STRUCTURED PROGRAMMING OF AIRCRAFT SYNTHESIS: THE DESIGN PROCESS FOR ROTORCRAFT: AN ENGINEERING SENIOR DESIGN CAPSTONE PROJECT

Josiah Everhart

STRUCTURED PROGRAMMING OF AIRCRAFT SYNTHESIS: THE DESIGN PROCESS FOR ROTORCRAFT: AN ENGINEERING SENIOR DESIGN CAPSTONE PROJECT

by

JOSIAH EVERHART

Presented to the Faculty of the Honors College of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

HONORS BACHELOR OF SCIENCE IN AEROSPACE ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

Spring 2020

ACKNOWLEDGMENTS

I would like to acknowledge Dr. Dudley Smith for his support in this project. He has lent his substantial experience in the world of rotorcraft to helping me to understand the design process. The equations can be taught easily, but the physical interpretation of these equations, and an intuitive knowledge of how the design variables work together and affect one another is something of great value that should be passed on to the engineers of the future. Dr. Smith has done just that with a teaching style in which the process drives the use of the equations, rather than the other way around. This leads to a superior understanding of the design process not just for rotorcraft, but for any design problem an engineer would be likely to come across.

I would also like to acknowledge my team. Without this group of talented and brilliant individuals, I have no doubt that I would not have been able to reach the point that I have, both in my coursework and projects thus far, and on this project specifically. Each team member brought a unique specialization to the team, and our learning experience together was beyond anything that can be achieved in a one-on-one or classroom learning experience. The discussions were continually stimulating, and they drove the most effective engine of learning, asking questions. I could hardly have asked for a better team.

May 08, 2020

ABSTRACT

STRUCTURED PROGRAMMING OF AIRCRAFT SYNTHESIS:

THE DESIGN PROCESS FOR ROTORCRAFT:

AN ENGINEERING SENIOR DESIGN

CAPSTONE PROJECT

Josiah Everhart, B.S. Aerospace Engineering

The University of Texas at Arlington, 2020

Faculty Mentor: Dudley Smith

The design process entails a collection of connected variables. Any program constructed to automate this process requires a carefully defined structure such that design modifications are easy to implement, and errors in the program are easy to trace. The most direct path to this involves the creation of a process flowchart that maps the intricate dependencies of design variables. A platform and data transmission method must then be selected that satisfy these capability criteria, along with any other design considerations.

The platform selected was MATLAB, with data transmission occurring via data structures. The modular nature of the program design allowed versatility in the use of each module. The flexibility of design variable modification allowed trade studies to be conducted with little modification to existing code, and minimal additional code. The result is a program that can complete preliminary component sizing, as well as synthesize the vehicle design to form a final configuration.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 Objective Overview

The design process requires a structured approach to navigate the complex relations amongst components and variables following a critical path. Each configuration requires many components to be sized, and their effects to be accounted for in the finalized configuration. The design process is generally considered to have four phases: preliminary design level 1 (PD1), preliminary design level 2 (PD2), synthesis, and trade studies. The PD1 design was completed in the previous semester, so the content here focuses on the final two phases. While PD1 design provides an approximation of what the configuration will look like, synthesis converges this estimate to a viable, mission-capable configuration. Trade studies serve to provide data that allows the designer to discern the optimal configuration of the trade study subject to meet performance design criteria. The objective of this work is to document the process of synthesis and conducting trade studies. The unique contribution discussed here is the work of program planning and structure to accomplish these ends.

1.2 Tools Used

The first step of implementing the synthesis design process was to decide on a platform from which to create the program. Several platforms were explored, including Microsoft Excel, Simulink and MATLAB. The initial PD1 design calculations were conducted in Excel because it provided an immediate visualization of the results to ensure that the component models being used were validated. Upon beginning the work on synthesis, however, it was discovered that Excel did not have the iterative capabilities required to process synthesis. This revelation necessitated a change in the platform. The next tool investigated was Simulink, a companion software product to MATLAB. This software is a GUI-based signal processing tool, one that it was hoped would be able to handle the myriad design dependencies inherent in the process. Unfortunately, since Simulink was designed for time-dependent simulations, it did not have the functionality that was needed. Although it was not ultimately utilized in the final program, the work done in Simulink was pivotal in the creation of the resultant program. Because Simulink is a GUI-based signal processing software, the model created was effectively a highly detailed flowchart. This meant that the Simulink model could be used as a map of sorts, depicting the exact data dependencies that would be required in the final program. A picture of the Simulink model described above is included below.

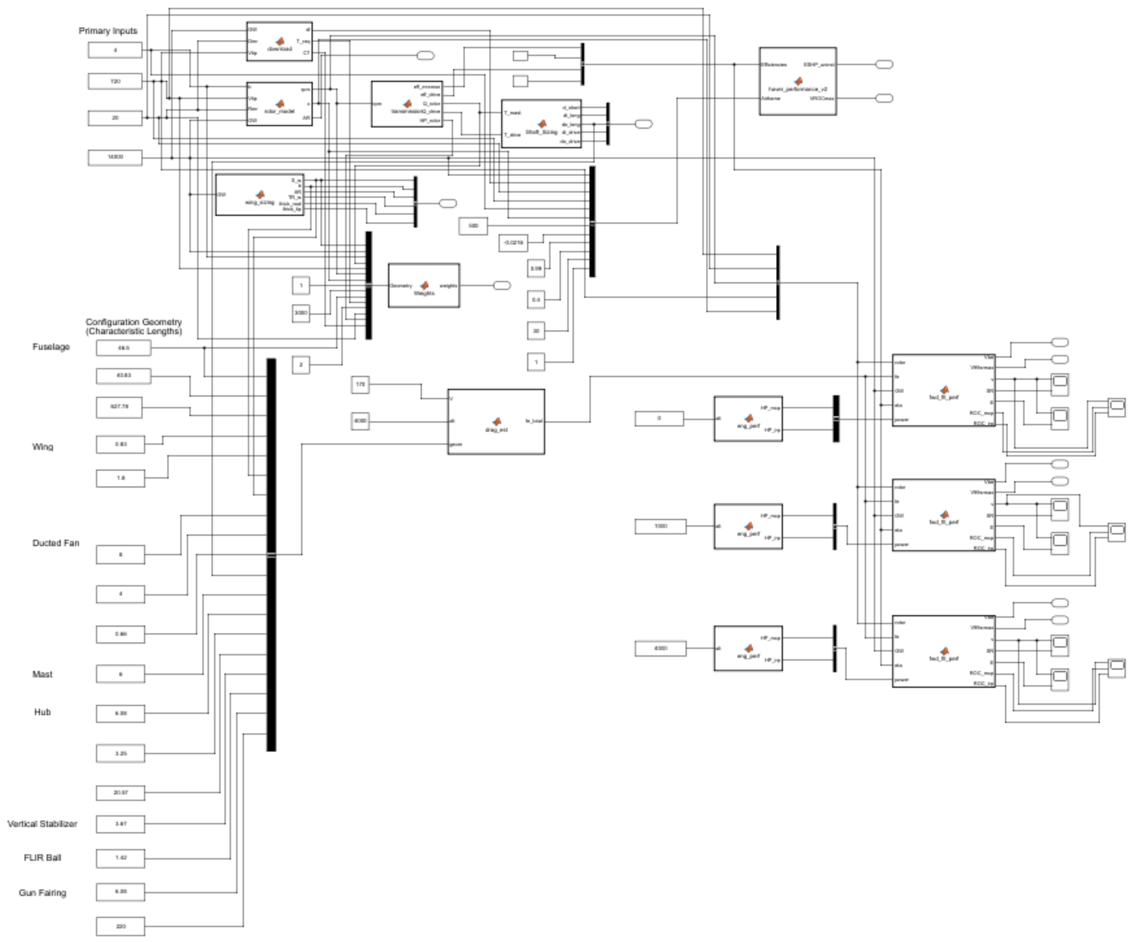


Figure 1.1: Simulink Model for PD1 Design

The next, and final tool that was used is MATLAB. This is the core product for which Simulink is a companion. MATLAB is a graphical programming language created by the MathWorks company. Its name stands for matrix laboratory, indicating its propensity for handling large data in an intuitive and user-friendly manner. Transitioning to a programming language allowed the individual sizing and calculation modules to be implemented as functions, to be called in the larger program. Functions in programming are small scripts of calculations or logic that take in some inputs, and create some outputs, through the script contained within them. This is a compact alternative to listing the entire code in one massive script, and it eliminates the large amounts of redundancy that would likely otherwise occur. This input/output format also allows the functions to be designed with versatility in mind, such that the same function can be used in multiple different places and contexts.

CHAPTER 2

METHODOLOGY

2.1 Data Transmission

The next consideration when designing a structured programming architecture is the method of data transmission. There were several options available in MATLAB, from the characteristic namesake matrices and arrays to data structures. The latter was chosen because it provided a compact way to store data about an object or component in a single place with unique and clearly identifiable names.

2.2 Program Structure

Once the platform and data transmission methods were chosen, then came the task of implementation. The step from preliminary design to synthesis entails the inclusion of mission analysis and a loop to converge the estimates for vehicle weight between the initial estimate and the value obtained from mission analysis using the components sized from the initial estimate. A simplified flowchart was developed to include this loop onto preliminary design data flow that was demonstrated in the Simulink model.

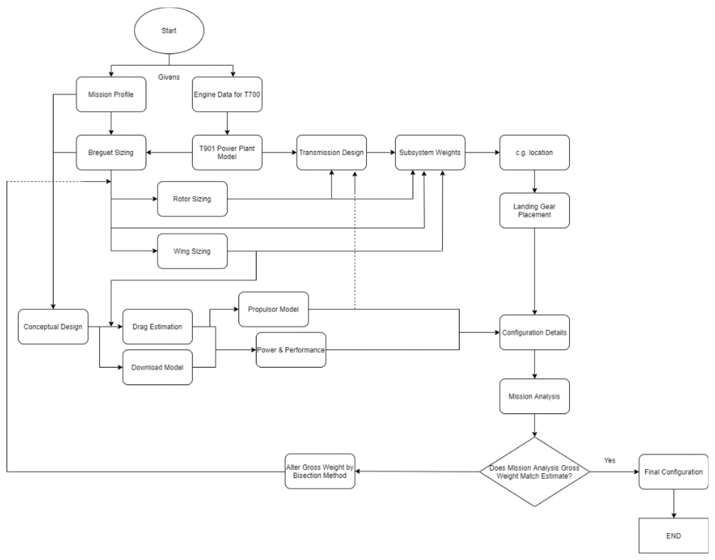


Figure 2.1: Synthesis Process Flowchart

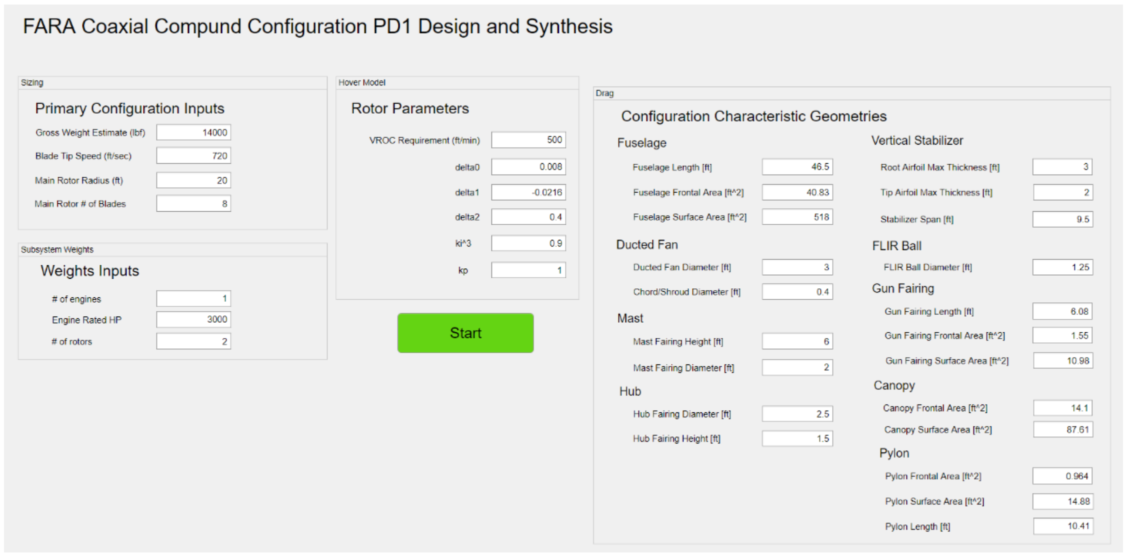
Each of the modules listed on this flowchart was created by a member of the design team, and all members had at least one module that was specifically created by them. As previously mentioned, these were designed as functions to be used throughout. Before mission analysis, the calculations are largely linear. Mission analysis, however, is where the functions begin to be used in ways not intuitively conveyed by their original design. In mission analysis, the functions are used such that not all the possible outputs are required for each use, and some outputs are required that are not required elsewhere. The function format of these modules allowed extra outputs to be added with minimal effort.

A final aspect of program structure is its implementation. The best programs involving many modules utilize bottom-up implementation, and top-down execution. This scheme makes troubleshooting easier, because it adds an extra path identifier for traceability to calculation results that are incorrect or compiler errors that occur.

The lynch pin for synthesis convergence to occur is mission analysis. This portion of the program entails utilizing the vehicle and engine performance models to determine fuel usage to complete the assigned mission. This is combined with the prescribed payload and the empty weights estimate, which is based on the gross weight estimate provided at the beginning of the program to provide a value for the configuration gross weight. The gross weight estimate to size all the components per the flowcharts above should be the same as the result from mission analysis, given the components sized from the estimate. If these are not the same, the program must contain a piece of convergence logic that will cause these two numbers to end up being the same. The method utilized in this program is a bisection method of sorts. It takes the average of the initial estimate and the mission analysis results and assigns this to the new estimate to repeat the program execution, until the numbers converge within a specified threshold. The result is a fully defined configuration, with components sized, performance evaluated, and weight determined.

2.3 User Inputs

In the interest of making this program versatile, the addition of user inputs was considered. MATLAB has menu pop-ups and user input functions built in that allow the program writer to ask the users for input when the program is run. Because of the sheer number of inputs required to effectively size and design a rotorcraft configuration, this option was considered non-viable. Some research unveiled another companion tool for MATLAB called MATLAB App Designer. This tool is a platform for graphically creating apps that can receive user inputs and display results on a dashboard for a user interface to be used with a larger program, or as a standalone. In this context, it was used as a method to create a graphical user interface (GUI) for the user to modify some of the set default input values, and then tell the program to execute. The values were then stored for use by the synthesis program. A picture of the GUI in its startup configuration is shown here.



FARA Coaxial Compund Configuration PD1 Design and Synthesis

Sizing

Primary Configuration Inputs

Gross Weight Estimate (lbf)	14000
Blade Tip Speed (ft/sec)	720
Main Rotor Radius (ft)	20
Main Rotor # of Blades	8

Subsystem Weights

Weights Inputs

# of engines	1
Engine Rated HP	3000
# of rotors	2

Hover Model

Rotor Parameters

VROC Requirement (ft/min)	500
delta0	0.008
delta1	-0.0216
delta2	0.4
ki^3	0.9
kp	1

Drag

Configuration Characteristic Geometries

Fuselage

Fuselage Length [ft]	46.5
Fuselage Frontal Area [ft^2]	40.83
Fuselage Surface Area [ft^2]	518

Ducted Fan

Ducted Fan Diameter [ft]	3
Chord/Shroud Diameter [ft]	0.4

Mast

Mast Fairing Height [ft]	6
Mast Fairing Diameter [ft]	2

Hub

Hub Fairing Diameter [ft]	2.5
Hub Fairing Height [ft]	1.5

Vertical Stabilizer

Root Airfoil Max Thickness [ft]	3
Tip Airfoil Max Thickness [ft]	2
Stabilizer Span [ft]	9.5

FLIR Ball

FLIR Ball Diameter [ft]	1.25

Gun Fairing

Gun Fairing Length [ft]	6.08
Gun Fairing Frontal Area [ft^2]	1.55
Gun Fairing Surface Area [ft^2]	10.98

Canopy

Canopy Frontal Area [ft^2]	14.1
Canopy Surface Area [ft^2]	87.61

Pylon

Pylon Frontal Area [ft^2]	0.964
Pylon Surface Area [ft^2]	14.88
Pylon Length [ft]	10.41

Figure 2.2: GUI User Input Screen

2.4 Trade Studies

Once a convergence is achieved, the final step in the preliminary design process is conducting trade studies to optimize the configuration. This involves varying some set of input parameters and observing the result on the desired design metric. This is most commonly done in the form of carpet plots. These are multi-variable plots that serve to display trends that exist in a system for variances in a set of variables. There are a couple types of carpet plots, including 4-variable carpet plots and 3-variable carpet plots (also known as “cheater plots”). The latter of these is what is used here. This type of plot displays the results on an output variable of two input carpet variables. The abscissa axis is non-dimensional, often referred to as the “cheater axis.” Because of the structural nature of the program, the trade studies could be conducted by simply modifying some of the front-end inputs.

CHAPTER 3

DISCUSSION

As mentioned previously, the results of this program were the final configuration, synthesized according to mission analysis. Some additional results were the trade studies. Beginning with the trade studies, the results are shown below.

3.1 Trade Studies

Once the synthesis was completed, trade studies could be done to improve certain characteristics of the configuration. A couple of examples are a wing trade study conducted to minimize download from a wing in hover while maintaining the lift offload it provides in forward flight. Download is a measure of the vertical drag on a fuselage and wing in hover or vertical climb. The geometry of the wing is varied in terms of its aspect ratio and taper ratio. The aspect ratio of the wing is the ratio of the wingspan to its planform area, and the taper ratio is the ratio of the wing chord at the root to the chord at the wing tip. Varying these parameters over reasonable ranges allows a download estimate to be predicted for each of the corresponding geometry configurations. One of the wing trade study carpet plots is shown below.

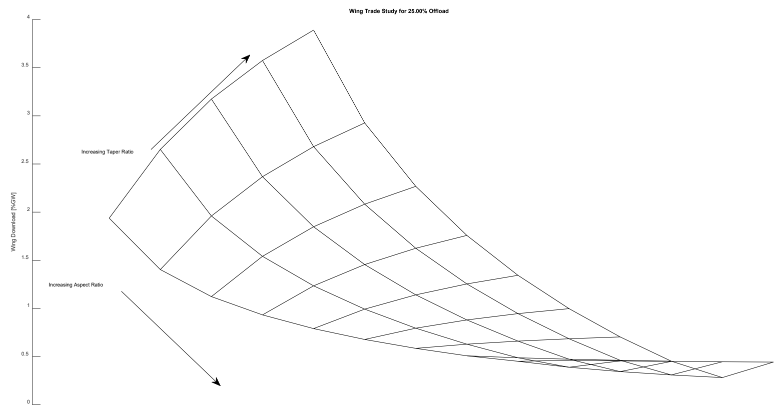


Figure 3.1: Wing Trade Study Example Plot

As described above, the desired result is the selection of a wing geometry such that the download is minimized. It can be obtained from inspection of the figure that there is one specific combination of the aspect ratio and taper ratio that produces a minimum download. This is the purpose of trade studies.

The other trade study conducted was a ducted fan trade study. The ducted fan is the propulsive offload device for the configuration. A ducted fan is a propeller contained in a shroud that increases its propulsive potential. The theory behind the ducted fan allows the ducted fan to be sized in conjunction with propeller theory. The ducted fan is sized to achieve the thrust necessary to offload the drag on the configuration. Given this constraint, a trade study can be conducted to size the ducted fan for a minimum weight configuration. The two carpet variables in this trade study were the ratio of the shroud chord to diameter ratio and the number of propeller blades. A sample from this trade study is shown here, where the geometry of a common propeller used in this application is a constant.

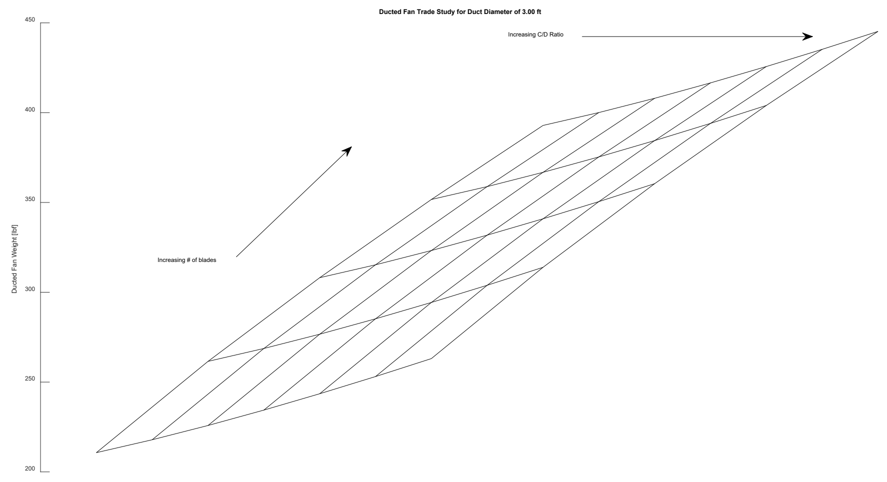


Figure 3.2: Ducted Fan Trade Study Carpet Plot

3.2 Synthesis Results

The synthesis process described entails resizing certain components of the rotorcraft, to ensure that the converged configuration is reflective of the final configuration gross weight. These components include the wing, rotor, and ducted fan. These resized components in turn contribute towards the convergence. Each of these components has an effect on the performance of the vehicle as well. This means that mission analysis will return different results each time because performance is the crux of mission analysis.

The structure of the program, utilizing data structures, meant that it was easy to see the values of the important parameters for diagnostic and troubleshooting purposes. This is a feature that was used numerous times in the development of the program, and is a testament to the forward-thinking program design that was implemented. The data structures also compartmentalized the configuration such that each team member working on a specific aspect of the configuration could have easy access to all the information about that aspect from a simple command window entry for a variable call. The result was a converged, synthesized rotorcraft configuration that could be moved forward in the design and development phase.

CHAPTER 4

CONCLUSION

A structured approach to aircraft synthesis programming is necessary to navigate the complex relation amongst design variables and analytical models. A robust program will also keep trade studies in mind, and consider data flow for troubleshooting purposes. The program described above achieves all these aspects. The use of data structures suits these objectives best, providing compact and traceable variable storage, and allowing for easy modifications in trade studies.

A few items learned through the execution of this project are as follows: the essential nature of data flow chart creation, the importance of data transmission methodology, and the nuances of developing these features while working on a team. The data flow chart vastly assisted in the process of planning the program structure. It provided a clear depiction of the relations to be set up in the final program. Data transmission methodology was a big question early in the program design process, and several data types were tried. Data structures provided a compact, efficient, and traceable solution. Lastly, working on a team for this project has highlighted the need for program developers and project leads in general to take feedback from the team and weave it into the final product.

Overall, this project has been a fulfilling foray into the world of engineering design, as well as an experience in leading an engineering team. The result of this project was an automated preliminary sizing and synthesis code that was hundreds of lines long, yet easy to navigate and use.

APPENDIX A

SYNTHESIS PROCESS PSEUDO CODE

```
n = 1;
GW_initial = value;
GW_estimate = 0;

For i=1:2
    if i == 1
        j = 20; % number of missile segments for recon mission profile
        Time(i,:) = vector; % vector of mission segment times for recon
        Alt(i,:) = vector; % vector of mission segment times for recon
    else
        j = 16; % number of missile segments for attack mission profile
        Time(i,:) = vector; % vector of mission segment times for attack
        Alt(i,:) = vector; % vector of mission segment times for attack
        GW_estimate = 0; % reset estimate for attack mission calculations
    end

    GW = GW_initial;

    while abs(GW-GW_estimate)/GW > 0.01
        for k = 1:j
            if k == 1
                % run all scripts constant across the mission
                GW = GW – sum(FW(i,1:k));
                FW(i,k) = value;
            elseif k == x || y || z
                % run functions for hover phases
                FW(i,k) = value;
            elseif k == x || y || z
                GW_temp = GW;
                for 1:Alt(i,k)/10
                    % run functions for climb phases
                    FW_delta = performance(alt,ROC,GW_temp)
                    GW_temp = GW_temp – FW_delta;
                End
                FW(i,k) = GW – GW_temp;
            else
                % run functions for cruise phases
                FW(i,k) = value;
            end
        end
        FW_total = sum(FW(i,:));
        GW_estimate = EW+PL+FW;
        GW = (GW+GW_estimate)/2;
    end
end
```

APPENDIX B

PRIMARY SYNTHESIS SCRIPT

```
%{
Josiah Everhart and Lorenzo Novoa
Objective: Accomplish PD1 Analysis and Synthesis for a Coaxial Compound
Helicopter, applying the results of trade studies
Helicopter Design for the FARA Program
Models Used:
Download: 14 degree linear twist
Hover Induced Velocity: Glauert Ideal Twist Model
Drag: Component Build-Up
Weights: 15-input military aircraft subsystem weights calculator
%}

run GUI.m
fprintf('Fixed-Geometry Synthesis\n')
set(0,'DefaultFigureWindowStyle','docked')
tic
run Tech_Factors.m
Input.techCdo = tech.MRCdo;
%% Begin PD1 Calculations and Synthesis
PL = [852,1284];%Payload weight, including two pilots (500 lbf. total) and
ordnance (784 lbf.)
mission_str = {'Recon','Attack'};

for i = 1:2
    if i == 1
        Time = [5, 1, NaN, 239/(170/60), NaN, 5, 20, 4, 5, 3, 5,...
            2, 20, 5, NaN, NaN, NaN, 1, 5, 30];
        Time = Time/60; % Change time from minutes to hours
        Alt = 1000*[0 0 4 4 1 1 1 0 0 0 0 1 1 1 4 4 0 0 0 1];
        Alt = Alt+4000; % Adjust altitude to account for 4K 95 conditions
        fprintf('Recon Mission Analysis\n')
    else
        Time = [5, 1, NaN, NaN, NaN, 0, 20, 0, 5, 3, 5, 0, 20, 0, NaN, NaN,...
            NaN, 1, 5, 30];
        Time = Time/60; % Change time from minutes to hours
        Alt = 1000*[0 0 2 2 0 0 0 0 0 0 0 0 0 0 2 2 0 0 0 1];
        Alt = Alt+4000; % Adjust altitude to account for 4K 95 conditions
        fprintf('Attack Mission Analysis\n')
    end

    m = 20; %  of mission segments
    GW_init=Input.GW;
    GW_estimate = 13900;

    while abs(GW_init-GW_estimate)>1

        GW_init = (GW_init+GW_estimate)/2;
        Input.GW = GW_init;

        %% Resize Based on GW
        %Constant for all mission segments

        %Wing
        Input.offload = 0.4;
        wing=wing_sizing_final(Input);

        %Rotor
        rotor=rotor_model_final(Input,wing);

        %Wing
        geom.wtr=wing.thick_root;%Wing Root Airfoil Thickness [ft]
        geom.wtt=wing.thick_tip;%Wing Tip Airfoil Thickness [ft]
        geom.wcr=wing.cr;%Wing Root Airfoil Chord [ft]
        geom.wct=wing.ct;%Wing Tip Airfoil Chord [ft]
        geom.wTR=wing.TR;%Wing Taper Ratio []
        geom.wspan=wing.b;%Wing Span [ft]
        geom.wmgc=wing.mgc;%Wing mgc [ft]
        geom.Sw=wing.Sw;%Wing Planform Area [ft^2]

        %Ducted Fan Sizing
        [ducted_fan] = Ducted_Fan_Sizing_v2(geom);
        weightin.df = ducted_fan.weight;

        %Download
        [TotDow,WingDow,FusDow,weightin.T]=download_v3(Input,rotor,wing);

        weightin.tcmax=rotor.tcmax;
        weightin.Sw=wing.Sw;
        weightin.GW=Input.GW;
        weightin.blades=Input.blades;
        weightin.mission = i;

        %Transmission
        [eta,Q]=transmission_v2(rotor,ducted_fan);

        weightin.Q_rotor=Q.rotor;
        weightin.vtip=Input.Vtip;
        weightin.rpm=rotor.rpm_rotor;
        weightin.c=rotor.c;
        weightin.lfuse=geom.Lfuse;
        weightin.Rmr=Input.Rmr;
        weightin.HP=Q.HP_rotor;

        weightin.fuel = 0;
        %Subsystem Weights
        [weights]=Weights_v5(weightin,tech);%c.g. calculator requires CAD model
subsystem positions

        %Mast Sizing
        [shaft]=Shaft_Sizing_v3(Q);

        %% Mission Analysis
        for k = 1:m
            %% SU/WU
            if k == 1   % SU/WU
                [~,eng_corr] = eng_perf_v2(Alt(k),0);
                [sfc,~] = eng_perf_v2(Alt(k),eng_corr.IDL);
                Output.FW = sfc*eng_corr.IDL*Time(k);
                Input.GW = Input.GW-Output.FW;
            %% HOGE
            elseif k == 2 || k == 10 || k == 18  % HOGE at 0 ft
                [ESHP_uninst_hover,~] =...
                    hover_performance_v3(Input,eta,rotorparam,rotor,TotDow);
                [sfc,eng_corr] = eng_perf_v2(Alt(k),ESHP_uninst_hover);
                if k == 2
                    Output.FW(k) = sfc*eng_corr.IRP*Time(k);
                else
                    Output.FW(k) = sfc*ESHP_uninst_hover*Time(k);
                end
                Input.GW = Input.GW-Output.FW(k);
            %% Climb
            elseif k == 3 || k == 12 || k == 15   % Climb
                delta_h = Alt(k)-Alt(k-1);
                Alt_increment = 10;
                Time_total = 0;
                Input_temp = Input;
                for l = 1:delta_h/Alt_increment
                    Alt_temp = Alt(k)+l*Alt_increment;
                    [~,~,~,~,~,~,ROC_mcp,~,HP_mcp,~]=...
                    fwd_flt_perf_v3(Input_temp,rotor,wing,eta,Alt_temp,geom);
                    [sfc,eng_corr] = eng_perf_v2(Alt(k),HP_mcp);
                    maxROC = max(ROC_mcp);
                    Time_temp = Alt_increment/maxROC/60;
                    FW_temp = sfc*HP_mcp*Time_temp;
                    Time_total = Time_total+Time_temp;
                    Input_temp.GW = Input_temp.GW-FW_temp;
                    clear element
                end
                Time(k) = Time_total;
                Output.FW(k) = Input.GW-Input_temp.GW;
                Input.GW = Input.GW-Output.FW(k);
            %% Cruise
            elseif k == 4 || k == 16   % Cruise
                [ESHP_uninst,~,V99SRmax,v,~,~,~,~,~,~]=...
                    fwd_flt_perf_v3(Input,rotor,wing,eta,Alt(k),geom);
                    Time(k) = 239/170;
                if k == 4
                    element = find(v >= 169,1);
                else
                    element = find(v == V99SRmax);
                end
                [sfc,~] = eng_perf_v2(Alt(k),ESHP_uninst(element));
                Output.FW(k) = sfc*ESHP_uninst(element)*Time(k);
                Input.GW = Input.GW-Output.FW(k);
                clear element
            %% Descent
            elseif k == 5 || k == 8 || k == 17   % Descent       Note: Ask
Dr.Smith about calculating descent in forward flight
                delta_h = Alt(k-1)-Alt(k);
                Alt_increment = 10;
                Time_total = 0;
                Input_temp = Input;
                for l = 1:delta_h/Alt_increment
                    Alt_temp = Alt(k)-l*Alt_increment;
                    rotorparam_temp = rotorparam;
                    rotorparam_temp.VROC = -250;
                    [ESHP_uninst_hover,~] =
hover_performance_v3(Input_temp,eta,rotorparam_temp,rotor,TotDow);
                    [sfc,eng_corr] = eng_perf_v2(Alt(k),ESHP_uninst_hover);
                    Time_temp = -Alt_increment/rotorparam_temp.VROC/60;
                    FW_temp = sfc*ESHP_uninst_hover*Time_temp;
                    Time_total = Time_total+Time_temp;
                    Input_temp.GW = Input_temp.GW-FW_temp;
                end
                Output.FW(k) = Input.GW-Input_temp.GW;
                Time(k) = Time_total;
                Input.GW = Input.GW-Output.FW(k);
                clear Input_temp
            %% Dash
            elseif k == 6 || k == 14 % Dash
                [ESHP_uninst,~,~,v,~,~,~,~,~,~]=...
                    fwd_flt_perf_v3(Input,rotor,wing,eta,Alt(k),geom);
                element = find(v == 200);
                [sfc,eng_corr] = eng_perf_v2(Alt(k),ESHP_uninst(element));
                Output.FW(k) = sfc*ESHP_uninst(element)*Time(k);
                Input.GW = Input.GW-Output.FW(k);
                clear element
           %% Loiter
            elseif k == 7 || k == 13 || k == 20 % Loiter
```

```
                [ESHP_uninst,Vbe,~,~,~,~,~,~,~,~,~]=...
                    fwd_flt_perf_v3(Input,rotor,wing,eta,Alt(k),geom);
                element = find(v == Vbe);
                [sfc,eng_corr] = eng_perf_v2(Alt(k),ESHP_uninst(element));
                Output.FW(k) = sfc*ESHP_uninst(element)*Time(k); % Vector does
not meet Vbe value
                Input.GW = Input.GW-Output.FW(k);
           %% NOE
            elseif k == 9 || k == 11 % NOE
                [ESHP_uninst,~,~,v,~,~,~,~,~,~,~]=...
                    fwd_flt_perf_v3(Input,rotor,wing,eta,Alt(k),geom);
                element = find(v >= 40,1);
                [sfc,eng_corr] = eng_perf_v2(Alt(k),ESHP_uninst(element));
                Output.FW(k) = sfc*ESHP_uninst(element)*Time(k);
                Input.GW = Input.GW-Output.FW(k);
                clear element
            end
        end

GW_estimate = weights.EW+PL(i)+sum(Output.FW);
    end
    mission_time = sum(Time);%hr
    weightin.fuel = sum(Output.FW);
    %Subsystem Weights
        [weights]=Weights_v5(weightin,tech);%c.g. calculator requires CAD model
subsystem positions
        %% C.G. Location Plots
        ang14_1 = [-50 0 50;-50*tand(14)+21.8 0+21.8 -50*tand(14)+21.8];
        ang14_2 = [-50*tand(14) 0 50*tand(14);-50+21.8 0+21.8 -50+21.8];
        if i==1
            figure(1)
        else
            figure(8)
        end
        plot(ang14_1(1,:),ang14_1(2,:))
        hold on
        plot(ang14_2(1,:),ang14_2(2,:))
        plot([weights.cg_x_EFW weights.cg_x_GW],[weights.cg_y_EFW
weights.cg_y_GW])
        plot(weights.cg_x_EW,weights.cg_y_EW,'x')
        plot(weights.cg_x_WCW,weights.cg_y_WCW,'x')
        legend('14^{o} From Rotor','14^{o} From Mast','Gross Weight to Empty
Fuel Weight','Empty Weight','Windchester','location','southeast')

        Input.GW = GW_estimate;
        format short
        fprintf('The Synthesis Gross Weight for the %s Mission: %.2f
lbf\n',mission_str{i},GW_estimate)
    %% Other Final Values
        %Drag
        [fe] = DragModel_v4(8000,170,geom);

        %Landing Gear Placement
        [gearlocation] = Landing_Gear_v4(weights);

        %Hover
        [ESHP_uninst_hover,VROCmax] =
hover_performance_v3(Input,eta,rotorparam,rotor,TotDow);

        %HOGE Ceiling
        [HOGE] = HOGE_v1(Input,eta,rotorparam,rotor,TotDow);

        %HIGE Ceiling
        [HIGE] = HIGE_v1(Input,eta,rotorparam,rotor,TotDow);

        %Forward Flight Performance
        %4000 ft above SL (at T/O level)

[ESHP_uninst,Vbe,V99srmax,v,SR,E,ROC_mcp,ROC_irp,HP_mcp,HP_irp]=fwd_flt_perf_v3
(Input,rotor,wing,eta,4000,geom);

        %5000 ft above SL (at mission loiter, 1000 ft above T/O level)

[ESHP_uninst_ltr,Vbe_ltr,V99srmax_ltr,v_ltr,SR_ltr,E_ltr,ROC_mcp_ltr,ROC_irp_lt
r,HP_mcp_ltr,HP_irp_ltr]=fwd_flt_perf_v3(Input,rotor,wing,eta,5000,geom);

        %8000 ft above SL (at mission cruise, 4000 ft above T/O level)

[ESHP_uninst_cr,Vbe_cr,V99srmax_cr,v_cr,SR_cr,E_cr,ROC_mcp_cr,ROC_irp_cr,HP_mcp
_cr,HP_irp_cr]=fwd_flt_perf_v3(Input,rotor,wing,eta,8000,geom);

        figure(2+(i-
1)*3),plot(v,ESHP_uninst,v,HP_mcp*ones(length(v),1),v,HP_irp*ones(length(v),1),
v,ESHP_uninst_hover*ones(length(v),1),'LineWidth',2),...
            ylabel('ESHPuninst [HP]','FontSize',18),xlabel('Speed
[ktas]','FontSize',18),title([mission_str{i},' Mission Speed Power Polar at T/O
4000 ft above SL'],'FontSize',18),xlim([30 220]),ylim([0 3000]),...
            legend('ESHP uninst fwd','HP mcp','HP irp','ESHP uninst
hover','location','southeast','FontSize',16)
            set(gca,'FontSize',14)
        figure(3+(i-
1)*3),plot(v_ltr,ESHP_uninst_ltr,v,HP_mcp_ltr*ones(length(v),1),v,HP_irp_ltr*on
es(length(v),1),'LineWidth',2),...
            ylabel('ESHPuninst [HP]','FontSize',18),xlabel('Speed
[ktas]','FontSize',18),title([mission_str{i},' Mission Speed Power Polar at
Loiter 5000 ft above SL'],'FontSize',18),xlim([30 220]),ylim([0 3000]),...
            legend('ESHP uninst fwd','HP mcp','HP
irp','location','southeast','FontSize',16)
            set(gca,'FontSize',14)
        figure(4+(i-
1)*3),plot(v_cr,ESHP_uninst_cr,v,HP_mcp_cr*ones(length(v),1),v,HP_irp_cr*ones(l
ength(v),1),'LineWidth',2),...
            ylabel('ESHPuninst [HP]','FontSize',18),xlabel('Speed
[ktas]','FontSize',18),title([mission_str{i},' Mission Speed Power Polar at
Cruise 8000 ft above SL'],'FontSize',18),xlim([30 220]),ylim([0 3000]),...
            legend('ESHP uninst fwd','HP mcp','HP
irp','location','southeast','FontSize',16)
            set(gca,'FontSize',14)
        if i==1
            save recon_configuration.mat
        else
            save attack_configuration.mat
        end
end
toc
fprintf('\n')
%%%%End Code%%%%
```

REFERENCES

Metzger, P. W., *Managing a programming project: people and processes*, Englewood Cliffs, NJ: Prentice Hall, 1973.

Stepniewski, W. Z., and Keys, C. N., *Rotary-wing aerodynamics Vol. 2*, Washington, D.C.: National Aeronautics and Space Administration, Scientific and Technical Information Office, 1979.

BIOGRAPHICAL INFORMATION

Josiah Everhart is an aerospace engineering senior graduating with a Bachelor of Science degree in aerospace engineering. He will then go on to obtain a Master of Engineering degree in aerospace engineering. Josiah has conducted university-funded research on active space debris removal technologies. He designed a space vehicle capable of removing space debris from lower Earth orbit. Additionally, he has participated in projects creating dynamic simulations of aerospace systems.

A few of his topical interests include robotics; guidance, navigation, and control (GNC); and aircraft conceptual design. He is a member of the Tau Beta Pi engineering honors society. Josiah acted as the project team lead for his senior design capstone project team. He led the team to develop a helicopter configuration with the best performance of any configuration of the teams participating in the class.

He held an internship with Lockheed Martin Missiles and Fire Control in the Summer of 2019. He currently occupies a contractor role since the Fall of 2019, and he begins a full-time position with Lockheed Martin upon his graduation. Josiah would like to work in systems engineering with a focus in performance, GNC, simulation, or conceptual design. The master's program in engineering will equip him to handle some of the advanced topics sure to be required in such positions. Josiah hopes to take his education, and his career as far as possible by continuing the learning process beyond the undergraduate program, all with the same dedication to excellence that was fostered by his experience in the Honors College.