor data provided to it, therefore it was imperative to choose a configuration that could be controlled easily with no sense of orientation in pitch, yaw, or roll.
2. *Survivability* – From analysis of AA241X reports from previous years, it was clear that crashes were inevitable. In order maximize efficiency of the efforts of the team, it was considered important to have a design that could survive crashes with limited damage such that efforts could be spent on the mission critical areas of control law testing and development rather than repair/rebuilding.
3. *Ease of Design and Construction* – It is not enough for a design to be survivable. In the event of a catastrophic crash in which the aircraft is a total loss (a relatively common occurrence in previous years) construction of another aircraft must be done

quickly in order to return the team to flight testing and progression towards meeting mission requirements.

4. *Aerodynamic Performance/Weight* – Given the fixed propulsion system it was readily apparent that having an efficient, lightweight design would best match the second mission requirement given that the amount of energy available to the propulsion system was fixed. This consideration, however, was placed last as the first three were essential to the development of the control law and if the control law was not properly tuned the constraint for the aircraft to operate completely autonomously from takeoff to landing could not be completed and the overall mission would be a failure.

A flying wing/blended wing body design was immediately removed from consideration as rudder/elevon mixing and limited C.G. range would make it a serious challenge and very time consuming from a stability and control point of view. A twin boom/pusher configuration was also removed from consideration due to the added weight of a second boom, which would likely make the benefit from the promotion of laminar flow over the wing by keeping it clear of the propwash negligible.

This led the team to consider two configurations in depth: a canard/pusher configuration and a conventional/tractor configuration. The canard/pusher configuration was an attractive option because it was aerodynamically efficient (both the wing and control surface would generate positive lift in a trimmed climb), and the wing would be kept out of the propwash, promoting laminar flow over the lifting surface. It did however, have some significant drawbacks. Due to the location of the wing and C.G resulting from the pusher configuration, it would likely require a large vertical tail (resulting from a small l_v which would require a large planform area for a fixed tail volume), increasing the weight of the design. Performance would also be very sensitive to the design of the canard and most importantly, the control surface would likely be the point of impact and completely destroyed in a crash, requiring more rebuild time.

The conventional/tractor configuration was an attractive option because it would be easier to design the control surfaces and would be more robust in a crash as the fuselage, not the relatively fragile control surface, would be the likely point of impact in a crash. However, the

tractor configuration would place the wing directly in the propwash, disrupting laminar flow and the design overall was not as aerodynamically efficient as a canard design due to the down force required on the horizontal tail for trim.

After weighing the relative merits of both configurations, Team Tempest decided to utilize a conventional/tractor configuration. It was surmised that the advantages in aerodynamic efficiency of a canard/pusher design over a conventional/tractor design would not be worth the increased risk in the time metric resulting from the disadvantages in robustness and increased complexity and sensitivity of performance to the design of the control surfaces.

2.2 Design Philosophy

In order to best satisfy mission requirements, Team Tempest began the quarter with the intent to construct two aircraft. The first aircraft, *Stephano*, would serve as test platform for the autopilot. It would be slightly over designed in order withstand crashes which would be of higher probably when the control law was in its first stages of development. Given that the team had little manufacturing experience, the first aircraft would also serve as a tutorial for the team in construction techniques and lessons learned would be incorporated in the second aircraft, *Prospero*, which would be a more aggressive design used for the fly-off.

3 AIRCRAFT DESIGN

3.1 Airfoil Design and Selection

Author: Amrita Mittal

Choosing the right airfoil for the mission is the most important step in wing design. Because of lack of much experience and time constraints, the group dropped the idea to design and optimize an airfoil for this mission and decided to go with a pre-existing airfoil which has been well tested in wind tunnels. The first step in airfoil selection was to estimate the operating Reynolds number which was obtained from previous aircrafts of similar size. Various pre-existing airfoils which are known to perform well in the desired Reynolds number regime were analysed in Xfoil for their lift, drag and moment characteristics. The selection criteria for airfoil choice were:

1. Benign stall characteristics
2. High C_{lmax}

Various airfoils were analyzed in X*foil* assuming an $N_{critical}$ of 7 for high turbulence levels (Figure 1).

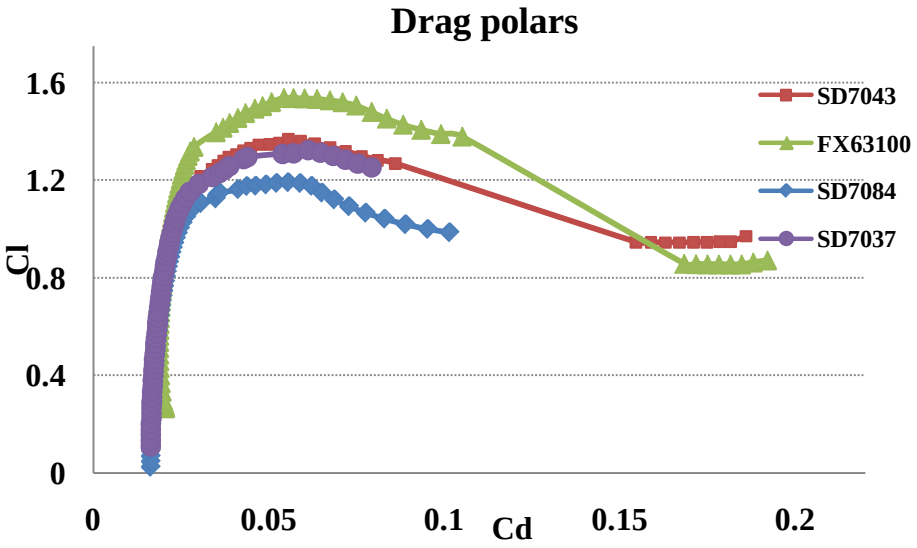


Figure 1: Drag Polar at Re = 70000

Most of the low Reynolds number airfoils have thickness to chord ratio in the range of 8% to 10%. All the airfoils analyzed have benign stall characteristics and a C_{lmax} of close to 1.3. FX63100 however has a relatively high C_{lmax} of 1.5 but it has a 10% thickness to chord ratio. Wing weight is proportional to airfoil thickness and therefore the next best airfoil SD7043 which is 9.1% thick was used. SD7043 performs very similar to FX63100 in the low to moderate Cl range which is the expected operational range.

SD7043 has a C_{lmax} of close to 1.3, zero lift angle of attack of -0.4 at Re = 70,000.

3.2 Modules in Optimization

There are three different modules used in the optimization process. Weight estimation module, aerodynamics analysis module and climb performance module are explained here.

3.2.1 Weight Estimation Module

Author: Gavin MacGarva

The initial weight model broke up the weight into four areas: the wing, fuselage/boom, tail, and required items. The weight of the wing in turn, was determined through the summation of four components: the foam core, carbon spar cap, fiberglass, and epoxy. The weight of the foam core was estimated by integrating the area of the selected airfoil spanwise over the chosen wing geometry and multiplying this volume by the density of the foam used. The weight of the carbon spar cap was estimated by determining a "weight per unit length" of the material and multiplying this value by the span of the design for which the spar cap would be included. The contribution of fiberglass to the weight was included by estimating the wetted area of the wing through the following equation available in the AA241A/B course notes and multiplying the result by the area density of the material:

$$S_{wetted} \approx 2.0\left(1 + \frac{0.2t}{c}\right) S_{exposed}$$

Based on advice from the TA, the weight of epoxy was estimated as the weight of the fiberglass. Later evolutions of the model would adjust this estimate based on wings the team manufactured throughout the quarter.

Weight break down for the wing of the second airplane was recorded for use by future groups to get an estimate of the weight of different components used in making the wing Table 1.

Table 1: Weight breakdown of the wing

Parameter	Weight (g)
Fiberglass cloth	26
Two spar caps	5.3
Center Section made with Spyder foam	28
Tips made of Dow foam	14
Total Weight before epoxy	73
Final Wing Weight	94

Initially the team had no data points to determine their own manufacturing abilities, therefore the weight of the fuselage/boom and tail was estimated based on a statistical analysis of data from previous years. The team determined the average and standard deviations of the weights of each of the components and the code allowed for the weight to be adjusted based on a design aggressiveness scale. This scale would specify how many standard deviations away from the mean the components would weigh. A "conservative design," for example, would weigh one standard deviation more than the mean for these components. Most evolution in the model throughout the quarter involved adjustments in this part of the code.

3.2.2 Aerodynamic Analysis Module

Author: Amrita Mittal

The aerodynamic module used in our optimization code takes as inputs the geometric parameters of the wing (aspect ratio, reference area, mean aerodynamic chord), the total weight of the plane, the climb angle and returns the coefficient of parasite drag, coefficient of induced

drag, coefficient of drag, coefficient of lift. Several simplifications were made to reduce the computational cost of the model.

First, tail design was not included in the optimization process because tail was expected to be robust to minor changes in the wing design and they don't weigh much. Tails were designed separately for the final optimum wing design. We simply assumed horizontal tail and vertical area to be 20% and 12% of wing reference area respectively in the optimization process. Second, the fuselage design process was not included in the optimizer because the design was mostly driven by packing space and location of the battery, and location of servos and the GPS antennae.

3.2.2.1 Drag Calculation of Fuselage, Boom and Tails

Parasite drag for the fuselage and tails was determined using AA 241A notes. Horizontal tail and vertical tail were made of flat sheets of balsa wood to save weight. Therefore the form factor, K, was assumed to be 1.05. The form factor for the fuselage depends upon the fineness ratio of the fuselage. Since our fuselage was not cylindrical, an effective diameter defined in Eq.1 was used to estimate the form factor using AA 241a notes.

$$D_{effective} = \left(\frac{4S}{\pi}\right)^{1/2}$$

3.2.2.2 Drag Calculation of Wing

A detailed analysis for drag estimation of the wing was done using Tornado Vortex Lattice Method and X*foil*. First, the section lift coefficient at different span-wise locations is computed using Tornado, a vortex lattice code implemented in MATLAB. The Tornado function was modified to take as input the wing geometry parameters (airfoil, twist, polyhedral fraction, span, root chord and taper ratio) and velocity conditions to estimate induced drag, the section C_L and root bending moment (used for structural analysis described in Section 3.2.3). Induced drag was also calculated using 241A notes and checked with results obtained from Tornado.

Information about sectional lift coefficient was used to estimate the parasite drag using X*foil*. To do this, a MATLAB based Perl script was written that takes the local coefficient of lift and Reynolds number as inputs, runs X*foil* for user specified turbulence and convergence characteristics and outputs a text file with information about the coefficient of drag, coefficient of parasite drag. This requires no input from the user. It turned out to be a very neat way to run X*foil* within the optimizer with no manual entry of inputs required. However, the optimizer would have problems when X*foil* failed to converge which was common for the low Reynolds number regime we are dealing with. In such cases, some X*foil* parameters including the iteration counts, the turbulence parameters had to be manually tweaked to achieve convergence. To circumvent this problem, X*foil* was run in a batch mode to generate coefficient of drag for a large number of C_l and Re values. A 3D surface was fit through the data and used to estimate C_{dp} for any R_e (Figure 2)

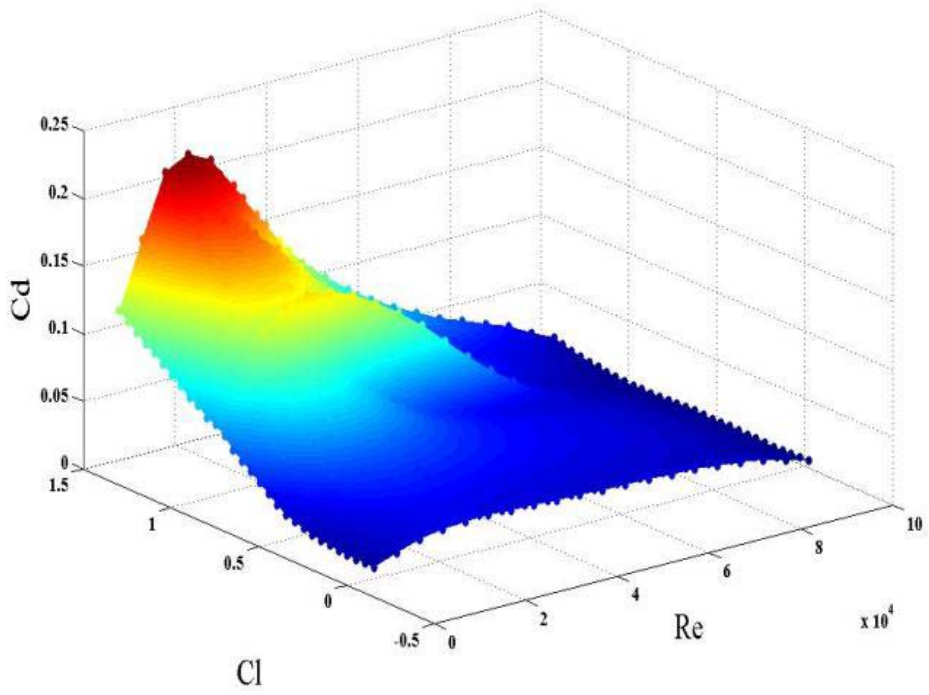


Figure 2: Coefficient of drag for SD7043

3.2.3 Structural Analysis Module

Author: Amrita Mittal

Tornado outputs the bending moment distribution over the wing. For each design a very simple structural analysis was done to make sure that the structure was stiff enough to take the bending loads. It was assumed that the spar caps carry all the bending loads. This made the calculation of second moment of area very simple.

Bending stress is given by: $\sigma = \frac{My}{I}$

Where, I = 2× (Area of spars) × (distance from the center)^2

M = output from Tornado

y = point of maximum stresses, which is the point furthest from the neutral axis

= airfoil thickness / 2

σ_{yield} for carbon strips in compression = 160000 psi

σ_{yield} stress for Spyder foam = 1400 psi

The Structures module of the optimizer checks each design for its structural strength. It was noted that all designs in consideration had bending stresses well below the yield stress. This meant we could try to use foam lighter than Spyder foam but less stiff without affecting the structural stiffness of the wing much. However, as will be mentioned in the section on manufacturing (section: 4), we had to cut the center section of wing in two parts at the root to include linear twist in our design. Since joints reduce the strength of section significantly, we decided to use Spyder foam in the center section but a ~25% lighter Dow foam at the tips in the second iteration. This also resulted in much better visibility of Prospero in flight.

3.2.4 Altitude Gain Estimation Module

Author: Kuldeep Lonkar

Objective function for altitude gain estimation module is, for a given aircraft configuration, maximize altitude gained till propulsion battery dies.

Altitude gained in time 't' seconds is given by: $h = \int_{t=0}^{t=t} \dot{h}\, dt$

For computational purpose it can be approximated as: $h \approx \sum_{n=1}^{n=n} \dot{h_n}\ \Delta t$

Where, Δt : time step size

$\dot{h_n}$: rate of climb at 'n'th time step, assuming it remains constant over time t = t_n to t_{n+1}

nf : time step when the battery dies

Here, Δt is positive and as we are only going to climb, we can assume that $\dot{h_n}$ is positive too. Hence maximizing total altitude gain 'h' is equivalent to maximizing $\dot{h_n}\, \Delta t$ at each time step, which is equivalent to maximizing $\dot{h_n}$ if time step is kept constant.

Now, our objective function becomes: for a given aircraft configuration, maximize rate of climb at each time step by varying climb angle and climb velocity.

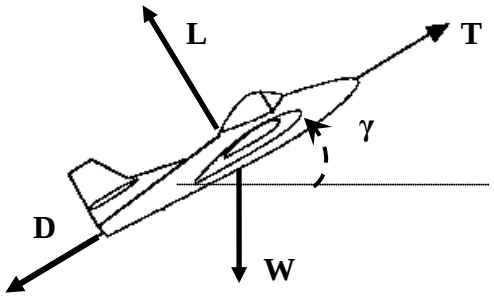


Figure 3: Free body diagram of aircraft in climb

Maximizing rate of climb at each time step:

Rate of climb is approximately given by: $\frac{dh}{dt} = V sin\gamma = \frac{(T-D)V}{W}$

Where,

T : thrust available D : drag

V : true airspeed of the aircraft γ : climb angle

W : weight of the aircraft, which remains constant in our case

Thrust and drag are implicit functions of velocity.

For the given propeller and motor, using motorPropeller.m code, we modeled thrust as a linear function of airspeed to make calculations simple.

$$T = -0.21\,V + 3.8$$

Steps taken in maximizing altitude gain can be represented by a flow chart (Figure 4):

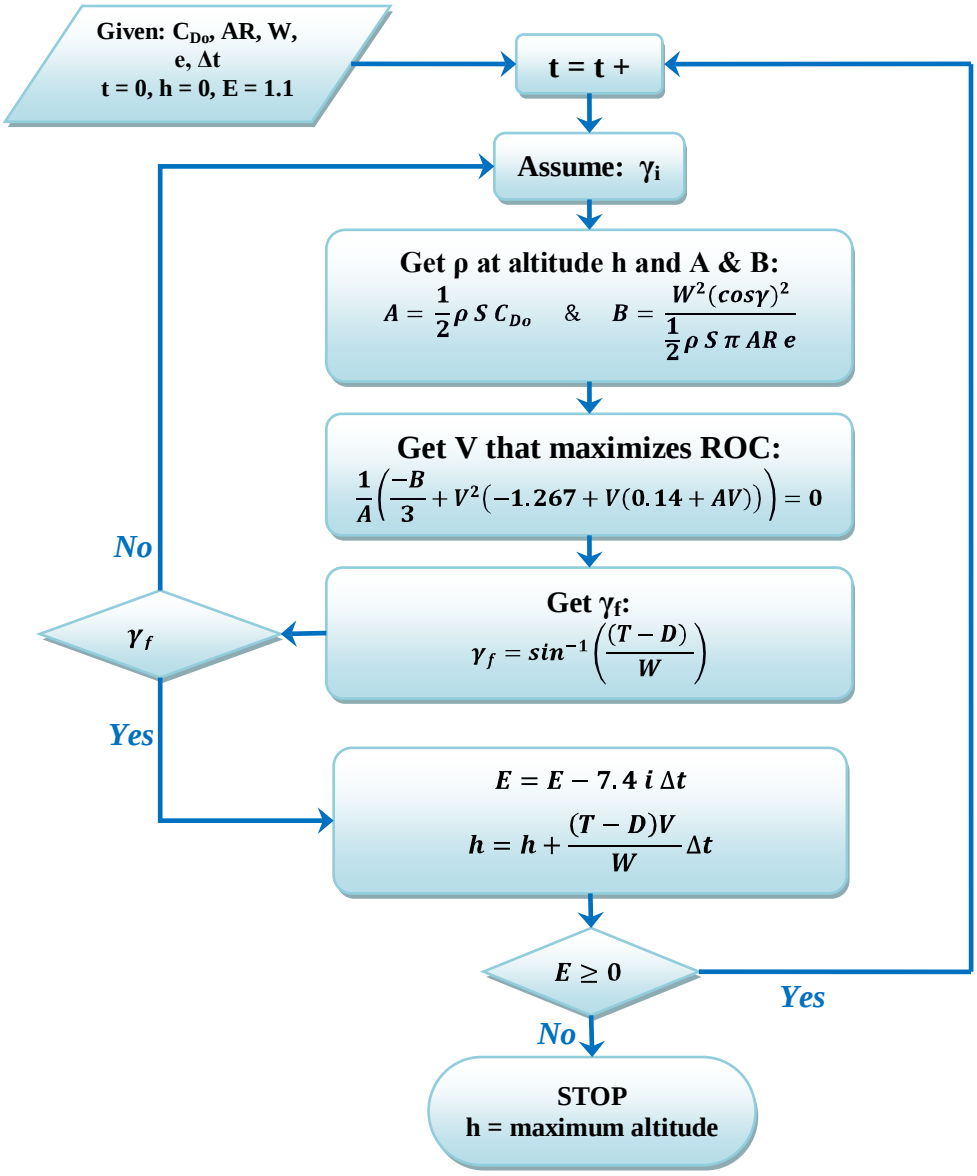


Figure 4: Maximizing altitude gain

3.3 Optimizer

Author: Amrita Mittal

A simple grid search method was used to search the design space for optimum designs. Architecture of the optimization is as follows:

Objective function: Total Altitude

Design Variables: AR, S_{ref}, Polyhedral fraction, Taper Ratio

Local Variables: Climb Velocity, Climb Angle

Constraints:

1) Manufacturing Constraint	Chord length > 10 cm everywhere
2) Stall Speed Constraint	Stall Speed < 6.5 m/s
3) Structural (chicken) Constraint	Root Chord > 12 cm

The flow of information is shown in Figure 5.

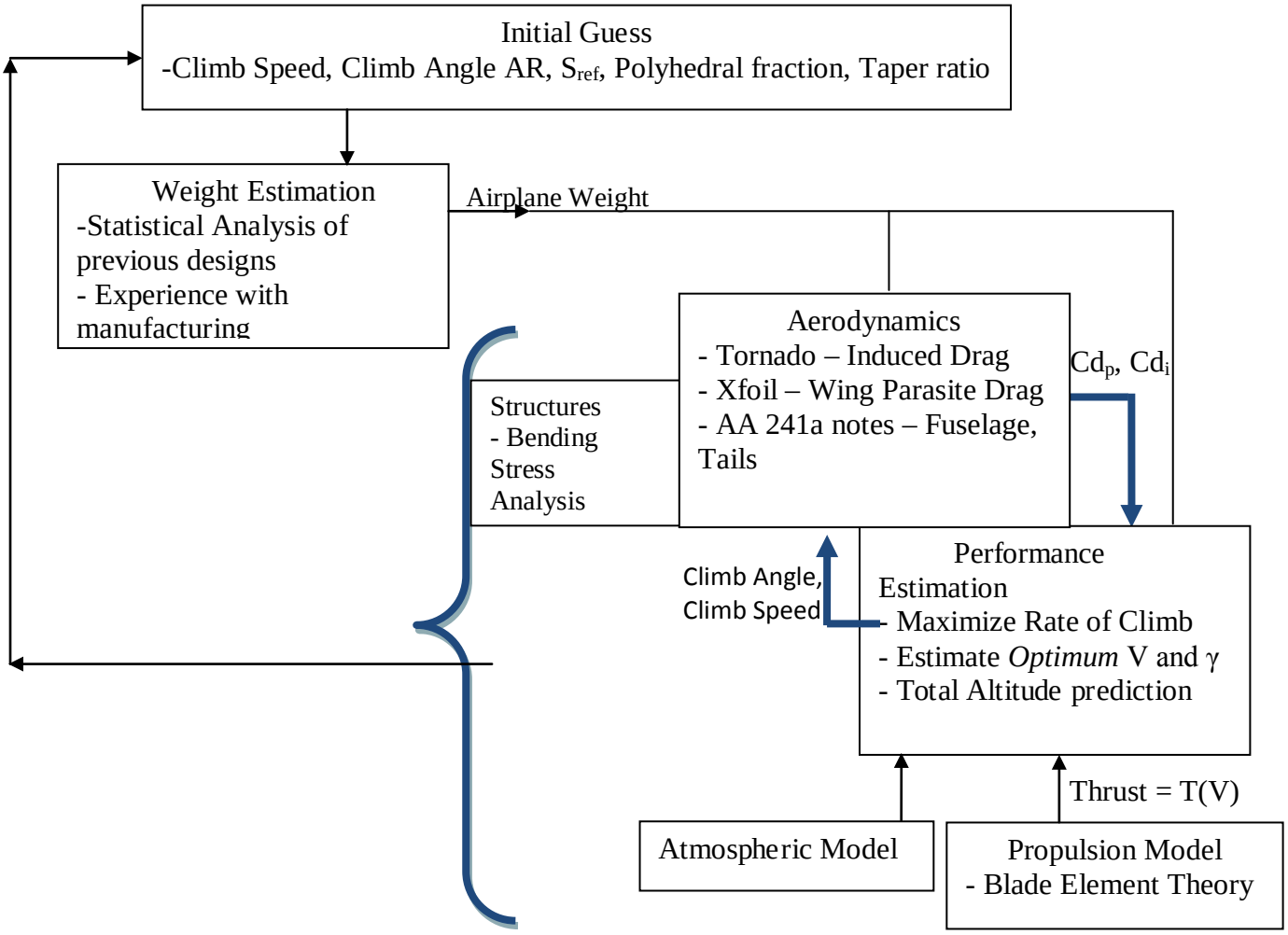


Figure 5: Optimizer

3.4 Optimization Results

Author: Amrita Mittal

The optimizer results in the following designs. [1]

[1] Note: We learnt that manufacturing a chord length of less than 10 cm was difficult only after Stephano was made. Therefore Stephano did not include the manufacturing constraint in its optimization. Stephano and Prospero were built before we included the variable polyhedral fraction in the optimizer. Therefore a constant value of 0.5 was chosen for the designs.

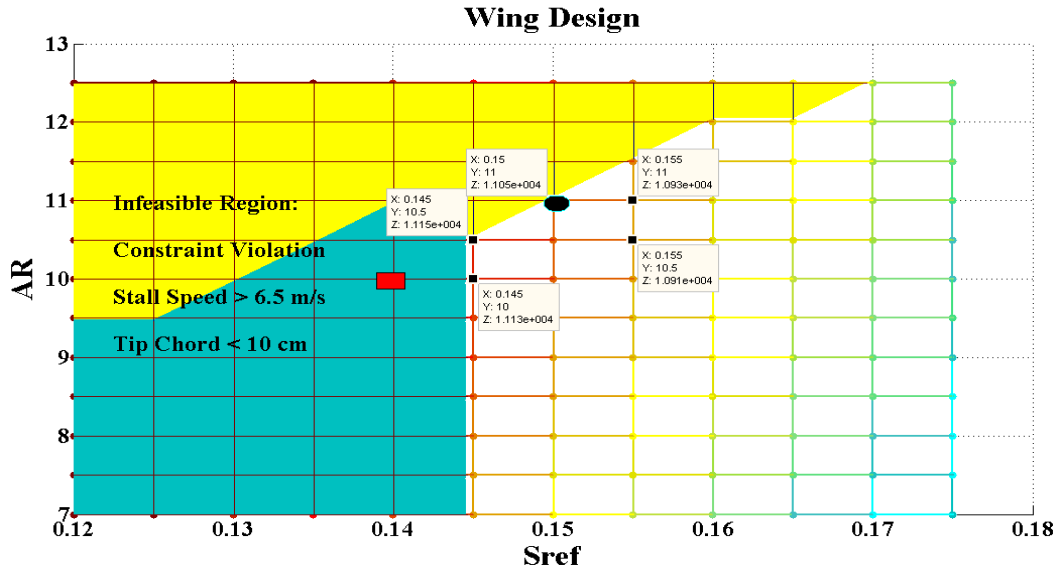


Figure 6: Optimization results

We see that altitude gain with these different designs is almost the same therefore the following dimensions were picked for Prospero.

Table 2: Design breakdown

Parameter	Stephano	Prospero
Reference Area	0.13 m^2	0.15 m^2
Aspect Ratio	10	11
Stall Speed	8.0 m/s	6.3 m/s
Twist	0	[0 -1 -3]
Polyhedral	15^o	20^o
Wing Weight	95 g	95 g
L/D	13.6	13.2
Airfoil	SD7043	SD7043
Root Chord	0.118 m	0.122 m
Tip Chord	0.094 m	0.101 m
Total Weight	**427 g**	**392 g**
Climb Speed	8.31m/s	8.3m/s
Rate Of Climb	3.8 m/s	4.1 m/s
Maximum Altitude	9600 ft	11050 ft

3.5 Stability Analysis and Tail Design

Author: Gavin MacGarva

3.5.1 Objectives

Stability analysis and design of the empennage was conducted in order to satisfy two primary objectives:

1. *Ensure the Resulting Design was Stable in All Dynamic Modes* - Given the fact that no sensors were allowed to be used to measure the orientation of the UAV in flight, the aircraft would be aware only of its location in space. The lack of such a sensor would severely limit the ability of any control law[2] to correct for disturbances. Therefore, it was essential that the UAV be stable in all dynamic modes in order for it to fly properly in autonomous mode and satisfy mission requirements.
2. *Provide Accurate Stability Derivative Information for Control Law Testing* – From analysis of reports from previous AA241X classes, it became apparent that the development of an adequate control law which functioned well in flight was a major obstacle faced in previous years and tended to be addressed only towards the end of the course. In order to better satisfy mission requirements and reduce risk to the project, Team Tempest decided to utilize simulations with six degrees of freedom. This would allow many different control laws to be tested with a high level of fidelity without risking an aircraft and analysis begin early in the quarter. In order to ensure these simulations would be in good agreement with reality, it was imperative that the stability analysis be conducted to provide accurate stability derivatives as inputs to these simulations.

3.5.2 Analysis Tools

Three primary analysis tools were considered for analyzing stability; Athena Vortex Lattice (AVL), TORNADO and LinAir, all of which are vortex lattice methods. The version of

[2] Which could be developed, tested, and be ready for service in the span of one quarter.

LinAir supplied to the class did not allow for control surface deflections to be modeled, whereas the first two methods were equipped with this feature. This feature was considered incredibly important as it allowed for the control power derivatives $C_{m,\delta e}$ and $C_{m,\delta r}$ to be determined for simulations. Thus, LinAir was removed from consideration as a primary analysis tool, leaving AVL and TORNADO. AVL was eventually selected over TORNADO and the motivation for this was the increased accuracy it offered over the latter. Although no data was found comparing these analysis tools to experimental data in the low Reynolds number regime of the design, a thesis by Melin at the Royal Institute of Technology comparing both these tools to experimental data for a Cessna 172 revealed AVL results to better match reality.[3] Furthermore, a study conducted by the U.S. Air Force in the development, construction, and testing of a micro UAV demonstrated the successful use of AVL as a design tool in the low Reynolds number flight regime.[4]

Therefore, AVL was chosen as the primary analysis tool. The results were cross-checked with LinAir by inputting the same geometry into the latter program but adjusting the incidence of the horizontal tail to allow the aircraft to be trimmed at the C_L of interest. Stability derivatives determined by both programs were in close agreement.

3.5.3 Design

3.5.3.1 Philosophy

Stability constraints were not incorporated into the optimization program (3.3: Optimizer) as it was deemed the extra time required to incorporate AVL in the MATLAB code would not be an efficient use of the effort of the group. Therefore, a wing dihedral angle would be recommended to the aerodynamics lead to ensure spiral stability, the wing shape would be optimized, and the results would be used to design a tail that would satisfy the first stability objective. Static longitudinal stability was ensured by placing the C.G. at the quarter chord of

[3] http://www.aero.polimi.it/~vige/bacheca/complementi/materiale_didattico/thesis.pdf

[4] http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA513695&Location=U2&doc=GetTRDoc.pdf
NOTE: The paper indicates that wind tunnel tests were conducted to compare analysis with AVL to experiment, however the data was not included in the report, citing that it was "not ready at the time of publication."

the aircraft as the conventional tail design would shift the neutral point aft of the aerodynamic center of the wing, resulting in a positive static margin. The magnitude of the static margin was checked with AVL and the C.G. was adjusted if it was too stiff or the margin was too low.

Finding an initial starting point for wing dihedral angle, horizontal and vertical tail volumes ($V_h and\ V_t$) proved to be a challenge. Statistical tail volume data for similar R/C aircraft was hard to come by and the same data for sailplanes and larger aircraft from texts such as Raymer were not used for two critical reasons: 1) these aircraft did not operate in the same Reynolds number regime and 2) these aircraft were not subject to the constraints of having no ailerons and being stable in the spiral mode. The team decided instead to utilize the limited data from the remains of aircraft from the previous year of AA241X to come up with an initial tail design and check stability with AVL. Flight testing would indicate whether or not the design was adequate, and the stability derivatives determined in AVL in conjunction with flight testing of the first aircraft would provide a reasonable benchmark by which the design could be adjusted in the final iteration.

3.5.3.2 Stephano

A dihedral angle of 15º for a dihedral fraction of 50% resulted in adequate spiral stability for teams from previous years; therefore it was that recommendation which was incorporated into the design of the wing for Stephano. The horizontal tail was designed such that the planform area of the surface (S_h) was of 20.0% of the wing planform area, the average horizontal tail fraction for all the first designs from aircraft from the previous year. The distance between the aerodynamic center of the wing and horizontal tail (l_h) was set to 70.0 cm. This starting point was also based on an average of previous aircraft.

The planform area for the vertical tails (S_h) from previous designs seemed to vary wildly. Concerned that a small horizontal tail would not have adequate control power to handle the torque of the propeller and make sharp turns when necessary, the team decided to set S_v at 12% of the wing planform area. From a manufacturing point of view, it was determined it would be much easier to design a tail that was too large then reduce its size if the control power was too

great rather than discovering the tail size was inadequate, manufacturing another tail and securing it to the boom. The tail design for Stephano is shown in Table 3 below.

Table 3: Stephano Tail Geometry[5]

	Horizontal Tail	Vertical Tail
Planform Area (cm^2)	286.7	167.4
Span (cm)	32.2	18.3
Aspect Ratio	3.62	2.0

AVL was then utilized to analyze the stability of the design using the inertia properties from the CAD model in all the dynamics modes. The program determined the dimensionless eigenvalues but supplies them to the used in the form: $\sigma \pm \omega_d i$

Where σ is the damping rate and ω_d is the damped natural frequency. For the lateral modes, this is related to the nondimensional eigenvalues by the characteristic time, $b_w/2V_o$ through the following equations[6]:

$$\sigma = -real(\lambda)\frac{2V_o}{b_w} \qquad \omega_d = |imag(\lambda)|\frac{2V_o}{b_w}$$

Where the 99.0 % damping time is determined by:

$$t_{99\%\ damping} = \frac{\ln(0.01)}{-\sigma}$$

The AVL results for Stephano at sea level are shown graphically in Figure 7 and numerically in Table 4. The data reflects the actual design that underwent flight testing, of which l_h and l_v was reduced to 62.8 cm from shifting the wing aft in order to compensate for the C.G. of the design being well aft of its intended location and ensure static longitudinal stability.

[5] NOTE: Problems with manufacturing resulted in some of the wing geometry being significantly different than intended (smaller chord lengths resulted in a 6.7 percent reduction in wing area). The control surfaces were redesigned to reflect this change and this altered design is shown in the table.

[6] Phillips, p. 768

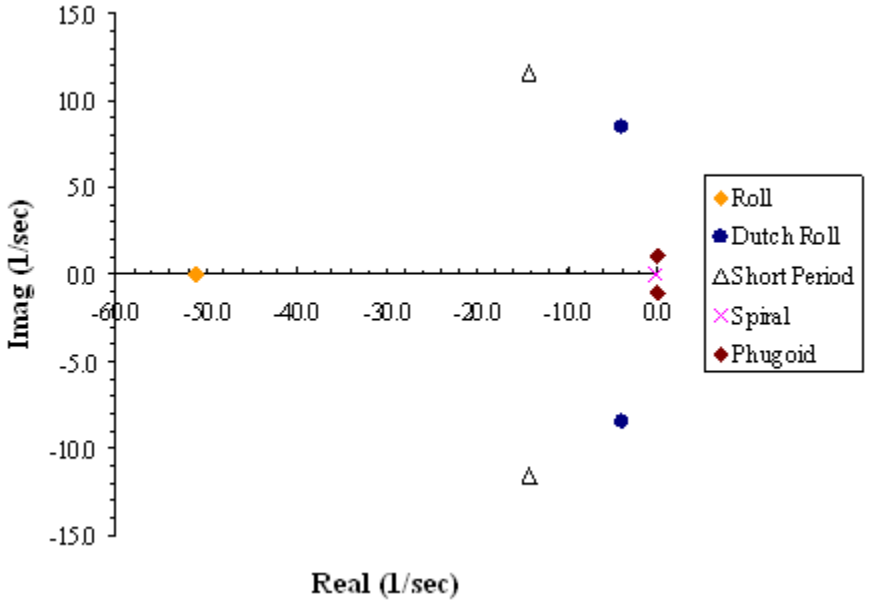


Figure 7: Trimmed Stephano dimensional eigenvalues plot at sea level

Table 4: Trimmed Stephano Eigenvalues at Sea Level

Common Name	Dimensional Eigenvalues Real (s^{-1})	Imaginary (s^{-1})	Damping Ratio	Time to 99.0% (s)
Roll	-51.0423	0.0000	1.0000	0.09
Dutch Roll	-3.9673	8.4311	0.4258	1.16
Dutch Roll	-3.9673	-8.4311	0.4258	1.16
Short Period	-14.1569	11.6188	0.7730	0.33
Short Period	-14.1569	-11.6188	0.7730	0.33
Spiral	-0.2060	0.0000	1.0000	22.35
Phugoid	-0.1008	1.0691	0.0939	45.68
Phugoid	-0.1008	-1.0691	0.0939	45.68

The results shown in Figure 7 and Table 4 showed that the design was stable in all modes and the design was put into production.

3.5.3.3 Sensitivity Analysis

Flight testing for Stephano revealed that the aircraft had very good control authority and all the modes were very well damped with the exception of the phugoid and spiral modes. Although the phugoid mode was lightly damped, this was deemed acceptable. The performance of the aircraft in the spiral mode, however, was unacceptable. An undesirable right wing tip stall would send Stephano into a "right turn death spiral" whenever the mode was activated. Poor manufacturing proved to be the cause of the problem (see section 6: FLIGHT TESTS), however the team sought to reduce the time to 99% in the next design while maintaining/accepting a limited decrease in the damping/time to 99% of the other modes. In order to determine which design changes would be best to achieve this goal, a sensitivity analysis was conducted analyzing the following cases while keeping all other design parameters constant:

1. Increase the boom length (effectively l_h and l_v) to 70.0 cm
2. Increase the dihedral angle from 15º to 20º
3. Increase S_v by 10 percent

Focus was placed on the Dutch roll and spiral modes as design changes increasing the stability of one were expected to decrease the stability of the other (increasing the dihedral angle, for example, tends to increase spiral stability but destabilize the Dutch roll mode). The results of the trade study for the Dutch roll and Spiral modes are shown in Table 5 and Table 6, respectively. Improvements to the baseline configuration are highlighted in green and adverse changes are highlighted in red.

The results of the study were as expected. Increasing the dihedral angle benefited the spiral mode but had an adverse effect on the Dutch roll mode, while increasing the size of the vertical tail benefited the Dutch roll mode but had an adverse effect on the spiral mode. More importantly, however, the study provided insight into the effectiveness of each design change on the two dynamic modes. Increasing the dihedral angle by 5º provided tremendous improvement in the spiral mode, nearly cutting the time to 99% by half, while the negative effect on the Dutch roll mode was marginal. Additionally, increasing the boom length or S_v provided significant improvements to the Dutch Roll mode (18.1 and 20.7% reduction in time to 99% respectively)

and the latter design change comparatively had a very small negative effect on the spiral mode (an increase by 4.3% in the time to 99%). From this part of the analysis it was determined that there was little penalty to having a slightly larger tail; the adverse effect it would have on the spiral mode is very small, and given the size and density of balsa used in construction, the increase in weight would be insignificant. Furthermore, the study also analyzed the effect of increasing the dihedral angle to 20º to increase spiral stability but lengthening the boom such that the damping of the Dutch roll mode would be identical to Stephano. In this analysis the same horizontal and vertical tail volumes (V_h and V_v) as Stephano were utilized. The net change of the boom-empennage system to achieve this end resulting from the increased weight of a longer boom but decreased weight of a smaller tail was only 1 gram out of an approximately 400 gram aircraft. Thus, changes in the design to achieve these more clearly defined stability criteria in the next iteration would have little impact on the weight of the design and therefore mission requirement to achieve the highest altitude possible.

Table 5: Effects of design changes analyzed on Dutch roll

Design Change	Dimensional Eigenvalue Real (s^{-1})	Imaginary (s-)	Damping Ratio	Time to 99 % (s)
Baseline	-3.9673	8.4311	0.4258	1.16
	-3.9673	-8.4311	0.4258	1.16
Increased Boom Length	-4.9999	8.5868	0.5032	0.92
	-4.9999	-8.5868	0.5032	0.92
20º Dihedral	-3.9311	8.4865	0.4203	1.17
	-3.9311	-8.4865	0.4203	1.17
Increased S_v	-4.8259	9.0906	0.4689	0.95
	-4.8259	-9.0906	0.4689	0.95

Table 6: Effects of design changes analyzed on spiral mode

Design Change	Dimensional Eigenvalue Real (s^{-1})	Imaginary (s^{-1})	Time to 99% (sec)
Baseline	-0.2060	0.0000	22.3
Increased Boom Length	-0.2679	0.0000	17.19
20º Dihedral	-0.3375	0.0000	13.64
Increased S_v	-0.1979	0.0000	23.27

3.5.3.4 Prospero

With the results of flight testing and the sensitivity analysis the aerodynamics lead optimized a wing with a 20º dihedral angle. From this design and aircraft weight estimate, the empennage for Prospero was designed with the goals highlighted in section 3.5.1. The same horizontal and vertical tail volumes as Stephano were utilized while l_h and l_v were increased to 70.0 cm in order make up for the reduction in damping of the Dutch roll mode due to the increased wing dihedral angle. The horizontal and vertical tails for Prospero were designed with the resulting planform areas that would satisfy these conditions, while maintaining same aspect ratio of the corresponding surface on Stephano. The geometry of the empennage for Prospero is shown in Table 7 below.

Table 7: Prospero Tail Geometry

	Horizontal Tail	Vertical Tail
Planform Area (cm^2)	249.8	136.8
Span (cm)	30.1	18.3
Aspect Ratio	3.62	2.0

The AVL results for Prospero at sea level are shown in Table 8 below. Improvements over Stephano are highlighted in green and adverse changes are highlighted in red.

Table 8: Trimmed Prospero Eigenvalues at Sea Level

Mode	Dimensional Eigenvalue		Damping Ratio	Time to 99% (s)
	Real (s^{-1})	Imaginary (s^{-1})		
Roll	-60.0140	0.0000	1.0000	0.1
Dutch Roll	-3.8388	7.4171	0.4597	1.2
Dutch Roll	-3.8388	-7.4171	0.4597	1.2
Short Period	-15.4208	10.2239	0.8335	0.3
Short Period	-15.4208	-10.2239	0.8335	0.3
Spiral	-0.4227	0.0000	1.0000	10.9
Phugoid	-0.1104	1.2128	0.0906	41.7
Phugoid	-0.1104	-1.2128	0.0906	41.7

The increase in the dihedral of the wing to 20º and the design of the empennage allowed Prospero to meet handling quality goals. The time to 99% for the spiral mode improved significantly, decreasing from 22.3 seconds to 10.9 seconds with minor adverse effects in only one mode. To meet the first stability objective, however, Prospero needed to be stable at all altitudes of the flight regime. Eigenmode analysis was conducted above the theoretical maximum altitude to ensure this objective was satisfied. The results are shown in Figure 8 and Table 9. The solid marks represent dimensional eigenvalues at sea level and the hollow marks represent dimensional eigenvalues at 10,000 ft. Note that the roll mode was left out of the plot as it is very stable in the far left plane in order to provide a better presentation of the modes closer to the right half plane.

As expected the modes were less stable at 10,000 ft than they were at sea level, shifting towards the right half plane. However all modes remained in the left half plane, indicating that the UAV would fulfill the first stability objective at all altitudes of the flight regime.

Elevator and rudder control power stability derivatives were also analyzed to ensure that they were not greater than that of Stephano. Flight testing of Stephano indicated that very small deflections were needed to control the aircraft. Out of 250 positions for the rudder servo only 3 “clicks” left or right were sufficient to turn the aircraft under autonomous control. In order to increase the resolution of the control law (to allow for more “feasible clicks” without over-banking the aircraft), Team Tempest wanted to ensure that the control power derivatives for

Prospero did not exceed that of Stephano. These derivatives were analyzed in AVL and the results are shown in Table 9 below.

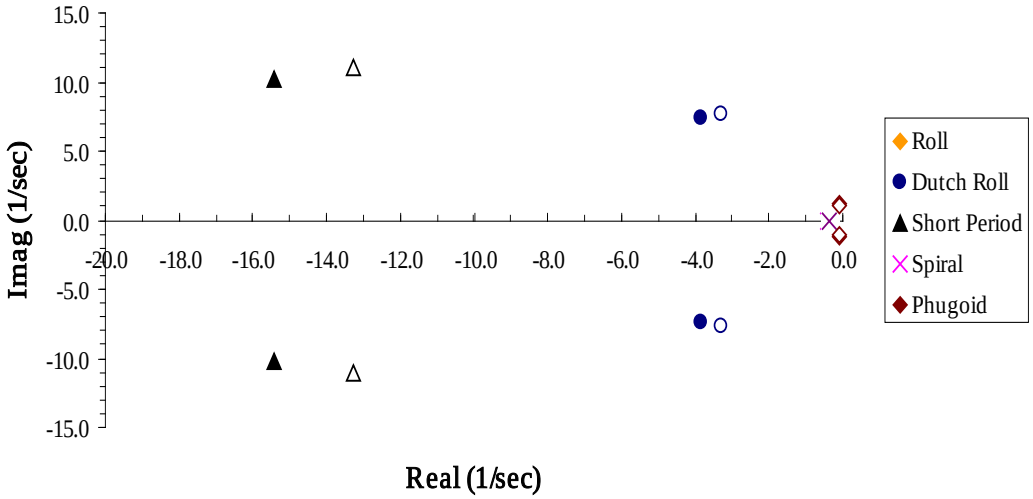


Figure 8: Trimmed Prospero dimensional eigenvalues plot at sea level and 10,000 ft MSL

Table 9: Comparison of Control Power Derivatives for Stephano and Prospero

Control Power Derivative	Stephano	Prospero	Percent Change
$C_{m,\delta e}$ **(per degree)**	-0.0339	-0.0282	-16.8
$C_{m,\delta r}$ **(per degree)**	-0.0021	-0.0016	-24.9

The empennage design met all the objectives and improvements determined to be necessary from the first iteration and Team Tempest decided to put the design into production with Prospero.

4 CONSTRUCTION AND CAD MODEL

4.1 Construction Methods

4.1.1 Fuselage

4.1.1.1 Stephano Fuselage

Author: Kevin Reynolds

The first fuselage design envisioned by the team would be one able to withstand the uncertainties expected of our initial flight tests. With very little known about the actual performance characteristics of the wing due to manufacturing imperfections, the aircraft test bed would need to be designed for crashworthiness in the event of wing tip stall or structural failure due to aerodynamic loads. The following items were desired characteristics of the first design named Stephano:

- Sustain minimal crash damage in the event of a dive from 152 m (500 ft)
- Allow for retrimming of the airplane without cutting new holes
- Enable the rapid repair of fuselage skins and frame in order to minimize down-time after a crash

The fuselage frame for Stephano was constructed out of 5 mm thick balsa wood in order to provide extra strength in bending in the event of a hard collision with the ground. Rectangular-shaped compartments were independently sized to house the motor, the batteries, and electronics. With the design being a tractor propeller design, the motor compartment was naturally positioned at the very front of the fuselage, allowing for the propeller assembly to extend out the front and clear the fuselage by at least 0.25 inch. A 2 mm thick bulkhead was used for mounting the motor from the front of the aircraft rather from the inside direction. This method of construction was chosen in order to have the bulkhead absorb the majority of the loads in the event of a crash rather than the frame itself. A seal of epoxy was used to strengthen the bulkhead to the fuselage. This seal ended up contributing several grams to the overall weight and was avoided in later designs.

The skins of the fuselage were constructed out of cured fiber glass sheets prepared in the same way as the fiber glass sheets used for wing layups. With excess epoxy squeezed out during the manufacturing process, the weight was slightly less than a 2 mm thick sheet of balsa wood of the same dimensions. From observation and testing, the carbon fiber skins offered an advantage in their durability and strength by being able to withstanding higher shear and torsional loads. Compared with the results from other team's aircraft, the skins were less susceptible to being punctured by tall brush during landing.

In one of our earlier flights, team Tempest experienced an instance where the pilot lost control of the airplane at 100-150 feet above the ground. It was later discovered that the autopilot battery had reached its low threshold value around 7.3 V and had lost the ability to be controlled via RC commands. During the crash, Stephano dove nose down into a tall pile of grass on the drier end of Lake Lagunita. Thanks in part to the robust construction of Stephano and in other part to pure fortunate, minimal structural damage was felt by the fuselage in the form of a cracked frame surrounding the motor mount compartment. The bulkhead had become dislodged from its prior fixed position, indicating it had been a critical load path during the crash. The frame was repaired that evening and Stephano flew again the following day.

4.1.1.2 Prospero Fuselage

Author: Amrita Mittal

We were flying Stephano at Lake Lagunita which has a cushiony bed of bushes, ideal for landing. Therefore we decided to go with a very aggressive design for the second fuselage. Cross sectional area of all the stringers was reduced to 75%. We got rid of all the cross bars from our first design which not only made the fuselage lighter but also easier to access. Monocot was used on the skin instead of fiberglass to reduce weight and provide better visibility. The second design was more streamlined compared to the first.

Other design considerations were mainly lessons learnt from flying Stephano. Stephano suffered from boom misalignment which made it very difficult to control. Therefore, after talking to Kyle Washabaugh about his experience from previous year, a jig was made using bulk

heads to ensure that the boom alignment is straight. Stephano also suffered from shift in CG location in flight therefore different compartments were made in the second design to ensure that the batteries did not move in flight. The GPS was nicely fitted against the side wall with two pieces of balsa which meant we didn't have to move it while changing batteries. The left side of the fuselage was taped to ensure easy access.

Author: Gavin MacGarva

A considerably greater amount of focus was placed on manufacturing quality for the second fuselage. To ensure the tail was not crocked with respect to the motor and the wing, holes the diameter of the boom were drilled with a drill press through the 3/16th inch liteply bulkheads with a drill press. These bulkheads were all threaded onto the boom and the 3/8th inch balsa longerons were bent around the bulkheads and secured with CA glue. Once the glue dried the fuselage was removed from the boom and reattached to the aft two bulkheads with a thin layer of epoxy.

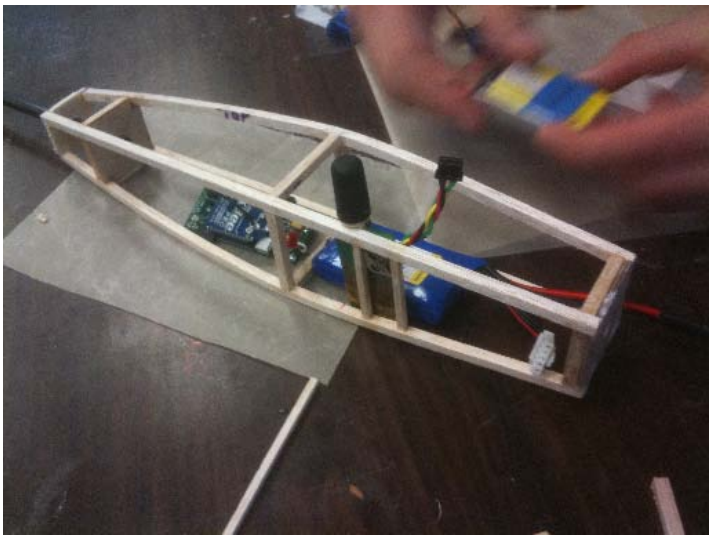


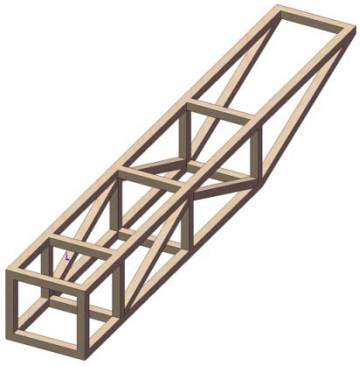
Figure 9: Fuselage structure with boom attached

Finally the monocot skin for the fuselage was attached. A small section was left open for access to components during flight testing and was closed with tape before each flight.

Figure 10: Completed Prospero Fuselage in the field

Listed below is a summary of the fuselage designs that were constructed and the tradeoffs made for each.

Author: Kevin Reynolds



	Stephano	Prospero
Key Features	Wgt: 54 g Fineness: 6.48 Wing AOA: 3 Skin: Fiber glass -Truss-like design -Lightweight skin -Platform for control law testing	Wgt: 37 g Fineness: 6.67 Wing AOA: 3 Skin: Monocot -Lighter weight frame -Easy access to internals -Stream-lined shape -Straight alignment of tail
Possible Improvements	-Smaller balsa frame -More conservative use of epoxy -More aerodynamic shape	-Lighter weight frame

FUSELAGE CRASHWORTHINESS TRADES		
What's Designed to Fail	**Pros**	**Cons**
Frame + Skins	Availability of materials (e.g. wood, glue, tape) Multiple load paths through structure On-site repair of skins No need to email TA!!	Geometric misalignment of structure/wing Exposure of electronics to the elements (e.g. dirt, water, wind, etc) Longer wait-time for repair Need to retrim the airplane
Motor Mount + Shaft	Localized damage to structure Greater strength of materials (e.g. metal) Potentially simple and easy to repair Reduction of weight in the fuselage	Bent/worn motor shaft Cracked motor mount and repair Need to retap screw holes Tools for tightening (e.g. wrench, screwdriver)
Wing + Wing Mount	Nylon bolts that break/shear	Snapped wing, misalignment of wing Having to remake/remount wing Robustness/weight of wing mount Need to retrim the airplane

4.1.2 Wing

Author: Gavin MacGarva

The wings for both aircraft were fabricated out of a foam core to act as a shear web and hold the airfoil shape. The foam cores were cut using the hotwire machine run by a computer in the basement of Durand which allowed the desired shape and twist to be cut at a much higher level of precision than if done by hand. Slots were sanded into the foam at the quarter chord to hold carbon fiber spar caps which were glued with a thin layer of epoxy to carry bending loads. Finally, fiberglass skins were vacuum bagged to surface of the wing. An extra layer of fiberglass consisting of a thin strip was also added to the leading edge during the layup to increase protection from denting in the event of a crash or encounter with thick brush upon landing. Construction of the wings for each aircraft followed this general procedure with some variations.

4.1.2.1 Stephano Wing

Two identical wings were manufactured for Stephano. Both were constructed with the blue spyder foam, included carbon fiber spar caps on the top surface which spanned the entire wing, and were vacuumed bag as three separate sections: the constant chord center sections and two outboard sections. These sections were joined together at the appropriate angle by sanding the dihedral angle into the outboard sections, gluing the exposed foam surfaces together with epoxy, then wrapping a layer of fiberglass around the joint. Unfortunately, this was done hastily and significant gaps were left between the surfaces of the two foam cores when they were joined together. Concern over the strength of this joint led to the decision to add another layer of fiberglass around each joint which made the wings heavier than desired.

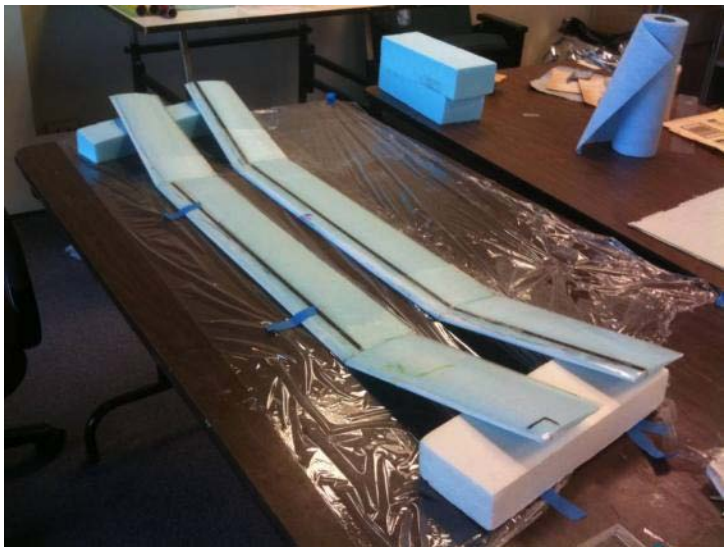


Figure 11: Joining the inboard and outboard sections

The wing performed well structurally, however it was a lousy performer aerodynamically. Because the sections were joined hastily, the right wing was unintentionally mounted at a positive incidence angle with respect to the center section which caused tip stall in flight. A look at the second wing after the tip stall was discovered in testing revealed it was mounted at an angle even worse than the first wing. The second wing, therefore, was never used.

4.1.2.2 Prospero Wing

Learning from the lessons of Stephano, the wing for Prospero was manufactured with much greater attention to manufacturing quality. The wing was cut in four separate sections as the inboard sections included washout. However, the center section was fiberglassed and vacuum bagged as one continuous section with carbon fiber spar caps on the top and bottom to act as an I-beam, and increase the strength of the wing at the root where bending loads would be the highest. The center sections were again cut out of the blue spyder foam, but the outer sections were cut out of the 25 percent lighter, pink dow insulation foam in order to reduce weight.

Figure 12: Joining the outboard sections

These sections were joined to the center section by sanding a half dihedral angle in the center and outboard sections, and using a elbow cut out of liteply at the appropriate angle to join the two sections together at the quarter chord. After these sections were secured together a thin band of fiberglass was placed around each joint.

Figure 13: Prospero

4.1.3 Empennage

Author: Gavin MacGarva

The empennage was cut out of ¼ inch thick balsa sheets. While it was desired to reduce weight by using thinner sheets, these proved not to be very stiff. It was determined that they would likely flex inconsistently in flight resulting in inconsistent performance of the aircraft in autonomous mode as the same trim setting would produce a different amount of lift or down force on the control surface. ¼ inch thick balsa was the smallest thickness available that did not appear to flex and was thus why it was used in the design. Control surfaces were cut out of the same material and secured to the horizontal and vertical tail with tape to form a hinge.

The empennage was then secured to a balsa block with CA glue which was grooved to fit to the boom and then secured to the boom with epoxy.

Figure 14: Control surface hinge (left) and attachment to boom (right)

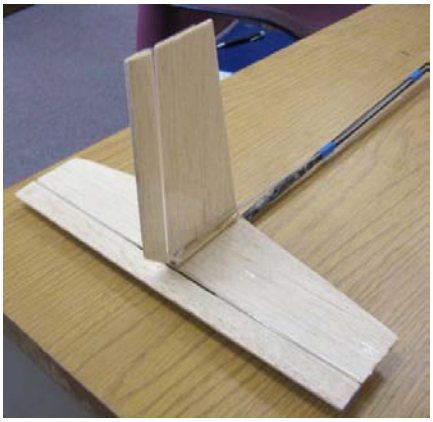


Figure 15: Completed empennage (Prospero)

4.2 CAD Model

Author: Kevin Reynolds

CAD model of Prospero is as shown below in Figure 16 and Figure 17.

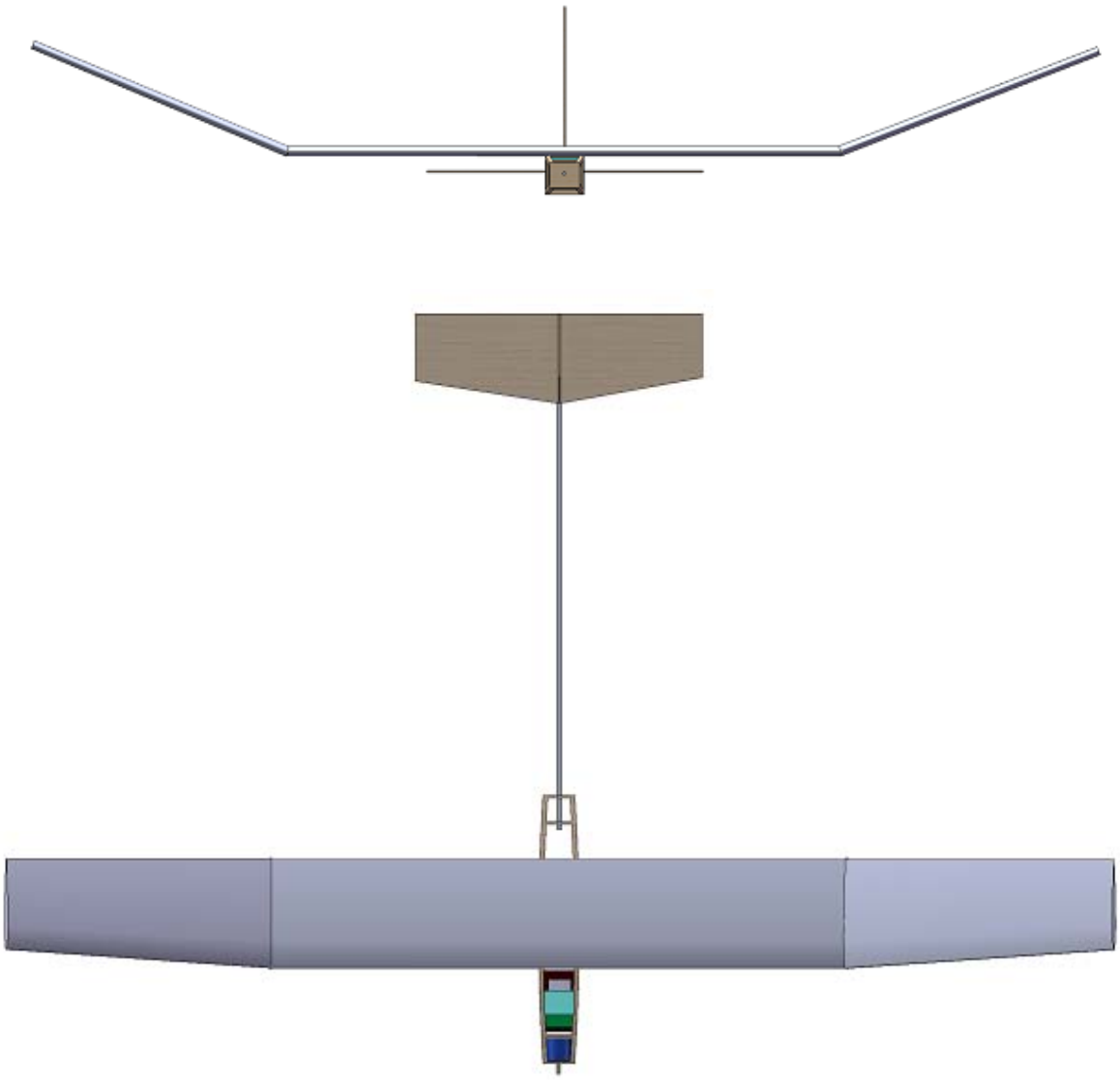


Figure 16: Front and top view of Prospero

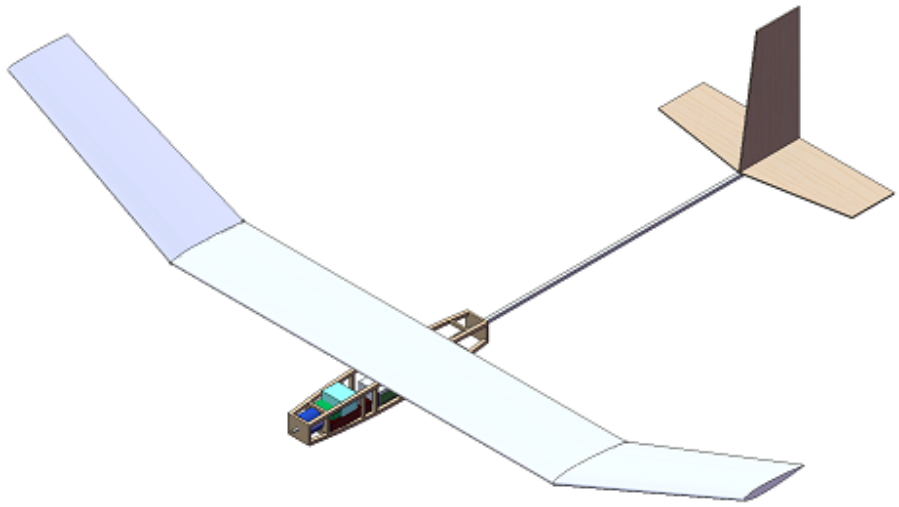


Figure 17: Isometric view of Prospero

4.3 Weight Breakdown

Author: Gavin MacGarva

Table 10 has the weight of required electronics items.

Table 10: Weight of required items

Component	Weight
HiMaxx 2808-0980 Motor w/mount	52 g
Thunderbird 09 ESC w/ connectors	14 g
9” x 5” Aeronaut CAM propeller w/BB Aluminum Spinner	24 g
Hitech HS-55 Servos (2)	8.5 g each (17 g total)
Hyperion 1100 mAH 2S (7.4V) Propulsion Battery	63 g
Autopilot Battery	29 g
Autopilot Board	12 g
GPS Board	21 g
xBeePro Radio	
Total Required Items	232 g

Table 11 compares the weight of all parts with predicted values. Even though Prospero's wing had a much larger surface area than that of Stephano, we managed to get the same weight by not using too much epoxy and by using Dow foam for the polyhedral section of the wing. Dow foam tips saved us 6-7 grams. We also saved some weight by making the second fuselage lighter.

Table 11: Weight breakdown

Component	Stephano (g) (estimated)	Prospero (g) (estimated)
Wing	95 (95)	95 (98)
Fuselage and Boom	64 (32)	50 (30+19)
Tail	17 (19)	15 (12)
Markup (used in estimates only)	- (9)	- (5)
Total	**408 (400)**	**392 (386)**

5 CONTROL LAW AND SIMULATIONS

Mission of this autonomous UAV is to climb to and descent from certain altitudes, repetitively. It is also required to stay inside a specified region, to take-off and to land autonomously. In order to do that we need to decide the states of state machine, design longitudinal and lateral control laws.

The steps followed in designing the control law are as follows:

1. State machine
2. Control law

5.1 State Machine

Author: Harsh Menon

The state machine for the airplane consists of six states – takeoff, climb, transition to descent, descent, transition to climb and land. The aircraft begins its flight in the takeoff state and then upon reaching an altitude of 40 m goes into the climb state. The airplane remains in the climb state till it reaches an altitude of 175 m after which it enters the transition to descent state. The transition to descent state is a 3 second state where the throttle is linearly decreased from full throttle (250) to no throttle (0). The aircraft then remains in the descent state till it reaches an altitude of 85 m after which it enters the transition to climb. This is a 6 second state where the throttle is linearly increased from 0 to 250. The airplane continues repeating the above states till a maximum number of cycles have been reached after which it enters the land state. In the land state, the airplane descends to the ground using the same descent control law except with an elevator setting for minimum sink rate. At about 25m, the airplane sets its rudder to the descent neutral trim and descends until it hits the ground. The state machine is visualized below (Figure 18) where ‘h’ represents the altitude and ‘t’ represents the state time.

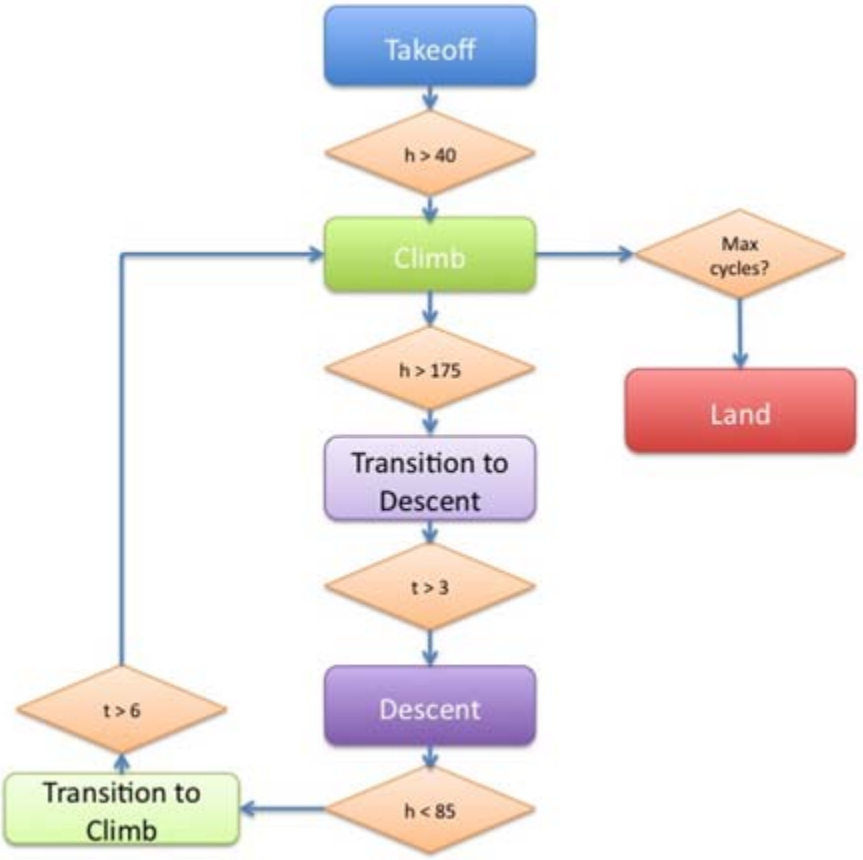


Figure 18: State machine

5.2 Selection of Control Methodology

Author: Harsh Menon

Prior to running any simulations, we decided to investigate what type of control methodology to use based on the linearized decoupled lateral and longitudinal equations of motion of the aircraft. Based on the linearized lateral equations of motion[7], and using the stability derivatives as well as the moments of inertia, we obtain the following transfer function between the rudder deflection and change in heading angle

[7] Blakelock, J. H. Automatic Control of Aircraft and Missiles: Second Edition. New York: John Wiley & Sons, Inc., 1991.

$$\frac{\theta(s)}{\delta_r(s)} = \frac{\begin{vmatrix} \frac{I_{xx}}{Sqb}s^2 - \frac{b}{2V_\infty}C_{l_p}s & C_{l_{\delta r}} & -C_{l_\beta} \\ -\frac{b}{2V_\infty}C_{n_p}s & C_{n_{\delta r}} & -C_{n_\beta} \\ C_L & C_{y_{\delta r}} & \frac{mV_\infty}{Sq}s - C_{y_\beta} \end{vmatrix}}{\begin{vmatrix} \frac{I_{xx}}{Sqb}s^2 - \frac{b}{2V_\infty}C_{l_p}s & -\frac{b}{2V_\infty}C_{l_r}s & -C_{l_\beta} \\ -\frac{b}{2V_\infty}C_{n_p}s & \frac{I_{zz}}{Sqb}s^2 - \frac{b}{2V_\infty}C_{n_r}s & -C_{n_\beta} \\ C_L & \frac{mV_\infty}{Sq} & \frac{mV_\infty}{Sq}s - C_{y_\beta} \end{vmatrix}}$$

Similarly, using the decoupled longitudinal equations of motion we can obtain the transfer function between the pitch angle and the elevator deflection. However, since we only obtain the groundspeed from the GPS, we cannot close the loop on the longitudinal dynamics transfer function. We obtain the heading angle from the GPS and hence we can use feedback control on the lateral dynamics of the airplane. Having obtained the transfer function, we then model the servo as a first-order actuator with a time constant of 0.1 seconds and add a zero-order hold (ZOH) with a sample time of 0.25 seconds. A representative block diagram is shown below (Figure 19) where K(s) is the controller, ZOH is the zero-order hold and G(s) is the plant consisting of the laterally dynamics and the servo model.

Figure 19: Lateral dynamics block diagram

Having established the block diagram, the system was analyzed using SISOTOOL in MATLAB. We first listed some of the important properties that we were concerned about and translated our desired properties to control response characteristics that we could tune the system

to achieve. Figure 20 shows the desired properties and their equivalents in terms of response characteristics.

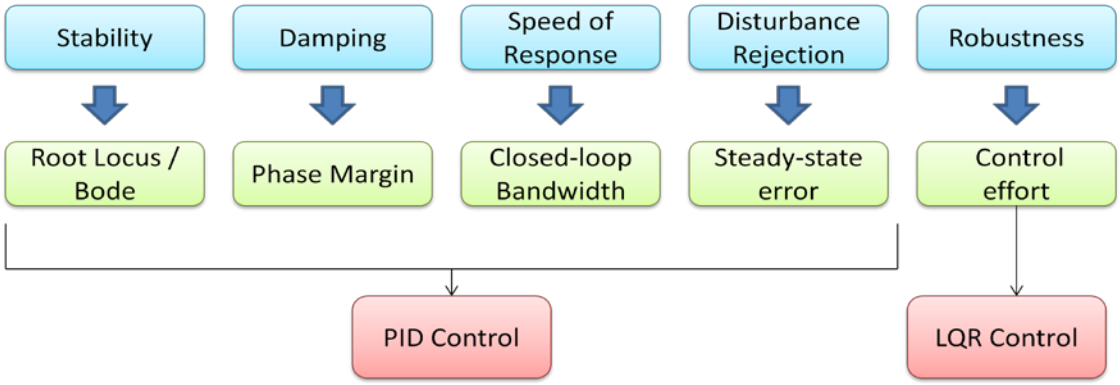


Figure 20: Desired control characteristics

The desired properties were stability, good damping, quick speed of response, good disturbance rejection and finally robustness. All but the last could be addressed using PID control and hence we decided to pursue PID control for the aircraft. We decided to incorporate issues about control effort into our control law but introducing dead band regions. Instead of establishing specifications such as a certain damping ratio or a certain closed loop bandwidth, we decided that it would make more sense to get as high a phase margin and closed loop bandwidth as possible and as low a steady state error as possible. We then tried tuning the system trying P, PI and PID control by keeping the bandwidth the same. The results are tabulated below Table 12.

Table 12: Comparison of control characteristics

Control Parameter	P Control	PI Control	PID Control
Phase Margin	30 deg	54 deg	80 deg
Steady-State Error	1%	0.1%	0.1%
Closed-loop Bandwidth	1.3 rad/s	1.3 rad/s	1.3 rad/s
Overshoot	8%	15%	7.5%
Settling Time	4 s	4.5 s	3.3 s

Having considered the above alternatives, proportional control seemed to be the most enticing due to the simplicity in implementing and due to the fact that PI and PID control did not

produce significantly greater control response characteristics than proportional control. However, we did realize at that point that we had not analyzed PD control. Thus, we tried to control the system using PD control. Using PD control it was hard to get the closed loop bandwidth to 1.3 rad/s, (we obtained a closed loop bandwidth of 0.13 rad/s). But at the given bandwidth, we were able to obtain a good phase margin (50 deg) and overshoot (10%). Thus, we were confident that using PD control would give us a quick response and good damping characteristics. Later on during flight testing, we realized that steady-state error was not a primary concern and thus when we were unable to keep the airplane within the desired region using proportional control, we tried PD control instead of PI control. As will be discussed in the later sections, adding the derivative term made the airplane return to the circle and greatly enhanced the effectiveness of the control law.

5.3 Control Law

Author: Kuldeep Lonkar

The aircraft is required to climb and descent inside a specified region, a robust lateral control law is required to bring the aircraft back safely if it ventures outside. To design a control law, first we have to formulate the error function and then determine what kind of control has to be used. While all combinations of proportional, derivative and integral controls were considered, we decided to use PD control. The steps followed in designing lateral control law are as follows:

1. Circular Region

To make the control law simple, it was decided to always have the aircraft in a gentle left turn while climbing and descending, while remaining inside a specified circular region (Figure 21). Error was calculated based on the difference between current heading angle and desired heading angle.

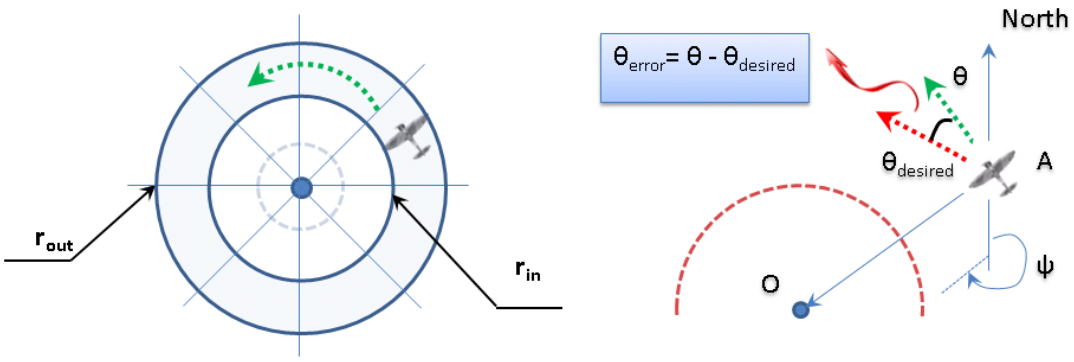


Figure 21: Circular region

O	:	center of the circles
r_{out}	:	outer radius of the circular region
r_{in}	:	inner radius of the circular region
A	:	current aircraft position
θ	:	current heading angle
r	:	current distance of the aircraft from the center
ψ	:	azimuth (angle that line segment OA makes with North, positive clockwise)
r_m	:	$(r_{out} + r_{in}) / 2$
$\theta_{desired}$	:	desired heading angle

2. Error in heading angle

There are two things that contribute to the total error.

i) The desired heading angle if the aircraft is inside the circular region is $(\psi + \frac{\pi}{2})$. Hence one part of total error is $\theta - (\psi + \frac{\pi}{2})$.

ii) And the other part is associated with the distance of the aircraft from the center.

If the aircraft is far outside, then we want it to head back towards the center i.e. desired heading angle is ψ and as it gets closer, it should align itself tangentially to the circular regions i.e. desired heading angle is $\psi + \frac{\pi}{2}$.

Similarly, if the aircraft is at the center, then we want it to head radially outwards i.e. desired heading angle is $\psi + \pi$ and as it goes out, it should align itself tangentially to the circular regions i.e. desired heading angle is $\psi + \frac{\pi}{2}$.

This means that we have to assign certain weight to $\frac{\pi}{2}$ in the expression of error.

To make the error calculations simple, we used a simple 1/r type function and normalized it so that its value is 1 at the circular region and zero at infinity and close to zero at the center.

Expressions for error in heading angle are given by:

$$\boldsymbol{\epsilon(r,\theta) = \theta - \left(\psi + \pi - \frac{\pi}{2} \times \frac{r_m - r_{in}}{r_m - r}\right)} \quad \ldots \quad \boldsymbol{if\ r < r_m}$$

$$\boldsymbol{\epsilon(r,\theta) = \theta - \left(\psi + \frac{\pi}{2}\right)} \quad \ldots \quad \boldsymbol{if\ r_{in} < r < r_{out}}$$

$$\boldsymbol{\epsilon(r,\theta) = \theta - \left(\psi + \frac{\pi}{2} \times \frac{r_{out} - r_m}{r - r_m}\right)} \quad \ldots \quad \boldsymbol{if\ r > r_m}$$

Terms in parentheses are desired heading angle depending on the distance of the aircraft from center O.

Once we have the error in heading angle, we need the rudder deflection to make the error zero. General expression for PID control is:

$$\delta r = K_P\, \epsilon(r,\theta) + K_D\, \frac{d\epsilon(r,\theta)}{dt} + \int K_I\, \epsilon(r,\theta)\ dt$$

Discretizing it for implementation in simulations and autopilot:

$$\boldsymbol{\delta r(n+1) = K_P\, \epsilon(n+1) + K_D\, \frac{\epsilon(n+1) - \epsilon(n)}{\Delta T} + \delta r^I(n+1)}$$

Where, $\delta r^I(n+1) = \delta r^I(n) + \Delta T \times K_I\, \epsilon(n+1)$: δr^I is the contribution of integral control term

$\delta r(n+1)$: δr required at time step $(n+1)$

ΔT : sampling time

5.4 Control Law Simulations using Simulink

Author: Kuldeep Lonkar

The aircraft is required to climb and descent inside a specified region, a robust lateral control law is required to bring the aircraft back

To test the performance of our control laws, a 6 DOF model was modified and implemented using AeroSim blockset from Unmanned Dynamics.

5.4.1 Propulsion Model

Propulsion model for propeller engines was modified (Figure 22) for HiMaxx motor and Aeronaut CAM propeller. MATLAB code provided to us was used to create loop-up tables for torque, current, RPM, as well as advance ratio, thrust coefficient and power coefficient.

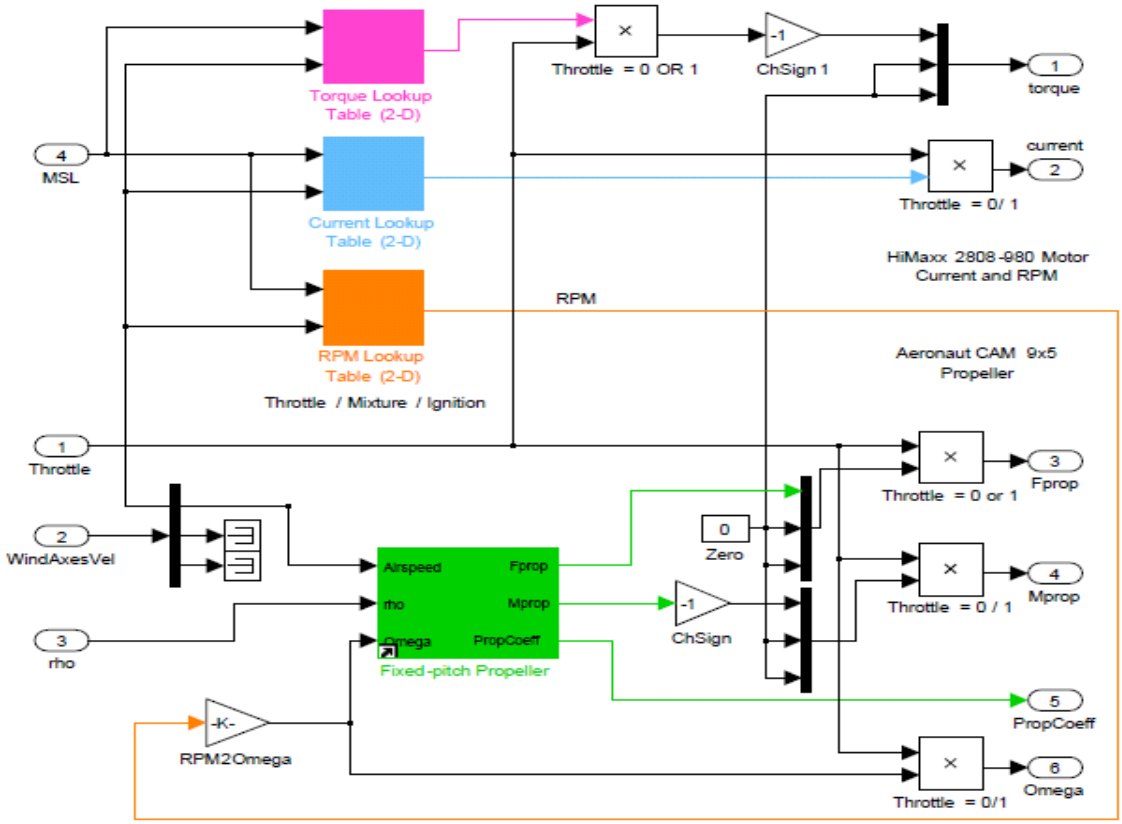


Figure 22: Propulsion block

The complete Simulink model is shown below (Figure 23):

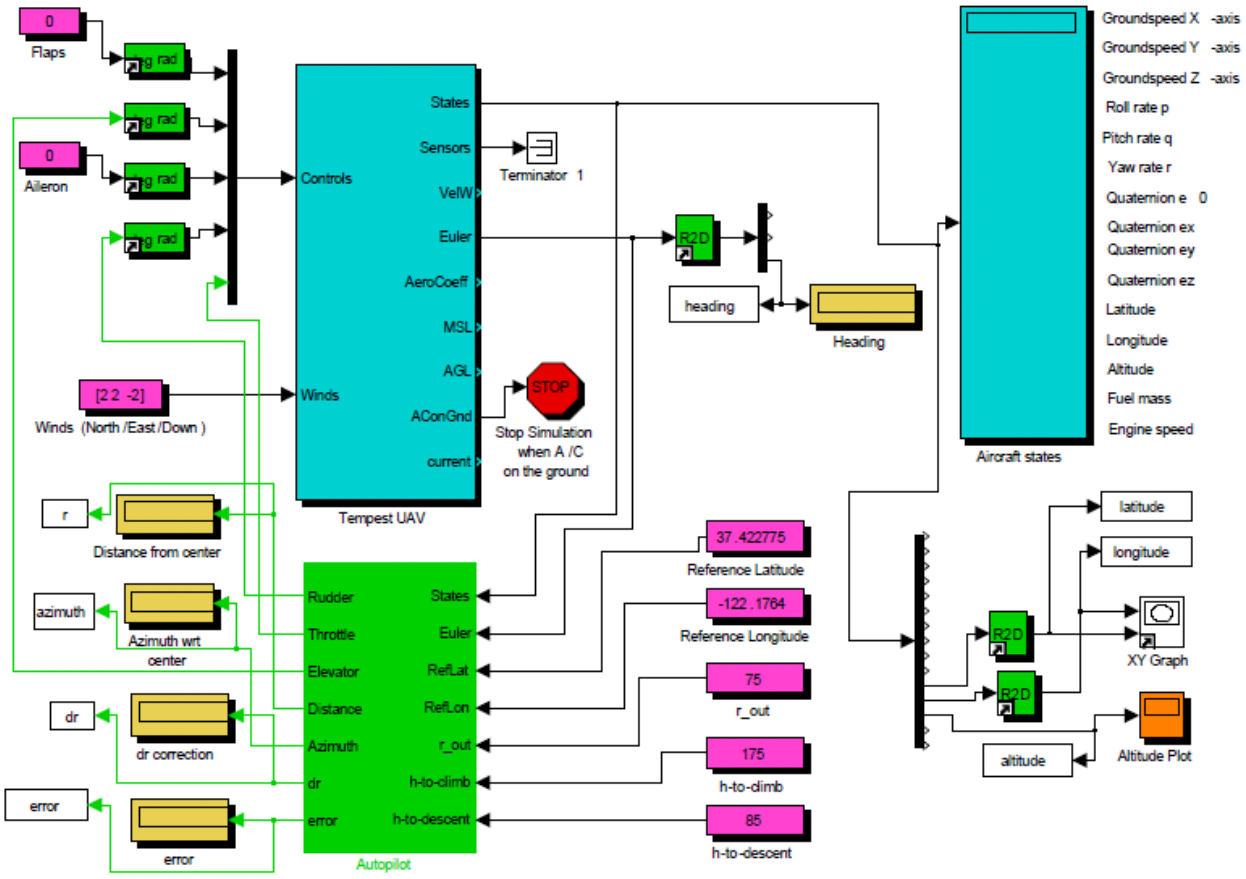


Figure 23: Complete Simulink model

Tempest UAV block has subsystems to get aerodynamic forces & moments, propulsion forces & moments, atmosphere block and a 6 DOF solver block.

5.4.2 Implementation of Control Law in Simulink

Autopilot subsystem (block) was created to implement control laws. Internal structure of this block is shown in the following figure. A MATLAB embedded function ‘get_err’ gives us the error for given set of parameters and current aircraft position, while ‘get_thrtl’ gives us the

state and throttle of the aircraft depending on the current altitude. Once we have the error, proportional gain, discrete derivative and integral blocks were used to implement PID control (Figure 24).

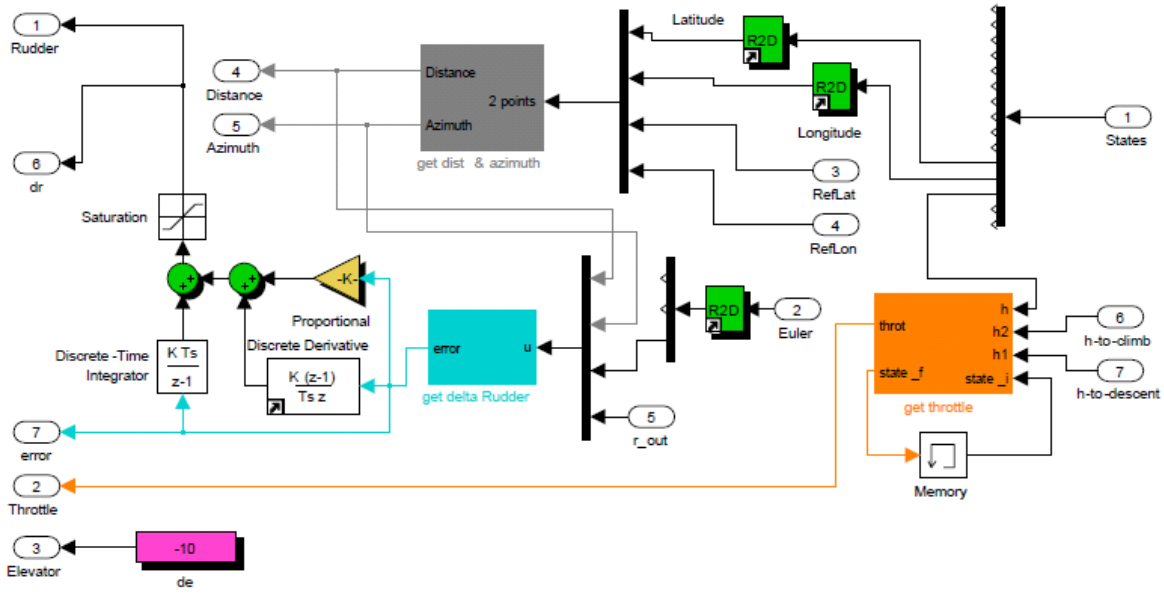


Figure 24: Autopilot block

5.4.3 Simulation Results

Parameters used in all simulations:

Reference latitude	: 37.422775°	Reference longitude	: -122.1764°
r_{out}	: 75 m	r_{in}	: 65 m
K_P	: 3/180	K_D	: 0.4
K_I	: 0 or 0.001	ΔT	: 0.05 or 0.25s (time-step)
δe	: -5°		

(Only climb state was simulated since the main objective of the simulations were to test lateral control law)

5.4.3.1 Comparison between PD and PID Control

In PID control, error in heading direction almost goes to zero because of the integral control but it affects the damping of the system and increases the settling time. This can be seen in the following figure, with PID control, the aircraft is inside the region but the radius of turn takes time to stabilize; however, with PD control, aircraft stabilizes quickly but has some steady state error (Figure 25).

Outer red circle in all plots has radius 75 m and the inner red circle has radius 60 m.

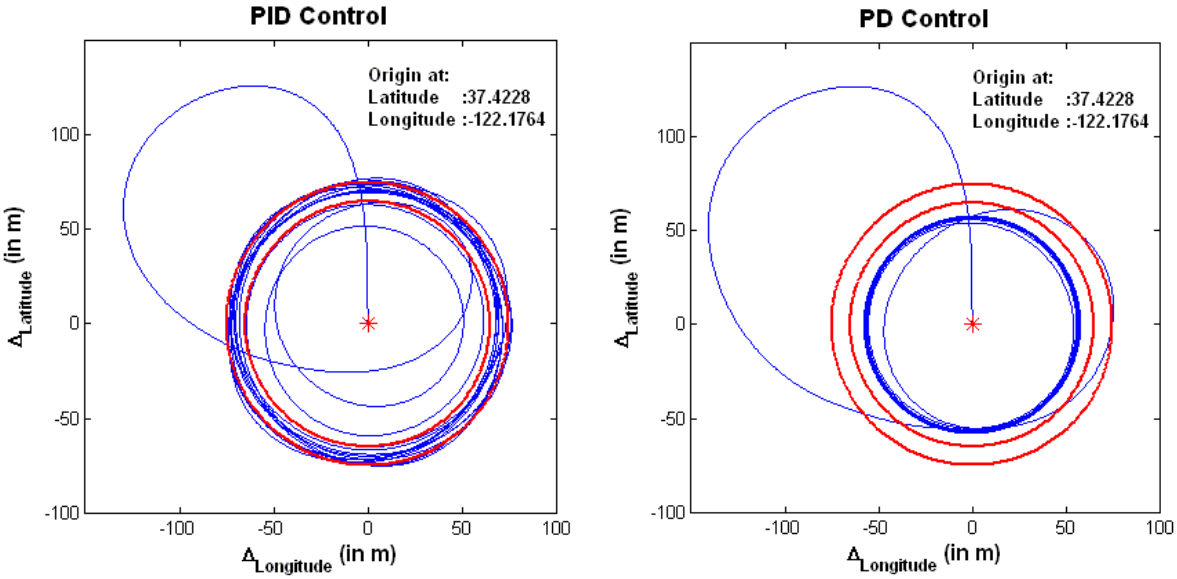


Figure 25: Comparison between PD and PID control

Robustness with respect to launching location and winds

To check the effect of starting point, the aircraft was launched far from the center, and in both cases (PD and PID control) it managed to get to circular region. Again, as expected, aircraft with PID control takes time to stabilize (Figure 26).

To investigate the effect of wind, 5.5 knots wind from NE was added to the simulations. PD control seemed to be more robust to winds than PID control (Figure 27). This was the main reason of choosing PD control over PID, even though the error in heading direction never goes to

zero. But having zero error is not the main objective anyways, having a robust control law is more important.

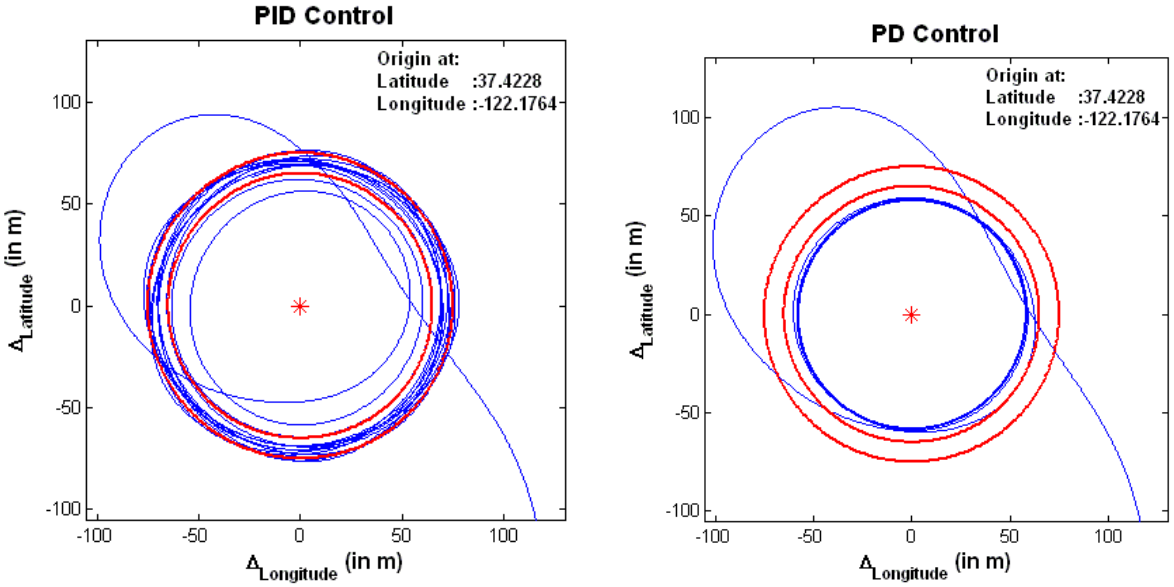


Figure 26: Robustness with respect to launching location

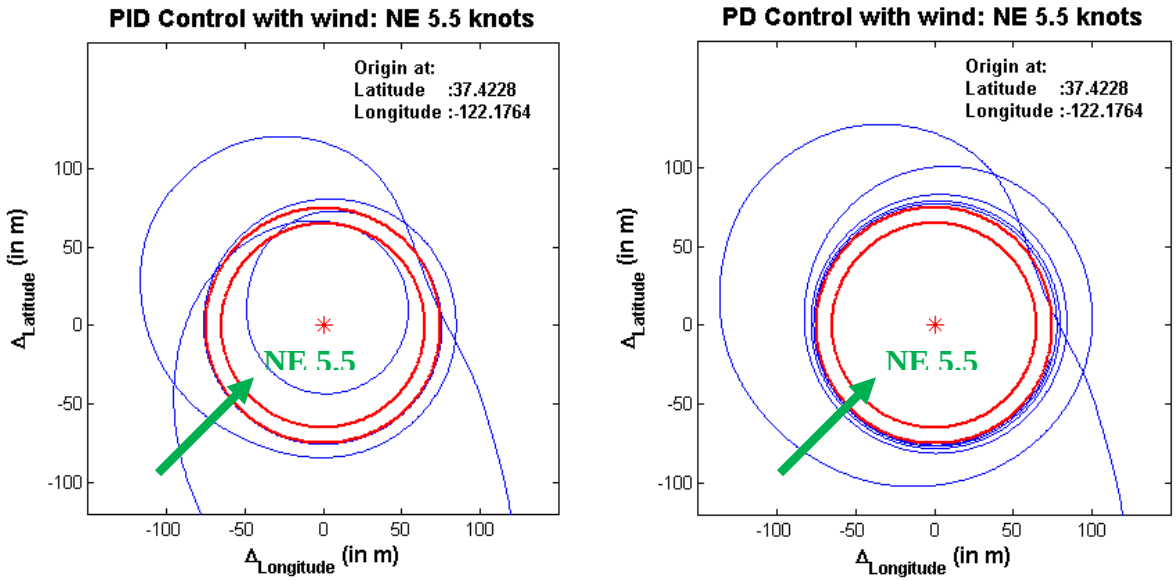


Figure 27: Robustness with respect to winds

Implementation of PD control in autopilot code has been explained in the following section.

5.5 Implementation of Control Law

Author: Harsh Menon

5.5.1 Details of the error formulation

From section 5.3, errors were defined to be:

$$\epsilon = \theta - \left(\psi + 180 - 90° \times \frac{r_m - r_{in}}{r_m - r}\right) \quad \ldots \quad if\ r < r_m$$

$$\epsilon = \theta - (\psi + 90°) \quad \ldots \quad if\ r_{in} < r < r_{out}$$

$$\epsilon = \theta - \left(\psi + 90° \times \frac{r_{out} - r_m}{r - r_m}\right) \quad \ldots \quad if\ r > r_m$$

Where ψ is the azimuth, θ is the course over ground, r_{in} is the inner radius and r_m is the average of the inner and outer radius (r_{out}).

Having defined the error, the error was then restricted to be between 0 and 360 degrees. This was accomplished using a while loop. Then, if the error was greater than 180 degrees, it was made negative since in this case the airplane would have to take a right turn instead of a left turn. Then, the rudder deflection was calculated using the following formula:

$$\delta_r(n) = floor\left(\frac{\left\{\frac{K_P}{180} + \frac{4K_D}{1000}\right\}\epsilon(n) - \left\{\frac{4K_D}{1000}\right\}\epsilon(n-1)}{0.41}\right)$$

The above formula uses the error at the current time step and the error at the previous time step (PD control). The gains K_P ,K_D were then varied from the trim buffers. The factors of 180 and 1000 were obtained using simulations and 4 is obtained from the fact that the sample time is 0.25 seconds. There is also a factor of 0.41. This is a conversion factor between the number of ticks (0-250) that we specify in the code and actual angular deflection measured. We calibrated the airplane by moving the rudder from 0-250 and measured the angular deflection as we changed the rudder setting. This gave us the conversion factor from degrees to ticks. Finally, we take a floor of the entire argument since we can only send unsigned char type variables to the aircraft. After calculating the above rudder deflection, we check whether the deflection is above

a given cutoff. If yes, then the rudder deflection is defined to be the neutral setting plus or minus the cutoff (depending on whether the error is negative or positive), otherwise the rudder deflection is set to be the neutral setting minus the value calculated above.

The above control law is used in the climb, descent and land states. In the climb and descent states, different values of the gains and cutoffs are used.

5.5.2 Details of C30 implementation and MATLAB simulation

The control law was implemented using a bottom-up approach. Initially a one-state machine was used where the elevator, rudder and throttle settings were inputs specified in the trim buffers. After attaining successful autonomous climb and descent, the one-state code was modified to a two-state climb and descent code. During testing, the team discovered that transition states were required for better performance and then finally an autonomous takeoff and land state were added resulting in the final code. In order to verify the control law implementation, the control law was also coded up in MATLAB and C++ and both implementations were used to predict rudder deflections generated by the code. The MATLAB implementation was modified so that it could input a data file obtained from flight testing and run the control law on the latitudes, longitudes and altitudes specified in the data file. Using this approach helped us predict and debug the aircraft performance, specifically in helping us determine what range of values of the proportional gain, derivative gain and cutoff would be optimal for the aircraft.

During initial testing, the MATLAB code helped us realize why our Stephano would go into a spiral mode. A plot of the rudder deflections from the MATLAB code is shown below (Figure 28).

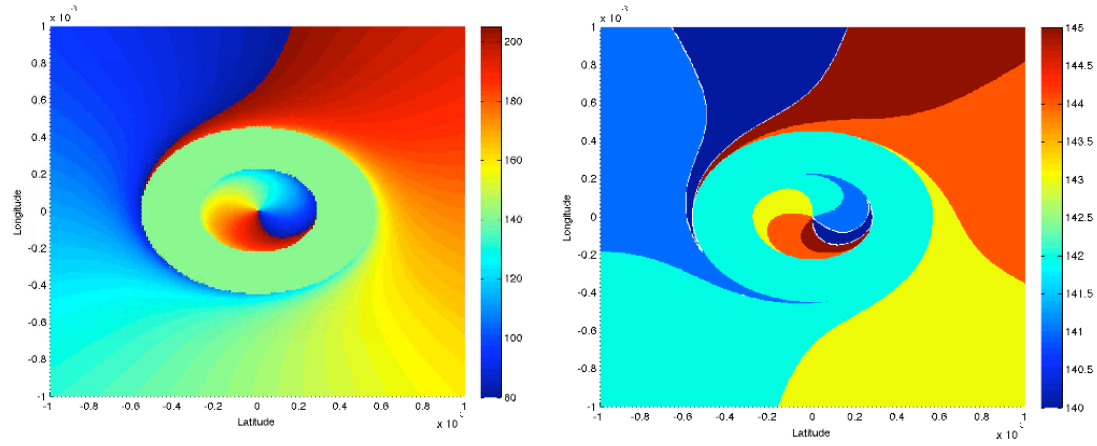


Figure 28: Rudder deflections before (left) and after (right)

Both the above figures are at a course over ground of 10 degrees. If we change the course over ground, it only rotates the above figures. The figure on the left shows the rudder deflections initially. We had initially tried to vary the deflections from 80 to 180. As we quickly realized in flight, this was a very high range for the deflections and so much control authority was not required (and in our case even led to instability). The plot on the right shows the rudder deflections after our realization and you can see that in this case we tried to keep the rudder deflections between 140 and 145. What you can also notice in the above plots is that in this control law there is always be a region where the rudder deflection changes abruptly. This is an artifact of the formulation of the control law and did not cause any instability in flight.

C30 code with control law implementation is given in the appendix.

6 FLIGHT TESTS

6.1 Test Procedures

Author: Gavin MacGarva

Numerous flight tests were required to make adjustments to ensure the UAV delivered on the fly-off day was capable of satisfying mission requirements. Team Tempest adhered to a strict test procedure in order to minimize risk to the project by incrementally introducing new elements of risk with each flight test. Although minor alterations were required due to the unique problems faced by each aircraft, the test and evaluation process adhered to the following procedure:

Glide Test – The aircraft would be hand launched into the wind from the edge of the dry Lake Lagunita in R/C mode. Commands were sent directly from the transmitter to a receiver on the aircraft (not through the Xbee radio) in order to remove the extra link in the chain of communication (the ground station run through the computer) and eliminate the probability of failure of this extra link from consideration. The objective of this test was to acquire reasonable trim settings.

Powered R/C Test – The aircraft would again be launched in R/C mode. As in the glide test, the ground station was not involved in the communication link; commands were sent directly from the transmitter to a receiver on the aircraft. The objective of this test was to evaluate the handling qualities of the aircraft in the various dynamic modes. The Dutch roll and spiral modes were of most concern. The Dutch roll mode was activated by alternating between full left and full right rudder inputs and returning to the neutral position to evaluate the response. The spiral mode was activated by one second full left or full right rudder inputs and returning to the neutral position to evaluate the response. This powered test under full R/C control was also used to validate the communications link between the autopilot board and the ground station. This was accomplished by carrying the autopilot board and GPS board and recording GPS data

of the flight through the ground station. The reliability of the link could then be established based on the number of data packets lost (if any).

Ground Station Evaluation/Neutral Trim Settings – The aircraft would be launched under R/C mode. However, unlike the first two tests remote control via the transmitter would be run through the ground station. The objective of this test was to acquire neutral rudder trim settings in climb and descent which would be observed through the ground station. Tests would also be conducted to establish a correlation between elevator deflection and rate of climb or descent.

Autonomous Testing – The aircraft would be launched under R/C mode and brought up to a safe altitude to allow recovery if the autopilot gave an unacceptable command. The UAV would then be put under autonomous control. Based on its behavior gains and trims settings would be adjusted in flight or the aircraft would be brought down under R/C control and the autopilot board reprogrammed with the necessary adjustments to the code. To reduce risk, the complexity of the code was increased gradually, initially testing only the climb state, then adding the descent state, and finally adding transitions between the two states. Takeoff and landing states were introduced last, and to reduce risk the UAV was never allowed to conduct a completely autonomous landing before the competition. Instead it was allowed to enter and stay in the landing state such that its behavior could be evaluated until it was seconds from touchdown. At that point the UAV was put back in R/C control.

6.1.1 Stephano

Glide testing for Stephano indicated that it had an undesirable right wing tip still. When the aircraft approached stall it would spiral down violently to right, making it a challenge to land. The cause of this issue was manufacturing. Close examination of the right wing revealed that an unintentional, positive incidence angle was put into the right wing where the center and outboard sections were joined. This would prove to be a significant issue which plagued the design.

Powered testing was conducted and the Dutch roll mode was very well damped. Testing of the spiral mode proceeded and the mode converged when it was activated to the left. When it was activated to the right, however, the aircraft would violently spiral down and to the right in a

maneuver the team called the “right turn death spiral” which, although recoverable under R/C control, would result in a significant loss of altitude.

Autonomous testing proceeded, however, the aircraft behaved poorly if it was ever disturbed in a turn to the right or banked too much to the right which would send Stephano into the “right turn death spiral.” Recovery would be required under R/C control and the test would need to be set up again. Valuable flight time was wasted in this process. Attempts to fix the tip stall issue by tripping the boundary layer using tape and sanding thin sections of the upper surface proved unsuccessful. Because of this, Stephano failed to be a suitable test bed for the autopilot and much effort was focused on getting Prospero ready for flight testing as quickly as possible.

6.1.2 Prospero

Prospero performed much better than Stephano. Glide tests revealed no sign of tip stall and the UAV could be landed easily and at a significantly lower speed. Stalls intentionally induced during the power testing phase did not favor yawing to the right or left side further suggesting that tip stall was not an issue with the aircraft. Testing of the dynamic modes revealed the Dutch roll mode to be well damped. More significantly, the spiral mode converged when activated both to the left and to the right and did so much more quickly than Stephano (when it converged to the left).

Rapid, resounding success in the first three phases of flight testing allowed Team Tempest to proceed to autonomous testing very quickly with Prospero. The majority of the time spent flight testing Prospero was done in this phase, and once the control law was tuned and gains were adjusted it performed admirably in autonomous mode.

6.2 Flight Tests

Author: Harsh Menon

Before we introduce the results, the following color convention is used to show the vehicle state of the airplane during the course of its flight (Table 13).

Table 13: Color convention

State	Color
Climb	Blue
Descent	Red
Transition to Climb	Green
Transition to Descent	Light Blue (Cyan)
Takeoff	White
Land	Yellow

Prior to our discussion of the flight results, we'd like to highlight some important issues that we will frequently encountered

1. Effect of Propwash
2. Effect of Derivative control
3. Effect of Decreasing Power

Through our discussion of the flight results, we will explain how the above effects were crucial to the airplane performance.

6.2.1 May 13, 2010

The flight data from May 13, 2010 is shown below (Figure 29). At this point, we were flying Stephano (with known tip stall issues) and the figure below shows a segment of autonomous flight for Stephano.

At this point of flight testing, we were using pure proportional control and were varying the control authority of the aircraft. As you can see from Figure 29, even though we were increasing the control authority of the aircraft, the airplane was drifting away from the circle. The airplane would eventually return to the circle but more often than not we would have to step in and switch to RC because the airplane would have drifted far beyond the allowed limits. All this prompted us to add derivative control to the control law with the hope that we could reduce the drift and bring the airplane back to the circle. Figure 30 shows the first implementation of derivative control on May 25, 2010.

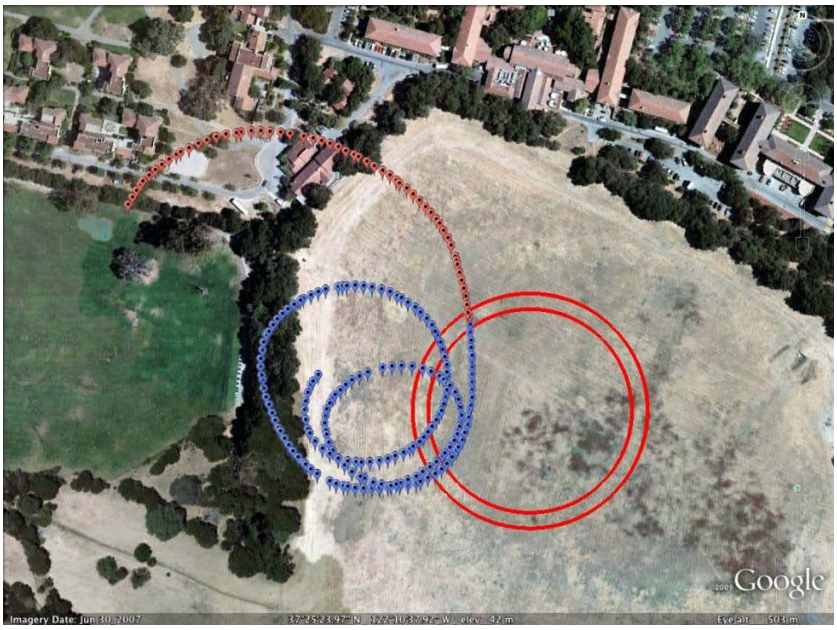


Figure 29: Flight data from May 13, 2010

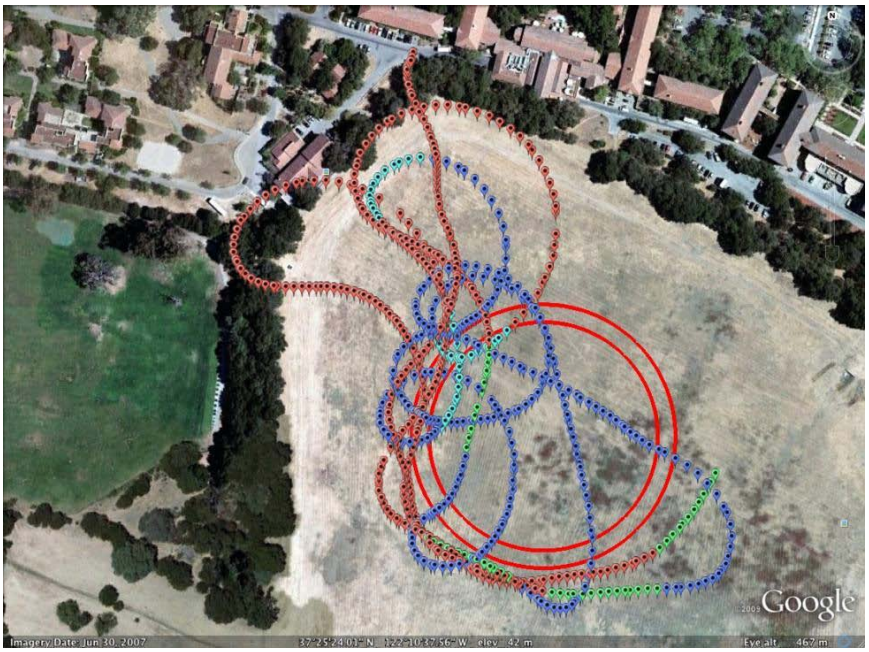


Figure 30: Flight data from May 25, 2010

One of the first things you can notice is that the airplane now returns to the circle, albeit mostly in the descent state. We flew the above flight using gain values that we had obtained from

simulation, i.e., a proportional gain of 3 and a derivative gain of 3. The fact that the simulation values worked the first time was a pleasant surprise. As time went by, we continued tweaking the gains and cutoffs.

6.2.2 June 1, 2010

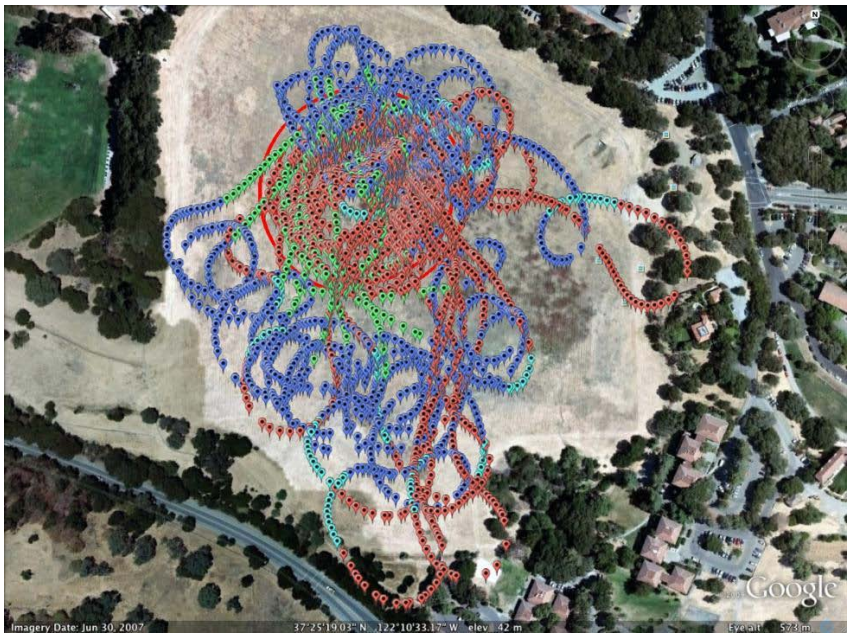


Figure 31: Flight data from June 1, 2010

Figure 31 shows the flight test data from June 1. This was a much longer flight (rehearsal for the final fly-off). Having addressed the effect of derivative control, we will now focus on why the control law is more effective in descent than in climb. Both climb and descent have the same control law implementation, but the core difference is the effect of propwash. The figure below shows the climb and descent states of the flight.

Figure 32: Climb and descent data from June 1, 2010

As can be seen from Figure 32, the aircraft came back to the circle and stayed within the region more in the descent state than the climb state. In fact, in the climb state, the aircraft had a tendency to make smaller radius turns as it climb to altitude. The control law was in effect in the climb state because even though the airplane was making smaller radius turns on average it was not moving away from the center of the circle. Figure 33 shows the circle fit to the data points to determine the turn radius and a plot of the turn radius versus the rudder deflection for the climb state.

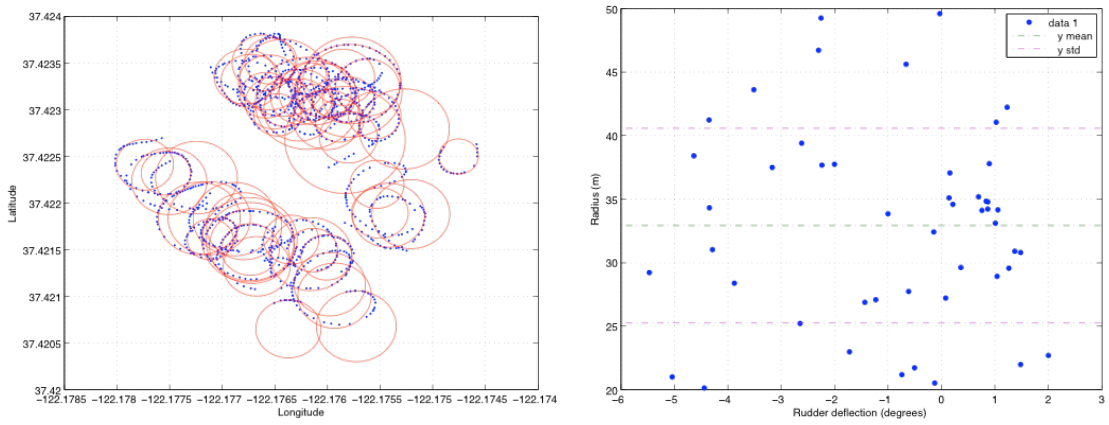


Figure 33: Climb data from June 1, 2010

Based on the above data, we see that during climb, the airplane can negotiate a mean turn radius of around 35 m with a standard deviation of 8 m. Similarly, we can generate similar plots for the airplane during descent (Figure 34).

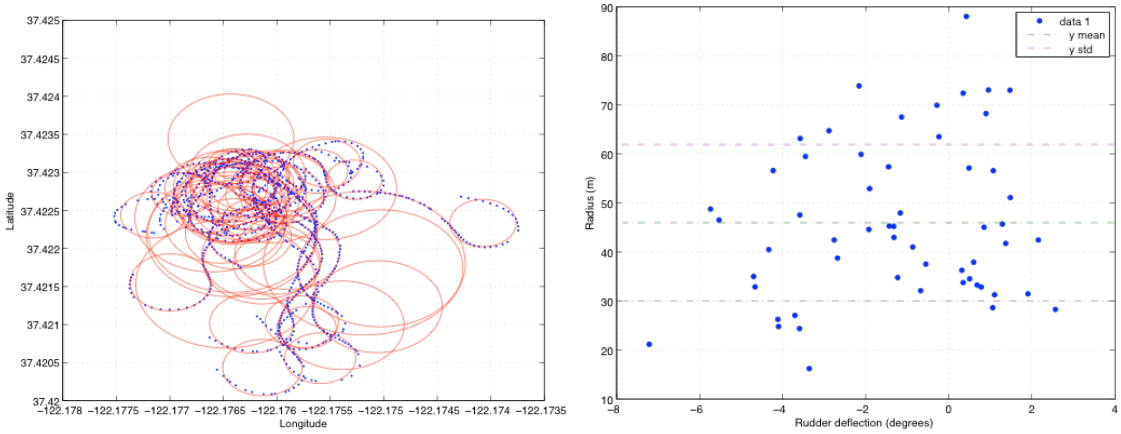
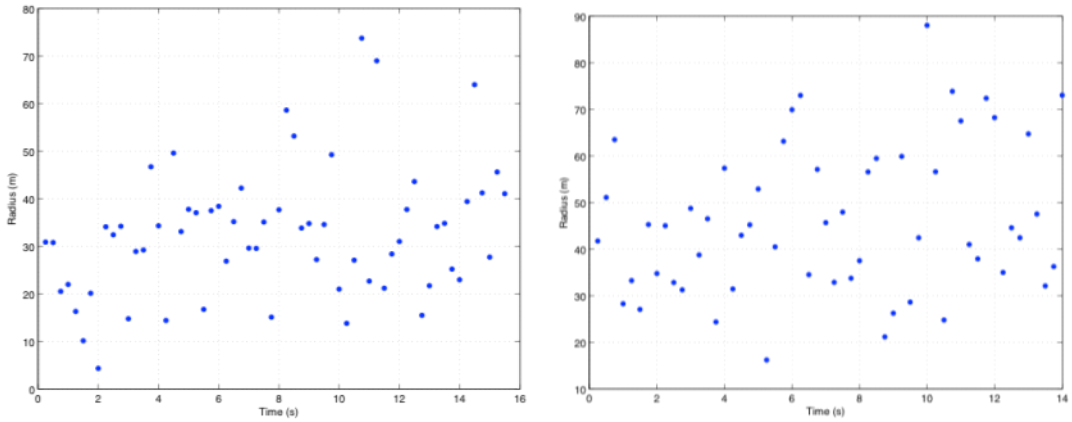


Figure 34: Descent data from June 1, 2010

From the descent data, we can see that the airplane was able to negotiate a mean turn radius of around 46 m with a standard deviation of 15 m. Therefore, we can see that the effect of the propwash was to effectively reduce the turn radius of the airplane. Thus, we can see in this data and in all other subsequent flight data, the airplane climbed to the desired altitude of 175 m in a circular path with a smaller turn radius than desired and this led to sub-optimal performance since the airplane was losing lift in order to negotiate such a sharp turn. The reason for this is not entirely obvious. One would assume that since the rudder is in the propwash, its effectiveness would be reduced and hence the airplane would actually end up with a larger turn radius. However, in the above reasoning we have neglected the torque of the propeller. The torque of the propeller gives the airplane a tendency to turn and hence the airplane ends up performing a much sharper turn than anticipated. For our airplanes, the propeller was rotating in the clockwise direction and hence gave the airplane a tendency to turn left.

Finally, we should note that as the aircraft continues flying its cycles, the propwash effect becomes less dominant since the propeller battery is being drained. As a result of this, we had to

implement elevator scheduling to try and use the lift of the airplane to account for the lack of energy from the propeller. Figure 35 shows the variation of turn radius with time during climb.



(a) Climb (b) Descent

Figure 35: Variation of climb and descent radii

From Figure 35, it's hard to determine whether or not the radius increases in time, but by averaging the data at two different time levels we see that during climb, the turn radius increases with time, while it is almost the same during descent.

6.3 Fly-off Flight Test Results

Author: Harsh Menon

The flight test results from the fly-off are presented in this section. The results from June 2, 2010 are shown below.

This flight was our conservative attempt. We had the airplane complete 16 cycles after which it went into a land state. Since we launched the airplane outside the circle, we can see that the airplane's initial climbs were outside the circular region. However, after a couple of descent cycles the airplane returned to the desired region. As expected, the airplane climbed in circles of radius around 35 m while the radii were longer during descent.

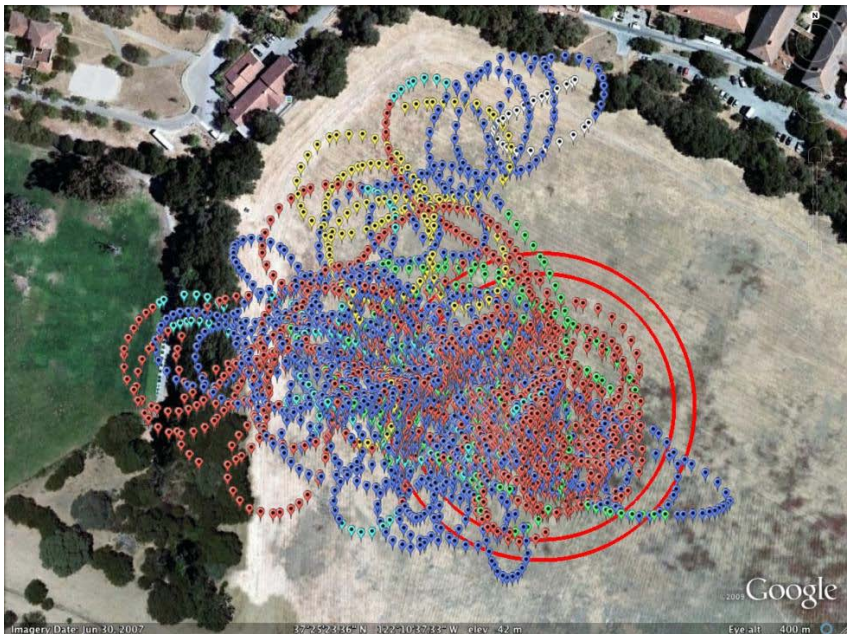


Figure 36: Flight data from June 2, 2010

Figure 37: Climb and descent data from June 2, 2010

The results from June 3, 2010 are shown below.

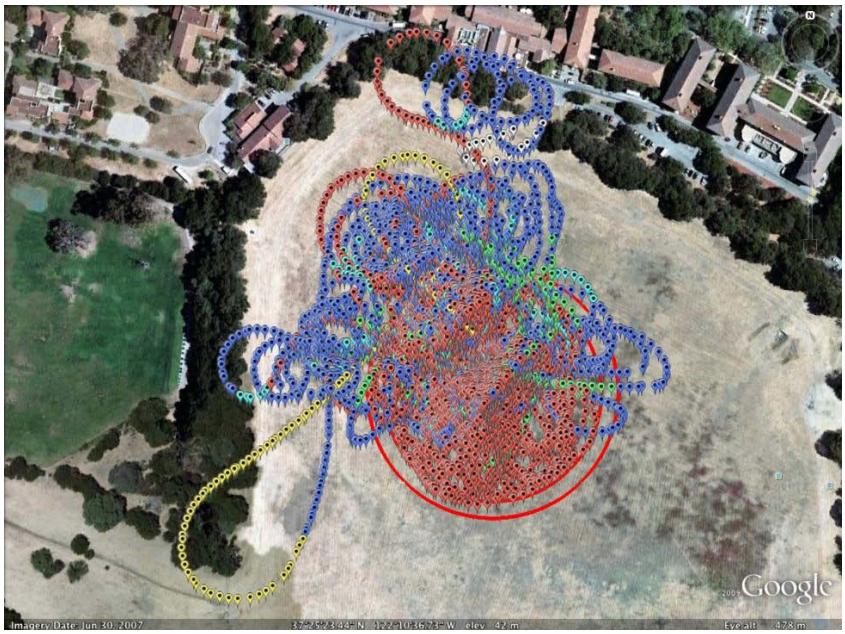


Figure 38: Flight data from June 3, 2010

Figure 39: Climb and descent data from June 3, 2010

The airplane performed better on June 3, 2010 as witnessed by the plots above. The airplane stayed within the desired region during the descent state and did not drift too far from the desired region during the climb state. During this flight, we attempted to climb as high as possible and hence we reached critical levels of the propulsion battery. This can be seen from the yellow line (land state) in the plot above. In the climb region just before the yellow line, the airplane was

trying to climb but was unable to do so and hence end up drifting away from the region. Eventually, we decided to land the airplane and as the figures above show, the airplane returned successfully to the landing region.

6.4 Fly-off Flight Test Data Analysis

Author: Amrita Mittal

A MATLAB based function was written that on one click of mouse would fully analyze the flight data; give a number for total altitude gain, rate of climb, and climb angle, climb history, true air speed, and wind speed which proved to be very helpful in deciding change in optimum elevator settings with flight time during testing in the field. Climb history helped us identify if we were stalling too many times in climb which would result in a low rate of climb and corrective measures were taken. Information about the wind speed and direction helped understanding the unexpected behavior of the aircraft at times.

On June 3, 2010, Prospero gained a cumulative altitude of 7060 ft, calculated by adding the individual peak to-peak difference in altitude for each climb and part of transition states where the aircraft gained altitude. Our predicted total altitude was close to 11000 ft which was due to underestimation of drag. Figure 40 shows a pictorial representation of how we estimate it.

Figure 41 shows the climb and descent cycles with the rate of climb plotted against the cycle. In the beginning, the rate of climb was pretty high and steady but as the battery power depleted, Prospero climbed slowly until the 19^{th} cycle which could not be completed due to low battery. The figure also shows the elevator and rudder settings with time. As the battery power deleted, more nose-up elevator was required to climb.

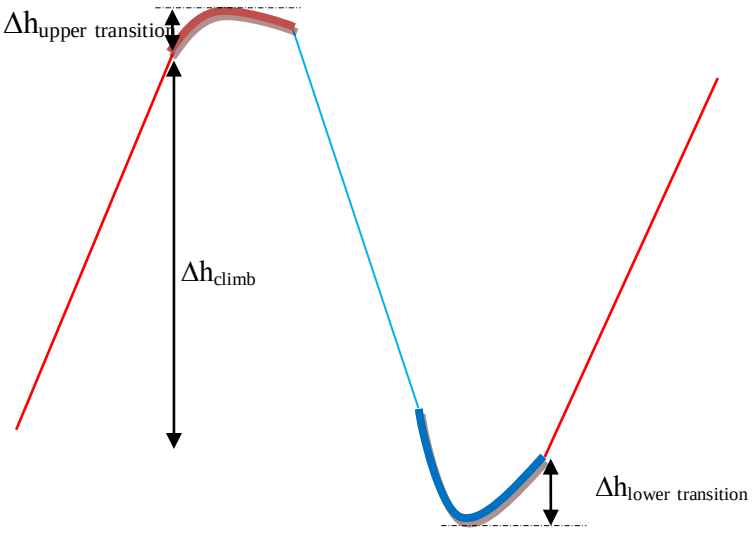


Figure 40: Total Altitude = $\sum \Delta h_{climb} + \Delta h_{upper\ transition} + \Delta h_{lower\ transition}$

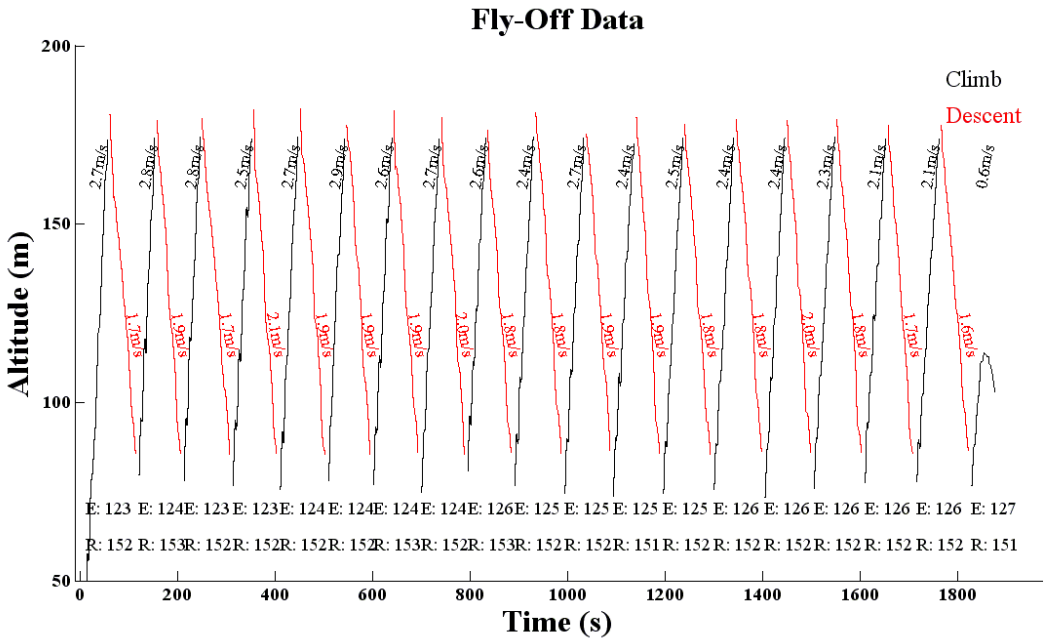


Figure 41: Rate of climb for different states

Analysis of climb angle and L/D required the knowledge of the true air speed which was estimated to be the mean of ground speed when plotted against the course over ground.

Figure 42 shows how this is done.

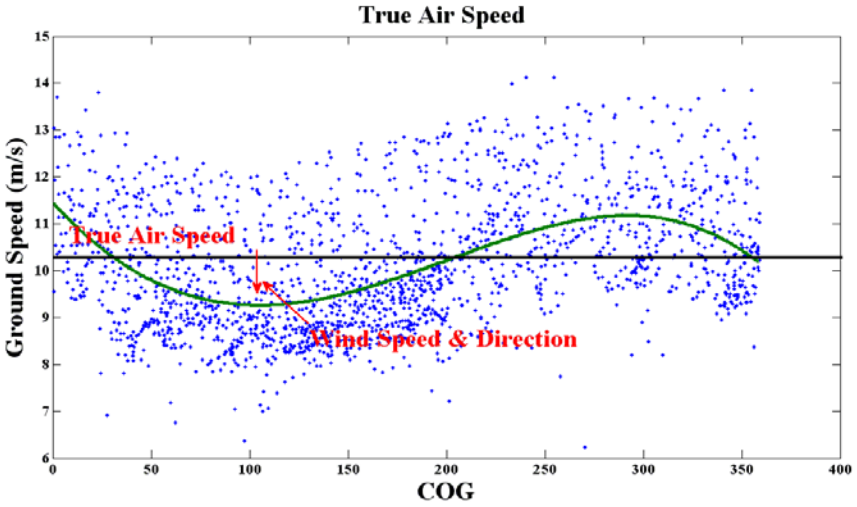
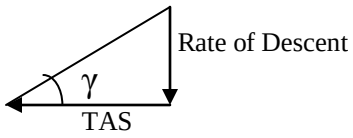


Figure 42: Estimation of true airspeed

Climb and glide angles are calculated from the knowledge of true air speed and rate of climb/descent respectively.

$\gamma = \tan^{-1}(\text{Rate of Descent}/\ \text{TAS})$



Information of the glide angle can be used to find the total lift,

$$L = W \times \cos\gamma$$

$D = W \times \sin\gamma$ in Descent State

$D = T - W \times \sin\gamma$ in Climb State

A theoretical estimate of thrust previously derived in the performance module of optimizer was used to estimate drag during climb.

Table 14 summaries all the test results in comparison to theoretical estimates.

Table 14: Tests results in comparison to theoretical estimates

	Theoretical	Flight Testing
Total Altitude	11500 ft	7060 ft
Rate of Climb	4.1 m/s	2.54 m/s
Climb Angle	28^{o}	15^{o}
Climb Speed	8.3 m/s	9.7 m/s
Glide Angle	-	10^{o}
C_L	0.5	0.45
L/D in climb	13	5
L/D in Descent	-	5.5
Best L/D in landing	-	13.3

7 LESSONS LEARNED

Author: Harsh Menon
Contribution: Everyone

1. The quality of manufacturing matters! If you are trying to be ahead of the curve and are trying to manufacture an airplane as fast as possible, make sure you know everything you need to know before you start manufacturing. Thing such as getting the boom straight and having a smooth wing will make all the difference – even if it is your first airplane.
2. Don't underestimate the effect of propwash. The propwash affected our climb performance and in retrospect, we might have chosen a canard-pusher configuration.
3. Some manufacturing tips –
    a. Introduce margins while using the laser cutter to manufacture the wing because the laser cutter eats more foam than expected.
    b. Do not put any foam pieces to enclose the wings while vacuum bagging as it may introduce unwanted twist.
    c. Check the vacuum pressure. Make sure it reads 20 psi.
    d. Make a jig to get a straight boom. This will give you better controllability over the airplane.
    e. Manufacture hinge joints for the control surfaces carefully.
4. For all the women who are going to take this class, make sure you tie your hair before you go to 353 to work. This will prevent you from having to cut hair when epoxy falls on it.
5. Make a back-up wing and fuselage. Your next flight may be your last.
6. Make sure your computer battery is fully charged. If your computer dies, you will have no control over your airplane.
7. Make sure your transmitter is connected properly to the computer. This cable is very sensitive. If plugged in too tight, it will cause the controls to oscillate very rapidly.

8. If you lose both autonomous and RC control, try restarting the ground station first. If that doesn't work, run after the airplane as fast as you can.
9. Make sure you get a lot of time testing the airplane getting the right trims and cutoffs. Also, try and see how these values change with wind conditions.
10. Simplicity is good, but you can make the control law more than just a constant rudder deflection – as long as it does not become too complicated.
11. Don't spend a lot of time modifying the control law. Rather spend that time testing the control law performance.
12. Don't underestimate the variation of the propulsion batteries. That one extra battery might give you the 10 m you need to get the highest altitude.
13. Make sure your stability derivatives with respect to the rudder and elevator deflections are not too high because you might not have enough resolution in your control surfaces.

APPENDIX A: Microcontroller C30 Code

Author: Harsh Menon

```
#include "controlLaw.h"
#define lockTimerThreshold 2.0                  //Time required with 3D GPS lock before going to rest of state machine
//#define nolockthreshold 3 //Time in seconds for how long without a lock before we go to NOLOCK state

#define power_on 250
#define power_off 0

#define ele_land 130
#define maxcycles 16

#define critVoltage 7.3 // Volts

//Initial GPS coordinates, center of Lake Lagunita. These get overwritten as soon as we get the initial GPS location
//static double latInit =       37.422775;          //Initial GPS latitude
//static double lonInit =  -122.1764; //Initial GPS longitude
//static double altInit = 0.;  //Initial GPS altitdue
//static double reflat = 37.423575;
//static double reflon = -122.1770;
static double reflat = 37.422775;
static double reflon = -122.1764;
static double refalt = 0.;

static double dist;
static double theta;
static double psi;

static double error;
static double error_last = 0.;
static double drud;
static int cutoff_climb = 7;
static int Kp_climb = 2;
static int Kd_climb = 7;
static int cutoff_desc = 15;
static int Kp_desc = 8;
static int Kd_desc = 7;

//static double alt_history[20]; // altitude history to determine when to land
//static double avg_delta_height = 0.;
//static int count = 0;
static int climbcount = 1;
static double r2 = 25.;

////////////////////////////////////////////////////////////////////////////
//Runs control laws, takes state vector input and determines how to move servos
```

```
void Control(StateVector *sv_ptr,ControlVector *con_ptr) {

//dr desc = 151
//rin = rout - 10
//hclimb = 175
//hdesc = 85
// rout = 25

unsigned char rud_on = con_ptr->trimBuffer[0];
unsigned char ele_on = con_ptr->trimBuffer[1];
unsigned char rud_off = con_ptr->trimBuffer[2];
unsigned char ele_off = con_ptr->trimBuffer[3];

/*
unsigned char cutoff_climb = con_ptr->trimBuffer[4];
unsigned char Kp_climb = con_ptr->trimBuffer[5];
unsigned char Kd_climb = con_ptr->trimBuffer[6];

unsigned char cutoff_desc = con_ptr->trimBuffer[7];
unsigned char Kp_desc = con_ptr->trimBuffer[8];
unsigned char Kd_desc = con_ptr->trimBuffer[9];
*/


        //check autoFlag (>125 = Auto, <=125 = RC), if RC do nothing to con_ptr
        if (con_ptr->autoFlag > autoOn) {
                int state = sv_ptr->state; //Get current state

                switch(abs(state)){

                        case INITIALIZE: // Neutral values
                                con_ptr->aileron = aileronInit;
                                con_ptr->elevator = elevatorInit;
                                con_ptr->throttle = throttleInit;
                                con_ptr->rudder = rudderInit;
                        break;

                        case TAKEOFF:
                                con_ptr->elevator = (unsigned char) (ele_on - 1);
                                con_ptr->throttle = (unsigned char) power_on;
                                con_ptr->aileron = con_ptr->rudder = (unsigned char) rud_on;

                        break;

                        case CLIMB:
                                dist = Distance(sv_ptr->lat, sv_ptr->lon, reflat, reflon);
                                psi = Azimuth(sv_ptr->lat, sv_ptr->lon, reflat, reflon);
                                theta = sv_ptr->cog;

                                // Lateral Control Law
                                if (dist < (r2 - 10.0)){

                                        error = -psi + theta - 180.0 + 90.0*(5.)/(r2 - 5. - dist);
```

```
}
else if ( (dist >= (r2 - 10.0)) && (dist <= r2) ){
        error = -psi + theta - 90.0;
}
else{
        error = -psi + theta - 90.0*(5.)/(dist - r2 + 5.);
}

        //Check if error is between 0 and 360
        if (error > 360) {
                                        while (error > 360) {
                                        error = error - 360;
                                        }
        }
        else if (error < 0){
                                        while (error < 0) {
                                        error = error + 360;
                                        }
        }

        if ( error >= 180 ) {
                error = error - 360;
        }

        //Calculate rudder deflection from trim value based on PD control

        drud = floor(( ( (Kp_climb/180.) + 4.0*Kd_climb/1000.)*error -
(4.0*Kd_climb/1000.)*error_last)/0.41);


                if (  abs(drud) > cutoff_climb )
                        {
                                if (error > 0)
                                {
                                con_ptr->aileron = con_ptr->rudder =
(unsigned char) (rud_on - cutoff_climb);
                                }
                                else
                                {
                                con_ptr->aileron = con_ptr->rudder =
(unsigned char) (rud_on + cutoff_climb);
                                }

                        }
                else
                        {
                        con_ptr->aileron = con_ptr->rudder = (unsigned
char) (rud_on - drud);
                        }

con_ptr->elevator = ele_on;
con_ptr->throttle = power_on;
```

```
            // Elevator scheduling due to battery depletion
            if ( (climbcount > 1) && (climbcount <= 4)  ) {
                        con_ptr->elevator = (unsigned char)(ele_on + 1);
            }
            else if ( (climbcount > 4) && ( climbcount <= 8 ) ){
                        con_ptr->elevator = (unsigned char)(ele_on + 2);
            }
            else if (climbcount > 8){
                        con_ptr->elevator = (unsigned char)(ele_on + 4);
      }

            //Update values
            error_last = error;

break;

case DESCEND:

            dist = Distance(sv_ptr->lat, sv_ptr->lon, reflat, reflon);
            psi = Azimuth(sv_ptr->lat, sv_ptr->lon, reflat, reflon);
            theta = sv_ptr->cog;

            // Lateral Control Law
            if (dist < (r2 - 10.0)){

                        error = -psi + theta - 180.0 + 90.0*(5.)/(r2 - 5. - dist);
            }
            else if ( (dist >= (r2 - 10.0)) && (dist <= r2) ){
                        error = -psi + theta - 90.0;
            }
            else{
                        error = -psi + theta - 90.0*(5.)/(dist - r2 + 5.);
            }

                        //Check if error is between 0 and 360
                        if (error > 360) {
                                                while (error > 360) {
                                                error = error - 360;
                                                }
                        }
                        else if (error < 0){
                                                while (error < 0) {
                                                error = error + 360;
                                                }
                        }

                        if ( error >= 180 ) {
                                    error = error - 360;
                        }

                        //Calculate rudder deflection from trim value based on PD control
```

```
                                        drud = floor( ( ( (Kp_desc/180.) + 4.0*Kd_desc/1000.)*error -
(4.0*Kd_desc/1000.)*error_last)/0.41 );


                                                if (  abs(drud) > cutoff_desc )
                                                        {
                                                                if (error > 0)
                                                                {
                                                                con_ptr->aileron = con_ptr->rudder =
(unsigned char) (rud_off - cutoff_desc);
                                                                }
                                                                else
                                                                {
                                                                con_ptr->aileron = con_ptr->rudder =
(unsigned char) (rud_off + cutoff_desc);
                                                                }

                                                        }
                                                else
                                                        {
                                                        con_ptr->aileron = con_ptr->rudder = (unsigned
char) (rud_off - drud);
                                                        }

                                con_ptr->elevator = ele_off;
                                con_ptr->throttle = power_off;

                                //Update values
                                error_last = error;

                        break;

                        case TRANSCLIMB:
                                con_ptr->elevator = ele_on;
                        break;

                        case TRANSDESC:
                                con_ptr->elevator = ele_off;
                        break;

                        case LAND:


                                if (sv_ptr->gpsAltitude > 25.) {

                                dist = Distance(sv_ptr->lat, sv_ptr->lon, reflat, reflon);
                                psi = Azimuth(sv_ptr->lat, sv_ptr->lon, reflat, reflon);
                                theta = sv_ptr->cog;

                                // Lateral Control Law
                                if (dist < (15. - 10.0)){

                                        error = -psi + theta - 180.0 + 90.0*(5.)/(15. - 5. - dist);
```

```
}
else if ( (dist >= (15. - 10.0)) && (dist <= 15.) ){
	error = -psi + theta - 90.0;
}
else{
	error = -psi + theta - 90.0*(5.)/(dist - 15. + 5.);
}

	//Check if error is between 0 and 360
	if (error > 360) {
		while (error > 360) {
		error = error - 360;
		}
	}
	else if (error < 0){
		while (error < 0) {
		error = error + 360;
		}
	}

	if ( error >= 180 ) {
		error = error - 360;
	}

	//Calculate rudder deflection from trim value based on PD control

	drud = floor( ( ( (Kp_desc/180.) + 4.0*Kd_desc/1000.)*error -
(4.0*Kd_desc/1000.)*error_last)/0.41 );


		if (  abs(drud) > cutoff_desc )
			{
				if (error > 0)
				{
				con_ptr->aileron = con_ptr->rudder =
(unsigned char) (rud_off - cutoff_desc);
				}
				else
				{
				con_ptr->aileron = con_ptr->rudder =
(unsigned char) (rud_off + cutoff_desc);
				}

			}
		else
			{
			con_ptr->aileron = con_ptr->rudder = (unsigned
char) (rud_off - drud);
			}


}
else
```

```
                                        {
                                                con_ptr->aileron = con_ptr->rudder = rud_off;
                                        }

                                        con_ptr->elevator = ele_land;
                                        con_ptr->throttle = power_off;

                                        //Update values
                                        error_last = error;

                                break;

                        default: //default action
                                        con_ptr->aileron = aileronInit;
                                        con_ptr->elevator = elevatorInit;
                                        con_ptr->throttle = throttleInit;
                                        con_ptr->rudder = rudderInit;
                        break;
                        }
                }
}


////////////////////////////////////////////////////////////////////////////
//Sets the vehicle state
void SetState(StateVector *sv_ptr, ControlVector *con_ptr) {

        static unsigned char stateChanged = 1;
        static double time = 0.0;                       //Seconds since initialization
        static double stateTime = 0.0;                  //For example state machine
        static double lockTimer = 0.0;                  //How long GPS has had a 3D lock
        static int last_state = INITIALIZE; //Used to recover to last known state from loss of GPS lock

        unsigned char rud_on = con_ptr->trimBuffer[0];
        unsigned char rud_off = con_ptr->trimBuffer[2];
        unsigned char gotoland = con_ptr->trimBuffer[4];


        time += GPSperiod_sec;
        int state = sv_ptr->state;
        //int i;

        //Run State Machine
        switch(abs(state)){

  /*********************************************************************
   INITIALIZE
   *********************************************************************/
                case INITIALIZE:
                // Entry action
                if (stateChanged){
                                //Entry actions here
                                lockTimer = 0.0;
```

```
		}

			// Input actions
			if (sv_ptr->gpsLock) {
				lockTimer +=  GPSperiod_sec;
			} else {
				lockTimer = 0.0;
			}

		// Exit condition(s) & exit actions
		if (lockTimer > lockTimerThreshold) {
		// Update state
		state = TAKEOFF;

				// Exit actions
				//latInit = sv_ptr->lat;			//Save initial latitude
				//lonInit = sv_ptr->lon;			//Save initial longitude
				//altInit = sv_ptr->gpsAltitude;	//Save initial altitude

		}
	break;

		case TAKEOFF:
		// Entry action
		if (stateChanged){
				//Entry actions here
				stateTime = 0.0;
		}


		// Exit condition(s) & exit actions
			if (stateTime > stateTimeThreshold) {

				if ( gotoland > 0) {
				state = LAND;
				}
				else if (sv_ptr->gpsAltitude > (refalt + 40.0)) {
				state = CLIMB;
				}
			}


	break;

/**********************************************************************
 State1
 **********************************************************************/

		case CLIMB:
		// Entry action
		if (stateChanged){
				//Entry actions here
```

```
                        stateTime = 0.0;
                        //can set other variables or call functions heer
                        //Update climb time
                }

                // Exit condition(s) & exit actions
                        // use if statements to determine condition(s) to exit state
                        // carry out actions as necessary, and set state=NEWSTATE
                        if (stateTime > stateTimeThreshold) {
                                if (sv_ptr->gpsAltitude > (refalt + 175.)) {
                                        if ( (gotoland > 0 ) || ( sv_ptr->voltage < 7.3) ){
                                                state = LAND;
                                        }
                                        else {
                                                state = TRANSDESC;
                                                climbcount = climbcount + 1;
                                        }

                                }

                        }
        break;

                case TRANSDESC:
                // Entry action
                if (stateChanged){
                        //Entry actions here
                        stateTime = 0.0;
                        //can set other variables or call functions heer
                }

                        // Input actions
                        con_ptr->throttle = (unsigned char) (250 - (250./3.)*stateTime);
                        con_ptr->aileron = con_ptr->rudder = (unsigned char) (rud_on + stateTime*((rud_off -
rud_on)/3.0) );

                // Exit condition(s) & exit actions
                        // use if statements to determine condition(s) to exit state
                        // carry out actions as necessary, and set state=NEWSTATE
                        if (stateTime > 3.0) {
                                state = DESCEND;
                        }
                break;

                case DESCEND:
                // Entry action
                if (stateChanged){
                        //Entry actions here
                        stateTime = 0.0;
                        //can set other variables or call functions heer
                }

                // Exit condition(s) & exit actions
```

```
                        // use if statements to determine condition(s) to exit state
                        // carry out actions as necessary, and set state=NEWSTATE
                        if (stateTime > stateTimeThreshold) {
                                if (sv_ptr->gpsAltitude < (refalt + 85.)) {
                                state = TRANSCLIMB;
                                }
                        }
        break;

                case TRANSCLIMB:
                // Entry action
                if (stateChanged){
                                //Entry actions here
                                stateTime = 0.0;
                                //can set other variables or call functions heer
                }

                        // Input actions
                        con_ptr->throttle = (unsigned char) ((250./6.)*stateTime);
                        con_ptr->aileron = con_ptr->rudder = (unsigned char) (rud_off + stateTime*((rud_on -
rud_off)/6.0) );

                // Exit condition(s) & exit actions
                        // use if statements to determine condition(s) to exit state
                        // carry out actions as necessary, and set state=NEWSTATE
                        if (stateTime > 6.0) {
                                state = CLIMB;
                        }
                break;


                case LAND:
                // Entry action
                if (stateChanged){
                                //Entry actions here
                                stateTime = 0.0;
                                //Change center of circle
                                reflat = 37.423575;
                                reflon = -122.1770;

                }

                // Exit condition(s) & exit actions
                        // use if statements to determine condition(s) to exit state
                        // carry out actions as necessary, and set state=NEWSTATE
                        // NO EXIT CONDITIONS!

                break;

        default:  // if state input isn't valid do nothing???
        break;
        }
        state = abs(state);
```

```
	// Update state in state vector
	stateTime    += GPSperiod_sec;
	stateChanged  = (sv_ptr->state != state);
	if(stateChanged) {
		last_state = sv_ptr->state;
		ee_buffer[15] = NEW_DATA;
	}
	sv_ptr->state = state;


}


////////////////////////////////////////////////////////////////////////////
//Computes the distance in meters between two points, assumes spherical Earth with local flat Earth linearization
double Distance(double lat1, double lon1, double lat2, double lon2) {
	double lat2m = 111194.93;                                          //Conversion between
degrees of latitude and meters
	double lon2m = lat2m*cos((lat1+lat2)*M_PI/360.0); //Good assumption as long as lat1 approximately
equal to lat2

	return sqrt(lon2m*(lon1-lon2)*lon2m*(lon1-lon2) + lat2m*(lat1-lat2)*lat2m*(lat1-lat2));
}


////////////////////////////////////////////////////////////////////////////
//Computes the heading in degrees from point 1 to point 2
double Azimuth(double lat1, double lon1, double lat2, double lon2) {
	double lat2m = 111194.93;                                          //Conversion between
degrees of latitude and meters
	double lon2m = lat2m*cos((lat1+lat2)*M_PI/360.0); //Good assumption as long as lat1 approximately
equal to lat2

	double azimuth = 90.0 - 180.0/M_PI*atan2(lat2m*(lat2-lat1),lon2m*(lon2-lon1));

	if (azimuth < 0.0) {
		azimuth = azimuth + 360.0;
	}
	return azimuth;
}
```

APPENDIX B: AVL Input Files

Author: Gavin MacGarva

Stephano

```
Team Tempest Baseline Design (Stephano)
0.0 Mach
0 0 0.0 iYsym iZsym Zsym
#
0.1306 0.1127 1.1650 !Sref Cref Bref reference area, chord, span
0.0295 0.0 0.0 Xref Yref Zref moment reference location ! set to quarter chord location
#
#
#============================================================
SURFACE
Wing
24 1.0!Nchord Cspace Nspan Space
#
YDUPLICATE
0.00000 !reflect image wing about y=0 plane
#
ANGLE
3.88000 twist angle bias for whole surface
#
#------------------------------------------------------------
SECTION
# XLE YLE ZLE chord angle
0.00000 0.00000 0.00000 0.118 0.000 14 -2.0 32 -2.0
AFIL 0.0 1.0
sd7043.dat
CDCL
0.1186 0.01979 0.7673 0.02223 1.263 0.04657!CL1 CD1  CL2 CD2  CL3 CD3
#
SECTION
# XLE YLE ZLE chord angle
0.00000 0.2895 0.00000 0.13332 0.000 18 1.0 32 -2.0
AFIL 0.0 1.0
sd7043.dat
CDCL
0.1186 0.01979 0.7673 0.02223 1.263 0.04657!CL1 CD1  CL2 CD2  CL3 CD3
#
SECTION
# XLE YLE ZLE chord angle
0.024 0.5825 0.0761 0.094 0.000 5 1.0
AFIL 0.0 1.0
sd7043.dat
```

```
CDCL
0.1186 0.01979 0.7673 0.02223 1.263 0.04657!CL1 CD1  CL2 CD2  CL3 CD3
#
#============================================================
SURFACE
Right Horizontal Tail
12 1.0 10 -2.0!Nchord Cspace Nspan Space
#
YDUPLICATE
0.00000
#
ANGLE
0.0000
#
TRANSLATE
# deltaX deltaY deltaZ
0.60960 0.00000 0.0000 position bias for whole surface
#
#------------------------------------------------------------
SECTION
0.00000 0.00000 0.00000 0.1000 0.000
CONTROL
elevator 1.0 0.705128 0.0 1.0 1.0 1.0
SECTION
0.02200 0.1611 0.00000 0.07800 0.000
CONTROL
elevator 1.0 0.770000 0.0 1.0 1.0 1.0
#============================================================
SURFACE
Vertical Tail
12 1.0 10 -2.0!Nchord Cspace Nspan Space
#
ANGLE
0.0000
#
TRANSLATE
# deltaX deltaY deltaZ
0.60960 0.00000 0.0000 position bias for whole surface
#
#------------------------------------------------------------
SECTION
0.00000 0.00000 0.00000 0.1 0.000
CONTROL
Rudder 1.0 0.77 0.0 0.0 -1.0 1.0
SECTION
0.017 0.0 0.183 0.083 0.000
CONTROL
Rudder 1.0 0.722892 0.0 0.0 -1.0 1.0
#
#
```

#

Prospero

Team Tempest Second Design Iteration (Prospero)

```
0.0 Mach
0 0 0.0 iYsym iZsym Zsym
#
0.1456 0.1173 1.2452 !Sref Cref Bref reference area, chord, span
0.0305 0.0 0.0 Xref Yref Zref moment reference location ! set to quarter chord location
#
#
#============================================================
SURFACE
Wing
24 1.0!Nchord Cspace Nspan Space
#
YDUPLICATE
0.00000 !reflect image wing about y=0 plane
#
ANGLE
3.000 twist angle bias for whole surface
#
#------------------------------------------------------------
SECTION
# XLE YLE ZLE chord angle
0.00000 0.00000 0.00000 0.1220 0.000 14 -2.0 32 -2.0
AFIL 0.0 1.0
sd7043.dat
CDCL
0.1186 0.01979 0.7673 0.02223 1.263 0.04657!CL1 CD1  CL2 CD2  CL3 CD3
#
SECTION
# XLE YLE ZLE chord angle
0.00000 0.3210 0.00000 0.1220 -1.000 18 1.0 32 -2.0
AFIL 0.0 1.0
sd7043.dat
CDCL
0.1186 0.01979 0.7673 0.02223 1.263 0.04657!CL1 CD1  CL2 CD2  CL3 CD3
#
SECTION
# XLE YLE ZLE chord angle
0.021 0.6226 0.1098 0.1010 -3.000 5 1.0
AFIL 0.0 1.0
sd7043.dat
CDCL
0.1186 0.01979 0.7673 0.02223 1.263 0.04657!CL1 CD1  CL2 CD2  CL3 CD3
#
```

```
#============================================================
SURFACE
Right Horizontal Tail
12 1.0 10 -2.0!Nchord Cspace Nspan Space
#
YDUPLICATE
0.00000
#
ANGLE
0.0000
#
TRANSLATE
# deltaX deltaY deltaZ
0.7200 0.00000 0.0000 position bias for whole surface
#
#------------------------------------------------------------
SECTION
0.00000 0.00000 0.00000 0.1000 0.000
CONTROL
elevator 1.0 0.80 0.0 1.0 1.0 1.0
SECTION
0.03400 0.1504 0.00000 0.06600 0.000
CONTROL
elevator 1.0 0.69697 0.0 1.0 1.0 1.0
#============================================================
SURFACE
Vertical Tail
12 1.0 10 -2.0!Nchord Cspace Nspan Space
#
ANGLE
0.0000
#
TRANSLATE
# deltaX deltaY deltaZ
0.7200 0.00000 0.0000 position bias for whole surface
#
#------------------------------------------------------------
SECTION
0.00000 0.00000 0.00000 0.1 0.000
CONTROL
Rudder 1.0 0.8 0.0 0.0 -1.0 1.0
SECTION
0.0290 0.0 0.16 0.0710 0.000
CONTROL
Rudder 1.0 0.71831 0.0 0.0 -1.0 1.0
#
#
#
```